PROCEEDINGS OF THE 11TH INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL CONFERENCE

EDITED BY
J. STEPHEN DOWNIE AND
REMCO C. VELTKAMP

August 9-13, 2010
Utrecht, Netherlands

ismir 2010 UTRECHT
11th International Society for Music Information Retrieval Conference

ISMIR 2010

Proceedings of the 11th International Society for Music Information Retrieval Conference

August 9-13, 2010
Utrecht, Netherlands

Edited by J. Stephen Downie and Remco C. Veltkamp

The 11th International Society for Music Information Retrieval Conference is organised by:

Universiteit Utrecht

Hogeschool voor de Kunsten Utrecht

PHILIPS

Edited by J. Stephen Downie and Remco C. Veltkamp

Logo and graphic design by Gijs Bekenkamp (945-ontwerp, Netherlands)

Printed by Hollandridderkerk

Published by the International Society for Music Information Retrieval
Website: http://www.ismir.net/
ISBN 978-90-393-53813

ISMIR 2010 gratefully acknowledges the support from the following sponsors:

Gemeente Utrecht

provincie :: Utrecht

ISMIR 2010 gratefully acknowledges the support from the following sponsors:

Conference Committee

General Chair
Frans Wiering

Program Chairs
J. Stephen Downie
Remco C. Veltkamp

Tutorials Chair
Steffen Pauws

Late Breaking and Demo Chairs
François Pachet
Kjell Lemström

Finance Chair
Wilke Schram

Industrial Relations, Exhibits and Sponsoring Chair
Marco Spruit

Publicity Chair
W. Bas de Haas

Local Organising Committee
Peter van Kranenburg
Geraldine Leebeek
Steven van de Par
Sandor Spruit
Hans Timmermans

USMIR Summer School Committee

Micheline Lesaffre
Chris Müller
Geraldine Leebeek
Frans Wiering

Program Committee

David Bainbridge
Juan Pablo Bello
Michael Casey
Oscar Celma
Elaine Chew
Sally Jo Cunningham
Ichiro Fujinaga
Emilia Gómez
Masataka Goto
Keiji Hirata
Youngmoo Kim
Carol Krumhansl
Olivier Lartillot
Kjell Lemstrom
Meinard Müller
François Pachet
Craig Sapp
Markus Schedl
Joan Serrà
Malcolm Slaney
Godfried Toussaint
George Tzanetakis
Anja Volk
Geraint Wiggins

Graphic design

Gijs Bekenkamp

Reviewers

Samer Abdallah
Teppo Ahonen
Christina Anagnostopoulou
Amelie Anglade
Josep Lluis Arcos
Gerard Assayag
Jean-Julien Aucouturier
Wolfgang Auhagen
David Bainbridge
Gabriele Barbieri
Luke Barrington
Sumit Basu
Stephan Bauman
Mert Bay
Juan Pablo Bello
Thierry Bertin-Mahieux
Abhijit Bhattacharjee
Elvira Brattico
John Ashley Burgoyne
Juan Jose Burred
Donald Byrd
Emilios Cambouropoulos
Pedro Cano
Jaime S. Cardoso
Mark Cartwright
Norman Casagrande
Michael Casey
Oscar Celma
Elaine Chew
Parag Chordia
Mads Græsbøll Christensen
Ching-Hua Chuan
Michael Clausen
Annabel Cohen
Darrell Conklin
Arshia Cont
Tim Crawford
Sally Jo Cunningham
Roger Dannenberg
Laurent Daudet
François Deliège
Arnaud Dessein
Johanna Devaney
Christian Dittmar
Simon Dixon
Zhiyao Duan
Jon Dunn
Douglas Eck
Tuomas Eerola
Jana Eggink
Andreas Ehmann
Dan Ellis
Valentin Emiya
Paulo Esquef
Slim Essid
Sebastian Ewert
Morwaread Farbood
George Fazekas
Ben Fields
Arthur Flexer
Sebastian Flossmann
Alexandre François
Judy Franklin
Duncan Freeman
Anders Friberg
Hiromasa Fujihara
Ichiro Fujinaga
Jörg Garbers
Martin Gasser
Emilia Gómez
Francisco Gómez
Michael Good
Masataka Goto
Fabien Gouyon
Maarten Grachten
Stephen Green
Graham Grindlay
Enric Guaus
Jin Ha Lee
W. Bas de Haas
Jinyu Han
Christopher Harte
Toni Heittola
Stephen Henry
Perfecto Herrera
Keiji Hirata
Jason Hockman
Xiao Hu
Jose Manuel Iñesta
Charlie Inskip
Hideki Isozaki
Akinori Ito
Ozgur Izmirli
Kurt Jacobson
Roger Jang
Tristan Jehan
Kristoffer Jensen
Bram de Jong
Hirokazu Kameoka
Kunio Kashino
Haruhiro Katayose
Robert Keller
Youngmoo E. Kim
Tetsuro Kitahara
Anssi Klapuri
Peter Knees
Noam Koenigstein
Peter van Kranenburg
Pedro Kroger
Carol L. Krumhansl
Frank Kurth
Mathieu Lagrange
Kamlesh Lakshminarayanan
Paul Lamere
Gert R.G. Lanckriet
Thibault Langlois
Olivier Lartillot
Kai Lassfolk
Cyril Laurier
Kyogu Lee
Kjell Lemström
Pierre Leveau
David Lewis
Richard Lewis
Cynthia Liem
David Little
Namunu Maddage
Søren Tjagvad Madsen
Veli Makinen
Michael I. Mandel
Arpi Mardirossian
Matija Marolt
Alan Marsden
Luis Gustavo Martins
Matthias Mauch
Brian McFee
Cory McKay
David Meredith

Annamaria Mesaros
Ray Migneco
Dirk Moelants
Meinard Müller
Tomoyasu Nakano
Teresa Nakra
Paolo Nesi
Kia Ng
Bernhard Niedermayer
Izumiya Nisen
Katy Noland
Yasunori Ohishi
Nicola Orio
François Pachet
Rui Pedro Paiva
Hélène Papadopoulos
Bryan Pardo
Jouni Paulus
Steffen Pauws
Marcus Pearce
Geoffroy Peeters
Aggelos Pikrakis
Tim Pohle
Laurent Pugin
Ian Quinn
Zafar Rafii
Yves Raimond
Christopher Raphael
Jeremy Reed
Josh Reiss
Gaël Richard
Jenn Riley
David Rizo
Axel Robel
Bill Rogers
Martin Rohrmeier
Perry Rolland
Florence Rossant
Robert Rowe
Pierre Roy
Andrew Sabin
Shigeki Sagayama
Markus Schedl
Erik Schmidt
Björn Schuller
Jeff Scott
Stefania Serafin
Joan Serrà
William Sethares
Hiroko Shiraiwa-Terasawa
Yu Shiu
Janto Skowronek
Malcolm Slaney
Paris Smaragdis
Christine Smit
Jordan B. L. Smith
Mohamed Sordo
Haruto Takeda
Atte Tenkanen
Dan Tidhar
Petri Toiviainen
Spencer Topel
Douglas Turnbull
Rainer Typke
George Tzanetakis
Remco C. Veltkamp
Emmanuel Vincent
Tuomas Virtanen
Anja Volk
Ye Wang
Claus Weihs
Eugene Weinstein
Ron Weiss
Kris West
Tillman Weyde
Brian Whitman
Gerhard Widmer
Geraint Wiggins
Yi-Hsuan Yang
Kazuyoshi Yoshii
Takuya Yoshioka

Preface

Welcome to the *11th International Society for Music Information Retrieval Conference* (ISMIR 2010). ISMIR 2010 will be convened in Utrecht, Netherlands, 9-13 August 2010 and is jointly organised by Utrecht University, the Utrecht School of the Arts, the Meertens Institute and Philips Research. The organisers have a strong conviction (which we believe is widely shared by the MIR community) that studying the human processing of music is a key issue in innovative MIR research. Therefore, MIR research and applications that model musical cognition and perception, that contribute to the human understanding and experience of music, or that make creative use of MIR research receive particular attention during ISMIR 2010.

ISMIR conferences have a long tradition of high quality interdisciplinary scholarship. We hope that you will find that these proceedings have met the high standards of excellence reached by previous ISMIR Program Committees (PC) and participating Music Information Retrieval (MIR) researchers. Like earlier ISMIRs, the success of ISMIR 2010 is founded upon the hard work of its 24 PC members, the thoughtful adjudications of its 262 reviewers, and the large number (176) of first-rate submissions it received from its vibrant research community.

ISMIR 2010 builds upon lessons learned from earlier ISMIR conferences and thus has carried on many practices from ISMIR's past. ISMIR 2010 submissions were reviewed double-blind to avoid bias. Each accepted paper was allotted six pages of proceedings space. As in the past, there was no quality distinction made concerning the mode of presentation (whether poster or oral). Those papers chosen for oral presentation were selected solely on the grounds of providing ISMIR 2010 participants with a conference programme that best reflected the wide-ranging interests, techniques and findings of ISMIR's multidisciplinary world. Unlike recent ISMIRs, there was no rebuttal phase because of the shortened review timelines caused by ISMIR 2010's earlier-than-usual August meeting date.

New to ISMIR 2010, however, was the creation of a new kind of submission category. This year we introduced the State-of-the-Art Report (STAR) paper. STAR papers are intended to summarize for the community the current research questions, accomplishments, and open problems in one of MIR's many subfields. STAR papers were allotted up to 12 pages to allow STAR authors to comprehensively cover their topics. This year, we received seven excellent STAR submissions. Because of time and space considerations, we were only able to include two STAR papers in the programne on the topics of *Music Emotion Recognition* (Kim, et al.) *and Audio-Based Music Structure Analysis* (Paulus, Mueller and Klapuri). We hope that ISMIR attendees find this new class of paper useful and that STAR papers become part of the ISMIR tradition.

Table 1. ISMIR Publication Statistics 2000-2009

		Presentations		Total	Total	Total	Unique	Pages/	Authors/	U. Authors/
Year	**Location**	**Oral**	**Posters**	**Papers**	**Pages**	**Authors**	**Authors**	**Paper**	**Paper**	**Paper**
2000	Plymouth	19	16	35	155	68	63	4.4	1.9	1.8
2001	Bloomington	25	16	41	222	100	86	5.4	2.4	2.1
2002	Paris	35	22	57	300	129	117	5.3	2.3	2.1
2003	Baltimore	26	24	50	209	132	111	4.2	2.6	2.2
2004	Barcelona	61	44	105	582	252	214	5.5	2.4	2.0
2005	London	57	57	114	697	316	233	6.1	2.8	2.1
2006	Victoria	59	36	95	397	246	198	4.2	2.6	2.1
2007	Vienna	62	65	127	486	361	267	3.8	2.8	2.4
2008	Philadelphia	24	105	105	630	296	253	6.0	2.8	2.4
2009	Kobe	38	85	123	729	375	292	5.9	3.0	2.4
2010	**Utrecht**	**24**	**86**	**110**	**656**	**314**	**263**	**6.0**	**2.9**	**2.4**

As stated before, ISMIR 2010 received 176 papers for review (not including the seven STAR papers). We were very pleased to see that ISMIR retained its international following as the submissions came from 29 different countries. The reviewers generated 696 reviews and meta-reviews. Of these, 108 papers were selected for publication. This yielded an acceptance rate of 61% which is in line with previous ISMIR rates. Twenty-two papers were selected for oral presentation. When combined with the two STAR papers, the total number of oral presentations comes to 24 as shown in Table 1. Eighty-six papers were chosen for poster presentation. Not shown in Table 1 (to keep the data consistent across years) are the six extra posters presented by those giving oral presentations. Also missing from Table 1 are the two informative papers from the 2nd Future of MIR—f(MIR)—workshop held in conjunction with ISMIR 2010 and included in the proceedings. Table 1 statistics do not include the many posters that were presented as part of the 2010 Music Information Retrieval Evaluation eXchange (MIREX) 2010 poster session. Finally, the 30 late-breaking poster/demo abstracts that have been made available via the ISMIR 2010 website are not part of the Table 1 data.

We are very proud to have two distinguished special speakers on the programme. Music psychologist Carol L. Krumhansl, giving the Keynote Lecture, is famous for her research into the perception of tonality, and her work is probably the most widely-used music-psychological source for MIR research. Joris de Man, our Invited Speaker, is the composer of the music for the computer game Killzone 2. He will discuss the merge of music and technology from a seemingly different point of view that may nevertheless have many implications for future MIR research and applications.

ISMIR 2010 is preceded by the one-week Utrecht Summer School in Music Information Retrieval (USMIR), which is jointly organised by the Department of Information and Computing Sciences of Utrecht University and the Institute for Psychoacoustics and Electronic Music of Ghent University (Belgium). This summer school follows the example on an earlier, highly successful event that preceded ISMIR 2004 in Barcelona. Twenty-eight students and 14 specialists from 15 different countries and will participate in this summer school, discussing and doing practical work on topics that range from melody retrieval to Wii-retrieval and from user-based music research method to toolboxes such as PHAT. As in ISMIR 2010, the human processing of music is the starting-point for the meaningful application of technology.

This year's social programme has been designed in such a way that it continues the broad themes of the conference in a more leisurely fashion. It confronts music technology, creativity and performance, and adds a historical and a Dutch dimension to mixture. You will come across street organs, carillons, prepared pianos, human and mechanical performers, and in particular the precursor of that ubiquitous MIR tool, piano roll notation, in its authentic, material form, punched in cardboard.

The Programme Chairs would like to express their deep gratitude to all those who have given so much to make ISMIR 2010 programme and proceedings a success. We thank the members of the Conference Committee, the Programme Committee, the reviewers, and the submitters. Many thanks

go to our sponsors, Gracenote, the City of Utrecht, the Province of Utrecht, the Netherlands Organisation for Scientific Research (NWO), Philips Research, Microsoft Research, Google Research and the Netherlands Research School for Information and Knowledge Systems (SIKS), for making many aspects of the programme possible and in particular for enabling us to fund a number of Student Travel Awards. Extra-special recognition goes to Drs. Bas de Haas, of Utrecht University, for his tireless efforts in making the managerial aspects of the whole conference programming process run as efficiently as possible.

Frans Wiering
General Chair, ISMIR 2010

J. Stephen Downie and Remco Veltkamp
Programme Chairs, ISMIR 2010

Table of Contents

Keynote Presentation

Carol L. Krumhansl, Department of Psychology, Cornell University

Music and Cognition: Links at Many Levels

The talk presents research showing that music and cognition have strong links at many levels. An example of a link at a deep level is the empirical support found for deeply theorized properties of music such as Lerdahl's theory of musical tension. Confirmation of this theory demonstrates that the cognitive representation of musical structure includes hierarchical trees similar to those proposed for language. At a somewhat higher level, sensitivity to statistically frequent patterns in the sounded events enables listeners to abstract a tonal framework for encoding and remembering music and generating expectations. Violations of these expectations contribute to the emotional response to music and produce neural responses in fMRI studies. Thus, statistical learning, found for language and other perceptual domains, extends to music where it has special significance. Finally, research on music recognition suggests a great deal of surface information is encoded in memory. Very short excerpts of popular music can be identified with artist, title, and release date. Even when an excerpt is not identified, emotion and style judgments are consistent. These results point to a long-term memory for music with large capacity and fine detail as well as schematic knowledge of style and emotional content.

A COMPARATIVE EVALUATION OF ALGORITHMS FOR DISCOVERING TRANSLATIONAL PATTERNS IN BAROQUE KEYBOARD WORKS

Tom Collins
The Open University, UK
t.e.collins@open.ac.uk

Jeremy Thurlow
University of Cambridge
jrt26@cam.ac.uk

Robin Laney, Alistair Willis, and Paul H. Garthwaite
The Open University, UK

ABSTRACT

We consider the problem of intra-opus pattern discovery, that is, the task of discovering patterns of a specified type within a piece of music. A music analyst undertook this task for works by Domenico Scarlattti and Johann Sebastian Bach, forming a benchmark of 'target' patterns. The performance of two existing algorithms and one of our own creation, called SIACT, is evaluated by comparison with this benchmark. SIACT out-performs the existing algorithms with regard to recall and, more often than not, precision. It is demonstrated that in all but the most carefully selected excerpts of music, the two existing algorithms can be affected by what is termed the 'problem of isolated membership'. Central to the relative success of SIACT is our intention that it should address this particular problem. The paper contrasts string-based and geometric approaches to pattern discovery, with an introduction to the latter. Suggestions for future work are given.

1. INTRODUCTION

This paper discusses and evaluates algorithms that are intended for the following task: given a piece of music in a semi-symbolic representation, discover so-called translational patterns [14] that occur within the piece. Translational patterns (in the geometric sense) are discussed further in Sec. 1.1. Although they are not the only type of pattern that could matter in music analysis, many music analysts would acknowledge that such a discovery task forms part of the preparation when writing an analytical essay [6]. Even if the final essay pays little or no heed to the discovery of translational patterns, neglecting this preparatory task entirely could result in failing to mention something that is musically very noticeable, or worse, very important. Hence we are motivated by the prospect of automating the discovery task, as it could have interesting implications for music analysts (and music listeners in general), enabling them to engage with pieces in a novel manner. We also consider this task to be an open problem within music information retrieval (MIR), so attempting to improve upon current solutions is another motivating factor. The algorithms are applied to Baroque keyboard pieces by Scarlatti (Sonatas L1 and 10) and—to ensure common ground with existing work [7, 13]—Bach (Preludes BWV849 and 854). Two existing algorithms and one of our own creation are evaluated by comparing their output with a music analyst's (the second author's) independent findings for these same keyboard pieces (Sec. 4).

1.1 Review of existing work

In MIR there do not seem to be clear distinctions between the terms pattern 'discovery' [5, 8, 14, 16], 'extraction' [10, 11, 17], 'identification' [7, 9] , and 'mining' [3], at least in the sense that most of the papers just cited address very similar discovery tasks to that stated at the beginning of Sec. 1. Conklin & Bergeron [5] give the label 'intra-opus' discovery to concentrating on patterns that occur *within* pieces. An alternative is 'inter-opus' discovery, where patterns are discovered *across* many pieces of music [5, 9]. This makes it possible to gauge the typicality of a particular pattern relative to the corpus style. Terms that *are* clearly distinguished in MIR are pattern 'discovery' and 'matching' [4]. Pattern matching is the central process in 'content-based retrieval' [18], where the user *provides* a query and then the algorithm searches a music database for more or less exact instances of the query. The output is ranked by some measure of proximity to the original query. This matching task is quite different from the intra-opus discovery task, where there is neither a query nor a database as such, just a single piece of music, and no obvious way of ranking an algorithm's output. While we have stressed their differences, some authors attempt to address both discovery and matching tasks in the same paper [12, 13], suggesting that representations/algorithms that work well for one task might be adapted and applied fruitfully to the other.

Some attempts at pattern discovery have been made with audio representations of music [15]. However, we, like the majority of work cited in this section, begin with a semi-symbolic representation, such as a MIDI file. Work on semi-symbolic representations can be categorised into string-based [2,3,5,8–11,16,17] and geometric approaches [7, 12–14], and which approach is most appropriate depends on the musical situation. For instance the string-based method is more appropriate for the excerpt in Fig. 1. We propose that the most salient pattern in this short ex-

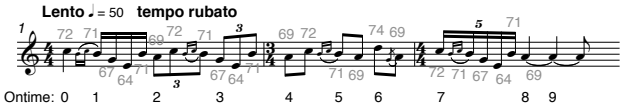


Figure 1. Bars 1-3 of the Introduction from *The Rite of Spring* by Igor Stravinsky, annotated with MIDI note numbers and ontimes in crotchets starting from zero. For clarity, phrasing is omitted and ornaments are not annotated.

cerpt consists of the notes C5, B4, G4, E4, B4, A4, ignoring ornaments for simplicity. The simplest way to discover the three occurrences of this pattern is to represent the excerpt as a string of MIDI note numbers and then to use an algorithm for pattern discovery in strings. The string 72, 71, 67, 64, 71, 69, ought to be discovered, and the user relates this back to the notes C5, B4, G4, E4, B4, A4. The geometric method is not appropriate here because each occurrence of the pattern has a different rhythmic profile.

On the other hand, the geometric method is better suited to finding the most salient pattern in Fig. 2a, consisting of all the notes in bar 13 except the tied-over G4. This pattern occurs again in bar 14, transposed up a fourth, and then once more at the original pitch in bar 15. Each note is annotated with its relative height on the stave (or *morphetic pitch number* [14]), taking C4 to be 60. Underneath the stave, ontimes are measured in quaver beats starting from zero. The first note in this excerpt, G3, can be represented by the datapoint $\mathbf{d}_1 = (0, 57)$, since it has ontime 0 and morphetic pitch number 57. A scatterplot of morphetic pitch number against ontime for this excerpt is shown in Fig. 2b. Meredith et al. [14] call the set of all datapoints representing an excerpt a *dataset*, denoted by D. Restricting attention to bars 13-15, we begin with the dataset

$$D = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{26}\}. \quad (1)$$

A *pattern* is defined as a non-empty subset of a dataset. In our example, we will choose to look at the patterns

$$P = \{\mathbf{d}_1, \ldots, \mathbf{d}_8\}, \text{ and } Q = \{\mathbf{d}_9, \mathbf{d}_{11}, \ldots, \mathbf{d}_{17}\}. \quad (2)$$

The vector that translates $\mathbf{d}_1$ to $\mathbf{d}_9$ is

$$\mathbf{d}_9 - \mathbf{d}_1 = (3, 60) - (0, 57) = (3, 3) = \mathbf{v}. \quad (3)$$

We have given this vector a label $\mathbf{v} = (3, 3)$. It is this same vector $\mathbf{v}$ that translates $\mathbf{d}_2$ to $\mathbf{d}_{11}$, $\mathbf{d}_3$ to $\mathbf{d}_{12}, \ldots, \mathbf{d}_8$ to $\mathbf{d}_{17}$. Recalling the definitions of P and Q from (2), it is more succinct to say that 'the translation of P by $\mathbf{v}$ is equal to Q'. This translation is indicated in Fig. 2c.

Looking at Fig. 2c it is evident that as well as Q being a translation of P, pattern R is also a translation of P. Meredith et al. [14] call $\{P, Q, R\}$ the *translational equivalence class* of P in D, notated

$$TEC(P, D) = \{P, Q, R\}. \quad (4)$$

The TEC gives all the occurrences of a pattern in a dataset.

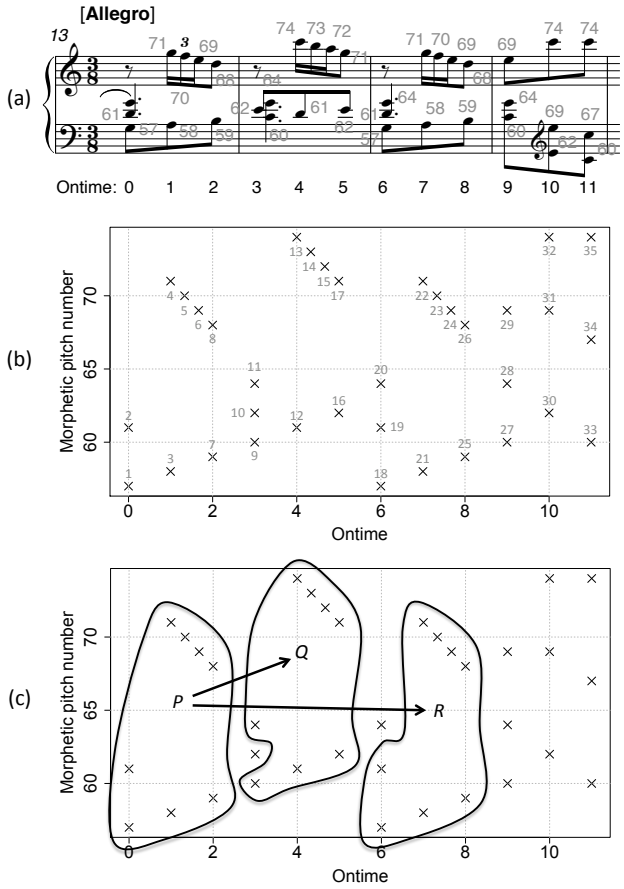


Figure 2. (a) Bars 13-16 of the Sonata in C major L3 by Scarlatti, annotated with morphetic pitch numbers and ontimes; (b) each note from the excerpt is converted to a point consisting of an ontime and a morphetic pitch number. Morphetic pitch number is plotted against ontime, and points are labelled in lexicographical order $\mathbf{d}_1$ to $\mathbf{d}_{35}$; (c) the same plot as above, with three ringed patterns, P, Q, R. Arrows indicate that both Q and R are translations of P.

So P is an example of a *translational pattern*, as translations of P, namely Q and R, exist in the dataset D. The formal definition of a translational pattern is as follows.

Definition. For a pattern P in a dataset D, the pattern P is a *translational pattern* if there exists at least one subset Q of D such that P and Q contain the same number of elements, and there exists *one* non-zero vector $\mathbf{v}$ that translates each datapoint in P to a datapoint in Q.

In the example in Fig. 2, two dimensions were considered (ontime and morphetic pitch number). The definitions and pattern discovery algorithms given by Meredith et al. [13] extend to k dimensions; specifically MIDI note number and duration are included as further dimensions.

The string-based method is not so well suited to Fig. 2a. The first step would be voice separation, generating perceptually valid melodies from the texture. Sometimes the scoring of the music makes separation simple [9], but even when voicing contains ambiguities, there are algorithms that can manage [1, 3]. Supposing fragments of the pattern in Fig. 2a were discovered among separated melodies,

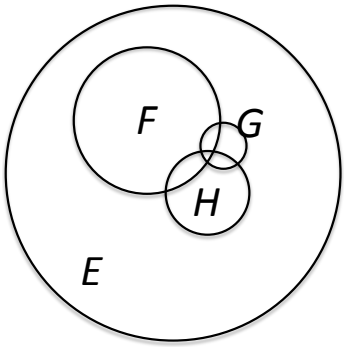


Figure 3. A Venn diagram (not to scale) for the number of patterns (up to translational equivalence) in a dataset. The total E is shown relative to the number typically returned by SIATEC (F), COSIATEC (G), and SIACT (H).

these fragments still would have to be correctly reunited. In this instance, even the most sophisticated string-based method [5] does not compare with the efficiency of the geometric method. The key difference between geometric and string-based approaches is the binding of ontimes to other musical information in the former, and the decoupling of this information in the latter. Both are valid methods for discovering patterns in music.

The reporting of existing *intra*-opus algorithms will often mention running time [3, 8, 12–14], occasionally *recall* is given [11, 17], and sometimes *precision* [10]. With the *inter*-opus discovery task an algorithm's output is seldom compared with a human benchmark [5,9]. The justification is that 'investigations of entire collections require considerable amounts of time and effort on the part of researchers' [9, p. 171]. Still, is it not worth knowing how an algorithm performs on a subset of the collection?

2. ALGORITHMS FOR PATTERN DISCOVERY

In equation 2, pattern P was introduced without explaining *how* it is discovered. It *could* be discovered by calculating all the TECs in the dataset D, and then certainly $TEC(P, D)$ will be among the output. However this approach is tremendously expensive and indiscriminate. It is expensive in terms of computational complexity, as there are 2^n patterns to partition into equivalence classes, where $n = |D|$ is the size of the dataset. Moreover, it is indiscriminate as no attempt is made to restrict the output in terms of 'musical importance': while P is arguably of importance, not all subsets of D are worth considering, yet they will also be among the output. The set E in Fig. 3 represents the output of this expensive and indiscriminate approach.

Therefore Meredith et al. [14] restrict the focus to a smaller set F, by considering how a pattern like P is *maximal*. Recalling (1) and (2), the pattern P is maximal in the sense that it contains all datapoints that are translatable in the dataset D by the vector $\mathbf{v} = (3, 3)$. It is called a *maximal translatable pattern* [14], written

$$P = MTP(\mathbf{v}, D) = \{\mathbf{d} \in D : \mathbf{d} + \mathbf{v} \in D\}. \quad (5)$$

Meredith et al.'s [14] structural inference algorithm (SIA) calculates all MTPs in a dataset, which requires $O(kn^2)$ calculations. While the TEC of each MTP must still be determined to give the set F in Fig. 3, this approach is enormously less expensive than partitioning 2^n patterns and involves a decision about musical importance: 'In music, MTPs often correspond to the patterns involved in perceptually significant repetitions' [14, p. 331]. SIA works by traversing the upper triangle of the similarity matrix

$$\mathbf{A} = \begin{pmatrix} \mathbf{d}_1 - \mathbf{d}_1 & \mathbf{d}_2 - \mathbf{d}_1 & \cdots & \mathbf{d}_n - \mathbf{d}_1 \\ \mathbf{d}_1 - \mathbf{d}_2 & \mathbf{d}_2 - \mathbf{d}_2 & \cdots & \mathbf{d}_n - \mathbf{d}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{d}_1 - \mathbf{d}_n & \mathbf{d}_2 - \mathbf{d}_n & \cdots & \mathbf{d}_n - \mathbf{d}_n \end{pmatrix}. \quad (6)$$

If the vector $\mathbf{w} = \mathbf{d}_j - \mathbf{d}_i$ is not equal to a previously calculated vector then a new MTP is created, $MTP(\mathbf{w}, D)$, with $\mathbf{d}_i$ as its first member. Otherwise $\mathbf{w} = \mathbf{u}$ for some previously calculated vector $\mathbf{u}$, in which case $\mathbf{d}_i$ is included in $MTP(\mathbf{u}, D)$. So it is possible to determine the set F for a dataset D by first running SIA on the dataset and then calculating the TEC of each MTP. The structural inference algorithm for translational equivalence classes (SIATEC) performs this task [14], and requires $O(kn^3)$ calculations.

To our knowledge there are two further algorithms that apply the geometric method to pattern discovery: the covering structural inference algorithm for translational equivalence classes (COSIATEC) [13] and a variant proposed by Forth & Wiggins [7]. COSIATEC *rates* patterns according to a heuristic for musical importance and produces a smaller output than SIATEC, the set labelled G in Fig. 3. The name COSIATEC derives from the idea of creating a *cover* for the input dataset:

1. Run SIATEC on $D_0 = D$, rate the discovered patterns using the heuristic for musical importance, and return the pattern P_0 that receives the highest rating.
2. Define a new dataset D_1 by removing from D_0 each datapoint that belongs to an occurrence of P_0.
3. Repeat step 1 for D_1 to give P_1, repeat step 2 to define D_2 from D_1, and so on until the dataset D_{N+1} is empty.
4. The output is

$$G = \{TEC(P_0, D_0), \ldots, TEC(P_N, D_N)\}. \quad (7)$$

Forth & Wiggins' variant on COSIATEC [7] uses a non-parametric version of the heuristic for musical importance and requires only one run of SIATEC. While only one run reduces the computational complexity of their version, it does mean that the output is always a subset of F, whereas running SIATEC on successively smaller datasets (steps 2 and 3 above) makes it possible to discover patterns beyond F (the portion $G \backslash F$ in Fig. 3).

3. THE PROBLEM OF ISOLATED MEMBERSHIP

In Sec. 2 we noted that pattern P from (2) could be discovered by running SIA on the dataset D from (1). This

is because P is the MTP (cf. equation 5) for the vector $\mathbf{v} = (3, 3)$ and SIA returns all such patterns in a dataset. However, D is a conveniently chosen example consisting only of bars 13-15 of Fig. 2a. How might an MTP be affected if the dataset is enlarged to include bar 16? Letting

$$D^+ = \{\mathbf{d}_1, \ldots, \mathbf{d}_{35}\}, \quad \mathbf{v} = (3, 3), \tag{8}$$

it can be verified that

$$P^+ = MTP(\mathbf{v}, D^+) = \{\mathbf{d}_1, \ldots, \mathbf{d}_8, \mathbf{d}_{18}, \mathbf{d}_{19}, \mathbf{d}_{22}\}. \tag{9}$$

Unfortunately P^+, the new version of P, contains three more datapoints, $\mathbf{d}_{18}$, $\mathbf{d}_{19}$, $\mathbf{d}_{22}$, that are isolated temporally from the rest of the pattern. This is an instance of what we call the 'problem of isolated membership'. It refers to a situation where a musically important pattern is contained *within* an MTP, along with other temporally isolated members that may or may not be musically important. Intuitively, the larger the dataset, the more likely it is that the problem will occur. Isolated membership affects all existing algorithms in the SIA family, and could prevent them from discovering some translational patterns that a music analyst considers noticeable or important (see Sec. 4 for further evidence in support of this claim).

Our proposed solution to the problem of isolated membership is to take the SIA output and 'trawl' *inside* each MTP from beginning to end, returning subsets that have a compactness greater than some threshold a and that contain at least b points. The *compactness* of a pattern is the ratio of the number of points in a pattern to the number of points in the region of the dataset in which the pattern occurs [13]. Different interpretations of 'region' lead to different versions of compactness. The version employed here is of least computational complexity $O(kn)$, and uses the lexicographical ordering shown in Fig. 2b. The *compactness* of a pattern $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$ in a dataset D is defined by

$$c(P, D) = l/|\{\mathbf{d}_i \in D : \mathbf{p}_1 \preceq \mathbf{d}_i \preceq \mathbf{p}_l\}|. \tag{10}$$

For instance, the compactness of pattern Q in Fig. 2c is $8/9$, as there are 8 points in the pattern and 9 in the dataset region $\{\mathbf{d}_9, \mathbf{d}_{10}, \ldots, \mathbf{d}_{17}\}$ in which the pattern occurs.

One of Meredith et al.'s [14] suggestions for improving/extending the SIA family is to 'develop an algorithm that searches the MTP TECs generated by SIATEC and selects all and only those TECs that contain convex-hull compact patterns' [p. 341]. The way in which our proposed solution is crucially different to this suggestion is to trawl *inside* MTPs. It will not suffice to calculate the compactness of an entire MTP, since we know it is likely to contain isolated members. Other potential solutions to the problem of isolated membership are to:

- Segment the dataset before discovering patterns. The issue is how to segment appropriately—usually the discovery of patterns *guides* segmentation [2], not the other way round.
- Apply SIA with a 'sliding window' of size r. Approximately, this is equivalent to traversing only the elements on the first r superdiagonals of $\mathbf{A}$ in (6). The issue is that the sliding window could prevent the discovery of very noticeable or important patterns, if their generating vectors lie beyond the first r superdiagonals.
- Consider the set of all patterns that can be expressed as an *intersection* of MTPs, which may not be as susceptible to the problem of isolated membership. The issue with this larger class is that it is more computationally complex to calculate, and does not aim specifically at tackling isolated membership.

The algorithmic form of our solution is called a *compactness trawler*. It may be helpful to apply it to the example of P^+ in (9), using a compactness threshold of $a = 2/3$ and points threshold of $b = 3$. The compactness of successive subsets $\{\mathbf{d}_1\}, \{\mathbf{d}_1, \mathbf{d}_2\}, \ldots, \{\mathbf{d}_1, \ldots, \mathbf{d}_8\}$ of P^+ remains above the threshold of $2/3$ but then falls below, to $9/18$, for $\{\mathbf{d}_1, \ldots, \mathbf{d}_8, \mathbf{d}_{18}\}$. So we return to $\{\mathbf{d}_1, \ldots, \mathbf{d}_8\}$, and it is output as it contains $8 \geq 3 = b$ points. The process restarts with subsets $\{\mathbf{d}_{18}\}$, $\{\mathbf{d}_{18}, \mathbf{d}_{19}\}$, and then the compactness falls below $2/3$ to $3/5$ for $\{\mathbf{d}_{18}, \mathbf{d}_{19}, \mathbf{d}_{22}\}$. So we return to $\{\mathbf{d}_{18}, \mathbf{d}_{19}\}$, but it is discarded as it contains fewer than 3 points. The process restarts with subset $\{\mathbf{d}_{22}\}$ but this also gets discarded for having too few points. The whole of P^+ has now been trawled. The formal definition follows and has computational complexity $O(kn)$.

1. Let $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$ be a pattern in a dataset D and $i = 1$.
2. Let j be the smallest integer such that $i \leq j < l$ and $c(P_{j+1}, D) < a$, where $P_{j+1} = \{\mathbf{p}_i, \ldots, \mathbf{p}_{j+1}\}$. If no such integer exists then put $P' = P$, otherwise let $P' = \{\mathbf{p}_i, \ldots, \mathbf{p}_j\}$.
3. Return P' if it contains at least b points, otherwise discard it.
4. If j exists in step 2, re-define P in step 1 to equal $\{\mathbf{p}_{j+1}, \ldots, \mathbf{p}_l\}$, set $i = j + 1$, and repeat steps 2 and 3. Otherwise re-define P as empty.
5. After a certain number of iterations P will be empty and the output can be labelled $P'_1, \ldots, P'_N$, that is N subsets of the original P, where $0 \leq N \leq l$.

We give the name 'structural inference algorithm and compactness trawler' (SIACT) to the process of calculating all MTPs in a dataset (SIA), followed by the application of the compactness trawler to each. The compactness-trawling stage in SIACT requires $O(kmn)$ calculations, where m is the number of MTPs returned by SIA. If desired, it is then possible to take the output of SIACT and calculate the TECs. These TECs are represented by the set H in Fig. 3. To our knowledge, this newest member of the SIA family is the only algorithm intended to solve the problem of isolated membership.

4. COMPARATIVE EVALUATION

A music analyst (the second author) analysed the Sonata in C major L1 and the Sonata in C minor L10 by Scarlatti, the Prelude in C♯ minor BWV849 and the Prelude in E major BWV854 by Bach. The brief was similar to the intra-opus discovery task described in Sec. 1: given a piece of music in staff notation, discover translational patterns that occur within the piece. Thus, a benchmark of translational patterns was formed for each piece, the criteria for benchmark membership being left largely to the analyst's discretion. One criterion that was stipulated was to think in terms of an analytical essay: if a pattern would be mentioned in prose or as part of a diagram, then it should be included in the benchmark. The analyst is referred to as 'independent' because of the relative freedom of the brief and because they were not aware of the details of the SIA family, or our new algorithm. The analyst was also asked to report where aspects of musical interest had little or nothing to do with translational patterns, as these occasions will have implications for future work.

Three algorithms—SIA [14], COSIATEC [13] and our own, SIACT—were run on datasets that represented L1, L10, BWV849, and BWV854. For COSIATEC the non-parametric version of the rating heuristic was used [7] and for SIACT we used a compactness threshold of $a = 2/3$ and a points threshold of $b = 3$. The choice of $a = 2/3$ means that at the beginning of an input pattern, the compactness trawler will tolerate one non-pattern point between the first and second pattern points, which seems like a sensible threshold. The choice of $b = 3$ means that a pattern must contain at least three points to avoid being discarded. This is an arbitrary choice and may seem a little low to some. Each point in a dataset consisted of an ontime, MIDI note number (MNN), morphetic pitch number (MPN), and duration (voicing was omitted for simplicity on this occasion). Nine combinations of these four dimensions were used to produce 'projections' of datasets [14], on which the algorithms were run. These projections always included ontime, bound to: MNN *and* duration; MNN; MPN *and* duration; MPN; duration; MNN mod 12 *and* duration; MNN mod 12; MPN mod 7 *and* duration; MPN mod 7. For the first time to our knowledge, the use of pitch modulo 7 and 12 enabled the concept of octave equivalence to be incorporated into the geometric method.

If a pattern is in the benchmark, it is referred to as a target; otherwise it is a non-target. An algorithm is judged to have discovered a target if a member of the algorithm's output is *equal* to the target pattern or a translation of that pattern. In the case of COSIATEC the output consists of TECs, not patterns. So we will say it has discovered a target if that target is a member of one of the output TECs. Table 1 shows the recall and precision of the three algorithms for each of the four pieces. Often COSIATEC did not discover any target patterns, so for these pieces it has zero recall and precision. This is in contrast to the parametric version's quite encouraging results for Bach's two-part inventions [12,13]. When it did discover some target patterns in L10, COSIATEC achieved a better precision than the other algorithms, as it tends to return far fewer patterns per piece (168 on average compared with 8,284 for SIACT and 385,299 for SIA). Hence the two remaining contenders are SIA and SIACT. SIACT, defined in Sec. 3, out-performs SIA in terms of both recall and precision. Having examined cases in which SIA and COSIATEC fail to discover targets, we attribute the relative success of SIACT to its being intended to solve the problem of isolated membership. Across the four pieces, the running times of SIA and SIACT are comparable (the latter is always slightly greater since the first stage of SIACT *is* SIA).

Piece →	L1	L10	BWV849	BWV854
Algorithm ↓	Recall			
SIA	.29	.22	.28	.22
COSIATEC	.00	.17	.00	.00
SIACT	.50	.65	.56	.61
	Precision			
SIA	$1.5\,e^{-5}$	$1.1\,e^{-5}$	$1.3\,e^{-5}$	$1.8\,e^{-5}$
COSIATEC	.00	.02	.00	.00
SIACT	$2.6\,e^{-3}$	$1.5\,e^{-3}$	$7.8\,e^{-4}$	$2.0\,e^{-3}$

Table 1. Results for three algorithms on the intra-opus pattern discovery task, applied to four pieces of music. Recall is the number of targets discovered, divided by the sum of targets discovered and targets not discovered. Precision is the number of targets discovered, divided by the sum of targets discovered and non-targets discovered.

5. DISCUSSION

This paper has discussed and evaluated algorithms for the intra-opus discovery of translational patterns. One of our motivations was the prospect of improving upon current solutions to this open MIR problem. A comparative evaluation was conducted, including two existing algorithms and one of our own, SIACT. For the pieces of music considered, it was found that SIACT out-performs the existing algorithms considerably with regard to recall and, more often than not, it is more precise. Therefore, our aim of improving upon the best current solution has been achieved. Central to this achievement was the formalisation of the 'problem of isolated membership'. It was shown that for a small and conveniently chosen excerpt of music, a maximal translatable pattern corresponded exactly to a perceptually salient pattern. However, when the excerpt was enlarged by just one bar, the MTP gained some temporally isolated members, and the salient pattern was lost inside the MTP. Our proposed solution, to trawl inside an MTP, returning compact subsets, led to the definition of SIACT.

The weight placed on the improved results reported here is limited somewhat by the extent of the evaluation, which includes only four pieces, all from the Baroque period, and all analysed by one expert. Extending and altering these conditions and assessing their effect on the performance of the three algorithms is a clear candidate for future work. There are also more sophisticated versions of

compactness and the compactness trawler algorithm that could be explored, and alternative values for the compactness and points thresholds, a and b. The discovery of 'exact repetition' has provided a sensible starting point for this research, but extending definitions such as (5) to allow for 'inexact repetition' is an important and challenging next step. Cases of failure, where `SIACT` does not discover targets, will be investigated. Perhaps some of these cases share characteristics that can be addressed in a future version of the algorithm. Although we have seen `SIA` presented before as the sorting of matrix elements [14], the connection that **A** in (6) makes with similarity matrices [15, 16] may lead to new insights or efficiency gains.

We will be trying to elicit more knowledge about the attributes of a pattern that matter to human analysts, so as to rank output patterns and to compare these attributes with the assumptions underlying `SIACT`. It could be that current pattern discovery methods overlook particular aspects of musical interest. If so a string-based or geometric method might be easily adapted, or very different methods may have to be developed. Could one focused algorithm encompass the many and diverse categories of musical pattern? It seems improbable, and the discussion of Figs. 1 and 2 in Sec. 1.1 could be interpreted as a counterexample. Hence, given the improved voice separation algorithms, and string-based and geometric methods that now exist, another worthy topic for future work would be the unification of a select number of algorithms within a single user interface. This would bring us closer to achieving our opening, more ambitious aim, of enabling music analysts, listeners, and students to engage with pieces of their choice in a novel and rewarding manner. To this end, the work reported here clearly merits further development.

6. ACKNOWLEDGEMENTS

This paper benefited from helpful discussions with David Meredith and Jamie Forth. We would also like to thank the four anonymous reviewers for their comments.

7. REFERENCES

[1] E. Cambouropoulos: "Voice and stream: perceptual and computational modeling of voice separation," *Music Perception*, Vol. 26, No. 1, pp. 75–94, 2008.

[2] E. Cambouropoulos: "Musical parallelism and melodic segmentation: a computational approach," *Music Perception*, Vol. 23, No. 3, pp. 249–267, 2006.

[3] S-C. Chiu, M-K. Shan, J-L. Huang, and H-F. Li: "Mining polyphonic repeating patterns from music data using bit-string based approaches," *Proceedings of the IEEE International Conference on Multimedia and Expo*, pp. 1170–1173, 2009.

[4] R. Clifford, M. Christodoulakis, T. Crawford, D. Meredith, and G. Wiggins: "A fast, randomised, maximal subset matching algorithm for document-level music retrieval," *Proceedings of the International Symposium on Music Information Retrieval*, 2006.

[5] D. Conklin and M. Bergeron: "Feature set patterns in music," *Computer Music Journal*, Vol. 32, No. 1, pp. 60–70, 2008.

[6] N. Cook: *A guide to musical analysis*, J.M. Dent and Sons, London, 1987.

[7] J. Forth and G. Wiggins: "An approach for identifying salient repetition in multidimensional representations of polyphonic music," *London algorithmics 2008: theory and practice*, College Publications, London, 2009.

[8] J-L. Hsu, C-C. Liu, and A. Chen: "Discovering nontrivial repeating patterns in music data," *IEEE Transactions on Multimedia*, Vol. 3, No. 3, pp. 311–325, 2001.

[9] I. Knopke and F. Jürgensen: "A system for identifying common melodic phrases in the masses of Palestrina," *Journal of New Music Research*, Vol. 38, No. 2, pp. 171–181, 2009.

[10] O. Lartillot: "Efficient extraction of closed motivic patterns in multi-dimensional symbolic representations of music," *Proceedings of the International Symposium on Music Information Retrieval*, pp. 191–198, 2005.

[11] C. Meek and W. Birmingham: "Automatic thematic extractor," *Journal of Intelligent Information Systems*, Vol. 21, No. 1, pp. 9–33, 2003.

[12] D. Meredith: "Point-set algorithms for pattern discovery and pattern matching in music," *Proceedings of the Dagstuhl Seminar on Content-Based Retrieval*, 2006.

[13] D. Meredith, K. Lemström, and G. Wiggins: "Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music," *Proceedings of the Cambridge Music Processing Colloquium*, 2003.

[14] D. Meredith, K. Lemström, and G. Wiggins: "Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music," *Journal of New Music Research*, Vol. 31, No. 4, pp. 321–345, 2002.

[15] G. Peeters: "Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach," *Proceedings of the International Symposium on Music Information Retrieval*, 2007.

[16] X. Ren, L. Smith, and R. Medina: "Discovery of retrograde and inverted themes for indexing musical scores," *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, pp. 252–253, 2004.

[17] P-Y. Rolland: "FlExPat: flexible extraction of sequential patterns," *Proceedings of the IEEE International Conference on Data Mining*, pp. 481–488, 2001.

[18] E. Ukkonen, K. Lemström, and V. Mäkinen: "Geometric algorithms for transposition invariant content-based music retrieval," *Proceedings of the International Symposium on Music Information Retrieval*, 2003.

A MULTI-PERSPECTIVE EVALUATION FRAMEWORK FOR CHORD RECOGNITION

Verena Konz
Saarland University
and MPI Informatik
vkonz@mpi-inf.mpg.de

Meinard Müller
Saarland University
and MPI Informatik
meinard@mpi-inf.mpg.de

Sebastian Ewert
Computer Science III
University of Bonn
ewerts@iai.uni-bonn.de

ABSTRACT

The automated extraction of chord labels from audio recordings constitutes a major task in music information retrieval. To evaluate computer-based chord labeling procedures, one requires ground truth annotations for the underlying audio material. However, the manual generation of such annotations on the basis of audio recordings is tedious and time-consuming. On the other hand, trained musicians can easily derive chord labels from symbolic score data. In this paper, we bridge this gap by describing a procedure that allows for transferring annotations and chord labels from the score domain to the audio domain and vice versa. Using music synchronization techniques, the general idea is to locally warp the annotations of all given data streams onto a common time axis, which then allows for a cross-domain evaluation of the various types of chord labels. As a further contribution of this paper, we extend this principle by introducing a multi-perspective evaluation framework for simultaneously comparing chord recognition results over multiple performances of the same piece of music. The revealed inconsistencies in the results do not only indicate limitations of the employed chord labeling strategies but also deepen the understanding of the underlying music material.

1. INTRODUCTION

In recent years automated chord recognition, which deals with the computer-based harmonic analysis of audio recordings, has been of increasing interest in the field of music information retrieval (MIR), see e. g. [1, 4, 5, 7, 12, 14]. The principle of harmony is a central attribute of Western tonal music, where the succession of chords over time often forms the basis of a piece of music. Such harmonic chord progressions are not only of musical importance, but also constitute a powerful mid-level representation for the underlying musical signal and can be applied for various tasks such as music segmentation, cover song identification, or audio matching [10, 13].

The evaluation of chord labeling procedures itself, which is typically done by comparing the computed chord labels with manually generated ground truth annotations, is far from being an easy task. Firstly, the assignment of chord labels to specific musical sections is often ambiguous due to musical reasons. Secondly, dealing with performances given as audio recording, the ground truth annotations have to be specified in terms of *physical units* such as seconds. Thus, specifying musical segments becomes a cumbersome task, which, in addition, has to be done for each performance separately. On the other hand, musicians trained in harmonics are familiar with assigning chord labels to musical sections. However, the analysis is typically done on the basis of musical scores, where the sections are given in terms of *musical units* such as beats or measures. When dealing with performed audio recordings, such annotations are only of limited use.

As one main contribution of this paper, we introduce an automated procedure for transferring annotations and chord labels from the score domain to the audio domain and vice versa, thus bridging the above mentioned gap between MIR researchers and musicians. Given the score of a piece of music, we assume that musical sections specified in terms of beats or measures are labeled using the conventions introduced by Harte [4]. In case the score is given in some computer-readable format such as MusicXML or LilyPond [6], recent software allows for exporting the score into an uninterpreted MIDI file, where the tempo is set to a known constant value. This allows for directly transferring the score-based ground truth annotations to a MIDI-based ground truth annotation. We then use music synchronization techniques [9] to temporally align the MIDI file to a given audio recording. Finally, the resulting alignment information can be used to temporally warp the audio annotations onto a common musically meaningful time axis, thus allowing a direct comparison to the ground truth annotations.

As a second contribution, we extend this principle by suggesting a novel multi-perspective evaluation framework, where we simultaneously compare chord recognition results over multiple performances of the same piece of music. In this way, consistencies and inconsistencies in the chord recognition results over the various performances are revealed. This not only indicates the capability of the employed chord labeling strategy but also lies the basis for a more detailed analysis of the underlying music material. As a final contribution, we indicate the potential of our framework by giving such detailed harmonic analyses by

means of three representative examples.

The remainder of this paper is organized as follows. First, in Sect. 2 we give an overview about music synchronization. Then, in Sect. 3 we present the multi-perspective evaluation framework. In Sect. 4 we demonstrate our framework giving an in-depth analysis of typical chord recognition errors. Conclusions and prospects on future work are given in Sect. 5.

2. MUSIC SYNCHRONIZATION

For the methods presented in the following sections the concept of music synchronization is of particular importance. In general, the goal of music synchronization is to determine for a given position in one version of a piece of music, the corresponding position within another version. Most synchronization algorithms rely on some variant of dynamic time warping (DTW) and can be summarized as follows. First, two given versions of a piece of music are converted into feature sequences, say $X := (X_1, X_2, \ldots, X_N)$ and $Y := (Y_1, Y_2, \ldots, Y_M)$, respectively. In this context, chroma features have turned out to yield robust alignment results even in the presence of significant artistic variations. In the following we employ CENS (**C**hroma **E**nergy **N**ormalized **S**tatistics) features, a variant of chroma features making use of short-time statistics over energy distributions within the chroma bands, for a detailed description see [9]. Additionally, we consider non-standard tunings similar to Gómez [3]. Then, an $N \times M$ cost matrix C is built up by evaluating a local cost measure c for each pair of features, i. e., $C(n,m) = c(x_n, y_m)$ for $n \in [1:N] := \{1, 2, \ldots, N\}$ and $m \in [1:M]$. Each tuple $p = (n,m)$ is called a *cell* of the matrix. A (global) *alignment path* is a sequence $(p_1, \ldots, p_L)$ of length L with $p_\ell \in [1:N] \times [1:M]$ for $\ell \in [1:L]$ satisfying $p_1 = (1,1)$, $p_L = (N,M)$ and $p_{\ell+1} - p_\ell \in \Sigma$ for $\ell \in [1:L-1]$. Here, $\Sigma = \{(1,0), (0,1), (1,1)\}$ denotes the set of admissible step sizes. The *cost* of a path $(p_1, \ldots, p_L)$ is defined as $\sum_{\ell=1}^{L} C(p_\ell)$. A cost-minimizing alignment path, which constitutes the final synchronization result, can be computed via dynamic programming from C. For a detailed account on DTW and music synchronization we refer to [9].

Based on this general strategy, we employ a synchronization algorithm based on high-resolution audio features as described in [2]. This approach, which combines the high temporal accuracy of onset features with the robustness of chroma features, generally yields robust music alignments of high temporal accuracy.

3. MULTI-PERSPECTIVE VISUALIZATION

A score in a computer readable format such as LilyPond or MusicXML is available for many classical pieces of music [11]. For a trained musician it is much more intuitive to annotate the chords of a piece on the basis of the underlying score than on the basis of an audio recording. However, such an annotation is not directly transferable to an audio recording of the same piece, as both use very different notions of time. Furthermore, this also implies that this annotation cannot be used directly to evaluate the results of an audio-based automatic chord labeling method. In this section, we present a method integrating music synchronization techniques, which allows for a direct comparison of chord labels derived from different versions of a piece of music. This approach has several advantages. Firstly, the manual annotation becomes much more intuitive. Secondly, the position of a chord recognition error in an audio recording can be easily traced back to the corresponding position in the score. This allows for a very efficient in-depth error analysis as we will show in Sect. 4. Thirdly, a single score-based annotation can be transfered to an arbitrary number of audio recordings for the underlying piece.

In the following, we assume that an audio recording and a score in computer readable format are given for a piece of music. Additionally, chord labels manually annotated by a trained musician on the basis of a score are given as well as labels automatically derived from the audio recording via some computer-based method. In a first step, we export the score to a MIDI representation. This can be done automatically using existing software. Beat and measure positions are preserved during the export, such that the score-based annotations are still valid for the MIDI file. In a next step, we derive CENS features from the MIDI as well as from the audio as mentioned in Sect. 2, say $X := (X_1, X_2, \ldots, X_N)$ and $Y := (Y_1, Y_2, \ldots, Y_M)$, respectively. Since each CENS feature corresponds to a time frame, we can also create two binary chord vector sequences, $A := (A_1, \ldots, A_N)$ and $B := (B_1, \ldots, B_M)$, which encode the given chord labels in a framewise fashion. Here, $A_n, B_m \in \{0,1\}^d$ for $n \in [1:N]$ and $m \in [1:M]$. The constant d equates the number of considered chords. A value of one in a vector component encodes the chord prevalent in the corresponding time frame. As we consider in the following only the 24 major and minor chords ($d = 24$), we have to map the given chord labels in a meaningful way to one of these. To this end, we employ the interval comparison of the dyad, which was used for MIREX 2009 [8] and takes into account only the first two intervals of each chord. Thus, augmented and diminished chords are mapped to major and minor respectively, as well as any other label having a major or minor third as its first interval. Using the first four measures of Chopin's Mazurka Op. 68 No. 3 as an example, we illustrate the sequences A for the score and B for the audio in Fig.1(b) and 1(c), respectively. Note that in Fig.1(b) the time is expressed in terms of measures, while in Fig.1(c) the time is given in seconds. This different notion of time prevents a comparison of A and B at this point.

The next step consists of synchronizing the two CENS features sequences X and Y as mentioned in Sect. 2. The resulting alignment path $p = (p_1, \ldots, p_L)$ encodes temporal correspondences between elements of X and Y. Following the same time frame division, the alignment path also encodes correspondences between the sequences A and B. Using this linking information, we locally stretch and contract the audio chord vector sequence B according to the warping information supplied by p. Here, we have to consider two cases. In the first case, p contains a subse-

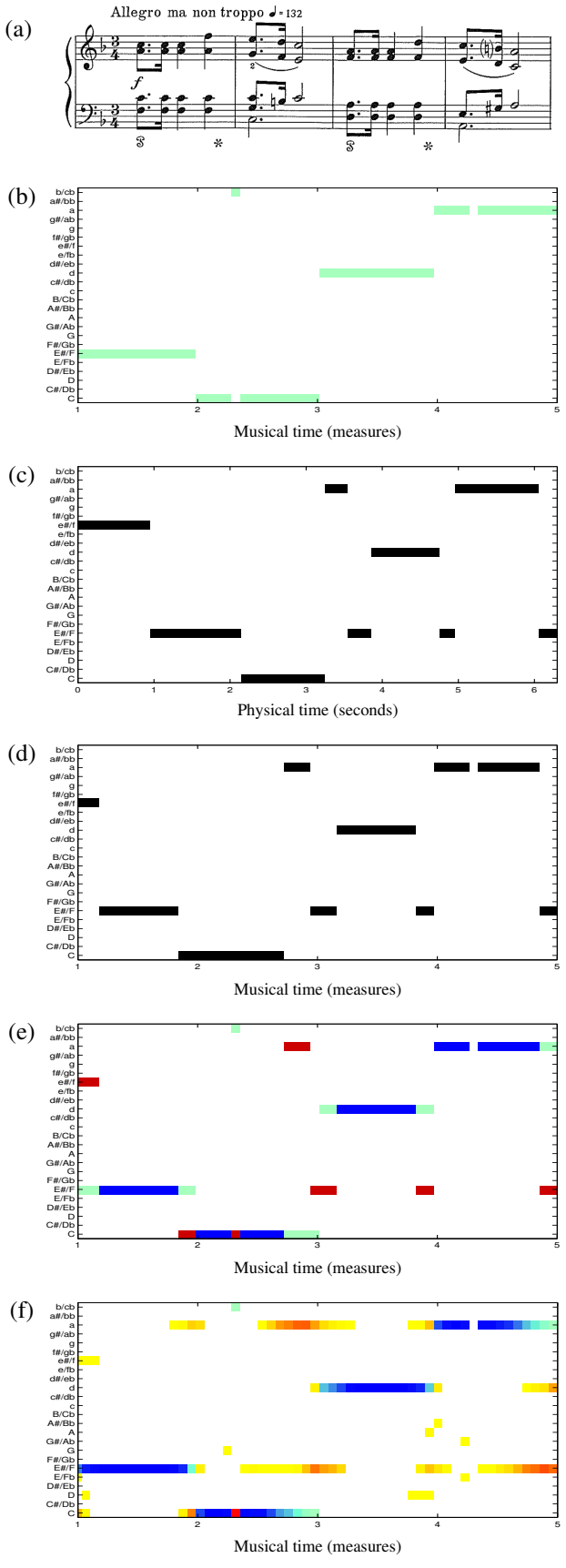


Figure 1. Various chord annotations visualized for the Chopin Mazurka Op. 68 No. 3 (F major), mm. 1-4. **(a)** Score. **(b)** Score-based ground truth chord labels. **(c)** Computed audio chord labels (physical time axis). **(d)** Warped audio chord labels (musical time axis). **(e)** Overlayed score and audio chord labels. **(f)** Multi-perspective overlay of score and audio chord labels.

quence of the form

$$(n, m), (n + 1, m), \ldots, (n + \ell - 1, m)$$

for some $\ell \in \mathbb{N}$, i. e., the ℓ score-related vectors $A_n, \ldots, A_{n+\ell-1}$ are aligned to the single audio-related vector B_m. In this case, we duplicate the vector B_m by taking ℓ copies of it. In the second case, p contains a subsequence of the form

$$(n, m), (n, m + 1), \ldots, (n, m + \ell - 1)$$

for some $\ell \in \mathbb{N}$, i. e., the score-related vector A_n is aligned to ℓ audio-related vectors $B_m, \ldots, B_{m+\ell-1}$. In this case, we replace the ℓ vectors by the vector $B_{m+\lfloor \ell/2 \rfloor}$. The resulting warped version of B is denoted by $\bar{B}$. Note that the length of $\bar{B}$ equals the length N of A, see Fig. 1(d). For the visualization we set all vectors in $\bar{B}$ to 0, where no groundtruth chord label is available, as for example in the middle of measure (abbreviated mm.) 4, see Fig. 1(d).

Overall, we have now converted the physical time axis of the audio chord vector sequence B to the musically meaningful measure axis, as used for A. Finally, we can visualize the differences between the score-based and the audio-based chord labels by overlaying A and $\bar{B}$, see Fig. 1(e). Here, the vertical axis represents the 24 major/minor chords, starting with the 12 major chords and continuing with the 12 minor chords. Blue entries now indicate areas, where the ground truth labels and the audio chord labels coincide. On the contrary, green and red encode the differences between the chord labels. Here, green entries correspond to the ground truth chord labels derived from the score, whereas red entries correspond to the audio chord labels. For example, at the beginning of mm. 2 the score as well as the audio chord labels indicate a C major chord. On the contrary, at the end of mm. 2 there is a C major chord specified in the score, while the chord labels derived from the audio incorrectly specify an A minor chord. Using the measure-based time information, we can look directly at the corresponding position in the score and analyze the underlying reason for this error. We will demonstrate this principle extensively in Sect. 4, where we present an in-depth analysis of typical errors produced by automatic chord labeling methods.

Next, we extend the just developed concept by introducing a multi-perspective visualization, see Fig. 1(f). Here, we make use of the fact that for classical pieces usually many different interpretations and recordings are available. Visualizing the chord recognition results simultaneously for multiple audio recordings of the same piece, we can analyze the consistency of errors across these recordings. On the one hand, if an error is not consistent, then this might indicate a chord ambiguity at the corresponding position. On the other hand, a consistent error might point to an intrinsic weakness of the automatic chord labeler, or an error in the manual annotations. This way, errors might be automatically classified before they are manually inspected.

In Fig. 1(f) the multi-perspective visualization for the first four measures of the Chopin Mazurka is represented. Here, we warped the automatically generated chord labels for 51 different audio recordings onto the musical time axis

using the steps described above. By overlaying the resulting chord vector sequences $\bar{B}$ for all pieces, we get a visualization similar to the previous one in Fig. 1(e), so that the visualization for one audio recording can be seen as a special case of the multi-perspective visualization. In the multi-perspective visualization, we distinguish two cases using two different color scales: one color scale ranges from dark blue to bright green, and the other color scale ranges from dark red to yellow. The first color scale from blue to green serves two purposes. Firstly, it encodes the score-based ground truth chord labels. Secondly, it shows the degree of consistency between the automatically generated audio labels and the score labels. For example, the dark blue entries at the beginning of mm. 2 show, that a C major chord is specified in the score labels, and most audio-based labels coincide with the score label here. At the end of mm. 2 the bright green shows that the score specifies a C major, but most audio-based results differ here from the score label. Analogously, the second color scale from dark red to yellow also fulfills two purposes. Firstly, it encodes the audio-based chord labels that differ from the score labels. Secondly, it shows how consistent an error actually is. For example, at the beginning of mm. 2 there are no red or yellow entries, since the score labels and the audio labels coincide here. However, at the end of mm. 2, most audio-based chord labels differ from the score labels. Here most chord labels either specify an F major or an A minor chord.

4. EVALUATION

None of the currently available automatic chord labeling approaches works flawlessly. Errors can either be caused by the inherent ambiguity in chord labeling, or by a weakness special to the employed chord labeler. An in-depth analysis allowing for a distinction between these error sources is a very hard and time-consuming task. In this section, we show how this process can be supported and accelerated using the evaluation and visualization framework presented in Sect. 3. To this end, we manually created score-based chord annotations for several pieces of music. Furthermore, we implemented a very simple baseline chord labeler to study very common sources of error in chord labeling.

4.1 Annotations

For the following evaluation, a trained musician (Verena Konz) manually annotated the chords for three pieces of Western classical music. Firstly, Mazurka in F major Op. 68 No. 3 by Chopin. Secondly, Prelude in C major BWV 846 by Bach. Thirdly, the first movement of Beethoven's Fifth Symphony, Op. 67. Using the underlying score, the annotations were created on the beat-level, and in the case of the Bach Prelude on the measure-level. The format and naming conventions used for the annotation were proposed by Harte [4]. The annotator paid much attention to capture even slight differences between adjacent chords. Hence, the bass tone as well as missing or added tones in chords are marked explicitly using the corresponding shorthands.

4.2 Baseline-method for chord recognition

A baseline chord labeler can be implemented using only a few simple operations. Given an audio recording, we first extract CENS features (see Sect. 2) resulting in a feature sequence $Y := (Y_1, Y_2, \ldots, Y_M)$. We derive ten features per second, with each feature considering roughly 1100 ms of the original audio signal. Non-standard tunings are considered as described in Sect. 2. Then, we define a total of 24 chord templates, 12 templates for the major chords and 12 for the minor chords. The considered templates are 12-dimensional vectors, in which the respective three tones of the corresponding major(minor) chord (the root note, the major(minor) third and the fifth) are set to 1 and the rest to 0. Thus, we obtain e. g. for C major the template

$$(1,0,0,0,1,0,0,1,0,0,0,0)$$

and for C minor the template

$$(1,0,0,1,0,0,0,1,0,0,0,0).$$

Let T in the following denote the set of all 24 chord templates. In a next step, we choose a distance function d, which measures the distance of the i-th feature vector Y_i to a template $t \in T$.

$$d : [0,1]^{12} \times [0,1]^{12} \mapsto [0,1]$$
$$d(t, Y_i) = 1 - \frac{\langle t, Y_i \rangle}{\|t\| \cdot \|Y_i\|},$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product and $\|\cdot\|$ the Euclidean norm. By minimizing over $t \in T$ we can find the best matching chord template t^* for the i-th feature vector.

$$t^* = \underset{t \in T}{\operatorname{argmin}} \quad d(t, y_i)$$

The chord label associated with t^* constitutes the final result for the i-th frame.

4.3 Experiments

We start our evaluation by looking again at our running example, Chopin Mazurka Op. 68 No. 3. Our proposed visualization method clearly reveals various chord recognition errors, see Fig. 1(e). Making use of the musical time axis, these errors can now easily be traced back to the corresponding position in the score and analyzed further. For example, at the beginning of the piece, the score-based ground truth annotation corresponds to F major, whereas the computed audio-based annotation corresponds to F minor. A mix-up of major and minor often appears in the chord recognition task. The next misclassification occurs at the end of mm. 1, where the ground truth still corresponds to F major, but the computed annotation specifies a C major, which is actually the subsequent chord in the ground truth. This may be a boundary problem or an error in the synchronization.

In the middle of mm. 2, we note that the ground truth chord is B minor, whereas the computed chord is C major. Having a look at the score, one can see that the chord in question is actually a B diminished chord. Due to the reduction of the manual annotation to major/minor chords, this chord is mapped to a B minor chord in the ground truth. Causing a misclassification here, this is often a problem in the major/minor evaluation based on the comparison of the dyad.

The next misclassifications are due to the musical ambiguity of chords. At the end of mm. 2 we observe in the score a C major chord, where the fifth is missing. Comparing on the dyad level, this chord is mapped to a C major chord in the ground truth. However, all the notes of the chord (C,E) are also part of an A minor chord, which is actually computed at this position. A similar problem occurs at the beginning and at the end of mm. 3, where the ground truth annotation corresponds to D minor, whereas the computed annotation corresponds to F major. The same phenomenon appears a last time at the end of mm. 4, where F major is recognized instead of A minor. This phenomenon is caused by ambiguities inherent to the chord labeling task and constitutes a very common problem. The chords in classical music rarely are pure major or minor chords, because tones are often missing or added. Hence, the recognition as well as the manual annotation process become a hard task.

Next, we illustrate what kind of additional information our multi-perspective visualization can provide compared to the just discussed visualization that only makes use of a single audio recording. Here, we consider again the first four measures of the Chopin Mazurka. Instead of using only one audio recording we overlay the chord recognition results for 51 different audio recordings in our multi-perspective visualization, see Fig. 1(f). Looking for consistencies and inconsistencies, it is possible to classify and investigate single errors even further. For example, the misclassified F minor chord in the beginning of mm. 1 (see Fig. 1(e)) seems to be an exception for the specific recording. This can be clearly seen from the multi-perspective visualization where only for a few of the 51 audio recordings F minor is computed instead of F major. Also, the misclassification at the end of mm. 4 (F major instead of A minor) is not consistent across all considered audio recordings. On the contrary, some of the misclassifications which we observed in the case of one audio recording (Fig. 1(e)), are consistently misclassified for most of the other audio recordings. For example, the diminished chord in the middle of mm. 2, the chord ambiguity problem occuring at the end of mm. 2 (A minor instead of C major), the beginning of mm. 3 (F major instead of D minor) and the end of mm. 4 (F major instead of A minor). Overall, the multi-perspective chord recognition allows for a classification of recognition errors into those specific to a recording and those independent of a recording.

As a further example we now consider the famous Bach Prelude in C major, BWV 846. The multi-perspective visualization for 5 different audio recordings for mm. 19-24 (see Fig. 2) again reflects the chord recognition problems

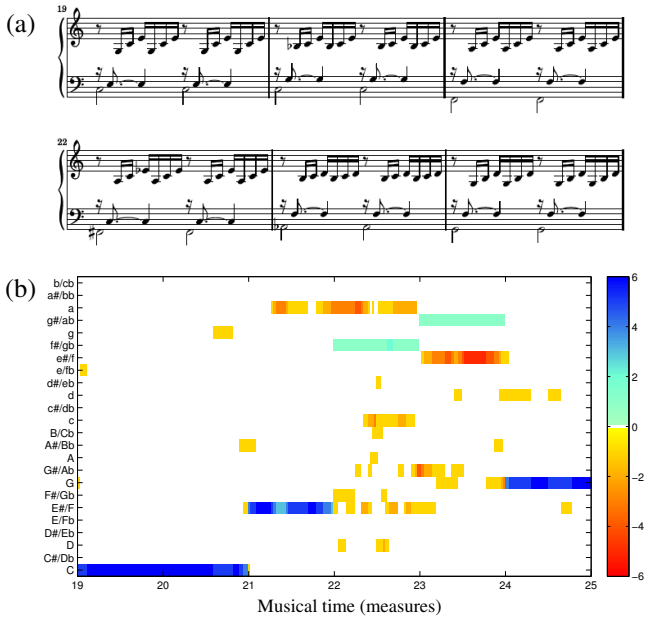


Figure 2. Bach BWV 846, mm. 19-24. **(a)** Score, **(b)** Multi-perspective overlay of score and audio chord labels.

related to diminished chords. At the beginning of the excerpt (mm. 19-21) and at the end (mm. 24) the chord recognition result for all audio recordings consistently agrees more or less with the ground truth. However, one can observe two passages with green entries in mm. 22-23. Looking at the corresponding position in the score, we find two diminished seventh chords, in mm. 22 an *F#:dim7* and in mm. 23 an *Ab:dim7*. Due to the reduction to major/minor chords these two chords are mapped to F# minor and Ab minor in the ground truth annotation, respectively, see Fig. 2. However, in most audio recordings an A minor chord is detected instead of *F#:dim7*, having two tones (A and C) in common. And instead of the *Ab:dim7* chord an F minor chord is found, for which even all three tones are present (F, Ab and C) due to the additional passing note C in the *Ab:dim7*. While the seventh chord in mm. 20 is recognized well for all recordings, we see that in mm. 21 the F major seventh chord was mistaken for an A minor chord, again due to chord ambiguity reasons.

As a last example we now consider the first movement of Beethoven's Fifth Symphony in 37 different audio recordings. Actually, this piece of music is much more complicated in terms of harmonic aspects than the previously considered Chopin and Bach examples. In the Beethoven example, we can often find the musical principles of suspension, passing notes or "unisono" passages. Here, the automatic chord recognition as well as the manual annotation are challenging and ambiguous tasks. One example for the use of nonharmonic tones in chords can be found in mm. 470-474, visualized in Fig. 3. Looking at the score, we observe in the left hand a D major chord with a missing fifth (mm. 470-473), but in the right hand a G is added in octaves to this D major chord. Being the fourth of D, the G can be seen as a nonharmonic tone in D major. This causes a chord misclassification for about 15 recordings, where G major or alternatively G minor is computed.

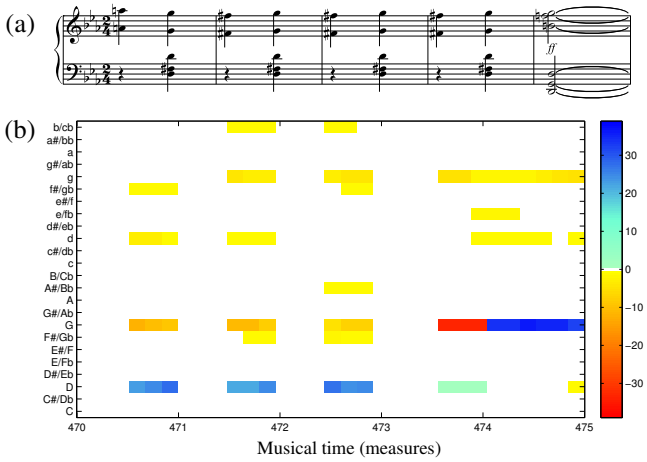


Figure 3. Beethoven's Fifth, mm. 470-474. **(a)** Score, **(b)** Multi-perspective overlay of score and audio chord labels.

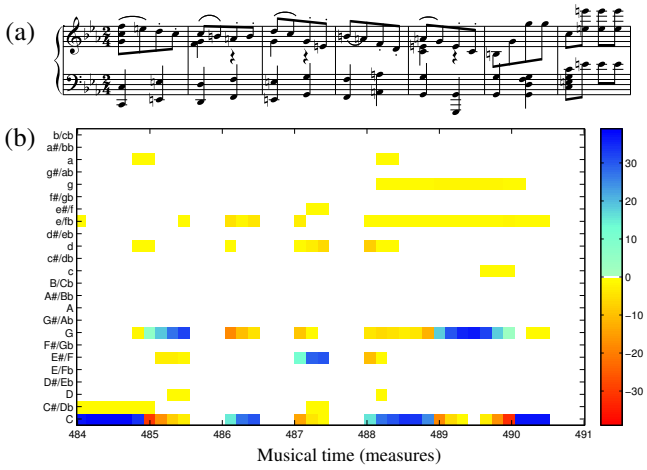


Figure 4. Beethoven's Fifth mm. 484-490. **(a)** Score, **(b)** Multi-perspective overlay of score and audio chord labels.

On the contrary, the G seventh chord in mm. 474 is recognized very well for all recordings. Note that the first beats of the measures 470-474 are not manually annotated, since the octaves do not represent meaningful chords.

Another example of a musical pattern that is found to be extremely problematic in the chord recognition task, is the principle of suspension. We illustrate the problems related to this musical characteristic using another excerpt (mm. 484-490) of Beethoven's Fifth, see Fig. 4. In each of the measures 484-488, one can find a suspension on the first eighth, which resolves into a major chord on the second eighth. This musical characteristic can easily be spotted in the multi-perspective visualization. Here, we see that at the beginning of each measure the number of audio recordings for which the computed annotation agrees with the ground truth is very low and gets higher afterwards. In mm. 490 finally the first complete pure major chord is reached. Note that the second beats of mm. 485-487 consist of passing notes to the next suspension. Hence, a meaningful chord cannot be assigned resulting in several beats missing a ground truth annotation.

5. CONCLUSIONS

In this paper, we have introduced a multi-perspective evaluation framework that allows for comparing chord label annotations across different domains (e. g., symbolic, MIDI, audio) and across different performances. This bridges the gap between MIR researchers, who often work on audio recordings, and musicologists, who are used to work with score data. In the future, we plan to apply our framework for a cross-modal evaluation of several computer-based chord labeling procedures, some of which working in the symbolic domain and others working in the audio domain. Furthermore, in a collaboration with musicologists, we are investigating how recurrent tonal centers of a certain key can be determined automatically within large musical works. Here, again, our multi-perspective visualization based on a musically meaningful time axis has turned out to be a valuable analysis tool.

Acknowledgement. The first two authors are supported by the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University. The third author is funded by the German Research Foundation (DFG CL 64/6-1).

6. REFERENCES

[1] J. P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proc. ISMIR, London, UK*, 2005.

[2] S. Ewert, M. Müller, and P. Grosche. High resolution audio synchronization using chroma onset features. In *Proc. of IEEE ICASSP*, Taipei, Taiwan, 2009.

[3] E. Gómez. Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing*, 18(3):294–304, 2006.

[4] C. Harte, M. Sandler, S. Abdallah, and E. Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proc. ISMIR, London, UK*, 2005.

[5] K. Lee and M. Slaney. A unified system for chord transcription and key extraction using hidden Markov models. In *Proc. ISMIR, Vienna, Austria*, 2007.

[6] LilyPond. `http://www.lilypond.org`.

[7] M. Mauch, D. Müllensiefen, S. Dixon, and G. Wiggins. Can statistical language models be used for the analysis of harmonic progressions? In *Proceedings of the 10th International Conference on Music Perception and Cognition, Sapporo, Japan*, 2008.

[8] MIREX 2009. Audio Chord Detection Subtask. `http://www.music-ir.org/mirex/2009/index.php/Audio_Chord_Detection`.

[9] M. Müller. *Information Retrieval for Music and Motion*. Springer, 2007.

[10] M. Müller, F. Kurth, and M. Clausen. Audio matching via chroma-based statistical features. In *Proc. ISMIR, London, GB*, 2005.

[11] Mutopia Project. `http://www.mutopiaproject.org`.

[12] J. T. Reed, Y. Ueda, S. Siniscalchi, Y. Uchiyama, S. Sagayama, and C.-H. Lee. Minimum classification error training to improve isolated chord recognition. In *Proc. ISMIR, Kobe, Japan*, 2009.

[13] J. Serrà, E. Gómez, P. Herrera, and X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech & Language Processing*, 16(6):1138–1151, 2008.

[14] A. Sheh and D. P. W. Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In *Proc. ISMIR, Baltimore, Maryland, USA*, 2003.

A PROBABILISTIC APPROACH TO MERGE CONTEXT AND CONTENT INFORMATION FOR MUSIC RETRIEVAL

Riccardo Miotto
University of Padova
miottori@dei.unipd.it

Nicola Orio
University of Padova
orio@dei.unipd.it

ABSTRACT

An interesting problem in music information retrieval is how to combine the information from different sources in order to improve retrieval effectiveness. This paper introduces an approach to represent a collection of tagged songs through an hidden Markov model with the purpose to develop a system that merges in the same framework both acoustic similarity and semantic descriptions. The former provides content-based information on song similarity, the latter provides context-aware information about individual songs. Experimental results show how the proposed model leads to better performances than approaches that rank songs using both a single information source and a their linear combination.

1. INTRODUCTION

The widespread diffusion of digital music occurred during the last years has brought music information retrieval (MIR) to the general attention. A central goal of MIR is to create systems that can efficiently and effectively retrieve songs from a collection of music content according to some sense of similarity with a given query. In information retrieval systems, the concept of similarity plays a key role and can dramatically impact performances. Yet, in music applications, the problem of selecting an optimal similarity measure is even more difficult because of the intrinsic subjectivity of the task: users may not consistently agree upon whether, or at which degree, a pair of songs or artists are similar.

In the last years, in order to deal with the subjective nature of music similarity, it became very common to describe songs as a collection of meaningful terms, or *tags*, as done in Last.fm [1] and Pandora [2]. In particular, tags are often, directly or indirectly, provided by end users and can represent a variety of different concepts including genre, instrumentation, emotions, geographic origins, and so on. Many approaches have been developed to collect tags, ranging from mining the Web and exploiting social behavior of users, to automatic annotation of music through machine learning algorithms. Tags are useful because they contextualize a song – for instance describing an historical period, a geographical area, or a particular use of the song – through an easy high-level representation. This information can then be used to retrieve music documents, to provide recommendations or to generate playlists.

Excluding the case of Pandora, where songs are annotated by human experts to guarantee high quality and consistency, in automatic systems or when the social behavior of users is kept into account, the semantic descriptions may be very noisy. In automatic approaches, for example, the quality of the prediction strictly depends on the quality of the training set, on the quality of the model, and on other issues such as parameter overfitting or term normalization. On the other hand, standard content-based music similarity, computed directly on music features, can be exploited to improve the quality of the retrieval, without requiring additional training operations.

The goal of this paper is to provide a general model to describe a music collection and easily retrieve songs combining both content-based similarity and context-aware tag descriptions. The model is based on an application of hidden Markov models (HMMs) and of the Viterbi algorithm to retrieve music documents. The main applicative scenario is cross-domain music retrieval, where music and text information sources are merged.

1.1 Related Work

There has been a considerable amount of research devoted to the topic of music retrieval, recommender systems and music similarity. Some of the most well-known commercial and academic systems have been described in [2]. The model proposed in this paper fits the scenario of item-based retrieval systems, combining pure acoustic similarity and semantic descriptions.

Methodologies that merge different heterogeneous sources of information have been recently proposed in [1] for the task of semantic discovery, in [9] for artist recommendation and in [16] for music classification. All of these approaches learn a metric space to join and compare the different sources of information in order to provide the user with a single ranking list. Our approach is consistently different, because it is built on a graph-based representation of the collection that model both sources of informa-

[1] http://www.last.fm
[2] http://www.pandora.com

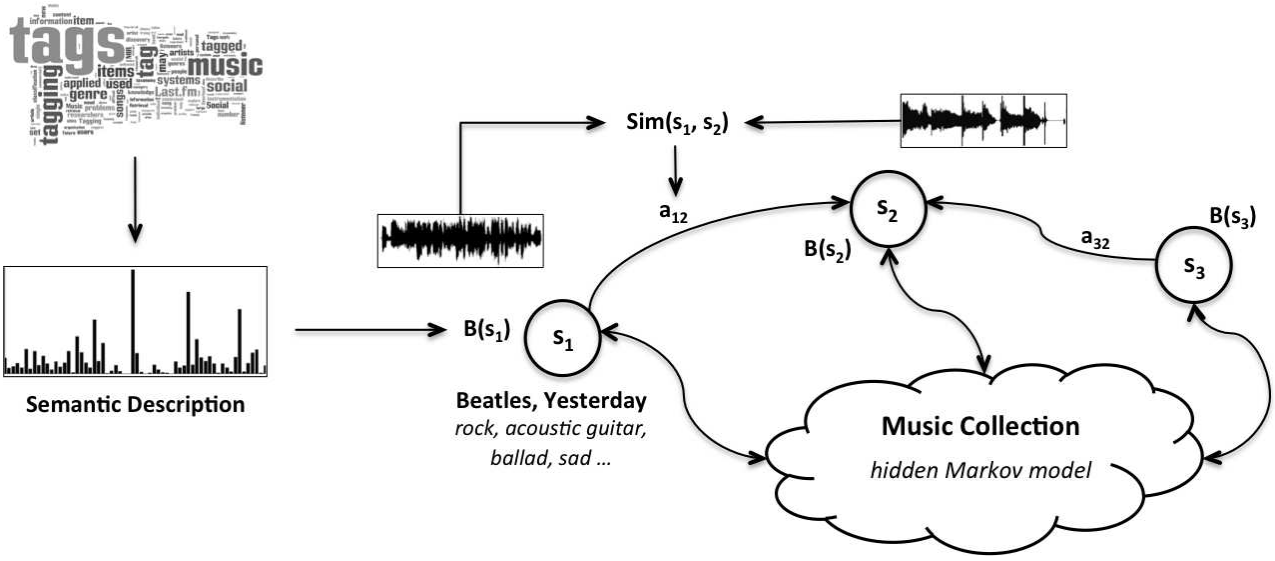


Figure 1. Overview of the proposed model. Songs s_i are states of a HMM: observation probabilities provide semantic descriptions, transitions probabilities are ruled by acoustic similarity between songs.

tion and thus it does not rely on an additional processing to combine them. Content-based music similarity can be computed directly on music features as done in [4, 7] or through a semantic space which describes music content with meaningful words [12, 18]. In our work, we exploit the properties of an HMM to combine these two descriptions to improve retrieval performances.

As it is well known, HMMs have been extensively used in many applications, which in particular involve processes through time such as speech recognition [13]. In the music information retrieval research area, they have been used in different scenarios: query-by-example [15], automatic identification [10], alignment [11], segmentation [14], and chord recognition [5]. At the best of our knowledge, this is the first application of HMMs in the task of cross-domain retrieval where music and text information is modeled in a single framework.

2. STATISTICAL MODELING OF A MUSIC COLLECTION

The general goal of music search engines is to retrieve a list of songs according to a particular principle. The principle could be described either directly by a general semantic indication, such as the tag "classic rock", or indirectly by a song, such as the set of tags assigned to "Yesterday". In both cases, the principle represents a user information need, and it can be assumed that the goal of an user is to *observe* consistently the application of this principle during the time of his access to the music collection. In the particular case of playlist generation, a system should be able to retrieve a list of music documents that are acoustically similar to the music the user likes and, at the same time, are relevant to one or more semantic labels that give a context to his information need.

The methodology presented in this paper aims at providing a formal and general model to retrieve music documents combining acoustic similarity and semantic descriptions given by social tags. That is, the goal is to propose a model that encompasses both content-based similarity and context-aware descriptors. To this end, HMMs are particularly suitable because they allow us to model two different sources of information. In fact, HMMs represent a doubly embedded stochastic process where, at each time step, the model performs a transition to a new state according to transition probabilities and emits a new symbol according to observation probabilities.

Thus HMMs can represent either content and context information, under the following assumptions:

- if each state represents a song in the collection, acoustic content-based similarity can be modeled by transition probabilities
- if the symbols emitted by the HMM are semantic labels, the context that describes each state can be modeled by observation probabilities.

A suitably built HMM (see Section 2.1) may be exploited to address the examples provided at the beginning of this section. On the one hand, the model can generate a path across songs while observing, for a defined number of time steps, the semantic label "classic rock". On the other hand, the model can start the path from the state associated to "Yesterday" and proceed to new states while observing the semantic labels associated to the seed song. In both cases, the songs in the path are likely to have a similar content because of transition probabilities and are likely to be in the same context because of emission probabilities.

Since states of a HMM are not directly observable, the paths across the song collection need to be computed by a decoding step, which highlights the most probable state sequence according to a sequence of observations. A representation of the proposed model is depicted in Figure 1.

2.1 Definition of the HMM

An HMM λ that represents a collections of tagged songs can be formally defined by:

1. The number of songs N in the collection, each song represented by a state of the HMM. The set of states is denoted as $S = \{s_1, s_2, ..., s_N\}$.

2. The number M of distinct tags that can be used to describe a song. The set of symbols is denoted as $V = \{v_1, v_2, ..., v_M\}$.

3. The state transition probability distribution $A = a_{ij}$, which defines the probability to move from state i to state j in a single step. Transition probabilities a_{ij} depends to the similarity between songs s_i and s_j.

4. The observation probability distribution of each state j, $B = b_j(k)$, which defines the probability that tag v_k is associated to song j. Observation probability values represent the strength of the relationships *song-tag*, which is indicated as *affinity* value.

5. The initial state distribution $\pi = \{\pi_i\}$, that defines the probability to start a path across the model beginning at state s_i. Differently from the standard definition of HMMs, the initial state distribution is computed dynamically at retrieval time, since it is strictly connected to the type of information need, as described in Section 2.3.

Although acoustic similarity is always a positive value, implying $a_{ij} > 0 \ \forall i, j$, with the aim of improving scalability, each state is directly connected to only the P most similar songs in the collection, while the transition probabilities with all the other states are set to 0. Heuristically, we set P to be the 10% of the global number of songs. At present, no deeper investigation has been carried out to highlight an optimal value of P. In order to obtain a stochastic model, both transition and emission probabilities are normalized, that is $\sum_j a_{ij} = 1$ and $\sum_k b_j(k) = 1$. Because of these two steps, transition probabilities are usually not symmetric, then $a_{ij} \neq a_{ji}$

After setting all the parameters, the HMM can be used to generate random sequences, where observed symbols are tags. Dually, well known algorithms can be used to decode the most probable state sequence according to a given observation sequence.

2.2 Computing the Relevance of Songs

The task at retrieval time is to highlight a sub-set of songs in the collection that are relevant to a particular query, either expressed by semantic labels or by a seed song. In the context of HMMs, the general problem can be stated as follows [13]: "given the model λ, and the observation sequence $\bar{O} = \{o(1), \ldots, o(T)\}$ with $o_j \in V$, the goal is to choose a state sequence $\bar{S} = \{s(1), \ldots, s(T)\}$ which is optimal in some sense". Clearly, the observations sequence represents the semantic description specified by the user need.

In literature, this problem is solved using the max-sum algorithm, which in HMMs applications is known as the Viterbi algorithm. The algorithm efficiently searches in the space of paths, in order to find the most probable one, with a computational cost that grows only linearly with the length of the chain. The algorithm is composed by a forward computation to find the maximization for the most probable path, and by a backward computation to decode the sequence of states. Although the general structure of the algorithm has been maintained, some key modifications in the recursion part of the forward computation have been introduced. Following the notation and the algorithm description provided in [13] the normal initialization and the modified recursion steps follow, for $1 \leq j \leq N$:

Initialization: for $t = 1$

$$\delta_t(j) = \pi_j \cdot obs_j(t) \tag{1}$$

$$\psi_t(j) = 0 \tag{2}$$

Recursion: for $2 \leq t \leq T$

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) \cdot a_{ij}] \cdot obs_j(t) \tag{3}$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) \cdot a_{ij}] \tag{4}$$

$$a_{kj} = \frac{a_{kj}}{d} \quad \text{with} \quad k = \psi_t(j) \tag{5}$$

As it can be seen, we introduce $obs_j(t)$, defined in the next section, which is a general function that indicates how the semantic description is considered during the retrieval process. This function plays the role of observations in typical decoding applications.

Equation 5 introduces a variation of the role of transition probabilities. In fact, because of the structure of the model, it could happen that the optimal path enters a loop between the same subset of songs or, in the worst case, jumps back and forth between two states. Clearly, this is a problem because the retrieved list would present the same set of songs multiple times. Moreover, the loop could be infinite, meaning that the algorithm cannot exit from it and the retrieval list would be composed by only few songs. We addressed this problem by introducing a decreasing factor d, which is applied to the transitions probabilities when they are selected in the forward step. So, when a transition is chosen, the probability a_{ij} is decreased by factor d (we set $d = 10$), as shown in Equation 5, in order to make unlikely that the state sequence would pass again through the corresponding edge. It has to be noted that the attenuation is carried out *locally*, meaning that it affects the structure of the model only during the current retrieval operation.

Another issue that has to be addressed is a limitation in the structure of standard HMMs. Because of the first-order Markov chain assumption, HMMs are generally poor at capturing long-range correlations between the observed

variables, that is between variables that are separated by many steps [3]. Earlier experiments showed that this limitation involved a decrease in precision when decoding long paths. In order to solve this problem, we considered the retrieval composed by many sub-retrieval operations, each one retrieving a sub-list of songs. Instead of performing a single backward decoding, the algorithm works for a subset of iterations, from which an optimal sub-path is built. Only the first n songs of this sub-path are considered in the final ranking list; at the end of each iteration the algorithm restarts from the last state of the n suggested. Given the locality of the approach, in this way we aim to keep constant the quality along the retrieved list, avoiding a decrease in precision.

2.3 Querying the Model

As often assumed in the interaction with music search engines, in our scenario a user can submit a query in two distinct ways: by providing a tag or by selecting a seed song in the collection. According to the kind of query, some of the model parameters are set differently.

In the *tag-based scenario*, the goal is to rank the songs according to their relevance with the provided tag and, at the same time, to their acoustic similarity. In this case, the observation sequence is composed simply by the chosen tag. We decided to set the initial state probability equal for all the states, in order to let the algorithm decide the beginning of the retrieved list. This scenario is very related to the standard HMMs case, then the function $obs_j(t)$ of Equations 1 and 3 is defined as

$$obs_j(t) = b_j(o_t) \tag{6}$$

for a generic state j, where observations o_t may be the same tag for all the time steps or it may change over time in case of playlist generation through more complex patterns.

In the *seed-song scenario*, when the query is submitted as a song q, the system is required to provide the user with a list of songs potentially similar to the query. In this case, the initial state distribution is forced to be 1 for the state representing the seed song and 0 for all the others. The observation sequence to be decoded is modeled as the vector of observations characterizing the seed song. The function $obs_j(t)$ of Equations 1 and 3 is proportional to the inverse of the Kullback-Leibler (KL) divergence between the semantic description of the seed song and the chosen state [6]. The choice of the KL divergence aims at generalizing the terms used for the tags, because it is related to the similarity of *concepts* associated to the tags rather than to the pure distance between lists of tags. It is important to note that the KL divergence is required also because each song is described by a set of tags. Clearly, we consider the inverse because the goal is to maximize the probability when the divergence is small. Therefore,

$$obs_j(t) \simeq \frac{1}{KL(b_j(\cdot), b_q(\cdot))} \tag{7}$$

for the generic state j and the initial seed state q; clearly, observations of q do not change over time t being linked to observations of the seed song. Since it is an observation probability, the actual value of $obs_j(t)$ undergoes a normalization process. It is worth noting that the use of KL divergence can be extended also to the tag-based scenario when the user provides a set of tags (instead of a single one) although this extension has not been tested yet.

3. EXPERIMENTAL EVALUATION

A big challenge when designing a music retrieval system is how to evaluate a novel methodology. Although several efforts have been made within the MIREX campaigns, because of well-known copyright issues, data of past campaigns are not always available to test new approaches. Ideally, the list of retrieved songs should be evaluated by humans, in order to consider effectively the subjective nature of the concept of music similarity. Being human evaluation a time consuming task, we use an automatic approach considering that reliable annotations on songs can be exploited to measure the quality of a ranking list.

We tested our model through the Computer Audition Lab (CAL500) dataset [18]: 502 songs played by 502 unique artists, each one annotated by a minimum of 3 individuals using a vocabulary of 174 tags. A song is considered to be annotated with a tag if 80% of the human annotators agreed that the tag would be relevant. CAL500 is a reasonable ground truth because annotations are highly reliable, complete and redundant – i.e., multiple persons explicitly evaluated the relevance of every tag for each song. So far, it has been mainly used to evaluate automatic music annotation systems, but we believe that it could be a reasonable ground truth also to evaluate qualitatively a retrieval task. Although the size of the dataset does not allow to perform experiments in terms of scalability, we argue that, at this point, it is more significant to test the effectiveness of the approach, to show if the model can provide improvements in the retrieval process.

In the experiments reported in this section, we require that each tag is associated with at least 30 songs and remove some tags that seemed to be redundant or overly subjective. The semantic space is then composed by 62 tags describing information about: genre, instrument, acoustic qualities, vocal characteristic, emotion, and usage.

Retrieval is evaluated with metrics considering both performances at the top and along the whole ranking list. Since a music retrieval system should maximize the quality of the retrieved items in the first positions, we evaluate the precision at the first 3, 5 and 10 positions (P3, P5, P10). Beside, we include the *mean average precision* (MAP) measure, in order to have also an evaluation along the whole ranking list. All these metrics are extensively used in the literature to assess the effectiveness of a retrieval system [8].

3.1 Acoustic Content-based Similarity

A number of methodologies have been proposed in literature to compute direct acoustic content-based similarity. In this set of experiments, we rely on the algorithm proposed in [7], which uses a single Gaussian with full covariance to

model a song. Although, some alternative approaches have been recently proposed [4], we use this one because of its efficiency and simplicity in the implementation. Songs are represented through vectors of Mel-Frequency Cepstral Coefficients together with their first and second derivatives (MFCC + delta) extracted from about one minute of music content, and the similarity between songs is computed using a symmetrized version of the KL divergence.

Section 2.1 describes how transition probabilities are computed from these similarity values, in particular by selecting for each state s_i the first P most similar songs and performing the normalization $\sum_j a_{ij} = 1$ with $s_j \in P$. It is important to note that we aim at proposing a general approach, which is independent on the way acoustic similarity is actually computed and which can be applied to other audio descriptors and other similarity measures. For this reason the computation of acoustic similarity is presented within the experimental evaluation section.

3.2 Semantic space

There are several approaches to collect tags for music, each with its own advantages and disadvantages [17]. Among all, we chose two different representations.

A first semantic description has been computed from the music content. We used the supervised multiclass labeling (SML) model described in [18] to automatically annotate songs with tags based on an audio content analysis. For a given song, the output of this algorithm is a vector of posterior probabilities named *semantic multinomial* that represents the strength of the relationship *tag-song*.

A second representation has been created by gathering the social tags from Last.FM, as reported on February 2010. For each song of the dataset, we collected two lists of social tags using their public data sharing AudioScrobbler [3] website. We gathered both the list of tags related to a song, and the list of tags related to an artist. The relevance score between a song and a tag is given by the sum of the scores in both lists, plus the tag score for any synonym or other wild matches of the tag in both lists [1]. Social tag scores are then mapped to the equivalent class in our semantic description. If no gathered tag for a given song belonged to the semantic space, the semantic description is represented by a uniform distribution, where all the tags share the same score. This lead to a very sparse and noisy description, which is useful to test the effectiveness of our approach.

We addressed these descriptions with two different evaluations, although they could be combined together in a single richer semantic description [1].

3.3 Tag-based Retrieval

In this first experiment, the model is queried using a tag; a semantic concept is provided to the system, and the goal is to rank all the songs according to their relationships with that term. Metrics are then averaged through all the terms in the vocabulary. Retrieval performances are measured by finding the positions, along the ranking list, of the documents annotated with the considered tag in the ground truth. HMM-based retrieval is compared with the retrieval performed by simply ranking the songs according to their affinity value for that tag. Results are reported in Table 1, considering both types of semantic description.

Semantic	Model	P3	P5	P10	MAP
SML	HMM	**0.516**	**0.488**	**0.452**	**0.361**
	Tag	0.419	0.431	0.405	0.332
Last.fm	HMM	**0.347**	**0.331**	**0.268**	**0.225**
	Tag	0.303	0.297	0.218	0.207

Table 1. Results of the tag-based retrieval experiments.

As it can be seen, HMM-based retrieval clearly outperforms the retrieval based on a single tag, with a major improvement in the quality at the top of the ranking list. On the other hand, retrieval along the full list tends to decrease its effectiveness, as it can be inferred by the lower improvement achieved by MAP. This is probably due to the problem, discussed in Section 2.2, of HMMs generally poor at capturing long-range correlations between the observed variables. Still we believe that the most important aspect to consider in a retrieval system is the quality on the top of the ranking list. Results based on Last.fm tags tend to have lower performances in terms of absolute values. This likely depends on the fact that the semantic descriptions are rather sparse and noisy and that sometimes songs were represented through a uniform distribution.

3.4 Seed Song Retrieval

In this experiment, retrieval is carried out by submitting to the system 50 randomly selected seed songs and considering the sequence of states highlighted by the optimal path as a ranking list of retrieved documents. A ground truth, against which retrieval results are compared, has been created for each query song by selecting the 30 most similar songs according to their human-based annotations. Semantic similarity has been computed using an application of the KL divergence to the set of tags for each pair of songs.

We compare different approaches: the HMM-based retrieval, a direct content-based retrieval where songs have been ranked according to their acoustic similarity with the seed ("Content"), a semantic similarity measured as KL divergence between the semantic descriptions of the seed song and each document in the collection ("Tags"), and a linear combination between the two distances ("LinComb").

As it can be seen from the results reported in Table 2, even in this case the proposed model leads to outperforming results; the same consideration reported in Section 3.3 can be extend to the current evaluation. The only different aspect is that, in this case, the Last.fm tags better quantize the similarity relationships among songs; thus, the absolute values of the metrics is not very different between the

[3] http://ws.audioscrobbler.com/2.0/

Semantic	Model	P3	P5	P10	MAP
SML	Tag	0.266	0.270	0.246	0.211
	Content	0.237	0.234	0.236	0.187
	LinComb	0.280	0.278	0.244	0.204
	HMM	**0.295**	**0.288**	**0.258**	**0.225**
Last.fm	Tag	0.273	0.272	0.262	0.191
	Content	0.237	0.234	0.236	0.187
	LinComb	**0.305**	0.292	0.262	0.198
	HMM	0.304	**0.299**	**0.284**	**0.219**

Table 2. Results of the song-based retrieval experiments.

two semantic representations.

4. CONCLUSIONS

We introduce a novel methodology that represents a music collection through an hidden Markov model with the purpose to build a music retrieval system that combines content-based acoustic similarity and context-aware semantic descriptions. In the model, each state represents a song, transitions probabilities depend on acoustic similarity and observation probabilities represent semantic descriptions. An application of the Viterbi algorithm allows us to create paths across the model, which provides a ranking list of the songs. This approach represents an application of cross-domain retrieval combining audio content and text for item-based retrieval. It is important to note that the approach can be generalized also to other multimedia tasks where content can be combined with context, such as video or image retrieval.

Some issues are still open and will be addressed in future work. First of all, evaluation tested only the effectiveness of the model; scalability needs to be evaluated with a larger collection, in terms of number of songs and tags. Moreover, future research will be also devoted to the analysis of the effects introduced by different content descriptors and similarity measures. Finally, the extension to other music retrieval tasks, such as music recommendation and playlist generation, will be explored.

5. REFERENCES

[1] L. Barrington, G. Lanckriet, D. Turnbull, and M. Yazdani: "Combining audio content and social context for semantic music discovery," *Proc. of ACM SIGIR*, pp. 387–394, 2009.

[2] L. Barrington, R. Oda, and G. Lanckriet: "Smarter than Genius? Human evaluation of music recommender systems," *Proc. of ISMIR*, pp. 357–362, 2009.

[3] C.M. Bishop: *Pattern Recognition and Machine Learning*, Springer, 2006.

[4] M. Hoffman, D. Blei, and P. Cook: "Content-based musical similarity computation using the hierarchical Dirichlet process," *Proc. of ISMIR*, pp. 349–354, 2008.

[5] M. Khadkevich, and M. Omologo: "Use of Hidden Markov Models and Factored Language Models for Automatic Chord Recognition," *Proc. of ISMIR*, pp. 561–566, 2009.

[6] S. Kullback, and R.A. Leibler: "On information and sufficiency," *Annals of Mathematical Statistics*, Vol. 2, No. 1, pp. 79-86, 1951.

[7] M. Mandel, and D. Ellis: "Song-level features and support vector machines for music classification," *Proc. of ISMIR*, pp. 594–599, 2005.

[8] C.D. Manning, P. Raghavan, and H. Schtze: *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[9] B. McFee, and G. Lanckriet: "Heterogenous embedding for subjective artist similarity," *Proc. of ISMIR*, pp. 513–518, 2009.

[10] R. Miotto, and N. Orio: "A methodology for the segmentation and identification of music works," *Proc. of ISMIR*, pp. 273–278, 2007.

[11] N. Montecchio, and N. Orio: "A discrete filter bank approach to Audio to Score matching for polyphonic music," *Proc. of ISMIR*, pp. 495–500, 2009.

[12] S.R. Ness, A. Theocharis, G. Tzanetakis, and L.G. Martins: "Improving automatic music tag annotation using stacked generalization of probabilistic SVM outputs," *Proc. of ACM MULTIMEDIA*, pp. 705–708, 2009.

[13] L.R. Rabiner: "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. of the IEEE*, Vol. 77, No. 2, pp. 257–286, 1989.

[14] C. Raphael: "Automatic segmentation of acoustic musical signals using hidden Markov models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, No. 4, pp. 360–370, 1999.

[15] J. Shifrin, B. Pardo, C. Meek, and W. Birmingham: "HMM-Based Musical Query Retrieval," *Proc. of ACM/IEEE JCDL*, pp. 295–300, 2002.

[16] M. Slaney, K. Weinberger, and W. White. Learning: "Learning a metric for music similarity," *Proc. of ISMIR*, pp. 313–138, 2008.

[17] D. Turnbull, L. Barrington, and G. Lanckriet: "Five approaches to collecting tags for music," *Proc. of ISMIR*, pp. 225–230, 2008.

[18] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet: "Semantic annotation and retrieval of music and sound effects," *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 16, No. 2, pp. 467–476, 2008.

A PROBABILISTIC SUBSPACE MODEL FOR MULTI-INSTRUMENT POLYPHONIC TRANSCRIPTION

Graham Grindlay
LabROSA, Dept. of Electrical Engineering
Columbia University
grindlay@ee.columbia.edu

Daniel P.W. Ellis
LabROSA, Dept. of Electrical Engineering
Columbia University
dpwe@ee.columbia.edu

ABSTRACT

In this paper we present a general probabilistic model suitable for transcribing single-channel audio recordings containing multiple polyphonic sources. Our system requires no prior knowledge of the instruments in the mixture, although it can benefit from this information if available. In contrast to many existing polyphonic transcription systems, our approach explicitly models the individual instruments and is thereby able to assign detected notes to their respective sources. We use a set of training instruments to learn a model space which is then used during transcription to constrain the properties of models fit to the target mixture. In addition, we encourage model sparsity using a simple approach related to tempering.

We evaluate our method on both recorded and synthesized two-instrument mixtures, obtaining average frame-level F-measures of up to 0.60 for synthesized audio and 0.53 for recorded audio. If knowledge of the instrument types in the mixture is available, we can increase these measures to 0.68 and 0.58, respectively, by initializing the model with parameters from similar instruments.

1. INTRODUCTION

Transcribing a piece of music from audio to symbolic form remains one of the most challenging problems in music information retrieval. Different variants of the problem can be defined according to the number of instruments present in the mixture and the degree of polyphony. Much research has been conducted on the case where the recording contains only a single (monophonic) instrument and reliable approaches to pitch estimation in this case have been developed [3]. However, when polyphony is introduced the problem becomes far more difficult as note harmonics often overlap and interfere with one another. Although there are a number of note properties that are relevant to polyphonic transcription, to date most research has focused on pitch, note onset time, and note offset time, while the problem of assigning notes to their source instruments has received substantially less attention. Determining the source of a note is not only important in its own right, but it is likely to improve overall transcription accuracy by helping to reduce cross-source interference. In order to distinguish between different instruments, we might wish to employ instrument-specific models. However, in general, we do not have access to the exact source models and so must estimate them directly from the mixture. This unsupervised learning problem is particularly difficult when only a single observation channel is available.

Non-negative Matrix Factorization (NMF) [8] has been shown to be a useful approach to single-channel music transcription [10]. The algorithm is typically applied to the magnitude spectrum of the target mixture, V, for which it yields a factorization $V \approx WH$ where W corresponds to a set of spectral basis vectors and H corresponds to the set of activation vectors over time. There are, however, several issues that arise when using NMF for unsupervised transcription. First, it is unclear how to determine the number of basis vectors required. If we use too few, a single basis vector may be forced to represent multiple notes, while if we use too many some basis vectors may have unclear interpretations. Even if we manage to choose the correct number of bases, we still face the problem of determining the mapping between bases and pitches as the basis order is typically arbitrary. Second, although this framework is capable of separating notes from distinct instruments as individual columns of W (and corresponding rows of H), there is no simple solution to the task of organizing these individual columns into coherent blocks corresponding to particular instruments.

Supervised transcription can be performed when W is known *a priori*. In this case, we know the ordering of the basis vectors and therefore how to partition H by source. However, we do not usually have access to this information and must therefore use some additional knowledge. One approach, which has been explored in several recent papers, is to impose constraints on the solution of W or its equivalent. Virtanen and Klapuri use a source-filter model to constrain the basis vectors to be formed from source spectra and filter activations [13]. Vincent et. al impose harmonicity constraints on the basis vectors by modeling them as combinations of narrow-band spectra [12]. In prior work, we proposed the *Subspace NMF* algorithm which learns a model parameter subspace from training examples and then constrains W to lie in this subspace [5].

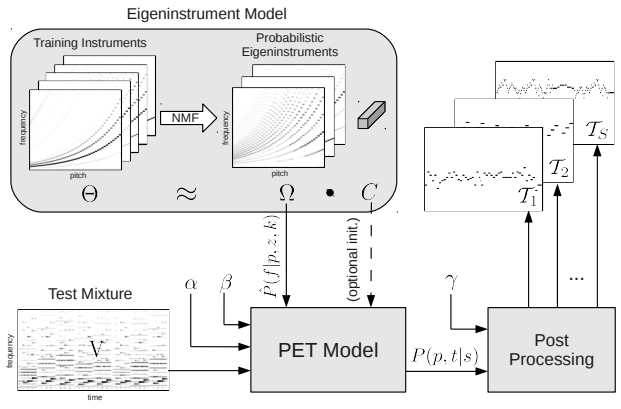


Figure 1. Illustration of the *Probabilistic Eigeninstrument Transcription* (PET) system. First, a set of training instruments are used to derive the eigeninstruments. These are then used by the PET model to learn the probability distribution $P(p,t|s)$, which is post-processed into source-specific binary transcriptions, $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_S$.

Recently, it has been shown [4, 9] that NMF is very closely related to *Probabilistic Latent Semantic Analysis* (PLSA) [6]. In this paper, we extend the *Subspace NMF* algorithm to a probabilistic setting in which we explicitly model the source probabilities, allow for multi-component note models, and use sparsity constraints to improve separation and transcription accuracy. The new approach requires no prior knowledge about the target mixture other than the number of instruments present. If, however, information about the instrument types is available, it can be used to seed the model and improve transcription accuracy.

Although we do not discuss the details here due to a lack of space, we note that our system effectively performs instrument-level source-separation as a part of the transcription process: once the model parameters have been solved for, individual sources can be reconstructed in a straightforward manner.

2. METHOD

Our system is based on the assumption that a suitably-normalized magnitude spectrogram, V, can be modeled as a joint distribution over time and frequency, $P(f,t)$. This quantity can be factored into a frame probability $P(t)$, which can be computed directly from the observed data, and a conditional distribution over frequency bins $P(f|t)$; spectrogram frames are treated as repeated draws from an underlying random process characterized by $P(f|t)$. We can model this distribution with a mixture of latent factors as follows:

$$P(f,t) = P(t)P(f|t) = P(t)\sum_z P(f|z)P(z|t) \quad (1)$$

Note that when there is only a single latent variable z this is the same as the PLSA model and is effectively identical to NMF. The latent variable framework, however, makes it much more straightforward to introduce additional parameters and constraints.

Suppose now that we wish to model a mixture of S instrument sources, where each source has P possible pitches, and each pitch is represented by a set of Z components. We can extend the model described by (1) to accommodate these parameters as follows:

$$\tilde{P}(f|t) = \sum_{s,p,z} P(f|p,z,s)P(z|s,p,t)P(s|p,t)P(p|t) \quad (2)$$

where we have used the notation $\tilde{P}(f|t)$ to denote the fact that our model reconstruction approximates the true distribution, $P(f|t)$. Notice that we have chosen to factor the distribution such that the source probability depends on pitch and time. Intuitively, this may seem odd as we might expect the generative process to first draw a source and then a pitch conditioned on that source. The reason for this factorization has to do with the type of sparsity constraints that we wish to impose on the model. This is discussed more fully in Section 2.2.2.

2.1 Instrument Models

$P(f|p,z,s)$ represents the instrument models that we are trying to fit to the data. However, as discussed in Section 1, we usually don't have access to the exact models that produced the mixture and a blind parameter search is highly under-constrained. The solution proposed in [5], which we extend here, is to model the instruments as mixtures of basis models or "eigeninstruments". This approach is similar in spirit to the eigenvoice technique used in speech recognition [7].

Suppose that we have a set of instruments models $\mathcal{M}$ for use in training. Each of these models $\mathcal{M}_i \in \mathcal{M}$ has FPZ parameters, which we concatenate into a super-vector, $\mathbf{m}_i$. These super-vectors are then stacked together into a matrix, Θ, and NMF with some rank K is used to find $\Theta \approx \Omega C$. [1] The set of coefficient vectors, C, is typically discarded at this point, although it can be used to initialize the full transcription system as well (see Section 3.4). The K basis vectors in Ω represent the eigeninstruments. Each of these vectors is reshaped to the F-by-P-by-Z model size to form the eigeninstrument distribution, $\hat{P}(f|p,z,k)$. Mixtures of this distribution can now be used to model new instruments as follows:

$$P(f|p,z,s) = \sum_k \hat{P}(f|p,z,k)P(k|s) \quad (3)$$

where $P(k|s)$ represents an instrument-specific distribution over eigeninstruments. This model reduces the size of the parameter space for each source instrument in the mixture from FPZ, which is typically tens of thousands, to K which is typically between 10 and 100. Of course the quality of this parametrization depends on how well the eigeninstrument basis spans the true instrument parameter space, but assuming a sufficient variety of training instruments are used, we can expect good coverage.

[1] Some care has to be taken to ensure that the bases in Ω are properly normalized so that each section of F entries sums to 1, but so long as this requirement is met, any decomposition that yields non-negative basis vectors can be used.

2.2 Transcription Model

We are now ready to present the full transcription model proposed in this paper, which we refer to as *Probabilistic Eigeninstrument Transcription* (PET) and is illustrated in Figure 1. Combining the probabilistic model in (2) and the eigeninstrument model in (3), we arrive at the following:

$$\tilde{P}(f|t) = \sum_{s,p,z,k} \hat{P}(f|p,z,k)P(k|s)P(z|s,p,t)P(s|p,t)P(p|t) \tag{4}$$

Once we have solved for the model parameters, we calculate the joint distribution over pitch and time conditional on source:

$$P(p,t|s) = \frac{P(s|p,t)P(p|t)P(t)}{\sum_{p,t} P(s|p,t)P(p|t)P(t)} \tag{5}$$

This distribution represents the transcription of source s, but still needs to be post-processed to a binary pianoroll representation so that it can be compared with ground truth data. This is done using a simple threshold γ (see Section 3.3). We refer to the final pianoroll transcription of source s as $\mathcal{T}_s$.

2.2.1 Update Equations

We solve for the parameters in (4) using the Expectation-Maximization algorithm. This involves iterating between two update steps until convergence. In the first (expectation) step, we calculate the posterior distribution over the hidden variables s, p, z, and k, for each time-frequency point given the current estimates of the model parameters:

$$P(s,p,z,k|f,t) = \frac{\hat{P}(f|p,z,k)P(k|s)P(z|s,p,t)P(s|p,t)P(p|t)}{\tilde{P}(f|t)} \tag{6}$$

In the second (maximization) step, we use this posterior to maximize the expected log-likelihood of the model given the data:

$$\mathcal{L} = \sum_{f,t} V_{f,t} \log\left(P(t)\tilde{P}(f|t)\right) \tag{7}$$

where $V_{f,t}$ are values from our original spectrogram. This results in the following update equations:

$$P(k|s) = \frac{\sum_{f,t,z} P(s,p,z,k|f,t)V_{f,t}}{\sum_{f,k,t,z} P(s,p,z,k|f,t)V_{f,t}} \tag{8}$$

$$P(z|s,p,t) = \frac{\sum_{f,k} P(s,p,z,k|f,t)V_{f,t}}{\sum_{f,k,z} P(s,p,z,k|f,t)V_{f,t}} \tag{9}$$

$$P(s|p,t) = \frac{\sum_{f,k,z} P(s,p,z,k|f,t)V_{f,t}}{\sum_{f,k,s,z} P(s,p,z,k|f,t)V_{f,t}} \tag{10}$$

$$P(p|t) = \frac{\sum_{f,k,s,z} P(s,p,z,k|f,t)V_{f,t}}{\sum_{f,k,p,s,z} P(s,p,z,k|f,t)V_{f,t}} \tag{11}$$

2.2.2 Sparsity

The update equations given in Section 2.2.1 represent a maximum-likelihood solution to the model. However, in practice it can be advantageous to introduce additional constraints. The idea of parameter sparsity has proved to be useful for a number of audio-related tasks [1, 11]. For multi-instrument transcription, there are several ways in which it might make sense to constrain the model solution in this way. First, it is reasonable to expect that if pitch p is active at time t, then only a small fraction of the instrument sources are responsible for it. This belief can be encoded in the form of a sparsity prior on the distribution $P(s|p,t)$. Similarly, we generally expect that only a few pitches are active in each time frame, which implies a sparsity constraint on $P(p|t)$.

One way of encouraging sparsity in probabilistic models is through the use of the *entropic prior* [2]. This technique uses an exponentiated negative-entropy term as a prior on parameter distributions. Although it can yield good results, the solution to the maximization step is complicated, as it involves solving a system of transcendental equations. As an alternative, we have found that simply modifying the maximization steps in (10) and (11) as follows gives good results:

$$P(s|p,t) = \frac{\left[\sum_{f,k,z} P(s,p,z,k|f,t)V_{f,t}\right]^{\alpha}}{\sum_{s}\left[\sum_{f,k,z} P(s,p,z,k|f,t)V_{f,t}\right]^{\alpha}} \tag{12}$$

$$P(p|t) = \frac{\left[\sum_{f,k,s,z} P(s,p,z,k|f,t)V_{f,t}\right]^{\beta}}{\sum_{p}\left[\sum_{f,k,s,z} P(s,p,z,k|f,t)V_{f,t}\right]^{\beta}} \tag{13}$$

When α and β are less than 1, this is closely related to the *Tempered EM* algorithm used in PLSA [6]. However, it is clear that when α and β are greater than 1, the $P(s|p,t)$ and $P(p|t)$ distributions are "sharpened", thus decreasing their entropies and encouraging sparsity.

3. EVALUATION

3.1 Data

Two data sets were used in our experiments, one containing both synthesized and recorded audio and the other containing just synthesized audio. There are 15 tracks, 3256 notes, and 18843 frames in total. The specific properties of the data sets are summarized in Table 1. All tracks had two instrument sources, although the actual instruments varied. For the synthetic tracks, the MIDI versions were synthesized at an 8kHz sampling rate using *timidity* and the SGM V2.01 soundfont. A 1024-point STFT with 96ms window and 24ms hop was then taken and the magnitude spectrogram retained.

The first data set is based on a subset of the woodwind data supplied for the MIREX Multiple Fundamental Frequency Estimation and Tracking task. [2] The first 21 sec-

[2] http://www.music-ir.org/mirex/2009/index.php/Multiple_Fundamental_Frequency_Estimation_&_Tracking

	Type	# Tracks	# Notes	# Frames
Woodwind	S/R	6	1266	5424
Bach	S	3	724	7995

Table 1. Summary of the two data sets used. S and R denote synthesized and recorded, respectively.

onds from the bassoon, clarinet, oboe, and flute tracks were manually transcribed. These instrument tracks were then combined in all 6 possible pairings. It is important to note that this data is taken from the MIREX *development* set and that the primary test data is not publicly available. In addition, most authors of other transcription systems do not report results on the development data, making comparisons difficult.

The second data set is comprised of three pieces by J.S. Bach arranged as duets. The pieces are: *Herz und Mund und Tat und Leben* (BWV 147) for acoustic bass and piccolo, *Ich steh mit einem Fuß im Grabe* (BWV 156) for tuba and piano, and roughly the first half of *Wachet auf, ruft uns die Stimme* (BWV 140) for cello and flute. We chose instruments that were, for the most part, different from those used in the woodwind data set while also trying to keep the instrumentation as appropriate as possible.

3.2 Instrument Models

We used a set of 33 instruments of varying types to derive our instrument model. This included a roughly equal proportion of keyboard, plucked string, bowed, and wind instruments. The instrument models were generated with *timidity*, but in order to keep the tests with synthesized audio as fair as possible, a different soundfont (Papelmedia Final SF2 XXL) was used. [3] Each instrument model consisted of 58 pitches (C2-A6#), which were built as follows: notes of duration 1s were synthesized at an 8kHz sampling rate, using velocities 40, 80, and 100. A 1024-point STFT was taken of each, and the magnitude spectra were averaged across velocities to make the model more robust to differences in loudness. The models were then normalized so that the frequency components (spectrogram rows) summed to 1 for each pitch. Next, NMF with rank Z (the desired number of components per pitch) was run on this result with H initialized to a heavy main diagonal structure. This encouraged the ordering of the bases to be "left-to-right".

One potential issue with this approach has to do with the differences in the natural playing ranges of the instruments. For example, a violin generally cannot play below G3, although our model includes notes below this. Therefore, we masked out (i.e. set to 0) the parameters of the notes outside the playing range of each instrument used in training. Then, as described in Section 2.1, the instrument models were stacked into super vector form and NMF with a rank of $K = 30$ (chosen empirically) was run to find the instrument bases, Ω. These bases were then unstacked to form the eigeninstruments, $\hat{P}(f|p, z, k)$.

[3] http://www.papelmedia.de/english/index.htm

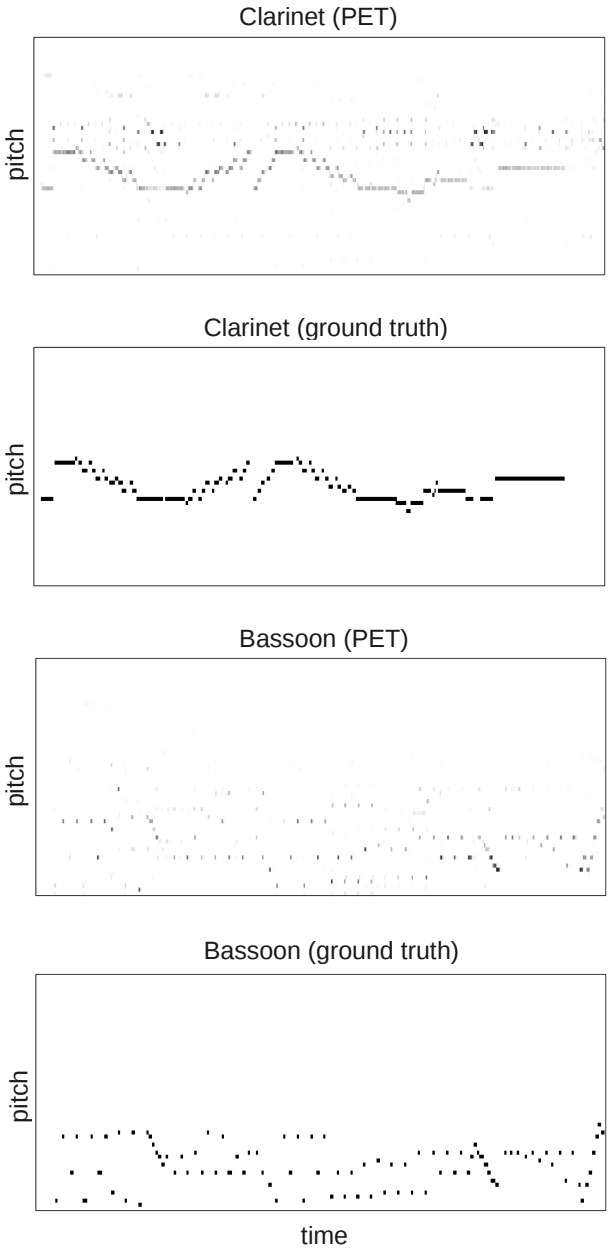


Figure 2. Example PET ($\beta = 2$) output distribution $P(p, t|s)$ and ground truth data for the bassoon-clarinet mixture from the recorded woodwind data set.

In preliminary experiments, we did not find a significant advantage to values of $Z > 1$ and so the full set of experiments presented below was carried out with only a single component per pitch.

3.3 Metrics

We evaluate our method using precision ($\mathcal{P}$), recall ($\mathcal{R}$), and F-measure ($\mathcal{F}$) on both the frame and note levels. Note that each reported metric is an average over sources. In addition, because the order of the sources in $P(p, t|s)$ is arbitrary, we compute sets of metrics for all possible permutations (two in our experiments since there are two sources) and report the set with the best frame-level F-measure.

When computing the note-level metrics, we consider a note onset to be correct if it falls within +/- 50ms of the ground truth onset. At present, we don't consider offsets for the note-level evaluation, although this information is reflected in the frame-level metrics.

The threshold γ used to convert $P(p,t|s)$ to a binary pianoroll was determined empirically for each algorithm variant and each data set. This was done by computing the threshold that maximized the average frame-level F-measure across tracks in the data set.

3.4 Experiments

We evaluated several variations of our algorithm so as to explore the effects of sparsity and to assess the performance of the eigeninstrument model. For each of the three data sets, we computed the frame and note metrics using the three variants of the PET model: PET without sparsity, PET with sparsity on the instruments given the pitches $P(s|p,t)$ ($\alpha = 2$), and PET with sparsity on the pitches at a given time $P(p|t)$ ($\beta = 2$). In these cases, all parameters were initialized randomly and the algorithm was run for 100 iterations.

Although we are primarily interested in blind transcription (i.e. no prior knowledge of the instruments present in the mixture), it is interesting to examine cases where more information is available as these can provide upper-bounds on performance. First, consider the case where we know the instrument types present in the mixture. For the synthetic data, we have access not only to the instrument types, but also to the oracle models for these instruments. In this case we hold $P(f|p,s,z)$ fixed and solve the basic model given in (2). The same can be done with the recorded data, except that we don't have oracle models for these recordings. Instead, we can just use the appropriate instrument models from the training set $\mathcal{M}$ as approximations. This case, which we refer to as "fixed" in the experimental results, represents a semi-supervised version of the PET system.

We might also consider using the instrument models $\mathcal{M}$ that we used in eigeninstrument training in order to initialize the PET model in the hope that the system will be able to further optimize their settings. We can do this by taking the appropriate eigeninstrument coefficient vectors $\mathbf{c}_s$ and using them to initialize $P(k|s)$. Intuitively, we are trying to start the PET model in the correct "neighborhood" of eigeninstrument space. These results are denoted "init".

Finally, as a baseline comparison, we consider generic NMF-based transcription (with generalized KL divergence as a cost function) where the instrument models (submatrices of W) have been initialized with a generic model defined as the average of the instrument models in the training set.

3.5 Results

The results of our approach are summarized in Tables 2–4. As a general observation, we can see that the sparsity factors have helped improve model performance in almost all cases, although different data sets benefit in different ways.

	Frame			Note		
	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$
PET	0.56	0.64	0.56	0.42	0.73	0.51
$PET_{\alpha=2}$	0.60	0.61	0.60	0.46	0.73	0.56
$PET_{\beta=2}$	0.57	0.64	0.56	0.51	0.79	0.58
PET_{init}	0.71	0.68	0.68	0.64	0.86	0.71
PET_{oracle}	0.84	0.84	0.84	0.82	0.93	0.87
NMF	0.34	0.48	0.39	0.19	0.59	0.29

Table 2. Results for the synthetic woodwind data set. All values are averages across sources and tracks.

	Frame			Note		
	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$
PET	0.52	0.52	0.50	0.41	0.73	0.50
$PET_{\alpha=2}$	0.49	0.57	0.51	0.41	0.78	0.51
$PET_{\beta=2}$	0.58	0.53	0.53	0.46	0.72	0.55
PET_{init}	0.60	0.60	0.58	0.48	0.82	0.58
PET_{fixed}	0.57	0.58	0.55	0.45	0.77	0.54
NMF	0.35	0.55	0.42	0.27	0.77	0.38

Table 3. Results for the recorded woodwind data set. All values are averages across sources and tracks.

For the synthetic woodwind data set, sparsity on sources, $P(s|p,t)$, increased the average F-measure on the frame-level, but at the note-level, sparsity on pitches, $P(p|t)$, had a larger impact. For the recorded woodwind data, sparsity on $P(p|t)$ benefited both frame and note-level F-measures the most. With the Bach data, we see that encouraging sparsity in $P(p|t)$ was much more important than it was for $P(s|p,t)$ on both the frame and note-level. In fact, imposing sparsity on $P(s|p,t)$ seems to have actually hurt frame-level performance relative to the non-sparse PET system. This may be explained by the fact that the instrument parts in the Bach pieces tend to be simultaneously active much of the time.

As we would expect, the baseline NMF system performs the worst in all test cases – not surprising given the limited information and lack of constraints. Also unsurprising is the fact that the oracle models are the top-performers on the synthetic data sets. However, notice that the randomly-initialized PET systems perform about

	Frame			Note		
	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$
PET	0.50	0.65	0.54	0.21	0.60	0.30
$PET_{\alpha=2}$	0.50	0.57	0.51	0.22	0.51	0.30
$PET_{\beta=2}$	0.55	0.66	0.59	0.24	0.65	0.34
PET_{init}	0.53	0.58	0.53	0.23	0.50	0.30
PET_{oracle}	0.91	0.85	0.87	0.53	0.83	0.64
NMF	0.36	0.50	0.42	0.09	0.46	0.14

Table 4. Results for the synthetic Bach data set. All values are averages across sources and tracks.

as well as the fixed model on recorded data. This implies that the algorithm was able to discover appropriate model parameters even in the blind case where it had no information about the instrument types in the mixture. It is also noteworthy that the best performing system for the recorded data set is the initialized PET variant. This suggests that, given good initializations, the algorithm was able to further adapt the instrument model parameters to improve the fit to the target mixture.

While the results on both woodwind data sets are relatively consistent across frame and note levels, the Bach data set exhibits a significant discrepancy between the two metrics, with substantially lower note-level scores. This is true even for the oracle model which achieves an average note-level F-measure of 0.64. There are two possible explanations for this. First, recall that our determination of both the optimal threshold γ as well as the order of the sources in $P(p, t|s)$ was based on the average frame-level F-measure. We opted to use frame-level metrics for this task as they are a stricter measure of transcription quality. However, given that the performance is relatively consistent for the woodwind data, it seems more likely that the discrepancy is due to instrumentation. In particular, the algorithms seem to have had difficulty with the soft onsets of the cello part in *Wachet auf, ruft uns die Stimme*.

4. CONCLUSIONS

We have presented a probabilistic model for the challenging problem of multi-instrument polyphonic transcription. Our method makes use of training instruments in order to learn a model parameter subspace that constrains the solutions of new models. Sparsity terms are also introduced to help further constrain the solution. We have shown that this approach can perform reasonably well in the blind transcription setting where no knowledge other than the number of instruments is assumed. In addition, knowledge of the types of instruments in the mixture (information which is relatively easy to obtain) was shown to improve performance significantly over the basic model. Although the experiments presented in this paper only consider two-instrument mixtures, the PET model is general and preliminary tests suggest that it can handle more complex mixtures as well.

There are several areas in which the current system could be improved. First, the thresholding technique that we have used is extremely simple and results could probably be improved significantly through the use of pitch dependent thresholding or more sophisticated classification. Second, and perhaps most importantly, although early experiments did not show a benefit to using multiple components for each pitch, it seems likely that the pitch models could be enriched substantially. Many instruments have complex time-varying structures within each note that would seem to be important for recognition. We are currently exploring ways to incorporate this type of information into our system.

5. ACKNOWLEDGMENTS

This work was supported by the NSF grant IIS-0713334. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

6. REFERENCES

[1] S.A. Abdallah and M.D. Plumbley. Polyphonic music transcription by non-negative sparse coding of power spectra. In *ISMIR*, 2004.

[2] M. Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11(5):1155–1182, 1999.

[3] A. de Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(1917), 2002.

[4] E. Gaussier and C. Goutte. Relation between PLSA and NMF and implications. In *SIGIR*, 2005.

[5] G. Grindlay and D.P.W. Ellis. Multi-voice polyphonic music transcription using eigeninstruments. In *WASPAA*, 2009.

[6] T. Hofmann. Probabilistic latent semantic analysis. In *Uncertainty in AI*, 1999.

[7] R. Kuhn, J. Junqua, P. Nguyen, and N. Niedzielski. Rapid speaker identification in eigenvoice space. *IEEE Transactions on Speech and Audio Processing*, 8(6):695–707, November 2000.

[8] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

[9] M. Shashanka, B. Raj, and P. Smaragdis. Probabilistic latent variable models as non-negative factorizations. *Computational Intelligence and Neuroscience*, 2008, 2008.

[10] P. Smaragdis and J.C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *WASPAA*, 2003.

[11] P. Smaragdis, M. Shashanka, and B. Raj. A sparse nonparametric approach for single channel separation of known sounds. In *NIPS*, 2009.

[12] E. Vincent, N. Bertin, and R. Badeau. Harmonic and inharmonic non-negative matrix factorization for polyphonic pitch transcription. In *ICASSP*, 2008.

[13] T. Virtanen and A. Klapuri. Analysis of polyphonic audio using source-filter model and non-negative matrix factorization. In *NIPS*, 2006.

A SEGMENTATION-BASED TEMPO INDUCTION METHOD

Maxime Le Coz, Helene Lachambre, Lionel Koenig and Regine Andre-Obrecht
IRIT, Universite Paul Sabatier,
118 Route de Narbonne, F-31062 TOULOUSE CEDEX 9
{lecoz,lachambre,koenig,obrecht}@irit.fr

ABSTRACT

The automatized beat detection and localization have been the subject of multiple research in the field of music information retrieval. Most of the methods are based on onset detection. We propose an alternative approach:

Our method is based on the "Forward-Backward segmentation": the segments may be interpreted as attacks, decays, sustains and releases of notes. We process the segment boundaries as a weighted Dirac signal. Three methods devived from its spectral analysis are proposed to find a periodicity which corresponds to the tempo.

The experiments are carried out on a corpus of 100 songs of the RWC database. The performances of our system on this base demonstrate a potential in the use of a " Forward-Backward Segmentation" for temporal information retrieval in musical signals.

1. INTRODUCTION

The automatized beat detection and localization have been the subject of multiple research in the field of music information retrieval. The study of beat is indeed important as the structure of a music piece lies in the beat. Western music uses however different levels in the hierarchy of scale measuring time. We have to distinguish the *tatum* which is "the regular time division that mostly coincides with all note onsets" [3] from the *tactus* which is defined as the rate at which most people would clap their hands when listening to the music [8]. Here, we look for the *tactus*, which will be named tempo and measured in beat per minute (BPM).

Several methods have been suggested in order to extract the tempo information from an audio signal. Most of them use an onset detection method as onset localization carries the temporal structure that leads to the estimation of the tempo. Theses methods use different observation features in order to propose a list of onset positions. They are very dependent on that detection. Dixon's first algoritm [4] uses an energy based detector in order to track the onset positions. Then a clustering is performed on the inter-onset-interval values. Some best clusters are chosen as possible hypothesis. A hypothesis is finally validated with a beat tracking.

In Alonso's algorithm [1], onset positions are deducted by using a time-frequency representation and a differentiator FIR filter to detect sudden changes in the dynamics, timbre or harmonic structure. The tempo is then deduced using either the autocorrelation or spectral product.

Klapuri [9] proposes a more complex way of extracting the onset positions. The loudness differentials in frequency subbands are computed and combined in order to create four accent bands. This aims at detecting harmonic or melodic changes as well as percussive changes. Using comb filter resonators to extract features, and probalistic models, the values of *tatum*, *tactus* and *measure* meter are computed.

Uhle [12] suggests a method based on the segmentation of the signal into long-term segments corresponding to its musical structure (for example, the verses and chorus of a song). The amplitude envelope of logarithmically spaced frequency subbands is computed; its slope signal aims to represent accentuation on the signal. The analysis of an autocorrelation function on 2.5 second segments inside each long-term segment gives the tatum estimator. A larger-scale analysis over 7.5 second segments is then performed in order to give values corresponding to the measure. The local maxima positions of the autocorrelation function are finally compared with a bank of pre-defined patterns in order to define the best value of the tempo on the long term segment.

Dixon [5] has proposed an alternative method to onset calculation. The signal is splitted into 8 frequency bands and autocorrelation is performed on each smoothed and downsampled subband. The three highest peaks of each band are selected and combined in order to determine the final tempo estimation.

Another algorithm is that of Scheirer [10]. This algorithm performs a comb filterbank that seeks for periodically spaced clock pulse that best matches the envelope of 6 frequency subbands.

Tzanetakis [11] suggests a method based on a wavelet transform analysis. This analysis is performed over 3 second signal segments with 50% of overlap. On each segment, amplitude envelope of 5 octave-spaced frequency bands is extracted. Autocorrelation is then computed. Three

kind of autocorrelation analysis are computed in order to estimate the value of the tempo. The first one is the median of highest peak of the sum of the envelopes over every window. The second one returns the median value of the highest peak on each subband and each segment. The last one computes several best peaks from the autocorrelation on the sum of every envelope and then chooses the most frequent value.

Our method is based on the analysis of an automatic segmentation of the signal into quasi-stationary segments : the segments may be interpreted as attacks, decays, sustains and releases of notes. So we propose to process the segment boundaries in order to find a periodicity which would correspond to the tempo.

In section 2, we describe the segmentation used as a front-end, the analysis of this segmentation in the frequency domain and the different methods we use to extract the value of the tempo in BPM. In the last part, we present the results of our experiments on the RWC [6, 7] corpus.

2. METHOD

Our method relies on the detection of quasi-stationnary segments in the audio signal waveform. A frequency analysis of the boundaries is then performed in order to find the most present periodicities and thereby estimate the tempo consequently.

The algorithm is based on three steps :

- Segmentation
- Boundary frequencial analysis
- Tempo extraction

2.1 Segmentation

We segment the signal using the "Forward Backward Divergence" [2]. The signal is assumed to be a sequence of quasi-stationnary units, each one characterized by the following gaussian autoregressive model :

$$\begin{cases} y_n = \sum a_i y_{n-i} + e_n \\ var(e_n) = \sigma_n^2 \end{cases} \quad (1)$$

where y_n is the signal and e_n an uncorrelated zero mean Gaussian sequence.
As the variance σ_n is constant over an unit and equals σ, the model of each area is parametered by the following vector :

$$(A^T, \sigma) = (a_1, ..., a_p, \sigma) \quad (2)$$

The strategy is to detect changes in the parameters, using a distance based on the mutual conditional entropy. A subjective analysis of the segmentation shows a sub note segmentation and the location of attacks, sustains and releases.

For a solo musical sound, the segments of the signal correspond to the different steps of a note. On Figure 1, we present a solo note of trombone. The note is segmented into four parts, which correspond to the attack, the sustain and the release. Note that the attack and decay phases of some notes are often grouped together into a single segment. In such cases, the attack period is too short for the segmentation algorithm as it imposes a minimal length to initiate the autoregressive model.

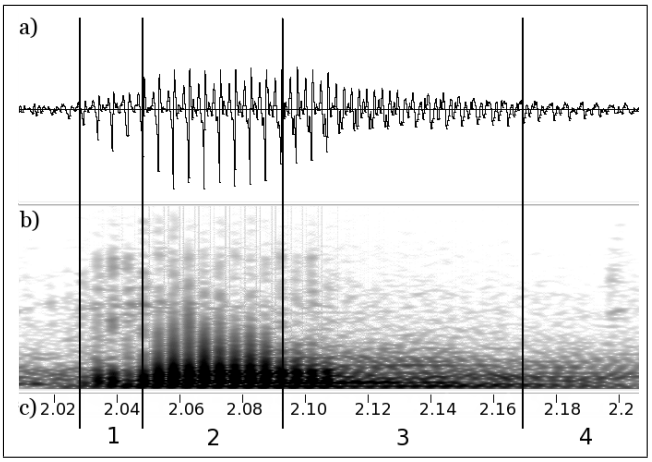


Figure 1. Segmentation of a trombone note. a) Waveform, b) Spectrogram, c) Time. 1) Attack, 2) Sustain, 3 & 4) Release. The vertical lines are the boundaries of the segments. The first boundary correspond to the onset.

As they represent a rupture point of the signal, we assume that onset localizations, containing the tempo information, are included in the list of boundaries time. We therefore focus on positions of the boundaries.

2.2 Boundary Frequencial analysis

The main objective is to find a periodicity in the localization of the boundaries that would be the effect of the song's rythmical pattern. In order to find the periodicity, a signal $b_w(t)$ is created. This signal is a weighted Dirac signal, where each Dirac is positioned at the time of a boundary t_k.

The Diracs are weighted in order to give more influence to the boundaries located at times that are most likely to be onsets. Asuming that at onset times, an increase of energy is observed, each Dirac is weighted by the difference between the energy of the spectrum computed on 20 ms after and before t_k (resp. e_k^+ and e_k^-).

$$w(t_k) = e_k^+ - e_k^- \quad (3)$$

We obtain $b_w(t)$ (see an example on Figure 2) :

$$b_w(t) = \sum_{k=1}^{N} \delta(t - t_k) w(t_k) \quad (4)$$

where N is the count of boundaries, t_k is the time of the k^{th} boundary.

We compute B_w, the Fourier transform of b_w to extract frequency information of this signal.

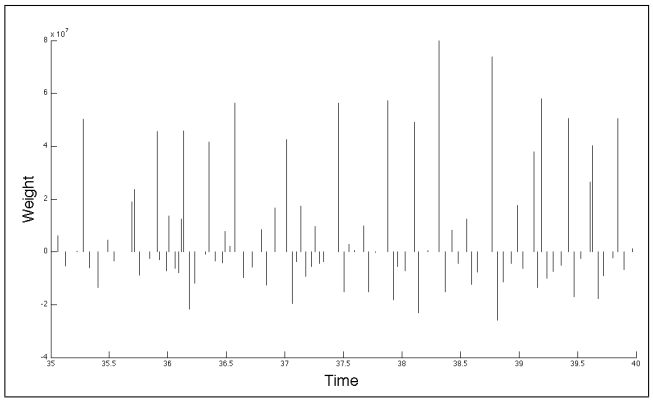


Figure 2. Representation of a $b_w(t)$

The expression of the Fourier transform $B_w(f)$ is :

$$\begin{aligned} B_w(f) &= \int_{\mathbb{R}} \sum_{k=1}^{N} \delta(t - t_k) e^{-2i\pi ft} w(t_k) dt \\ &= \sum_{k=1}^{N} e^{-2i\pi f t_k} w(t_k) \end{aligned} \tag{5}$$

This formula offers the advantage of being fast to calculate.

2.3 Tempo extraction

2.3.1 Spectrum analysis

We analyse the spectrum B_w on the range of frequencies 30 - 400 BPM (an example is given on Figure 3). We find the positions of the highest peaks as a base for each decision.

We then extract the positions and energies of the main peaks in terms of energy. As it is computed over a long time, the peaks of the spectrum are high and narrow, which makes the localization easier.

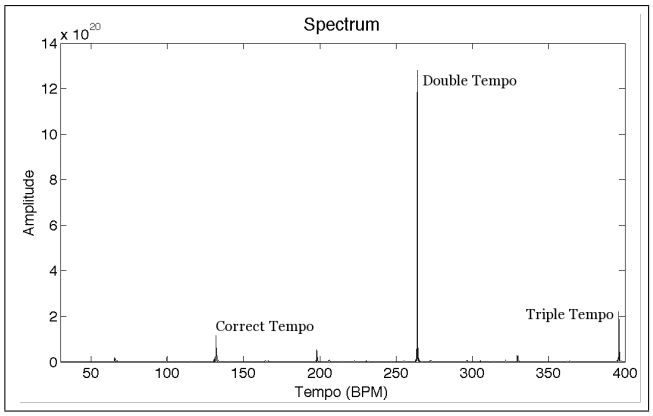


Figure 3. Spectrum $|(B_w(f)|^2$ of a whole song.

This localization is obtained by detecting the local maxima. This algorithm considers a point p and its two direct neighbors. p is a local maxima if

$$\begin{aligned} |B_w(p-1)|^2 < |B_w(p)|^2 \\ |B_w(p+1)|^2 < |B_w(p)|^2 \end{aligned} \tag{6}$$

We then choose several of the highest peaks with the only constraint that the distance between two peaks has to be greater than 3 BPM. Only a few peaks are really higher than others in the spectrum, so we choose to select only the four greatest peaks in terms of energy, the position selected for further peaks would be considered as noise. Let $P = \{p_1\ p_2\ p_3\ p_4\}$ be the list of selected peak positions under the constraint : $|B_w(p_i)|^2 > |B_w(p_{i+1})|^2$. We observe that every selected peak carries information that can be exploited in order to find out the value of the correct tempo. We finally apply a decision algorithm on P to find the tempo.

Two strategies are concidered. The first one looks for the correlation between the length of the segments and each value p in the temporal domain. The second one tries to find the best comb matching the spectrum.

2.3.2 "Inter-Boundaries-Intervals decision"

The first approach is in the temporal domain, and uses the boundaries of the segmentation. Theses boundaries are filtered on their weights in order to keep only the boundaries where a high increase of energy is experienced: we only keep the boundaries with a significant weight. This filtering is computed in order to keep instants which are most likely onset instants. The set I of intervals between each couple of following boundaries is then computed.

For each p_i, we perform the pseudo periods corresponding to 1/4, 1/3, 1/2, 1, 2, and 3 times p_i. These pseudo periods have been chosen as they correspond to the period of half, quarter, eighth and sixteenth note in duple meter or triple meter.

The score $Num(p_i)$ is the number of intervals in I whose durations correspond to one of these pseudo periods.

The estimated tempo $\widehat{p_b}$ is given by :

$$\widehat{p_b} = \underset{p_i, i=1,...,4}{\operatorname{argmax}} (Num(p_i)) \tag{7}$$

2.3.3 "Comb decision"

The second method uses the spectrum and is in frequency domain. This method is based on the first peak p_1, as we assume that it is always significant for the tempo detection. We then consider 7 tempi, which are $\frac{1}{4}p_1$, $\frac{1}{3}p_1$, $\frac{1}{2}p_1$, $2p_1$, $3p_1$ and $4p_1$, as well as p_1 itself, noted tp_i, $i = 1, ..., 7$. We only keep, among this list of tempi, those which are in the range 30 - 240 BPM, assuming that a value outside of these bounds would hardly be considered as the main tempo.

For each tempo value tp_i, we compute the product of the spectrum and a Dirac comb with the 10 harmonic teeth corresponding to the tempo value.

The mean amplitude value of the so filtred spectrum gives a score $Ampl(tp_i)$.

The estimated tempo $\widehat{p_c}$ is given by

$$\widehat{p_c} = \underset{tp_i, i=1,...,7}{\operatorname{argmax}} (Ampl(tp_i)) \tag{8}$$

2.3.4 Combination of the strategies

In order to take advantage of both methods, we propose a combined decision algorithm. Using p_{c1} and p_{c2} the two best tempi returned by the "Comb decision" algorithm, we apply the "Inter-Boundaries-Intervals" strategy to compare the two values $Num(p_{c1})$ and $Num(p_{c2})$.

The tempo with the best Num is chosen as a final decision.

3. EXPERIENCE

3.1 Corpus

We choose to test our method on the part of the RWC database [6, 7] that is BPM-annotated. This corpus has been created in order to provide a benchmark for experimentation on music information retrieval and is now well known and widely used in this research field. It therefore seems interesting to use it in order to facilitate comparisons between our algorithm's results and others. This corpus is a compilation of 100 tracks of Japanese Pop songs. Each song lasts from 2 minutes 50 seconds to 6 minutes 07 seconds.
As the method needs no learning, our experiment protocol consists in applying our algorithm on each full track.

3.2 Experiments

The methods are based on the Forward Backward divergence segmentation: in order to implement this algorithm, we choose to use the parameters defined in [2] for voiced speech signal. No specific adaptation is performed for music.

As previously mentioned, we observe that the highest peak of the spectrums has a strong link with the tempi. Over the 100 tracks computed, the highest peak position is linked with the tempo 98 times: it is located twice on a position corresponding to the half of the ground-truth tempo, 3 times on the correct position, 60 times on the double tempo and 32 times on a position corresponding to 4 times the tempo.

To assess quantitatively each version of our method, we introduce a confident interval : the tempo value is considered as "Correct" if its difference with the ground-truth value at strictly less than 4 BPM. The ratios and multiples are considered good when their distance to 2, 3, 4, 1/3 or 1/2 is strictly less than 0.03.

Two metrics are computed in order to evaluate the accuracy of each method. The first one is the ratio of correctly estimated tempi over the whole corpus.

$$Accuracy_1 = \frac{\#\ of\ correctly\ estimated\ tempi}{L} \quad (9)$$

where L is the number of evaluated tracks.

The second one is more flexible and assumes that the tempi corresponding to half, third double and three time the annotated tempo are correct. This metric is computed taking take into account that tempo value is subjective and can vary from one listener to another.

$$Accuracy_2 = \frac{\#\ of\ correct\ or\ multiple\ tempi}{L} \quad (10)$$

3.2.1 "Inter-Boundaries-Intervals decision"

The filltring of the boundaries involves a threshold: the selectionned boundaries have a weight greater than 10% of the maximum weight among the boundaries. The detailled results of the Inter-Boundaries-Intervals decision are visible in Table 1. The global result are 56 % of $Accuracy1$ and 95% of $Accuracy2$.

Ratios with the correct tempo						
1/2	1	2	4	No link	Acc_1	Acc_2
7	56	28	1	5	56	95

Table 1. "Inter-Boundaries decision Decision" : Number of music tracks in function of the ratios between the estimated tempo and the ground truth value. $Accuracy_1$ and $Accuracy_2$ are deducted.

3.2.2 Comb decision

In order to optimize the results of this method and to be sure to get the peak value on each hypothesis multiple, the returned value is the maximum of 7 equally spaced tempi in a neighborhood of $\pm 1\ BPM$ around each p multiple value. Applying this method to our corpus and returning the best two hypothesis, we observe that the ground-truth tempo is present for 98 of the tracks. The global result of this method, choosing only the best comb as result, is 64% for $Accuracy_1$ and 96% for $Accuracy_2$. The detailled results are visible in Table 2.

Ratios with the correct tempo						
1/2	1	2	3	No link	Acc_1	Acc_2
3	64	29	0	4	64	96

Table 2. "Comb Decision" : Number of music tracks in function of the ratios between the estimated tempo and the ground truth value. $Accuracy_1$ and $Accuracy_2$ are deducted.

3.2.3 Combination of the strategies

As shown in Table 3, the combination of the two previous methods largely improves the results. The results in terms of $Accuracy_1$ is 78% and 93% in terms of $Accuracy_2$.

Ratios with the correct tempo						
1/2	1	2	3	No link	Acc_1	Acc_2
13	78	2	0	7	78	93

Table 3. Percentage of the returned values ratio of the ground truth for the Fusion of the two algorithms

The differences between their results is essentially due to the detection of the "double tempo". This type of error dissapears. The number of serious errors is stable.

3.3 Discussion

The 2004 MIREX evaluation was the last MIREX session which the task of tempo estimation was evaluated. These results were obtained on a corpus of 3199 tempo-annotated files ranging from 2 to 30 seconds, akd divided into three kinds : loops, ballroom music and songs exerpts.

The algorithms evaluated during this campaign are detailed and compared in [8]. The Klapuri's algorithm [9] obtained the best score on this evaluation with an $Accuracy_1$ of 67.29% and an $Accuracy_2$ of 85.01% among the total set of evaluated signals and reaching 91.18% of $Accuracy_2$ on the song's subset.

An exhaustive search for the best combination of five algorithms, using a voting mechanism, has also be computed. The best combination achieved 68% in terms of $Accuracy_1$, whereas the best $Accuracy_2$ reached 86%.

The MIREX corpus and the RWC part we use are different (in particular in terms of length). Nevertheless, our results are comparable and experiments will be realized on short extracts of the songs in order to define the robustness of our method.

4. CONCLUSIONS

In this paper, we presented a tempo estimator based on an automatic segmentation of the signal into quasi-stationnary zones. The use of this segmentation for the tempo induction seems to be rather significant: the spectrum of the Dirac signal derivate from the segmentation shows a predominant value directly linked with the tempo on 98% of our tests. The three methods which exploit this property have good performence. These methods are still rather simple, so we will investigate some potential improvements:

- Some experiments will be realized in order to evaluate the sensitiveness of our method to the use of short extract. Good results would allow the use of this method on slipping windows of few dozens of second. Such treatment could be realized in order to detect changes in the tempo.

- The use of the phase of $B_w(p)$ seems promissing for the developpement of a precise onset localizator.

5. REFERENCES

[1] M. Alonso, B. David, and G. Richard. Tempo and beat estimation of music signals. In *Proc. Int. Conf. Music Information Retrieval*, pages 158–163, 2004.

[2] R. André-Obrecht. A new statistical approach for the automatic segmentation of continuous speech signals. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 36(1):29–40, 1988.

[3] J. Bilmes. Timing is of the essence: Perceptual and computational techniques for representing, learning, and reproducing expressive timing in percussive rhythm. Master's thesis, MIT, Cambridge, Mass., USA, 1993.

[4] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 2001.

[5] S. Dixon, E. Pampalk, and G. Widmer. Classification of dance music by periodicity patterns. In *Proc. Int. Conf. Music Information Retrieval*, pages 159–165, 2003.

[6] M. Goto. Development of the RWC music database. In *Proceedings of the 18 th International Congress on Acoustics (ICA 2004)*, pages 553–556, 2004.

[7] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical, and jazz music databases. In *Proc. 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pages 287–288, 2002.

[8] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Trans. on Audio, Speech, and Language Processing*, 14(5):1832 – 1844, september 2006.

[9] A. Klapuri, A. Eronen, and J. Astola. Analysis of the meter of accoustic musical signals. *IEEE Trans. on Audio, Speech, and Language Processing*, 14(1):342–355, 2006.

[10] E. Scheirer. Tempo and beat analysis of acoustic music signals. *Journal of the Acoustical Society of America*, 104:588–601, 1998.

[11] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Trans. on Speech and Audio Processing*, 10(5):293–302, 2002.

[12] C. Uhle, J. Rohden, M. Cremer, and J. Herre. Low complexity musical meter estimation from polyphonic music. In *Proc. AES 25th International Conference*, pages 63–68, June 2004.

AMUSE (ADVANCED MUSIC EXPLORER) - A MULTITOOL FRAMEWORK FOR MUSIC DATA ANALYSIS

Igor Vatolkin
Chair of Algorithm Engineering, TU Dortmund
igor.vatolkin@udo.edu

Wolfgang Theimer
Research in Motion, Bochum
wolfgang.theimer@ieee.org

Martin Botteck
martin.botteck@ieee.org

ABSTRACT

A large variety of research tools is available now for music information retrieval tasks. In this paper we present a further framework which aims to facilitate the interaction between these applications. Since the available tools are very different in target domain, range of available methods, learning efforts, installation and runtime characteristics etc., it is not easy to find software which is optimal for certain research goals. Another problematic issue is that many incompatible data formats exist, so it is not always possible to use output from one tool just as input for another one. At first we describe some of the available projects and outline our motivation starting the development of AMUSE framework for audio data analysis. Requirements and application purposes are given. The structure of our framework is introduced in detail and the information for efficient application is provided. Finally we discuss several ideas for further work.

1. INTRODUCTION AND MOTIVATION FOR A NEW FRAMEWORK

During the recent years more and more scientific tools for music information retrieval and related research areas have been developed. To name just a few, Marsyas is one of the oldest available MIR projects for different analysis and synthesis tasks [12]. jMIR tools refer to different applications from feature extraction to data mining methods [7]. MusicMiner established new navigation techniques for large music collections based on self-organized maps and three-dimensional landscapes [9]. MIR Toolbox includes a large number of different adjustable Matlab functions for extraction of features from time signal characteristics to complex harmony and major/minor key descriptors [4]. The Chroma Toolbox provides advanced features related to chroma and pitch [10]. RapidMiner is aimed to solve a wide range of different data mining tasks (not only for music and audio classification domain) and supports numerous methods [8].

The motivation to start our own project developing a new software framework arose after the discussion and definition of several promising MIR applications and in-depth comparison of different above mentioned and further tools. Typically each existing tool has several main focus points as well as certain application advantages and disadvantages. Therefore the choice of software to use depends strongly on the defined scenario. Possible three examples could be: If the researcher develops own classification methods, it may be interesting for her/him to gather many available audio features from the corresponding tools. If the aim is to run advanced low-level signal analysis and create the features by himself, the researcher would create this code and use some ready products, e.g. WEKA toolbox [13] for the revision, how well the novel features are suited for audio classification. The last example is that for different reasons multi-objective evaluation of algorithmic chain can be significant. Here the focus point is to collect different metrics (confusion matrix-based measures, runtime and disc space demands etc.) and to run multi-objective optimization algorithms searching for the best tradeoff between several solutions.

Another aspect is that many tools which are very helpful for MIR research are either too specific and concentrate on limited audio retrieval domains, e.g. Chroma Toolbox (so we may need several of them!) or are on the other side too powerful and generic (e.g. RapidMiner) and it is not easy to create the appropriate solution. The input and output data formats differ from tool to tool and even the support of the WEKA ARFF format does not mean, that the written attributes are the same. Therefore it made sense for us to develop a multi-tool framework, which provides own data interchange formats and own evaluation methods. The integration of further tools and also the extension of methods with own code belonged to requirements. Further consideration was that some fields had been underrepresented in many available tools and it was important for us to emphasize them as independent tasks in music retrieval chain: feature processing is an intermediate step between the extracted raw features and ready classifier input. The way how the labeled vector is built from the frame-based signal features for training of classification models can be very different and has a strong impact on classification result quality. Another issue is the inclusion of optimization methods, e.g. heuristics, to search for the best parametrization of the algorithm chain, for example the estimation of

satisfactory time frame size or pruning of feature set.

The current version of AMUSE made possible to run different large experiment studies including feature extraction from several tools, processing with many methods, classification for user-defined music categories and also optimization of some parameters [3, 14, 15].

2. FRAMEWORK STRUCTURE

2.1 Background, Requirements and Functionality

AMUSE (Advanced MUSic Explorer) is a GPL-licensed framework implemented in Java [1] . Therefore the main component can be run on any operating system which supports the Java Runtime Environment. The integrated tools have no usage restrictions with regard to their source codes. If they are not available as Java libraries, executable versions must be provided. In that case it may certainly lead to the dependence on the running operating system.

The AMUSE core provides different functionalities. With own sound processing methods mp3s can be converted to waves. Downsampling and stereo to mono conversion methods are included as well. It is possible to split automaticly the wave files (we had experiences that very long songs supplied to some tools led to memory problems or unacceptable running time). Scalability is supported either using multi-threading on one machine or providing the tasks to grid systems like Sun Grid Engine or LSF Batch. Efficient data set management which directly supports the WEKA ARFF format (well-established for various data mining tasks) and a logger component are integrated.

Several user interfaces are available: Definition and application of tasks can be easily done within a graphical user interface, see Figure 5 for screenshots. In command-line mode AMUSE runs one or more tasks from given configuration files. In loop mode AMUSE is pre-loaded in memory and waits for new tasks by scanning for corresponding configuration files in a task folder.

Project packages are organized in the way so that the core and extendable components are strictly separated. Integration of external tools requires writing of adapter classes which take care of input / output conversion and start these tools as library or by system call. AMUSE plugins allow to create such integrations without changes on the main project and be easily installed and deinstalled.

2.2 Music Retrieval Chain and Integrated Methods

We distinguish between subtasks in a MIR chain. Figure 1 gives a complete overview. The rectangles correspond to AMUSE *tasks* which are run by the related node component. Each task can adhere to the larger number of AMUSE *jobs* which can be calculated on several processing units - e.g. a feature extraction task for a hundred music files can be distributed to several machines as one hundred jobs.

2.2.1 Feature Extraction

Feature extraction provides low-level or high-level numerical descriptors from the audio signal. It can be a complete task (melody extraction) or a part of a longer chain, where audio files are categorized using the extracted features. AMUSE provides a generic mechanism to select the features which must be extracted by external tools. For each tool a so called *base script* must exist which allows to extract all supported features. After the AMUSE extraction task is loaded into memory, some parts of these base scripts are omitted, if the corresponding transforms or features should not be extracted this time.

2.2.2 Feature Processing

Feature processing is an intermediate step between raw extracted features and ready-labeled input for classification. Starting with a matrix of M features over N time frames at the beginning, different methods change this matrix. Some of them extend the dimensionality (e.g. calculating the derivations for all features) or reduce the dimensionality (pruning the features or deselecting time frames using specific information like temporal structure of a song). The last step is the conversion of the feature matrix to a vector which can be labeled for supervised classification. This can be done e.g. by Gaussian, histogram or autoregressive models. Since the source time frames may differ between features, the matrix is automaticly adjusted using the smallest existing time frame. For example if the first feature is calculated from 1024 sample frames and the second one from larger windows (number of beats per minute) and the third from the complete song (music track length), N will be set to the number of 1024 sample frames in the complete song. A music track length feature will then have the same values for all corresponding matrix entries.

2.2.3 Classification and Training

Supervised classification training creates models from given data and requires the ground truth information. Classification applies the previously learned models and computes the list of relations to the given categories for the provided music tracks. Unsupervised classification techniques categorize data without any given information by e.g. clustering. It is possible to run preprocessing before the classification, for example removing the outliers. Since RapidMiner [8] and WEKA [13] are also Java-based projects, they are integrated into AMUSE directly as libraries. It is also possible to connect to Matlab or R engines starting further classification methods.

2.2.4 Validation

The classification validator is responsible for evaluation of classification results. Confusion matrix-based metrics and error rates are well known and measure the quality of classification results. Other measures relate to the balance aspect - if a data set contains too much positive instances, accuracy may be high inspite of the poorly designed algorithm which tends to categorize everything as positive. Further it is possible to measure correlation between ground truth and predicted category relationships. All these metrics can be either calculated on the lower data level (measuring the classification success for smaller audio intervals as data instances) or on the higher data level

[1] http://amuse-framework.sourceforge.net

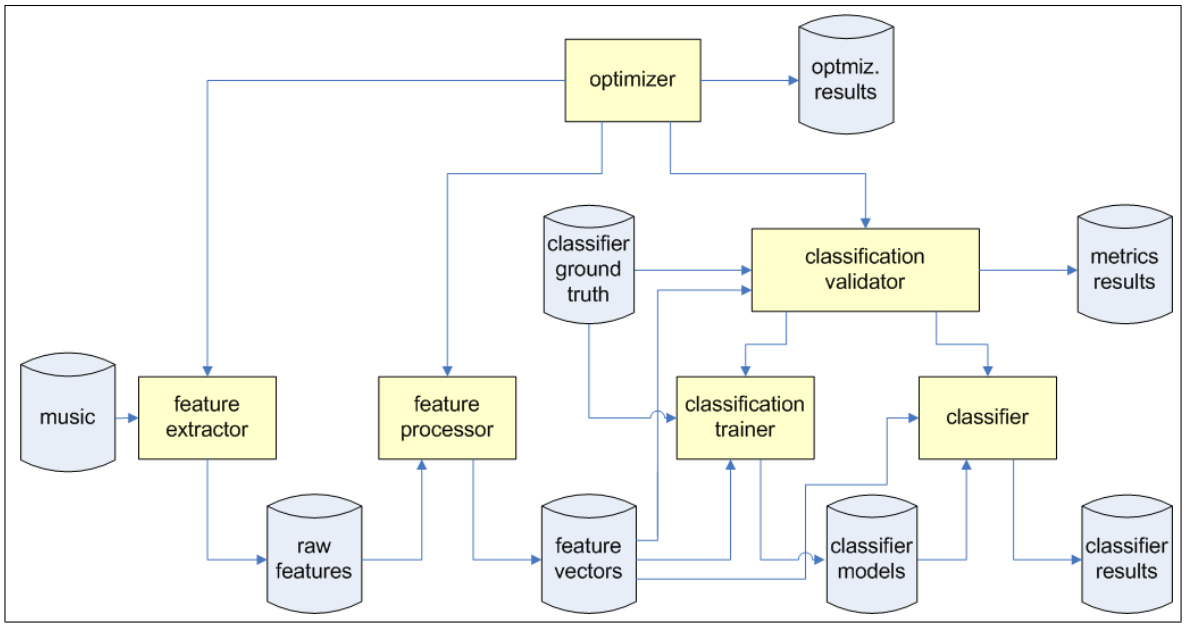


Figure 1. AMUSE task chain.

(evaluating classification averaged for the complete music tracks). Other metric group (data reduction rate) calculates the amount of data used for training related to the size of the original feature matrix.

2.2.5 *Optimization*

Each of the above mentioned tasks has more or less parameters and it is obviously, that it is not a simple task to find an optimal combination of them. For example the larger time frames for low-level feature extraction allow a more precise frequency resolution up to the Nyquist frequency. But if they are too long, different notes are mixed together and it becomes harder to learn anything. The optimization of the music data analysis chain is very rarely supported by related MIR tools. Indeed many optimization toolboxes exist (e.g. CILIB [11], SPOT [1] etc.) but the application is often too generic and must be adapted to the MIR domain. Therefore the goal of the optimization node is to run methods searching for optimal parameter settings. Currently several evolution strategies [2] are directly implemented in AMUSE and can be used for optimization.

2.3 Database Structure and Data Formats

Information provided by user (ground truth, music tracks) and the generated output are stored in folders called AMUSE databases. Most of the data is currently saved as text ARFF format [13] for several reasons: It is very comprehensible, is supported by most tools and is much more compact than XML. However it is an option to support further formats in future using e.g. MySQL database which requires more storage place but provides a very fast searching routine.

Music and *category* folders store the songs provided by the user and the corresponding ground truth for any related categories (music genres, information about harmony and melody etc.). *Feature* folders save the extracted features, the folder *processed features* stores the unlabeled feature

```
@RELATION 'Music feature'
%rows=6
%columns=11027
%sample_rate=22050
%window_size=512

@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE WindowNumber NUMERIC

@DATA
0,0,0,0,0,0,1
0,0,0,0,0,0,2
0,0,0,0,0,0,3
-0.0441210748392,-0.0319323236324,0.0317222975496,0.09711013405
0.0826042987199,-0.0714777017575,0.183883296449,0.0516827560259
-0.0635795107293,-0.126756415105,0.130888042853,0.11355759812,0
-0.14163669473,-0.0068830245434,0.0742737169193,0.0130153116809
-0.12271131203,0.0118226451875,0.0259055608418,-0.0554048395048
0.0388410122573,-0.0367863107091,0.136631523033,0.263694548501,
-0.216505347038,0.193737677751,0.085697018908?,0.159217103086,0
```

Figure 2. Example ARFF file with extracted features.

vectors for classification. Binary classification models are placed in the *model* directory, *metric* database is used for evaluation of music data analysis experiments. *Optimization* database contains of optimization logs in search for optimal parameter settings. Currently AMUSE does not support any visualization methods, but the data can be easily read into well-established tools like Gnuplot or Matlab.

An example for a feature file is given in Figure 2. Here the extended ARFF format is used: AMUSE attributes are placed as comments after the relation description and store the information about the data set size, sampling frequency and time frame size. Since for each feature the corresponding time frame is saved in attribute WindowNumber, it is simple to detect the time intervals from which the features have been extracted.

The AMUSE experiments are also saved as ARFFs.

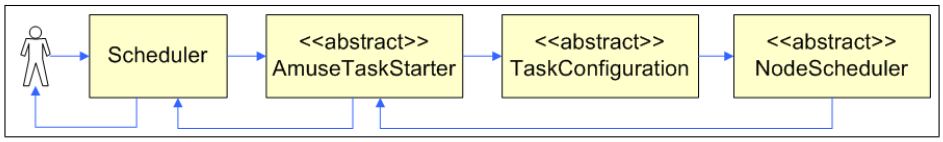


Figure 3. Data flow in AMUSE.

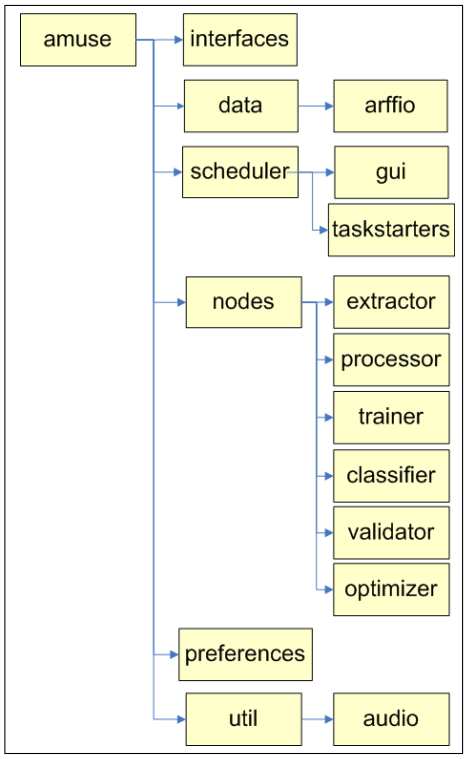


Figure 4. Package structure.

3. DETAILS FOR DEVELOPERS

The data flow during an AMUSE experiment is shortly depicted in Figure 3. User starts the main SCHEDULER component either from GUI or from the command line. After the configuration of the experiment the tasks are completely described in the corresponding TASKCONFIGURATION objects which are provided to the appropriate TASKSTARTER component. Here one or more AMUSE jobs are generated. These jobs are run on the same machine or are processed to a grid system. During the runtime of a single AMUSE instance the scheduler counts up the jobs. After they are ready, the next experiment can be started.

3.1 Package Structure

The most important AMUSE Java packages are shown in Figure 4. Scheduler and GUI packages interact with user, the computing of jobs is done in objects which are placed in nodes package and extend an abstract class NODESCHEDULER. Data package handles AMUSE data objects and ARFF input / output routines. Preferences store different configuration parameters (database folders, downsampling rate, path to grid scripts etc.). The util package contains logger and audio processing methods.

3.2 Guidelines for Tool Integration

Here we give a brief overview for several steps needed to extend AMUSE with new tools:

- Tool setup: Software which should be integrated into AMUSE must be tested for execution on the current operating system. It must be possible to start it either as Java library or by system call using a previously configured batch file.
- Writing an adapter class: Here the tool will be started. The functions which convert input and output data to AMUSE format must be implemented. If e.g. a feature extractor program saves the data as XML, it has to be converted into ARFF format for features as given in Figure 2.
- Plugin definition: The default way to integrate a new tool into AMUSE is to create the corresponding plugin. Several plugin installation files (mostly ARFFs) must describe the changes in AMUSE which will be applied after the installation. Each feature and each method used in AMUSE has a unique id number. The configuration file featureTable.arff lists all currently available features with given ids. If a new tool allows the extraction of several new features, this file must be updated. The same procedure is essential for further algorithms. There is a list with all available classification methods, validation metrics, processing and preprocessing algorithms etc.
- Plugin installation and integration should be tested. After the successful evaluation the job is done!

4. ONGOING WORK

The core framework has been already developed, however a lot of work remains. In the near future we will provide comprehensive introduction and developer manuals. Integration of further tools and extension with own methods belongs to the current and ongoing activities. Especially the optimization node will be extended with new methods related to multi-objective evaluation and computational intelligence algorithms.

As further steps we plan to add some visualization possibilities for experiment results and navigation possibilites through given music collections. The algorithms for symbolic and community-based retrieval can be also integrated.

5. ACKNOWLEDGEMENTS

We thank the Klaus Tschira Foundation for the financial support.

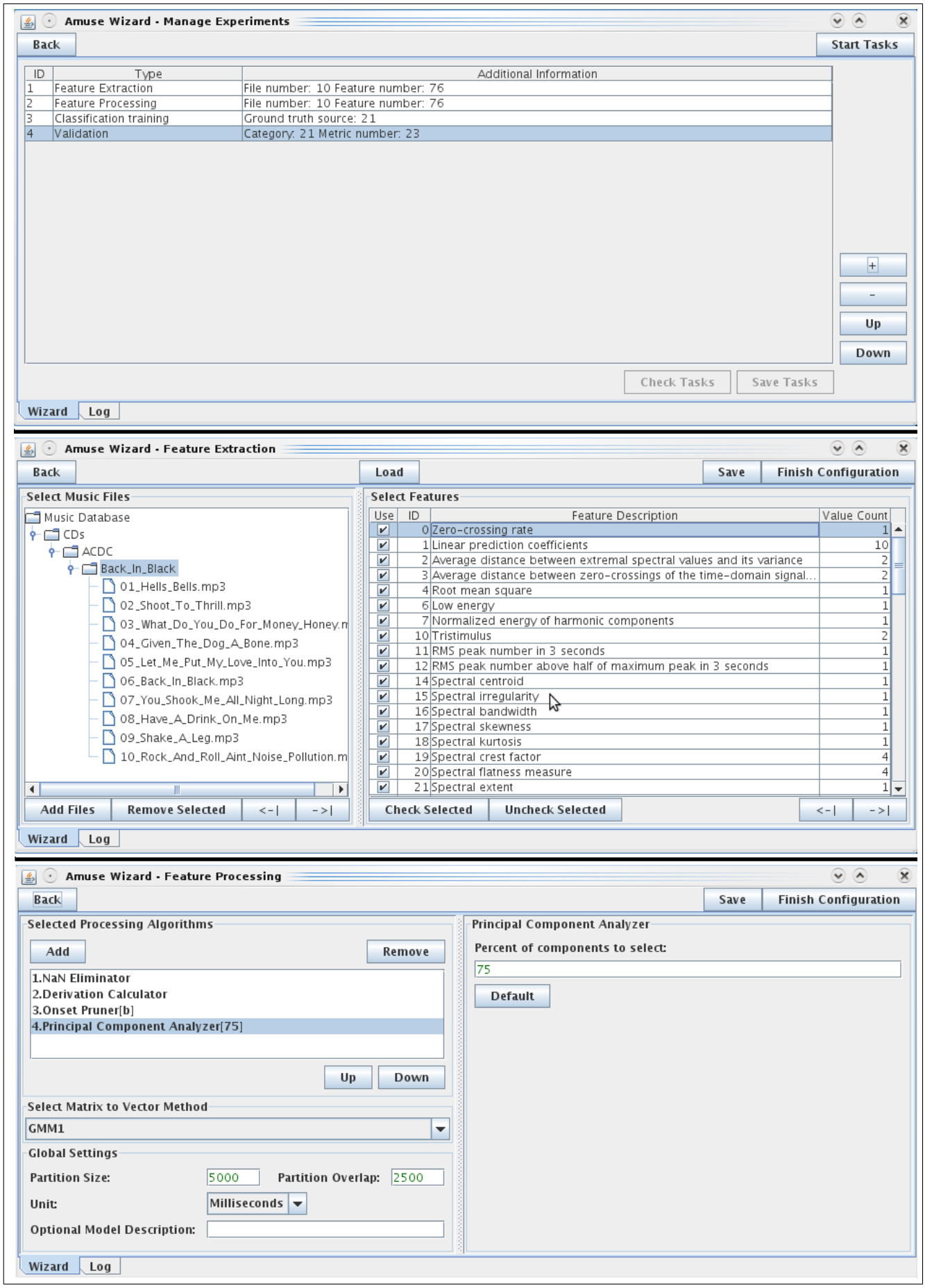


Figure 5. AMUSE GUI: Management of experiments (top); Feature extraction experiment setup (middle); Feature processing experiment setup (bottom).

6. APPENDIX: LIST WITH INTEGRATED TOOLS

Here we give an alphabetically sorted list of currently integrated tools. In AMUSE it is easy to create complex experiments using different algorithms, e.g. extracting features with jAudio and MIR Toolbox, preprocessing them with Matlab, classifying with WEKA and validating them with metrics available in AMUSE.

- Chroma Toolbox: Extraction of different novel chroma and pitch features [10].
- CMRARE: A set of cepstral modulation ratio regression (CMRARE) parameters for audio signal [5].
- jAudio: Java application for audio feature extraction [6].
- Matlab: Corresponding AMUSE adapter allows to run Matlab code. Path to the installed Matlab version must be set in AMUSE configuration.
- MIR Toolbox: A large set of Matlab fuctions for extraction of low-level and high-level audio descriptors [4].
- R: Corresponding AMUSE adapter allows to run R code. Path to the installed R version must be set in AMUSE configuration.
- RapidMiner (former Yale): Java framework for data mining [8]. A large number of different classification and data processing algorithms is available, audio feature extraction is provided by ValueSeries plugin.
- WEKA: An established framework for machine learning which is integrated as library in RapidMiner [13].

7. REFERENCES

[1] T. Bartz-Beielstein: *Experimental Research in Evolutionary Computation - The New Experimentalism*, Springer Verlag, 2006.

[2] H.-G. Beyer and H.-P. Schwefel: "Evolution Strategies - A Comprehensive Introduction," *Natural Computing*, Vol. 1, No. 1, pp. 3–52, 2002.

[3] B. Bischl, I. Vatolkin and M. Preuss: "Selecting Small Audio Feature Sets in Music Classification by Means of Asymmetric Mutation," Accepted for the *11th International Conference on Parallel Problem Solving from Nature (PPSN)*, Krakow, 2010.

[4] O. Lartillot and P. Toiviainen: "MIR in Matlab (II): A Toolbox for Musical Feature Extraction From Audio," *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)* pp. 127–130, 2007.

[5] R. Martin and A. Nagathil: "Cepstral Modulation Ratio Regression (CMRARE) Parameters for Audio Signal Analysis and Classification," *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 321–324, 2009.

[6] D. McEnnis, C. McKay and I. Fujinaga: "jAudio: Additions and Improvements," *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, pp. 385–386, 2006.

[7] C. McKay and I. Fujinaga: "jMIR: Tools for Automatic Music Classification," *Proceedings of the International Computer Music Conference (ICMC)*, pp. 65-68, 2009.

[8] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, T. Euler: "YALE: Rapid Prototyping for Complex Data Mining Tasks," *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06)*, pp. 935–940,2006.

[9] F. Mörchen, A. Ultsch, M. Noecker, C. Stamm: "Databionic Visualization of Music Collections According to Perceptual Distance," *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, pp. 396–403, 2005.

[10] M. Müller: *Information Retrieval for Music and Motion*, Springer Verlag, 2007.

[11] G. Pamparà, A.P. Engelbrecht and T. Cloete: "CIlib: A collaborative framework for Computational Intelligence algorithms - Part I," *Proceedings of the 2008 IEEE World Congress on Computational Intelligence (WCCI)*, pp. 1750–1757, 2008.

[12] G. Tzanetakis and P. Cook: "Marsyas: A framework for Audio Analysis," *Organised Sound*, Vol. 4, No. 3, pp. 169–175, 2000.

[13] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes and S.J. Cunningham: "Weka: Practical Machine Learning Tools and Techniques with Java Implementations," *Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop on Emerging Knowledge Engineering and Connectionist-Based Information Systems*, pp. 192–196, 1999.

[14] I. Vatolkin and W. Theimer: "Optimization of Feature Processing Chain in Music Classification by Evolution Strategies," *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN)*, Dortmund, pp. 1150-1159, 2008.

[15] I. Vatolkin, W. Theimer and G. Rudolph: "Design and Comparison of Different Evolution Strategies for Feature Selection and Consolidation in Music Classification," *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC 2009)*, IEEE Press, Piscataway (NJ), 2009.

AN IMPROVED HIERARCHICAL APPROACH FOR MUSIC-TO-SYMBOLIC SCORE ALIGNMENT

Cyril Joder, Slim Essid, Gaël Richard
Institut TELECOM, TELECOM ParisTech, CNRS LTCI
{cyril.joder, slim.essid, gael.richard}@telecom-paristech.fr

ABSTRACT

We present an efficient approach for an off-line alignment of a symbolic score to a recording of the same piece, using a statistical model. A hidden state model is built from the score, which allows for the use of two different kinds of features, namely chroma vectors and an onset detection function (spectral flux) with specific production models, in a simple manner. We propose a hierarchical pruning method for an approximate decoding of this statistical model. This strategy reduces the search space in an adaptive way, yielding a better overall efficiency than the tested state-of-the art method.

Experiments run on a large database of 94 pop songs show that the resulting system obtains higher recognition rates than the dynamic programming algorithm (DTW), with a significantly lower complexity, even though the rhythmic information is not used for the alignment.

1. INTRODUCTION

We address the problem of synchronizing a polyphonic musical score with an audio performance of this score, in the "off-line" version of this task. This allows one to consider the whole recording before estimating the positions of the score notes. We are interested in an alignment at the "symbolic level", which means that the result is the time indexes of the score notes or chords.

Applications of such a system can be a score retrieval from a musical query, or the ability to use both the audio and symbolic (score) content for music indexing. Some musical content analysis tasks, such as motif detection or chord transcription, may indeed be easier on symbolic data than on raw audio files.

While most on-line score following systems use statistical models which can be rather complex [4, 8, 14], many off-line algorithms simply rely on the DTW algorithms or refinements of it [6, 9]. These latter algorithms are often faster than the former and can also be applied to audio-to-audio synchronization.

However, their complexity (in time and space) is quadratic in the number of audio frames. This complexity problem has been addressed in [10], where a "short-time" DTW is proposed, which reduces the memory space requirement, at the cost of a greater time complexity. In [11], Müller *et al* introduce a "multi-scale" DTW (MsDTW) which allows for an efficient pruning strategy in a coarse-to-fine fashion.

To the authors' knowledge, hierarchical approaches have not been used for music synchronization, appart from [11]. In [3], Cont exploits a Hierarchical Hidden Markov Model. However, although its advantage in terms of interpretation, this structure is equivalent to a "flat" HMM [12].

With a dynamic programming framework, the use of different kinds of descriptors can be difficult. Hence such systems use a single feature representation, generally chroma vectors. A notable exception can be found in [6], where a strategy is proposed to combine local distances resulting from chroma vectors and onset features in a DTW scheme. The use of a statistical model makes the fusion of different pieces of information more natural. This structure is often used in real-time systems [2, 5], which model each feature distribution with a Gaussian mixture.

The hidden state model presented here exploits a different model for two different sets of features: a "histogram model" (see Sec. 2.1.1) for chroma vectors and a logistic model (see 2.1.2) for an onset indicator feature. This system obtains a very good alignment precision with a significantly lower complexity than the DTW algorithm.

We also introduce a hierarchical approach for search space reduction, which performs a pruning of the unlikely states in a hierarchical way. We take advantage of structural information given by the score (namely beat and bars), which allows for a meaningful hierarchical segmentation of the music. This method provides an alternative to the commonly used *beam search* strategy, which consists in maintaining only a fixed (small) number of paths at each decoding step. Our approach proves advantageous compared to both beam search and MsDTW, in terms of global search space size and runtime, without affecting the alignment performance in practice.

In the next section we present our baseline models for audio-to-score alignment. Then, a hierarchical pruning method for an approximate decoding of these models is proposed in Section 3. We expose the results of our experiments in Section 4 before suggesting some conclusions in Section 5.

THIS WORK WAS PARTLY SUPPORTED BY THE EUROPEAN COMMISSION UNDER THE OSEO PROJECT QUAERO.

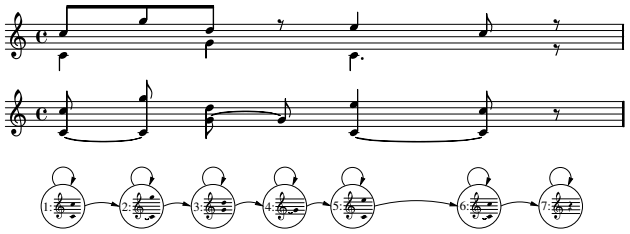


Figure 1. Score Representations. Top: The original graphical score. Middle: The score as a sequence of chord. Bottom: The finite state machine representing the score.

2. BASELINE MODELS FOR AUDIO-TO-SCORE ALIGNMENT

Similarly to [15], we segment the musical score into *chords*, which are sets of notes that sound at the same time. Every time a note appears or disappears, a new chord is created. We can then fit a hidden state model to the audio signal, the states of which are defined by the chords of the score. The score is seen as an automaton, as represented in Figure 1.

In this work, we chose not to take into account the rhythmic information given by the score, as we consider that we have no prior knowledge of the tempo. We use the Maximum Likelihood (ML) path in the automaton as the alignment path. Let $\mathbf{y}{=}y_1, \ldots, y_N$ be the feature sequence extracted from the signal. Let S_n be the random variables describing the current state at time n. The ML path $\hat{\mathbf{S}}$, which can be efficiently computed by the Viterbi algorithm, is:

$$\hat{\mathbf{S}} = \underset{\mathbf{S}\in\mathcal{S}}{\operatorname{argmax}}\, P\big(\mathbf{y}\big|\mathbf{S}\big) = \underset{\mathbf{S}\in\mathcal{S}}{\operatorname{argmax}} \prod_{n=1}^{N} P(y_n|S_n), \quad (1)$$

where $\mathcal{S}$ is the set of acceptable paths. We consider as acceptable the paths which go through all the states in the right order. This model is thus a left-right Hidden Markov Model whose only transitions are self-transitions and transitions from one state to the following one. All these transitions have the same probabilities.

2.1 Observation Models

Similarly to [13], two kinds of information are used in this work: the pitch content and the onset information. Thus, we use two types of features in order to take them into account. *Chroma vectors* are used in order to model the pitch content of the signal, and the *spectral flux* is supposed to detect the note onsets.

2.1.1 Chroma Vectors

As observed in [9], *chroma vectors* provide a compact, yet efficient representation of the pitched content of a musical signal for music-to-score matching. A chroma vector is a twelve-dimension vector, each of whose component represent the "power" in all the frequency bands of a chromatic class (from A to G#). The chroma vectors we use are computed according to [16], with a 50 Hz time resolution.

For each state s, a probability distribution $\{\tilde{g}(i)\}_{i=1\ldots12}$ over the 12 chroma components is built, as the superposition of one-note distributions which correspond to the notes that are present in the state. A one-note distribution is a simple Kronecker function $\{\delta(i,j)\}_{i=1\ldots12}$ where j is the pitch class of the considered note. Then, a constant component q is added in order to model noise, and we obtain a distribution g defined by $g(i) = (1-q)\tilde{g}(i) + \frac{q}{12}$. A value of 0.7 has been found satisfactory for the noise parameter q. For example, the distribution values corresponding to the chord $\{\mathrm{C}_3, \mathrm{E}_3, \mathrm{G}_3, \mathrm{C}_4\}$ are (represented in a vector) $\frac{1-q}{4}(0,0,0,2,0,0,0,1,0,0,1,0)+\frac{q}{12}\mathbb{1}$, where $\mathbb{1}$ is a vector of ones.

In order to calculate the likelihood of each state, we use the model exposed in [15]. The values of the chroma vector v extracted from the audio is considered as a histogram of random samples drawn from the distribution g (corresponding to chord c). The probability of having v as a result of such a sampling is:

$$p\big(v|c\big) = Z(v)\prod_{i=1}^{12} g(i)^{\alpha v(i)}. \quad (2)$$

Here, α is a scaling parameter. Since the value of this parameter has no effect on the decoding result, it is fixed to 1. $Z(v)$ is a positive number which only depends on the observation v, hence it is the same for every path and its value is not considered.

2.1.2 Spectral Flux Feature

In order to render the "burst of energy" which appears at a note onset, we exploit the *spectral flux* feature, which has been proven efficient in a beat tracking task [1]. We use this feature for a "probabilistic" onset detector.

The spectral flux values are first normalized so that their maximum is 1. A local threshold is then computed by applying a 67% rank filter of length 200 ms to the output. We then obtain an "onset feature" by subtracting this local threshold to the normalized spectral flux. Finally, a simple logistic model is used in order to calculate the likelihood of an onset. We denote by A the random variable representing the attack (onset) indicator ($\{A = 1\}$ means that there is an attack). For a value f of the onset feature, we have:

$$p\big(A = 1|f\big) = \frac{e^{bf}}{1+e^{bf}} \quad (3)$$

where b is a positive parameter, which controls the "confidence" of the onset detector: when the value increases, the decision is closer to a deterministic detector (with probabilities 0 or 1).

2.2 *Chord* and *Onset* Models

Two structures of HMMs are evaluated in this work. In the *Chord* structure, a chord is represented by a single state, and only the pitch information (described by the chroma vectors) is taken into account. The spectral flux is not considered.

The *Onset* model is a refinement of the previous structure which takes the onset information into account. In this model, a lower "level of hierarchy" is added in order to model two possible phases of a chord: *attack* and *sustain*. Earch chord corresponding to an onset is split into two successive *phase* states: attack ($A = 1$) and sustain ($A = 0$),

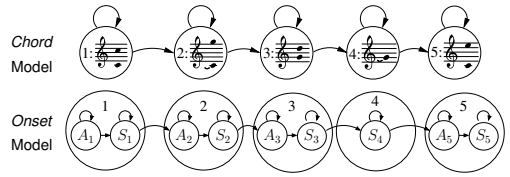


Figure 2. State Structure of the *chord* and *onset* models for the previous score (A and S stand for respectively *attack* and *sustain*).

which share the same chroma vector model. The two types of features are supposed to be independent. The chroma feature is assumed to depend only on the *chord* state while the onset feature only depends on the *phase* state. Hence, if we assume an uninformative prior about the *phase* state, *i.e.* $p(A = a) = \frac{1}{2}$, we have:

$$p\big(v, f|c, a\big) \propto p\big(v|c\big)p\big(A = a|f\big) \tag{4}$$

where these other probabilities are expressed in (2) and (3). Each state of the model is then the combination of a chord and a phase: we write $S_n = (C_n, A_n)$. Equation (1) is then:

$$\hat{\mathbf{S}} = \underset{\mathbf{S}\in\mathcal{S}}{\operatorname{argmax}} \prod_{n=1}^{N} p(v_n|C_n)p(A_n|f_n). \tag{5}$$

Six different values are tested for the parameter b of eq. (3): 0, 0.1, 1, 10, 50 and 100.

The structures of the two models are compared in Fig. 2. For the *Onset* model, the lower level states are represented inside the "chord super-states". In the example, the fourth super-state contains only a *sustain* state, because the transition from the previous chord to the fourth one does not correspond to an onset, but to the extinction of one note.

3. A NOVEL HIERARCHICAL PRUNING APPROACH

In order to speed up the decoding phase, we use a hierarchical pruning approach, inspired by the multi-scale Dynamic Time Warping (MsDTW) algorithm [11]. The idea is to first obtain a coarse alignment and then use the result to prune the search space at a more precise level.

For these coarse alignments, we take advantage of higher musical structures than what we call chords, namely *beat* and *bars*. These structures, given by the score, allow for a meaningful hierarchical segmentation of the music. At each of these levels, a HMM can be built, whose states correspond respectively to the beats and to the bars of the score. As the considered temporal units are larger and the precision needed at these levels is lower, the observations used for the alignment are calculated over longer windows, with a smaller time resolution. Figure 3 illustrates the construction of the automata and the calculation of the observations, at the three levels of hierarchy.

The algorithm proceeds as follows: on the highest level automaton, we calculate for every state s and every frame n, the maximum likelihood that can be obtained by going

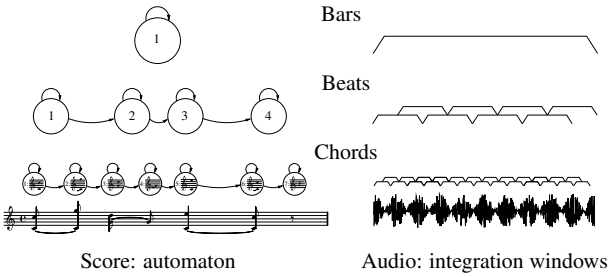


Figure 3. Finite state machines (modeling the score) and integration windows (over which are calculated the observations) at the three considered levels of hierarchy.

through state s at time n. This value is written

$$\bar{P}(s, n) = \max_{\mathbf{S}\in\mathcal{S}, S_n = s} \big\{P(\mathbf{y}|\mathbf{S})\big\} \tag{6}$$

where $\mathcal{S}$ is the set of acceptable paths and $\mathbf{y}$ is the observation sequence. This calculation can be done by a "forward-backward version" of the Viterbi algorithm. It is very similar to the forward-backward algorithm, and can be deduced from it by replacing the `sum` operation by a `max`. This algorithm allows for the calculation of the optimal path $\hat{\mathbf{S}} = \hat{S}_1, \ldots, \hat{S}_N$ at the same time.

The values $\bar{P}(s, n)$ are then used to prune the low-score paths. We do not use the posterior probabilities $P(S_n|\mathbf{y})$ instead of $\bar{P}(s, n)$ for the pruning process since we are interested in the path's scores and not in the states'. Since a state probability is the sum of the probabilities of the path going through this state, there is a risk that some states containing many average-score paths may be favored compared to a state containing an isolated high-score path.

This "pruning score" $\bar{P}(s, n)$ constitutes an important difference with the beam search strategy. Indeed, beam search operates directly at the low level and it uses the partial Viterbi score

$$\tilde{P}_n(s, n) = \max_{\mathbf{S}\in\mathcal{S}, S_n = s} \big\{P(y_1, \ldots, y_n|S_1, \ldots, S_n)\big\} \tag{7}$$

in order to prune the low-score path. Hence it only considers the observation up to the current frame, whereas our approach takes into account the whole signal.

The structure of the automaton is left-right, thus the relation defined on the set of states by: $s \leq s'$ iff there is a path from s to s', is a total order. It is then possible to define the "furthest admissible states" S_n^- and S_n^+ for each time n by:

$$S_n^- = \min\big\{s \;\big|\; \tilde{P}(s, n) \geq \frac{P(\mathbf{y}|\hat{\mathbf{S}})}{\eta}\big\} \tag{8}$$

$$S_n^+ = \max\big\{s \;\big|\; \tilde{P}(s, n) \geq \frac{P(\mathbf{y}|\hat{\mathbf{S}})}{\eta}\big\}, \tag{9}$$

where η is a parameter which controls the minimum likelihood of the paths that are kept in the pruning process. We define the *tolerance radii* δ_- and δ_+ as the maximum number of states that separate respectively S_n^- from $\hat{S}_n$ and $\hat{S}_n$ from S_n^+, for $n \in \{1, \ldots, N\}$.

These tolerance radii specify a set of states around the alignment path, which allows for a reduction of the search

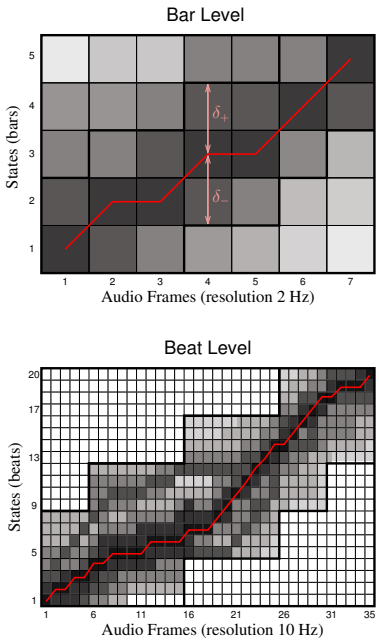


Figure 4. Principle of the hierarchical pruning method. The grey scale of a cell correspond to the maximum likekihood of the paths going through this cell. At the beat level, only the domain delimited by the black lines is explored.

space at the lower level. Hence, the alignment at the lower level is calculated by exploring only this domain. Figure 4 illustrates this pruning process. The same procedure is repeated at each level.

The observations used in the higher levels are integrated (moving average) versions of the chroma vectors, with a lower time resolution. This use of averaged observations is musically justified since the harmony (and thus the chroma information) is in general homogeneous over a whole beat (or bar) duration. The spectral flux is not considered in these levels. The integration windows are chosen in order to take into account the fastest reasonable tempi. For the beat level, this duration is 200 ms, corresponding to a very fast tempo of 300 beats per minute. For the bar level, the integration window is 1 s, that is a four-four time with a tempo of 240 bpm. A 50% overlap is used, yielding time resolutions of respectively 10 Hz and 2 Hz. The histogram model exposed in Sec. 2 is used and the distribution g corresponding to a state (beat or bar) is the superposition of the distributions associated to the chords that it contains, weighted by their theoretical durations (in beat).

The main difference between the MsDTW pruning approach and ours is that the tolerance widths δ are not given as a parameter, but they are computed from the data in an adaptive way, controlled by the parameter η. It is often more advantageous to set the tolerance in terms of likelihood (parameter η) than in terms of deviation from the alignment path (parameter δ). Indeed, it is possible that a wrong path obtains a slightly higher score than the right one at a coarse level (for example a path following a different repetition of a musical phrase). If this wrong path is far (in terms of states) from the right alignment, the latter one will be discarded by the fixed-radii pruning process. On the other hand, it is reasonable to suppose that the "real" alignment path always obtains a high likelihood, and thus is not pruned out by our method.

4. EXPERIMENTS

4.1 Database and Evaluation Measure

The database used in this work comprises 94 songs of the RWC-pop database [7]. These songs are polyphonic multi-instrumental pieces of length 2 to 6 minutes, most of which contain percussion. The alignment ground-truth is given by the synchronized MIDI files provided with the recordings. The same MIDI files are exploited as target scores. However, as we intend to be able to handle any type of score, in particular scores with missing or unreliable tempo indications, we artificially introduce (rather extreme) tempo modifications in these MIDI files: every 4 bars, a random tempo change (between 40 and 240 bpm) is added.

The chosen evaluation measure is the recognition rate, defined as the fraction of onsets which are correctly detected less than $\theta = 300$ ms away from the real onset time. This threshold is based on the MIREX'06 contest [1] .

4.2 Reference System: DTW

We compare our alignment models to a reference DTW (Dynamic Time Warping) system. The DTW algorithm searches for the alignment between two sequences which minimizes the cumulative costs along the alignment path.

This method is used to synchronize the sequence of observations (chroma vectors and spectral flux) extracted from the audio with a sequence built from the score. This "pseudo-synthesis" is performed by associating to each chord a chroma vector template (having the same values as the probability distributions of Sec. 2.1.1) and a duration given by the score. The obtained sequence is then linearly stretched so that its length is the same as the recording. For the onset detection feature, the reference sequence is a sequence of zeros and ones, the ones correponding to the onset locations in the "pseudo-synthesis".

For this system, the spectral flux sequence is locally normalized so that its maximum is 1 on a 2-s sliding window. The local distance between the observation $(v, \bar{f})$ (respectively chroma vector and locally normalized spectral flux) and the template counterpart (g, a) is given by:

$$D\Big((v, f), (g, a)\Big) = \frac{v \cdot g}{\|v\|\|g\|} + w\,|f - a|\,, \qquad (10)$$

where $\cdot$ denotes the inner product and w is a non-negative parameter which controls the weight given to the onset detection feature. Between three different values which have been tested $\{\frac{1}{2}, 1, 2\}$, the value $w = 1$ has been found the most efficient on our database. A DTW system which considers only the chroma observation (corresponding to $w = 0$) is also evaluated.

[1] Music Information Retrieval Evaluation eXchange 2006, score following task: http://www.music-ir.org/mirex/2006/index.php/Score_Following_Proposal

System	Recognition Rate	Search Space
DTW (only chroma)	78.77%	100%
DTW (chroma+onset)	**86.07**%	
Chord	64.49%	16.2%
Onset ($b = 0$)	69.70%	26.3%
Onset ($b = 0.1$)	70.49%	
Onset ($b = 1$)	73.14%	
Onset ($b = 10$)	82.90%	
Onset ($b = 50$)	**87.16**%	
Onset ($b = 100$)	84.71%	

Table 1. Recognition rate and mean search space (fraction of the DTW algorithm search space) as a function of the alignment system.

4.3 Performances of the Baseline Systems

The recognition rates and average search space of several settings are summed up in Table 1. The search space is the number of explored cells (state/frame pairs, or audio frame/pseudo-synthesis frame pairs, depending on the system) over the total number of cells required for the DTW algorithm (the square of the number of audio frames).

First, it can be seen that the DTW which considers only the chroma observations performs better than the *chord* model. This is easily explained by the fact that the former system implicitly models the note durations in the pseudo-synthesis stage, whereas the statistical models do not take them into account. This increases the precision, but also the search space (from 16.2% to 100%).

However, the use of the onset information allows the *onset* model to overcome this shortcoming and to obtain a slighty better precision than the DTW systems, with a still lower complexity. Indeed, a recognition rate of 87.16% is obtained with a value of $b = 50$, against 86.07% for the DTW system wich takes into account the onset observation, whereas the mean search space is 26.3%.

The increase of accuracy induced by the onset observation is smaller in the DTW system than in the statistical models. This is probably due to the difficulty of modeling the spectral flux process. Indeed, this onset detection function is not very well modeled by our binary templates, and the logistic model of (3) seems to be more relevant to this process than the local distance of (10).

The increase of search space in the *onset* model is beneficial to the alignment precision. Indeed, even with a value of $b = 0$ (which means that the onset information is not used) the recognition rate increases from 64.49% (*chord* model) to 69.70%. The explanation lies in the fact that most chords are then represented by two states. Thus the minimum duration of each chord is two frames instead of one, which prevents the system from rapidly skipping several states and leads to a smoother alignment path.

4.4 Pruning Evaluation

This hierarchical pruning method is run on the RWC popular music database. The lowest-level model uses the *onset* structure with parameter $b = 50$. Several values of the pruning parameter η have been tested and the experimental results are summed up in Table 2. The mean search space sizes are displayed for each pruning setting at the beat and chord level, as the fraction of explored cells over the total number of cells used by the DTW algorithm. At the bar level, it is 0.16% for MsDTW and 0.04% for all the other systems. For each setting, the total run-time is also presented, as well as the number of "pruning errors" on the whole database (94 songs). A pruning error occurs when a part of the ground truth alignment path is discarded by the pruning process. The implementation of the algorithms is in MATLAB, and was run on a Intel Core2, 2.66 GHz with 3.6 Go RAM under Linux.

System	Search Space		Run time (in s)	Errors (nb)
	Beats	Onsets		
Onset $b = 50$	–	26.26%	482	0
BS $N_h = 700$	–	5.74%	733	0
MsDTW $\delta = 150$	2.24%	14.02%	1180	0
$\delta = 60$	0.81%	7.93%	362	0
$\eta = 1000$	0.42%	4.53%	300	0
$\eta = 200$	0.35%	4.07%	276	0
$\eta = 100$	0.33%	3.82%	265	0
$\eta = 50$	0.30%	3.59%	256	0
$\eta = 20$	0.26%	3.22%	240	0
$\eta = 10$	0.23%	2.97%	229	2
$\eta = 5$	0.19%	2.59%	215	2

Table 2. Performance of our implementation of the alignment algorithm using different settings of the hierarchical pruning method.

The performance of three additional reference systems is displayed. This first one is the baseline onset model with no pruning. The second reference system uses beam search (BS). This algorithm performs the decoding of the statistical model similarly to the Viterbi algorithm, but maintains only the best N_h paths, according to the partial Viterbi score of (7). The minimum value of N_h for which the decoded path is the same as without pruning is $N_h = 700$.

The third reference system is a MsDTW (multi-scale DTW) system [11]. This system performs a DTW alignment, but it uses a coarse-to-fine pruning process in order to keep only a fixed neighborhood around the alignment path, at each level. The same three levels as in 3 are used. The deviation parameter value $\delta = 150$ is the minimum value yielding no pruning error on our database.

Finally, the last one uses constant tolerance radii $\delta_- = \delta_+ = 60$. This value δ is the lowest one for which no pruning errors are made.

In terms of alignment precision, all the systems which do not make pruning errors obtain the same scores as the reference system (87.16%). Thus, the reduction of the search space does not affect the alignment precision.

The results show the benefits of this pruning method, since the search space and run time of all the tested systems which use it are lower than the reference system (without pruning). As expected, the explored space decreases with the value of η. No pruning error occurs until a value of $\eta = 20$, whose corresponding run-time is half of the reference system (240 s against 484 s).

The benefit of this method compared to a fixed radius

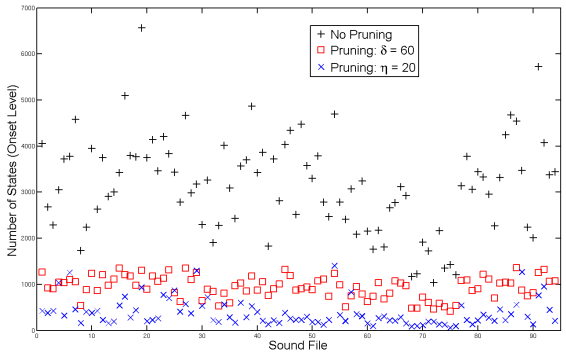


Figure 5. Number of explored states per audio frame at the onset level for each song of the database, without and with pruning (*onset* model with $b = 50$).

δ can also be seen: the tested system with $\delta = 60$ (the minimum value for no pruning error) runs in 362 s, and requires more space than our "adaptive" pruning strategy.

The hierarchical strategy allows our approach to be more effective than beam search (3.53% search space against 5.74%). Hence, considering the whole signal (although at a coarse level) seems to reduce the risk of following a promising path which will eventually come to a "dead-end". This problem could be adressed by estimating a tempo process in a beam search approach, such as in [15] or [4]. However the complexity of these models would be much higher.

In Fig. 5 are displayed the numbers of explored states per audio frame in each song of our database, for three different pruning strategies: the reference system (without pruning), the system using a fixed radius $\delta = 60$ and the system using our adaptive approach with $\eta = 20$. Both pruning strategies achieve a significant reduction of the search space on all the songs. More interestingly, we can see that the search space width obtained with our pruning strategy can greatly vary from songs to songs, whereas it is more or less constant with a fixed δ (only affected by the number of onset states in a beat). This variability is uncorrelated to the original number of states in the score, indicating that our approach manages to adapt the pruning process to the data. Thus, whereas in some cases, the width obtained with our method is greater than with a constant δ, it is most of the time significantly smaller.

5. CONCLUSION

In this paper, we show that a novel hierarchical pruning approach for the approximate decoding of a hidden state model leads to a good precision in our alignment task, with a low complexity. In our experiments, we find that the recognition rate is even higher than a DTW system when a description of note onsets is used additionally to the chroma vectors, while keeping a lower complexity than this algorithm in the decoding phase.

The proposed hierarchical pruning method further reduces the complexity without affecting the accuracy of the system. The main advantage of this strategy compared to the one used in [11] is that the tolerance radii can adapt to the data, yielding a better overall efficiency.

In the continuation of this work, we will address the use of a more elaborate model at the lowest level, which is now feasible thanks to the pruning strategy. We will also try to further reduce the number of states in the model, by taking advantage of the repetitions in the musical structure.

6. REFERENCES

[1] M. Alonso, G. Richard, and B. David. Extracting note onsets from musical recordings. In *Proc. of ICME*, 2005.

[2] P. Cano, A. Loscos, and J. Bonada. Score-performance matching using hmms. In *Proc. of the ICMC*, 1999.

[3] A. Cont. Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical hmms. In *Proc. of ICASSP*, 2006.

[4] A. Cont. A coupled Duration-Focused architecture for Real-Time Music-to-Score alignment. *IEEE Trans. on PAMI*, 32(6):974–987, June 2010.

[5] A. Cont, D. Schwarz, and N. Schnell. Training ircam's score follower. In *Proc. of ICASSP*, 2005.

[6] S. Ewert, M. Müller, and P. Grosche. High resolution audio synchronization using chroma onset features. In *Proc. of ICASSP*, 2009.

[7] M. Goto. Rwc music database: Popular, classical, and jazz music databases, 2002.

[8] L. Grubb and R. Dannenberg. A stochastic method of tracking a vocal performer. In *Proc. of ICMC*, 1997.

[9] N. Hu, R. Dannenberg, and G. Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *Proc. of WASPAA*, 2003.

[10] H. Kaprykowsky and X. Rodet. Globally optimal short-time dynamic time warping: Application to score to audio alignment. In *Proc. of ICASSP*, 2006.

[11] M. Müller, H. Mates, and F. Kurth. An efficient multiscale approach to audio synchronization. In *Proc. of ISMIR*, 2006.

[12] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. UC Berkeley, july 2002.

[13] N. Orio and D. Schwarz. Alignment of monophonic and polyphonic music to a score. In *Proc. of ICMC*, 2001.

[14] C. Raphael. Automatic segmentation of acoustic musical signals using hidden markov models. *IEEE Trans. on PAMI*, 21:360–370, 1999.

[15] C. Raphael. Aligning music audio with symbolic scores using a hybrid graphical model. *Machine Learning Journal*, 65:389–409, 2006.

[16] Y. Zhu and M. Kankanhalli. Precise pitch profile feature extraction from musical audio for key detection. *IEEE Trans. on Multimedia*, 8(3):575–584, 2006.

An Improved Query by Singing/Humming System Using Melody and Lyrics Information

Chung-Che Wang
MIR Lab, Dept. of CS,
Tsing Hua Univ., Taiwan
geniusturtle
@mirlab.org

Jyh-Shing Roger Jang
MIR Lab, Dept. of CS,
Tsing Hua Univ., Taiwan
jang@mirlab.org

Wennen Wang
Institute for Information
Industry, Taiwan
wennen@iii.org.tw

ABSTRACT

This paper proposes an improved query by singing/humming (QBSH) system using both melody and lyrics information for achieving better performance. Singing/humming discrimination (SHD) is first performed to distinguish singing from humming queries. For a humming query, we apply a pitch-only melody recognition method that has been used for QBSH task at MIREX with rank-1 performance. For a singing query, we combine the scores from melody recognition and lyrics recognition to take advantage of the extra lyrics information. Lyrics recognition is based on a modified tree lexicon that is commonly used in speech recognition. The performance of the overall QBSH system achieves 39.01% and 23.53% error reduction rates, respectively, for top-20 recognition under two experimental settings, indicating the feasibility of the proposed method.

1. INTRODUCTION

Query by singing/humming (QBSH) is an intuitive method for music retrieval. With a QBSH system, users are able to retrieve intended songs by singing or humming a portion of the intended song in order to retrieve it. Most of the QBSH researches so far utilize melody information as the only cue for retrieval [1-3]. Ghias et al. [1] proposed one of the early papers of query by humming, which used three different characters ('U', 'D', and 'S') are to represent pitch contours. McNab et al. [2] enhanced the representation by considering rhythm information of segmented notes. Jang and Gao [3] proposed the first QBSH system using dynamic time warping (DTW) over frame-based pitch contours, which accommodates natural singing/humming for better performance. More recently, QBSH task is held in MIREX since 2006 and quite a few related methods and corresponding performance can be found therein [12].

Lyrics are also an important part of a song which serve the cue for detecting the song's identity, or its mood or genre. However, the use of lyrics for content-based music analysis appears much later. Wu et al. [4] and Chen [14] used lyrics to enhance music mood estimation. Wang et al. [5] proposed one of the few music information retrieval systems which used both lyrics and melody information. However, queried lyrics were input by the user instead of decoded from the user's acoustic input. Xu et al. [6] suggested that acoustic distance must be considered for a lyrics search when the user input approximate lyrics query. Our method also takes advantage of the extra information provided by the lyrics, except that we attempt to decode the queried lyrics from the user's singing input directly, which does not impose extra efforts on the user. Suzuki et al. [13] also proposed a similar system which took singing input for lyrics recognition, and the results were verified by the corresponding melody information. However, their system could not handle humming input, which is likely to happen in a music retrieval system. Moreover, the corpus used for their experiments is too small to justify the method's feasibility.

The proposed QBSH system uses singing/humming discrimination (SHD) to detect whether there exists lyrics information. If yes, we apply speech recognition to decode the lyrics information and come up with a lyrics score. The lyrics score is combined with the melody score to enhance the recognition performance.

The remainder of this paper is organized as follows. The proposed QBSH system is introduced in section 2. Experimental results are shown in section 3. We conclude this paper and address directions for future work in section 4.

2. SYSTEM OVERVIEW

Figure 1 shows the schematic diagram of the proposed system, in which blocks enclosed by thicker lines are the methods proposed by this paper. In the offline part, acoustic models and test corpus are used to obtain the model similarity, where each model is characterized by an RCD (right-context dependent) bi-phone. Phone-level similarity is used for SHD, and syllable-level similarity is used for lyric scoring. Lexicon network is also created in this part for lyrics recognition. In the online part, SHD is first performed to decide if the acoustic input is singing or humming. If the input is classified as humming, the result is based on melody recognition alone. On the other hand, if the input is classified as singing, lyrics recognition is performed to obtain a decoded string of lyrics. The output of our system then uses the combined scores of melody and lyrics. The melody recognition module uses UPDUDP [11] for pitch extraction and linear scaling (LS) [7] for comparison, which achieved the

best performance during QBSH task in MIREX2009 [12]. The other components will be explained in detail in the following subsections.

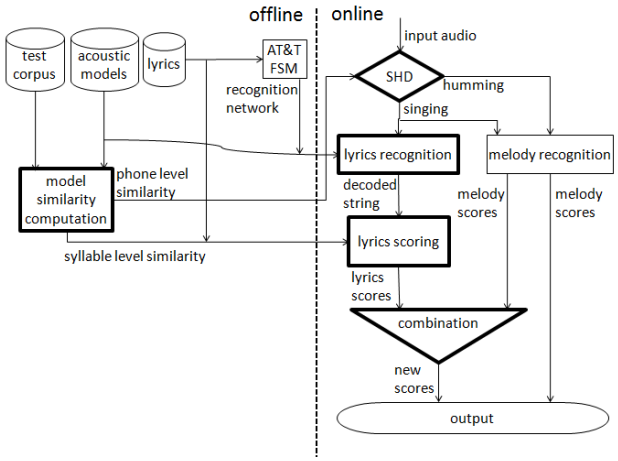


Figure 1. The proposed system.

2.1 Phone and Syllable Similarity

An intuitive approach to SHD is based on the number of distinct phones decoded in the user's acoustic input. The more distinct phones in an acoustic query, the more likely the query is singing instead of humming. In counting distinct phones, we also need to take phone similarity into consideration for achieve most robust results. Moreover, we also need to have similarity between syllables for obtaining lyrics score. The procedure for computing phone and syllable similarity is explained next.

Firstly, we can obtain the confusion matrix of 156 biphone models by performing free phone decoding on a speech corpus by HTK [8]. Then we can use the accuracy rates of the confusion matrix as the similarity measure between any two phone models. It should be noted that the similarity is not symmetric, but this does not affect the proposed methods.

After defining phone similarity K, the similarity matrix of 423 Mandarin syllables can be computed via dynamic programming (DP) as follows. Considering two syllables Syl_A and Syl_B, with phone sequence $a_1, a_2, \dots, a_m$ and $b_1, b_2, \dots, b_n$, respectively, the definition of the similarity between Syl_A and Syl_B is:

$$sim(Syl_A, Syl_B) = \frac{t_{A,B}(m,n)}{\max(m,n)}, \quad (1)$$

where the recursive formula of $t_{A,B}$ is:

$$t_{A,B}(i,j) = \max\begin{pmatrix} t_{A,B}(i-1,j) \\ t_{A,B}(i,j-1) \\ K(a_i,b_j)+t_{A,B}(i-1,j-1) \end{pmatrix}, \quad (2)$$

with boundary conditions :

$$t_{A,B}(i,j) = 0, \ if\ i = 0\ or\ j = 0. \quad (3)$$

Figure 2 shows the image of 423 x 423 similarity matrix in gray scale, where white points represent 1, and black points represent 0.

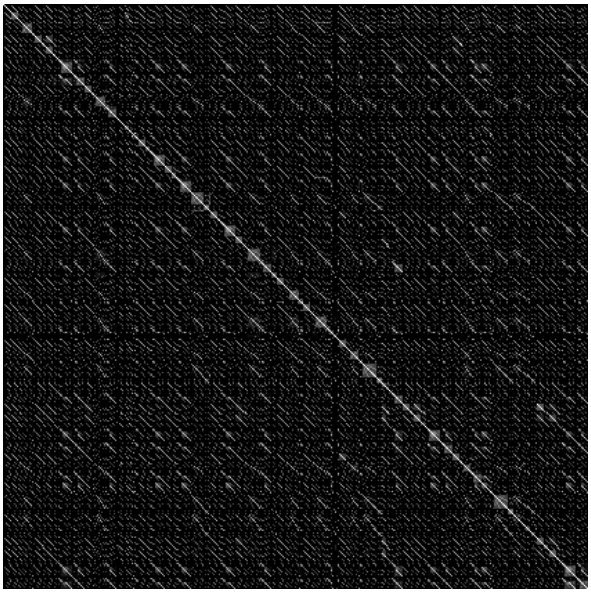


Figure 2. The similarity matrix of 423 Mandarin syllables displayed as a gray-scale image.

2.2 Singing/Humming Discrimination

The basic rationale behind SHD is that the number of distinct phones occurring in humming is often less than that in singing. Thus, free phone decoding is performed on the singing input to obtain a phone sequence. If these phones are similarity in acoustics, then the effective count of distinct phones should be less. Assume there are n distinct phones $a_1, a_2, \dots, a_n$ in the decoded phone sequence, then the effective count is defined as follows. Let P to be a sub matrix of sim for $a_1 a_2, \dots, a_n$, then we can calculate the effective phone count in the sequence:

$$r = n + 1 - median(s) \quad (4)$$

where r is the effective phone count, and s is the column sum of P. A lower effective phone count indicates that the acoustic query is more likely to be humming instead of singing. In particular, when $a_1 a_2, \dots, a_n$ are very different in pronunciation, then $median(s)$ is close to 1 and r is close to n. On the other hand, if these phones are very similar in pronunciation, then $median(s)$ is close to n and r is close to 1.

Thus an optimum threshold of effective phone count can be set for SHD for minimizing classification error.

2.3 Lyrics Recognition

If an acoustic query is classified as singing, we can apply lyrics recognition for better performance. Since the aver-

age length of a singing query is about 7 seconds, the first 30 syllables of each song are used to build up the recognition network. (Without loss of generality, here we assume the anchor position of each query is the beginning of a song. If not, then we can simply use the phrase onset as the beginning to select the first 30 syllables for building the network.) By considering the recognition network as a finite state machine, this network is determinized and minimized by AT&T FSM tool [9]. Moreover, to handle the case of "stop in the middle", an epsilon transition is added between each internal state and the terminal state. Figure 3 shows an example of the network, consisting of the first 3 syllables of 2 songs, where Song-i-Syl-j denotes the j-th syllable of the i-th song, and <eps> denotes the epsilon transition.

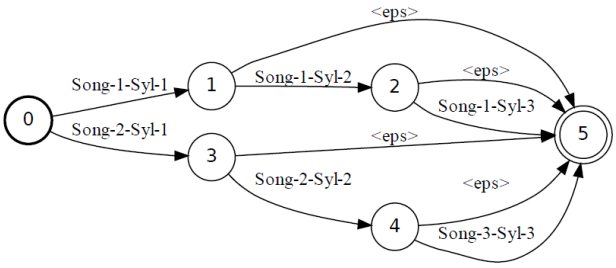


Figure 3. An example of the recognition network.

2.4 Lyrics Scoring and Combination

After running Viterbi search over the recognition network, we can decode a syllable sequence that has the maximum likelihood. To obtain the similarity score based on lyrics, we then compare the decoded syllable sequence with the 30 syllables of each song. This is again achieved by DP instead of using exact string matching since we want to have a score indicating the similarity. Consider two syllables sequences, Seq_A and Seq_B containing syllables $A_1, A_2, \dots, A_m$ and $B_1, B_2, \dots, B_n$ respectively. The recursive formula for DP is:

$$t(i,j) = max\begin{pmatrix} t(i-1,j) \\ t(i,j-1) \\ sim(A_i, B_j) + t(i-1,j-1) \end{pmatrix}, \quad (5)$$

with boundary conditions :

$$t(i,j) = 0, if\ i = 0\ or\ j = 0 \quad (6)$$

Thus, $t(m,n)$ can be taken as a similarity score between the decoded string from the query and the lyrics of each song in the database. In implementation, we let Seq_A to be the decoded string, and the first k syllables of the lyrics (with k equal to the length of Seq_A) to be Seq_B for computing the score.

For score combination, we need to normalize each individual score. For a given query to a song database of 2000 songs, we will eventually obtain vector L of size 2000 representing the lyrics raw similarity scores, and vector M of size 2000 representing the melody raw distance measures (computed by LS). We than linearly normalize M and $-L$, respectively, to the range [0, 1]. The normalized distance vectors M' and L' are then combined via the following formula:

$$C = p \times \log M' + (1-p) \times \log L' \quad (7)$$

Then the minimum entry in C corresponds to the most likely song considering both lyrics and melody. Note that if p is equal to 0, only the lyric information is considered. On the other hand, if p is equal to 1, only the melody information is considered. The value of p was set to 0.5 empirically in our experiments.

3. EXPERIMENT

3.1 The Dataset

The public corpus MIR-QBSH [10] is used extensively in our experiment, where the anchor positions for all queries are from the beginning. Since our speech recognition engine is for Mandarin, therefore we selected 2469 clips from the corpus which correspond to 23 Mandarin songs. To increase the complexity of the comparison, we added 2131 noise songs to the database, such that the total size of the database is 2154.

First of all, 80 clips are selected from the corpus with evenly distributed gender and types (singing/humming). These clips were hand labeled to have the ground truth, and then used to train the threshold-based classifier for SHD. The remaining 2389 clips are used for testing the overall performance of our QBSH system.

Acoustic models of RCD bi-phones for computing phone/syllable similarity were obtained by training over a normal Mandarin speech of 80 subjects.

3.2 Experimental Results

3.2.1 Results of SHD

Figure 4 shows the detection error tradeoff (DET) curve of SHD using the training data of 80 clips, where singing clips are viewed as positive while humming clips negative. Based on this plot, the threshold of effective count is set to 20.4958 for SHD to achieve equal error rates of false positive and false negative. Figure 5 gives the distribution of the effective phone counts of the training data, together with the identified Gaussian models via maximum likelihood estimate. The Gaussian models for positive data (of size n_1) and negative data (of size n_2) are denoted as g_1 and g_1, respectively, in the figure.

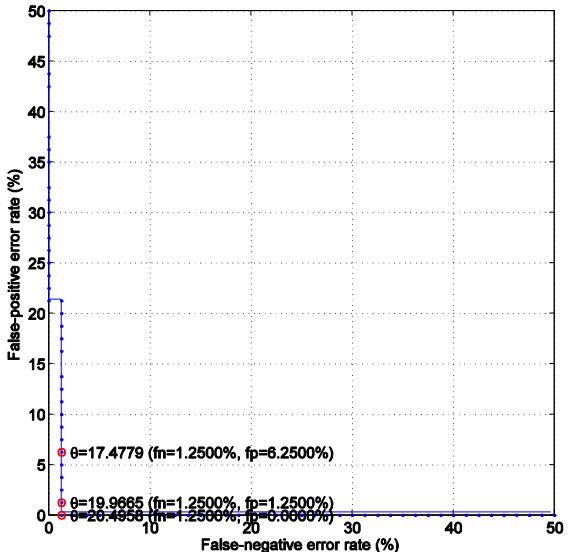


Figure 4. The DET curve of the training data for SHD.

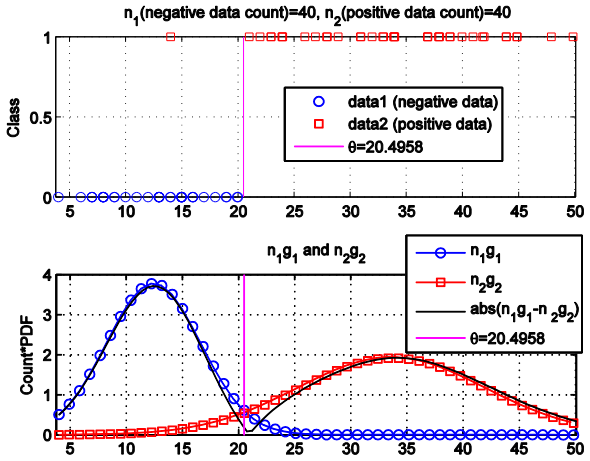


Figure 5. The distribution of the effective phone counts of the training data of SHD. The Gaussian models for positive data (of size n_1) and negative data (of size n_2) are g_1 and g_1, respectively.

To evaluate the performance of SHD over unseen data, 1183 clips were selected (out of the 2389 clips) and hand labeled as singing or humming. Table 1 shows confusion matrix of SHD, with a recognition rate of 78.61%. In particular, 10.99% of the humming clips are misclassified as singing, which may generate erroneous output in lyrics recognition. Initial error analysis indicates that some of the misclassified humming clips are caused by a variety of pronunciation during the humming. On the other hand, 24.51% of the singing clips are misclassified as humming, which is not so detrimental to the overall performance since the accuracy of melody recognition is already high.

ground truth \ recognition results	singing	humming
singing	75.49% (687)	24.51% (223)
humming	10.99% (30)	89.01% (243)

Table 1. Recognition result of SHD.

3.2.2 Lyrics recognition and Combined Results

When we applied SHD to 2389 clips, 1477 of them were classified as singing. Figure 6 shows the top-20 recognition rate of these 1477 clips over a song database of size 2154. The top-20 recognition rate is 72.99%.

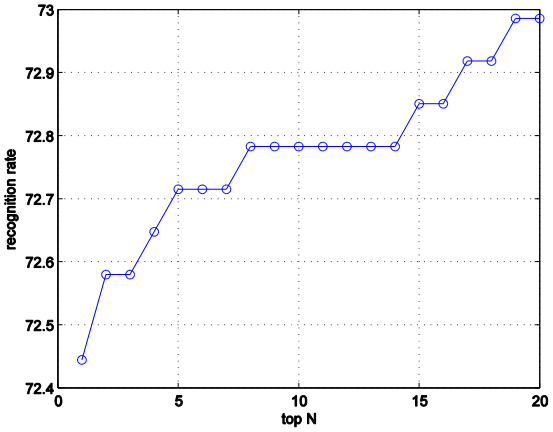


Figure 6. The top-20 lyrics recognition rate of 1477 clips classified as singing.

The value of p in Eq. (7) was set to 0.5 empirically. Now it is time to do a post analysis by plotting the overall recognition rates versus the values of p, as shown in Figure 7. Apparently the performance stays much the same for these two cases of LS resolution equal to 11 and 51, respectively, as long as the value of p lies within [0.3, 0.9]. This confirms our selection of p value of 0.5.

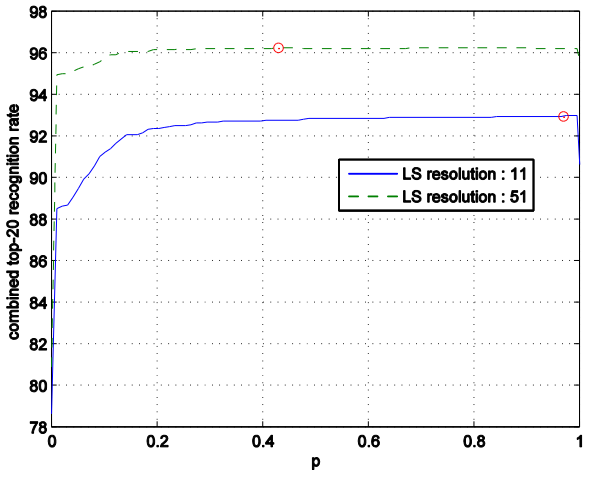


Figure 7. The plots of overall recognition rates with respect to the value of p, for two cases of different LS resolutions. The maximum of these two curves are labeled with circles.

Figure 8 shows the overall performance of the proposed QBSH system. Different values of resolution in LS, 11 and 51, were used in this experiment. The lower and upper ratios of LS were set to be 0.5 and 2. The top-20 recognition rates of resolution 11 and 51 are 92.80% and 96.19%, respectively, which outperform the baseline system (88.20% and 95.02%). The error reduction rates are 39.01% and 23.53%, respectively.

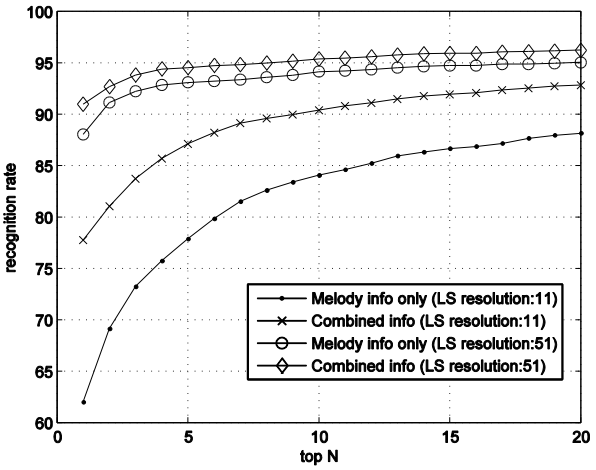


Figure 8. The top-N recognition rate of 2389 clips.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed an improved QBSH system that distinguishes singing queries from humming ones, and then applies different procedures in order to take advantage of the lyric information of singing input. The experimental results demonstrate the effectiveness of the proposed system, with error reduction rates of 39.01% and 23.53% for LS with resolutions of 11 and 51, respectively.

Several directions for immediate future work are under way. Currently, our acoustic models were obtained by training on normal speech corpus. This can be improved by training or simply adapting using singing corpora instead. Moreover, it would be desirable to incorporate multi-lingual speech recognition since there are quite a few famous songs with the same tune but different lyrics in different languages.

5. ACKNOWLEDGEMENT

This study is conducted under the "III Innovative and Prospective Technologies Project" of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs of the Republic of China.

6. REFERENCES

[1] A. J. Ghias, D. C. Logan, and B. C. Smith, "Query by humming-musical information retrieval in an audio database," in *Proc. ACM Multimedia'95*, San Francisco, 1995, pp. 216–221.

[2] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham, "Toward the digital music library: Tune retrieval from acoustic input," in *Proc. ACM Digital Libraries*, 1996, pp. 11–18.

[3] J.-S. R. Jang and M.-Y. Gao, "A query-by-Singing system based on dynamic programming," in *Proc. Int. Workshop Intell. Syst. Resolutions (8th Bellman Continuum)*, Hsinchu, Taiwan, R.O.C., Dec. 2000, pp. 85–89.

[4] Y.-S. Wu, W.-r. Chu, C.-Y. Chi, D. C. Wu, R. T.-H. Tsai, and J Y.-j Hsu, "The Power of Words: Enhancing Music Mood Estimation with Textual Input of Lyrics," *International Conference on Affective Computing & Intelligent Interaction*, pp. 1–6, 2009.

[5] T. Wang, D.-J. Kim, K.-S. Hong, and J.-S. Youn, "Music Information Retrieval System using Lyrics and Melody Information," *Asia-Pacific Conference on Information Processing*, pp. 601–604, 2009.

[6] X. Xu, M. Naito, T. Kato, and H. Kawai, "Robust and Fast Lyric Search Based on Phonetic Confusion Matrix," *Proceedings of the International Symposium on Music Information Retrieval*, pp. 417–422, 2009.

[7] J.-S. R. Jang, H.-R. Lee, M.-Y. Kao, "Content-based Music Retrieval Using Linear Scaling and Branch-and-Bound Tree search," in *Proc. of IEEE*

International Conference on Multimedia and Expo, August 2001.

[8] Cambridge University Engineering Department , HTK Web-Site, http://htk.eng.cam.ac.uk/, 2006

[9] AT&T Labs Research , AT&T Labs Research - FSM Library , http://www2.research.att.com/~fsmtools/fsm/, 2008

[10] J.-S. R. Jang, "MIR-QBSH Corpus", MIR Lab, CS Dept, Tsing Hua Univ, Taiwan. Available at the "MIR-QBSH Corpus" link at http://www.cs.nthu.edu.tw/~jang.

[11] J.-C. Chen, J.-S. R. Jang, "TRUES: Tone Recognition Using Extended Segments", *ACM Transactions on Asian Language Information Processing, No. 10, Vol. 7*, Aug 2008.

[12] MIREX 2009, http://www.music-ir.org/mirexwiki/index.php/Main_Page, 2009

[13] M. Suzuki, T. Hosoya, A. Ito, and S. Makino, "Music Information Retrieval from a Singing Voice Using Lyrics and Melody Information," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, Article ID 38727, 8 pages, 2007. doi:10.1155/2007/38727

[14] J.-H Chen, "Content-based Music Emotion Analysis and Recognition", Master Thesis, CS Dept., National Tsing Hua University, June 2006

AN INTERCHANGE FORMAT FOR OPTICAL MUSIC RECOGNITION APPLICATIONS

Andrew Hankinson[1] **Laurent Pugin**[2] **Ichiro Fujinaga**[1]
[1]Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT)
Schulich School of Music, McGill University
[2]RISM Switzerland & Geneva University
andrew.hankinson@mail.mcgill.ca, lxpugin@gmail.com,
ich@music.mcgill.ca

ABSTRACT

Page appearance and layout for music notation is a critical component of the overall musical information contained in a document. To capture and transfer this information, we outline an interchange format for OMR applications, the OMR Interchange Package (OIP) format, which is designed to allow layout information and page images to be preserved and transferred along with semantic musical content. We identify a number of uses for this format that can enhance digital representations of music, and introduce a novel idea for distributed optical music recognition system based on this format.

1. INTRODUCTION

Page appearance and layout for music notation is a critical component of the overall musical information contained in a document. For example, musically semantic information, such as note duration, is often visually augmented by adjusting horizontal spacing to reflect a spatial representation of note duration [1]. Some scholars infer geographical origin or time period based on note shapes, or even, in the case of handwritten manuscripts, by the particular "hand" of a scribe [2, 3]. The layout may also reveal some subtle intent of the composer, especially in sketches and autograph manuscripts [4].

To date, however, there has been little effort to attempt to preserve this information when a page is scanned and processed by optical music recognition (OMR) software. This presents several opportunities for improvement. By maintaining a direct relationship between recognized musical symbols and the original image it was extracted from, we contend that musicians and music scholars will be better able to understand and interpret digital facsimiles of musical documents while simultaneously providing the ability to index, search, and retrieve these documents.

For OMR researchers, this also presents an opportunity to build large global ground-truth datasets. By maintaining the relationship between the graphical representation and the semantic interpretation of a musical symbol, we can build sets of training data which exemplar-based adaptive supervised-learning software can use to train and test its recognition models. Furthermore, by allowing for these datasets to be shared between different adaptive OMR platforms, we can take advantage of work done by others who have created different datasets to further improve recognition software. This is discussed further in Section 4.

In this paper, we present the OMR Interchange Package (OIP) format, a common interchange format for OMR applications that bundles notation, images, and metadata together in a single file. Work on this format was inspired by functionality present in large, established digitization projects, most notably Google Books and the Internet Archive. These projects use file formats designed to preserve layout information in textual materials. We discuss two such formats, hOCR and DjVu, and examine them for ideas of how we might construct a similar music notation-specific format.

Rather than build a completely separate set of specifications, the OIP format combines established standards into an application profile—that is, we provide specifications on how these standards should be combined. These standards concern music, image, and metadata encoding formats, contained within an established standard for packaging and serializing these files into a single file, for easy transport across multiple systems. By taking an application profile approach, instead of establishing a new, monolithic standard, we hope to take advantage of existing software to manipulate component files, e.g., reading and writing images, and delegate the maintenance and improvement of the component standards to their respective communities.

One of the goals for developing the OIP format is to provide a mechanism for interchange between different elements in an OMR digitization workflow, from capture through recognition and into any number of potential uses. Specific design considerations were made to ensure that non-common practice notation systems are

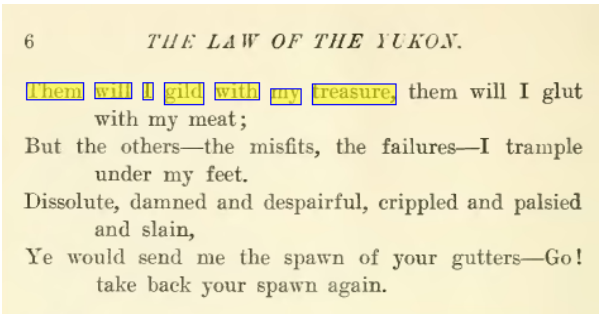
accommodated, to allow for encoding earlier musical print and manuscript sources.

2. BACKGROUND

2.1 Optical Character Recognition

The Google Books project [5] and the Internet Archive [6] are industrial-scale initiatives to convert physical textual items, e.g., books, magazines, and newspapers, to searchable digital representations. As items in these collections are digitized, their page images are processed by OCR software, extracting the textual content, and facilitating full-text searching of their collections.

Within the OCR workflow, the precise location on the page where a word occurs is saved through the use of a bounding box that defines a region around the word. When the words on the page are converted to searchable text, the bounding box coordinates are stored, along with the word itself. In some cases, similar coordinates can be stored to outline higher-level page elements such as lines, columns, or paragraphs. Figure 1 shows an example from the Internet Archive of a page image returned from a full-text search with the phrase "Them will I gild with my treasure" highlighted in reference to its position on the original page scan.


6 THE LAW OF THE YUKON.

Them will I gild with my treasure, them will I glut
with my meat;
But the others—the misfits, the failures—I trample
under my feet.
Dissolute, damned and despairful, crippled and palsied
and slain,
Ye would send me the spawn of your gutters—Go!
take back your spawn again.

Figure 1: Document with search terms highlighted *in situ*. (Source: Internet Archive)

In contrast, we can find little evidence to suggest similar techniques are in widespread use for databases of music documents. Instead, collections either choose to simply display the page image with no transcription of the source available (e.g., [7, 8]), or transcribe the content into a searchable and manipulable digital format without reference to the original page layout (e.g., [9]). For music documents, where the layout of the symbols can play a critical role in determining the intended interpretation of the music, we posit that a hybrid approach is needed, similar to that demonstrated by Google Books or the Internet Archive.

Critical to the development of these systems is a common standard that allows various systems in an OMR workflow to capture and preserve images, layout, and music semantics. To help inform our development of such a standard, we identified formats used in the textual domain for encoding layout information: The hOCR format, developed as an output format for the Google-sponsored OCRopus document analysis software, and DjVu, a third-party document imaging solution adopted by the Internet Archive for displaying its digitized texts.

2.1.1 hOCR

hOCR [10] is a format that uses standard HTML tags, but embeds OCR-specific information that can be read and manipulated by other OCR software. According to the authors of the hOCR specification, it can be used to encode "layout information, character confidences, bounding boxes, and style information" [11]. For generic HTML rendering software, like a web browser, the OCR-specific information is ignored and the page is rendered without interference.

For the developers of hOCR, HTML was preferred over the definition of a new XML format since the HTML specification already contains many tags for defining document elements, such as headings, tables, paragraphs, and page divisions. Furthermore, the files can be viewed, manipulated, and processed with a wide range of existing tools, such as browsers, editors, converters, and indexers.

To encode information about a page layout, hOCR uses the "class" and "title" attributes of HTML tags. For example, a bounding box outlining a paragraph may be defined as:

```
  <div class="ocr_par" id="par_7"
title="bbox 313 324 733 652">
   ...paragraph text...
  </div>
```

Figure 2: hOCR format defining a paragraph bounding box

The bounding box is given as two sets of pixel co-ordinates corresponding to the upper-left and lower-right corners of the box, relative to the upper-left corner of the image.

Page images corresponding to the text output are linked from the hOCR document with either a local path name or an HTTP URL. The identity and integrity of the image file can be verified by embedding the MD5 checksum of the image file in the hOCR file.

2.1.2 DjVu

DjVu is primarily designed as a highly efficient method of compressing and transferring images and documents. Included in its specification, however, is the ability to include a "hidden" text layer within a binary DjVu file.

The DjVu format specification [12] defines seven different types of document "zones," each featuring a bounding box defined by an offset co-ordinate from a previously defined zone and a given width and height. These zones can define boundaries for pages, columns,

regions, paragraphs, lines, words, or characters. Text is encoded as UTF-8.

2.2 Music Applications

hOCR and DjVu are not the only formats that can provide positional information about text. The popular PDF standard allows for this functionality as well. They serve, however, as examples of existing formats in the textual domain from which we can begin to discuss similar approaches in the musical domain.

To begin building our standard, we define five basic criteria that the OIP format should conform to:

2.2.1 Must be self-contained

Files conforming to the OIP format should be self-contained in a single file. The choices here are between defining a unique binary format, as the DjVu format does, or allowing multiple files to be packaged as a single file.

2.2.2 Must encapsulate multi-page documents

Both hOCR and DjVu encode multiple pages in a single file. hOCR provides only the textual content of those pages and links to externally stored images, while DjVu stores both image and content for multiple pages within a single file.

2.2.3 Must encapsulate notation, images, and metadata

For each page in the document our format must include a page image, the notation content, and, if available, any other metadata about that page. Here, the music domain requires a different approach than the text domain, owing largely to the complexity of encoding music notation over encoding text. In Section 3, we discuss the specific standards chosen for this criteria.

2.2.4 Must use existing standards

Drawing largely on the arguments made by the hOCR developers to justify their use of HTML over creating a new format [10], we specify that, wherever possible, existing standards must be used in preference to creating one. This is especially true for encoding notation, where new formats are introduced every few years, often designed to meet very specific needs, and fall out of use within a few years of being introduced. By using existing standards, we hope to ensure a broader support community beyond our specific application.

2.2.5 Must allow extended information

Beyond the required notation, images, and metadata storage, we see the OIP format as a general-purpose container for storing any extra information about the page content. However, this extra information should be opaque to clients that do not support it, and should not interfere with their ability to read and write OIP files. For example, a specific application could save extended colour-space information about an image in the OIP, available to applications that can use it, but ignored by clients that cannot use it.

3. FORMAT SPECIFICATION

As discussed in the previous sections, we have chosen to combine existing standards into an application profile. In this section we will discuss our specific choices of standards and how they should be combined to create an OIP-compliant file. In the interests of space we will specifically avoid any in-depth explanation about the component standards themselves, since they are freely available for consultation.

3.1 Packaging

An OIP file is, at its most basic representation, a collection of files and folders serialized as a single file. Rather than simply allowing an *ad hoc* method of bundling these files and folders together, we chose to use a very minimal standard for organizing the content of these files.

There are several ways to approach this problem. One type of solution is exemplified by formats such as the Metadata Encoding and Transmission Standard (METS). Data typically represented in binary formats (e.g., images) can be stored, for example, within an XML file by Base64 encoding. A single METS file containing many high-quality images could potentially be many gigabytes in size, however.

A second approach is the file bundle approach. This is used by many formats, including Microsoft's XML-based Office formats (e.g., DOCX) and the Java JAR format. These files are simple file and folder hierarchies containing component files, such as images or text files. They appear as a single file archive by using a well-known file archiving system (e.g., ZIP or TAR). Once these bundles have been uncompressed, read and write operations on the smaller component files can be done directly via the native file system and not on the single monolithic XML file.

The BagIt format is a lightweight file bundling specification. It was created and is maintained by the Library of Congress and the California Digital Library. It is currently in the process of becoming an IETF standard [13]. The name refers to a colloquial rendering of the Enclose and Deposit method [14], also known as the "bag it and tag it" method.

This format defines a simple hierarchy of files and folders, known as a "bag." These can be represented plainly on any computer system as standard files and folders, or they can be converted into a single file using ZIP or TAR packaging.

Minimally, one directory and two files must be present in every bag in order to be considered compliant to the standard. A data directory contains any arrangement of files or folders are stored. This is the bag's "payload." One of the required files is a bagit.txt

file that simply stores the version of the BagIt specification to which that bag conforms and the character encoding used for the metadata files. The second required file is a manifest file listing checksums for each file within the data directory, helping to ensure the integrity and identity of each of the files in the bag. Other optional files are outlined in the BagIt specification [13].

```
<bag-directory>
      |- bagit.txt
      |- manifest-md5.txt
      |- [other optional bagit files]
      |- data
          |- [page 1]
          |      |- [image files]
          |      |- [notation files]
          |      |- [metadata files]
          |- [page 2]
          |      |- [image files]
          |      |- [notation files]
          |      |- [metadata files]
          |- [etc.]
```

Figure 3: A generalized OIP structure.

For the OIP format, we further specify a file hierarchy within the data directory of a bag. A folder is created in the data directory for each page in a multi-page document, allowing the format to accommodate documents of any size. In each page folder, we store files relating to this page. A generalized example of the OIP structure is given in Figure 3.

This does not create incompatibilities with the original BagIt specification, as there is no structure to which the data directory must conform. Software for processing BagIt files will guarantee the integrity and identity of each file in the bag without needing to understand the OIP format.

3.2 Notation

There are many file formats for encoding music notation, but for this specific application we require a format that can encode positional coordinates for every musical element on the page. This eliminates many traditional formats used for notation interchange, such as MIDI. The Notation Interchange File Format (NIFF) fits this requirement, but is no longer actively maintained and is considered an obsolete standard [15]. The SharpEye output format (MRO) [16] also encodes this information and is used by [17] to provide positioning information for musical elements. This format, however, is specifically designed for use with common Western notation (CWN), limiting its usefulness for older or alternative notation systems. MusicXML [18] and NeumesXML [19] focus respectively on CWN and neumed notation, limiting their applicability for a broad range of notation systems.

For OIPs, we recommend the use of the Music Encoding Initiative (MEI) format as a notation encoding scheme. MEI inherits many features of the Text Encoding Initiative (TEI), a format specifically designed for scholars representing original text sources in digital form. MEI can also adequately represent CWN as well as other notation systems [20].

MEI allows for bounding boxes, or "zones," to be defined for a given image and identified with a unique ID. These *id*'s can then be attached to semantically defined musical elements in MEI. A brief example is shown in Figure 4.

```
<facsimile source="s2">
   <surface>
     <graphic xml:id="s2p1"
xlink:href="m000001719_0001.tif"/>
     <zone xml:id="s2p1z1" lrx="0"
lry="0" ulx="0" uly="0"/>
     <zone xml:id="s2p1z2" lrx="1"
lry="1" ulx="10" uly="10"/>
   </surface>
</facsimile>
<!-- ... -->
<measure n="1" xml:id="d1e656"
facs="s2p1z1"/>
```

Figure 4: A MEI-formatted example showing bounding box definitions.

In MEI, the `<graphic>` element defines a link to a page image, while subsequent `<zone>` elements outline regions of this image, identified with a unique `xml:id` attribute. These zones are then used later within the music notation markup, as illustrated in Figure 4 by the `<measure>` tag. It uses the `facs` attribute to link a defined bounding box to a measure definition. This attribute is available to all music notation elements.

3.3 Images

For image formats, we follow the guidelines given in [21] for musical master archival images. These guidelines recommend lossless file encoding formats such as TIFF or PNG for archival formats. While there is no technical reason for not using other formats such as lossy JPEG, we suggest lossless formats to maintain the highest possible image quality.

One issue we have not yet addressed is how to reconcile the differences between an original image and an image file that has been cropped, de-skewed, and prepared for processing by an OMR package. Since any geometric manipulation will affect the co-ordinates of the musical elements on the page, it would be difficult to automatically reconcile the position of musical elements in an original image, when the notation was extracted

using a processed image. This becomes especially important when considering the OIP format as an interchange format between multiple OMR systems, each of which may use different image processing techniques, or even require that certain elements of an image be removed prior to recognition, such as staff lines.

To reconcile this, we specify that, at a minimum, an OIP should contain the original page image, and a page image that the OMR system used during the recognition stage prior to removing any musically relevant elements. The additional inclusion of any intermediary images used by OMR software is permitted, but not required. For an OIP that has been processed by multiple OMR packages, each package should save its source image and recognized notation in MEI.

3.4 Metadata

MEI has the facilities to capture bibliographic, analytic, and editorial metadata. There is also the possibility that other metadata can be captured and stored within the file hierarchy. While we do not require any further metadata beyond what can be captured in MEI, we do not prevent the inclusion of other files with metadata formats describing, for example, detailed image processing techniques, historical and archival information, or library-specific local information.

4. APPLICATIONS

We have formulated the OIP format as an interchange format between multiple elements of an OMR workflow, from digitization through recognition, and finally into a delivery format specifically designed to capture and transfer page layout along with the semantic music content. In this section, we identify three specific applications where OIP files can be implemented as a standardized format for constructing tools useful for music scholars and OMR research.

4.1 Diplomatic Facsimiles

While there is some disagreement on the actual definition of the term, we define diplomatic facsimile as "a visualization (on-screen or in print) from the digital transcription of a source artifact, such that it has the same semantic content as the source, and its glyphs and layout are similar to the original source" [22].

For notation styles outside of the CWN tradition, a diplomatic facsimile provides the ability to transcribe a musical source with its original layout and symbols, without interpreting it by using modern music notation symbols. Barton, Caldwell, and Jeavons provide an excellent overview of the importance of this distinction [23]. Diplomatic facsimiles also allow libraries and archives to withhold distribution of original images due to copyright restrictions, while simultaneously allowing scholars access to a faithful electronic reproduction of the original musical content, including precise positioning for each musical element in the document.

4.2 Online Music Document Databases

An online database of music documents, similar to Google Books or the Internet Archive's display of textual documents, could be constructed with OIP as a source format for these documents. In an OMR workflow, OIP files would serve as an interchange format between the OMR software and a database system designed to organize, index, and display these documents.

As mentioned in the introduction, music scholars often use visual cues in the layout of a page of music to determine how a piece of music might be performed, or where it came from. Viewing these documents in their original form, while still making them available for content-specific searching and indexing, would provide a valuable research tool for many music scholars.

Furthermore, an online music document database could highlight relevant musical phrases matching a user's query, displayed as an invisible layout on the original image. Advanced computer processing could potentially provide links between similar passages within, or across, musical pieces, allowing users to navigate a document by musical phrase.

4.3 Distributed Optical Music Recognition

The extent, variety, and variability of musical symbols pose a unique problem to optical music recognition software. These symbols encompass indications of pitch, duration, dynamics, tempo, or performer interpretation (e.g., turns and trills). Different printing practices or fonts also introduce variations in these shapes.

Adaptive OMR (AOMR) software attempts to account for this variability by using machine-learning methods for understanding and interpreting new shapes, or variations on known shapes. These systems are often trained using human annotators or correctors, who provide a system with the correct musical interpretation of a graphical shape [24].

This training process is often the most tedious and expensive part of the OMR process. Developing training sets of sufficient quantity and variety is an expensive and labour-intensive process. Similarly, a poorly trained recognition system will require more human intervention, leading to lower overall throughput for any digitization and recognition initiative. For large digitization projects, this can have a significant impact on the overall cost of digitizing these materials [25].

With a common interchange format, however, these data sets could be built cumulatively. As new pieces of music are recognized and corrected, this work can be saved and used to train other AOMR clients with no further intervention by a human annotator.

Perhaps more importantly, this concept can be used to build a distributed global network of AOMR clients. Sharing training data with other networked OMR clients

would allow them to build their recognition models using data previously provided by other members of the network. For example, an archive that has provided a data set of examples from a 16th-Century Italian music printer can make this data set available immediately to other members of the network, allowing these clients to immediately re-train their recognition systems to take advantage of this new data and increase their accuracy on this particular repertoire.

5. CURRENT AND FUTURE WORK

To date, we have finished the initial release of an open source Python library for reading and writing BagIt files, available at [26]. This is part of a larger project to develop a distributed optical music recognition system, a networked collection of adaptive OMR clients.

6. REFERENCES

[1] Blostein, D., and L. Haken. 1991. Justification of printed music. *Communications of the ACM* 34 (3): 88–99.

[2] Warmington, F. 1999. A survey of scribal hands in the manuscripts. In *The treasury of Petrus Alamire: Music and art in Flemish court manuscripts 1500–1535*, ed. H. Kellman, 41–52. Amsterdam: Ludion Ghent.

[3] Luth, N. 2002. Automatic identification of music notations. In *Proceedings of the International Conference on Web Delivering of Music (WEDELMUSIC),* 203–10.

[4] Hall, P. 1997. *A view of Berg's Lulu through the autograph sources*. Berkeley: University of California Press.

[5] Google Books. http://books.google.com [Accessed 23 March 2010].

[6] Internet Archive. http://www.archive.org [Accessed 23 March 2010].

[7] Schubert Autographs. http://www.schubert-online.at/activpage/index_en.htm [Accessed 23 March 2010].

[8] Digital Image Archive of Medieval Music. http://www.diamm.ac.uk [Accessed 23 March 2010].

[9] The Computerized Mensural Music Editing Project. http://www.cmme.org [Accessed 23 March 2010].

[10] Breuel, T. M., and U. Kaiserslautern. 2007. The hOCR microformat for OCR workflow and results. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 1063–7.

[11] hOCR-tools. http://code.google.com/p/hocr-tools [Accessed 23 March 2009].

[12] Celartem/Lizartech DjVu format reference. http://djvu.org/docs/DjVu3Spec.djvu [Accessed 23 March 2010].

[13] The BagIt File Packaging Format. https://svn.tools.ietf.org/html/draft-kunze-bagit-04 [Accessed 23 March 2010].

[14] Tabata, K., T. Okada, M. Nagamori, T. Sakaguchi, and S. Sugimoto. 2005. A collaboration model between archival systems to enhance the reliability of preservation by an enclose-and-deposit method. In *Proceedings of the International Web Archiving Workshop.*

[15] Castan, G. 2009. NIFF. http://www.music-notation.info/en/formats/NIFF.html. [Accessed 29 May 2010].

[16] Jones, G. OMR engine output file format. http://www.visiv.co.uk/tech-mro.htm [Accessed 29 May 2010].

[17] Kurth, F., D. Damm, C. Fremerey, M. Müller, and M. Clausen. 2008. A framework for managing multimodal digitized music collections. In *Research and advanced technology for digital libraries*, 334–45. Berlin: Springer.

[18] Good, M. 2009. Using MusicXML 2.0 for music editorial applications. In *Digitale Edition zwischen Experiment und Standardisierung*, ed. P. Stadler and J. Veit, 157–74. Tübingen: Max Niemeyer.

[19] Barton, L. 2002. The NEUMES project: Digital transcription of medieval chant manuscripts. In *Proceedings of the Web Delivering of Music (WEDELMUSIC)*, 211–8.

[20] Roland, P. 2009. MEI as an editorial music standard. In *Digitale Edition zwischen Experiment und Standardisierung*, ed. P. Stadler and J. Veit, 175–94. Tübingen: Max Niemeyer.

[21] Riley, J., and I. Fujinaga. 2003. Recommended best practices for digital image capture of musical scores. *OCLC Systems and Services* 19 (2): 62–9.

[22] Glossary of terms used by the NEUMES project. http://www.scribeserver.com/NEUMES/help/glossary.htm [Accessed 23 March 2010].

[23] Barton, L., J. Caldwell, and P. Jeavons. 2005. E-library of medieval chant manuscript transcriptions. In *Proceedings of the Annual Joint Conference on Digital Libraries (JCDL)*, 320–39.

[24] Fujinaga, I. 1997. *Adaptive optical music recognition*. PhD diss., McGill University.

[25] Pugin, L., J. Burgoyne, and I. Fujinaga. 2007. Reducing costs for digitising early music with dynamic adaptation. In *Proceedings of the European Conference on Digital Libraries (ECDL)*, 471–4.

[26] Hankinson, A. 2010. PyBagIt 1.0 documentation. http://www.musiclibs.net/pybagit [Accessed 29 May 2010].

ARE TAGS BETTER THAN AUDIO FEATURES? THE EFFECT OF JOINT USE OF TAGS AND AUDIO CONTENT FEATURES FOR ARTISTIC STYLE CLUSTERING

Dingding Wang
School of Computer Science
Florida International University
Miami, FL USA
dwang003@cs.fiu.edu

Tao Li
School of Computer Science
Florida International University
Miami, FL USA
taoli@cs.fiu.edu

Mitsunori Ogihara
Department of Computer Science
University of Miami
Coral Gables, FL USA
ogihara@cs.miami.edu

ABSTRACT

Social tags are receiving growing interests in information retrieval. In music information retrieval previous research has demonstrated that tags can assist in music classification and clustering. This paper studies the problem of combining tags and audio contents for artistic style clustering. After studying the effectiveness of using tags and audio contents separately for clustering, this paper proposes a novel language model that makes use of both data sources. Experiments with various methods for combining feature sets demonstrate that tag features are more useful than audio content features for style clustering and that the proposed model can marginally improve clustering performance by combing tags and audio contents.

1. INTRODUCTION

The rapid growth of music the Internet both in quantity and in diversity has raised the importance of music style analysis (e.g., music style classification and clustering) in music information retrieval research [10]. Since a music style is generally included in a music genre (e.g., the style Progressive Rock within the genre of Rock) a style provides finer categorization of music than its enclosing genre. Also, for much the same reason that all music in a single genre has some commonality, all music in a single style has some commonality belonging to a same style, and the degree of commonality is stronger within a style than within its enclosing genre. These properties suggest that by way of appropriate music analysis, it is possible to computationally organize music sources into not only musicologically meaningful groups but also into hierarchical clusters that reflect style and genre similarities. Such organizations are likely to enable efficient browsing and navigation of music items.

Much of the past work on music style analysis methods is based solely on audio contents and various feature extraction methods have been tested. For example, [32] presents a study on music classification using short-time analysis along with data mining techniques to distinguish among five music styles. Pampalk et al. [17] combine different similarity sources based on fluctuation patterns and use a nearest neighbor classifier to categorize music items. More recently Chen and Chen [3] use long-term and short-term features that represent the time-varying behavior of music and apply support vector machines (SVM) to classify music into genres. Although these audio-content-based classification methods are successful, music style classification and clustering are difficult problems to tackle, in part because music style classes are more numerous than music genres and thus computation quickly reaches a limit in terms of the number of styles to classify music into. One then naturally asks whether adding non-audio features push style classification/clustering beyond the limit of audio-feature-based analysis.

Fortunately, the rapid development of web technologies has made available a large quantity of non-acoustic information about music, including lyrics and social tags, latter of which can be collected by a variety of approaches [24]. There has already been some work toward social tag based music information retrieval [1,11,13,16,23]. For example, Levy and Sandler [16] demonstrate that the co-occurrence patterns of words in social tags are highly effective in capturing music similarity, Bischoff et al. [1] discuss the potential of different kinds of tags for improving music search, and Symeonidis et al. [23] propose a music recommendation system by performing latent semantic analysis and dimensionality reduction using the higher order SVD technique on a user-tag-item tensor.

In this paper we consider social tags as the source of non-audio information. We naturally ask whether we can effectively combine the non-audio and audio information sources to improve performance of music retrieval. Some prior work has demonstrated that using both text and audio features can improve the ranking quality in music search systems. For example, Turnbull et al. [25] successfully combine audio-content features (MFCC and Chroma) with social tags via machine learning methods for music searching and ranking. Also, Knees et al. [12] incorporate audio contents into a text-based similarity ranking process.

However, few efforts have been made to examine the effect of combining tags and audio-contents for music style analysis. We thus the question of, given tags and representative pieces for each artist of concern, whether the tags and the audio-contents of the representative pieces complement each other with respect to artist style clustering, and if so, how efficiently those pieces of information can be combined.

In this paper, we study the above questions by treating the artist style clustering problem as an unsupervised clustering problem. We first apply various clustering algorithms using tags and audio features separately, and examine the usefulness of the two data sources for style clustering. Then we propose a new tag+content (TC) model for integrating tags and audio contents. A set of experiments is conducted on a small data set to compare our model with other methods, and then we explore whether combining the two information sources can improve the clustering performance or not.

The rest of this paper is organized as follows. In Section 2 we briefly discuss the related work. In Section 3 we introduce our proposed TC model for combining tags and contents for artist style clustering. We conduct comprehensive experiments on a real world dataset and the experimental results are presented in Section 4. Section 5 concludes.

2. RELATED WORK

Audio content based automatic music analysis (clustering, classification, and similarity search in particular) is one of the most important topics in music information retrieval. The most widely used audio features are timbral texture features (see, e.g., [26]), which usually consist of Short Term Fourier Transform (STFT) and Mel-Frequency Cepstral Coefficients (MFCC) [20]. Researchers have applied various data mining and statistically methods on these features for classifying or clustering artists, albums, and songs (see, e.g., [3, 5, 18, 19, 26]).

Music social tags have recently emerged as a popular information source for curating music collections on the web and for enabling visitors of such collections to express their feelings about particular artists, albums, and pieces. Social tags are free-text descriptions of any length (though in practice there sometimes is a limit in terms of number of characters) with no restriction on the words that are used. Social tags thus can be as simple as a single word and as complicated as a long, full sentence. Popular short tags include *heavy rock, black metal, and indie pop* and long tags can be like "I love you baby, can I have some more?"

As can be easily seen social tags are not as formal as descriptions that experts such as musicologists provide. However, by collecting a large number of tags for one single piece of music or for one single artist, it seems possible to gain understanding of how the song or the artist is received by the general listeners. As Lamere and Pampalk point out [13] social tags are widely used to enhance simple search, similarity analysis, and clustering of music items [13]. Lehwark, Risi, and Ultsi [15] use Emergent-Self-Organizing-Maps (ESOM) and U-Map techniques on tagged music data to conduct clustering and visualization in music collections. Levy and Sandler [16] apply latent semantic dimension reduction methods to discover new semantics from social tags for music. Karydisi et al. [11] propose a tensor-based algorithm to cluster music items using 3-way relational data involving song, users, and tags.

In the information retrieval community a few attempts have been made to complement document clustering using user-generated tags as an additional information source (see, e.g., [21]). In such work the role that social tags play is only supplementary because the texts appearing in the original data are, naturally, highly more informative than tags.

The situation in the MIR community seems different from this and the use of tags seems to show much stronger promise. This is because audio contents, which are the standard source of information, have to go through feature extraction for syntactic or semantic understanding and thus the distance between the original data source and the tag in terms of informativeness appears to be much smaller in MIR than in IR.

There has been some work exploring the effectiveness of joint use of the two types of information sources for retrieval, including including the work in [25] and [12] where audio contents and tags are combined for searching and ranking and the work in [30] that attempts to integrate audio contents and tags for multi-label classification of music styles. These prior efforts are concerned with supervised learning (i.e., classification) while the present paper is concerned with unsupervised learning (i.e., clustering).

3. TAG+CONTENT MODEL (TC)

Here we present our novel language model for integrating tags and audio contents and how to use the model for artistic style clustering.

3.1 The Model

Let $\mathcal{A}$ be the set of artists of interest, $\mathcal{S}$ the set of styles of interest, and $\mathcal{T}$ the set of tags of interest. We assume that for each artist, for each style, and for each artist-style pair, its tag set (as a multiset in which same elements may be repeated more than once) is generated by mutually independent selections. That is, for each artist $a \in \mathcal{A}$ and for each nonempty set of tags $t = (t_1, \ldots, t_n)$, $t_1, \ldots, t_n \in \mathcal{T}$, we define the language model, $p(t \mid a)$, by

$$p(t \mid a) = \prod_{i=1}^{n} p(t_i \mid a)$$

Similarly, for each style $s \in \mathcal{S}$, we define its language model $p(t \mid s)$, by

$$p(t \mid s) = \prod_{i=1}^{n} p(t_i \mid s)$$

Although we might want to consider the artist-style joint language model $p(t \mid a, s)$, we assume that the model is

dictated only by the style and that it is independent of the artist. Thus, we assume

$$p(t \mid a, s) = p(t \mid s)$$

for all tags $t \in \mathcal{T}$. Then the artist language model can be decomposed into several common style language models:

$$p(t \mid a) = \sum_{s \in \mathcal{S}} p(t \mid s) p(s \mid a).$$

Instead of directly choosing one style for artist a, we assume that the style language models are mixtures of some models for the artists linked to a, i.e.,

$$p(s \mid a) = \sum_{b \in \mathcal{A}} p(s \mid b) p(b \mid a),$$

where b is an artist linked to artist a. Combining these yields the following model:

$$p(\vec{t} \mid a) = \prod_{i=1}^{n} \sum_{s \in \mathcal{S}} \sum_{b \in \mathcal{A}} p(t_i \mid s) p(s \mid b) p(b \mid a).$$

We use the empirical distribution of the observed artists similarity graph for $p(b \mid a)$ and let $\mathbf{B}_{b,a} = \tilde{p}(b \mid a)$. The model parameters are $(\mathbf{U}, \mathbf{V})$, where

$$\mathbf{U}_{t,s} = p(t \mid s), \quad \mathbf{V}_{b,s} = p(b \mid s).$$

Thus, $p(t_i \mid a) = [\mathbf{U}\mathbf{V}^\top \mathbf{B}]_{t,a}$.

The artist similarity graph can be obtained using methods described in Section 3.2. Now we take the Dirichlet distribution, the conjugate prior of multinomial distribution, as the prior distribution of $\mathbf{U}$ and $\mathbf{V}$. The parameter estimation is maximum a posteriori (MAP) estimation. The task is

$$\mathbf{U}, \mathbf{V} = \arg\min_{\mathbf{U},\mathbf{V}} \ell(\mathbf{U}, \mathbf{V}), \tag{1}$$

where $\ell(\mathbf{U}, \mathbf{V}) = \mathrm{KL}\left(\mathbf{A} \| \mathbf{U}\mathbf{V}^\top \mathbf{B}\right) - \ln \Pr(\mathbf{U}, \mathbf{V})$.

Using an algorithm similar to the nonnegative matrix factorization (NMF) algorithm in [14], we obtain the following updating rules:

$$\begin{aligned} \mathbf{U}_{ts} &\leftarrow \mathbf{U}_{ts} \left[\mathbf{C}\mathbf{B}^\top \mathbf{V}\right]_{ts} \\ \mathbf{V}_{bs} &\leftarrow \mathbf{V}_{bs} \left[\mathbf{B}\mathbf{C}^\top \mathbf{U}\right]_{bs} \end{aligned}$$

where $\mathbf{C}_{ij} = \mathbf{A}_{ij} / [\mathbf{U}\mathbf{V}^\top \mathbf{B}]_{ij}$. The computational algorithm is given in Section 3.3.

3.2 Artist Similarity Graph Construction

Based on the audio content features, we can construct the artist similarity graph using one of the following popular methods, which is due to Zhu [33].

ϵ NN graphs A strategy for artist graph construction is the ϵ-nearest neighbor algorithm based on the distance between the feature values of two artists. For a pair of artists i and j, if the distance $d(i, j)$ is at most ϵ, draw an edge between them. The parameter ϵ controls the neighborhood radius. For the distance measure d, the Euclidean distance is used throughout the experiments.

exp-weighted graphs This is a continuous weighting scheme where $W_{ij} = \exp(-d(i, j)^2 / \alpha^2)$. The parameter α controls the decay rate and is set to 0.05 empirically.

3.3 The Algorithm

Algorithm 1 is our method for estimating the model parameters.

Algorithm 1 Parameter Estimation

Input: $\mathbf{A}$: tag-artist matrix.
$\mathbf{B}$: artist-artist relation matrix;
Output: $\mathbf{U}$: tag-style matrix;
$\mathbf{V}$: artist-style matrix.

begin
1. **Initialization:**
Initialize $\mathbf{U}$ and $\mathbf{V}$ randomly,
2. **Iteration:**
repeat
2.1 Compute $\mathbf{C}_{ij} = \mathbf{A}_{ij} / [\mathbf{U}\mathbf{V}^\top \mathbf{B}]_{ij}$;
2.2 Assign $\mathbf{U}_{ts} \leftarrow \mathbf{U}_{ts} \left[\mathbf{C}\mathbf{B}^\top \mathbf{V}\right]_{ts}$,
2.3 Compute $\mathbf{C}_{ij} = \mathbf{A}_{ij} / [\mathbf{B}\mathbf{U}\mathbf{V}^\top]_{ij}$;
2.4 Assign $\mathbf{V}_{bs} \leftarrow \mathbf{V}_{bs} \left[\mathbf{B}\mathbf{C}^\top \mathbf{U}\right]_{bs}$,
until convergence
3. **Return** $\mathbf{V}$
end

3.4 Relations with Other Models

The TC model uses mixtures of some existing base language models as topic language models. The model is different with some well-known topic models such as Probabilistic Latent Semantic Indexing (PLSI) [8] or Latent Dirichlet Allocation (LDA) [2] since they assume the topic distribution of each object is independent of those of others. However, this assumption does not always hold in practice since in music style analysis, artists (as well as songs) are usually related to each other in certain ways. Our TC model incorporates an external information source to model such relationships among artists. Also, when the base matrix $\mathbf{B}$ is an identity matrix, this model is identical to PLSI (or LDA), and the algorithm is the same as the NMF algorithm with Kullback-Leibler (KL) divergence loss [6, 29].

4. EXPERIMENTS

4.1 Data Set

For experimental purpose, we use the data set in [30]. The data set consists of 403 artists and one representative song per artist. The style and tag descriptions are obtained respectively from All Music Guide and Last.fm, as described below.

4.1.1 Music Tag Information

Tags were collected from Last.fm (http://www.last.fm). A total of 8,529 tags were collected. The number of tags for an artist ranged from 3 to 100. On average an artist had 89.5 tags. Note that, the tag set is a multiset in that the same tag may be assigned to the same artist more than once. For example, Michael Jackson was assigned "80s" for 453 times.

4.1.2 Audio Content Features

For each song we extracted 30 seconds of audio after the first 60 seconds. Then from each of the 30-second audio clips, we extracted 12 timbral features using short-term Fourier transform following the method described in [27]. The twelve features are based on Spectral Centroid, Spectral Rolloff, and Spectral Flux. For each of these three spectral dynamics, we calculate the mean and the standard deviation over a sliding window of 40 frames. Then from these means and variances we compute the mean and the standard deviation across the entire 30 seconds, which results in $2 \times 2 \times 3 = 12$ features. We mention here that we actually began our exploration with a much larger feature set of size 80, which included STFT, MFCC, and DWCH, but in an attempt to improve results all the features but STFT were consolidated which was consistent with the observations in [9].

4.1.3 Style Information

Style information was collected from All Music Guide (http://www.allmusic.com). All Music Guide's data are all created by musicologists. Style terms are nouns like Rock & Roll, Greek Folk, and Chinese Pop as well as adjectives like Joyous, Energetic, and New Romantic. Styles for each artist/track are different from the music tags described in the above, since each style name appears only once for each artist. We group the styles into five clusters, and assign each artist to one style cluster. In the experiments, the five groups of styles are: (1) Dance-Pop, Pop/Rock, Club/Dance, etc., consisting of 100 artists including *Michael Jackson*; (2) Urban, Motown, New Jack Swing, etc., consisting of 72 artists including *Bell Biv DeVoe*; (3) Free Jazz, Avant-Garden, Modern Creative, etc., consisting of 51 artists including *Air Band*; (4) Hip-Hop, Electronica, and etc., consisting 70 artists including *Afrika Bambaataa*; (5) Heavy Metal, Hard Rock, etc., consisting of 110 artists including *Aerosmith*.

4.2 Baselines

We compare our proposed method with several state-of-the-art clustering methods including K-means, spectral clustering (Ncuts) [31], and NMF [14]. For each clustering method, we perform it on two data matrices, i.e., the tag-artist matrix and the content-artist matrix, respectively. We also perform them on an artist similarity graph which is the linear combination of two similarity graphs generated based on tags and contents respectively using the graph construction method described in Section 3.2. NMF is not suitable for symmetric similarity matrices, there exists its symmetric matrix version, SNMF [28]. We use SNMF to deal with the artist similarity matrix. We also use PHITS-PLSI, a probabilistic model [4] which is a weighted sum of PLSI and PHITS, to integrate tag and audio content information for artist clustering. The summary of the baseline methods is listed in Table 4.2.

clustering methods	tags only	content only	both
K-means	√	√	√
Ncuts	√	√	√
NMF	√	√	
SNMF			√
PHITS-PLSA			√

Table 1. The implemented baseline methods.

	K-means	Ncuts	NMF
Accuracy	0.2953	0.4119	0.4020
NMI	0.0570	0.1166	0.1298

Table 2. Clustering results using tag information only.

4.3 Evaluation Methods

To measure the clustering quality, we use accuracy and normalized mutual information (NMI) as performance measures.

- Accuracy measures the relationship between each cluster and the ground truth class assignments. It is the total matching degree between all pairs of clusters and classes. The greater accuracy, the better clustering performance.
- NMI [22] measures the amount of statistical information shared by two random variables representing cluster assignment and underlying class label.

4.4 Experimental Results

4.4.1 Tags-only or Content-only

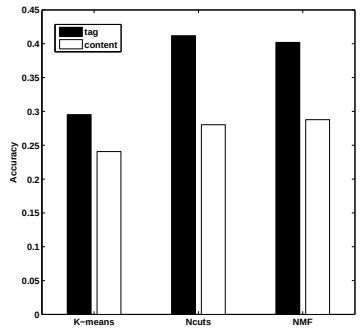
Tables 2 and 3 respectively show the clustering performance using tag information only and the performance using content features only. We observe that the tags are more effective than the audio content features for artist style clustering. Figure 1 better illustrates this observation.

4.4.2 Combining Tags and Content

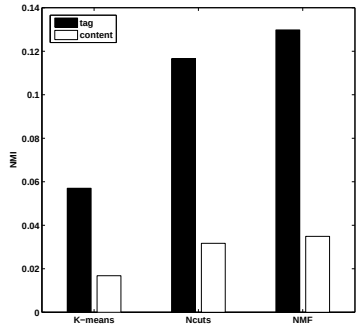
Table 4.4.2 show the performance of different clustering methods using both tag and content information. Since the

	K-means	Ncuts	NMF
Accuracy	0.2407	0.2803	0.2878
NMI	0.0168	0.0317	0.0349

Table 3. Clustering results using content features only.



(a) Accuracy



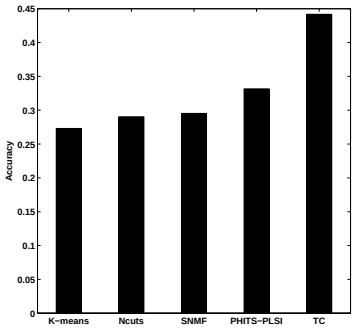
(b) NMI

Figure 1. Clustering performance using tag or content information.

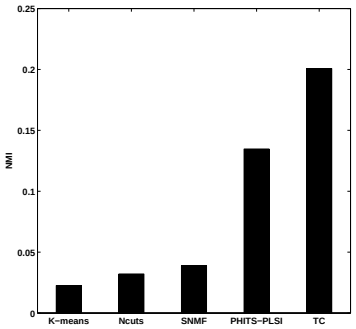
first three clustering algorithms are originally designed for clustering one data matrix, we first construct an artist similarity graph as follows. (1) We compute the pairwise Euclidean distances of artists using the tag-artist matrix (normalized by tags (rows)) to obtain a symmetric distance matrix d_t, and another distance matrix d_c can be calculated in the similar way using the content- artist matrix. (2) Since d_t and d_c are in the same scale, we can simply combine them linearly to obtain the pairwise artist distance. (3) The corresponding artist similarity graph can be constructed using the strategies introduced in Section 3.2. Once the artist similarity graph is generated, the clustering can be conducted using any clustering method. Since both PHITS-PLSI and our proposed method are designed to combine two types of information, we can directly use the tag-artist matrix as the original data matrix, and the similarity graph is constructed based on content features. Figure 2 illustrates the results visually.

From the results, we observe the following:

- The artist clustering performance is not necessarily improved by incorporating content features. This means that the tags are more informative than contents for clustering artist styles.
- Advanced methods, e.g. PHITS-PLSI and our proposed method, can naturally integrate different types of information and they outperform other traditional clustering methods. In addition, our proposed method outperforms PHITS-PLSI because PHITS-PLSI is more suitable for incorporating explicit link information while our method is more suitable for handling implicit links (graph).
- Continuous similarity graph construction such as exp-weighted method performs better than discrete methods, e.g. ϵ NN.
- Our proposed method with combined tags and contents using ϵ NN graph construction outperforms all the methods using only tag information. This demonstrates our model is effective for combining different sources of information, although the content features do not contribute much.



(a) Accuracy



(b) NMI

Figure 2. Clustering performance combining tags and contents.

5. CONCLUSION

In this paper, we study artistic style clustering based on two types of data sources, i.e., user-generated tags and audio content features. A novel language model is also proposed to make use of both types of information. Experimental results on a real world data set demonstrate that tag information is more effective than music content information for artistic style clustering, and our model-based method can marginally improve the clustering performance by combining tags and contents. However, other simple combination methods fail to enhance the clustering results by incorporating content features into tag-based analysis.

		K-means	Ncuts	SNMF	PHITS-PLSI	TC
ϵ NN graph	Acc	0.2680	0.2804	0.2630	0.3152	0.3648
	NMI	0.0193	0.0312	0.0261	0.0709	0.1587
exp-weighted graph	Acc	0.2730	0.2903	0.2953	0.3316	0.4417
	NMI	0.0226	0.0321	0.0389	0.1347	0.2008

Table 4. Clustering results combining tags and content.

6. ACKNOWLEDGMENT

The work is partially supported by the FIU Dissertation Year Fellowship, NSF grants IIS-0546280, CCF-0939179, and CCF-0958490, and an NIH Grant 1-RC2-HG005668-01.

7. REFERENCES

[1] K. Bischoff, C. Firan, W. Nejdl, and R. Paiu: "Can all tags be used for search?," *Proceedings of CIKM*, 2008.

[2] D. Blei, A. Ng, and M. Jordan: "Latent Dirichlet allocation," *NIPS*, 2002.

[3] S. Chen and S. Chen: "Content-based music genre classification Usinu timbral feature vectors and support vector machine," *Proceedings of ICIS*, 2009.

[4] D. Cohn and T. Hofmann: "The missing link - a probabilistic model of document content and hypertext connectivity," *NIPS*, 2000.

[5] H. Deshpande, R. Singh, and U. Nam: "Classification of music signals in the visual domain," *Proceedings of the the COST-G6 Conference on Digital Audio Effects*, 2001.

[6] C. Ding, T. Li, and W. Peng: "On the equivalence between Non-negative Matrix Factorization and Probabilistic Latent Semantic Indexing,". *Comput. Stat. Data Anal.*, 52(8):3913-3927.

[7] C. Ding, T. Li, W. Peng, and H. Park. "Orthogonal nonnegative matrix tri-factorizations for clustering," *SIGKDD*, 2006.

[8] T. Hofmann: "Probabilistic latent semantic indexing," *SIGIR*, 1999.

[9] T. Li, M. Ogihara, and Q. Li: "A comparative study on content-based music genre classification," *SIGIR*, 2003.

[10] T. Li and M. Ogihara: "Towards intelligent music information retrieval," *IEEE Transactions on Multimedia*, 8(3):564-575, 2006.

[11] I. Karydis, A. Nanopoulos, H. Gabriel, and M. Spiliopoulou: "Tag-aware spectral clustering of music items," *ISMIR*, pp. 159–164, 2009.

[12] P. Knees, T. Pohle, M. Schedl, D. Schnitzer, K. Seyerlehner, and G. Widmer: "Augmenting text-based music retrieval with audio similarity," *ISMIR*, 2009.

[13] P. Lamere and E. Pampalk: "Social tags and music information Retrieval," *ISMIR*, 2008.

[14] D. Lee and H. Seung: "Algorithms for non-negative matrix factorization," *NIPS*, 2001.

[15] P. Lehwark, S. Risi, and A. Ultsch: "Data analysis, machine learning and applications," in *Visualization and Clustering of Tagged Music Data*, pp. 673–680. Springer Berlin Heidelberg, 2008.

[16] M. Levy and M. Sandler: "Learning latent semantic models for music from social tags" *Journal of New Music Research*, 37:137–150, 2008.

[17] E. Pampalk, A. Flexer, and G. Widmer: "Improvements of audio-based music similarity and genre classificaton," *ISMIR*, 2005.

[18] W. Peng, T. Li, and M. Ogihara: "Music clustering with constraints," *ISMIR*, 2007.

[19] D. Pye: "Content-based methods for managing electronic music," *ISCASSP*, 2000.

[20] L. Rabiner and B. Juang: *Fundamentals of Speech Recognition*, Prentice-Hall, NJ, 1993.

[21] D. Ramage, P. Heymann, C. Manning, and H. Garcia: "Clustering the tagged web," *ACM International Conference on Web Search and Data Mining*, 2009.

[22] A. Strehl and J. Ghosh: "Clustering ensembles - a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, 3:583-617, 2003.

[23] P. Symeonidis, M. Ruxanda, A. Nanopoulos, and Y. Manolopoulos: "Ternary semantic analysis of social tags for personalized music Recommendation," *ISMIR*, 2008.

[24] D. Turnbull, L. Barrington, and G. Lanckriet: "Five approaches to collecting tags for music," *ISMIR*, 2008.

[25] D. Turnbull, L. Barrington, M. Yazdani, and G. Lanckriet: "Combining audio content and social context for semantic music discovery," *SIGIR*, 2009.

[26] G. Tzanetakis and P. Cook: "Musical Genre Classification of Audio Signals," *IEEE Transactions on Speech and Audio Processing*, 10:5, 2002.

[27] G. Tzanetakis: "Marsyas submissions to MIREX 2007," *MIREX 2007*.

[28] D. Wang, S. Zhu, T. Li, and C. Ding: "Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization," *SIGIR*, 2008.

[29] D. Wang, S. Zhu, T. Li, Y. Chi, and Y. Gong: "Integrating clustering and multi-document summarization to improve document understanding," in *CIKM*. pp. 1435-1436, 2008.

[30] F. Wang, X. Wang, B. Shao, T. Li, and M. Ogihara: "Tag integrated multi-label music style classification with hypergraph," in *ISMIR*, pp. 363–368, 2008.

[31] J. Shi and J. Malik: "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2002.

[32] Y. Zhang and J. Zhou: "A study on content-based music Classification," *IEEE Signal Processing and Its Applications*, 2003.

[33] X. Zhu: "Semi-supervised learning with graphs," *Doctoral Thesis*, Carnegie Mellon University, 2005.

AUTOMATED MUSIC SLIDESHOW GENERATION USING WEB IMAGES BASED ON LYRICS

Shintaro Funasawa† Hiromi Ishizaki‡ Keiichiro Hoashi‡
Yasuhiro Takishima‡ Jiro Katto†
†Waseda University ‡KDDI R&D Laboratories Inc.
shint@katto.comm.waseda.ac.jp, {ishizaki,hoashi,takisima}@kddilabs.jp
katto@waseda.jp

ABSTRACT

In this paper, we propose a system which automatically generates slideshows for music, by utilizing images retrieved from photo sharing web sites, based on query words extracted from song lyrics. The proposed system consists of two major steps: (1) query extraction from song lyrics, (2) image selection from web image search results. Moreover, in order to improve the display duration of each image in the slideshow, we adjust image transition timing by analyzing the duration of each lyric line in the input song. We have conducted subjective evaluation experiments, which prove that the proposal can generate impressive music slideshows for any input song.

1. INTRODUCTION

Music video, i.e., a series of visual content displayed with music, is a popular and effective way to increase the entertainability of the music listening experience. The synergetic effect generated by combining visual and audio signals is known as the sympathy phenomenon in the field of psychology [1]. While it is easy to enjoy music videos created by others (usually by experts), it is extremely difficult for common users to create music video by themselves. Namely, the cost to collect video and/or image material that is suitable for the selected music is expensive. Furthermore, the editing process to fuse the material with music also requires much intensive effort.

An important factor which reflects the image of a song is its lyrics. Many songs have lyrics which impressively represent its visual scenery, which are difficult to be extracted from their acoustic features. Numerous research efforts focusing on song lyric analysis have been presented recently. For example, extraction of song genre, topic and mood, have been investigated in recently presented work [2-5].

This paper proposes a system which generates a music slideshow automatically, by using images retrieved from the web based on query words that are derived from song lyrics. By utilizing images from the web, which provides an abundant and diverse resource of images, our proposal is able to generate slideshows of wide variety, without applying any burden to the user. In order to generate such a system, we focus on two major issues. One is the automatic extraction of words from the lyrics that are appropriate for web image search. The other is to select an optimal image to be displayed with each lyric line, from the set of candidate images obtained by web image search.

In this paper, we firstly propose a query extraction method from song lyrics based on the frequency of social tags attached to retrieved images. This method is effective to generate appropriate queries to avoid the retrieval of images that are unsuitable for slideshows. Secondly, we propose a method which selects images from the search results, based on entire impression of the song lyrics. This method is expected to increase the unity among the images within the slideshow. Moreover, we apply a method to adjust image transition time within the slideshow, by analysis of the duration time per lyric line. Subjective user evaluations will show that the proposal is capable of generating high-quality music slideshows automatically.

2. RELATED WORK

Mainly, two types of methods have been proposed for automatic generation of visual content from music. One is to generate visual contents using personal videos and/or photos [6-8], and the other is to utilize web images [9][10]. An advantage for using personal videos/photos is that the resulting slideshow will be more familiar to the user. However, in order to generate high-quality slideshows, a sufficient amount of personal material must be prepared, which is a heavy burden for casual users.

The web image-based approach has two major issues: query selection and image selection. Appropriate selection of query words is expected to be effective for the retrieval of images for slideshows. However, existing works [9][10] have utilized naive methods for query word selection, such as stop word rejection, and selection of specific parts of speech (e.g., nouns). Using values to measure the significance of words, e.g. TF*IDF, can be utilized to select query words which are significant within the lyrics. However, it is unclear whether or not such measures are appropriate to select query words for web image search to generate slideshows.

For the image selection problem, an idea has been proposed in [10] to select images containing human faces and

outdoor scenery. However, no evidence has been provided that such images are optimal for music slideshows. A naive approach is to use the top-ranked images in the search results for the image selection. In this case though, highly ranked images are expected to be selected repetitively for the same query, hence, the same image may be used for different songs with similar lyrics. Therefore, this approach is expected to generate slideshows with a lack of diversity, which may cause boredom for system users.

3. SYSTEM CONFIGURATION

The configuration of the proposed system is illustrated in Figure 1. The system selects one image for each lyric line of the input song. The selected image is displayed on the slideshow application (Figure 2) during music play. Images for the slideshow are collected from Flickr, a highly popular photograph sharing site [11], by using the Flickr API. As illustrated in Figure 1, we assume that a database which contains songs with their corresponding lyrics and timing information is prepared beforehand, as in the case of karaoke systems.

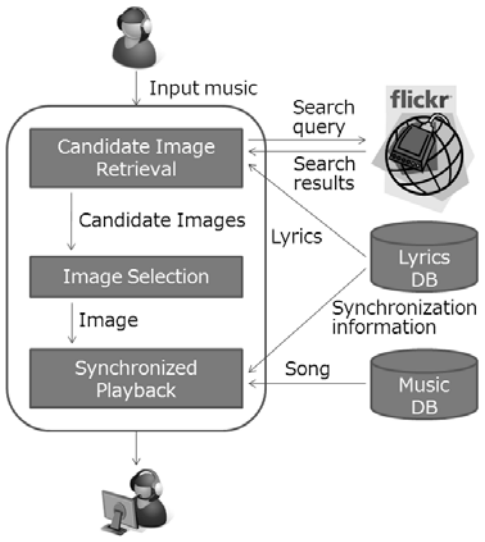


Figure 1. System configuration

The process flow of the system consists of the following three steps.

1. *Candidate Image Retrieval*
   This step extracts a candidate set of images per lyric line, by selecting appropriate query words from each line of the lyrics of the input song.
2. *Image Selection*
   This step selects an image from the previously extracted candidate image set for each line, to compose the slide show.
3. *Synchronized Playback*
   Selected images for each line are displayed with the song, according to the prepared timing information.

Figure 2. A screenshot of the proposed system

The following section explains the slideshow generation method, namely the candidate image retrieval and image selection steps, in detail.

4. SLIDESHOW GENERATION METHOD

4.1 Candidate Image Retrieval

In this step, the system generates a query (set of words) for each lyric line of the input song. The image search result from Flickr, obtained by the generated query is utilized as the candidate image set for the lyric line. The query is generated by analyzing the frequency of query words that are applied to the images in the search result, as social tags. This method is based on the hypothesis that, query words which are frequently used as social tags in Flickr have a significant meaning in the web image database, thus are expected to be effective to retrieve images which are expressive of the song lyrics. This method extracts the optimum combination of query words for each lyric line, based on the following three ideas:

- Words used in a lyric line should be prioritized, since such words accurately represent the content of the line.
- The query should be composed with as many words as possible, since such queries are more specific than single word queries, thus should result in more accurate image retrieval.
- Multiple words within a query tend to co-occur as image social tags.

4.1.1 Process Flow of Social Tag-Based Query Selection

Let $\mathbf{N}_{line}(l_i)$ represent the set of nouns used at the *i-th* line of the lyrics, $\mathbf{N}_{para}(l_i)$ represent the nouns used in the paragraph which contains the *i-th* line, and $\mathbf{N}_{all}(m)$ represent the word set which describes the general impression of song m (hereafter referred to as "general impression words", details explained in Section 4.1.2). Furthermore, when $\mathbf{W}$ expresses the set of words used as the query for

the Flickr API, let DF(**W**) (Document Frequency) represent the number of images in the search results, and UF(**W**) (User Frequency) represent the number of unique users (counted by the user ID information of the Flickr images) in the search results.

The proposed method extracts candidate query words for the *i-th* line in lyrics of music piece *m*, from $\mathbf{N}_{line}(l_i)$ and $\mathbf{N}_{para}(l_i)$. Words which have DF or UF value less than a pre-defined threshold are omitted. The thresholds for DF and UF are empirically set as 40 and 10, respectively.

Next, let $\mathbf{P}(\mathbf{N}_{line}(l_i))$ express the power set of $\mathbf{N}_{line}(l_i)$: $\mathbf{P}(\mathbf{N}_{line}(l_i))=\{\mathbf{W}_{line,1}, \mathbf{W}_{line,2}, \ldots \mathbf{W}_{line,x}\}$, where $\mathbf{W}_{line,x}$ expresses the *x-th* set of words in $\mathbf{P}(\mathbf{N}_{line}(l_i))$. From $\mathbf{P}(\mathbf{N}_{line}(l_i))$, $\mathbf{W}_{\max}$ is selected under the condition that DF($\mathbf{W}_{\max}$) is not zero and that $|\mathbf{W}_{\max}|$ is the highest in $\mathbf{P}(\mathbf{N}_{line}(l_i))$, where $|\mathbf{W}|$ expresses the number of words in **W**. If more than one $\mathbf{W}_{\max}$ can be selected, the set which has the highest UF($\mathbf{W}_{\max}$) is selected. In this way, $\mathbf{W}_{\max}$ is regarded as the set of queries for the *i*-th line, $\mathbf{Q}_{line}(l_i)$.

Then, in order to maximize the number of query words (which is assumed to reduce the number of candidate images, and improve search accuracy), we expand the query by using words in $\mathbf{N}_{para}(l_i)$. Namely, expanded sets of words, which are composed of the power set of $\mathbf{N}_{para}(l_i)$, plus the previously derived $\mathbf{Q}_{line}(l_i)$ are generated as $\mathbf{P'}(\mathbf{N}_{para}(l_i))=\{\mathbf{W}_{para,1}+\mathbf{Q}_{line}(l_i), \mathbf{W}_{para,2}+\mathbf{Q}_{line}(l_i), \ldots \mathbf{W}_{para,y}+\mathbf{Q}_{line}(l_i)\} = \{\mathbf{W'}_{para,1}, \mathbf{W'}_{para,2}, \ldots \mathbf{W'}_{para,y}\}$. Then, in the same way as explained above, $\mathbf{W'}_{\max}$ is selected from $\mathbf{P'}(\mathbf{N}_{para}(l_i))$.

Finally, by sending the all elements of $\mathbf{W'}_{\max}$ under the condition of 'AND' combination to Flickr, the system retrieves the candidate images for each line. If $\mathbf{W'}_{\max}$ has no elements, $\mathbf{N}_{all}(m)$ is used as the query.

4.1.2 Estimation of General Song Impression

As mentioned above, $\mathbf{N}_{all}(m)$ is the general impression word set, i.e., a set of words which expresses the collective impression of song *m*. This word set can be used for lyric lines from which no effective query words could be extracted. Furthermore, the general impression word set is also effective to generate slideshows with a sense of unity, as will be described in the next section.

The general impression of a song is estimated by text-based classification based on its entire lyrics. Namely, song classifiers are preliminarily constructed by SVM [12] for each of the categories showed in Table 1. The categories are divided into three concepts: Season, Weather, and Time. Each concept consists of several categories. The concepts/categories in Table 1 are selected because they all represent important aspects of song lyrics, and are expressed by discriminative words. For the classifier, we used the software *SVM*^{light} [13] with a linear kernel for learning. Here, lyrics have been vectorized by TF*IDF, and the training data for the classifiers learning have been obtained by a manually collected database of Japanese pop songs with human-applied labels. If the classifier determines that a song *m* is positive for its respective category, the name of the category is added to $\mathbf{N}_{all}(m)$. Note that multiple words may be included in $\mathbf{N}_{all}(m)$.

Concepts	Category labels
Season	Spring, Summer, Autumn, Winter
Weather	Sunny, Cloudy, Rain, Snow, Rainbow
Time	Morning, Daytime, Evening, Night

Table 1. Concepts and category labels for describing general impression of music.

4.2 Image Selection

The next step is to select an image to compose the slideshow from the candidate image set for each lyric line. We propose an image selection method based on an impression score, which represents strength of association between the image and the general impression words of the input song. Consideration of the impression score is expected to select images that are more fitting to the overall theme of the input song, thus increases the sense of unity among the images which compose the slideshow.

4.2.1 Relevant Tag Extraction Based on Co-occurrence Probability

Relevant tags for calculating the impression score are extracted based on co-occurrence probability of social tags on Flickr. In this paper, the co-occurrence probability is calculated based on UF instead of DF, since there are many tags with unusually high DF on Flickr, due to users who upload many images with the exact same tag set, while UF is more robust to the effect of such user behavior.

The relevance score between a general impression word $n_{all} \in \mathbf{N}_{all}(m)$, and a given tag *t,* is calculated by the co-occurrence probability of *t* and n_{all}, and also the impression words which belong to the same concept as n_{all}. For example, when the relevance score between "summer" and tag *t* is calculated, the same score for all other general impression words in the "Season" concept, i.e., "spring", "autumn", and "winter", are also calculated. In this way, it is possible to extract tags which have specifically high relevance to n_{all}, and decrease the score of generally popular tags, i.e., words which co-occur frequently with many other words.

The co-occurrence score between general impression word n_{all} and tag *t*, $CoScore(t,n_{all})$, is defined as:

$$CoScore(t, n_{all}) = \frac{UF(t \cap n_{all})}{UF(n_{all})} \quad (1)$$

Then, the relevance score R between n_{all} and t is defined as:

$$R(t, n_{all}) = CoScore(t, n_{all}) - \frac{\sum_{n \in C, n \neq n_{all}} P(t \mid n)}{|C| - 1} \times wgt \quad (2)$$

where $\boldsymbol{C}$ is the set of general impression words which belong to the same concept of n_{all}. For example, when n_{all} = "spring", $\boldsymbol{C}$ = {"spring", "summer", "autumn", "winter"}, since "spring" belongs to the "Season" concept. In the definition of the relevance score in Eq.(2), the first term increases the score of tags which have high co-occurrence probability with n_{all}. Subtraction of the second term decreases the score of tags with high co-occurrence probability of impression words which belong to the same concept as n_{all}. Note that *wgt* is a coefficient to adjust the impact of the second term. This coefficient is set to 3, empirically.

Based on Eq.(2), the relevance score between each general impression word, and all tags which co-occur with the general impression word, are calculated. Tags whose relevance scores are over 0.024, and UF value exceeds 5, are regarded as relevant tags of each impression word.

4.2.2 Definition of Impression Score

For image selection, we calculate the impression score for all images in the candidate image set, based on the tags applied to the image, and the above relevance score. The object of this method is to select images with tags which have high relevance to the general impression words of the input song. As a result of this process, the impression score of images with "noisy" tags, *i.e.*, tags with low relevance to the general impression of the input song, will be degraded.

The impression score of image i is determined by

$$score(i) = \sum_{n_{all} \in N_{all}(m)} \frac{\sum_{t \in T_i \cap T_{related}(n_{all})} R(t, n_{all})}{|T_i| - |T_i \cap T_{related}(n_{all})|} \quad (3)$$

where $\boldsymbol{T}_i$ is the set of tags applied to image i, $\boldsymbol{T}_{related}(n_{all})$ is the relevance tag set of general impression word n_{all}, and $R(t, n_{all})$ is the relevance score between n_{all} and tag t.

This impression score is computed for each candidate image, and the image with the highest score is selected to be displayed with its respective lyric line, during the slideshow.

4.3 Image Transition Timing Adjustment

In the proposed system, the images obtained per lyric line are displayed in synchronization with each line during the song playback. Adequate usage of the line information leads to natural image transition during the slideshow, since lines represent a semantic unit in the lyrics. However, display duration of each image may be too short/long when using the line information naively. For example, in a rap song with many lines, the image display time maybe too short, so that users may not be able to comprehend the images in the slideshow. On the other hand, in a slow ballad song, images may be displayed for a long time, which may cause boredom.

In order to improve the overall quality of the slideshow, we propose an image transition timing adjustment method, which adjusts the display time of images according to the duration of each lyric line. In this process, we first estimate the typical duration time of images in a song. Then, the line of lyrics is "combined" or "divided", based on the difference of the duration of the line and the typical duration time of the input song. In the "combining" process, lines with short duration time are combined with their adjacent lines, and a single image is displayed for the combined set of lines. In the "dividing" process, lines with long duration time are "divided" into plural sub-lines, and an image is to be displayed along with each sub-line.

The process flow for image transition timing adjustment consists of the following steps.

1. Typical duration time of song m is calculated from the lyrics data. Namely, the mode value of the line duration time is used as the typical image duration time I_m.
2. Lines which have less than 4 [sec] duration time are "combined" with the next line. If there is no next line, it is "combined" with the previous line. However, lines are "combined" only if they belong to the same paragraph. An image is retrieved for the newly combined line.
3. A line which has more than 12 [sec] duration time is "divided" equally. The number of divisions is controlled so that approximate duration time of the new "divided" line is equivalent to I_m. When the line is "divided" into n lines, n images are displayed from the candidate image set, which is retrieved based on the lyrics of the original line.
4. Interlude sections (which generally have no lyrics) are divided by the same process as step 3. The general impression words are used as query for image retrieval.

5. EXPERIMENTS

5.1 Outline

In order to evaluate the quality of the proposed method, we have conducted a subjective evaluation experiment. This experiment compares the proposed method with other conventional methods, by asking 42 subjects to rate the slideshows generated by all methods. The subjects are asked to view the music slideshows of the same song, which are generated by the proposed and comparative methods (details of the methods are explained in Section

5.2). Then, each subject is asked to apply a five-ranked rating for each slideshow, based on the following evaluation measures:

a) Accordance between lyrics and images [content]
b) Appropriateness of image display time [duration]
c) Unity of all images in slideshow [unity]
d) Overall quality [quality]

In this experiment, we use 10 Japanese pop songs and 28 ~ 29 subjects have provided evaluation results for each song. In order to evaluate the method described in Section 4.3, we have selected songs so that half of these songs include "combined" lyric lines (hereafter referred to as the "combined set") in the process of adjustment of image transition timing explained in Section 4.3, and the other half include "divided" lines (hereafter referred to as the "divided set").

5.2 Evaluated Methods

The next three methods were evaluated and compared.

A) *MusicStory* [9]

The first comparative method generates slideshows based on the method proposed for MusicStory [9]. Namely, all nouns are extracted from the entire lyrics of the input song, and are sent to the Flickr API under the 'OR' combination. The images in the search result are displayed according to the transition timing determined by the BPM (beats per minute) of the input song

B) *TF*IDF based method*

The second comparative method extracts query words from the lyrics based on TF*IDF. The process flow to obtain the image for the *i-th* line in the lyrics is described as follows. First, the nouns extracted from the *i-th* line in the lyrics, are sent to Flickr as query under the condition of 'AND' combination. If the result has no images, the noun with the smallest TF*IDF is removed, and the rest of the nouns are sent to Flickr again. This process is repeated until a set of images are obtained. If the system is unable to retrieve images by any of the nouns in the line, the images from the previous line are re-used. Finally, the highest-ranked image in the search result (according to the Flickr "interestingness" ranking) is selected. Images are obtained for each line and switched in synchronization with line appearance within input song. In this paper, the DF element of TF*IDF is calculated based on our database, which contains 3062 Japanese pop songs.

C) *Proposed method*

The third method is our proposal. Queries are generated from the lyrics by the social tag-based method, images are selected from the image search results based on the impression score, and the image transition timing is adjusted by the method described in Section 4.3.

5.3 Experimental Results

Figure 3 shows the average rating of all subjects, for each evaluation measure and method. The results in this Figure show that the proposed method has received the highest ratings, compared to the other methods for all evaluation measures. Most significantly, the proposed method has received the best rating for the overall quality, a difference which is statistically significant to the others based on t-test ($p<0.001$). These results prove that the proposed method is capable of generating high-quality slideshows.

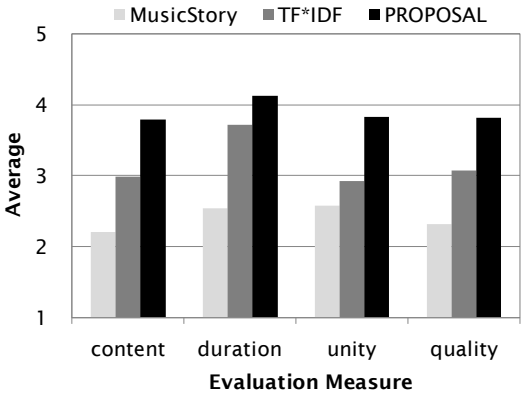


Figure 3. Average ratings of each evaluation measure.

Lyrics line	*"If this separation means departure, I will give my all smiles to you."*	
Query	TF*IDF based	"departure" (line word)
	Proposal	"smile" (line word)
Lyrics line	*"Will the memory of our encounter and the town we'd walked in be kept in our heart?"*	
Query	TF*IDF based	"heart" (line word)
	Proposal	"town" (line word)
Lyrics line	*"I wish I could stay with you for even a moment."*	
Query	TF*IDF based	"moment" (line word)
	Proposal	"car" and "night scene" (paragraph words)
Lyrics line	*"But the shining days will never return to me today or tomorrow."*	
Query	TF*IDF based	"today" (line word)
	Proposal	"evening" (general impression word)

Table 2. Examples of image search queries generated by TF*IDF based and proposed methods.

In order to analyze the query selection process of the proposed method, we compare the queries generated by the proposal to those of the TF*IDF based method. Examples are written in Table 2. This table shows examples of lyrics lines (English translations by the authors from the original Japanese lyrics) and the queries generated from the lines by the two methods. In the first two examples in this table, it is clear that the proposal has success-

fully selected words which represent visual concepts. Contrarily, the TF*IDF method has selected words which are important, but also are difficult to be represented in a visual manner. This is due to the characteristic of the proposed method, which considers the UF values of the words in Flickr. Furthermore, when there are no "visual" words in the lyrics, the proposal can appropriately generate queries, either from the lyric paragraph, or general impression words, as shown in the last two examples. These examples indicate that the proposed method is effective to generate good queries from any song lyric.

Moreover, even when the queries generated by the both methods are the same, the proposal is capable of selecting more suitable images for the song. For example, when both methods retrieve images by the query "town" for a winter song, the proposal appropriately selects an image of a town with falling snow, while the TF*IDF based method selects a general image of a town. Examples like this indicate that the proposed image selection method based on impression score can generate suitable slideshows which represent the overall theme of the song.

Additionally, in the "duration" measure, the proposal has achieved ratings superior to the TF*IDF based method for 9 songs, indicating that the proposed adjustment method has succeeded in improving slideshow quality. The difference of the average ratings between the proposal and the TF*IDF based method for "combined sets" is 0.21, while the difference for "divided sets" is 0.61. This result implies that the proposed method is more effective to improve slideshows for songs with lyrics that are slowly sung, as in slow ballads.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a system to generate slideshows for any given song, by using words in their lyrics to retrieve web images. We have proposed a query generation method for image search and an image selection method to compose slideshows from the image search results. Moreover, we proposed a method to adjust image transition timing based on the lines of lyrics. Results of subjective evaluations have shown that our system can generate highly satisfactory music slideshows.

In the future, we plan to expand our system to utilize not only the lyrics, but also the acoustic features of the input song. For example, displaying slideshows with various effects, such as zooming and panning, in accordance with the excitement of the song; as well as the use of beat information for image transition all are expected to improve the impression of the generated slideshows.

7. REFERENCES

[1] S. Iwamiya: "The interaction between auditory and visual processing when listening to music via audio-visual media," *The Journal of the Acoustical Society of Japan*, Vol.48, No.3, pp.146-153, 1992. [in Japanese]

[2] R. Mayer, R. Neumayer, and A. Rauber: "Rhyme and Style Features for Musical Genre Classification by Song Lyrics," *Proceedings of ISMIR 2008*, pp. 337-342, 2008.

[3] F. Kleedorfer, P. Knees, and T. Pohle: "Oh Oh Oh Whoah! Towards Automatic Topic Detection in Song Lyrics," *Proceedings of ISMIR 2008*, pp. 287-292, 2008.

[4] Y. Hu, X. Chen, and D. Yang: "Lyric-based Song Emotion Detection with Affective Lexicon and Fuzzy Clustering Method," *Proceedings of ISMIR 2009*, pp. 123-128, 2009.

[5] X. Hu, J. S. Downie, and A. F. Ehmann: "Lyric Text Mining in Music Mood Classification," *Proceedings of ISMIR 2009*, pp. 411-416, 2009.

[6] X. -S. Hua, L. Lu, and H. -J. Zhang: "P-Karaoke: Personalized Karaoke System," *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pp.172-173, 2004.

[7] T. Terada, M. Tsukamoto, and S. Nishino: "A System for Presenting Background Scenes of Karaoke Using an Active Database System," *Proceedings of the ISCA 18th International Conference on Computers and Their Applications*, pp. 160-165, 2003.

[8] S. Xu, T. Jin, and F. C. M. Lau: "Automatic Generation of Music Slide Show using Personal Photos," *Proceedings of 10th IEEE International Symposium on Multimedia*, pp. 214-219, 2008.

[9] D. A. Shamma, B. Pardo, and K. J. Hammond: "MusicStory: a Personalized Music Video Creator," *Proceedings of the 13th Annual ACM International Conference on Multimedia*, pp.563-566, 2005.

[10] R. Cai, L. Zhang, F. Jing, W. Lai, and W. -Y. Ma,: "Automated Music Video Generation Using Web Image Resource," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2007, Vol.2, pp. 737-740, 2007.

[11] Flickr: http://www.flickr.com/

[12] C. Cortes and V. Vapnik: "Support Vector Networks," *Machine Learning*, Vol. 20, pp.273-297, 1995.

[13] SVM-Light Support Vector Machine: http://svmlight.joachims.org/

AUTOMATIC CHARACTERIZATION OF DIGITAL MUSIC FOR RHYTHMIC AUDITORY STIMULATION

Eric Humphrey
Music Engineering Technology Group
University of Miami
Coral Gables, FL 33124
humphrey.eric@gmail.com

ABSTRACT

A computational rhythm analysis system is proposed to characterize the suitability of musical recordings for rhythmic auditory stimulation, a neurologic music therapy technique that uses rhythm to entrain periodic physical motion. Current applications of RAS are limited by the general inability to take advantage of the enormous amount of digital music that exists today. The system aims to identify motor-rhythmic music for the entrainment of neuromuscular activity for rehabilitation and exercise, motivating the concept of musical "use-genres." This work builds upon prior research in meter and tempo analysis to establish a representation of rhythm chroma and alternatively describe beat spectra.

1. INTRODUCTION

Digital multimedia is now an integral, and somewhat inescapable, aspect of modern life. Personal handheld devices are designed to streamline the acquisition, management and playback of large volumes of content as cutting-edge computing devices approach ubiquity. This trend, in tandem with the commercial success of devices like the iPod and iPhone, has encouraged an environment where both content providers and end-consumers have access to enormous digital music collections. As a result, individuals are consuming and purveying more music than ever before and this realization introduces the classic logistical issue of content navigation; when a library becomes sufficiently large, more complex paradigms must be developed to facilitate the searching, indexing, and retrieval of its items.

Conventional music library systems employ metadata to organize the content maintained within them, but are typically limited to circumstantial information regarding each music track – such as the artist's name or the year it was produced – in addition to the somewhat amorphous attribute of genre. Understandably, stronger information concerning the specific nature of a track allows for more insightful and context-driven organizations or queries of a library.

The need for content-specific metadata introduces the challenge that someone, or something, must extract the relevant information necessary. One approach, like the one taken by the Music Genome Project, is to manually annotate a predetermined set of attributes by a diligent group of human listeners, a scheme with clear benefits and drawbacks. While this method is substantiated by the observation that no computational system has yet matched itsreliability, it simply takes a human listener far too much time to parse music. As an example, it would require about 68 years to listen to every track currently available in the iTunes Store, [1] which now contains some 12 million tracks.

Needless to say, the development of computational algorithms to extract meaningful information from digital music provides the ability to process content as fast as an implementing machine can manage. Many efforts over the last twenty years proceed to these ends in varying levels of scope and success. As mentioned however, no single solution has been able to rival the performance and versatility of even moderately skilled human listeners. It has been proposed previously that, in this period of continued research toward improved machine-listening technologies, algorithms are likely to perform best when developed for a specific application.

It is in this spirit that a computational system is proposed to characterize the suitability of musical recordings for rhythmic auditory stimulation, a neurologic music therapy technique that uses rhythm to entrain periodic physical motion. The remainder of the paper is structured as follows: Section II addresses the background of motor-rhythmic music as a use-genre and the physiological motivations; Section III briefly reviews relevant computational models of human rhythm perception and details the proposed system; Section IV explores the evaluation and visualization of the algorithm results; and Section V discusses the system behavior, observations, and directions of future work.

[1] With an average track duration of 3 minutes.

2. BACKGROUND

Music and motion share a long and intertwined relationship throughout human history. Dance comprised an integral role in many ancient civilizations for spiritual and social purposes and work song served to synchronize the physical labor of crews, as was common on sea-faring vessels. In modern times, physical exercise is often tightly coupled with music, ranging from joggers with personal media players to fitness classes.

Many individuals empirically find that music facilitates exercise, and recent advances in music therapy and neuroscience give this notion credence. Through an increased understanding of the underlying mechanisms involved in a human's physiological response to music, current knowledge supports the position that rhythm serves as a powerful external timing mechanism capable of entraining gait parameters and neuromuscular activity [1]. Building upon this principle, rhythmic auditory stimulation (RAS) is "a neurological technique using the physiological effects of auditory rhythm on the motor system to improve the control of movement in rehabilitation and therapy" [2].

The impact of rhythmic auditory stimuli on movement can be summarized as three primary components. Sensory motor control provides priming and timing cues to individual in guiding a motor response. Motor programs are thought to be developed in the brain to control complex motor movement, where rhythmic stimuli encourage the creation of more efficient and fluid programs for cyclical movement. Also, RAS supports goal-directed movement where motion is cued by anticipation, a key musical element, rather than by explicit events like heel strikes.

Appropriate music to achieve RAS, best described as *motor-rhythmic*, must exhibit certain criteria: a strong beat-percept, regular meter, little to no tempo deviation, and maintain a tempo that encourages the desired entrainment frequency, referred to in the literature as an individual's resonant frequency or limit cycle. The ability to succinctly describe a class of musical content for a specific application motivates its distinction as a use-genre.

A fundamental problem faced in RAS-based research and applications is the inability to harness the abundance of available digital music as external entrainment stimuli, as no solution exists to characterize music for this purpose. It is for this reason that nearly all uses of RAS are confined to closely-monitored clinical settings that heavily rely on human supervision to provide, and sometimes compose, appropriate motor-rhythmic music. An automated system would not only facilitate the practice of RAS as a clinical rehabilitation technique, but also allow the integration of RAS methodologies on a significantly broader scale, such as exercise classes or personal fitness technologies.

Some previous systems attempt to link the rhythm, and more specifically the tempo, of music and physical motion in the form of running [3]. Each effort, however, incorporates the assumption that all content is accurately and sufficiently described by a single tempo value. Quickly considering the great diversity of musical content available, it is intuitive to conclude that this is inadequate. With these goals in mind, we seek to develop a system capable of quantifying the motor-rhythmic attributes of digital music content for use in applications of RAS.

3. PROPOSED SYSTEM

Computational rhythm analysis algorithms for digital music recordings have been extensively researched over the last twenty years. Early systems were developed to perform tempo extraction of individual tracks and excerpts to ascertain a single tempo value, and beat tracking to annotate the location of musical pulses in a recording, both achieving notable success. More recent efforts aim to improve upon these results by employing alternate mechanisms to fulfill various system tasks or seek to determine further information, such as meter [4] and beat spectrum [5]. A more thorough review of recent leading systems is provided in [6].

Being that human rhythm analysis remains the best performing system, explicit modeling of the human auditory system would appear to be a viable approach toward the development of a machine-listening algorithm for rhythmic analysis. By reducing the task of rhythm perception to the functional components of the overall biological process, each stage can be approximated computationally. At the most rudimentary level, human rhythm perception is achieved in a two-stage process of event perception and periodicity estimation.

The idea of determining meaningful events in music perception is admittedly a loaded topic. However, a semantic debate can be mostly avoided in considering that there are arguably three orthogonal dimensions in basic music perception: rhythmic, tonal and timbral. In the context of characterizing the suitability of music for RAS, the focus of meaningful events can – and should – be constrained primarily to rhythmic, or energy-based, events. Neglecting the other two dimensions serves to emphasize the importance of rhythmic content.

Periodicity estimation can be computationally achieved in a variety of different manners depending on performance concerns, such as causality and complexity. One common school of thought regarding human beat induction claims that the phenomena of felt-beat it is achieved through the resonating, or entrainment, of oscillator banks in the brain as an interval-period based process [2]. This is a particularly attractive option given the correlation between the oscillations of the human body as a dynamic mechanical system during movement and those of a mathematical model.

Coincidentally, these are essentially the main system components presented by Scheirer in [7] and Klapuri et al in [4]. Building upon the work outlined therein, the proposed system proceeds in the following manner: an input signal is first decomposed into twenty-two subband components via a maximally-decimated filterbank closely approximating the critical bands of the cochlea and rhythmic events are derived for each. These onset events are reduced to a single stream of pulses and periodicity estimation is performed using a bank of modified comb-filter oscillators. The resulting beat spectra is transformed into

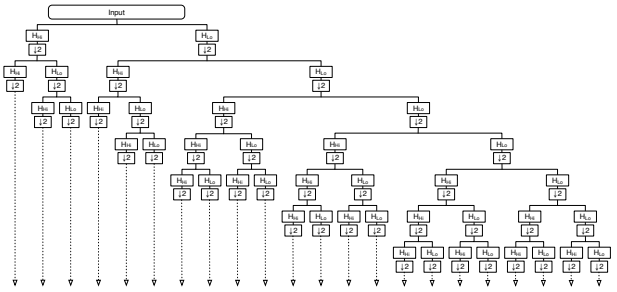


Figure 1. A perceptually-motivated dyadic filterbank for the decomposition of an input audio waveform.

Band	Range (Hz)	Band	Range (Hz)
1	0 – 125	12	1750 – 2000
2	125 – 250	13	2000 – 2500
3	250 – 375	14	2500 – 3000
4	375 – 500	15	3000 – 3500
5	500 – 625	16	3500 – 4000
6	625 – 750	17	4000 – 5000
7	750 – 875	18	5000 – 6000
8	875 – 1000	19	6000 – 8000
9	1000 – 1250	20	8000 – 10000
10	1250 – 1500	21	10000 – 12000
11	1500 – 1750	22	12000 – 16000

Table 1. Frequency ranges for the resulting subband components.

rhythm chroma over time, from which global features are calculated to compactly describe the entirety of a music track.

3.1 Cochlear Modeling

At this point in time, it is commonly held that the human auditory system is reasonably understood so far as the point where electrical signals are encoded and transmitted to the brain via the auditory nerve. Most stages prior to neural processing though, such as diffraction of the pinnae or dynamic compression from the bones of the inner ear, are not overly integral to the perception of rhythm. However, the cochlea does perform a coarse frequency decomposition as transduction occurs across the critical bands of the organ. Scheirer observed that the perception of rhythm is maintained when amplitude modulating white noise with the envelopes of as few as four subbands of an audio waveform [7]. Therefore, it is proposed that monitoring the fluctuation of energy in each critical band serves as a reasonable approximation of preconscious observation of meaningful rhythmic events.

Motivated in part by the system developed by Tzanetakis et al [8], a multi-resolution time-domain filterbank is used to decompose an input waveform into twenty-two subbands. Whereas wavelet processing implements complimentary half-band filters and a true pyramidal structure, the filterbank divides frequency content similarly to the cochlea, the ranges of which are listed in Table 1 and diagramed in Figure 1.

It is important to note that, given the cascaded nature of the structure, non-linear phase distortion introduced by IIR filters is unacceptable and errors will propagate differently in each band. This is particularly troublesome in the context of a system developed to analyze the temporal relationship between events. Therefore, half-band FIR filters of Daubechies' coefficients are chosen, and appropriate all-pass filters are designed to flatten the group delay at each successive level to ensure alignment of the resulting subband components. The accumulative delay and complexity of the filterbank decomposition is mainly dependent on the length of the Daubechies' filter shape selected (N = 40 in our experiments), though the impact of using different filter lengths on performance has yet to be explored.

3.2 Rhythm Event Detection

Following decomposition, each subband signal is processed identically to identify rhythm event candidates. Consistent with [7] and [4], subband envelopes are calculated by half-wave rectifying and low-pass filtering each subband waveform with a half-Hanning window, defined by Equations 1 and 2.

$$X_{HWR_k}[n] = \max(X_k[n], 0) \tag{1}$$

$$E_k[n] = \sum_{i=0}^{N_k-1} X_{HWR_k}[n] * W_k[i-n] \tag{2}$$

Subband envelopes are then uniformly down-sampled to 250 Hz, influenced by the temporal resolution of the human auditory system, and compression is applied to the resulting signals according to Equation 3. Event candidates are calculated by filtering the subband envelopes with the Canny operator defined in Equation 4, commonly used in digital image processing for edge detection and first applied to audio processing in [9]. The frequency response of the Canny operator is more desirable than that of a first-order differentiator, being band-limited in nature and serving to attenuate high-frequency content.

$$E_{C_k}[n] = \frac{\log_{10}(1+\mu * E_k[n])}{\log_{10}(1+\mu)} \tag{3}$$

$$C[n] = \frac{-n}{\sigma^2}\exp(-n^{\frac{2}{2\sigma^2}}), \quad \text{where } n = [-L, L] \tag{4}$$

At this stage, event candidates effectively represent the activation potential of their respective critical bands in the cochlea. Though there are multiple hair cell transduction theories concerning the significance of place and rate on pitch perception, the fact remains that temporal masking is caused by the necessary restoration time inherent to the chemical reaction associated with neural encoding. Known as the precedence effect, sounds occurring within a 50 millisecond window–about 10 milliseconds before and 40 milliseconds behind–are perceived as a single event. This phenomena is modeled by a sliding window to eliminate imperceptible or unlikely event candidates.

Rhythm event detection concludes with the summation of subband events to a single train of pulses and a zero-order hold to reduce the effective frequency of the pulses.

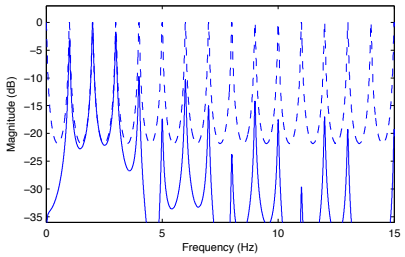


Figure 2. Magnitude response of a typical comb-filter (dashed line) and cascaded with a Canny filter (solid line).

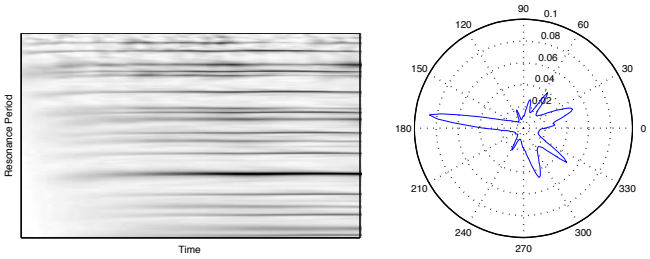


Figure 3. Example of a tempogram and chroma for *bonus5.wav*, from the MIREX practice data set.

A single-sample pulse is the half-wave rectified counterpart to a single period of the highest frequency that can be represented by the current sampling rate. Rhythmic frequency content, such as the tactus or felt-beat, typically exists on the range of .25–4 Hz (or 30–240 BPM), with tatum and metrical levels falling just above and below that range, respectively. Therefore, a zero-order hold of 50 ms is applied to band-limit the signal, constraining frequency content to 20Hz while maintaining the temporal accuracy necessary.

3.3 Periodicity Estimation

In continuing with modeling preconscious rhythm audition, periodicity estimation is performed using a set of tuned comb-filters spanning the frequency range of interest. This method was pioneered as a computational model of rhythm induction by Scheirer in [7], and has since been incorporated in a variety of derivative works due to reliability and modest computational complexity. Importantly, modifications are introduced here to improve performance and tailor the model to better suit the target application.

Unlike previous systems that aim to set a constant resonance half-life across each oscillator, we propose that perceived resonance of a pulse train is dependent not on time but the number of pulses observed. It seems intuitive that a 40 BPM click track at 40BPM should take longer to perceive at the same strength as one at 180 BPM. Though a more perceptually-motivated method may better capture this nuance, the value of α is set at 0.825 to require a period of regularity before resonating, while maintaining the capacity to track modulated tempi.

Beat spectra is computed over time for each delay lag T, as defined by the comb-filter difference equation in Equation 5, varied linearly from 50–500 samples, inversely spanning the range of 30–300 BPM. Each comb-filter is also cascaded with a band-pass filter – the Canny operator – to augment the frequency response of the periodicity estimation stage. As shown in Figure 2, this attenuates the steady-state behavior of the comb-filter effectively lowering the noise floor, while additionally suppressing resonance of frequency content in the range of pitch perception over 20Hz. The Canny filter is also corrected by a scalar multiplier to achieve a passband gain of 0 dB.

$$y_k[n] = (1-\alpha) * x[n] + \alpha * y_k[n - T_k] \qquad (5)$$

Instantaneous tempo is calculated by low-pass filtering the energies of each oscillator over time. Scheirer previously described this process of determining the energy in the delay line over the length of the resonance period, and is analogous to computing an unweighted-average. A Hanning window of length W_k, set corresponding to the delay lag of its respective comb-filter channel and given in Equation 6, serves as an estimation of "resonance memory." This time-frequency representation is referred to as a *tempogram* and estimates perceived tempo strength over time, an example of which is shown in Figure 3.

$$R_k[n] = \frac{1}{W_k} \sum_{i=0}^{T_k-1} w_k[i] * (y_k[n-i])^2 \qquad (6)$$

3.4 Chroma Transformation

As observed by Kurth et al [5], the duality of pitch and rhythm allows the representation of beat spectra in terms of chroma. In the same way that all pitches can be described as having a height and class, various metrical levels exhibit a similar relationship. Octave errors, a typical issue faced in tempo extraction, are mitigated by eliminating the subjective aspect of rhythm and reducing the task to a purely objective one. Fundamental tempo class is especially important to RAS-applications, and is the ultimate focus of the system.

Rhythm chroma is computed by first transforming beat spectra to a function of frequency, rather than period, scaled by the base-2 logarithm and referenced to 30 BPM. Three tempo octaves (30–60, 60–120, and 120–240 BPM) are collapsed by summing beat spectra with identical chroma, as detailed in Equation 7. Understanding this representation is facilitated by plotting amplitude as a function of $\log_2$ tempo class in the polar coordinate system, shown in Figure 3, such that the harmonic structure of a given input becomes readily apparent.

For clarity, rhythm chroma consists of a radial amplitude and an angular frequency, referred to as a class and measured in units of degrees or radians. The transformation from tempo, in BPM, to class, in normalized radians, is defined by Equation 8. This is a many-to-one mapping, and is not singularly invertible. Visualizing rhythm chroma in this alternative manner allows for deeper insight into the nature of musical content and the extraction of novel features, and will be discussed in greater detail shortly.

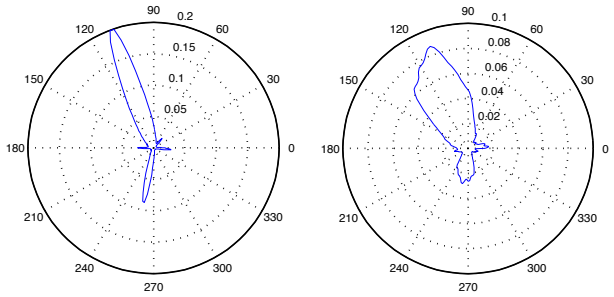


Figure 4. Chroma diagrams for a 148 BPM click track, before and after tempo automation. Note the difference in scale and amplitude of the fundamental.

$$\Psi_n[\omega] = \frac{1}{L}\sum_{i=0}^{L-1} R_n[\omega + 2\pi * k] \tag{7}$$

$$\omega_{class} = \log_2 \frac{BPM}{BPM_{reference}} \tag{8}$$

3.5 Feature Vector Representation

A single rhythm chroma is obtained for a track by summing over time and normalizing by the length. Several key features of interest are emphasized by producing a global chroma, though this set presented is not intended to be exhaustive by any means. Beat strength is effectively described by the amplitude of the largest lobe, and fundamental tempo class is given by the angle of this peak. Other lobes are actually subharmonics of the fundamental, and provide further information about the rhythmic composition. It is important to note that the radius and angle of all harmonics, the fundamental as well as the partials, are significant, as they describe what is best referred to as rhythmic timbre. Amplitude ratios between the fundamental and the various partials serve as a metric of beat salience– the clarity of the prevailing rhythmic percept– as well as a confidence interval regarding system reliability.

An added benefit of averaging the rhythm chroma is found in the fact that frequency modulations of the fundamental chroma manifest as a widening of the primary lobe. Due to the behavior of comb-filter resonance, tempo deviations will inherently attenuate the amplitude of the fundamental. From these observations, optimal music for RAS will exhibit a large, narrow and clearly-defined fundamental with smaller, though still clearly-defined, partials.

4. EVALUATION

Since there are, to our knowledge, no previous attempts to mathematically quantify the motor-rhythmic attributes of musical content, system behavior is explored for a small set of content defined as ground-truths. Initially, we examine the responses for a constant-tempo click track and a frequency-modulated version of itself. For familiarity, select content from the MIREX tempo tracking practice data is then processed by the proposed system.

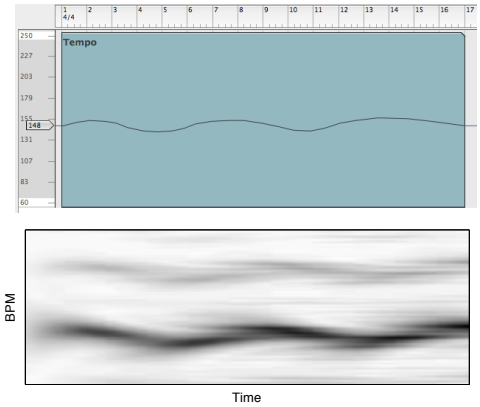


Figure 5. Image of the tempo automation used to modulate the tempo of the click track, and the corresponding chromagram after analysis.

The prominent role of metronomes and click tracks in past RAS research is indicative of the fact that they are the most basic form of motor-rhythmic stimuli. A thirty-second audio click track was created using a sampled clave in Propellorhead's Reason software and the tempo was set at 148 BPM. The software also offers the capability of tempo automation and allowed for the creation of a second, frequency-modulated click track to model an expressive performance. As shown in Figure 4, the constant-tempo click track produces a chroma with clearly defined fundamental and several smaller subharmonics, while the chroma lobes of the frequency-modulated click track are smeared and roughly half the amplitude. While salient, given the ratio of the significant peaks, the widening of the lobes is a direct result of the tempo variance in over time. Importantly, a chromagram is shown above the tempo automation curve used to modulate the tempo of the click track in Figure 5. Though the chromagram incurs some delay in tracking the modulation of the click track, the system is able to follow the tempo throughout.

Though informative and worthwhile examples to consider, click tracks are not the primary focus of this system and it is necessary to also examine the chroma of real music data. For ease of access and familiarity within the research community, musical content is selected from practice data available on the MIREX website [10]. The set of excerpts contains a variety of different styles, but there are two tracks in particular – *train8.wav* and *train12.wav* – that serve as prime examples of what is and what is not motor-rhythmic music.

Figure 6 shows the chroma for the two separate tracks. It is evident from the diagram that *train8.wav*, an electronic piece by Aphex Twin, is significantly more motor-rhythmic than *train12.wav*, an orchestral performance of a composition by J. S. Bach, with a beat strength nearly 40 times greater in amplitude. Despite the lack of harmonic definition in the chroma of the orchestral track, this system is capable of identifying the correct fundamental class for both excerpts according to metadata provided.

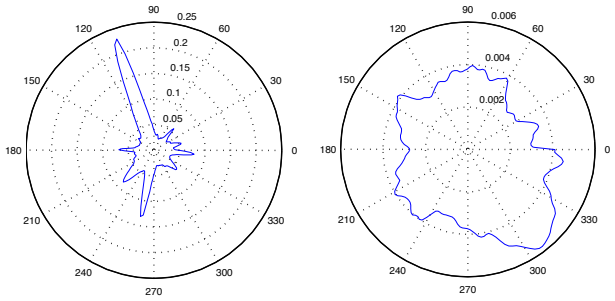


Figure 6. Instances of good (left) and poor (right) motor-rhythmic music.

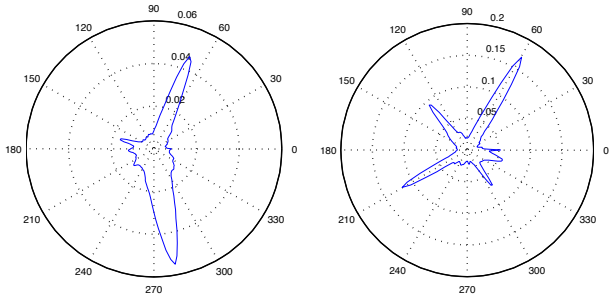


Figure 7. Chroma representations for non-binary meter tracks performed in $6/8$ (left) and $7/8$ (right).

5. DISCUSSION

Content analysis algorithms for the computation of feature-specific metadata will no doubt play a vital role in the future as digital music libraries continue to increase in volume seemingly without bound. The system presented here details one such application of a relatively straightforward use-genre that extends previous machine listening efforts. The task of characterizing music for RAS benefits greatly from the circumstances of the context in which it is used, wherein the most relevant attributes of motor-rhythmic music are objectively quantifiable.

Furthermore, representing the global rhythm in terms of chroma allows for a compact description of the temporal structure of music. Succinctly stated, the degree of tempo variation inherent in a track influences both the width and height of the chroma partials. Any music track can be reasonably approximated as a set of rhythmic partials with corresponding amplitudes, angles, and widths.

5.1 Future Work

One of the more interesting observations to result from this work is the realization that the harmonic structure of rhythm chroma may provide information about the meter and other time segmentations. Figure 7 shows the global chroma of two tracks of note from the MIREX practice data set: *train5.wav* and *bonus3.wav*. These tracks are of particular interest as they are not binary meter; the former is $6/8$ and the latter is $7/8$. The chroma of *train5.wav* is really only comprised of a fundamental and a closely-competing subharmonic at a difference angle of about 150°. Alternatively, *bonus3.wav* is comprised of a variety of subharmonics, but the partial located 70° from the fundamental is not even remotely present in any other chroma representations observed. More work is necessary to determine the true depth of the information contained within these data.

6. REFERENCES

[1] M. Thaut, G. McIntosh, S. Prassas and R. Rice, "Effect of Rhythmic Auditory Cuing on Temporal Stride Parameters and EMG Patterns in Normal Gait." *Journal of Neurologic Rehabilitation*, Vol. 4, No. 6, pp. 185–190, 1992.

[2] M. Thaut, *Rhythm, Music, and the Brain: Scientific Foundations and Clinical Applications.* Routledge, 2008.

[3] N. Masahiro, H. Takaesu, H. Demachi, M. Oono and H. Saito, "Development of an Automatic Music Selection System Based on Runner's Step Frequency." *Proc of the 9th Int Conf on MIR*, pp. 193–198, 2008.

[4] A. Klapuri, A. Eronen and J. Astola, "Analysis of the Meter of Acoustic Musical Signals." *IEEE-TSAP*, 2006.

[5] F. Kurth, T. Gehrmann and M. Muller, "The Cyclic Beat Spectrum: Tempo-related Audio Features for Time-scale Invariant Audio Identification." *Proc of the 7th Int Conf on MIR*, pp. 35–40, 2006.

[6] M. McKinney, D. Moleants, M. Davies and A. Klapuri, "Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms." *New Music Research*, 2007.

[7] E. Scheirer. "Tempo and Beat Analysis of Acoustic Musical Signals." *Journal Acoustical Society of America*, 1998.

[8] G. Tzanetakis and P. Cook. "Musical Genre Classification of Audio Signals." *IEEE-TSAP*, Vol. 10. No. 5. pp. 293–302, 2002.

[9] L. Lu, D. Liu and H. J. Zhang. "Automatic Mood Detection and Tracking of Music Audio Signals", *IEEE-TSAP*, Vol. 14, No. 1, pp. 5–18, 2006.

[10] MIREX Website, [Online]. http://www.music-ir.org/mirex/2006/index.php/Audio_Tempo_Extraction.

AUTOMATIC MOOD CLASSIFICATION USING TF*IDF BASED ON LYRICS

Menno van Zaanen
Tilburg Center for Cognition and Communication
Tilburg University
Tilburg, The Netherlands
mvzaanen@uvt.nl

Pieter Kanters
Tilburg Center for Cognition and Communication
Tilburg University
Tilburg, The Netherlands
pieterkanters@gmail.com

ABSTRACT

This paper presents the outcomes of research into using lingual parts of music in an automatic mood classification system. Using a collection of lyrics and corresponding user-tagged moods, we build classifiers that classify lyrics of songs into moods. By comparing the performance of different mood frameworks (or dimensions), we examine to what extent the linguistic part of music reveals adequate information for assigning a mood category and which aspects of mood can be classified best.

Our results show that word oriented metrics provide a valuable source of information for automatic mood classification of music, based on lyrics only. Metrics such as term frequencies and tf*idf values are used to measure relevance of words to the different mood classes. These metrics are incorporated in a machine learning classifier setup. Different partitions of the mood plane are investigated and we show that there is no large difference in mood prediction based on the mood division. Predictions on the valence, tension and combinations of aspects lead to similar performance.

1. INTRODUCTION

With the current boost in music sharing (alongside sharing files in other formats) [6], Celma and Lamere [4] state that we see a transformation from albums to individual MP3s and mixes. This also changes the way people interact with their music collection and the demands they place on the software that allows this interaction.

Due to the increasing size of online or digital music collections, users would like to be able to access their collections through more and more advanced means [13]. For instance, users would like to be able to search for songs based on various properties, such as year, genre, play count, online recommendation (Web 2.0) or even based on a set of songs used as seed to find similar ones. One particular property that people use when creating playlists is mood [17]. Currently, this is often done manually by selecting songs that belong to a particular mood and naming the playlist according to the mood, such as "relaxing". Here we investigate the possibility of assigning such information automatically, without user interaction.

In recent years, automatic playlist generation has been introduced to cope with the problem of the tedious and time consuming manual playlist selection. Furthermore, browsing the entire music library manually to select songs for the playlist is felt to be difficult by most music listeners. This becomes especially difficult if music collections become prohibitively large, as the user will not know or remember all songs in it.

In fact, it turns out that people have large amounts of music in their music collection that they never even listen to. This phenomenon is called *The Long Tail* [2] as many songs fall in the region of songs that are hardly ever listened to, which is visualized as a long tail on the right size in a histogram. Automatically generating playlists based on certain properties, such as mood, can expose songs from the long tail and allow for the user to explore "lost music" in their music collections.

Here, we will focus on the automatic classification of music into moods, which is sometimes called "music emotion classification" [18]. Given a music collection containing songs that do not yet have these moods assigned to them (or when adding a new, untagged song to the collection), the process automatically adds the mood tags to the song, allowing selection of songs based on moods.

We think the melodic part of songs contains important information that can help the mood classification [10]. However, we will currently focus on the linguistic aspects of songs only. The idea is that lyrics contain lexical items that emphasize a certain mood and as such can be used to identify the underlying mood. Even though in spoken language, just like in music, other aspects such as loudness and pitch may also be important triggers to identify the song's emotion, we assume here that the actual words can have an emotional load without being spoken or sung. For instance, words such as "happy" or "dead" do not have to be pronounced to have an emotional load. This corresponds to Beukeboom and Semin's idea [3] that mood affects word choice and that lexical items can express moods.

There has been previous work on the influence of lyrics on the mood of a song, such as approaches that concen-

trate on more generic properties present in the lyrics [7], combining music and lyrics using LSA and vector space models [12] or using only the musical aspects [9]. Our approach is quite similar to the work presented in [8], where multi-label classification is performed, resulting in a different classification task. In this paper we concentrate on the influence of the different divisions of classes on the final results.

2. APPROACH

In this article, we describe the development of a machine learning approach to classifying songs, based on lyrics only, into classes that describe the mood of the song [11]. For this, we need several components. Firstly, we need to identify which classes (moods) we are going to classify into. Secondly, we need to select a collection of features that allow us to describe properties of the lyrics. Using these components, we can take a collection of lyrics (describing songs), extract the features and classify the lyrics into their mood class. This mood information can be used to automatically create mood-oriented playlists.

2.1 Class divisions

When building a system that can classify songs into moods, we require a set of classes that describes the allowable moods. For this, we follow [15] in which two dimensions are identified: arousal and valence. These dimensions create a two-dimensional plane with four areas, dividing the plane in positive and negative parts on both dimensions. The moods in this plane range from "angry" and "nervous" (negative valence, positive arousal) to "happy" and "excited" (positive valence and arousal) and from "sad" and "sleepy" (negative valence and arousal) to "relaxed" and "calm" (positive valence, negative arousal). These words used here are only used as examples, indicative of the area in the plane. Other emotionally-laden words can also be placed in this plane.

To be able to work with a more fine grained set of classes, we partition the arousal/valence plane into sixteen parts. This divides both arousal and valence axes into four parts (two on the positive and two on the negative side of the axis). The arousal parts are called A–D and the valence parts 1–4, which leads to individual classes described by a letter and a number. Based on this division, we investigate four different class divisions. The first division uses all sixteen classes. This division is called *fine-grained* and ranges from A1–D4. The second, *arousal*, and third, *valence*, focus only on one aspect of the emotional plane. These class divisions are created by merging four classes. They correspond to only using A–D and 1–4 of the fine-grained division, respectively. Finally, we will use the *Thayer* division, which clusters all fine-grained classes into four areas based on the positive/negative areas in Thayer's arousal/valence plane.

2.2 Features

We will experiment with a collection of features. These are divided into two classes: *global* and *word-based*. The global features describe an aspect of the lyric as a whole. Here, we have experimented with very simple features, which should be treated as an informed baseline. We consider character count *cc* (the number of characters in a lyric), word count *wc* (the number of words in a lyric) and line count *lc* (the number of lines in a lyric).

The word-based features are more complex and use information of the specific words used and their typical occurrence in the lyrics of a particular mood. These features are heavily influenced by metrics from the field of information retrieval [16]. In particular, we use the tf*idf metric and its components. This is a powerful technique to emphasize the importance of a term (word) compared to all documents in a large document collection [14]. Originally, tf*idf was devised to search for relevant documents in large document collections given one or more search terms. The metric is used to compute relevance of the documents with respect to the search terms.

In this research, we consider the use of tf*idf to describe the relative importance of a word for a particular mood class. In contrast to the typical context, however, we start with the lyrics of a song instead of search keywords. The tf*idf value of each word in the lyrics under consideration is used as weights to indicate relevance with respect to mood classes. This allows us to compute which mood is most relevant given lyrics, where the mood is described by the combined lyrics of all songs that have that particular mood assigned.

The approach sketched indicates that we take the lyrics of all songs of a particular mood and combine them as if they are one document. This "document" can be seen as describing a particular mood. This means that there will be as many documents as there are moods. Each mood class corresponds to one document.

The tf*idf metric consists of two components: term frequency (tf) and the inverse document frequency (idf). These components are multiplied when computing the tf*idf.

The first word-based feature is the term frequency (tf). This metric measures the importance of word t_i in document, i.e. mood, d_j with $n_{i,j}$ occurrences of the word in document d_j, divided by the sum of the number of occurrences of all words in document d_j.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad (1)$$

In this situation, it measures the number of times a word occurs with a particular document (or mood). Words occurring more often in the lyrics of a particular mood will have a higher tf for that mood.

The problem with using term frequency is that most words that typically occur very often are function words, such as "the", "a" or "in". These words are not likely to help in classifying lyrics to moods as they do not represent terms that typically describe a mood. What we are really interested in are words that occur in only a sub-set (or only

one) of the moods. The inverse document frequency (idf) measures the importance of the word with respect to a document.

$$idf_i = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|} \quad (2)$$

The total number of documents (representing moods) D is divided by the number of documents in which the word (t_i) appears, and taking the logarithm of that quotient. The idf measures the importance of a word combined with a specific mood against all moods. In this particular situation, idf will be high if it occurs in the text of one or only few moods and will be low when it occurs in multiple moods (or even zero when it occurs with all moods).

The idf value by itself is not particularly useful as it is too course-grained (especially when there are only a handful of moods), but can be multiplied to weigh the tf value, resulting in the tf*idf.

$$tf * idf_{i,j} = tf_{i,j} \times idf_i \quad (3)$$

The tf*idf is used to calculate the relevance of a word for a particular mood: high tf*idf values indicate high relevance of the word to the mood.

The tf*idf provides for one particular word, a score or weight for each of the classes. Lyrics typically contain more than one word, which allows for a more robust computation of the relevance of the mood document for the lyrics under consideration. Practically, we can combine the tf*idf values of all the words in the lyrics for classification by adding the values of the separate words.

Taking the lyrics of all songs of a particular mood as one document results in having between four and sixteen documents (depending on the mood division). This is significantly less than the amount of documents normally under consideration in a tf*idf setting. In fact, many words will occur in all moods, which means that in those cases the idf will be zero which results in a zero tf*idf for all mood classes for that word. This turns out to be a very useful aspect of tf*idf weights in small document collections. In particular, words that do not help in deciding the correct mood of lyrics, such as function words, are automatically filtered out, as their tf*idf value will be zero. There is no way these words can contribute to the final weight of the lyrics, so there is no need to consider these words when analyzing the lyrics.

To investigate whether the zero tf*idf scores really are useful, we also experimented with Laplace smoothing, also known as "add-one smoothing", which reduces the amount of words that have a zero tf*idf. Before computing idf, one is added to the total number of documents. This means that the idf will now always be non-zero, albeit very small. In the case where normally the idf would be zero, the idf will now be small and the same for all classes, but this allows the system to use the information from the tf (which is not possible if idf is zero).

A potential advantage of the smoothed tf*idf is that in the case of all words having a zero non-smoothed tf*idf (for example in the case of very short lyrics), which leads to a zero tf*idf for all classes (and requiring a random choice for the class), the smoothing lets the system back-off to using tf. By not multiplying tf with zero (idf), the tf is retained in the final score, which makes it still possible to classify using tf.

Another extension that is implemented normalizes over the length of the lyrics. The tf can be larger if longer lyrics are used (simply because more words are present in those lyrics). The normalized tf*idf simply divides the tf*idf values computed from the lyrics by the length (i.e. number of words) of the lyrics. This should remove the preference for higher tf in longer lyrics.

Using tf*idf has several advantages. No linguistically motivated tools are required and the approach is inherently language independent. There is no need for the lyrics to be English (or any other language). The simple occurrence of the same words in the training data and in the test data will allow for classification. Obviously, it may be the case that certain words in one language may also occur in another language, but we expect that lyrics in different languages typically use different words. However, more research into the impact of language dependency needs to be done.

3. RESULTS

To measure the effectiveness (and illustrate the feasibility) of using tf*idf in classifying songs into moods, we set up a set of experiments. Taking a collection of songs of which the mood class is known, we extract the lyrics and apply a machine learning classifier to these, allowing us to classify the lyrics into classes based on the different class divisions. For each of these combinations, we discuss the results.

3.1 Experimental settings

To be able to train a machine learning classifier and to evaluate our experiments, we require a data set containing a set of pairs of song (or at least the lyrics) and the corresponding mood. The data set is provided by Crayonroom (`http://www.crayonroom.com/`), a small company creating music applications. The data set comes from their Moody application.

Moody lets users tag songs in iTunes in order to generate mood-based playlists. The tagging is done by manually assigning colors to songs where each color corresponds to a particular mood. The user can choose between 16 moods, which are presented in a four by four square. The colors provided are similar to the hue colors of mood [1]. Note that according to Voong and Beale [17] it is easier for a user to tag using colors instead of tagging using keywords.

The mood information is stored in the comment field of the song's ID3-tag and is exported to Moody's database. The information stored in Moody's database, which contains artist and song title information combined with the mood tag can also be used to automatically tag new songs. This application relies on user input to collect the mood information, but using that information it also helps users tag more songs in their personal collection. As such, it can be seen as a Web 2.0 application, which relies on collaborative tagging of songs.

The mood set used by Moody corresponds well with the two dimensional mood plane by Thayer [15]. The sixteen classes, placed in a four by four grid, can be mapped exactly on the plane with four mood tags in each of the areas in the plane.

Crayonroom provided us with a set of 10,000 random entries from the Moody database. This is a subset of the entire database containing mostly popular songs in different genres. Most of the songs have an English title, but there has not been an explicit selection of songs that have lyrics (as this information is not present in the database itself).

The information we received is a list of pairs of artist and song title, combined with the corresponding mood tag. Based on this information we started collecting the lyrics of the songs. Many lyrics can be found online, so we used the artist and song titles to find the lyrics automatically. This was done by automatically searching a collection of lyrics databases given the artist and song information.

Unfortunately, all spaces were removed from the artist and title fields in the database. This makes automatically finding lyrics hard. Furthermore, there are situations such as "AC/DC" which may be spelled in different ways, such as "AC DC", "AC-DC", or "ACDC". We experimented with several heuristics to re-introduce spaces and reduce the punctuation problems in the artist and song fields. Applying these heuristics and trying to find the resulting artists and song titles led to 5,631 lyrics to be found in the online databases.

The lyrics were then cleaned up and normalized. All HTML information was removed, leaving plain text lyrics. Furthermore, labels such as "chorus", "repeat until fade out" and "4x" were removed as they are not properly part of the lyrics. We realize that this may influence the count of certain words in the lyrics. However, it is often unclear, for instance, where the chorus ends exactly. Similarly, it is often unclear how many repeats are required (in the case of "repeat until fade out"). Simply removing these labels will affect the tf, but apart from manually analyzing the music and correcting all lyrics, we do not see an easy solution. Manual correction is not a feasible alternative at the moment.

From the found lyrics we extracted features and combined them together with their mood tag into machine learning instances. Each instance corresponds to one song. This information is then used for training and testing (allowing for evaluation) in a machine learning setting. The different class divisions and the distributions of instances can be found in Table 1 and Table 2.

Each of the experiments is computed using ten fold cross-validation. This meant that the collection of songs is divided into ten parts and ten experiments are performed, leaving out one part for evaluation. It is important to realize that for the tf*idf features, the tf*idf values for each of the words are recomputed for each experiment. This is needed, because the distribution of words in the training data may be different for each experiment. Intermediate tf*idf tables are computed from the training data first, which are then used to compute the actual tf*idf values for the lyrics to be classified. Similarly, the tf*idf tables will be different for each of the different class divisions.

Fine-grained		Arousal A	B	C	D	
Valence	1	295	236	248	182	961
	2	387	575	564	261	1,787
	3	360	650	531	205	1,746
	4	253	413	338	133	1,137
		1,295	1,874	1,681	781	5,631

Table 1. Distribution of instances: fine-grained (16 classes), Valence and Arousal (both 4 classes).

1 = A3+A4+B3+B4	1,676
2 = A1+A2+B1+B2	1,493
3 = C1+C2+D1+D2	1,255
4 = C3+C4+D3+D4	1,207

Table 2. Distribution of instances: Thayer division (4 classes).

Also keep in mind that for the computation of the tf*idf values, all lyrics belonging to a particular class are combined to serve as one document as described above. When computing the features for each instance separately, the tf*idf values that have been computed beforehand (and stored in a tf*idf table) are used to compute tf*idf scores for each of the classes.

To classify the test data we used TiMBL, a k-NN classifier [5]. This classifier has been developed at Tilburg University and contains a collection of algorithms with many parameters to set. In the experiments described here, we simply used the default parameter setting. This means that the IB1 algorithm (k nearest distances with $k = 1$) is used with the weighted overlap metric and GainRatio weighting. This means that higher accuracy scores may be reached when fine-tuning the classifier parameters. In this paper, we are mostly interested in the feasibility of the approach.

3.2 Experimental results

The results of applying TiMBL to the mood data are summarized in Table 3. The table shows results on the four different mood divisions and different feature settings.

The baseline shown in the table is the majority class baseline. This shows that the data is relatively well balanced as can also be seen from Tables 1 and 2. Keep in mind that the Arousal, Valence, and Thayer divisions all contain four classes, whereas fine-grained is a 16 class division. A completely random distribution of instances would lead to a baseline of 25.00 (four classes) and 6.25 (sixteen classes).

All global features and all of their combinations have worse performance with respect to the baseline. It turns out that the information present in these features is simply not specific enough. For instance, one of the initial ideas we had before we started this research, that the length of the lyrics may be different for lyrics in the different classes,

	Arousal		Valence		Thayer		Fine-grained	
Baseline	33.28		31.74		29.76		11.54	
cc	30.12	(1.77)	29.02	(1.08)	28.15	(1.92)	9.73	(0.67)
wc	31.44	(0.84)	32.59	(1.62)	28.16	(1.65)	11.06	(1.09)
lc	31.81	(1.45)	29.37	(1.69)	27.58	(0.85)	9.85	(0.84)
cc+wc	29.02	(2.26)	28.84	(1.71)	28.47	(2.00)	8.77	(0.90)
cc+lc	29.61	(1.13)	27.77	(1.74)	26.94	(1.74)	8.12	(0.78)
wc+lc	28.92	(1.28)	28.43	(2.05)	27.01	(1.08)	7.74	(0.91)
cc+wc+lc	28.42	(1.69)	27.65	(1.96)	27.03	(1.84)	8.08	(0.84)
tf	33.36	(0.18)	31.77	(0.13)	29.85	(0.15)	11.45	(0.12)
tf*idf	77.18	(1.02)	76.26	(2.03)	75.79	(1.34)	70.89	(1.51)
tf+tf*idf	77.23	(1.02)	76.29	(2.07)	75.85	(1.37)	70.89	(1.50)

Table 3. Mean accuracy and standard deviation of different feature settings and class divisions.

is not true. This is also reflected in the other features used. To allow for classification into moods, more specific information is required.

All advanced features based on tf*idf (apart from tf by itself) significantly outperform the baseline. The tf by itself does not help to classify songs into the correct mood class. The reason for this is that the words that occur most frequently (typically function words, such as "the") greatly outnumber the content words. Even when function words occur approximately the same for each class, minor variations still have a large impact with respect to otherwise more useful content words, which normally do not occur very often. For classification purposes, we are mostly interested in words that help identifying the mood of the lyrics. Words that occur in the lyrics of all moods have limited usefulness. Unfortunately, because function words occur most frequent, they have a large impact on the tf.

When adding idf, which happens with the tf*idf features, the accuracy goes up dramatically. Adding idf removes (or reduces) all weights for words that occur in lyrics of all or several classes. This means that only words that do not occur in lyrics of all moods remain or have a higher impact. This metric seems to coincide with the notion of usefulness that we are trying to implement.

The words with the highest tf*idf score for a particular class are not what we expected. These are words that occur very frequently in only one song. Examples of words with high idf and tf are: "aaah", "dah", or "yoy". However, these words are not often used in classification either.

The results of the experiments using a combination of the tf*idf metric and the tf metric is slightly better than simply using the tf*idf metric only. We expect that this has to do with the situation where there are none or not many words with a non-zero tf*idf in the lyrics. This may occur, for instance, when a song contains non-English lyrics. In that case, the tf*idf values are too often zero, but the tf features allow for a back-off strategy. The differences, however, are minor and non-significant.

As mentioned earlier, we have also implemented a normalized (dividing by the number of words in all the lyrics of a particular mood) and a Laplace smoothed version of the metrics. Since the normalization and smoothing can also be applied together, this leads to three more versions of all the tf and tf*idf experiments described so far. The results of these experiments are not shown in Table 3 as these experiments yield exactly the same mean accuracy and standard deviation as the normal tf and tf*idf features. Obviously, the tf and tf*idf values for each of the words are different in each case, but the classification is the same.

We think that length normalization does not help in classification because the length of the lyrics in each class is too similar. This means that all tf*idf values are divided by a (near) constant. Effectively, similar figures are then used to classify. Furthermore, Laplace smoothing does not help because most of the time the lyrics contain enough non-zero idf words to allow for correct classification. Additionally, when smoothing, words occurring in all classes are used as well, but since they occur in all classes, they do not have a large impact in deciding the correct class.

The different class divisions (arousal, valence, Thayer, and fine-grained) were devised to show which aspect of emotion is easiest to classify. The results show that at least using the technique described here, there is no clear difference. We originally thought that valence would be easier to classify. Positive or negative moods can easily be described using words such as "happy" and "sad". However, the intensity (described by arousal) can just as easily be classified. Most interesting is the fact that the fine-grained class division can be classified effectively as well. Remember that the fine-grained division has sixteen classes whereas the other divisions only have four.

4. CONCLUSION AND FUTURE WORK

This paper describes an attempt to design, implement and evaluate a mood-based classification system for music based on lyrics. The ultimate aim is the automatic assignment of mood-based tags for songs in a users' music database, based on lyrics only. By automatically assigning mood tags to songs, users do not have to assign mood properties to all songs in a potentially large music collection manually. Having access to the mood information ultimately allows for the easy creation of playlists based on moods.

To measure the usefulness of words in lyrics with respect to the mood classes, we used a standard information retrieval metric: tf*idf. This metric is normally used to

measure relevance of terms with respect to documents in a large document collection, but when the same metric is used in a very small set of documents, it shows some interesting and useful properties. The main property used here is that in very small document collections, the tf*idf filters out words occurring in all documents. These are words that are not useful for finding out which document (mood in our case) fits best.

The results show that the tf*idf feature improves the results significantly with respect to the majority class baseline. This shows that tf*idf can be used effectively to identify words that typically describe mood aspects of lyrics. This outcome shows that the lingual part of music reveals useful information on mood.

One has to keep in mind that the experiments reported here only take the linguistic aspects of songs into account. In order to improve results further, other characteristics, such as tempo, timbre or key, should be taken into consideration as well. However, using these aspects requires access to the music (in addition to the lyrics).

The evaluation of the current system is against the mood tags provided by Moody. These tags are based on human annotation. However, it may be that different people assign (slightly) different tags to the songs. We do not know exactly how this is handled in the Moody application, but this may have an impact on the evaluation of the system. Also, we do not know what the inter-annotator agreement is. In future research we need to consider this potential spread of human annotation, for example by taking the confidence of the system for the different moods into account.

A related problem is that the boundaries between the different moods is not clear-cut. A possible solution to this problem and that of the possible variation of annotation is to evaluate using a metric that takes distances between moods in to account. For instance, classifying A2 instead of A1 is better than classifying D4.

5. REFERENCES

[1] A. Albert. Color hue and mood: The effect of variation of red hues on positive and negative mood states. *Journal of the Behavioral Sciences*, 1, 2007.

[2] Chris Anderson. The long tail. *Wired*, 12.10, October 2004.

[3] C.J. Beukeboom and G.R. Semin. How mood turns on language. *Journal of experimental social psychology*, 42(5):553–566, 2005.

[4] O. Celma and P. Lamere. Tutorial on music recommendation. Eight International Conference on Music Information Retrieval: ISMIR 2007; Vienna, Austria, 2007.

[5] Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. TiMBL: Tilburg memory-based learner. Technical Report ILK 02-10, Tilburg University, Tilburg, the Netherlands, November 2002.

[6] B. Haring. *Beyond the Charts: Mp3 and the Digital Music Revolution*. JM Northern Media LLC, 2000.

[7] Hui He, Jianming Jin, Yuhong Xiong, Bo Chen, Wu Sun, and Ling Zhao. Language feature mining for music emotion classification via supervised learning from lyrics. In *Proceedings of the Third International Symposium, ISICA 2008; Wuhan, China*, volume 5370 of *Lecture Notes in Computer Science*, pages 426–435, Berlin Heidelberg, Germany, December 2008. Springer-Verlag.

[8] Xiao Hu, J. Stephen Downie, and Andreas F. Ehman. Lyric text mining in music mood classification. In *Proceedings of the tenth International Society for Music Information Retrieval Conference (ISMIR); Kobe, Japan*, pages 411–416, October 2009.

[9] Xiao Hu, J. Stephen Downie, Cyril Laurier, Mert Bay, and Andreas F. Ehrmann. The 2007 mirex audio mood classification task: Lessons learned. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR08), Philadelphia:PA, USA*, pages 462–467, 2008.

[10] Patrik N. Juslin and Petri Laukka. Expression, perception, and induction of musical emotions: A review and a questionnaire study of everyday listening. *Journal of New Music Research*, 33(3):217–238, September 2004.

[11] Pieter Kanters. Automatic mood classification for music. Master's thesis, Tilburg University, Tilburg, the Netherlands, June 2009.

[12] C. Laurier, J. Grivolla, and P. Herrera. Multimodal music mood classification using audio and lyrics. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA08); San Diego:CA, USA*, pages 688–693, Los Alamitos:CA, USA, 2008. IEEE Computer Society Press.

[13] O.C. Meyers. A mood-based music classification and exploration system. Master's thesis, Massachusetts Institute of Technology, Cambridge:MA, USA, 2007.

[14] J. Ramos. Using tf-idf to determine word relevance in document queries. In *First International Conference on Machine Learning*, New Brunswick:NJ, USA, 2003. Rutgers University.

[15] R.E. Thayer. *The Biopsychology of Mood and Arousal*. Oxford University Press, New York:NY, USA, 1982.

[16] C. J. van Rijsbergen. *Information Retrieval*. University of Glasgow, Glasgow, UK, 2nd edition, 1979. Printout.

[17] Michael Voong and Russell Beale. Music organisation using colour synaesthesia. In *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, pages 1869–1874, New York:NY, USA, 2007. ACM Press.

[18] Y. Yang, C. Liu, and H. Chen. Music emotion classification: A fuzzy approach. In *Proceedings of the 14th annual ACM international conference on Multimedia; Santa Barbara:CA, USA*, pages 81–84, New York:NY, USA, 2007. ACM Press.

AUTOMATIC MUSIC TAGGING WITH TIME SERIES MODELS

Emanuele Coviello
University of California, San Diego
Dept. of Electrical and Computer Engineering
ecoviell@ucsd.edu

Luke Barrington
University of California, San Diego
Dept. of Electrical and Computer Engineering
lukeinusa@gmail.com

Antoni B. Chan
City University of Hong Kong
Dept. of Computer Science
abchan@cityu.edu.hk

Gert. R. G. Lanckriet
University of California, San Diego
Dept. of Electrical and Computer Engineering
gert@ece.ucsd.edu

ABSTRACT

State-of-the-art systems for automatic music tagging model music based on bag-of-feature representations which give little or no account of temporal dynamics, a key characteristic of the audio signal. We describe a novel approach to automatic music annotation and retrieval that captures temporal (e.g., rhythmical) aspects as well as timbral content. The proposed approach leverages a recently proposed song model that is based on a generative time series model of the musical content — the dynamic texture mixture (DTM) model — that treats fragments of audio as the output of a linear dynamical system. To model characteristic temporal dynamics and timbral content at the tag level, a novel, efficient hierarchical EM algorithm for DTM (HEM-DTM) is used to summarize the common information shared by DTMs modeling individual songs associated with a tag. Experiments show learning the semantics of music benefits from modeling temporal dynamics.

1. INTRODUCTION

This paper concerns *automatic tagging* of music with descriptive keywords (e.g., genres, emotions, instruments, usages, etc.), based on the content of the song. Music annotations can be used for a variety of purposes, such as searching for songs exhibiting specific qualities (e.g., "jazz songs with female vocals and saxophone"), or retrieval of semantically similar songs (e.g., generating playlists based on songs with similar annotations).

State-of-the-art music "auto-taggers" model a song as a "bag of audio features" [7, 9–11, 14]. The bag of features representation extracts audio features from the song at a regular time interval, but then treats these features independently, ignoring the temporal order or dynamics between them. Hence, this representation fails to account for the longer term musical dynamics (e.g. tempo and beat) or temporal structures (e.g. riffs and arpeggios), which are clearly important characteristics of a musical signal.

We address this limitation by adopting the dynamic texture (DT) model [6], a generative, *time-series model* of musical content that captures longer-term time dependencies. The DT model is similar to the Hidden Markov model (HMM) which has proven robust in music identification [12]. The difference is that HMMs require to quantize the audio signal into a fixed number of discrete "phonemes", while the DT has a continuous state space that is a more flexible model for music.

Musical time series often show significant structural changes within a single song and have dynamics that are only locally homogeneous. Hence, [1] proposes to model the audio fragments from a single song as a dynamic texture mixture (DTM) model [3], for the task of automatic music *segmentation*. These results demonstrated that the DTM provides an accurate segmentation of music into homogeneous, perceptually similar segments (corresponding to what a human listener would label as 'chorus', 'verse', 'bridge', etc.) by capturing *temporal* as well as *textural* aspects of the musical signal.

In this paper, we adopt the DTM model to propose a novel approach to the task of automatic music *annotation* that accounts for both the timbral content and the temporal dynamics that are predictive of a semantic tag. We first model all songs in a music database as DTMs, capturing longer-term time dependencies and instantaneous spectral content at the *song-level*. Second, the characteristic temporal and timbral aspects of musical content that are commonly associated with a semantic tag are identified by learning a *tag-level* DTM that summarizes the common features of a (potentially large) set of song-level DTMs for the tag. Given all song-level DTMs associated with a particular tag, the common information is summarized by clustering similar song-level DTs using a novel, efficient hierarchical EM (HEM-DTM) algorithm. This gives rise to a tag-level DTM with few mixture components (as opposed to tag-level Gaussian mixture models in [14], which do not capture temporal dynamics). Experimental results show that the proposed time-series model improves annotation and retrieval, in particular for tags with temporal dynamics that unfold in the time span of a few seconds.

The remainder of this paper is organized as follows. In Section 2, we present the annotation and retrieval system using time-series data, while in in Section 3, we present an efficient hierarchical EM algorithm for dynamic texture mixtures. Finally, in Sections 4 and 5, we present experiments using DTM for music annotation and retrieval.

2. ANNOTATION AND RETRIEVAL

In this section we formulate the tasks of annotation and retrieval of audio data as a semantic multi-class labeling (SML) problem [2] in the context of time-series models.

2.1 Notation

A song s is represented as a collection of T overlapping time series $\mathcal{Y} = \{y^1_{1:\tau}, \ldots, y^T_{1:\tau}\}$, where each $y^t_{1:\tau}$ represents τ sequential audio feature vectors extracted by passing a short-time window over the audio signal (also called an audio *fragment*). The number of fragments, T, depends on the length of the song. The semantic content of a song with respect to a vocabulary $\mathcal{V}$ of size $|\mathcal{V}|$ is represented in an annotation vector $\mathbf{c} = [c_1, \ldots, c_{|\mathcal{V}|}]$, where $c_k > 0$ only if there is a positive association between the song and the word w_k, otherwise $c_k = 0$. Each *semantic weight*, c_k, represents the degree of association between the song and word w_k. The data set $\mathcal{D}$ is a collection of $|\mathcal{D}|$ song-annotation pairs $(\mathcal{Y}_d, \mathbf{c}_d)$.

2.2 Music Annotation

We treat annotation as a semantic multi-class problem [2, 14] in which each class is a word w, from a vocabulary $\mathcal{V}$ of unique tags (e.g., "bass guitar", "hip hop", "boring"). Each word w_k is modeled with a probability distribution over the space of audio fragments, $p(y^t_{1:\tau}|w_k)$. The annotation task is to find the subset $\mathcal{W} = \{w_1, \ldots, w_{\mathcal{A}}\} \subseteq \mathcal{V}$ of $\mathcal{A}$ words that best describe a novel song $\mathcal{Y}$.

Given the audio fragments of a novel song $\mathcal{Y}$, the most relevant words are the ones with highest posterior probability, computed using Bayes' rule:

$$p(w_k|\mathcal{Y}) = \frac{p(\mathcal{Y}|w_k)\, p(w_k)}{p(\mathcal{Y})}, \qquad (1)$$

where $p(w_k)$ is the prior of the k^{th} word, and $p(\mathcal{Y}) = \sum_{k=1}^{|\mathcal{V}|} p(\mathcal{Y}|w_k)p(w_k)$ is the song prior. To promote annotation using a diverse set of words, we assume an uniform prior, $p(w_k) = 1/|\mathcal{V}|$. We follow [14] in estimating the likelihood term in (1) with the geometric average of the individual sequence likelihoods, $p(\mathcal{Y}|w_k) = \prod_{t=1}^{T} \left(p(y^t_{1:\tau}|w_k)\right)^{\frac{1}{T}}$. Note that, unlike bag-of-features models that discard any dependency between audio features vectors (each describing milliseconds of audio), we only assume independence between different *sequences* of audio feature vectors (describing seconds of audio). Correlations within a single sequence are accounted for by the model presented in Section 3.

The probability that the song $\mathcal{Y}$ can be described by word w_k is

$$p_k = p(w_k|\mathcal{Y}) = \frac{\prod_{t=1}^{T} \left(p(y^t_{1:\tau}|w_k)\right)^{\frac{1}{T}}}{\sum_{l=1}^{|\mathcal{V}|} \prod_{t=1}^{T} \left(p(y^t_{1:\tau}|w_l)\right)^{\frac{1}{T}}}. \qquad (2)$$

Finally, the song can be represented as a semantic multinomial, $\mathbf{p} = [p_1, \ldots, p_{|\mathcal{V}|}]$, where each $p_k = p(w_k|\mathcal{Y})$ represents the relevance of the k^{th} word for the song, and $\sum_{i=1}^{|\mathcal{V}|} p_i = 1$. We annotate a song with the most likely tags according to $\mathbf{p}$, i.e., we select the tags with the words with the largest probability.

2.3 Music Retrieval

Given a query word, songs in the database can be retrieved based on their relevance to the semantic query word [1]. In particular, the song's relevance to the query word w_k is equivalent to the posterior probability of the word, $p(w_k|\mathcal{Y})$ in (2). Hence, retrieval involves rank-ordering the songs in the database based on the k-th entry (p_k) of the semantic multinomials $\mathbf{p}$.

2.4 Learning DTM tag models

In this paper, we model the tag-level distributions, $p(y^t_{1:\tau}|w_k)$, as dynamic texture mixture models. The tag-level distributions are estimated from the set of training songs associated with the particular tag. One approach is to extract all the audio fragments from the relevant training songs, and then run the EM algorithm [3] directly on this data to learn the tag-level DTM. This approach, however, requires storing large amounts of audio fragments in memory (RAM) for running the EM algorithm. For even modest-sized databases, the memory requirements can exceed the RAM capacity of most computers.

To allow efficient training in both computation time and memory requirements, we propose to break the learning procedure into two steps. First, a DTM is estimated for each song using the standard EM algorithm [3]. Next, each tag-level model is estimated using the hierarchical EM algorithm on all the song-level DTMs associated with the particular tag. Because the database is first processed at the song-level, the computation can be easily done in parallel (over the songs) and the memory requirement is greatly reduced to that of processing a single song. The memory requirements for computing the tag-level models is also reduced, since each song is succinctly modeled by the parameters of a DTM.

Such a reduction in computational complexity also ensures that the tag-level models can be learned from cheaper, weakly-labeled data (i.e., missing labels, labels without segmentation data, etc.) by pooling over large amounts of audio data to amplify the appropriate attributes. In summary, adopting DTM, or time-series models in general, as a tag-model for SML annotation requires an appropriate HEM algorithm for efficiently learning the tag-level models from the song-level models. In the next section, we review the DTM and present the HEM algorithm for DTM.

3. HIERARCHICAL EM FOR DTMS

In this section, we first review the dynamic texture (DT) and dynamic texture mixture (DTM) models for modeling musical time-series. We then present the hierarchical EM algorithm for efficiently learning a tag-level DTM from a set of song-level DTMs.

3.1 The Dynamic Texture Model

A dynamic texture [6] (DT) is a generative model that takes into account both the acoustics and the dynamics of audio sequences [1]. The model consists of two random variables, y_t, which encodes the acoustic component (audio

[1] Note that although this work focuses on single-word queries, our representation easily extends to multiple-word queries [13].

feature vector) at time t, and x_t, which encodes the dynamics (evolution) of the acoustic component over time. The two variables are modeled as a *linear dynamical system*,

$$x_t = Ax_{t-1} + v_t, \quad (3)$$

$$y_t = Cx_t + w_t + \bar{y}, \quad (4)$$

where $x_t \in \mathbb{R}^n$ and $y_t \in \mathbb{R}^m$ are real vectors (typically $n \ll m$). Using such a model, we assume that the dynamics of the audio can be summarized by a more parsimonious ($n < m$) *hidden state process* x_t, which evolves as a first order Gauss-Markov process, and each *observation variable* y_t, which encodes the acoustical component (audio feature vector at time t) is dependent only on the current hidden state x_t.

The matrix $A \in \mathbb{R}^{n\times n}$ is a *state transition matrix*, which encodes the dynamics or evolution of the hidden state variable (e.g., the evolution of the audio track), and the matrix $C \in \mathbb{R}^{m\times n}$ is an *observation matrix*, which encodes the basis functions for representing the audio sequence. The vector $\bar{y} \in \mathbb{R}^n$ is the mean of the dynamic texture (i.e. the mean audio feature vector). v_t is a *driving noise process*, and is zero-mean Gaussian distributed , e.g., $v_t \sim \mathcal{N}(0, Q)$, where $Q \in \mathbb{R}^{n\times n}$ is a covariance matrix. w_t is the *observation noise* and is also zero-mean Gaussian, e.g., $w_t \sim \mathcal{N}(0, R)$, where $R \in \mathbb{R}^{m\times m}$ is a covariance matrix. Finally, the *initial condition* is specified as $x_1 \sim \mathcal{N}(\mu, S)$, where $\mu \in \mathbb{R}^n$ is the mean of the initial state, and $S \in \mathbb{R}^{n\times n}$ is the covariance. The dynamic texture is specified by the parameters $\Theta = \{A, Q, C, R, \mu, S, \bar{y}\}$.

Intuitively, the columns of C can be interpreted as the principal components (or basis functions) of the audio features vectors over time. Hence, each audio feature vector y_t can be represented as a linear combination of principal components, with corresponding weights given by the current hidden state x_t. In this way, the DT can be interpreted as a time-varying PCA representation of an audio feature vector time-series.

3.2 The Dynamic Texture Mixture Model

A song is a combination of heterogeneous sequences with significant structural variations, and hence is not well represented as a single DT model. To address this lack of global homogeneity, [1] proposed to represent audio fragments, extracted from a song, as samples from a dynamic texture mixture (DTM) [3], effectively modeling local structure of the song. The DTM model [3] introduces an assignment random variable $z \sim \text{multinomial}(\pi_1, \cdots, \pi_K)$, which selects one of the K dynamic texture components as the source of the audio fragment. Each mixture component is parameterized by $\Theta_z = \{A_z, C_z, Q_z, R_z, \mu_z, S_z, \bar{y}_z\}$, and the DTM model is parameterized by $\Theta = \{\pi_z, \Theta_z\}_{z=1}^K$.

Given a set of audio samples, the maximum-likelihood parameters of the DTM can be estimated with recourse to the expectation-maximization (EM) algorithm [3], which is an iterative optimization method that alternates between estimating the hidden variables with the current parameters, and computing new parameters given the estimated hidden variables (the "complete data"). The EM algorithm for DTM alternates between estimating second-order statistics of the hidden-states, conditioned on each audio sequence, with the Kalman smoothing filter (E-step), and computing new parameters given these statistics (M-step).

Previous work in [1] has successfully used the DTM for the task of segmenting the structure of a song into acoustically similar sections (e.g., intro, verse, chorus, bridge, solo, outro). In this work, we demonstrate that the DTM can also be used as a tag-level annotation model for music annotation and retrieval. We next present a hierarchical EM algorithm for efficiently estimating these tag-level DTMs from large sets of song-level DTMs, previously estimated for the set of training songs associated with a tag.

3.3 Hierarchical EM for learning DTM hierarchies

Given a DTM model of each training song as learned in the previous section, the goal now is to learn a tag-level DTM model that summarizes the common features of the corresponding song-level DTMs. First, all song-level DTMs with a particular tag are pooled together into a single, large DTM. Next, the common information is summarized by clustering similar DT components together, forming a new *tag-level* DTM with fewer mixture components.

The DT components are clustered using the hierarchical expectation-maximization (HEM) algorithm [15]. At a high level, this is done by generating *virtual samples* from each of the song-level component models, merging all the samples, and then running the standard EM algorithm on the merged samples to form the reduced tag-level mixture. Mathematically, however, using the virtual samples is equivalent to marginalizing over the distribution of song-level models. Hence, the tag model can be learned directly and efficiently from the parameters of the song-level models, without generating any virtual samples.

The HEM algorithm was originally proposed in [15] to reduce a Gaussian mixture model (GMM) with many components to a representative GMM with fewer components and has been successful in learning GMMs from large datasets for the annotation and retrieval of images [2] and music [14]. We next present an HEM algorithm for mixtures with components that are *dynamic textures* [4].

3.3.1 HEM Formulation

Formally, let $\Theta^{(s)} = \{\pi_i^{(s)}, \Theta_i^{(s)}\}_{i=1}^{K^{(s)}}$ denote the combined song-level DTM with $K^{(s)}$ components, where $\Theta_i^{(s)}$ are the parameters for the i^{th} DT component. The likelihood of observing an audio sequence $y_{1:\tau}$ with length τ from the combined song-level DTM $\Theta^{(s)}$ is given by

$$p(y_{1:\tau}|\Theta^{(s)}) = \sum_{i=1}^{K^{(s)}} \pi_i^{(s)} p(y_{1:\tau}|z^{(s)} = i, \Theta^{(s)}), \quad (5)$$

where $z \sim \text{multinomial}(\pi_1^{(s)}, \cdots \pi_{K^{(s)}}^{(s)})$ is the hidden variable that indexes the mixture components. $p(y_{1:\tau}|z^{(s)} = i, \Theta^{(s)})$ is the likelihood of the audio $y_{1:\tau}$ under the i^{th} DT mixture component, and $\pi_i^{(s)}$ is the prior weight for the i^{th} component. The goal is to find a tag-level annotation DTM, $\Theta^{(a)} = \{\pi_j^{(a)}, \Theta_j^{(a)}\}_{j=1}^{K^{(a)}}$, which

represents (5) using fewer number of mixture components, $K^{(a)}$, (i.e., $K^{(a)} < K^{(s)}$). The likelihood of observing an audio sequence $y_{1:\tau}$ from the tag-level DTM $\Theta^{(a)}$ is

$$p(y_{1:\tau}|\Theta^{(a)}) = \sum_{j=1}^{K^{(a)}} \pi_j^{(a)} p(y_{1:\tau}|z^{(a)} = j, \Theta^{(a)}), \quad (6)$$

where $z^{(a)} \sim \text{multinomial}(\pi_1^{(a)}, \cdots, \pi_{K^{(a)}}^{(a)})$ is the hidden variable for indexing components in $\Theta^{(a)}$. Note that we will always use i and j to index the components of the song-level model, $\Theta^{(s)}$, and the tag-level model, $\Theta^{(a)}$, respectively. To reduce clutter, we will also use the shorthand $\Theta_i^{(s)}$ and $\Theta_j^{(a)}$ to denote the i^{th} component of $\Theta^{(s)}$ and the j^{th} component of $\Theta^{(a)}$, respectively. For example, we denote $p(y_{1:\tau}|z^{(s)} = i, \Theta^{(s)}) = p(y_{1:\tau}|\Theta_i^{(s)})$.

3.3.2 Parameter estimation

To obtain the tag-level model, HEM [15] considers a set of N virtual observations drawn from the song-level model $\Theta^{(s)}$, such that $N_i = N\pi_i^{(s)}$ samples are drawn from the i^{th} component. We denote the set of N_i virtual audio samples for the i^{th} component as $Y_i = \{y_{1:\tau}^{(i,m)}\}_{m=1}^{N_i}$, where $y_{1:\tau}^{(i,m)} \sim \Theta_i^{(s)}$ is a single audio sample and τ is the length of the virtual audio (a parameter we can choose). The entire set of N samples is denoted as $Y = \{Y_i\}_{i=1}^{K^{(s)}}$. To obtain a consistent hierarchical clustering, we also assume that all the samples in a set Y_i are eventually assigned to the same tag-level component $\Theta_j^{(a)}$. The parameters of the tag-level model can then be estimated by maximizing the likelihood of the virtual audio samples,

$$\Theta^{(a)^*} = \arg\max_{\Theta^{(a)}} \log p(Y|\Theta^{(a)}), \quad (7)$$

where

$$\log p(Y|\Theta^{(a)}) = \log \prod_{i=1}^{K^{(s)}} p(Y_i|\Theta^{(a)}) \quad (8)$$

$$= \log \prod_{i=1}^{K^{(s)}} \sum_{j=1}^{K^{(a)}} \pi_j^{(a)} \int p(Y_i, X_i|\Theta_j^{(a)}) dX_i \quad (9)$$

and $X_i = \{x_{1:\tau}^{(i,m)}\}$ are the hidden-state variables corresponding to Y_i. Computing the log-likelihood in (9) requires marginalizing over the hidden assignment variables $z_i^{(a)}$ and hidden state variables X_i. Hence, (7) can also be solved with recourse to the EM algorithm [5]. In particular, each iteration consists of

E-Step: $\mathcal{Q}(\Theta^{(a)}, \hat{\Theta}^{(a)}) = \mathbb{E}_{X,Z|Y,\hat{\Theta}^{(a)}}[\log p(X, Y, Z|\Theta^{(a)})]$

M-Step: $\Theta^{(a)^*} = \arg\max_{\Theta^{(a)}} \mathcal{Q}(\Theta^{(a)}, \hat{\Theta}^{(a)})$

where $\hat{\Theta}^{(a)}$ is the current estimate of the tag-level model, $p(X, Y, Z|\Theta^{(a)})$ is the "complete-data" likelihood, and $\mathbb{E}_{X,Z|Y,\hat{\Theta}^{(a)}}$ is the conditional expectation with respect to the current model parameters.

As is common with the EM formulation, we introduce a hidden assignment variable $\mathbf{z}_{i,j}$, which is an indicator variable for when the audio sample set Y_i is assigned to the j^{th} component of $\Theta^{(a)}$, e.g., when $z_i^{(a)} = j$. The complete-data log-likelihood is then

$$\log p(X, Y, Z|\Theta^{(a)}) \quad (10)$$
$$= \sum_{i=1}^{K^{(s)}} \sum_{j=1}^{K^{(a)}} \mathbf{z}_{i,j} \log \pi_j^{(a)} + \mathbf{z}_{i,j} \log p(Y_i, X_i|\Theta_j^{(a)}).$$

The $\mathcal{Q}$ function is then obtained by taking the conditional expectation of (10), and using the law of large numbers to remove the dependency on the virtual samples. The result is a $\mathcal{Q}$ function that depends only on the parameters of the song-level DTs $\Theta_i^{(s)}$.

The HEM algorithm for DTM is summarized in Algorithm 1. In the E-step, the expectations in Eq. (11) are computed for each song-level DT $\Theta_i^{(s)}$ and current tag-level DT $\hat{\Theta}_j^{(a)}$. These expectations can be computed using "suboptimal filter analysis" or "sensitivity analysis" [8] on the Kalman smoothing filter (see [4]). Next, the probability of assigning the song-level DT $\Theta_i^{(s)}$ to the tag-level DT $\hat{\Theta}_j^{(a)}$ is computed according to (12), and the expectations are aggregated over all the song-level DTs in (14). In the M-step, the parameters for each tag-level component $\hat{\Theta}_j^{(a)}$ are recomputed according to the update equations in (15). More details are available in [4].

4. MUSIC DATA

In this section we describe the music collection and the audio features used in our experiments.

The CAL500 [14] dataset consists of 502 Western popular songs from the last 50 years from 502 different artists. Each song has been annotated by at least 3 humans, using a semantic vocabulary of 174 words that includes genres, instruments, vocal characteristics, emotions, acoustic characteristics, and song usages. CAL500 provides hard binary annotations, which are 1 when a tag applies to the song and 0 when the tag does not apply. We find empirically that accurately fitting the HEM-DTM model requires a significant number of training examples so we restrict our attention to the 78 tags with at least 50 examples.

A popular feature for content-based music analysis, Mel-frequency cepstral coefficients (MFCCs) concisely summarize the short-time content of an acoustic waveform by using the discrete cosine transform (DCT) to decorrelate the bins of a Mel-frequency spectral histogram [2]. In Section 3.1 we noted how the DT model can be viewed as a time varying PCA representation of the audio features. This idea suggests that we can represent the spectrum over time as the output of the DT model y_t. In this case, the columns of the observation matrix C (PCA matrix) are analogous to the DCT basis functions, and the hidden states x_t are the coefficients (analogous to the MFCCs). The advantage with this formulation is that a different C matrix, i.e., basis functions, can be learned to best represent the particular song or semantic concept of interest.

[2] This decorrelation is usually convenient in that it reduces the number of parameters to be estimated.

Algorithm 1 HEM algorithm for DTM

1: **Input**: combined song-level DTM $\{\Theta_i^{(s)}, \pi_i^{(s)}\}_{i=1}^{K^{(s)}}$, number of virtual samples N.
2: Initialize tag-level DTM, $\{\hat{\Theta}_j^{(a)}, \pi_j^{(a)}\}_{j=1}^{K^{(a)}}$.
3: **repeat**
4: {E-step}
5: Compute expectations using sensitivity analysis for each $\Theta_i^{(s)}$ and $\hat{\Theta}_j^{(a)}$ (see [4]):

$$\begin{aligned}
\hat{x}_{t|j}^{(i)} &= \mathbb{E}_{y|\Theta_i^{(s)}}\left[\mathbb{E}_{x|y,\hat{\Theta}_j^{(a)}}[x_t]\right], \\
\hat{P}_{t,t|j}^{(i)} &= \mathbb{E}_{y|\Theta_i^{(s)}}\left[\mathbb{E}_{x|y,\hat{\Theta}_j^{(a)}}[x_t x_t^T]\right], \\
\hat{P}_{t,t-1|j}^{(i)} &= \mathbb{E}_{y|\Theta_i^{(s)}}\left[\mathbb{E}_{x|y,\hat{\Theta}_j^{(a)}}[x_t x_{t-1}^T]\right], \\
\hat{W}_{t|j}^{(i)} &= \mathbb{E}_{y|\Theta_i^{(s)}}\left[(y_t-\bar{y}_j)\mathbb{E}_{x|y,\hat{\Theta}_j^{(a)}}[x_t]^T\right], \\
\hat{U}_{t|j}^{(i)} &= \mathbb{E}_{y|\Theta_i^{(s)}}\left[(y_t-\bar{y}_j)(y_t-\bar{y}_j)^T\right], \\
\hat{u}_t^{(i)} &= \mathbb{E}_{y|\Theta_i^{(s)}}\left[y_t\right], \\
\ell_{i|j} &= \mathbb{E}_{\Theta_i^{(s)}}[\log p(y_{1:\tau}|\hat{\Theta}_j^{(a)})].
\end{aligned} \tag{11}$$

6: Compute assignment probability and weighting:

$$\hat{\mathbf{z}}_{i,j} = \frac{\pi_j^{(a)} \exp\left(N_i \ell_{i|j}\right)}{\sum_{j'=1}^{K^{(a)}} \pi_{j'}^{(a)} \exp\left(N_i \ell_{i|j'}\right)} \tag{12}$$

$$\hat{\mathbf{w}}_{i,j} = \hat{\mathbf{z}}_{i,j} N_i = \hat{\mathbf{z}}_{i,j} \pi_i^{(s)} N \tag{13}$$

7: Computed aggregate expectations for each $\hat{\Theta}_j^{(a)}$:

$$\begin{aligned}
\hat{N}_j &= \textstyle\sum_i \hat{\mathbf{z}}_{i,j}, & \eta_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \hat{P}_{1,1|j}^{(i)}, \\
\hat{M}_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j}, & \gamma_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=1}^{\tau} \hat{u}_t^{(i)}, \\
\xi_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \hat{x}_{1|j}^{(i)}, & \beta_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=1}^{\tau} \hat{x}_{t|j}^{(i)}, \\
\Phi_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=1}^{\tau} \hat{P}_{t,t|j}^{(i)}, & & \\
\Psi_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t,t-1|j}^{(i)}, & & \\
\varphi_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t,t|j}^{(i)}, & & \\
\phi_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=2}^{\tau} \hat{P}_{t-1,t-1|j}^{(i)}, & & \\
\Lambda_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=1}^{\tau} \hat{U}_{t|j}^{(i)}, & & \\
\Gamma_j &= \textstyle\sum_i \hat{\mathbf{w}}_{i,j} \sum_{t=1}^{\tau} \hat{W}_{t|j}^{(i)}. & &
\end{aligned} \tag{14}$$

8: {M-step}
9: Recompute parameters for each component $\hat{\Theta}_j^{(a)}$:

$$\begin{aligned}
C_j^* &= \Gamma_j \Phi_j^{-1}, & R_j^* &= \tfrac{1}{\tau \hat{M}_j}(\Lambda_j - C_j^* \Gamma_j), \\
A_j^* &= \Psi_j \phi_j^{-1}, & Q_j^* &= \tfrac{1}{(\tau-1)\hat{M}_j}(\varphi_j - A_j^* \Psi_j^T), \\
\mu_j^* &= \tfrac{1}{\hat{M}_j}\xi_j, & S_j^* &= \tfrac{1}{\hat{M}_j}\eta_j - \mu_j^*(\mu_j^*)^T, \\
\pi_j^* &= \tfrac{\sum_{i=1}^{K^{(s)}} \hat{\mathbf{z}}_{i,j}}{K^{(s)}}, & \bar{y}_j^* &= \tfrac{1}{\tau \hat{M}_j}(\gamma_j - C_j^* \beta_j).
\end{aligned} \tag{15}$$

10: **until** convergence
11: **Output**: tag-level DTM $\{\Theta_j^{(a)}, \pi_j^{(a)}\}_{j=1}^{K^{(a)}}$.

Furthermore, since we explicitly model the temporal evolution of the spectrum, we do not need to include the instantaneous deltas of the MFCCs.

Our experiments use 34 Mel-frequency bins, computed from half-overlapping, 46ms audio segments. Each audio fragment is described by a time series $y_{1:\tau}^t$ of $\tau = 450$ sequential audio feature vectors, which corresponds to 10 seconds. Song-level DTM models are learned from a dense sampling of audio fragments of 10 seconds, extracted every 1 second.

Model	P	R	F-score	AROC	MAP	P10
HEM-GMM	0.49	0.23	0.26	0.66	0.45	0.47
CBA	0.41	0.24	0.29	0.69	0.47	0.49
HEM-DTM	0.47	0.25	0.30	0.69	0.48	0.53

Table 1. Annotation and retrieval results for HEM-DTM and HEM-GMM.

5. EVALUATION

Song-level DTMs were learned with $K = 16$ components and state-space dimension $n = 7$, using EM-DTM. Tag-level DTMs were learned by pooling together all song models associated with a given tag and reducing the result to a DTM with $K^{(r)} = 2$ components with HEM-DTM. To reduce the effect of low likelihoods in high dimensions, we normalize the single-segment likelihood terms, e.g., $p(y_{1:\tau}^t|w_k)$, by the length of the sequence τ.

To investigate the advantage of the DTM's temporal representation, we compare the auto-tagging performance of our model (HEM-DTM) to the hierarchically trained Gaussian mixture models (HEM-GMM) from [14], a generative model that ignores temporal dynamics. A comparison to the CBA model of [9] is provided as well. We follow the procedure of [14] for training HEM-GMMs, and our CBA implementation follows [9], with the modification that the codebook is constructed using only songs from the training set. All reported metrics are the results of 5-fold cross validation where each song appeared in the test set exactly once.

5.1 Annotation and Retrieval

Annotation performance is measured following the procedure in [14]. Test set songs are annotated with the 10 most likely tags in their semantic multinomial (Eq. 2). Annotation accuracy is reported by computing precision, recall and F-score for each tag, and then averaging over all tags. For a detailed definition of the metrics, see [14].

To evaluate retrieval performance, we rank-order test songs for each single-tag query in our vocabulary, as described in Section 2. We report mean average precision (MAP), area under the receiver operating characteristic curve (AROC) and top-10 precision (P10), averaged over all the query tags. The ROC curve is a plot of true positive rate versus false positive rate as we move down the ranked list. Random guessing would result in an AROC of 0.5. The top-10 precision is the fraction true positives in the top-10 of the ranking. MAP averages the precision at each point in the ranking where a song is correctly retrieved.

5.2 Results

Annotation and retrieval results are presented in Table 1, demonstrating superior performance for HEM-DTM, compared to HEM-GMM,for all metrics except for precision. This indicates that HEM-DTM is slightly more aggressive when annotating songs, but still its annotations are more accurate, as evidenced by the higher F-score. HEM-DTM performs better than CBA in all metrics. For retrieval, although AROC scores are comparable for CBA and HEM-DTM, HEM-DTM clearly improves the top of the ranked list more, as evidenced by the higher precision-at-10 score.

Tag	HEM-DTM		HEM-GMM	
	F-score	MAP	F-score	MAP
HEM-DTM better than HEM-GMM				
male lead vocals	0.44	0.87	0.08	0.81
female lead vocals	0.58	0.69	0.42	0.44
fast rhythm	0.40	0.48	0.20	0.42
classic rock	0.41	0.37	0.18	0.36
acoustic guitar	0.44	0.43	0.31	0.44
electric guitar	0.32	0.35	0.14	0.34
HEM-GMM better than HEM-DTM				
mellow	0.34	0.41	0.37	0.49
slow rhythm	0.45	0.60	0.44	0.62
weak	0.22	0.26	0.26	0.25
light beat	0.36	0.58	0.53	0.61
sad	0.13	0.23	0.28	0.30
negative feelings	0.27	0.33	0.35	0.36

Table 2. Annotation and retrieval results for some tags with HEM-DTM and HEM-GMM.

HEM-DTM	**classic rock**, **driving**, **energy**, **fast**, **male lead vocals**, **electric guitar**, **electric**, indifferent, powerful, rough
HEM-GMM	boring, major, acoustic, **driving**, not likeable, female lead vocals, **recording quality**, cold, **synthesized**, pop , guitar

Table 3. Automatic 10-word annotations for 'Every little thing she does is magic' by Police.

HEM-DTM performs better on average by capturing temporal dynamics (e.g., tempo, rhythm, etc.) over seconds of audio content. Modeling temporal dynamics can be expected to prove beneficial for some tags, while adding no benefit for others. Indeed, some tags might either be modeled adequately by instantaneous characteristics alone (e.g., timbre), or require a global song model. Table 2 lists annotation (F-score) and retrieval (MAP) results for a subset of our vocabulary. As expected, HEM-DTM shows strong improvements for tags associated with a clear temporal structure. For the genre "classic rock", which has characteristic tempo and rhythm, HEM-DTM achieves an F-score of approximately 0.4, doubling the performance of HEM-GMM. Similarly, HEM-DTM proves particularly suitable for tags with significant temporal structure, e.g., "male lead vocals" and "fast rhythm", or the instruments such as electric or acoustic guitar. Conversely, our HEM-DTM shows no improvement over HEM-GMM when predicting tags for which temporal structure is less significant, such as "mellow" and "negative feelings".

Finally, Tables 3 and 4 show example annotations and retrieval rankings for both HEM-DTM and HEM-GMM. Ground truth results are marked in bold.

6. CONCLUSIONS

We have presented the dynamic texture mixture model; a principled approach for capturing the temporal, as well as timbral qualities of music. We derived a hierarchical algorithm for efficiently learning DTM models from large training sets, enabling its usage as a tag model for semantic annotation and retrieval. Experimental results demonstrate that the new model improves accuracy over current bag-of-feature approaches.

Rank	HEM-DTM	
1	**James Taylor**	**'Fire and rain'**
2	**Arlo Guthrie**	**'Alices restaurant massacree'**
3	Zombies	'Beechwood park'
4	**Crosby, Stills, Nash and Young**	**'Teach your children'**
5	**Donovan**	**'Catch the wind'**
6	**American music club**	**'Jesus hands'**
7	Aaron Neville	'Tell it like it is'
8	10cc	'For you and i'
9	Byrds	'Wasn't born to follow'
10	Beautiful south	'One last love song'
Rank	HEM-GMM	
1	Stranglers	'Golden brown'
2	**Crosby, Stills, Nash and Young**	**'Teach your children'**
3	Pet shop boys	'Being boring'
4	Counting crows	'Speedway'
5	Beth quist	'Survival'
6	Beautiful south	'One last love song'
7	Neutral milk hotel	'Where you'll find me now'
8	Police	'Every little thing she does is magic'
9	Eric clapton	'Wonderful tonight'
10	Belle and Sebastian	'Like Dylan in the movies'

Table 4. Top retrieved songs for 'acoustic guitar'.

Acknowledgments E.C., L.B. and G.R.G.L. wish to acknowledge support from NSF grants DMS-MSPA 0625409 and CCF-0830535.

7. REFERENCES

[1] L. Barrington, A.B. Chan, and G. Lanckriet. Modeling music as a dynamic texture. *IEEE TASLP*, 18(3):602–612, 2010.

[2] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE TPAMI*, 29(3):394–410, March 2007.

[3] A. B. Chan and N. Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE TPAMI*, 30(5):909–926, May 2008.

[4] A.B. Chan, E. Coviello, and G. Lanckriet. Clustering dynamic textures with the hierarchical em algorithm. In *CVPR*, 2010.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *JRSS B*.

[6] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto. Dynamic textures. *Intl. J. Computer Vision*, 51(2):91–109, 2003.

[7] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems*, 2007.

[8] A. Gelb. *Applied Optimal Estimation*. MIT Press, 1974.

[9] M. Hoffman, D. Blei, and P. Cook. Easy as CBA: A simple probabilistic model for tagging music. In *ISMIR*, 2009.

[10] M.I. Mandel and D.P.W. Ellis. Multiple-instance learning for music information retrieval. In *International Conference on MIR*, 2008.

[11] S.R. Ness, A. Theocharis, G. Tzanetakis, and L.G. Martins. Improving automatic music tag annotation using stacked generalization of probabilistic svm outputs. In *Proceedings of ACM MULT.*, 2009.

[12] J. Reed and C.H. Lee. A study on music genre classification based on universal acoustic models. *ISMIR*, 2006.

[13] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic description using the CAL500 data set. *SIGIR*, page 439446, 2007.

[14] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE TASLP*, 16(2):467–476, February 2008.

[15] N. Vasconcelos and A. Lippman. Learning mixture hierarchies. In *NIPS*, 1998.

AUTOREGRESSIVE MFCC MODELS FOR GENRE CLASSIFICATION IMPROVED BY HARMONIC-PERCUSSION SEPARATION

Halfdan Rump, Shigeki Miyabe, Emiru Tsunoo, Nobukata Ono, Shigeki Sagama
The University of Tokyo, Graduate School of Information Science and Technology
{rump,miyabe,tsunoo,onono,sagayama}@hil.t.u-tokyo.ac.jp

ABSTRACT

In this work we improve accuracy of MFCC-based genre classification by using the Harmonic-Percussion Signal Separation (HPSS) algorithm on the music signal, and then calculate the MFCCs on the separated signals. The choice of the HPSS algorithm was mainly based on the observation that the presence of harmonics causes the high MFCCs to be noisy. A multivariate autoregressive (MAR) model was trained on the improved MFCCs, and performance in the task of genre classification was evaluated. By combining features calculated on the separated signals, relative error rate reductions of 20% and 16.2% were obtained when an SVM classifier was trained on the MFCCs and MAR features respectively. Next, by analyzing the MAR features calculated on the separated signals, it was concluded that the original signal contained some information which the MAR model was capable of handling, and that the best performance was obtained when all three signals were used. Finally, by choosing the number of MFCCs from each signal type to be used in the autoregressive modelling, it was verified that the best performance was reached when the high MFCCs calculated on the harmonic signal were discarded.

1. INTRODUCTION

Music information retrieval (MIR) is a diverse research field with many different areas of interest, such as chord detection, melody extraction etc. One of the popular tasks is classifying music into genres, which not only serves to ease organization of large music databases, but also drives the general development of features for representing the various important aspects of music. The task of genre classification draws upon many different kinds of information which means that one can either use features capable of expressing the music as a whole, or use many different types of features, each describing specific aspects of the music, such as the beat, melody, timbre etc. A low level feature frequently used for modelling music is the Mel-Frequency Cepstral Coefficients (MFCC), originally proposed in [1], (see [2] for a comprehensive review). The MFCCs are often calculated on the unaltered spectrum, thus containing information of all aspects of the music. The MFCCs effectively function as a lossy compression of a short part of the music signal into a small number of coefficients. It may happen that certain characteristics of the music signal which could be useful for genre classification are blurred by the compression. A possible way to resolve this issue is to break down the music signal into several signals, each containing a specific kind of information about the signal, and then calculate the MFCCs on the new signals. An example could be to separate the instruments and then calculate the MFCCs for the signals, each containing only a single instrument. However, it is possible that such a separation will fail, thus generating unpredictable results which might actually be worse than just using the original signal for classification. In this work we have used a simple algorithm that separates the music signal into two signals, one containing harmonics and the other containing percussion. The choice of this algorithm is based on some observations about the nature of the MFCCs, discussed in section 2.

After the music signal has been separated, MFCCs can be calculated on all three signals (original signal, harmonics and percussion). A classifier can be trained directly on the MFCCs, or more elaborate models can be constructed and used for classification. In this paper we investigate if higher classification performance can be achieved by separating the music signal as described above. We train a multivariate autoregressive (MAR) model on the MFCCs from the three signal types, and use it in a classifier.

The MAR model has proven to be efficient for the task of genre classification. First of all, the MAR model integrates the short time feature frames temporally, and secondly it is capable of modelling the covariances between the MFCCs. Since the ultimate goal of genre classification algorithms is to reach an accuracy of 100%, it is most meaningful to analyse the model with the highest accuracy. Therefore the article will focus mostly on the results obtained when using the MAR model for classification. Furthermore, by comparing performance of the MAR features calculated on the different signal types, it can be inferred which aspects of the music the MAR model analyses.

2. THE MEL-FREQUENCY CEPSTRAL COEFFICIENTS

The Mel-Frequency Cepstral Coefficient (MFCC) feature extraction is a useful way of extracting timbre information. The music signal is divided into a number of short time frames. For each frame, N_m coefficients are calculated, thus yielding N_m time series to be modelled by the MAR model, described in section 3.

In the following we explain the motivation for including a separation step by considering how the MFCCs are calculated. In the Mel filter-bank analysis, the bandwidth of each filter is linear for frequencies under around 1 kHz, and thereafter grows logarithmically. Therefore each of the lower Mel coefficients is the mean of a relatively narrow frequency band. If the spectrum is characterized by narrow pitch spikes, the difference between two adjacent Mel coefficients is likely to be large. Since the MFCCs are obtained by applying the DCT transform, these differences will be described by the high MFCCs. In other words, the high MFCCs are capable of closely fitting the pitch present in the frame on which they are calculated. Pitch is usually not a very good indicator for music genre, and therefore the high MFCCs should be discarded. On the other hand, if the spectrum has a smooth envelope the high order MFCCs will not model pitch, and therefore may be usable for genre classification. Most music signals contain both harmonics (pitch spikes) and percussion (smooth spectral envelope). Since the presence of pitch is harmful to the information content of the high MFCCs, it seems feasible to separate harmonics from percussion.

Furthermore it is possible that the shape of the spectral envelope of harmonics and percussion when they have been separated is useful for genre classification, and that the information content of the lower MFCCs will be improved by separating the music signal.

3. THE MULTIVARIATE AUTOREGRESSIVE MODEL

The MAR model is similar to the normal autoregressive model, in that it predicts the next sample of a time series as a linear combination of past samples. The MAR model extends the capabilities of the normal AR, as it capable of making predictions for multiple time series and utilizes correlations between time series for prediction. The prediction of the n'th N_m time series is calculated as

$$\mathbf{x}_n = \sum_{p=1}^{P} \mathbf{A}_p \mathbf{x}_{n-I(p)} + \mathbf{u}_n \quad (1)$$

where $\mathbf{x}_n$ is a $N_m \times 1$ vector containing the predictions, and n is the frame index. P is the model order which specifies the number of time lags used for prediction. The MAR model is not constrained to using only time lags $1 \dots P$, but an arbitrary set of time lags $I = \{\tau_1 \dots \tau_P\}$ can be chosen. $\mathbf{A}_1 \dots \mathbf{A}_P$ are the $N_m \times N_m$ weight matrices for time lags $\tau_1 \dots \tau_P$. Element $[A]_p^{i,j}$ is the weight that controls how much of signal j, time-lagged τ_p samples, is used for prediction of signal i. $\mathbf{u}_n$ is the offset vector and can be omitted if each time series is subtracted by it's mean before estimating the coefficient matrices. The model parameters can be estimated by using the least mean squares approach. The P weight matrices $\mathbf{A}_1 \dots \mathbf{A}_P$ and the the offset vector $\mathbf{u}_n$ are stacked into a $PN_m^2 + N_m$ dimensional vector, and this constitutes the feature vector used for classification.

A basic assumption of the MAR model is that the time series upon which it is calculated has a stationary distribution. At first glance this assumption does not seem to go well with the nature of the percussive signal since it does not have a smooth time envelope. However, over longer periods roughly the same percussion sounds and thus MFCCs will appear again and again, which can be interpreted as stationarity. On the other hand, even though the harmonic signal has a smooth time envelope for a given note, meaning that the MFCCs will have a stationary distribution during the note, the distribution will change as the next note is struck. Since the exact same combination of harmonics, or in other words the same pitch spikes which are modelled by the high order MFCCs, is unlikely to occur more than maybe a few times, the distribution cannot be assumed stationary.

High order models are characterized by a high variance which gives them the power to fit closely to a time series, but also makes them prone to over-fitting. Low order models are more dominated by bias which makes them more suitable in cases where the signal envelope is the desired target. In [3], the MAR model was found to perform best with $P = 3$ when the task was genre classification, but the optimal value might differ according to the application for the reasons listed above.

4. HARMONIC-PERCUSSION SIGNAL SEPARATION

The Harmonic-Percussion Signal Separation (HPSS) algorithm proposed in [5], is a simple and fast method of dividing a musical signal, **N**, into two signals, **H** and **P**, each containing only the harmonic and percussive elements respectively. HPSS can be thought of as a two-cluster soft clustering, where each spectrogram grid-point is assigned a graded membership to a cluster representing harmonics and a cluster representing percussion. The algorithm uses the fact that percussion has a short temporal duration and is rich in noise, while harmonic elements have a long temporal duration with most of the signal energy concentrated in pitch spikes. Thus in the spectrogram, percussion appears as vertical lines of high power, whereas harmonic elements appear as horizontal lines.

In broad terms, the HPSS algorithm works by assuming independence between **H** and **P**, and using Bayes formula to calculate $p(\mathbf{H}, \mathbf{P}|\mathbf{N})$

$$\log p(\mathbf{H}, \mathbf{P}|\mathbf{N}) = \log p(\mathbf{N}|\mathbf{H}, \mathbf{P}) + \log p(\mathbf{H}) + \log p(\mathbf{P}) \quad (2)$$

The prior distributions $p(\mathbf{H})$ and $p(\mathbf{P})$ are defined as functions that measure the degree of smoothness in time and frequency respectively.

$$\log p(\mathbf{H}) = \sum_{\omega,\tau} \frac{-1}{2\sigma_H^2}(H^{\gamma}_{\omega,\tau-1} - H^{\gamma}_{\omega,\tau})^2 \quad (3)$$

$$\log p(\mathbf{P}) = \sum_{\omega,\tau} \frac{-1}{2\sigma_P^2}(P^{\gamma}_{\omega-1,\tau} - P^{\gamma}_{\omega,\tau})^2 \quad (4)$$

Where σ_H, σ_P and γ has been manually specified as in [5]. Thus the prior for **H** will be high when each row of the spectrogram is characterized by slow fluctuations, and similarly the prior for **P** will be high when this is the case for columns of the spectrogram. The likelihood function has been defined by measuring the I-divergence between **N** and **H** + **P**:

$$\log p(\mathbf{N}|\mathbf{H},\mathbf{P}) = \quad (5)$$
$$-\sum_{\omega,\tau}\left(N_{\omega,\tau}\log\frac{N_{\omega,\tau}}{H_{\omega,\tau}+P_{\omega,\tau}} - N_{\omega,\tau} + H_{\omega,\tau} + P_{\omega,\tau}\right)$$

and so the likelihood is maximized when $N_{\omega,\tau} = H_{\omega,\tau} + P_{\omega,\tau}$ for all ω and τ. The log-likelihood function is maximized by using the EM-algorithm. The update equations have been omitted in this work, but can be found in [5].

It is important to realize that since the HPSS algorithm is not a source separation algorithm but rather a decomposition of the original signal, no criteria of success has been defined, and so the algorithm cannot fail unless it fails to converge.

5. DATASET

We used the TZGENRE dataset proposed in [8]. The dataset has $N_s = 1000$ songs divided equally into 10 genres: blues, classic, country, disco, hip-hop, jazz, metal, pop, reggae and rock. Each song is a 30s sound snippet, and only one MAR model is calculated for the whole song. Other methods for calculating multiple MAR models on a single song and combining them afterwards can be found in [3] and [4].

6. EXPERIMENTAL SETUP

First the music signal was separated by using HPSS, and MAR features were calculated for each signal. If the MAR model is capable of using both harmonics and percussive elements at the same time, such a decomposition will not result in higher performance. However, if for instance the MAR model analyses the harmonic elements, then removing percussion will enable the MAR features to perform better. In the following, MAR features calculated on the harmonics, percussion and normal signals will be referred to as $\mathbf{m}_h$, $\mathbf{m}_p$, $\mathbf{m}_n$ respectively, whereas MFCCs will be referred to as $\mathbf{c}_h$, $\mathbf{c}_p$ and $\mathbf{c}_n$. In addition to the three single signal feature types, four combinations features of the MAR features and four combinations of the MFCCs were constructed: $\mathbf{m}_{hp}$, $\mathbf{m}_{hn}$, $\mathbf{m}_{pn}$, $\mathbf{m}_{hpn}$, $\mathbf{c}_{hp}$, $\mathbf{c}_{hn}$, $\mathbf{c}_{pn}$ and $\mathbf{c}_{hpn}$.

The sample-rate of the songs was 22.05 kHz. The MFCCs were calculated on 20 ms windows with an overlap of 10 ms. 40 filter-banks were used in the MFCC calculation. Since the number of MFCCs used to calculate the MAR features has a great influence on performance, each combination of features was evaluated with 19 different values of N_m. For each combination an $N_s \times D$ data matrix was created by stacking the N_s features vectors, each of dimension D. For features containing only MAR combinations, the dimension is $D = c(PN_m^2 + N_m)$, where $c \in \{1, 2, 3\}$ is the number of stacked MAR models.

The classifier used was a support vector machine with a Gaussian kernel. Kernel parameters σ and C were not tuned, but each column of the data matrix was normalized with respect to standard deviation. 500-fold cross validation was used for each of the 19 values of N_m, resulting in a $N_s \times 19$ matrix, where each column contained the average accuracy for each song for a given N_m. The overall performance for a given N_m was obtained by taking the mean of that column.

7. RESULTS

In this section the results of the experiments described in section 6 are presented and discussed.

7.1 Combining features from the separated signals

Figure 1 shows the classification performance of the seven combinations when the classifiers were trained directly on the MFCCs. The difference between the classifier trained on the MFCCs calculated on the original signal to the best performing feature, $\mathbf{c}_{hp}$, is 7.5%, corresponding to a relative error rate reduction of 20.0%. This is a significant improvement, and confirms that the MFCCs have problems expressing both harmonic and percussive information when present at the same time.

$\mathbf{c}_h$ reaches its near peak performance for low N_m. This means that for the harmonic signal, very little usable information is contained in the high MFCCs. The MFCCs are fairly low-dimensional which means that the SVM classifier is still able to achieve optimal performance, and thus performance only degrades slightly. Performance of $\mathbf{c}_p$ keeps increasing when including more MFCCs, meaning that the higher MFCCs in the percussion signal contains usable information. Furthermore, the performance gained by including higher MFCCs is more than for the harmonics signal but less than for the percussion signal. This confirms that the presence of harmonics degrades the information quality of the higher MFCCs.

Next, we use the MAR model for classification and test performance of $\mathbf{m}_h$, $\mathbf{m}_p$ and $\mathbf{m}_n$, and of the combinations of them. The performance of the seven combination features is shown on Figure 2. $\mathbf{m}_n$ is the most powerful of the three single model features peaking with a performance of 74.1%. Pleasingly, all three single model features have a lower performance than the combination features. $\mathbf{m}_{hnp}$ had a peak performance of 77.6%, a gain of 3.6% compared to the best single signal model.

As was also seen when using the MFCCs in the classifier, $\mathbf{m}_{hp}$ performs significantly better than $\mathbf{m}_n$. This shows that the autoregressive modelling of the MFCCs cal-

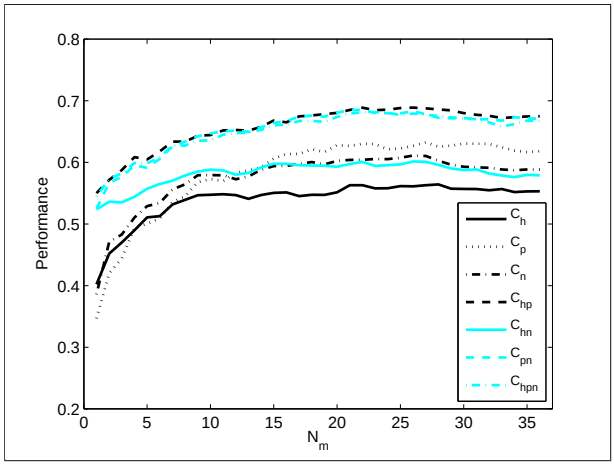


Figure 1. Performance curves for the classifier trained on MFCCs

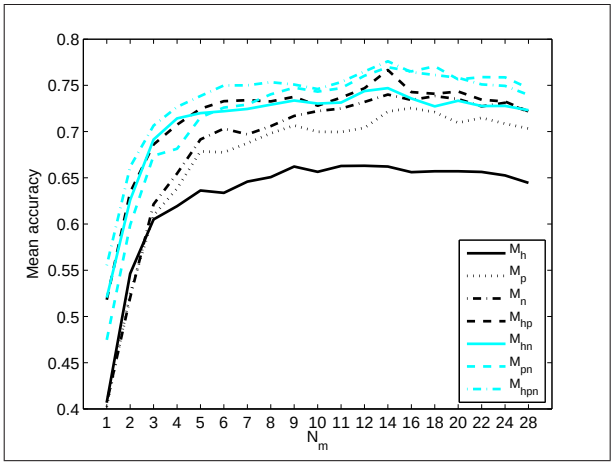


Figure 2. Performance curves for the classifier trained on MAR features

culated on the original signal cannot compensate for the MFCCs' inability to handle the mixture of harmonic and percussive information.

An important difference between using MFCCs or MAR features in the classifier is that $\mathbf{m}_{hpn}$ outperformed $\mathbf{m}_{hp}$, whereas $\mathbf{c}_{hpn}$ and $\mathbf{c}_{hp}$ had the same level of performance. Thus the MAR model is capable of modelling some properties of the original signal **N**, which are present in neither **H** nor **P**. More specifically, the MAR model can in some cases predict percussion from harmonics or vice versa, due to the autoregressive modelling. This is a reasonable claim when keeping in mind that the HPSS algorithm is not a source separation algorithm, and that some instruments will produce both harmonics and percussive sounds.

As an example, when a note is played on a piano the hammer hits the string causing it to vibrate, resulting in a sound with a high attack part and a slowly declining envelope. Since this will happen every time the piano is used, the MAR model can use the attack part to make a prediction about the rest of the sound. When using HPSS to separate the signal however, percussion is assumed to be independent from harmonics, and the attack part, which is rich in noise and has a short temporal duration, is assigned to the percussion signal while the rest of the sound is assigned to the harmonic signal. When this happens the MAR model can no longer model the dependencies, so including MAR features calculated on the original signal increases performance.

7.2 Differences between the signal type MAR features

In this section we analyse some of the differences between the MAR features calculated on each of the separated signals.

An important step towards understanding the MAR features and specify their application domain is to investigate to which degree features calculated on the different signal types classify the same songs or not. In the former case, classification accuracy with different signal types is largely genre dependent, and in the latter case there will be some

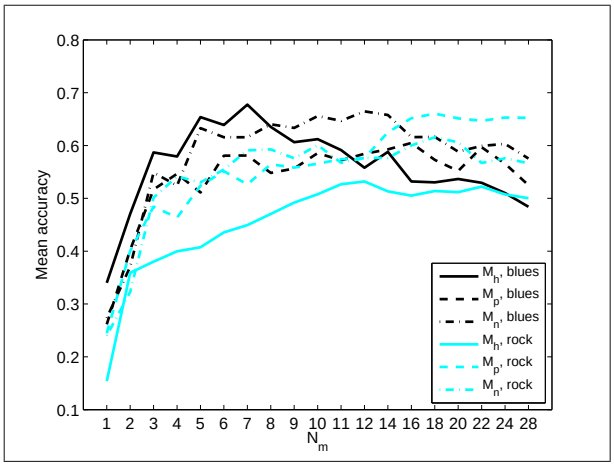


Figure 3. Examples of genre specific performance, only MAR features

easy songs which can be classified by all signal models, and some hard songs that only the features with an overall high performance can classify.

Analysis is carried out by finding the point where all signal models have approximately the same accuracy, and calculating the correlation between the $N_s \times 1$ song accuracy vectors. It was observed that there is a low correlation between which songs $\mathbf{m}_h$ and $\mathbf{m}_p$ classify. This suggests that the two signal models contains different information which allows for the classification of different songs, and thus are efficient with different kinds of music. For most genres $\mathbf{m}_n$ is slightly better than $\mathbf{m}_p$, with $\mathbf{m}_h$ being the worst performing of the three. However, for some genres $\mathbf{m}_h$ achieves the best performance when the high MFCCs were discarded, as can be seen on Figure 3. Furthermore, the fact that the correlation of the song classification vectors of $\mathbf{m}_p$ and $\mathbf{m}_n$ was high, means that they classify more of the same songs than $\mathbf{m}_h$ and $\mathbf{m}_n$, which is consistent with the fact that $\mathbf{m}_{hn}$ and $\mathbf{m}_p$ classify more of the same songs than $\mathbf{m}_{pn}$ and $\mathbf{m}_h$. These results suggest that MAR

Feature	Performance	Relative ERR
$\mathbf{c}_n$	61.1%	N/A
$\mathbf{c}_{hp}$, Constr.	68.9%	20.0%
$\mathbf{m}_n$	74.1%	N/A
$\mathbf{m}_{hpn}$, Constr.	77.6%	13.5%
$\mathbf{m}_{hpn}$, N.Constr.	78.3%	16.2%

Table 1. Overview of the best performing features. Constr. or N.Constr. refer to the constraint on N_m.

features calculated on the original music reflect the percussive elements to a higher degree than the harmonics elements.

The fact that $\mathbf{m}_{pn}$ is even higher than $\mathbf{m}_{hp}$ seems like a contradiction to the statement made earlier that $\mathbf{m}_n$ is more correlated with $\mathbf{m}_p$ than with $\mathbf{m}_h$. The explanation to this is most likely that the gains from combining uncorrelated features, i.e. $\mathbf{m}_{hp}$ and $\mathbf{m}_{hn}$, cannot match the penalty caused by the low performance of $\mathbf{m}_h$. Although $\mathbf{m}_p$ and $\mathbf{m}_n$ are somewhat correlated, there are still some differences in what songs they classify, and this seems to results in a performance gain when combined.

7.3 Selecting N_m for each signal type

Figure 2 in section 7.1 shows that the MAR features calculated on the different signal types perform best for different values of N_m. In this section we investigate if performance can be improved by removing the constraint that the number of MFCCs used to calculate the MAR model must be the same for all signal types. Since it is possible that simply combining the best performing models does not achieve the highest performance, the five best models of each signal type were used to form a number of combination features.

Figure 4 shows the performance plotted versus the dimensionality of the feature vector, using the same number of MFCCs, and with different number of MFCCs. The figure makes it easy to compare feature efficiencies, as a point that is situated higher and on the left side of another point of the same type, means that a feature of lower dimensionality had higher performance.

From Figure 4 it seems that the method of selecting N_m for each single MAR model is not particularly capable of producing low dimensional features, but the method do achieve the highest overall performance. However, since it is in general infeasible to try all combinations of N_m before selecting the best one, a general tendency must be discovered. In section 2 it was suggested that the high MFCCs calculated on the harmonics signal should be discarded, whereas high MFCCs from the percussion signal could be used. This was the case when the classifier was trained directly on the MFCCs, and when the classifier was trained on the MAR features. It is not surprising therefore, that the best performance of 78.3% was obtained by discarding the high MFCCs for the harmonic signal and using high MFCCs from the percussion signal.

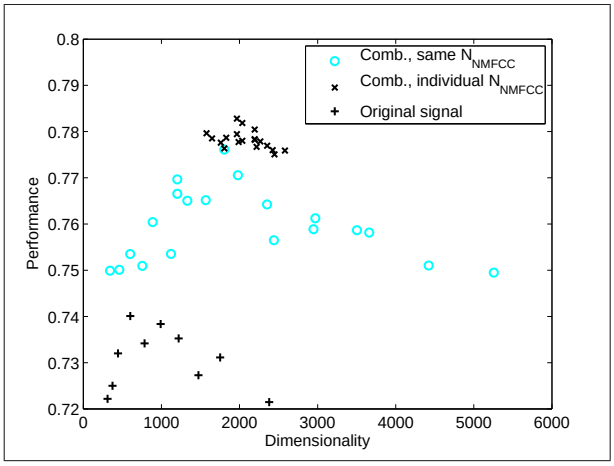


Figure 4. Performance and dimensionality of combination models

8. PERFORMANCE DEMONSTRATION

This section contains a short demonstration of the performance obtained when combining the improved features with two other features types, each describing different aspects of music. The first type is the Rhythm Map features, proposed in [6], which are calculated on the percussion signal. A song is represented as a ten dimensional vector, each element describing the membership to a rhythmic template extracted from the entire dataset. The second feature type, henceforth referred to as TZ-features, represents a song as an 68-dimensional vector containing a set of timbre related features proposed in [8]. The Rhythm Map is of special interest since it is calculated on the percussive signal provided by the HPSS algorithm, and thus provide no information about the harmonics. The TZ-features were chosen because they were tested in combination with Rhythm Map (see [7]), where it was shown that the two feature types compliment each other well. An accuracy of 75.0% was obtained on the dataset by the combination of Rhythm Map and TZ-features. When the MAR features calculated on the original signal were included as well, a performance of 80.1% was achieved. Finally, by separating the signal with HPSS and calculating MAR features on the three signals as proposed, a performance of 82.46% was obtained, corresponding to a relative error rate reduction of 12.0%.

9. CONCLUSION

In this work we proposed that separating the music signal into more signals, each containing certain characteristics of the original signal, could produce better features, leading to increased performance in the task of music genre classification. Based on the observation that the presence of harmonics causes the high MFCCs to be noisy, we used the HPSS algorithm to separate the signal into two signals, one containing harmonics and the other containing percussion. The separation increased performance significantly, both when the classifier was trained on the MFCCs and when it was trained on the MAR features. The best perfor-

mance obtained with the MAR features was 78.3%, corresponding to a relative error rate reduction of 16.2%. It was seen that the MAR model uses both harmonic and percussive information to make predictions, but that the percussive information seems to be the dominating. The fact that the best performance was reached when the MAR features from the separated signals were combined with the original signal showed us that the MAR-model could, to some extend, model dependencies between harmonic and percussive elements. The combination of MFCCs calculated on the harmonics signal and MFCCs calculated on the percussion signal performed better than MFCCs calculated on the original signal, and this was interpreted as an inability of the MFCCs to model the presence of both harmonics and percussion in the same signal. An important conclusion of this is that separating the music signal as proposed simply creates better low level features, which means that models trained on these features will also be improved.

10. REFERENCES

[1] S. B. Davis and P. Marmelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Trans. Acoustics, Speech, Signal Proces.*, Vol. 28, No. 4, pp. 357–366, 1980

[2] B. Logan, "Mel Frequency Cepstral Coefficients for Music Modeling," *Proc. ISMIR*, 2000

[3] A. Meng, P. Ahrendt, J. Larsen and L. K. Hansen, "Temporal Feature Integration for Music Genre Classification," *IEEE Trans. Audio, Speech, Lang. Proces.*, Vol. 15, No. 5, pp. 1654–1664, 2007

[4] J. S. Shawe-Taylor and A. Meng, "An Investigation of Feature Models for Music Genre Classification Using the Support Vector Classifier," *Proc. ISMIR*, pp. 604–609, 2005

[5] N. Ono, K. Miyamoto, H. Kameoka, and S. Sagayama, "A real-time equalizer of harmonic and percussive componets in music signals," *Proc. ISMIR*, pp. 139–144, 2008

[6] E. Tsunoo, N. Ono and S. Sagayama, "Rhythm map: Extraction of unit rhythmic patterns and analysis of rhythmic structure from music acoustic signals," *Proc. ICASSP*, pp. 185–188, 2009

[7] E. Tsunoo, G. Tzanetakis, N. Ono and S. Sagayama, "Audio genre classification using percussive pattern clustering combined with timbral features," *Proc. ICME*, pp. 382–385, 2009

[8] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech, Audio Processing*, Vol. 10, No. 5, pp. 293–302, 2002

BASS PLAYING STYLE DETECTION BASED ON HIGH-LEVEL FEATURES AND PATTERN SIMILARITY

Jakob Abeßer
Fraunhofer IDMT
Ilmenau, Germany
(abr@idmt.fraunhofer.de)

Paul Bräuer
Piranha Musik & IT
Berlin, Germany

Hanna Lukashevich, Gerald Schuller
Fraunhofer IDMT
Ilmenau, Germany

ABSTRACT

In this paper, we compare two approaches for automatic classification of bass playing styles, one based on high-level features and another one based on similarity measures between bass patterns. For both approaches, we compare two different strategies: classification of patterns as a whole and classification of all measures of a pattern with a subsequent accumulation of the classification results. Furthermore, we investigate the influence of potential transcription errors on the classification accuracy, which tend to occur when real audio data is analyzed. We achieve best classification accuracy values of 60.8% for the feature-based classification and 68.5% for the classification based on pattern similarity based on a taxonomy consisting of 8 different bass playing styles.

1. MOTIVATION

Melodic and harmonic structures were often studied in the field of Music Information Retrieval. In genre discrimination tasks, however, mainly timbre-related features are somewhat satisfying to the present day. The authors assume, that bass patterns and playing styles are missing complementaries. Bass provides central acoustic features of music as a social phenomenon, namely its territorial range and simultaneous bodily grasp. These qualities come in different forms, which are what defines musical genres to a large degree. Western popular music with its worldwide influence on other styles is based upon compositional principles of its classical roots, harmonically structured around the deepest note. African styles also often use tonal bass patterns as ground structure, while Asian and Latin American styles traditionally prefer percussive bass sounds. In contrast to the melody (which can easily be interpreted in "cover versions" of different styles), the bass pattern most often carries the main harmonic information as well as a central part of the rhythmic and structural information.

A more detailed stylistic characterization of the bass instrument within music recordings will inevitably improve classification results in genre and artist classification tasks. Within the field of Computational Ethnomusicology (CE) [19], the automatic detection of the playing styles of the participating instruments such as the bass constitutes a meaningful approach to unravel the fusion of different musical influences of a song. This holds true for many contemporary music genres and especially for those of a global music background.

The remainder of this paper is organized as follows. After outlining the goals and challenges in Sec. 2 and Sec. 3, we provide a brief overview over related work in Sec. 4. In Sec. 5, we introduce novel high-level features for the analysis of transcribed bass lines. Furthermore, we propose different classification strategies, which we apply and compare later in this paper. We introduce the used data set and describe the performed experiments in Sec. 6. After the results are discussed, we conclude this paper in Sec. 7.

2. GOALS

The goal of this publication is to compare different approaches for automatic playing style classification. For this purpose, we aim at comparing different classification approaches based on common statistical pattern recognition algorithms as well as on the similarity between bass patterns. In both scenarios, we want to investigate the applicability of a aggregation classification based on the subpatterns of an unknown pattern.

3. CHALLENGES

The extraction of score parameters such as note pitch and onset from real audio recordings requires reliable automatic transcription methods, which nowadays are still error-prone when it comes to analyzing multi-timbral and polyphonic audio mixtures [4, 13]. This drawback impedes a reliable extraction of high-level features that are designed to capture important rhythmic and tonal properties for a description of an instrumental track. This is one problem addressed in our experiments. Another general challenge is the translation of musical high-level terms such as syncopations, scale, or pattern periodicity into parameters that are automatically retrievable by algorithms. Information regarding micro-timing, which is by the nature of things impossible to encompass in a score [9], is left out.

4. PREVIOUS APPROACHES

Within the last years, the use of score-based high-level features became more popular for tasks such as automatic genre classification. To derive a score-based representation from real audio recordings, various automatic transcription algorithms have been proposed so far. The authors of [18], [13], and [4] presented algorithms to transcribe bass lines. Musical high-level features allow to capture different properties from musical domains such as melody, harmony, and rhythm [1, 3, 10, 11]. Bass-related audio features we used for genre classification in [18], [1], and [17].

An excellent overview over existing approaches for the analysis of expressive music performance and artist-specific playing styles is provided in [23] and [24]. In [7], different melodic and rhythmic high-level features are extracted before the performed melody is modeled with an evolutionary regression tree model. The authors of [15] also used features derived from the onset, inter-onset-interval and loudness values of note progression to quantify the performance style of piano players in terms of their timing, articulation and dynamics. To compare different performances in terms of rhythmic and dynamic similarity, the authors of [14] proposed a numerical method based on the correlation at different timescales.

5. NOVEL APPROACH

5.1 Feature extraction

In this paper, we use 23 multi-dimensional high-level features that capture various musical properties for the tonal and rhythmic description of bass lines. The feature vector consists of 136 dimensions in total. The *basic note parameters*, which we investigate in this paper, are the absolute pitch $\boldsymbol{\Theta_P}$, the loudness $\boldsymbol{\Theta_V}$, the onset $\boldsymbol{\Theta_O^{[s]}}$and $\boldsymbol{\Theta_O^{[M]}}$, and the duration $\boldsymbol{\Theta_D^{[s]}}$and $\boldsymbol{\Theta_D^{[M]}}$of each note. The indices [s] and [M] indicate that both the onset and the duration of a note can be measured in seconds as well as in lengths of measures. All these parameters are extracted from symbolic MIDI files by using the MIDI-Toolbox for MATLAB [5].

Afterwards, further *advanced note parameters* are derived before features are extracted. From the pitch differences $\boldsymbol{\Delta\Theta_P}$ between adjacent notes in semitones, we obtain vectors containing the interval directions $\boldsymbol{\Delta\Theta_P^{(D)}}$ (being either ascending, constant, or descending), and the pitch differences in terms of functional interval types $\boldsymbol{\Delta\Theta_P^{(F)}}$. To derive the functional type of an interval, we map its size to a maximum absolute value of 12 semitones or one octave by using the modulo 12 operation in case it is larger than one octave upwards or downwards (12 semitones). Then each interval is assigned to a function interval type (prime, second, third etc.) according to well known music principles. In addition to the high-level features presented in [1], we use various additional features related to tonality and rhythm in this paper, which are explained in the following subsections.

Features related to tonality

We derive features to measure if a certain *scale* is applied in a bass pattern. Therefore, we take different binary scale templates for natural minor (which includes the major scale), harmonic minor, melodic minor, pentatonic minor (subset of natural minor which also includes the pentatonic major scale), blues minor, whole tone, whole tone half tone, arabian, minor gypsy and hungarian gypsy [21] into account. Each scale template consists of 12 values representing all semitones of an octave. The value 1 is set for all semitones that are part of the scale, the value 0 for those that are not. All notes within a given pattern, which are related to a certain scale, are accumulated by adding their normalized note loudness values $\Theta_V/\Theta_{V,max}$ with $\Theta_{V,max}$ being the maximum note loudness in a pattern. The same is done for all notes, which are not contained in the scale. The ratio of both sums is calculated over all investigated scales and over all 12 possible cyclic shifts of the scale template. This cyclic shift is performed to cope with each possible root note position. The maximum ratio value over all shifts is determined for each scale template and used as a feature value, which measures the presence of each considered scale. We obtain the relative frequencies p_i of all possible values in the vector that contains the interval directions ($\boldsymbol{\Delta\Theta_P^{(D)}}$) as well as the vector that contains the functional interval types ($\boldsymbol{\Delta\Theta_P^{(F)}}$) and use them as features to characterize the variety of different pitch transitions between adjacent notes.

Features related to rhythm

Syncopation embodies an important stylistic means in different music genres. It represents the accentuation on weak beats of a measure instead of an accentuation on a neighbored strong beat that usually would be emphasized. To detect syncopated note sequences within a bass-line, we investigate different temporal grids in terms of equidistant partitioning of single measures. For instance, for an eight-note grid, we map all notes inside a measure towards one of eight segments according to their onset position inside the measure. In a $\frac{4}{4}$ time signature, these segments correspond to all 4 quarter notes (on-beats) and their off-beats in between. If at least one note is mapped to a segment, it is associated with the value 1, otherwise with 0. For each grid, we count the presence of the following segment sequences - (1001), (0110), (0001), or (0111). These sequences correspond to sequences of alternating on-beat and off-beat accentuations that are labeled as syncopations. The ratios between the number of syncopation sequences and the number of segments are applied as features for the rhythmical grids 4, 8, 16, and 32.

We calculate the ratio $\Theta_D^{(M)}(k)/\Delta\Theta_O^{(M)}(k)$ between the duration value of the k-th note in measure lengths and the inter-onset-interval between the k-th note and its succeeding note. Then we derive the mean and the variance of this value over all notes as features. A high or low mean value indicates whether notes are played *legato* or *staccato*. The variance over all ratios captures the variation between these two types of *rhythmic articulation* within a

given bass pattern. To measure if notes are mostly played on *on-beats* or *off-beats*, we investigate the distribution of notes towards the segments in the rhythmical grids as explained above for the syncopation feature. For example, the segments 1, 3, 5, and 7 are associated to on-beat positions for an eight-note grid and a $\frac{4}{4}$ time signature. Again, this ratio is calculated over all notes and mean and variance are taken as feature values. As additional rhythmic properties, we derive the frequencies of occurrence of all commonly used note lengths from half notes to 64th notes, each in its normal, dotted, and triplet version. In addition, the relative frequencies from all note-note, note-break and break-note sequences over the complete pattern are taken as features.

5.2 Classification based on statistical pattern recognition

We investigate the applicability of the well-established Support Vector Machines (SVM) using the Radial Basis Function (RBF) as kernel combined with a preceding feature selection using the Inertia Ratio Maximization using Feature Space Projection (IRMFSP) as a baseline experiment. The feature selection is applied to choose the most discriminative features and thus to reduce the dimensionality of the feature space prior to the classification. Therefore, we calculate the high-level features introduced in 5.1 for each bass pattern, which results in an 136 dimensional feature space. Details on both the SVM and the IRMFSP can be found for instance in [1].

5.3 Classification based on pattern similarity

In this paper, we apply 2 different kinds of pattern similarity measures, *pairwise similarity measures* and *similarity measures based on the Levenshtein distance*. To compute similarity values between patterns, the values of the onset vector $\mathbf{\Theta}_{\boldsymbol{O}}^{[M]}$ and the absolute pitch vector $\mathbf{\Theta}_{\boldsymbol{P}}$ are simply converted into character strings. In the latter case, we initially subtract the minimum value of $\mathbf{\Theta}_{\boldsymbol{P}}$ for each pattern separately to remain independent from pitch transpositions. This approach can of course be affected by potential outliers, which do not belong to the pattern.

5.3.1 Similarity measures based on the Levenshtein distance

The Levenshtein distance D_L offers a metric for the computation of the similarity of strings [6]. It measures the minimum number of edits in terms of insertions, deletions, and substitutions, which are necessary, to convert one string into the other. We use the Wagner-Fischer algorithm [20] to compute D_L and derive a similarity measure S_L between two strings of length l_1 and l_2 from

$$S_L = 1 - D_L / D_{L,max} \; . \tag{1}$$

The lengths l_1 and l_2 correspond to the number of notes in both patterns. $D_{L,max}$ equals the maximum value of l_1 and l_2. In the experiments, we use the rhythmic similarity measure $S_{L,R}$ and the tonal similarity measure $S_{L,T}$ derived from the Levenshtein distance between the onset $\mathbf{\Theta}_{\boldsymbol{O}}^{[M]}$ and the pitch $\mathbf{\Theta}_{\boldsymbol{P}}$ as explained in the previous section. Furthermore, we investigate

$$S_{L,RT,Max} = \begin{cases} S_{L,R} & , S_{L,R} \geq S_{L,T} \\ S_{L,T} & , S_{L,T} > S_{L,R} \end{cases} \tag{2}$$

and

$$S_{L,RT,Mean} = \frac{1}{2}(S_{L,R} + S_{L,T}) \tag{3}$$

by using the maximum and the arithmetic mean between of $S_{L,R}$ and $S_{L,T}$ as aggregated similarity measures.

5.3.2 Pairwise similarity measures

In general, we derive a pairwise similarity measure

$$S_P = \frac{1}{2}\left(\frac{N_{n,m}}{N_n} + \frac{N_{m,n}}{N_m}\right) \tag{4}$$

$N_{n,m}$ denotes the number of notes in pattern n, for which at least one note in pattern m exists that have the same absolute pitch value (for the similarity measure $S_{P,T}$) or onset value (for the similarity measure $S_{P,R}$). $N_{m,n}$ is defined vice versa. By applying the constraint that both onset and absolute pitch need to be equal in Eq. 4, we obtain the measure $S_{P,RT}$. Furthermore, we derive the aggregated similarity measures $S_{P,RT,Max}$ and $S_{P,RT,Mean}$ analogous to Eq. 2 and Eq. 3.

6. EVALUATION

6.1 Data-set

We assembled a novel dataset from instructional bass literature [12, 21], which consists of bass patterns from the 8 genres *Swing* (SWI), *Funk* (FUN), *Blues* (BLU), *Reggae* (REG), *Salsa & Mambo* (SAL), *Rock* (ROC), *Soul & Motown* (SOU) and *Africa* (AFR), a rather general term which here signifies Sub-Saharan Popular Music Styles [16]. For each genre, 40 bass-lines of 4 measure length have been stored as symbolic audio data as MIDI files. Initial listening tests revealed that in this data set, which was assembled and categorized by professional bass players, a certain amount of stylistic overlap and misclassification between genres as for instance Blues and Swing or Soul & Motown and Funk occurs. The overlap is partly inherent to the approach of the data sets, which treat all examples of a style (e.g. Rock) as homogenous although the sets include typical patterns of several decades. In some features, early Rock patterns might resemble early Blues patterns more than they resemble late patterns of their own style [22]. Thus, the data set will be extended further and revised by educated musicologists for future experiments.

6.2 Experiments & Results

6.2.1 Experiment 1 - Feature-based classification

As described in Sec. 5.2, we performed a baseline experiment that consists of IRMFSP for chosing the best $N = 80$ features and the SVM as classifier. The parameter N has

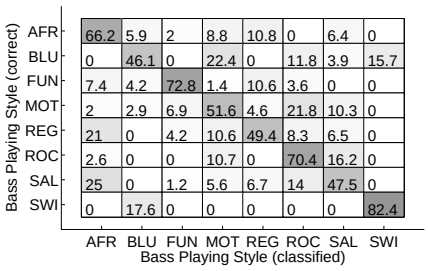


Figure 1. Exp. 1 - Confusion matrix for the feature-based pattern-wise classification (all values given in %). Mean classification accuracy is 60.8% with a standard deviation of 2.4%.

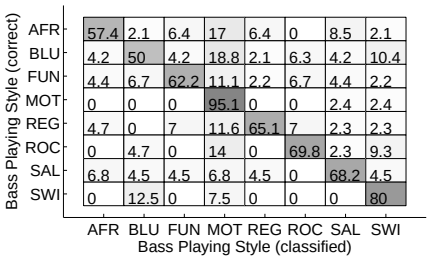


Figure 2. Exp. 2 - Confusion matrix for the best similarity-based configuration (measure-wise classification using the $S_{P,RT,Max}$ similarity measure - all values given in %). Mean classification accuracy is 68.5% with a standard deviation of 3.1%.

been determined to perform best in previous tests on the data-set. A 20-fold cross validation was applied to determine the mean and standard deviation of the classification accuracy. For a feature extraction and classification based on complete patterns, we achieved 60.8% of accuracy with a standard deviation of 2.4%. The corresponding confusion matrix is shown in Fig. 1. It can be seen, that best classification results were achieved for the styles Funk, Rock, and Swing. Strong confusions between Blues and Motown respectively Swing, Motown and Rock, Reggae and Africa as well as between Salsa and Africa can be identified. These confusions support the musicological assessment of the data-set given in Sec. 6.1. In addition, they coincide with historical relations between the styles in Africa, the Caribbean, and Latin America, as well as relations within North America as it is common musicological knowledge [8].

As a second classification strategy, we performed the feature extraction and classification based on sub-patterns. Therefore, we divided each pattern within the test set into $N = 4$ sub-patterns of one measure length. It was ensured, that no sub-patterns of patterns in the test set were used as training data. After all sub-patterns were classified, the estimated playing style for the corresponding test set pattern was derived from a majority decision over all sub-pattern classifications. In case of multiple winning classes, a random decision was applied between the winning classes. For the accumulated measure-wise classification, we achieved only 56.4% of accuracy. Thus, this approach did not improve the classification accuracy. We assume that the majority of the applied high-level features that are based on different statistical descriptors (see Sec. 5.1 for details), can not provide a appropriate characterization of the sub-patterns, which themselves only consist of 6 to 9 notes in average.

6.2.2 Experiment 2 - Pattern Similarity

This experiment is based on a leave-one-out cross-validation scheme and thus consists of $N = 320$ evaluation steps according to the 320 patterns in the data-set. Within each evaluation step, the current pattern $\mathcal{P}_k$ is used as test data while all remaining patterns $\mathcal{P}_l$ with $l \neq k$ are used as training data. We derive the class estimate $\hat{c}_k$ of $\mathcal{P}_k$ from the class label $\hat{c}$ of the best-fitting pattern $\widehat{\mathcal{P}}$ as

$$\hat{c}_k = c_{\hat{l}} \Leftrightarrow \hat{l} = \arg\max_l S_{k,l} \tag{5}$$

with $S_{k,m}$ representing the similarity measure between $\mathcal{P}_k$ and $\mathcal{P}_m$ in the given case. As in Sec. 6.2.1, if multiple patterns have the same (highest) similarity, we perform a random decision among these candidates. This experiment is performed for all similarity measures introduced in Sec. 6.2.2.

Exp. 2a: Pattern-wise classification. The basic approach for a pattern-based classification is to use each pattern of 4 measures length as one item to be classified.

Exp. 2b: Accumulated measure-wise classification. Bass patterns are often structured in a way, that the measure or a part of the measure, which precedes the pattern repetition, is often altered rhythmically or tonally and thus often varies greatly from the pattern. These figures separating or introducing pattern repetition are commonly referred to as *pickups* or *upbeats*, meaning that they do not vary or overlap the following pattern repetition which starts on the first beat of the new measure. A pattern-wise classification as described above thus might overemphasize the difference between the last measure because the patterns are compared over their complete length. Hence, we investigate another decision aggregation strategy in this experiment.

As described in Sec. 6.2.1, we divide each bass pattern into sub-patterns of one measure length each. Within each fold k, we classify each sub-pattern $\mathcal{SP}_{k,l}$ of the current test pattern $\mathcal{P}_k$ separately. At the same time, we ensure that only sub-patterns of the other patterns $\mathcal{P}_i$ with $i \neq k$ are used as training set for the current fold. To accumulate the classification results in each fold, we add all similarity values $\boldsymbol{S_{k,l}}$ between each sub-pattern $\mathcal{SP}_{k,l}$ towards their assigned winning pattern(s) $\boldsymbol{\mathcal{P}_{k,l,win}}$. The summation is done for each of the 6 genres separately. The genre that achieve the highest sum is considered as the winning genre.

As depicted in Fig. 3, the proposed accumulated measure-wise classification strategy led to higher classification accuracy values (blue bars) in comparison to a pattern-wise classification (red bars). This approach can be generalized and adopted to patterns of arbitrary length.

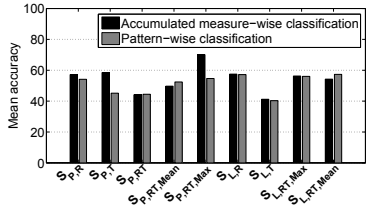


Figure 3. Mean classification accuracy results for experiment 2

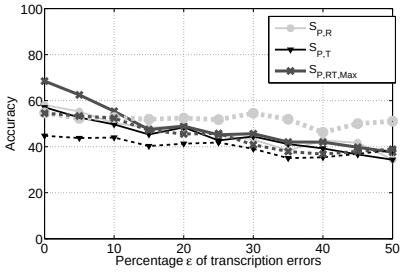


Figure 4. Exp. 3 - Mean classification accuracy vs. percentage ε of pattern variation (dotted line - pattern-wise similarity, solid line - accumulated measure-wise similarity).

The similarity measure $S_{P,RT,Max}$ clearly outperforms the other similarity measures by over 10 percent points of accuracy. The corresponding confusion matrix is shown in Fig. 2. We therefore assume that it is beneficial to use similarity information both based on pitch and onset similarity of bass patterns. For the pattern-wise classification, it can be seen that similarity measures based on tonal similarity generally achieve lower accuracy results in comparison to measures based on the rhythmic similarity. This might be explained by the frequently occurring tonal variation of patterns according to the given harmonic context such as a certain chord of a changed key in different parts of a song. The most remarkable result in confusion matrix is the very high accuracy of 95.1% for the Motown genre.

6.2.3 Experiment 3 - Influence of pattern variations

For the extraction of bass-patterns from audio recordings, two potential sources of error exist. In most music genres, the dominant bass patterns are object of small variations throughout a music piece. An automatic system might recognize the basic pattern or a variation of the basic pattern. Furthermore, automatic music transcription systems are prone to errors in terms of incorrect pitch, onset, and duration values of the notes. Both phenomena directly have a negative effect on the computed high-level features. We therefore investigate the achievable classification accuracy dependent on the percentage of notes with erroneous note parameters.

We simulate the mentioned scenarios by manipulating a random selection of ε percents of all notes from each unknown pattern and vary ε from 0% to 50%. The manipulation of a single note consists of either a modification of the onset $\mathbf{\Theta}_{\boldsymbol{O}}^{[M]}$ by a randomly chosen difference $-0.25 \leq \Delta\Theta_O^{(M)} \leq 0.25$ (which corresponds to a maximum shift distance of one beat for a $\frac{4}{4}$ time signature), a modification of the absolute pitch $\mathbf{\Theta}_{\boldsymbol{P}}$ by a randomly chosen difference $-2 \leq \Delta\Theta_P \leq 2$ (which corresponds to a maximum distance of 2 semitones), or a simple deletion of the current note from the pattern. Octave pitch errors that often appear in automatic transcription algorithms were not considered because of the mapping of each interval to a maximum size of one octave as described in Sec. 5.1. Insertions in terms of additional notes, which are not part of the pattern will be taken into account in future experiments.

As depicted in Fig. 4, the accuracy curve of the three different pair-wise similarity measures $S_{P,R}$, $S_{P,T}$ and $S_{P,RT,Max}$ falls until about 40% for a transcription error rate of 50% Interestingly, the pattern-wise classification based on $S_{P,R}$ seems to be more robust to transcription errors above 15% in comparison to the accumulated measure-wise classification even though it has a lower accuracy rate for the assumption of a perfect transcription.

6.2.4 Comparison to the related work

The comparison of the achieved results to the related work is not directly feasible. On one side, it is caused by the fact, that different data sets have been utilized. Tsunoo et al. [18] reported an accuracy of 44.8% for the GZTAN data set [1] while using only bass-line features. On the other side, the performance of only bass-line features was not every time stated. The work of Tsuchihashi et al. [17] showed an improvement of classification accuracy from 53.6% to 62.7% while applying bass-line features complimentary to other timbre and rhythmical features, but the results of genre classification with only bass features were not reported.

7. CONCLUSIONS & OUTLOOK

In this paper, different approaches for the automatic detection of playing styles from score parameters were compared. These parameters can be extracted from symbolic audio data (e.g. MIDI) or from real audio data by means of automatic transcription. For the feature-based appraoch, a best result of 60.8% of accuracy was achieved using a combination of feature selection (IRMFSP) and classifier (SVM) and a pattern-wise classification. Regarding the classification based on pattern similarity, we achieved 68.5% of accuracy using the combined similarity measure $S_{P,RT,Max}$ and a measure-wise aggregation strategy based on the classification of sub-patterns. The random baseline is 12.5%. This approach outperformed the common approach to classify the complete pattern as once.

For analyzing real-world audio recordings, further musical aspects such as micro-timing, tempo range, applied plucking & expression styles [2], as well as the interac-

[1] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. IEEE Transaction on Speech and Audio Processing, 10(5):293-302, 2002.

tion with other participating instruments need to be incorporated into a all-embracing style description of a specific instrument in a music recording. The results of experiment 4 emphasize the need for a well-performing transcription system for a high-level classification task such as playing style detection.

8. ACKNOWLEDGEMENTS

This work has been partly supported by the German research project *GlobalMusic2One*[2] funded by the Federal Ministry of Education and Research (BMBF-FKZ: 01/S08039B). Additionally, the Thuringian Ministry of Economy, Employment and Technology supported this research by granting funds of the European Fund for Regional Development to the project *Songs2See*[3], enabling transnational cooperation between Thuringian companies and their partners from other European regions.

9. REFERENCES

[1] J. Abeßer, H. Lukashevich, C. Dittmar, and G. Schuller. Genre classification using bass-related high-level features and playing styles. In *Proc. of the Int. Society of Music Information Retrieval (ISMIR Conference), Kobe, Japan*, 2009.

[2] J. Abeßer, H. Lukashevich, and G. Schuller. Feature-based extraction of plucking and expression styles of the electric bass guitar. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

[3] P. J. Ponce de Léon and J. M. Iñesta. Pattern recognition approach for music style identification using shallow statistical descriptors. *IEEE Transactions on System, Man and Cybernetics - Part C : Applications and Reviews*, 37(2):248–257, March 2007.

[4] C. Dittmar, K. Dressler, and K. Rosenbauer. A toolbox for automatic transcription of polyphonic music. In *Proc. of the Audio Mostly*, 2007.

[5] Tuomas Eerola and Petri Toiviainen. *MIDI Toolbox: MATLAB Tools for Music Research*. University of Jyväskylä, Jyväskylä, Finland, 2004.

[6] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, Cambridge, UK, 1997.

[7] A. Hazan, M. Grachten, and R. Ramirez. Evolving performance models by performance similarity: Beyond note-to-note transformations. In *Proc. of the Int. Symposium*, 2006.

[8] Ellen Koskoff, editor. *The Garland Encyclopedia of World Music - The United States and Canada*. Garland Publishing, New York, 2001.

[9] Gerhard Kubik. *Zum Verstehen afrikanischer Musik*. Lit Verlag, Wien, 2004.

[10] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *Proc. of the Int. Symposium of Music Information Retrieval (ISMIR)*, 2004.

[11] C. McKay and I. Fujinaga. jSymbolic: A feature extractor for MIDI files. In *Int. Computer Music Conference (ICMC)*, pages 302–305, 2006.

[12] H.-J. Reznicek. *I'm Walking - Jazz Bass*. AMA, 2001.

[13] M. P. Ryynänen and A. P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32:72–86, 2008.

[14] C. S. Sapp. Hybrid numeric/rank similarity metrics for musical performance analysis. In *Proc. of the Int. Symposium on Music Information Retrieval (ISMIR)*, pages 501–506, 2008.

[15] E. Stamatatos and G. Widmer. Automatic identification of music performers with learning ensembles. *Artificial Intelligence*, 165:37–56, 2005.

[16] Ruth M. Stone, editor. *The Garland Encyclopedia of World Music - Africa*, volume 1. Garland Publishing, New York, 1998.

[17] Y. Tsuchihashi, T. Kitahara, and H. Katayose. Using bass-line features for content-based MIR. In *Proc. of the Int. Conference on Music Information Retrieval (ISMIR), Philadelphia, USA*, pages 620–625, 2008.

[18] E. Tsunoo, N. Ono, and S. Sagayama. Musical bass-line clustering and its application to audio genre classification. In *Proc. of the Int. Society of Music Information Retrieval (ISMIR Conference), Kobe, Japan*, 2009.

[19] George Tzanetakis, Ajay Kapur, W. Andrew Schloss, and Matthew Wright. Computational ethnomusicology. *Journal of Interdisciplinary Music Studies*, 1(2):1–24, 2007.

[20] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.

[21] Paul Westwood. *Bass Bible*. AMA, 1997.

[22] Peter Wicke. *Handbuch der populären Musik: Geschichte, Stile, Praxis, Industrie*. Schott, Mainz, 2007.

[23] G. Widmer, S. Dixon, W. Goebl, E. Pampalk, and A. Tobudic. In search of the Horowitz factor. *AI Magazine*, 24:111–130, 2003.

[24] G. Widmer and W. Goebl. Computational models of expressive music performance: The state of the art. *Journal of New Music Research*, 33(3):203–216, 2004.

[2] see http://www.globalmusic2one.net

[3] see http://www.idmt.de/eng/research_topics/songs2see.html

BEAT CRITIC: BEAT TRACKING OCTAVE ERROR IDENTIFICATION BY METRICAL PROFILE ANALYSIS

Leigh M. Smith
IRCAM
leigh.smith@ircam.fr

ABSTRACT

Computational models of beat tracking of musical audio have been well explored, however, such systems often make "octave errors", identifying the beat period at double or half the beat rate than that actually recorded in the music. A method is described to detect if octave errors have occurred in beat tracking. Following an initial beat tracking estimation, a feature vector of metrical profile separated by spectral subbands is computed. A measure of subbeat quaver (1/8th note) alternation is used to compare half time and double time measures against the initial beat track estimation and indicate a likely octave error. This error estimate can then be used to re-estimate the beat rate. The performance of the approach is evaluated against the RWC database, showing successful identification of octave errors for an existing beat tracker. Using the octave error detector together with the existing beat tracking model improved beat tracking by reducing octave errors to 43% of the previous error rate.

1. STRUCTURAL LEVELS IN BEAT PERCEPTION

The psychological and computational representation of listeners experience of musical time is of great application to music information retrieval. Correctly identifying the beat rate (*tactus*) facilitates further understanding of the importance of other elements in musical signals, such as the relative importance of tonal features.

Considerable research has proposed theories of an hierarchical structuring of musical time [12–14, 18, 20, 27], with the favouring of particular temporal levels. The tactus has been shown to be influenced by temporal preference levels [10], proposed as a resonance or inertia to variation [25]. At the metrical level [1], [21] argue that preestablished mental frameworks ("schemas") for musical meter are used during listening. They found a significant difference in performance between musicians and non-musicians, arguing that musicians hold more resilient representations of meter, which favours hierarchical subdivision of the measure, than the non-musicians.

The fastest pulse has been used in ethnomusicology [16, 24] or reciprocally, the *tatum* in cognitive musicology [1] as a descriptive mechanism for characterising rhythmic structure. While it is not assumed to be a model of perception used by listeners and performers [16], the tatum is used to form a rhythmic grid of equally spaced intervals. It therefore represents the limit of hierarchical temporal organisation in complex rhythmic structures.

2. ERRORS IN BEAT TRACKING

Beat tracking or foot-tapping has a long history [7, 19], spurred on by the demands of music information retrieval [8, 15, 22, 23]. Common methods of beat tracking involve extraction of a mid-level representation, or onset detection function [23], typically derived from the spectral flux, thereby avoiding the requirement of identifying each individual onset. A number of methods have been proposed to then determine a time varying frequency analysis of the onset detection function, including comb filterbanks [6, 15, 23], autocorrelation [2, 9], dynamic time warping [8], Bayesian estimation [3], combined frequency and time lag analysis [22], coupled oscillators [17] and wavelet analysis [4].

Despite reporting very good results, there are areas for improvement to these approaches. A common task faced by many of these approaches is selecting the appropriate structural level from several viable candidates. It is a common occurance to select a beat rate which is twice as fast as the actual performed rate, termed an "octave error". For many of these systems, a reselection of the correct structural level from the candidates would be possible if the octave error could be detected.

The concept of fastest pulse can be used as an indicator of the highest structural level and therefore a datum. This appears in terms of the fastest alternation of events. Checking for quaver (1/8 note) alternation indicates if there is evidence of the fastest pulse appearing at the expected structural level, given the assumed tactus level. This paper proposes a method to evaluate the beat tracking and identify octave errors using an analysis of metrical profiles. This forms a combined feature vector of metrical profile over separate spectral subbands, described in Section 3. The behaviour of the metrical profile is analysed in

[1] A periodic repetition of perceived accentuation, notated in music as $\frac{4}{4}, \frac{3}{4}$ etc.

terms of quaver alternation to identify beat tracking which has performed an octave error. This approach is evaluated against an annotated dataset for beat tracking and tempo estimation as described in Section 4. The results of evaluation against datasets of recorded music are reported in Section 5.

3. METHOD

To identify the fastest pulse or tatum requires identifying the higher level rhythmic structural levels. To do so, the beat period (tactus) and metrical period (duration of the bar) is computed from the audio signal of the musical example using a beat-tracker, in this case as developed by Peeters [22]. From the nominated beat times, a metrical profile is computed.

3.1 Metrical Profile

The metrical profile, indicating the relative occurrence of events in each metrical position within the measure, has been demonstrated by [21] to represent metrical structure and matches closely with listeners judgements of metrical well-formedness. The metrical profile is computed from the likelihood of an onset at each tatum (shortest temporal interval) within a measure. The likelihood of onsets are determined from the presence of onset detection function (ODF) energy e described in [22]. The probability of an onset o_t at each tatum location t is

$$o_t = \begin{cases} \frac{\bar{e_t}}{\bar{e}+\gamma\sigma_e+\epsilon}, & o_t < 1 \\ 1 & o_t > 1 \end{cases} \quad (1)$$

where $\bar{e_t}$ is the mean energy of the ODF over the region of the tatum t, $\bar{e}$ and σ_e are the mean and standard deviation of the entire ODF energy respectively, ϵ is a small value to guard against zero $\bar{e}$, and γ is a free parameter determining the maximum number of standard deviations above the mean to assure an onset has occurred. By informal testing, $\gamma = 2$. The onset likelihoods are then used to create an histogram m_t, for $t = 1, \ldots, n$, of the relative amplitude and occurrence at each tatum, by averaging each o_t across all M measures

$$m_t = \frac{\sum_{\mu=0}^{M} o_{t+n\mu}}{M}. \quad (2)$$

To normalise for varying tempo across each piece and between pieces, the duration of each measure is derived from the beat-tracker [22]. Using the beat locations identified by the beat-tracker, each beat duration is uniformly subdivided into 1/64th notes (hemi-demi-semiquavers), that is $0 < t < 64$ for a measure of a semibreve (whole note) duration. Such a high subdivision attempts to categorise swing timing occurring within the measure and to provide sufficient resolution for accurate comparisons of metrical structure. Using the tatum duration set to equal subdivisions of each beat duration does not capture expressive timing occuring within that time period. However, the error produced from this is minimal since the expressive timing which modifies each beat and measure period is respected.

Channel c	Low band ω_c (Hz)	High band ω'_c (Hz)
1	60	106
2	106	186
3	186	327
4	327	575
5	575	1012
6	1012	1781
7	1781	3133
8	3133	5512

Table 1. Sub-band channel frequency ranges used to calculate local spectrum onset detection functions in Equation 3.

The effect of this error is to blur the peak of each tatum onset. The metrical profile is then downsampled (by local averaging of 4 tatums) to semiquavers (1/16 notes).

3.2 Spectral Sub-band Profiles

Listeners categorise sounds using their individual spectral character, and the identification of their reoccurance aids rhythmic organisation. To distinguish the possibly competing timing of different instruments and in order to match categorization used by listeners, metrical profiles are separated by spectral energy. This is produced by computing spectral sub-bands of the half wave rectified spectral energy. The sub-bands are computed by summing over non-overlapping frequencies:

$$F_{c,t} = \sum_{b=b_c}^{b'_c} e_{HWR}(\omega_b, t), \quad (3)$$

where $F_{c,t}$ is the spectral flux for the sub-band channel $c = 1, \ldots, C$ at time t, over the spectral bands $b = [\omega_c, \omega'_c]$ of the half-wave rectified spectral energy $e_{HWR}(\omega_b, t)$ at frequency band ω_b computed as described by [22]. The sub-band channels used are listed in Table 1 for $C = 8$. These form logarithmically spaced spectral bands that approximate different time keeping functions in many forms of music. A set of subband metrical profiles is then m_{tc} for $t = 1, 2, \ldots, n, c = 1, \ldots, C$.

3.3 Quaver Alternation

With the metrical profile reduced to semiquavers, a measure of the regularity of variation at the supposed quaver period can be calculated. Since the tatums at strong metrical locations are expected to vary strongly regardless of metrical level, only the variation for the *sub-beats* falling at metrically weaker locations is used. For example, in a $\frac{4}{4}$ measure, $n = 16$, metrically strong semiquavers are $r = \{1, 5, 9, 13\}$. The subbeat vector of length S is defined as $s = r \not| t$. Using the same example meter, $s = \{2, 3, 4, 6, 7, 8, 10, 11, 12, 14, 15, 16\}$.

The average quaver alternation q for a rhythm is the normalised first order difference of subbeat profiles m'_s

$$q = \frac{\sum_{c=1}^{C} \sum_{i \in s} |m'_{ic}|}{SC \max(m_s)}. \quad (4)$$

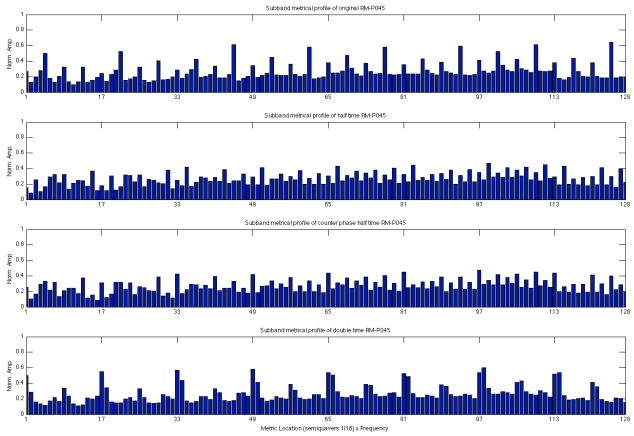


Figure 1. Metrical profiles of an example from the RWC dataset which was beat tracked with octave error. The top plot displays a metrical profile of 16 semiquavers per measure for each of the spectral subands ($c = 1, \ldots, 8$). The second, third and fourth plots displays the subband metrical profiles created for half time, half time counterphase and double time interpretations respectively.

A low quaver alternation measure indicates that variation between adjacent sub-beat semiquavers is low. This is most likely either in the case that there is little activity in the music, or the structural level chosen as the quaver is incorrect, i.e an octave error has occurred. To identify the case of an octave error, the quaver alternation of the metrical profile of a track is compared to metrical profiles of the same track formed from half and double the number of beats. The half tempo profile $\dot{q}$ is formed from simply skipping every second beat identified by the beat tracker. A similar counter-phase half tempo profile $\acute{q}$ is formed by also skipping the initial beat. The double time profile $\ddot{q}$ is formed from sampling at onsets o_t linearly bisecting each original inter-beat interval.

Comparisons between metrical profiles of an example rhythm is shown in Figure 1. The metrical pattern is displayed on the top plot, with $n = 16$ tatums per measure, the $C = 8$ subband profiles arranged adjacent in increasing frequency band. On the lower plots, the patterns created by assuming half tempo, half tempo counterphase, and double tempo are displayed. It can be seen that the alternation which occurs on the half tempo and half tempo counterphase plots is more regular than the original metrical pattern or the double time pattern. This indicates that for this example, an octave error has occurred.

A measure of octave error e is computed by comparing the ratio of the half tempo quaver alternation to original quaver alternation and the ratio of double tempo to original quaver alternation,

$$e = \frac{\dot{q} + \acute{q}}{2q} + \frac{\ddot{q}}{q}. \tag{5}$$

Equation 5 represents the degree that the alternation at the half or double tempo exceeds the original quaver alternation. Values of $\frac{\dot{q}+\acute{q}}{2q} > 1$ or $\frac{\ddot{q}}{q} > 1$ indicates there is an octave error from either the double or half quaver alternation being greater, but in practice, the threshold $e > e'$ needs to be higher. The threshold was determined experimentally as half a standard deviation above $\bar{e}$ as derived from the RWC dataset at $e' = 3.34$.

3.4 Reestimation of Tempo

The beat tracking for each piece which was nominated by the algorithm as being an octave error is then recomputed with the prior tempo estimate set to half the tempo first computed. In the case of the Viterbi decoding of the beat tracker used [22], this prior tempo estimate weights the likely path of meter and tempo selection towards the half rate. In this case, even if the prior tempo is set at half, it is not guaranteed to be chosen as half the rate, if the original tempo is a more likely path which outweighs the new reestimation. This makes the beat tracker robust to false positive classifications from the beat critic.

4. EVALUATION

Two evaluation strategies for octave errors are possible: 1) evaluation of beat tracking, where the phase of the beat tracking is correct, but the beat frequency is twice the true rate and 2) evaluation of tempo alone, where the beat frequency is twice the true rate and the phase of the beat tracking is not assessed. These two evaluations meet different needs, the former if beat tracking accuracy is required, the latter if a correct median tempo measure is sufficient.

To evaluate the discrimination of the algorithm, the commonly used RWC dataset was used [11]. This dataset consists of 328 tracks in 5 sets (Classical, Jazz, Popular, "Genre" and "Royalty Free") annotated for beat times. A subset of 284 tracks was produced by eliminating pieces whose annotations were incorrect or incomplete in the RWC dataset. [2]

Since the algorithm evaluates metrical profiles, this requires meter changes to be accurately identified by the beat tracker, which currently lacks that capability. Therefore pieces with changing meters are expected to reduce the performance of the algorithm. However since this would have reduced the dataset further, and added beats or time signature changes are common in many genres of music, the dataset was used with these potential noise sources.

To evaluate octave error detection independent of the quality of the beat tracking, pieces which were incorrectly beat tracked were eliminated from the test set. This was defined as a beat tracking F-score below 0.5 using a temporal window of each annotated beat position within 15% of each inter-beat interval [5,26]. A ground truth set of octave error examples was produced by comparing the ratio of the beat tracking recall R to precision P measures, with:

$$\hat{e} = \lfloor R/P + 0.5 \rfloor, \tag{6}$$

where $\hat{e} = 2$ indicates an octave error. These ground truth candidates were then manually auditioned to verify that they were truly octave errors.

This produced a resulting dataset of 195 pieces, termed "Good", with 46 pieces identified as actually being beat tracked at double time (an octave error). This formed the

[2] For several of the Jazz examples and the Genre examples, only the minim (half note) level was annotated.

Dataset	C.	True	S.	Prec.	Rec.	F
Good	30	46	55	0.545	0.652	0.594
Full	29	46	82	0.354	0.630	0.453

Table 2. Results of octave error detection by metrical profile analysis (beat critic). "C." indicates the number of tracks correctly identified as an octave error, "True" as the ground truth number of octave errors manually identified. "S." indicates the number of tracks selected as being an octave error. "Prec.", "Rec." and "F" indicates the precision, recall and F-score measures respectively.

			Pre-Reest.		Post-Reest.		
Dataset	Meth.	Size	OE	NE	OE	NE	%
Good	BT	195	46		20		43
Good	BPM	195	44	10	24	12	54
Full	BT	284	63		37		58
Full	BPM	284	57	42	38	46	66

Table 3. Number of tracks with beat tracking octave errors (OE) before (Pre) and after (Post) reestimation using the beat critic. The column labelled "%" indicates the reduction in octave errors. NE columns indicates non-octave errors.

ground truth to evaluate the octave error identification algorithm. From these, standard precision, recall and F-score measures can be computed [26]. The entire set of 284 pieces (termed "Full") was also used to evaluate performance when beat tracking does not perform optimally.

To determine the improvement the beat critic makes to beat tracking, pieces which were determined to be beat tracked with octave error were recomputed with half the prior tempo. This would occur for false as well as true positives. The beat tracker would then use the new weighting towards the half tempo, but could produce the same result as the original beat tracking if the Viterbi decoding still biased towards the original tempo estimate [22].

The Good and Full datasets were also assessed for their fidelity to the annotated median tempo measurement τ of each track. This was computed as $\tau = 60/\tilde{i}$, where $\tilde{i}$ is the median inter-beat interval in seconds. A beat tracked tempo which was within 3% of the annotated tempo was deemed a successful tempo estimation.

5. RESULTS

The results of evaluating the beat critic with the Good and Full RWC datasets appear in Table 2. On the "Good" dataset, while the critic is able to identify 65% of the pieces with octave errors (the recall), it produces a sizeable number of false positives (the precision) which reduces the F-score. As to be expected, with the "Full" dataset, the performance is worse. The substantially higher number of false positives for this dataset indicate that the octave error measure is sensitive to beat tracking error. As the algorithm is defined, the measure of sub-beat alternation is probably too reliant on the expectation that the beat is correctly tracked.

Despite the relatively low scoring results, Table 3 indicates the success of the beat critic when used to reestimate the beat tracker. The column "Meth." describes the method of evaluation, either "BT" for beat tracking, comparing each beat location against annotated beats, or "BPM", comparing estimated tempo against annotated tempo. "Size" describes the number of tracks in the dataset. "OE" indicates the number of tracks that were beat tracked that are evaluated to have been an octave error. "Pre" and "Post" indicates the number of tracks before and after reestimating using the beat critic to bias prior tempo of the beat tracker. "NE" indicates the number of tracks that were not beat tracked correctly but were not octave errors. While it is possible to identify non-octave errors with BPM evaluation within a perceptually meaningful tolerance (3%, see Section 4), this can not be defined properly when the measure of beat tracking is calculated in terms of precision, recall and F-score.

In the case of the BT evaluation, the number of octave errors were reduced to 43% and 58% of the former number of errors for the Good and Full datasets respectively. This indicates that the Viterbi decoding of the beat tracker has benefitted from reestimation and is reasonably robust to the false positives identified as octave errors. The tempo evaluation also showed similar improvements, reducing octave errors to 54% and 66% (Good and Full). The slight increase in non-octave errors after reestimation indicates cases where the false positives have lead to mistracking. Depending on the application, this may be an unacceptable deterioration in performance despite an increase in the overall number of correctly tracked pieces.

6. CONCLUSIONS

A method for the detection of octave errors in beat tracking has been proposed and evaluated. The approach was evaluated with an audio dataset that represents a variety of genres of music. This approach, while currently applied to only one beat tracker, depends only on the presence of a mid-level representation, and the determination of beat and meter periods, commonly produced by many beat trackers. It is applicable to beat trackers which benefit from reestimation or convergence in the selection of the beat tracking frequency.

While the performance of the beat critic is well below perfection, when applied to a beat tracker, it has been shown to improve overall performance, reducing the number of octave errors, at the cost of a slight increase in mistracking. The beat critic's applicability and usefulness is ultimately dependent on the cost of false positives.

A number of improvements are possible. The use of a threshold for the octave error classification is simplistic and possibly difficult to set accurately. A machine learning classifier promises to perform better in this task. However, the best features to be used are not yet clear, preliminary experiments with the quaver alternation measures q, $\acute{q}$, $\dot{q}$ and $\ddot{q}$ indicate that these are insufficient features to dis-

criminate the octave error classification. The alternative, using the entire profiles, or reductions thereof, as features produces too high a dimensionality for accurate learning. Another issue is the relative computational cost of such an approach, when the current threshold approach is computationally low. In principle the approach could be used to identify beat tracking at half the correct rate, although such beat tracking errors did not occur using the dataset and therefore have not been evaluated.

The beat critic exploits knowledge of rhythmic behaviour as represented in musicologically based models of metrical profiles to compare temporal levels. The comparison of the relative activity of levels is used to identify octave errors. By examining the behaviour of events in the time domain, the goal has been to circumvent limitations in the temporal resolution of frequency based analysis in the identification of beat levels.

7. ACKNOWLEDGEMENTS

This research was supported by the French project Oseo "Quaero". Thanks are due to Geoffroy Peeters for provision of the beat-tracker and onset detection code.

8. REFERENCES

[1] Jeffrey A. Bilmes. Timing is of the essence: Perceptual and computational techniques for representing, learning, and reproducing expressive timing in percussive rhythm. Master's thesis, Massachusetts Institute of Technology, September 1993.

[2] Judith C. Brown. Determination of the meter of musical scores by autocorrelation. *Journal of the Acoustical Society of America*, 94(4):1953–7, 1993.

[3] Ali Taylan Cemgil and Bert Kappen. Monte Carlo methods for tempo tracking and rhythm quantization. *Journal of Artifical Intelligence Research*, 18:45–81, 2003.

[4] Martin Coath, Susan Denham, Leigh M. Smith, Henkjan Honing, Amaury Hazan, Piotr Holonowicz, and Hendrik Purwins. Model cortical responses for the detection of perceptual onsets and beat tracking in singing. *Connection Science*, 21(2):193—205, 2009.

[5] Matthew E. P. Davies and Mark D. Plumbley. A spectral difference approach to downbeat extraction in musical audio. In *EUSIPCO*, 2006.

[6] Matthew E. P. Davies and Mark D. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech and Language Processing*, 15(3):1009–20, 2007.

[7] Peter Desain and Henkjan Honing. Foot-tapping: A brief introduction to beat induction. In *Proceedings of the International Computer Music Conference*, pages 78–9. International Computer Music Association, 1994.

[8] Simon Dixon. Evaluation of the audio beat tracking system BeatRoot. *Journal of New Music Research*, 36(1):39–50, 2007.

[9] Douglas Eck. Beat induction with an autocorrelation phase matrix. In M. Baroni, A. R. Addessi, R. Caterina, and M. Costa, editors, *Proceedings of the 9th International Conference on Music Perception and Cognition (ICMPC)*, page 931, Bologna, Italy, 2006. SMPC and ESCOM.

[10] Paul Fraisse. Rhythm and tempo. In Diana Deutsch, editor, *The Psychology of Music*, pages 149–80. Academic Press, New York, 1st edition, 1982.

[11] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, Classical, and Jazz music databases. In *Proceedings of the International Symposium on Music Information Retrieval*, pages 287–288, October 2002.

[12] Mari Riess Jones. Time, our lost dimension: Toward a new theory of perception, attention and memory. *Psychological Review*, 83(5):323–55, 1976.

[13] Mari Riess Jones. Musical time. In *Oxford Handbook of Music Psychology*, pages 81–92. Oxford University Press, 2009.

[14] Mari Riess Jones and Marilyn Boltz. Dynamic attending and responses to time. *Psychological Review*, 96(3):459–91, 1989.

[15] Anssi P. Klapuri, Antti J. Eronen, and Jaakko T. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):342–55, 2006.

[16] James Koetting. What do we know about African rhythm? *Ethnomusicology*, 30(1):58–63, 1986.

[17] Edward W. Large and John F. Kolen. Resonance and the perception of musical meter. *Connection Science*, 6(2+3):177–208, 1994.

[18] Justin London. *Hearing in Time: Psychological Aspects of Musical Meter*. Oxford University Press, 2004.

[19] H. Christopher Longuet-Higgins and Christopher S. Lee. The perception of musical rhythms. *Perception*, 11:115–28, 1982.

[20] James G. Martin. Rhythmic (hierarchical) versus serial structure in speech and other behaviour. *Psychological Review*, 79(6):487–509, 1972.

[21] Caroline Palmer and Carol L. Krumhansl. Mental representations for musical meter. *Journal of Experimental Psychology - Human Perception and Performance*, 16(4):728–41, 1990.

[22] Geoffroy Peeters. Template-based estimation of time-varying tempo. *EURASIP Journal on Advances in Signal Processing*, (67215):14 pages, 2007. doi:10.1155/2007/67215.

[23] Eric D. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, 103(1):588–601, 1998.

[24] Uwe Seifert, Fabian Olk, and Albrecht Schneider. On rhythm perception: Theoretical issues, empirical findings. *Journal of New Music Research*, 24(2):164–95, 1995.

[25] Leon van Noorden and Dirk Moelants. Resonance in the perception of musical pulse. *Journal of New Music Research*, 28(1):43–66, 1999.

[26] C. V. van Rijsbergen. *Information Retrieval*. Butterworth, London; Boston, 2nd edition, 1979.

[27] Maury Yeston. *The Stratification of Musical Rhythm*. Yale University Press, New Haven, 1976. 155p.

BOOSTING FOR MULTI-MODAL MUSIC EMOTION CLASSIFICATION

Qi Lu, Xiaoou Chen, Deshun Yang, Jun Wang
Peking University
Institute of Computer Science & Technology
{luqi,chenxiaoou,yangdeshun,wangjun}@icst.pku.edu.cn

ABSTRACT

With the explosive growth of music recordings, automatic classification of music emotion becomes one of the hot spots on research and engineering. Typical music emotion classification (MEC) approaches apply machine learning methods to train a classifier based on audio features. In addition to audio features, the MIDI and lyrics features of music also contain useful semantic information for predicting the emotion of music. In this paper we apply AdaBoost algorithm to integrate MIDI, audio and lyrics information and propose a two-layer classifying strategy called Fusion by Subtask Merging for 4-class music emotion classification. We evaluate each modality respectively using SVM, and then combine any two of the three modalities, using AdaBoost algorithm (MIDI+audio, MIDI+lyrics, audio+lyrics). Moreover, integrating this in a multimodal system (MIDI+audio+lyrics) allows an improvement in the overall performance. The experimental results show that MIDI, audio and lyrics information are complementary, and can be combined to improve a classification system.

Key Words: Music Emotion Classification, Multi-Modal, AdaBoost, Fusion by Subtask Merging

1. INTRODUCTION AND RELATED WORKS

Music Information Retrieval is a sub-area of information retrieval. Important research directions include for example similarity retrieval, musical genre classification, or music analysis and knowledge representation. As the music databases grow, classification and retrieval of music by emotion [2]-[7] has recently received increasing attention.

Traditionally music emotion classification (MEC) applies algorithms of machine learning on audio features, such as Mel frequency cepstral coefficient (MFCC), to recognize the emotion embedded in the audio signal. Meanwhile we can also use some mid-level audio features such as chord [5] or rhythmic patterns [8] for this problem, but sometimes it can't get a promising result because of the semantic gap.

Complementary to audio features, lyrics are semantically rich and expressive and have profound impact on human perception of music [17]. It is often easy for us to tell from the lyrics whether a song expresses love, sadness, happiness, or something else. Incorporating lyrics in the analysis of music emotion is feasible because most popular songs sold in the market come with lyrics and because most lyrics are composed in accordance with music signal [18].

Besides music's audio and lyrics features, the MIDI features of music have been ever used in music instrument classification and retrieval. As a popular file format for storing music, MIDI carries more abstract music information than audio. In this paper we firstly apply the music's MIDI file to the music emotion classification.

A multi-modal analysis approach using audio and lyrics features has been proposed and evaluated in music genre classification by Mayer and Neumayer [1]. And promising results have been achieved by combining the audio and lyrics using various types of machine learning algorithms such as SVM and k-NN. Besides, several multi-modal fusion methods using audio and lyrics for music emotion classification are proposed by Yang [2]. However, little has been reported in the literature that applies

AdaBoost to multi-modal automatic music emotion classification. In this paper, we propose a new multi-modal fusing approach that uses features extracted from MIDI files, audio signal and lyrics for 4-class music emotion classification. We focus on how to combine the three modalities: MIDI, audio and lyrics using AdaBoost.

The remainder of the paper is organized as follows. Section 2 describes the MIDI, audio and lyrics features we need respectively. Section 3 describes the details of the proposed multi-modal approach. Section 4 provides the result of a performance study, and Section 5 concludes the paper.

2. FEATURES

In our experiment we use a free program jMIR1.0 with default parameter values to extract MIDI and audio features. jAudio and jSymbolic are two important components of jMIR for extracting audio and MIDI features. jAudio is a software package for extracting features from audio files. These extracted features can then be used in many areas of music information retrieval (MIR) research. jSymbolic is a software package for extracting high-level musical features from symbolic music representations, specifically MIDI files.

2.1 MIDI Features

The MIDI music files are firstly transformed from the corresponding waveform files by a computer tool WIDI Recognition System Professional 4.1 which could be found on the internet [19]. And then we use jSymbolic with default parameter values to extract MIDI features from the MIDI files. The extracted MIDI features, which are listed in **Table 1**, are adopted in our experiments.

#	Feature	Dimensions
1	Duration	1
2	Acoustic Guitar Fraction	1
3	Average Melodic Interval	1
……		
101	Voice Separation	1
102	Woodwinds Fraction	1

Table 1. MIDI features extracted by jSymbolic.

From **Table 1** we can see there are 102 features extracted by jSmbolic from each MIDI music file. As each feature just has one dimension, a whole MIDI feature vector has 102 dimensions.

2.2 Audio Features

We use jAudio to extract a number of low-level audio features from the waveform files. The extracted features, which are listed in **Table 2**, have been commonly used for MEC in pervious works [3]-[5].

#	Feature	Dimensions
1	Magnitude Spectrum	Variable
2	FFT Bin Frequency Labels	Variable
3	Spectral Centroid	1
……		
25	Zero Crossings	1
26	Beat Sum	1

Table 2. Audio features extracted by jAudio.

From **Table 2** we can see there are 26 features extracted by jAudio from each audio file. Among these 26 features, there are 5 features such as Magnitude Spectrum and MFCC with variable dimensions, other ones with 1 dimension. In our experiment, an audio feature vector has 79 dimensions.

2.3 Lyrics Features

Lyrics are normally available on the web and downloadable with a simple crawler. The acquired lyrics are preprocessed with traditional information retrieval operations such as stopword removal, stemming, and tokenization. In our experiment, two algorithms are adopted to generate textual features.

Uni-gram A standard textual feature representation counts the occurrence of uni-gram terms (words) in each document, and constructs the bag-of-words model [10], which represents a document as a vector of terms weighted by a tf-idf function defined as:

$$tfidf(t_i, d_j) = \#(t_i, d_j) \log \frac{|D|}{\#D(t_i)} \quad (1)$$

where $\#(t_i, d_j)$ denotes the frequency of term t_i oc-

curs in document d_j, $\#D(t_i)$ the number of documents in which t_i occurs, and $|D|$ the size of the corpus. We compute the tf-idf for each term and select the M most frequent terms as our features (M is empirically set to 2000 in this work by a validation set).

Bi-gram N-gram is sequences of N consecutive words [10]. An N-gram of size 1 is a uni-gram (single word), size 2 is a bi-gram (word pairs). N-gram models are widely used to model the dependency of words. Since negation terms often reverse the meaning of the words next to them, it seems reasonable to incorporate word pairs to the bag-of-words model to take the effect of negation terms into account. To this end, we select the M most frequent uni-gram and bi-gram in the bag-of-words model and obtain a new feature representation.

3. TAXONOMY

We adopt Thayer's arousal-valence emotion plane [15] as our taxonomy and define four emotion classes happy, angry, sad, and relaxing, according to the four quadrants of the emotion plane, as shown in **Figure 1**. As arousal (how exciting/calming) and valence (how positive/ negative) are the two basic emotion dimensions found to be most important and universal [16], we can also view the four-class emotion classification problem as the classification of high/low arousal and positive/negative valence. This view will be used in mutli-modal music emotion classification.

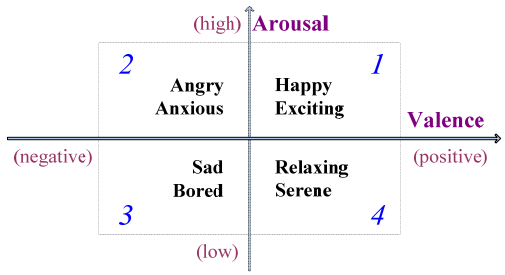


Figure 1. Thayer's arousal-valence emotion plane. We define four emotion classes according to the four quadrants of the emotion plane. We can also subdivide the four-class emotion classification to binary arousal classification and valence classification.

4. PROPOSED APPROACH

In this paper, we use AdaBoost, an ensemble learning algorithm, to train a classifier by integrating MIDI, audio and lyrics features. Boosting is a method to combine a collection of weak classification functions (weak learner) to form a stronger classifier [21]. AdaBoost is an adaptive algorithm to boost a sequence of classifiers, in that the weights are updated dynamically according to the errors in previous learning [22].

Tieu and Viola [12] adapted AdaBoost algorithm for natural image retrieval. They made the weak learner work in a single feature each time. So after T rounds of boosting, T features are selected together with the T weak classifiers. We adapted AdaBoost algorithm of Tieu and Viola's version for music emotion classification and retrieval. In each iteration, we made the weak learner work on each modality independently. So we can get three classifiers which are trained according to MIDI, audio and lyrics features respectively each time. And then we select the classifier of the minimum learning error as the representative of this iteration. After T rounds of boosting, T weak classifiers are produced in the end.

The classic AdaBoost algorithm is only used for binary classification. In a 4-class scenario, we propose a two-layer classifying strategy called Fusion by Subtask Merging.

•**Fusion by Subtask Merging (FSM):** Use AdaBoost to classify arousal and valence separately and then merge the result. To enhance readability, we denote the classification model trained by AdaBoost for classifying arousal and valence as M_A and M_V, respectively. For example, a negative arousal (predicted by M_A) and negative valence (predicted by M_V) would be merged to class 3. We make the three modalities focus on different emotion classification subtasks because empirical test reveals MIDI, audio and text clues are complementary and useful for different subtasks. In addition, training models for arousal and valence separately has been shown adequate.

4.1 AdaBoost

The AdaBoost algorithm We adapted in our experiment as follows:

Input: 1) n training examples

$(x_1, y_1), \ldots, (x_n, y_n)$ with $y_i = 1$ or 0;

2) the number of iterations T.

Initialize weights $w_{1,i} = \frac{1}{2l}$ or $\frac{1}{2m}$ for $y_i = 1$ or 0, with $l + m = n$.

Do for $t = 1, \ldots, T$ && $\varepsilon_t \leq 0.5$:

1. Train one hypothesis h_j for each modality j with w_t, and error $\varepsilon_j = \sum_{i=1}^{n} (h_j(x_i) - y_i) * w_{t,i}$.

2. Choose $h_t(\cdot) = h_k(\cdot)$ such that $\forall j \neq k, \varepsilon_k \prec \varepsilon_j$. Let $\varepsilon_t = \varepsilon_k$.

3. Update: $w_{t+1,i} = w_{t,i} \beta_t^{e_i}$, where $e_i = 1$ or 0 for example x_i classified correctly or incorrectly respectively, and $\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$.

4. Normalize the weights so that they are a distribution, $w_{t+1,i} \leftarrow \frac{w_{t+1,i}}{\sum_{j=1}^{n} w_{t+1,j}}$.

Output the final hypothesis,

$$h_f(x) = \begin{cases} 1 \ \ if \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \qquad\qquad otherwise \end{cases} \quad (2)$$

where $\alpha_t = \log \frac{1}{\beta_t}$.

4.2 Support Vector Machine

Support vector machine (SVM) learns an optimal separating hyperplane (OSH) given a set of positive and negative examples. Kernel functions are used for SVM to learn a non-linear boundary if necessary. See Vapnik [14] for a detailed introduction of SVM. Li and Guo [13] tried to use the SVM for audio classification and retrieval. In this paper, SVM is selected as our weak learner. In our experiment we use the SMO which is a fast implementation of SVM algorithm provided by WEKA3.6.1 [20].

5. EXPERIMENTS

The music database is made up of 500 Chinese pop songs, whose emotions are labeled through a subjective test conducted by 8 participants. The corresponding lyrics are downloaded from the Internet by a web crawler. Classification accuracy is evaluated by randomly selecting 400 songs as training data and 100 songs as testing data. We conducted 2 experiments. To assure the confidence, we performed the experiments based on a five-fold cross validation. We use the features extracted by jSymbolic for MIDI feature representation, the features extracted by jAudio for audio feature representation and the uni-gram and bi-gram based bag-of-words model for lyrics feature representation.

5.1 Single Feature Sets

In our first experiment, we apply SVM to mono-modal based music emotion 4-class classification (MEC) using MIDI, audio and lyrics information respectively. Therefore, we got three SVM classifiers which are trained on each mono-modality. Our SVM implementation is the SMO algorithm provided by WEKA3.6.1 and the kernel function is Polynomial. To enhance readability, we denote the classification model trained by MIDI, audio and textual features as **MO**, **AO** and **LO** respectively.

- **MIDI-Only (MO):** Use MIDI features only and apply SVM to classify emotion. This serves as a baseline. MO is used to assess the importance of the MIDI modality.
- **Audio-Only (AO):** Use audio features only and apply SVM to classify emotion. This serves as a baseline because many existing MEC work adopts it [1-2]. AO is used to assess the importance of the audio modality.
- **Lyrics-Only (LO):** Use lyrics features only and apply SVM to classify emotion. This serves as a baseline because many existing MEC work adopts it [1-2]. LO is used to assess the importance of the text modality.

The Results of experiment 1 are shown in **Table 3**:

Classifier Name	Features	Accuracy(4-class)
MO	MIDI	0.586
AO	audio	0.598
LO	lyrics	0.491

Table 3. Results of mono-modal method using SVM for 4-class emotion classification.

5.2 Multi-Modal Feature Set Combinations

In our second experiment, we apply AdaBoost to multi-modal based music emotion classification. And we select SVM as the weak learner in AdaBoost. We develop and evaluate the following method for fusing MIDI, audio and lyrics. To enhance readability, we denote the classification model trained by MIDI and audio features set, MIDI and lyrics features set, audio and lyrics features set, MIDI, audio and lyrics features set as **MA**, **ML**, **AL** and **MAL** respectively.

• **MIDI+Audio (MA):** Use MIDI and audio features and apply AdaBoost to classify emotion. The weak learner is SVM.

• **MIDI+Lyrics (ML):** Use MIDI and lyrics features and apply AdaBoost to classify emotion. The weak learner is SVM.

• **Audio+Lyrics (AL):** Use audio and lyrics features and apply AdaBoost to classify emotion. The weak learner is SVM.

• **MIDI+Audio+Lyrics (MAL):** Use MIDI, audio and lyrics features and apply AdaBoost to classify emotion. The weak learner is SVM.

The Results of experiment 2 are shown in **Table 4**:

Classifier Name	Features	Accuracy(4-class)
MA	MIDI+audio	0.616
ML	MIDI+lyrics	0.712
AL	audio+lyrics	0.72
MAL	MIDI+audio+lyrics	0.724

Table 4. Results of multi-modal fusion method using AdaBoost for 4-class emotion classification.

4.3 Comparison and Analysis of Experimental Results

Because of the different database, it is difficult to quantitatively compare the proposed approach with existing ones. Alternatively, we treat MO, AO and LO as the three baselines, and compare the classification accuracy of mono-modal and multi-modal approaches.

It can be observed from row 2 to 4 of **Table 3** that MIDI features, audio features and textual features performs very poor on 4-class emotion classification, with MO's accuracy 58.6%, AO's accuracy 59.8%, LO's accuracy 49.1%. But from row 2 to 4 of **Table 4**, we can see MIDI features, audio features and lyrics features are fairly complementary, because the combination of any two of them outperforms the mono-modal approach, with MA's accuracy 61.6%, ML's accuracy 71.2%, AL's accuracy 72.0%. **Table 4** also indicates that the 4-class emotion classification accuracy can be significantly improved by fusing all the three modalities. Among the fusion methods (rows 2-5 of **Table 4**), MAL achieves the best classification accuracy (72.4%) and contributes a 23.3% relative improvement over the lyrics-only (LO) baseline. This seems to imply the individual strength of the three modalities should be emphasized separately.

6. CONCLUSION

In this paper we have described a preliminary multi-modal approach to music emotion classification that exploits features extracted from the MIDI, audio and the lyrics of a song. We apply AdaBoost algorithm to ensemble the three modalities. A new approach of multi-modal fusion method called Fusion by Subtask Merging (FSM) is developed and evaluated. Experiments on a moderately large-scale database show that MIDI, audio and lyrics indeed carry semantic information complementary to each other. By the proposed fusion by subtask merging strategy, we can improve the classification accuracy from 49.1% to 72.4%. Using lyrics features also significantly improves the accuracy of valence classification from 61.6% to 72.4%. Meanwhile, we find that MIDI and audio features contribute fairly to the music emotion classification. From the result, we can see that the accuracy of MO is 58.6%, while that of AO is 59.8%. Besides, the accuracy of ML is 71.2%, while that of AL is 72.0%. An explanation for this phenomenon is that there exists some redundancy between MIDI and audio information. As well, an exploration of more natural language processing algorithms and more effective features for modeling the characteristics of lyrics is underway. Besides, we're trying to verifying more ensemble learning algorithms on multi-modal music emotion classification.

7. REFERENCES

[1] Rudolf Mayer et al: "Multi-modal Analysis of Music: A large-scale Evaluation," *In Proceedings of the Workshop on Exploring Musical Information Spaces* , 2009.

[2] Yang, Y.-H. et al: "Toward multi-modal music emotion classification," *Proc. PCM*, pp. 70-79, 2008.

[3] Yang, Y.-H. et al: "A regression approach to music emotion recognition," *IEEE Trans. Audio, Speech and Language Processing*, Vol. 16, No. 2, pp. 448-457, 2008.

[4] Lu, L. et al: "Automatic mood detection and tracking of music audio signals," *IEEE Trans. Audio, Speech and Language Processing*, Vol. 14, No. 1, pp. 5-18, 2006.

[5] Cheng, H.-T. et al: "Automatic chord recognition for music classification and retrieval," *Proc. ICME*, pp. 1505-1508, 2008.

[6] Yang, D. et al: "Disambiguating music emotion using software agents," *Proc. ISMIR*, pp. 52-58, 2004.

[7] Chuang, Z.-J. et al: "Emotion recognition using audio features and textual contents," *Proc. ICME*, pp. 53-56, 2004.

[8] Chua, B.-Y. et al: "Perceptual rhythm determination of music signal for emotion-based classification," *Proc. MMM*, pp. 4-11, 2006.

[9] Yo-Ping Huang and Guan-Long Guo et al: "Using Back Propagation Model to Design a MIDI Music Classification System," *Int. Computer Symposium*, Dec. 15-17, 2004, Taipei, Taiwan.

[10] Sebastiani F.: "Machine learning in automated text categorization," *ACM CSUR*, Vol. 34, No. 1, pp. 1-47, 2002.

[11] G. Guo, H. Zhang, and S. Z. Li: "Boosting for content-based audio classification and retrieval: an evaluation," *Microsoft Research Tech. Rep.* MSR-TR-2001-15.

[12] K.. Tieu and P. Viola: "Boosting image retrieval," in *Proc. of Computer Vision and Pattern Recognition*, v. 1, pp. 228-235, 2000.

[13] S. Z. Li and G. Guo: "Content-based audio classification and retrieval using svm learning," *(invited talk), PCM*, 2000.

[14] V. N. Vapnik: "Statistical learning theory," John Wiley& Sons, New York, 1998.

[15] Thayer, R. E. et al: "The Biopsychology of Mood and Arousal", Oxford University Press, New York, 1989.

[16] Russel, A.: "A circumplex model of affect", *Journal of Personality & Social Science*, Vol. 39, No. 6, pp. 1161-1178, 1980.

[17] Omar Ali, S. et al: "Songs and emotions: are lyrics and melodies equal partners", *Psychology of Music*, Vol. 34, No. 4, pp. 511-534, 2006.

[18] Formas, J.: "The words of music", *Popular Music and Society*, Vol. 26, No. 1, 2003.

[19] http://www.widisoft.com/english/download.html

[20] http://www.cs.waikato.ac.nz/ml/weka/

[21] Yoav Freund and Robert E. Schapire: "A short introduction to boosting," Journal of Japanese Society for Artificial Intelligence, Vol. 14, No. 3, pp. 771–780, 1999.

[22] Y. Freund and R. E. Schapire: "A decision-theoretic generalization of online learning and an application to boosting," *Journal* of Computer and *System* Sciences, Vol. 55, No. 1, pp. 119-139, 1997.

[23] C. Cortes and V. Vapnik: "Support-Vector Networks," Machine Learning, Vol. 20, No. 3, pp. 273-297, 1995.

CLUSTERING BEAT-CHROMA PATTERNS IN A LARGE MUSIC DATABASE

Thierry Bertin-Mahieux
Columbia University
tb2332@columbia.edu

Ron J. Weiss
New York University
ronw@nyu.edu

Daniel P. W. Ellis
Columbia University
dpwe@ee.columbia.edu

ABSTRACT

A musical style or genre implies a set of common conventions and patterns combined and deployed in different ways to make individual musical pieces; for instance, most would agree that contemporary pop music is assembled from a relatively small palette of harmonic and melodic patterns. The purpose of this paper is to use a database of tens of thousands of songs in combination with a compact representation of melodic-harmonic content (the beat-synchronous chromagram) and data-mining tools (clustering) to attempt to explicitly catalog this palette – at least within the limitations of the beat-chroma representation. We use online k-means clustering to summarize 3.7 million 4-beat bars in a codebook of a few hundred prototypes. By measuring how accurately such a quantized codebook can reconstruct the original data, we can quantify the degree of diversity (distortion as a function of codebook size) and temporal structure (i.e. the advantage gained by joint quantizing multiple frames) in this music. The most popular codewords themselves reveal the common chords used in the music. Finally, the quantized representation of music can be used for music retrieval tasks such as artist and genre classification, and identifying songs that are similar in terms of their melodic-harmonic content.

1. INTRODUCTION

The availability of very large collections of music audio present many interesting research opportunities. Given millions of examples from a single, broad class (e.g. contemporary commercial pop music), can we infer anything about the underlying structure and common features of this class? This paper describes our work in this direction.

What are the common features of pop music? There are conventions of subject matter, instrumentation, form, rhythm, harmony, and melody, among others. Our interest here is in the tonal content of the music – i.e. the harmony and melody. As a computationally-convenient proxy for a richer description of the tonal content of audio, we use the popular chroma representation, which collapses an acoustic spectrum into a 12-dimensional description, with one bin for each semitone of the western musical octave. In addition, we simplify the time axis of our representation to take advantage of the strong beat present in most pop music, and record just one chroma vector per beat. This beat-synchronous chromagram representation represents a typical music track in a few thousand values, yet when resynthesized back into audio via modulation of octave-invariant "Shepard tones", the melody and chord sequences of the original music usually remain recognizable [7]. To the extent, then, that beat-chroma representations preserve tonal content, they are an interesting domain in which to search for patterns – rich enough to generate musically-relevant results, but simplified enough to abstract away aspects of the original audio such as instrumentation and other stylistic details.

Specifically, this paper identifies common patterns in beat-synchronous chromagrams by learning codebooks from a large set of examples. The individual codewords consist of short beat-chroma patches of between 1 and 8 beats, optionally aligned to bar boundaries. The additional temporal alignment eliminates redundancy that would be created by learning multiple codewords to represent the same motive at multiple beat offsets. The codewords are able to represent the entire dataset of millions of patterns with minimum error given a small codebook of a few hundred entries. Our goal is to identify meaningful information about the musical structure represented in the entire database by examining individual entries in this codebook. Since the common patterns represent a higher-level description of the musical content than the raw chroma, we also expect them to be useful in other applications, such as music classification and retrieving tonally-related items.

Prior work using small patches of chroma features includes the "shingles" of [3], which were used to identify "remixes", i.e., music based on some of the same underlying instrument tracks, and also for matching performances of Mazurkas [2]. That work, however, was not concerned with extracting the deeper common patterns underlying different pieces (and did not use either beat- or bar-synchronous features). Earlier work in beat-synchronous analysis includes [1], which looked for repeated patterns within single songs to identify the chorus, and [7], which cross-correlated beat-chroma matrices to match cover versions of pop music tracks. None of these works examined or interpreted the content of the chroma matrices in any detail. In contrast, here we hope to develop a codebook whose entries

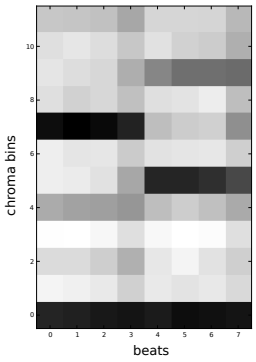


Figure 1: A typical codeword from a codebook of size 200 (code 7 in Figure 4), corresponding to a tonic-subdominant chord progression. The patch is composed of 2 bars and the pattern length was set to 8 beats.

are of interest in their own right.

2. APPROACH

2.1 Features

The feature analysis used throughout this work is based on Echo Nest analyze API [4]. For any song uploaded to their platform this analysis returns a chroma vector (length 12) for every music event (called "segment"), and a segmentation of the song into beats and bars. Beats may span or subdivide segments; bars span multiple beats. Averaging the per-segment chroma over beat times results in a beat-synchronous chroma feature representation similar to that used in [7]. Echo Nest chroma vectors are normalized to have the largest value in each column equal to 1.

Note that none of this information (segments, beats, bars) can be assumed perfectly accurate. In practice, we have found them reasonable, and given the size of the data set, any rare imperfections or noise can be diluted to irrelevance by the good examples. We also believe that patch sizes based on a number of beats or bars are more meaningful than an arbitrary time length. This is discussed further in Section 5.1.

2.2 Beat-Chroma Patches

We use the bar segmentation obtained from the Echo Nest analysis to break a song into a collection of beat-chroma "patches", typically one or two bars in length. Because the bar length is not guaranteed to be 4 beats, depending on the meter of a particular song, we resample each patch to a fixed length of 4 beats per bar (except where noted). However, the majority (82%) of our training data consisted of bars that were 4 beats long, so this resampling usually had no effect. Most of the remaining bars (10%) were 3 beats in length. The resulting patches consist of 12×4 or 12×8 matrices.

Finally, we normalize the patches with respect to transposition by rotating the pattern matrix so that the first row contains the most energy. This can be seen in the example codeword of Figure 1. Each patch within a song is normalized independently, so reconstruction of the original song requires knowledge of the rotation index for each patch.

The representation resulting from this process is invariant to both the key and meter of the original song. This enables the study of broad harmonic patterns found throughout the data, without regard for the specific musical context. In the context of clustering this avoids problems such as obtaining separate clusters for every major triad for both duple and triple meters.

2.3 Clustering

We use an online version of the vector quantization algorithm [8] to cluster the beat-chroma patches described in the previous section. For each sample from the data, the algorithm finds the closest cluster in the codebook and updates the cluster centroid (codeword) to be closer to the sample according to a learning rate ℓ. The clusters are updated as each data point is seen, as opposed to once per iteration in the standard k-means algorithm. The details are explained in Algorithm 1. As in standard k-means clustering, the codebook is initialized by choosing K random points from our dataset. Note that this algorithm, although not optimal, scales linearly with the number of patches seen and can be interrupted at any time to obtain an updated codebook.

Algorithm 1 Pseudocode for the online vector quantization algorithm. Note that we can replace the number of iterations by a threshold on the distortion over some test set.

ℓ learning rate
$\{P_n\}$ set of patches
$\{C_k\}$ codebook of K codes
Require: $0 < \ell \leq 1$
  for $nIters$ **do**
    for $p \in \{P_n\}$ **do**
      $c \leftarrow \min_{c \in C_k} \mathrm{dist}(p, c)$
      $c \leftarrow c + (p - c) * \ell$
    end for
  end for
  return $\{C_k\}$

3. EXPERIMENTS

In this section we present different clustering experiments and introduce our principal training and test data. Some detailed settings of our algorithm are also provided. As for any clustering algorithm, we measure the influence of the number of codewords and the training set size.

3.1 Data

Our training data consists of $43,300$ tracks that were uploaded to morecowbell.dj, [1] an online service based on the Echo Nest analyze API which remixes uploaded music by adding cowbell and other sound effects synchronized in

[1] http://www.morecowbell.dj/

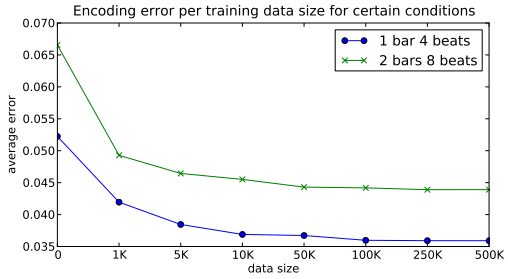


Figure 2: Distortion for a codebook of size 100 encoding one bar at a time with by 4 columns. Therefore, each codeword has $12 \times 4 = 48$ elements. Distortion is measured on the test set. Training data sizes range from 0 (just initialization) to $500,000$. Patterns were selected at random from the dataset of approximately 3.7 million patterns.

time with the music. The 43.3K songs contain 3.7 million non-silent bars which we clustered using the approach described in the previous section.

For testing, we made use of low quality (32kbps, 8 kHz bandwidth mono MP3) versions of the songs from the *uspop2002* data set [5]. This data set contains pop songs from a range of artists and styles. *uspop2002* serves as test set to measure how well a codebook learned on the Cowbell data set can represent new songs. We obtained Echo Nest features for the 8651 songs contained in the dataset.

3.2 Settings

We take one or two bars and resample the patches to 4 or 8 columns respectively. We learn a codebook of size K over the Cowbell dataset using the online VQ algorithm (Algorithm 1). We use a learning rate of $\ell = 0.01$ for 200 iterations over the whole dataset. We then use the resulting codebook to encode the test set. Each pattern is encoded with a single code. We can measure the average distance between a pattern and its encoding. We can also measure the use of the different codes, i.e., the proportion of patterns that quantize to each code.

We use the average squared Euclidean distance as the distortion measure between chroma patches. Given a pattern p_1 composed of elements $p_1(i,j)$, and a similar pattern p_2, the distance between them is:

$$\text{dist}(p_1, p_2) = \sum_{i,j} \frac{(p_1(i,j) - p_2(i,j))^2}{\text{size}(p_1)} \qquad (1)$$

We assume p_1 and p_2 have the same size. This is enforced by the resampling procedure described in Section 2.

3.3 Codebook properties

This section presents some basic results of the clustering. While unsurprising, these results may be useful for comparison when reproducing this work.

- Encoding performance improves with increasing training data (Figure 2). Distortion improvements plateau by around 1000 samples per codeword ($100,000$ samples for the 100-entry codebook of the figure).

Codebook size	Distortion
1	0.066081
10	0.045579
50	0.038302
100	0.035904
500	0.030841

Table 1: Distortion as a function of codebook size for a fixed training set of $50,000$ samples. Codebook consists of 1 bar (4 beat) patterns.

- Encoding performance improves with increasing codebook size (Table 1). Computation costs scales with codebook size, which limited the largest codebooks used in this work, but larger codebooks (and more efficient algorithms to enable them) are clearly a promising future direction.

- Larger patterns are more difficult to encode, thus requiring larger codebooks. See Figure 3. The increase is steepest below 4 beats (1 bar), although there is no dramatic change at this threshold.

4. VISUALIZATION

4.1 Codebook

We trained a codebook containing 200 patterns sized 12×8, covering 2 bars at a time. The results shown are on the *artist20* test set described in Section 5.2.

The 25 most frequently used codewords in the test set are shown in Figure 4. The frequency of use of these codewords is shown in Figure 5. The codewords primarily consist of sustained notes and simple chords. Since they are designed to be key-invariant, specific notes do not appear. Instead the first 7 codewords correspond to a single note sustained across two bars (codeword 0), perfect fifth (codewords 1 and 2) and fourth intervals (codewords 3 and 6, noting that the fourth occurs when the per-pattern transposition detects the fifth rather than the root as the strongest chroma bin, and vice-versa), and a major triads transposed to the root and fifth (codewords 5 and 4, respectively). Many of the remaining codewords correspond to common

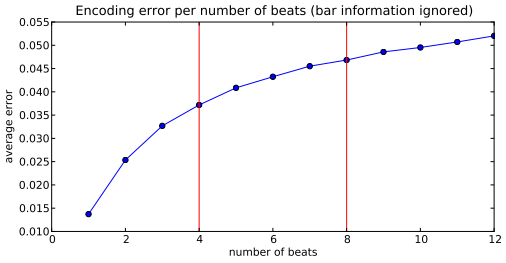


Figure 3: Encoding patterns of different sizes with a fixed size codebook of 100 patterns. The size of the pattern is defined by the number of beats. Downbeat (bar alignment) information was not used for this experiment.

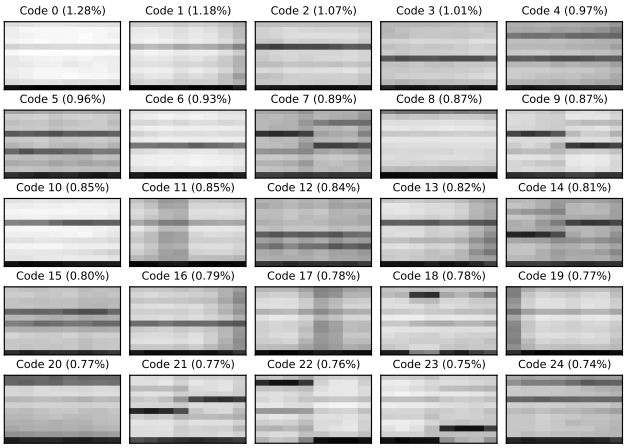


Figure 4: The 25 codes that are most commonly used for the *artist20* test set. Codes are from the 200-entry codebook trained on 2 bar, 12×8 patches. The proportion of patches accounted for by each pattern is shown in parentheses.

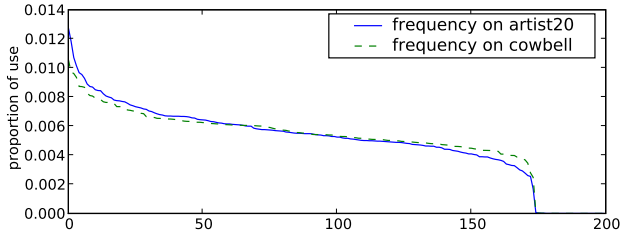


Figure 5: Usage proportions for all 200 codewords on the *artist20* test set (which comprises $71,832$ patterns). Also shown are the usage proportions for the training set ("cowbell"), which are similar. Note that even though all codewords are initialized from samples, some are used only once in the training set, or not at all for test set. This explains why the curves drop to 0.

transitions from one chord to another, e.g. a V-I transition in codes 7 and 9 (e.g., Gmaj $\rightarrow$ Cmaj, or G5 $\rightarrow$ C5 as a guitar power chord) and the reverse I-V transition in code 21 (e.g., Cmaj $\rightarrow$ Gmaj).

In an effort to visualize the span of the entire codebook, we used Locally linear embedding (LLE) [9] [2] to arrange the codewords on a 2D plane while keeping similar patterns as neighbors. Figure 6 shows the resulting distribution along with a sampling of patterns; notice sustained chords on the top left, chord changes on the bottom left, and more complex sustained chords and "wideband" noisy patterns grouping to the right of the figure.

Noting that many of the codewords reflect sustained patterns with little temporal variation, Figure 7 plots the average variance along time of all 200 patterns. Some 26% of the codewords have very low variance, corresponding to stationary patterns similar to the top row of Figure 4.

We made some preliminary experiments with codebooks based on longer patches. Figure 8 presents a codewords from an 8 bar (32 beat) codebook. We show a random selection since all the most-common codewords were less interesting, sustained patterns.

[2] implementation: `http://www.astro.washington.edu/users/vanderplas/coding/LLE/`

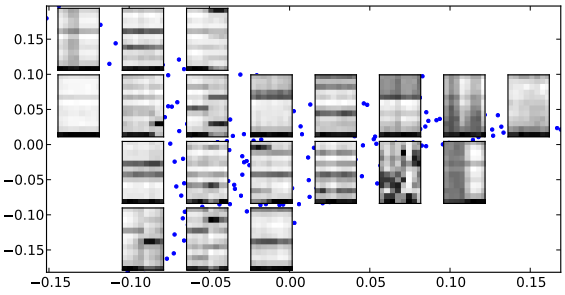


Figure 6: LLE visualization of the codebook. Shown patterns are randomly selected from each neighborhood.

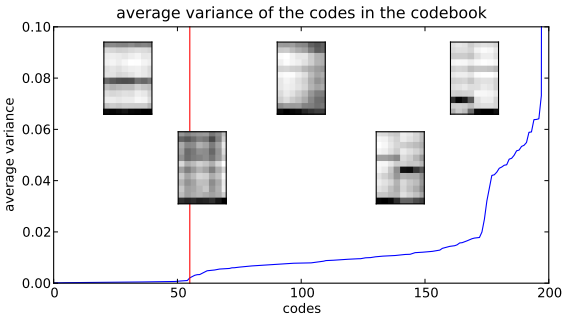


Figure 7: Average variance of codewords along the time dimension. The vertical axis cuts at the 53rd pattern, roughly the number of codewords consisting entirely of sustained chords. Representative patterns are shown in each range.

4.2 Within-cluster behavior

In addition to inspecting the codewords, it is important to understand the nature of the cluster of patterns represented by each codeword, i.e., how well the centroid describes them, and the kind of detail that has been left out of the codebook. Figure 9 shows a random selection of the 639 patterns from the *artist20* test set that were quantized to codeword 7 from Figure 4, the V-I cadence. Figure 10 illustrates the difference between the actual patterns and the quantized codeword for the first three patterns; although there are substantial differences, they are largely unstruc-

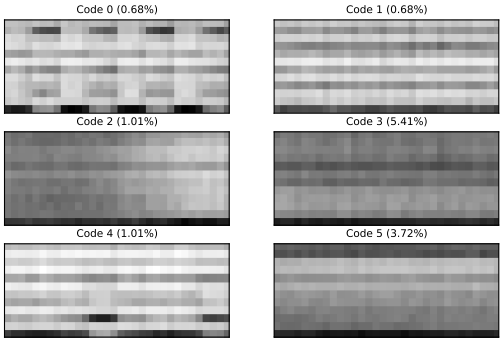


Figure 8: Sample of longer codewords spanning 8 bars. Codewords were randomly selected from a 100-entry codebook. Percentages of use are shown in parentheses. Most patterns consist of sustained notes or chords, but code 0 shows one-bar alternations between two chords, and code 4 contains two cycles of a $1 \rightarrow 1 \rightarrow 1 \rightarrow 2$ progression.

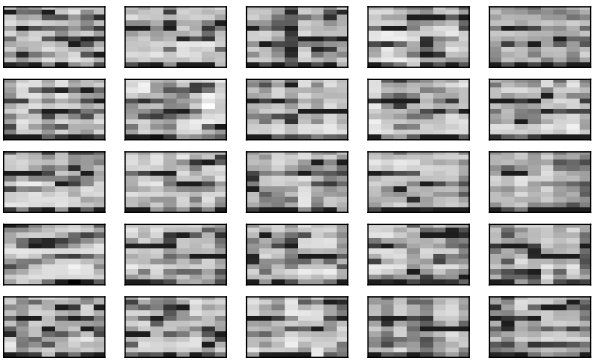


Figure 9: Cluster around centroid presented in Figure 1. Taken from the *artist20* dataset, the cluster size is actually 639. Shown samples were randomly selected. This gives an intuition of the variance in a given cluster.

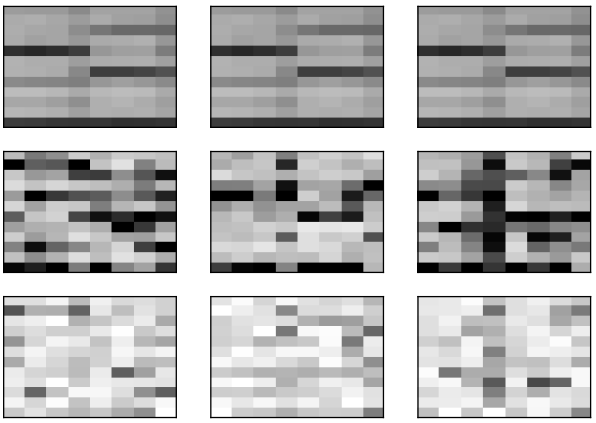


Figure 10: First three patterns of figure 9 (2nd line) presented with the centroid from Figure 1 (1st line) and the absolute difference between both (3rd line).

tured, indicating that the codebook has captured the important underlying trend.

4.3 Example song encoding

Figure 11 gives an example of encoding a song using the codebook, showing both the full, original data, and the reconstruction using only the quantized codewords (at the correct transpositions). The quantized representation retains the essential harmonic structure of the original features, but has smoothed away much of the detail below the level of the 2 bar codewords.

5. APPLICATIONS

We present two applications of the beat-chroma codebooks to illustrate how the "natural" structure identified via unsupervised clustering can provide useful features for subsequent supervised tasks. We will discuss how the codewords can be used in bar alignment, and artist recognition.

5.1 Bar Alignment

Since the clustering described in Section 2 is based on the segmentation of the signal in to bars, the codewords should

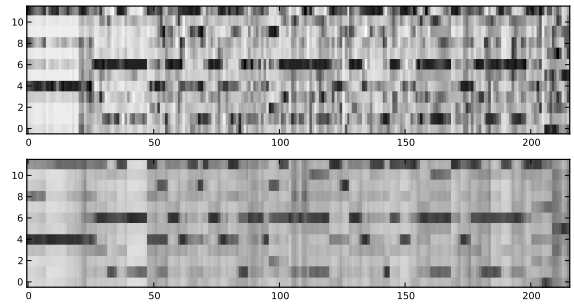


Figure 11: Good Day Sunshine by *The Beatles*. Original song and encoding with a 200 entry codebook of 2 bar patterns.

Offset	% of times chosen
0	**62.6**
1	16.5
2	9.4
3	11.5

Table 2: Bar alignment experiment: offsets relative to ground-truth 4-beat bar boundaries that gave minimum distortion encodings from the bar-aligned codebook.

contain information related to bar alignment, such as the presence of a strong beat on the first beat. In this section we investigate using the codebook to identify the bar segmentation of new songs. We train a codebook of size 100 on bars resampled to 4 beats. Then, we take the longest sequence of bars of 4 beats for each song in the test set (to avoid the alignment skew that results from spanning irregularly-sized bars). We then encode each of these sequences using an offset of 0, 1, 2 or 3 beats, and record for each song the offset giving the lowest distortion. The results in Table 2 show that the "true" offset of 0 beats is chosen in 62% of cases (where a random guess would yield 25%). Thus, the codebook is useful for identifying bar (downbeat) alignments. A more flexible implementation of this idea would use dynamic programming to align bar-length patterns to the entire piece, including the possibility of 3- or 5-beat bars (as are sometimes needed to accommodate beat tracking errors) with an appropriate associated penalty.

5.2 Artist Recognition

We apply our codebook to a simple artist recognition task. We use the *artist20* data set, composed of 1402 songs from 20 artists, mostly rock and pop of different subgenres. Previously published results using GMMs on MFCC features achieve an accuracy of 59%, whereas using only chroma as a representation yields an accuracy of 33% [6].

Although beat-chroma quantization naturally discards information that could be useful in artist classification, it is interesting to investigate whether some artist use certain patterns more often than others.

The dataset is encoded as histograms of the codewords used for each song, with frequency values normalized by the number of patterns in the song. We test each song in a

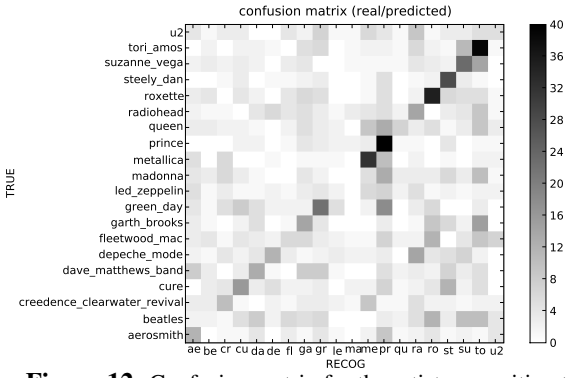
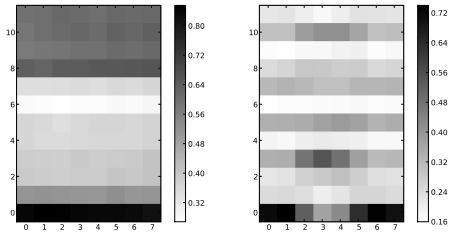


Figure 12: Confusion matrix for the artist recognition task.



(a) Metallica (b) Tori Amos / Suzanne Vega

Figure 13: Typical patterns for different artists.

leave-one-out setting, and represent each of the 20 artists by the average of their (remaining) song-histograms. The test song is matched to an artist based on minimum Euclidean distance to these per-artist averages. This gives an accuracy of 23.4%, where the random baseline is around 5%. The confusion matrix can be seen in Figure 12, showing that certain artists are recognized at an accuracy far above the average.

It is interesting to inspect the most "discriminative" patterns for individual artists. To find these patterns, we compare a pattern's use by one artist and divide by its use across all artists. Figure 13 shows the dominant patterns for Metallica, and for Tori Amos and Suzanne Vega (who shared a 'favorite' pattern). These three artists were easily identified. Artists like Metallica are characterized by "wideband" patterns, with energy spread across multiple adjacent chroma bins, indicative of noise-like transients in the audio.

6. CONCLUSION AND FUTURE WORK

We have presented a practical method to perform large-scale clustering of tonal patterns, and assessed the basic properties of the method through experiments on a large collection of music. We have explored several ways to inspect and interpret the data and suggested the merit of the representation through further experiments. We have discussed the possibility to move to even larger scales and we provide our source code for other interested researchers [3].

Future work may include more sophisticated clustering that moves beyond simple Euclidean-distance-based quantization, perhaps by separately modeling the spread within each cluster (i.e., a Gaussian mixture or other generative model). Summarizing patches with Gaussians, and then comparing the distance between those Gaussians, could reduce the influence of the noise in the distance measure.

Moving on to larger scales, we would like to pursue a scheme of incrementally splitting and merging codewords in response to a continuous, online stream of features, to create an increasingly-detailed, dynamic model. We could also cluster codebooks themselves, in a fashion similar to hierarchical Gaussian mixtures [10].

[3] See *Papers* section at `www.columbia.edu/~tb2332/`

7. ACKNOWLEDGEMENTS

Thanks to Graham Grindlay for numerous discussions and helpful comments. T. Bertin-Mahieux is supported by a NSERC scholarship. This material is based upon work supported by IMLS grant LG-06-08-0073-08 and by NSF grant IIS-0713334. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

8. REFERENCES

[1] M. A. Bartsch and G. H. Wakefield. To catch a chorus: using chroma-based representations for audio thumbnailing. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, New York, October 2001.

[2] M. Casey, C. Rhodes, and M. Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech & Language Processing*, 2008.

[3] M. Casey and M. Slaney. Fast recognition of remixed music audio. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

[4] The Echo Nest Analyze, API, `http://developer.echonest.com`.

[5] D. Ellis, A. Berenzweig and B. Whitman. The "uspop2002" pop music data set. `http://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html`.

[6] D. Ellis. Classifying music audio with timbral and chroma features. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, 2007.

[7] D. Ellis and G. Poliner. Identifying cover songs with chroma features and dynamic programming beat tracking. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

[8] A. Gersho and R. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, 1991.

[9] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[10] N. Vasconcelos. Image indexing with mixture hierarchies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I–3–I–10 vol.1, 2001.

WHAT'S HOT ? ESTIMATING COUNTRY-SPECIFIC ARTIST POPULARITY

Markus Schedl[1], **Tim Pohle**[1], **Noam Koenigstein**[2], **Peter Knees**[1]

[1] Department of Computational Perception
Johannes Kepler University, Linz, Austria

[2] Faculty of Engineering
Tel Aviv University, Tel Aviv, Israel

ABSTRACT

Predicting artists that are popular in certain regions of the world is a well desired task, especially for the music industry. Also the cosmopolitan and cultural-aware music aficionado is likely be interested in which music is currently "hot" in other parts of the world. We therefore propose four approaches to determine *artist popularity rankings* on the country-level. To this end, we mine the following data sources: *page counts from Web search engines*, *user posts on Twitter*, *shared folders on the Gnutella file sharing network*, and *playcount data from last.fm*. We propose methods to derive artist rankings based on these four sources and perform cross-comparison of the resulting rankings via overlap scores. We further elaborate on the advantages and disadvantages of all approaches as they yield interestingly diverse results.

1. INTRODUCTION

To determine popular artists for a certain country or cultural region of the world, one can obviously look into publicly available music charts, such as the "Billboard Hot 100", released weekly for the United States of America by the *Billboard Magazine* [6]. However, this straightforward strategy is hardly feasibly when we aim at broaden the scope to the whole world. The reasons are manifold.

First, not all countries do release music charts for various reasons. Causes may be, for example, a lack of capability to determine music sales or an underdevelopment of music distribution at large. Even if data is available, it is often not publicly accessible, and even if so, not always in an easy-to-use format, e.g., via a Web service.
Second, even if charts are available for a specific country, they often cover only certain ways of music distribution. Commonly they are strongly biased towards sales figures of music albums. In some countries, however, they also include digital music sales via online stores. This inhomogeneity between countries, i.e., the in- or exclusion of certain distribution channels, make such data hardly comparable between different countries of the world. Another aspect to be considered here are possible heavy distortions caused by (illegal) music sharing channels, since legislation in this area varies severely between countries. In fact, the majority of today's music distribution is affected via file sharing networks [2]. Thus, traditional charts, such as the "Billboard Hot 100", are becoming less and less relevant.
Third, if the aim is to come up with a list of the most popular artists ever, countries lacking solid historical records constitute an obvious problem.

Summarizing these challenges, we conclude that analyzing which kind of music is popular in a specific country or cultural region necessitates taking a deeper look into various distribution channels and data sources. In this paper, we therefore present four different approaches to estimate artist popularity rankings on the country-level, each of which makes use of a different data source. The first one is based on *page count estimates* of Web search engines, the second approach analyzes *Twitter posts*, the third one derives information from meta-data of *users' shared folders in a Peer-to-Peer network*, and the fourth one uses *playcount data from last.fm*.

In the remainder of this paper we review related literature (Section 2), present four approaches to determine artist popularity on the country-level (Section 3), elaborate on the conducted evaluation experiments and discuss their results (Section 4), and finally draw conclusions (Section 5).

2. RELATED WORK

Related work falls into two categories: literature that particularly tackles the task of chart prediction, and work that relates to the four approaches we propose for this task.

Targeting the problem of predicting music charts, Koenigstein and Shavitt [26] present an approach to predict the charts based on search queries issued within the Peer-to-Peer (P2P) file sharing network *Gnutella* [35]. The authors show that a song's popularity in the P2P network highly correlates with its ranking in the Billboard charts. The authors' approach can further predict upcoming charts with high accuracy. However, for their analysis Koenigstein and Shavitt only consider the United States.
Pachet and Roy [33] try to predict the popularity of a song based on audio features and a variety of manual labels. The authors' conclusion is, however, that even state-of-the-art machine learning techniques fail to learn factors that determine a song's popularity, irrespective of whether they are trained on signal-based features or on high-level human annotations.

In [38] Schedl et al. propose several heuristics to determine which artists are popular within a certain genre. They relate occurrence counts of artist names on Web pages via an approach similar to *Google*'s backlink and forward link analysis [34]. The authors show that downranking factors for artist names equaling common speech terms improve accuracy when comparing the resulting rankings against a ground truth popularity categorization extracted from *allmusic.com* [3].
In [22] Grace et al. derive popularity rankings from user comments in the social network *MySpace* [32]. To this end, the authors apply various annotators to crawled *MySpace* artist pages in order to spot, for example, names of artists, albums, and tracks, sentiments, and spam. Subsequently, a data hypercube (OLAP cube) is used to represent structured and unstructured data, and to project the data to a popularity dimension. A user study showed that the list generated by this procedure was on average preferred to the Billboard charts.

Previous work that relates to the four approaches proposed here comprise the following.
Our heuristic that uses *page counts* returned by search engines builds upon work from [20, 39], where Web co-occurrences of artist names and terms specific to the music domain are used to categorize artists, a process also known as "autotagging" [13]. In [37] Schedl et al. propose a similar approach to estimate artist similarity. The authors suggest a simple probabilistic model that defines similarity between two artists a and b as the conditional probability of a to be mentioned on a Web page known to relate to b and vice versa. Accuracies of up to 85% were reported for genre classification.
To the best of our knowledge, *Twitter* [41] has not been scientifically investigated for music information extraction and retrieval yet. Although there do exist certain commercial services, such as *BigChampagne* [7] and *Band Metrics* [9], which seem to incorporate microblogging data into their artist and song rankings, no details on their approach are available. Furthermore, they strongly focus their services on the USA. A general study on the use of *Twitter* can be found in [24]. Java et al. report that *Twitter* is most popular in North America, Europe, and Asia (Japan), and that same language is an important factor for cross-connections ("followers" and "friends") over continents. The authors also distill certain categories of user intentions to microblog. Employing the *HITS* algorithm [25] on the network constructed by "friend"-relations, Java et al. derive user intentions from structural properties. They identified the following categories: information sharing, information seeking, and friendship-wise relationships. Analyzing the content of *Twitter* posts, the authors distill the following intentions: daily chatter, conversations, sharing information/URLs, and reporting news.
Using *Peer-to-Peer networks* as data source for music information retrieval, [8, 14, 31, 43] rely on data extracted from *OpenNap* to derive music similarity information. All of these papers seem to build upon the same data set, which comprises of metadata on shared content (approximately 3,000 shared music collections were analyzed). Logan et al. [31] compare similarities defined by artist co-occurrences in shared folders, by expert opinions from *allmusic.com*, by playlist co-occurrences from *Art of the Mix* [4], by data gathered from a Web survey, and by MFCC features [5]. To this end, they calculate a "ranking agreement score", i.e., the pairwise overlap between the N most similar artists according to each data source. The main findings are that the co-occurrence data from *OpenNap* and from *Art of the Mix* show a high degree of overlap, the experts from *allmusic.com* and the participants of the Web survey show a moderate agreement, and the signal-based MFCC measure had a rather low agreement with the music context-based data sources. More recently, in [40] Shavitt and Weinsberg mine the *Gnutella* file sharing network to derive artist and song similarities. The authors gathered metadata of shared music files from about one million *Gnutella* users in November 2007, which yielded information on half a million songs. Analyzing the 2-mode graph of users and songs revealed that most users share similar files. The authors further propose a method for artist recommendation based on the gathered data.
Taking a closer look at the data source of *music information systems*, which corresponds to the fourth approach, not only *last.fm* [28] provides popularity rankings via their API [29]. *Echonest* [15] offers a function to retrieve a ranking based on the so-called "hotttness" of an artist [17]. This ranking is based on editorial, social, and mainstream aspects [16]. However, this Web service does not provide country-specific information.

3. DETERMINING ARTIST POPULARITY ON THE COUNTRY LEVEL

We propose the following four heuristics to determine an artist's popularity in a certain country, and consequently create an artist popularity ranking. To this end, we first retrieve a list of 240 countries from *last.fm* [30], based on which the following approaches operate.

3.1 Search Engine Page Counts

This approach makes use of a search engine's number of indexed Web pages for a given query, a count usually referred to as *page count*. These page counts are, however, only rough estimates of the real number of available Web pages related to the query. Nevertheless, for the purpose of classifying music artists into genres [20, 37, 39] and for classifying general instances according to a given ontology as well as for learning sub- and superconcept relations [11, 12], this method yielded respectable results.

For the paper at hand, we queried the search engines *Google* [21] and *Exalead* [18], using their API or issuing HTTP requests. The page count values returned for all $\langle$artist, country$\rangle$ tuples were retrieved. To avoid excessive bandwidth consumption, we restrict the number of search results to be transmitted to the smallest value (this is usually one result). Since we are only interested in the page count estimates, this restriction effectively reduces network traffic without effecting the results.
The two main challenges of this approach are directing the search towards pages related to the music domain and alleviating the distortions caused by artist names that equal common speech words. We address these issues by using queries of the form

```
"artist name" "country name" music
```

and weighting the resulting page count values with a factor resembling the *inverse document frequency (idf)* [46]. The final ranking score is thus calculated according to Formula 1, where $pc_{c,a}$ is the page count value returned for the country-specific query for artist a and country c, N is the total number of countries for which data is available, and df_a is the number of countries in which artist a is known

according to the data source (i.e., the number of countries with $pc_{c,a} > 0$).

$$popularity\_pc_{c,a} = pc_{c,a} \cdot \log_2 \left(1 + \frac{N}{df_a}\right) \quad (1)$$

3.2 Twitter Posts

Many *Twitter* posts reveal information about what people are doing or thinking right now. We are interested in posts containing information about which music is currently being played by users in a given country. To accomplish this, we retrieve posts using the *Twitter* Search API [42]. The posts are then narrowed in two ways. First, we only search for posts containing the hashtag *#nowplaying*. This restriction is directly supported by the *Twitter* API. As a second restriction, the search is narrowed to a specific country. Not being aware of a more direct implementation for the second restriction, we search only for posts whose users are located within a certain radius around a GPS coordinate. More specifically, for a given country, we determine the coordinates of larger cities (with more than 100,000 inhabitants) and search for posts originating from a circle of 100 kilometers around the respective coordinates. The names of the cities are taken from *Wikipedia*, e.g., [45], and the coordinates are determined by using *Freebase* [19]. For each city location for which geolocation data is resolved successfully, all *Twitter* posts available through the *Twitter* API are retrieved, which yields a maximum of about 1,500 posts per city location.

One of the advantages of using this kind of data is certainly its recentness. Thus, the retrieved data may contain artists that do not appear in our list of most popular artists (cf. Section 4.1). A first look at the format of the texts reveals that automatic tokenization seems not easily to accomplish due to the large variation of wording and creative methods to use the available number of characters. We therefore opt to scan the retrieved texts for the artists contained in the artist list, and we count the number of their appearances for a given country c. This count equals the term frequency ($tf_{c,a}$) of a in an aggregated document comprising all posts gathered for cities in country c. Formula 2 gives the ranking score. The rightward term again represents an *idf*-factor that downranks artists that are popular everywhere, and thus not specific to country c. N is the total number of countries, and df_a is the number of aggregated country documents in which artist a occur.

$$popularity\_twi_{c,a} = tf_{c,a} \cdot \log_2 \left(1 + \frac{N}{df_a}\right) \quad (2)$$

3.3 Shared Folders in a P2P Network

Collecting shared folder data from *Gnutella* users is a two-staged-process. First, a *crawler* needs to discover the current network topology (which is very dynamic). Subsequently, a *browser* queries the active users for their shared folders data. The crawler treats the network as a graph, and performs a breadth-first exploration, where newly discovered nodes are enqueued in a list of un-crawled addresses. The crawler provides a list of active IP addresses to the browser, which sends *Gnutella* "Query" messages [1] to the clients. The clients reply with "QueryHit" messages, that lists their shared folder content. These messages are the basis for our P2P data set.

The system described above is a different system than the one used by Koenigstein and Shavitt in [26], which collected *Gnutella* search queries for song ranking. One advantage of a shared folder data set over queries is the availability of ID3 tags and hash keys, which simplifies the process of associating the digital content with a musical artist. However, when singles ranking is considered (as in [26]), queries tend to better reflect the changing popularity trends of pop songs over short time intervals. In this study, we associate artists with digital content by matching the artist names against the content of the ID3 tags. Occasionally, the content in ID3 tags is missing or misspelled. We therefore, match the artists names against the file names as well.

In order to build popularity charts for specific countries, one needs to resolve the geographical location of the users. The geo-identification is based on the IP addresses. First, we generate a list of all unique IP addresses in the data set (typically over a million). We resolve the geography of IP addresses using the commercial *IP2Location* [23] database. Each IP address is bounded with its country code, city name, and latitude-longitude values. This accurate geographical information pin points artists' fans and enables tracking spatial diffusion of artists popularity [27].

After the digital files are associated with artists names and geography, building popularity charts is straightforward. For each country, we aggregate the total number of digital content that is associated with each artist. Ranking is then performed according to frequency.

3.4 Last.fm Playcounts

We further estimate country-specific artist popularity based on the community of *last.fm* users. Despite the issues of *hacking and vandalism* as well as the *community bias* [36], which are inherent to collaborative music information systems, the playcounts of *last.fm* users can be expected to reflect to a certain extent which music is currently popular. We therefore gathered the top 400 listeners of each country at the end of 2009. We subsequently extracted the top-played artists for each of the resulting top-listeners-sets. [1] Aggregating the playcounts for each artist over a country's top listeners finally yielded a popularity ranking for the country under consideration.

4. EVALUATION

4.1 Test Set

We used *last.fm*'s Web API [29] to gather the most popular artists for each country of the world, as of November 2009. We then aggregated this data into a single list of 201,135 unique artist names.

4.2 Experiments

As we aim at assessing the pros and cons of the various approaches, without yet having an established ground truth for this kind of experiments, we choose to perform a pairwise comparison of the approaches. Each approach produces a ranked list of artists for the various countries. Expecting that the absolute numbers obtained by the various approaches are not immediately comparable, we compare the produced artist popularity rankings of two approaches

[1] In the meantime, *last.fm* has extended its API with a `Geo.getTopArtists` function, which can be used to directly retrieve the top-played artists among a certain country's users. Quick empirical comparisons showed that the implementation behind this function seems to resemble our approach.

A_j and A_k. This comparison is done separately for each country c. In the next subsections, we describe the applied data preprocessing steps and the used evaluation measure in detail.

4.2.1 Preprocessing

We start our analysis by processing the artist names in the artist popularity list for country c of each approach in a basic way (e.g., each artist name is represented in lower case, repeated whitespace characters are removed, and UTF-8-encoded characters are transformed to canonical ASCII representations).

Instead of using raw artist counts directly, we normalize them, attempting to avoid dominance of common-speech words, or globally popular artists whose popularity is not highly country specific. For each artist, the number of countries this artist appears in is counted. Each country-specific artist count $\mathrm{ac}_{c,a}$ is then normalized as indicated in Equation 1.

Artist names appearing in the two lists (given by the pair of approaches under investigation) are matched against each other, and only artists appearing in both lists are kept. Based on this data, we calculate the overlap between the rankings obtained with the two prediction approaches, as described next.

4.2.2 Evaluation Measures

The top-n rank overlap for country c between approaches A_j and A_k is calculated as

$$\mathrm{ro}_{c,A_j,A_k,n} = \frac{1}{n} \cdot \left|\{a | \max\left(r_{A_j,c,a}, r_{A_k,c,a}\right) \leq n\}\right| \quad (3)$$

where $r_{A_j,c,a}$ denotes the ranking of artist a in country c according to approach A_j, only considering the artists for which both approaches (A_j and A_k) yield a ranking score. In other words, the top-n rank overlap is the fraction of artists appearing within the top n ranked artists in both approaches. For example, if one artist is within the top-2 ranked artists of both approaches, the top-2 rank overlap is 0.5. Obviously, n can take values up to the number of artists $n_{\max,c}$ for which both approaches deliver rank data for country c, and the top-$n_{\max,c}$ rank overlap is always 1.

To obtain an overall measure for two approaches and a given country, we define the country-wise rank overlap as

$$\mathrm{cro}_{c,A_j,A_k} = \frac{1}{n_{\max,c}} \sum_{n=1}^{n_{\max,c}} \mathrm{ro}_{c,A_j,A_k,n} \quad (4)$$

which has a trivial (random) baseline of about 0.5 and a maximum value of 1.0 when both rankings are identical. The country-wise rank overlaps are further combined to obtain one overall scalar value for approaches A_j and A_k. To account for the different quantity of available information, we weight the overlap score of each country with the number of artists for which information is available. We define the overall overlap measure between approaches A_j and A_k as

$$\mathrm{ov}\left(A_j, A_k\right) = \frac{\sum_{c \in C} n_{\max,c} \cdot \mathrm{cro}_{c,A_j,A_k}}{\sum_{c \in C} n_{\max,c}} \quad (5)$$

The measure ov also has a trivial baseline of about 0.5 and a maximum value of 1.0.

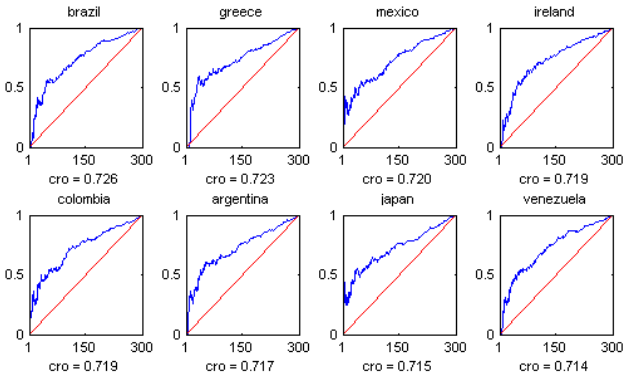


Figure 1. Top 8 countries for *pc_google* vs *p2p*.

To give an illustrative example, Figure 1 shows for the comparison of approach *pc_google* and *p2p* the 8 countries with highest ro value, as a chart from $1..n_{\max,c}$.

4.3 Results and Discussion

Each approach offers at least a slightly different view on reality since the data sources are of distinct nature. There is also no such thing as a "ground truth" for this task, as each data source (even "Billboard"-style charts) is biased, as elaborated below. Nevertheless, we would like to point out certain interesting observations.

Looking at Figure 2, the highest overlap score of 0.67 is found between *Google page counts* and *P2P*. One reason may be that the two sources have broadest coverage. Another explanation may be the time dependency. *Twitter* and *last.fm* are much more time dependent, whereas *P2P shared folders* and *amounts of Web pages* change much slower. In fact, the content of the data sources behind P2P networks and Web search engines, i.e., users' music collections and Web pages, respectively, is accumulated over years. Microblogging posts and *last.fm* data, in contrast, change much faster and are therefore more likely to reflect trends.

Second, the *page counts* approach using *Google* and the same approach using *Exalead* do not produce similar results, as we would have expected. In fact, the rankings reveal a non-significant overlap of 0.51. A possible explanation is that the two search engine providers may use very different page count estimation techniques.

Exalead shows the lowest overlap with other sources. Its highest overlap is realized, not surprisingly, with *Google* and with *P2P*, but it remains slightly above the baseline (0.53). An explanation for *Exalead*'s low overlap score becomes apparent when looking at Figure 3. *Exalead* has by far the highest number of matching artists, which may induce a high noisiness.

In terms of country coverage (cf. Figure 3), the *last.fm* and the *page counts* approaches offer data for nearly every country in the world.

To account for the different nature and scope of the proposed approaches (and underlying data sources), we compare them according to several aspects in Table 1, elaborating on specific advantages and disadvantages. One issue is that certain approaches are prone to a specific bias. For example, the average *last.fm* user does not represent the average music listener of a country, i.e., *last.fm* data is distorted by a "community bias". The same is true for *Twitter*, which is biased towards artists with very active fans. On the other hand, some very popular artists may have fans

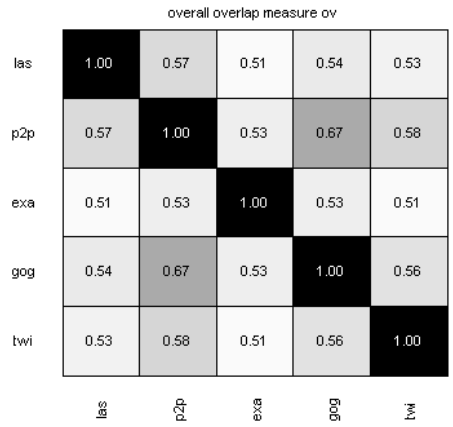


Figure 2. Overlap ov between each pair of approaches.

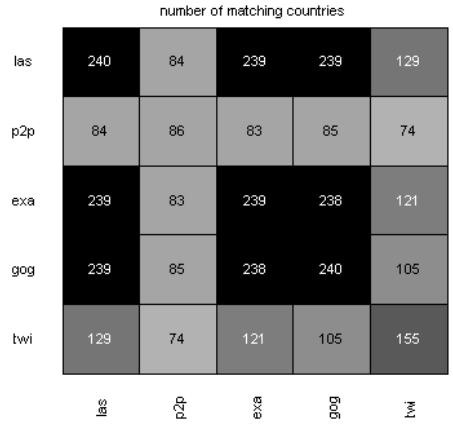


Figure 3. Number of countries with non-empty overlap.

that twitter to a much lower degree. This issue becomes especially apparent when thinking of live artists vs. dead ones: The live ones keep making new headlines, and probably also have many more active fans, while the dead ones have an inherent problem with this. Traditional charts are biased towards the data the music industry uses to derive them, usually record sales figures.

Another aspect according to which the approaches differ considerably is the availability of data. While *page count estimates* are available for all countries of the world, the *P2P* and *Twitter* approaches suffer from a very unbalanced coverage, strongly depending on the country under consideration. Also traditional music charts vary strongly between countries and continents with respect to availability. According to [44], only one country in Africa publishes official music charts, while this number amounts to 19 for Europe.

A big advantage of traditional charts is their virtual immunity against noise. *Page count estimates*, in contrast, are easily distorted by ambiguous artist or country names. *last.fm* data suffers from hacking and vandalism [10], as well as from unintentional input of wrong information and misspellings.

In the dimension of time dependence, the approaches can be categorized into "current" and "accumulating", depending on whether they reflect the instantaneous popularity, or a general, all-time popularity in that they accumulate popularity levels over time.

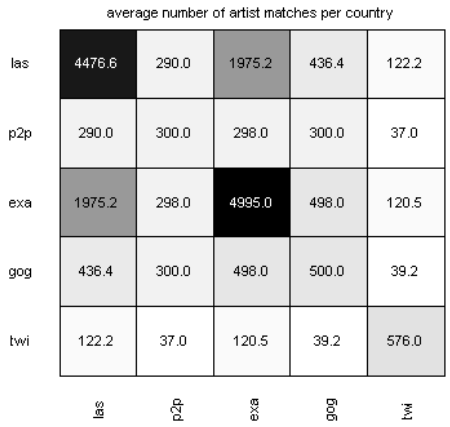


Figure 4. Average number of artists per country ($n_{\max,c}$).

5. CONCLUSIONS AND FUTURE WORK

We presented four approaches to determine country-specific artist popularity rankings based on different data sources (search engine's page counts, *Twitter* posts, shared folders in the *Gnutella* network, and playcounts of *last.fm* users). In the absence of a standardized ground truth, we performed pairwise comparison of the approaches and elaborated on particular advantages and disadvantages. Most approaches showed only weak overlaps, probably due to the different nature of their data sources. We found, however, a considerable overlap between *Google* page counts and P2P data, which is probably explained by the similar time scope the two data sources cover. As a general conclusion, we can state that artist popularity can be derived from various, quite inhomogeneous data sources. The remarkably weak overlap between most of them indicates that the quest for artist popularity is a multifaceted and challenging task, in particular in today's era of multi-channel music distribution. To derive one overall popularity measure, we will need to combine the different sources.

Future work will hence foremost aim at elaborating hybrid approaches that account for the different quantity and quality of information output by the four heuristics. We will also work on refining our approaches to capture artist popularity within certain genres, e.g., by incorporating methods similar to [38]. We will further look at the various processing steps in more detail. Most of the current implementations were created in an ad-hoc manner, and some of the choices might degrade the performance. For example, better string comparison algorithms may improve results for artists whose names may be spelled in various ways. Alternative ways of normalizing artist counts for the individual approaches are also likely to yield improvements.

6. ACKNOWLEDGMENTS

This research is supported by the *Austrian Fonds zur Förderung der Wissenschaftlichen Forschung* (FWF) under project numbers L511-N15 and Z159.

7. REFERENCES

[1] The Gnutella Protocol Specification v0.41. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf (access: March 2010).

Source/Aspect	Bias	Availability	Noisiness	Time Dependence
Page Counts	Web users	comprehensive	high	accumulating
Twitter	community	country-dependent	medium	current
P2P	community	country-dependent	low–medium	accumulating
Last.fm	community	high	medium–high	accumulating
Traditional Charts	music industry	country-dependent	low	current

Table 1. A comparison of different approaches according to various dimensions.

[2] Digital Music Report 2009. http://www.ifpi.org/content/library/DMR2009.pdf (access: May 2010), January 2009.

[3] http://www.allmusic.com (access: January 2010).

[4] http://www.artofthemix.org (access: February 2008).

[5] Jean-Julien Aucouturier, François Pachet, and Mark Sandler. "The Way It Sounds": Timbre Models for Analysis and Retrieval of Music Signals. *IEEE Transactions on Multimedia*, 7(6):1028–1035, December 2005.

[6] http://en.wikipedia.org/wiki/Billboard_Hot_100 (access: May 2009).

[7] http://www.bigchampagne.com (access: May 2010).

[8] Adam Berenzweig, Beth Logan, Daniel P.W. Ellis, and Brian Whitman. A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures. In *Proceedings of ISMIR*.

[9] www.bandmetrics.com (access: May 2010).

[10] Òscar Celma and Paul Lamere. ISMIR 2007 Tutorial: Music Recommendation.

[11] Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. Towards the Self-Annotating Web. In *Proceedings of ACM WWW*, 2004.

[12] Philipp Cimiano and Steffen Staab. Learning by Googling. *ACM SIGKDD Explorations Newsletter*, 6(2):24–33, 2004.

[13] Douglas Eck, Thierry Bertin-Mahieux, and Paul Lamere. Autotagging Music Using Supervised Machine Learning. In *Proceedings of ISMIR*, 2007.

[14] Daniel P.W. Ellis, Brian Whitman, Adam Berenzweig, and Steve Lawrence. The Quest For Ground Truth in Musical Artist Similarity. In *Proceedings of ISMIR*, 2002.

[15] http://echonest.com (access: March 2010).

[16] http://developer.echonest.com/docs/method/get_hotttnesss (access: March 2010).

[17] http://developer.echonest.com/docs/method/get_top_hottt_artists (access: March 2010).

[18] http://www.exalead.com (access: February 2010).

[19] http://www.freebase.com (access: March 2010).

[20] Gijs Geleijnse and Jan Korst. Web-based Artist Categorization. In *Proceedings of ISMIR*, 2006.

[21] http://www.google.com (access: March 2010).

[22] Julia Grace, Daniel Gruhl, Kevin Haas, Meenakshi Nagarajan, Christine Robson, and Nachiketa Sahoo. Artist Ranking Through Analysis of On-line Community Comments. In *Proceedings of ACM WWW*, 2008.

[23] http://www.ip2location.com (access: March 2010).

[24] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why We Twitter: Understanding Microblogging Usage and Communities. In *Proceedings of WebKDD/SNA-KDD*, 2007.

[25] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5), 1999.

[26] Noam Koenigstein and Yuval Shavitt. Song Ranking Based on Piracy in Peer-to-Peer Networks. In *Proceedings of ISMIR*, 2009.

[27] Noam Koenigstein, Yuval Shavitt, and Tomer Tankel. Spotting Out Emerging Artists Using Geo-aware Analysis of P2P Query Strings. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[28] http://last.fm (access: March 2010).

[29] http://last.fm/api (access: March 2010).

[30] http://www.last.fm/community/users (access: March 2010).

[31] Beth Logan, Daniel P.W. Ellis, and Adam Berenzweig. Toward Evaluation Techniques for Music Similarity. In *Proceedings of ACM SIGIR: Workshop on the Evaluation of Music Information Retrieval Systems*, 2003.

[32] http://www.myspace.com (access: November 2009).

[33] François Pachet and Pierre Roy. Hit Song Science is Not Yet a Science. In *Proceedings of ISMIR*, 2008.

[34] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. In *Proceedings of ASIS*, 1998.

[35] Matei Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network. In *Proceedings of IEEE Peer-to-Peer Computing*, 2001.

[36] Markus Schedl and Peter Knees. Context-based Music Similarity Estimation. In *Proceedings of LSAS*, 2009.

[37] Markus Schedl, Peter Knees, and Gerhard Widmer. A Web-Based Approach to Assessing Artist Similarity using Co-Occurrences. In *Proceedings of CBMI*, 2005.

[38] Markus Schedl, Peter Knees, and Gerhard Widmer. Investigating Web-Based Approaches to Revealing Prototypical Music Artists in Genre Taxonomies. In *Proceedings of ICDIM*, 2006.

[39] Markus Schedl, Tim Pohle, Peter Knees, and Gerhard Widmer. Assigning and Visualizing Music Genres by Web-based Co-Occurrence Analysis. In *Proceedings of ISMIR*, 2006.

[40] Yuval Shavitt and Udi Weinsberg. Songs Clustering Using Peer-to-Peer Co-occurrences. In *Proceedings of the IEEE ISM: International Workshop on Advances in Music Information Research (AdMIRe)*, San Diego, CA, USA, 2009.

[41] http://twitter.com (access: February 2010).

[42] http://apiwiki.twitter.com/Twitter-API-Documentation (access: March 2010).

[43] Brian Whitman and Steve Lawrence. Inferring Descriptions and Similarity for Music from Community Metadata. In *Proceedings of ICMC*, 2002.

[44] http://en.wikipedia.org/wiki/Music_charts (access: March 2010).

[45] http://en.wikipedia.org/wiki/List_of_towns_and_cities_with_100,000_or_more_inhabitants/country:_A-B (access: March 2010).

[46] Justin Zobel and Alistair Moffat. Exploring the Similarity Space. *ACM SIGIR Forum*, 32(1):18–34, 1998.

IDENTIFYING REPEATED PATTERNS IN MUSIC USING SPARSE CONVOLUTIVE NON-NEGATIVE MATRIX FACTORIZATION

Ron J. Weiss and Juan Pablo Bello
Music and Audio Research Lab (MARL), New York University
{ronw,jpbello}@nyu.edu

ABSTRACT

We describe an unsupervised, data-driven, method for automatically identifying repeated patterns in music by analyzing a feature matrix using a variant of sparse convolutive non-negative matrix factorization. We utilize sparsity constraints to automatically identify the number of patterns and their lengths, parameters that would normally need to be fixed in advance. The proposed analysis is applied to beat-synchronous chromagrams in order to concurrently extract repeated harmonic motifs and their locations within a song. Finally, we show how this analysis can be used for long-term structure segmentation, resulting in an algorithm that is competitive with other state-of-the-art segmentation algorithms based on hidden Markov models and self similarity matrices.

1. INTRODUCTION

Repetition has been widely-recognized to be a ubiquitous feature of music, closely related to structural units in music, such as beats, bars, motives and sections [10]. This applies both to popular music, often composed of nearly exact repetitions of a small number of sections, e.g. verse, chorus, and bridge; and to more sophisticated genres, e.g. jazz or orchestral music, where recurrences are often masked by complex transformations, including key modulations and tempo variations. The analysis of repeated patterns and their temporal organization is central to the understanding of music. However, while repetitions are apparent in symbolic representations of music, their extraction from musical audio poses a number of challenges stemming from factors such as the presence of background noise, the influence of multiple instruments and sonic textures, timing variations and other attributes of musical expression, etc.

The automatic analysis of repetition in music audio has been an important focus of attention in MIR, with applications including thumbnailing [1], retrieval [2], and, notably, long-term segmentation using methods such as self-similarity matrices and hidden Markov models [11, 8, 5]. However, with a few exceptions [7, 1], the emphasis has been on locating repetitions rather than on extracting of characteristic, repetitive patterns. Previous research on detecting motif occurrences across a collection [9] and cover-song retrieval based on short-snippets [3], illustrate the utility of extracting such patterns.

In this paper we propose a novel approach for the automatic extraction and localization of repeated patterns in music audio. The approach is based on sparse shift-invariant probabilistic latent component analysis [14] (SI-PLCA), a probabilistic variant of convolutive non-negative matrix factorization (NMF). The algorithm treats a musical recording as a concatenation of a small subset of short, repeated patterns, and is able to simultaneously estimate both the patterns and their repetitions throughout the song. The analysis naturally identifies the long-term harmonic structure within a song, while the short-term structure is encoded within the patterns themselves. Furthermore, we show how it is possible to utilize sparse prior distributions to learn the number of patterns and their respective lengths, minimizing the number of parameters that must be specified exactly in advance. Finally, we explore the application of this approach to long-term segmentation of musical pieces.

The remainder of this paper is organized as follows: Section 2 reviews the proposed analysis based on SI-PLCA and describes its relationship to NMF. Sections 3 and 4 describe prior distributions over the SI-PLCA parameters and the expectation maximization algorithm for parameter estimation. Sections 5 and 6 discuss how the proposed analysis can be used for structure segmentation and provide experimental results. Finally, we conclude in Section 7.

2. PROPOSED APPROACH

2.1 From NMF to PLCA

Conventional NMF decomposes a non-negative matrix V into the product of two non-negative matrices W and H:

$$V \approx WH \tag{1}$$

In the context of audio analysis, if V represents a time-frequency decomposition of an audio signal, each column of W can be thought of as a frequency template used repeatedly throughout V, and each row of H can be thought of as the activations of the corresponding basis in time. In this paper we focus on the analysis of beat-synchronous chromagrams [4], but the method is equally applicable to any non-negative time-frequency representation such as a magnitude spectrogram.

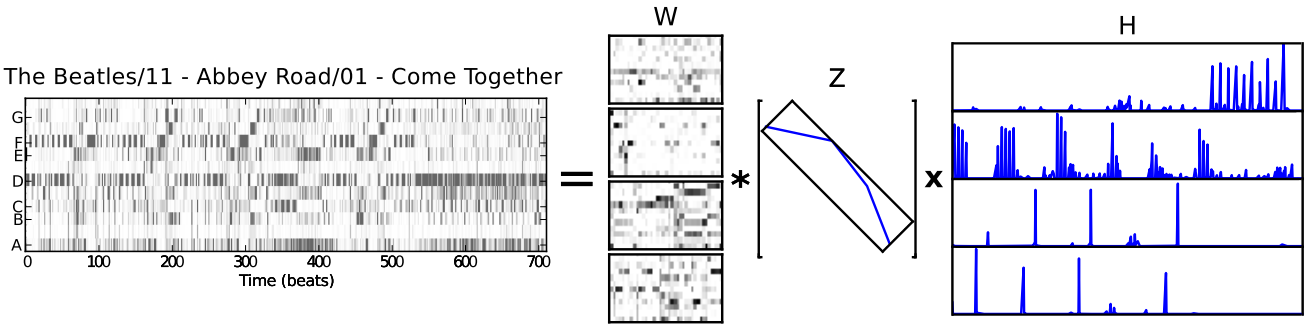


Figure 1. Demonstration of the SI-PLCA analysis of a chromagram. The decomposition was initialized with $L = 40$, and $K = 10$ with $\alpha_z = 0.98$, and no sparsity on W_k or $\mathbf{h}_k^T$. The parameter estimation algorithm pruned out most of the initial bases due to the sparse prior on $\mathbf{z}$, converging on only 4 bases.

Probabilistic Latent Component Analysis (PLCA) [14] recasts this analysis in a probabilistic framework. PLCA represents each column of W and each row of H as multinomial probability distributions and adds an additional distribution over each basis, i.e. a mixing weight. The decomposition can be rewritten in NMF terms as follows:

$$V \approx WZH = \sum_{k=0}^{K-1} \mathbf{w}_k \, z_k \, \mathbf{h}_k^T \tag{2}$$

where $Z = \mathrm{diag}(\mathbf{z})$ is a diagonal matrix of mixing weights $\mathbf{z}$ and K is the rank of the decomposition (i.e. the number of bases in W). Contrary to standard NMF, each of V, $\mathbf{w}_k$, $\mathbf{z}$, and $\mathbf{h}_k^T$ are normalized to sum to 1 since they correspond to probability distributions.

The probabilistic foundation makes for a convenient framework for imposing constraints on the parameters $\mathbf{w}_k$, $\mathbf{h}_k^T$, and $\mathbf{z}$ through the use of prior distributions. This will be discussed in detail in Section 3.

2.2 Adding shift-invariance

A shift-invariant extension to the PLCA model which allows for *convolutive* bases is described in [14]. Unlike the single frame bases $\mathbf{w}_k$ described in Section 2.1, each SI-PLCA basis is expanded to form a fixed duration template W_k containing L frames. Therefore, the $F \times K$ matrix W becomes an $F \times L \times K$ tensor $\mathcal{W}$, and the normalized basis $\mathbf{w}_k$ becomes a normalized matrix W_k. The factors $\mathcal{W}$ and H are combined via a convolution operation instead of matrix multiplication in a process analogous to equation (2):

$$V \approx \sum_k W_k * z_k \mathbf{h}_k^T \tag{3}$$

Figure 1 shows an example SI-PLCA decomposition of a chromagram using $K = 4$ basis patterns of length $L = 40$.

3. SPARSE PRIOR DISTRIBUTIONS

A common strategy used throughout the NMF literature is to favor sparse settings, i.e. one containing many zeros, for W or H in order to learn parsimonious, parts-based decompositions of the data. Sparse solutions can be encouraged when estimating the parameters in equation (3) by imposing constraints using an appropriate prior distribution. In the following sections we describe how this process can be used to automatically learn the number and length of the repeated patterns within a song.

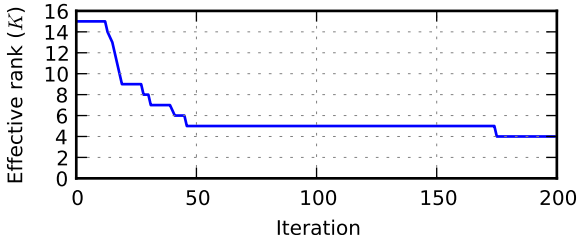


Figure 2. Typical behavior of the automatic relevance determination effect of a sparse prior on $\mathbf{z}$. The initial rank of the decomposition is set to $K = 15$, and as the estimation algorithm iterates it is pruned down to a final effective rank (the number of bases with non-zero z_k) of 4.

3.1 Learning the number of patterns K

The Dirichlet distribution is conjugate to the multinomial distributions W_k, $\mathbf{z}$, and $\mathbf{h}_k^T$, making it a natural choice for a prior. The Dirichlet prior on $\mathbf{z}$ has the following form:

$$P(\mathbf{z} \,|\, \alpha_z) \propto \prod_k z_k^{\alpha_z - 1}, \quad \alpha_z \geq 0 \tag{4}$$

where the hyperparameter α_z is fixed across all K components. If $\alpha_z < 1$ this prior favors solutions where many components are zero, i.e. where the distributions are sparse.

If $\mathbf{z}$ is forced to be sparse, the learning algorithm will attempt to use as few bases as possible. This enables an automatic relevance determination strategy in which: (a) the algorithm is initialized to use many bases (large K), and (b) the sparse prior on $\mathbf{z}$ prunes out bases that do not contribute significantly to the reconstruction of V. Only the most relevant patterns "survive" to the end of the parameter estimation process, as is shown in the example in Figure 2. This approach is useful because it removes the need to specify the exact rank of the decomposition K in advance. The parameter estimation simply learns the underlying number of patterns needed by the data. A similar approach to automatically determining the rank of a standard NMF decomposition is described in [15].

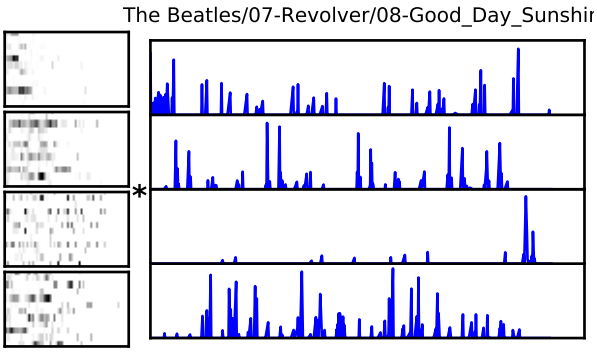


Figure 3. Demonstration of the SI-PLCA decomposition of a chromagram using $L = 60$ and sparsity in all parameters ($\alpha_z = 0.98$, $c = 16$, $m = -10^{-8}$, and $\alpha_h = 1 - 10^{-5}$).

3.2 Learning the pattern length L

The other parameter that must be specified in advance is the length L of the convolutive bases. In fact, different patterns within the same piece often have different intrinsic lengths, e.g. if the chorus uses a shorter riff than the verse or if the time signature changes. Therefore it is useful to automatically identify the length of each basis independently instead of using a fixed length across all bases.

We employ a similar strategy to that described in Section 3.1 by setting L to an upper bound on the expected pattern length and constructing a prior distribution that encourages the use of shorter bases. This is accomplished by using a Dirichlet prior on W_k with a parameter that depends on the time position τ within each basis:

$$P(W_k \,|\, \boldsymbol{\alpha}_w) \propto \prod_{\tau} \prod_{f} w_{kf\tau}^{\alpha_{w\tau}-1} \tag{5}$$

$\boldsymbol{\alpha}_w$ is constructed as a piecewise function which is uninformative for small τ and then becomes increasingly sparse:

$$\alpha_{w\tau} = \begin{cases} 1, & \tau < c \\ 1 + m\,(\tau - c), & \tau \geq c \end{cases} \tag{6}$$

This prior only effects patterns longer than c frames with a penalty that increases with the pattern length.

An example of the effect of this prior is shown in Figure 3. Most of the information in the top basis is contained within the first 12 columns, while the other bases have effective lengths between 30 and 40.

3.3 Basis/activation trade-off

It is often worthwhile to enforce sparsity on $\mathbf{h}_k^T$ using a similar approach to equation (4), with a single parameter α_h tied across all points within $\mathbf{h}_k^T$. The rationale is that if most of the activations in $\mathbf{h}_k^T$ are zero, then more of the information in V will be captured by W_k, and vice versa. A sparse $\mathbf{h}_k^T$ promotes more parsimonious patterns for W_k, at the cost of a reduced time resolution.

This is illustrated by the example in Figure 1. The second basis pattern is relatively sparse, while the corresponding row of H contains many non-zero entries. In fact, the spacing between adjacent activations in $\mathbf{h}_1^T$ is smaller than the length of the pattern; i.e. it is continually mixed with delayed versions of itself. The pattern repeats about every 8 beats, roughly corresponding to the underlying meter.

In contrast, the bottom two bases contain significantly more information while the corresponding rows of H contain only about 4 peaks. The sparsity setting α_h, in combination with $\alpha_{w\tau}$, control the trade-off between these qualitatively different solutions. A sparse H leads to more musically meaningful bases that are exactly repeated throughout the piece, while a sparse $\mathcal{W}$ leads to temporal patterns in H that are organized according to to the underlying rhythm.

4. PARAMETER ESTIMATION

The decomposition of equation (3) can be computed iteratively using an expectation maximization (EM) algorithm. The full derivation of the algorithm can be found in [13]. Here we extend it to incorporate the prior distributions described in Section 3. Since we are using conjugate prior distributions, this extension is straightforward to derive.

In the E-step, the posterior distribution over the hidden variables k and τ is computed for each cell in V. For notational convenience we represent this distribution as a set of matrices $\{R_{k\tau}\}$ for each setting of k and τ. Each point in the $F \times T$ matrix $R_{k\tau}$ corresponds to the probability that the corresponding point in V was generated by basis k at time delay τ. It can be computed as follows:

$$R_{k\tau} \propto \mathbf{w}_{k\tau} \otimes z_k \overrightarrow{\mathbf{h}_k^T}^{\tau} \tag{7}$$

where $\otimes$ denotes the outer product, and $\overrightarrow{x}^{t}$ shifts x t places to the right. The set of $R_{k\tau}$ matrices are normalized such that each point in $\sum_{k\tau} R_{k\tau}$ is one.

Given this posterior distribution, the parameters can be updated in the M-step as follows:

$$z_k \propto \sum_{\tau} \sum_{ft} V \cdot R_{k\tau} + \alpha_z - 1 \tag{8}$$

$$\mathbf{w}_{k\tau} \propto \sum_{t} V \cdot R_{k\tau} + \alpha_{w\tau} - 1 \tag{9}$$

$$\mathbf{h}_k^T \propto \sum_{\tau} \sum_{f} \overleftarrow{V}^{\tau} \cdot \overleftarrow{R_{k\tau}}^{\tau} + \alpha_h - 1 \tag{10}$$

where $\cdot$ denotes the element-wise matrix product and the parameters are normalized so that $\mathbf{z}$, W_k, and $\mathbf{h}_k^T$ sum to 1.

The overall EM algorithm proceeds by initializing W_k, $\mathbf{z}$, and $\mathbf{h}_k^T$ randomly, and then iterating equations (7) to (10) until convergence. This algorithm is only guaranteed to converge to a local optimum, so the quality of the factorization is somewhat dependent on initialization. In our experiments we found that initializing $\mathbf{z}$ and $\mathbf{h}_k^T$ uniformly while setting the initial W_k randomly leads to more consistent results.

5. STRUCTURE SEGMENTATION

As mentioned in the introduction, the analysis described in this paper can be applied to the task of music structure segmentation. It naturally identifies the long-term temporal

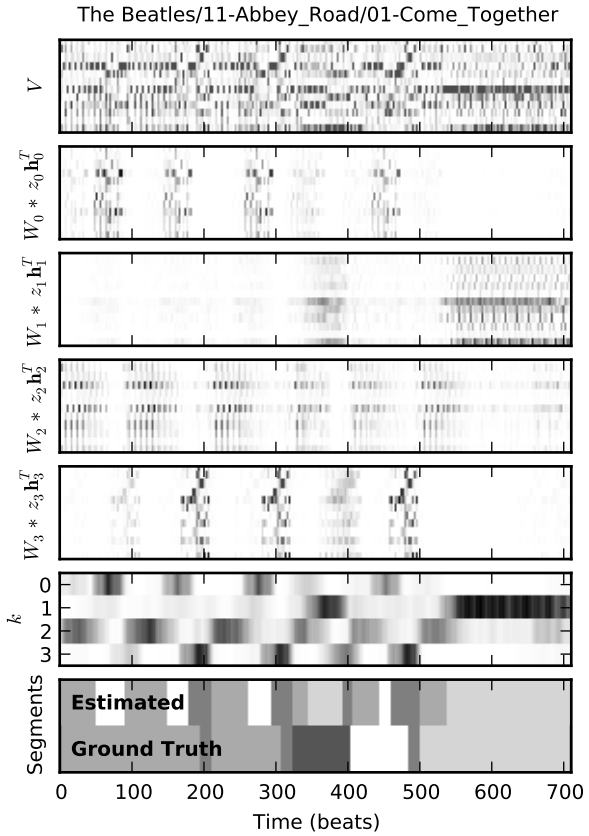


Figure 4. Song structure segmentation using the SI-PLCA decomposition shown in Figure 1. The pairwise F-measure of the estimated segmentation is 0.52.

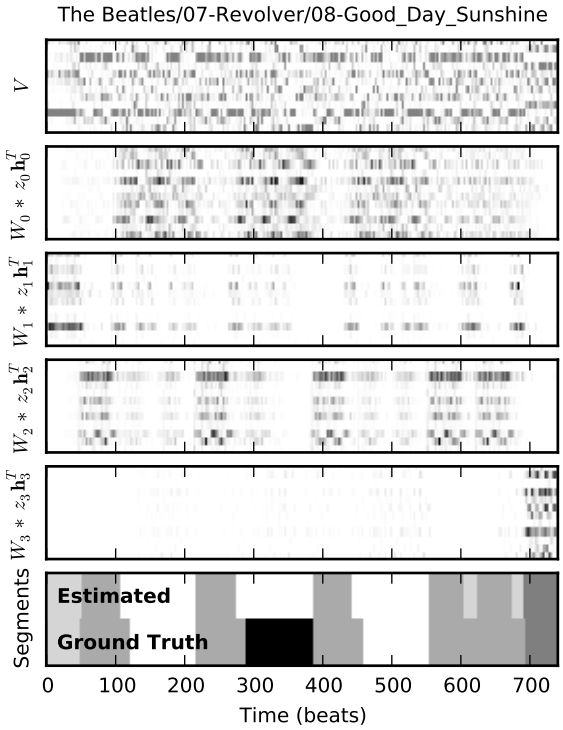


Figure 5. Song structure segmentation using the SI-PLCA decomposition shown in Figure 3 (PFM = 0.69).

structure within a song, encoded by H. At the same time, the short-term structure is captured within the bases $\mathcal{W}$.

We use the beat-synchronous chroma feature extraction from [4]. Each frame of V is normalized so that the maximum energy is one. Analysis of these features identifies repeated motifs in the form of chord patterns. We assume a one-to-one mapping between these chord patterns and the underlying song structure, i.e. we assume that each pattern is used within only one segment. The mapping is derived by computing the contribution of each pattern to the chroma gram by summing equation (3) across all pitch classes:

$$\ell_k(t) = \sum_f W_k * z_k \mathbf{h}_k^T \tag{11}$$

The segmentation labels are then found by smoothing the K "pattern usage" functions $\ell_k(t)$ using a rectangular window, and finding the most active pattern at each frame:

$$\ell(t) = \underset{k}{\text{argmax}} \ \ell_k(t) * \mathbf{1}_S \tag{12}$$

where $\mathbf{1}_S$ is a length S vector of ones. Finally, the per-frame segment labels $\ell(t)$ are post-processed to remove segments shorter than a given minimum segment length.

5.1 Examples

An example of this segmentation procedure is shown in Figure 4. The top panel shows the original chromagram of the song. The following four panels show the contribution of each pattern to the chromagram, and the bottom two panels show the smoothed $\ell_k(t)$ and the final segmentation.

There are some interesting differences between the ground truth segmentation and that derived from the proposed algorithm in Figure 4. For example, the proposed algorithm breaks the beginning of the song into repeated subsections: basis 2 (mid-gray) → basis 0 (white), while the ground truth labels this sequence as a single segment. When inspecting the actual patterns it is clear that these segments are composed of distinct chord patterns, despite serving a single musical role together ("intro/verse" as annotated in the ground truth). In fact the mid-gray and white segments are reused in different contexts throughout the song in regions with different ground-truth annotations. The analysis has no notion of musical role, so it tends to converge on solutions in which bases are reused as often as possible.

One way to address this limitation is to increase the length L of the convolutive bases (or the corresponding parameters of $\alpha_{w\tau}$), in which case the repeated sub-segments would be merged into a single long segment. This highlights an inherent trade-off in the proposed analysis between identifying simple chord patterns that are frequently repeated (short W_k, many activations in $\mathbf{h}_k^T$) as opposed to deriving long-term musical structure (longer W_k, sparser $\mathbf{h}_k^T$). This trade-off is a recognized ambiguity in the concept of musical segmentation [12].

When high-level segments are more closely correlated with the harmonic structure identified by our method, the proposed analysis leads to good segmentation. An example of this is shown in Figure 5. Note that the ground truth labels make a distinction between "verse"(white) and "verse/break" (black) which is not present in our analysis.

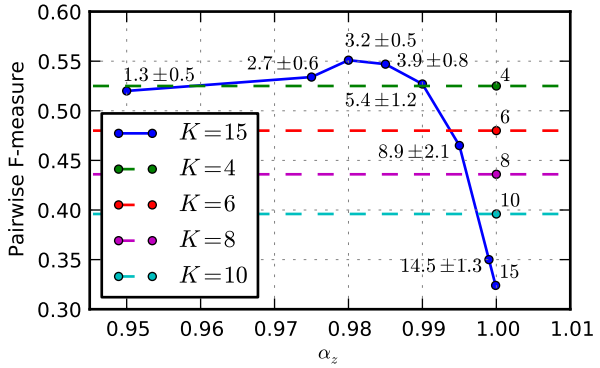


Figure 6. PFM as a function of α_z (solid line). $K = 15$, $L = 60$, and no other priors are used. The average effective rank for each setting of α_z is displayed. Also plotted is PFM for $\alpha_z = 1$ for different settings of K (dashed lines).

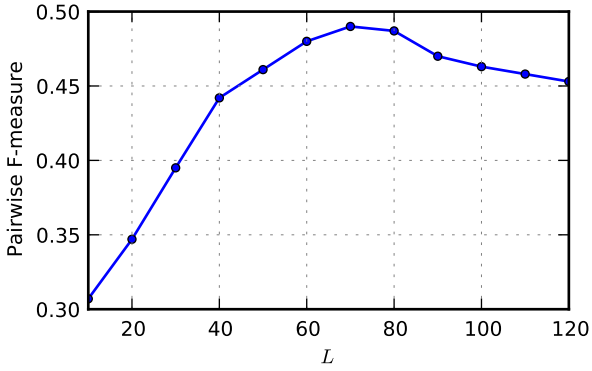


Figure 7. PFM as a function of the pattern length L. The rank is fixed at $K = 6$ and no sparse priors are used.

6. EXPERIMENTS

In this section we evaluate the proposed approach to structure segmentation. We quantify the effect of the various prior distributions described in Section 3 and compare our approach to other state-of-the-art algorithms. The test set consists of 180 songs from the recorded catalog of The Beatles, annotated into verse, chorus, refrain, etc. sections by the Centre for Digital Music. [1] Each song contains an average of about 10 segments and 5.6 unique labels.

Segmentation performance is measured using the pairwise recall rate (PRR), precision rate (PPR), and F-measure (PFM) metrics proposed in [5] which measure the frame-wise agreement between the ground truth and estimated segmentation regardless of the exact segment label. We also report the entropy-based over- and under-segmentation scores (S_o and S_u, respectively) as proposed in [6].

6.1 Number of patterns

Since our segmentation algorithm assumes a one-to-one relationship between patterns and segments, the appropriate choice of the number of patterns K is critical to obtaining good performance. We evaluate this effect by segmenting the data set with varying settings for K with $\alpha_z = 1$, and by fixing K to 15 and varying α_z. No smoothing of the resulting labels is performed ($S = 1$).

The results are shown in Figure 6. For $\alpha_z = 1$, segmentation performance decreases as K increases, peaking at $K = 4$. Performance improves when the sparse prior is applied for most settings of α_z. The average effective rank and its standard deviation both increase with decreasing α_z (increasing sparsity). The best performance is obtained for $\alpha_z = 0.98$, leading to an average effective rank of 3.2 ± 0.5. These results demonstrate the advantage of allowing the number of patterns to adapt to each song.

6.2 Pattern length

As described in Section 5.1, the length of the patterns used in the decomposition has a large qualitative effect on the segmentation. To measure this effect, we segmented the entire corpus varying L between 10 and 120 beats. No sparsity was enforced, so the pattern length remained fixed for all bases and all songs. The results are shown in Figure 7.

As predicted, segmentation performance is poor for small L since the ground truth segments are often divided into many distinct short segments. Performance improves with increasing L, until it reaches a peak at $L = 70$. When L grows larger than the average segment length in the ground truth (78 beats) the performance decreases.

Enforcing sparsity on W_k and varying c leads to similar results. However, we have found that allowing for varying pattern length has negligible effect on segmentation performance, despite often resulting in qualitatively better patterns. Following this trend, we have also found that $\alpha_h \neq 1$ has minimal effect on performance, so it is set to 1 in the remaining experiments. These results are not surprising since the segmentation is derived from the *combination* of $\mathcal{W}$ and H. Shifting the sparsity from one factor to another should not have significant impact on $\ell_k(t)$.

6.3 Comparison to the state-of-the-art

We compare the proposed segmentation system with other state-of-the-art approaches, including Levy and Sandler's HMM-based segmentation system [2] [5] (QMUL) and a more recent system from Mauch et al [8] based on analysis of self-similarity matrices derived from beat-synchronous chroma. As in Section 6.1, we found that QMUL has optimal PFM when the number of segments is set to 4.

We compare these to the proposed system using fixed rank $K = 4$ (SI-PLCA) and a variant using sparse $\mathbf{z}$ with $\alpha_z = 0.995$ and $K = 15$ (SI-PLCA-α_z). L was fixed at 70 for both systems, and the minimum segment length S was set to 32. Also included is a baseline random segmentation where each frame is given one of 4 randomly selected labels.

The results are shown in Table 1. The system from Mauch et al performs best, followed by SI-PLCA-α_z, SI-PLCA, and QMUL. All systems perform significantly better than the baseline. All of the segmentation systems have roughly comparable pairwise precision and S_u. The differences are primarily in the recall (and S_o) with Mauch et al

[1] http://isophonics.net/content/reference-annotations-beatles

[2] Available: http://vamp-plugins.org/plugin-doc/qm-vamp-plugins.html

System	PFM	PPR	PRR	S_o	S_u
Mauch et al [8]	0.66	0.61	0.77	0.76	0.64
SI-PLCA-α_Z	0.60	0.58	0.68	0.61	0.56
SI-PLCA	0.58	0.60	0.59	0.56	0.59
QMUL [5]	0.54	0.58	0.53	0.50	0.57
Random	0.30	0.36	0.26	0.07	0.24

Table 1. Segmentation performance on the Beatles data set. The number of labels per song was fixed to 4 for SI-PLCA, QMUL, and Random. The average effective ranks for SI-PLCA-α_z and Mauch et al were 3.9 and 5.5, respectively.

outperforming SI-PLCA-α_z by 12% (15%), and SI-PLCA-α_z in turn outperforming QMUL by 15% (11%).

Aside from our algorithm's tendency to over-segment, the most obvious qualitative difference between Mauch et al's and the proposed system lies in more accurate boundary detection in the former system. This is partially a result of the smoothing performed in equation (12) which tends to blur out the segmentation. A more sophisticated set of heuristics for deriving segment labels from the SI-PLCA decomposition might not suffer from this problem.

7. CONCLUSION

We have described an algorithm for identifying repeated patterns in music using shift-invariant probabilistic component analysis and shown how it can be applied to music segmentation. The source code is freely available online. [3]

We demonstrate that the use of simple sparse prior distributions on the SI-PLCA parameters can be used to automatically identify the bases that are most relevant for modeling the data and discard those whose contribution is small. We also demonstrate a similar approach to estimating the optimal length of each basis. The use of these prior distributions enables a more flexible analysis and eliminates the need to specify these parameters exactly in advance.

Although this paper has focused on structure segmentation, the proposed analysis has many other potential applications. For example, basis patterns could be extracted from a collection of pieces to search for common motifs used throughout a corpus of music, e.g. retrieval of cover songs or musical variations. Similarly, Mauch et al demonstrate that chord recognition performance can be improved by pooling data from repeated sections to smooth over variations [8]. In the context of the proposed analysis this amounts to simply analyzing the bases W_k.

Other potential future work includes extracting the hierarchical structure within a piece by repeating the SI-PLCA analysis at different time scales. Finally, we mention that it is possible to extend the SI-PLCA decomposition to be key-invariant by using the 2D extension to SI-PLCA which allow for shifts in pitch class/frequency as well as time [14]. Such an extension would allow for structure segmentation that is insensitive to key modulations within a piece.

[3] http://marl.smusic.nyu.edu/resources/siplca-segmentation/

8. ACKNOWLEDGEMENTS

The authors would like to thank Matthias Mauch for sharing the implementation of the algorithm from [8]. This material is based upon work supported by the NSF (grant IIS-0844654) and by the IMLS (grant LG-06-08-0073-08).

9. REFERENCES

[1] M.A. Bartsch and G.H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Trans. Multimedia*, 7(1):96–104, 2005.

[2] J.P. Bello. Grouping recorded music by structural similarity. In *Proc. ISMIR*, pages 531–536, 2009.

[3] M. Casey and M. Slaney. Song Intersection by Approximate Nearest Neighbor Search. In *Proc. ISMIR*, 2006.

[4] D.P.W. Ellis and G.E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *Proc. ICASSP*, pages IV–1429–1432, 2007.

[5] M. Levy and M. Sandler. Structural Segmentation of Musical Audio by Constrained Clustering. *IEEE Trans. Audio, Speech, and Language Processing*, 16(2), 2008.

[6] H. Lukashevich. Towards quantitative measures of evaluating song segmentation. In *Proc. ISMIR*, 2008.

[7] M. Marolt. A mid-level representation for melody-based retrieval in audio collections. *IEEE Trans. Multimedia*, 10(8):1617–1625, 2008.

[8] M. Mauch, K. C. Noland, and S. Dixon. Using musical structure to enhance automatic chord transcription. In *Proc. ISMIR*, pages 231–236, 2009.

[9] M. Müller, F. Kurth, and M. Clausen. Audio matching via chroma-based statistical features. In *Proc. ISMIR*, pages 288–295, 2005.

[10] A. Ockelford. *Repetition in music: theoretical and metatheoretical perspectives. Volume 13 of Royal Musical Association monographs*. Ashgate Publishing, 2005.

[11] J. Paulus and A. Klapuri. Music structure analysis using a probabilistic fitness measure and a greedy search algorithm. *IEEE Trans. Audio, Speech, and Language Processing*, 17(6):1159–1170, 2009.

[12] G. Peeters and E. Deruty. Is Music Structure Annotation Multi-Dimensional? A Proposal for Robust Local Music Annotation. In *Proc. LSAS*, 2009.

[13] P. Smaragdis and B. Raj. Shift-Invariant Probabilistic Latent Component Analysis. Technical Report TR2007-009, MERL, December 2007.

[14] P. Smaragdis, B. Raj, and M. Shashanka. Sparse and shift-invariant feature extraction from non-negative data. In *Proc. ICASSP*, pages 2069–2072, 2008.

[15] V.Y.F. Tan and C. Févotte. Automatic Relevance Determination in Nonnegative Matrix Factorization. In *Proc. SPARS*, 2009.

LOCATING TUNE CHANGES AND PROVIDING A SEMANTIC LABELLING OF SETS OF IRISH TRADITIONAL TUNES

Cillian Kelly, Mikel Gainza, David Dorran and Eugene Coyle
Audio Research Group
DIT Kevin St.
Dublin 8
Ireland
cillian.kelly@dit.ie

ABSTRACT

An approach is presented which provides the tune change locations within a set of Irish Traditional tunes. Also provided are semantic labels for each part of each tune within the set. A set in Irish Traditional music is a number of individual tunes played segue. Each of the tunes in the set are made up of structural segments called parts. Musical variation is a prominent characteristic of this genre. However, a certain set of notes known as 'set accented tones' are considered impervious to musical variation. Chroma information is extracted at 'set accented tone' locations within the music. The resulting chroma vectors are grouped to represent the parts of the music. The parts are then compared with one another to form a part similarity matrix. Unit kernels which represent the possible structures of an Irish Traditional tune are matched with the part similarity matrix to determine the tune change locations and semantic part labels.

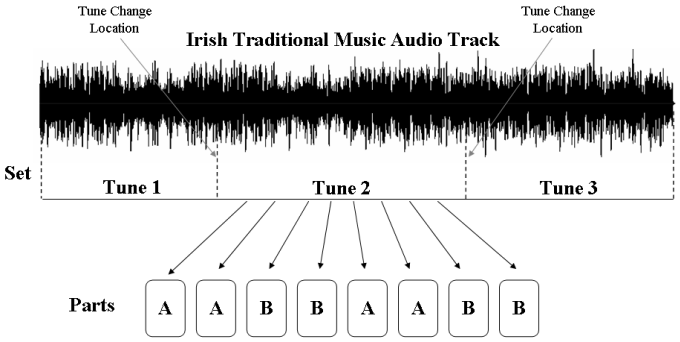


Figure 1. A representation of the structure present within a piece of Irish Traditional music. There are two distinct hierarchical levels of segmentation. The piece of music consists of segments called tunes, and each tune consists of further segments called parts.

1. INTRODUCTION

The approach presented here is specific to Irish Traditional Music. This music type consists of structural segments called 'tunes' which are concatenated to form 'sets'. The tunes are themselves made up of shorter structural segments called 'parts'. The structure of Irish Traditional Music is illustrated in Figure 1. Within this music type performers are encouraged to introduce musical variation. Parts which are notated as equivalent are aurally different due to this musical variation. The approach presented in this paper has two aims. The first is to determine the locations where tune changes occur within 'sets' of Irish Traditional tunes. The second aim is to assign a semantic label to each of the parts of each of the tunes within the music.

The information provided by a structural segmentation can be used for audio browsing. Instead of browsing through the music manually, using the structural segmentation information the user can browse directly to the part of interest within the music. Looping is a further application of the information provided by structural segmentation. Once a user has browsed to the required part within the music, the part can be looped to facilitate repeated playback of a certain segment. The structural segmentation information can provide exact loop points so that the start and end of the selected loop will align rhythmically. This promotes aural learning which is common practice for Irish Traditional musicians. Structural segmentation information can also be used to create an audio thumbnail. For popular music, an audio thumbnail is the most repeated segment, often considered to be the chorus. For Irish Traditional Music however there is no chorus and no segment which repeats more often than others. Using the structural segmentation information a reduced form of the music can be created by discarding repeated sections.

Detailed in this paper is an approach which aims to extract tune change locations within sets of Irish Traditional music. The approach also provides a semantic labelling for each part of the tunes within each set. A set of notes known as 'set accented tones' [17] are utilised which are considered impervious to any musical variation. These notes are extracted and are used to represent the music. Using the harmonic information of these notes, sets are segmented into separate tunes, semantic labels are provided for each part of each resulting tune using a kernel matching technique.

The remainder of this paper is divided as follows: Section 2 details previous relevant approaches toward the structural segmentation of music. In Section 3 an overview of the structure of Irish Traditional Music is provided. The proposed approach toward locating tune changes and providing a semantic labelling of sets of Irish Traditional tunes is detailed in Section 4. Section 5 provides the results of testing the approach on a database of Irish Traditional tunes. The presented approach is compared directly with a previous approach which also attempts to locate the tune

change locations within sets of Irish Traditional tunes. Finally, in Section 6 conclusions based on the results are provided.

2. LITERATURE REVIEW

Approaches which provide a structural segmentation of music aim to search for similarities within the audio signal. The signal is divided into audio frames and certain audio features are extracted for each of the resulting audio frames. Low level audio features such as zero crossings or spectral centroid are extracted and are combined to produce an aggregated description of the audio signal such as in [8, 11, 16, 21]. Mel Frequency Cepstral Co-efficients (*MFCCs*) are a further example of a low level audio feature which is commonly used for approaches which attempt to extract structure from music. In [13] *MFCCs* are extracted for each audio frame and the resulting frames are clustered in order to locate the repeated phrase or chorus within a 'rock' or 'pop' song. Certain heuristics are then used to choose a key phrase which corresponds to the chorus. These low level audio features are indicators of the timbre and loudness of the music. For Irish Traditional music, the timbre and loudness often remain constant throughout an entire musical piece therefore using these features is not suitable for this musical genre.

The audio features used for structural segmentation approaches may also be of a higher level of abstraction, such as pitch [2, 14] or chroma [1, 7] (See Section 4.3). Sections of the audio which share similar audio feature values are grouped together. The resulting groups of frames indicate the overall structure of the music. Extracting pitch or chroma values for every audio frame of an Irish Traditional music piece would include any musical variation present within the piece. This increases the difficulty of determining which structural segments are similar.

In [5], an approach is presented which segments audio using a measure of audio novelty. A Short Time Fourier Transform (*STFT*) is applied to the signal and each resulting frame is compared with every other frame to create an audio similarity matrix. The method of comparison used in [5] is the cosine distance measure. Points of significant musical change are determined by using kernel correlation. In [5], a checkerboard unit kernel is correlated along the diagonal of the resulting self-similarity matrix. Locations which result in a high correlation value are considered to be points of significant musical change which themselves are considered possible structural segmentation boundaries.

Structural segmentation of Irish Traditional music has attracted the attention of researchers. There have been a number of approaches which have attempted to structurally segment this music type [3, 4, 9, 10].

In [9] an approach is presented which aims to segment Irish Traditional tunes into their constituent parts and to provide a semantic labelling for the resulting parts. The 'set accented tones' which are considered impervious to variation are located within the music using a beat tracker. Pitch values are determined at these specific locations using a pitch detector. This results in a selective pitch contour. Melodic patterns are searched for amongst this pitch contour to determine the overall structure of the music. This approach was tested on a database of monophonic pieces of Irish Traditional music. The approach presented in [9] is extended further in [10] where chroma is calculated at 'set accented tone' locations rather than single pitch values. Following this, the chroma vectors are grouped according to heuristics specific to Irish Traditional Music. The resulting groups of chroma vectors correspond to the structural segments of the music and are compared using three different distance measures to determine which of the segments are similar. Extracting chroma rather than single pitch values at 'set accented tone' locations allows the approach in [10] to be applied to polyphonic music rather than only monophonic music as in [9].

In [4] an approach is presented which aims to provide the locations of tune changes (see Section 3) within a set of Irish Traditional tunes. This approach relies on a pre-existing database of transcribed Irish Traditional tunes. The music is transcribed using a pitch detection algorithm and is converted into a format consistent with the ABC music notation language [20]. Sections of the audio are compared with tunes contained within a database of ABC notated tunes using the edit distance algorithm [12]. Once the identity of the tune contained within the section has been determined the version of the tune from the database is compared with every possible section of the transcribed music again using the edit distance. This allows the algorithm to determine where the end of the current tune is located within the music. This process repeats for subsequent tunes within the set until all tunes have been processed.

The approaches presented in both [9] and [10] were tested on a database of pieces of Irish Traditional music containing one tune only. There is no attempt made to structurally segment sets of Irish Traditional tunes. The approach presented in [4] specifically addresses the problem of segmenting a set of Irish Traditional tunes by providing the locations of tune changes within the music. However in [4] the requirement of a pre-existing database of Irish Traditional tunes is a notable limitation. If a tune within a set is not contained within the pre-existing database of tunes, the segmentation of that set will not be successful. The approach presented in Section 4 attempts to provide a semantic labelling of a piece of Irish Traditional music as in [9]. However, unlike [9] the approach presented in Section 4 aims to provide this semantic labelling for sets of tunes rather than single tunes. Section 4 also details a method to locate each tune change within a set of Irish Traditional tunes as in [4]. A method using unit kernels is detailed which overcomes the requirement in [4] of a pre-existing database of transcribed tunes.

3. IRISH TRADITIONAL MUSIC

Irish Traditional Music is comprised of short musical pieces called tunes. Each tune is made up of two or more 'parts' which are notated using upper case letters as can be seen in Figure 1. Although each tune is quite short (a typical two part tune consists of sixteen bars), the parts are repeated to extend the tune and the tune itself can also be repeated in its entirety.

In both studio recordings and live performances of this music type often two or more tunes are concatenated into 'sets' to extend the music even further as shown in Figure 1. For example, a piece of music consisting of a two part tune followed by a three part tune may be played with a musical structure of *AABBAABB/AABBCCAABBCC*.

While two renditions of the same 'A' part may be notated identically, they are rarely performed identically. This is due to the large presence of musical variation inherent with this music type. Embellishments introduced by a musician will render two identical parts as being aurally different.

Despite the considerable presence of musical variation within this genre, for each tune there are a certain set of notes which are left unchanged by the musician. These notes are called the

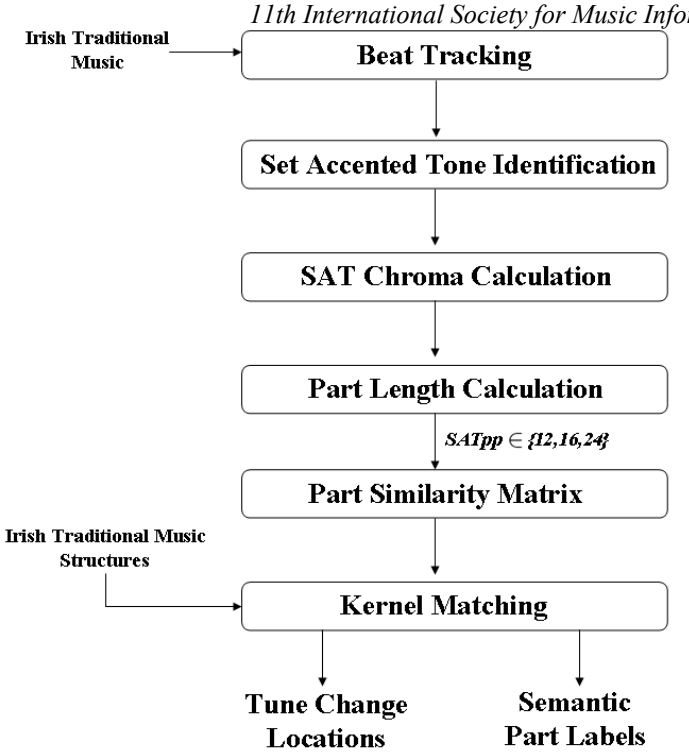


Figure 2. A block diagram of the proposed approach.

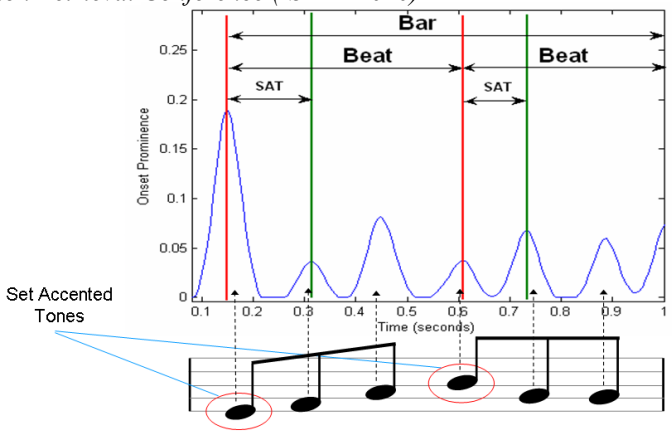


Figure 3. An onset detection function of one bar of Irish Traditional music. Each 'set accented tone' is located between the start of the beat and the next detected onset. For each 'set accented tone' a window is created between these two points of the onset detection function. Chroma is calculated for each resulting window.

'set accented tones' of the tune [9, 17]. In order to avoid the influence which musical variation has on determining which parts are equivalent, the 'set accented tones' are used to characterise the music for this approach.

4. PROPOSED APPROACH

4.1 Overview

An approach toward locating tune changes and providing a semantic labelling of sets of Irish Traditional tunes is detailed in this section. This approach is illustrated in the block diagram in Figure 2. The locations of the 'set accented tones' are determined using a beat tracker. Following this, chroma vectors are calculated at the resulting 'set accented tone' locations and are compared to create a part similarity matrix. Kernel matching is performed on the resulting matrix using unit kernels which represent the various musical structures present within this genre. The kernel matching technique presented here provides a solution to both determining the location of tune changes within a set and also to assigning a semantic label to each resulting part.

4.2 Beat Tracking and Set Accented Tone Identification

To extract chroma at 'set accented tone' locations within the music, the locations of the 'set accented tones' must be defined. Within Irish Traditional Music these notes are considered to be the first note of each beat. Therefore a beat tracker is employed to determine the location of each beat within the music. The beat tracker used for this approach is detailed in [6]. The beat tracker provides the location of each beat of the music along with an onset detection function which provides the locations of each note within the music. To encapsulate each 'set accented tone' a window is created extending from the start of each beat to the next detected onset as illustrated in Figure 3. This maximises the available harmonic information when determining chroma values at each 'set accented tone' location. Following the creation of each 'set accented tone' window, chroma information is extracted at each of these locations.

4.3 Chroma Calculation

To create the part similarity matrix detailed in Section 4.6 chroma must be calculated at each 'set accented tone' location. In [10] results showed that extracting chroma at 'set accented tone' locations provide better structural segmentation results than extracting a single pitch value. As such, for this approach chroma will also be used to represent each 'set accented tone'. Chroma is a spectral representation of music in which frequencies are mapped onto a set of 12 chroma values which correspond to the 12 notes of the equal tempered scale [18].

To calculate the chroma a Harmonic Pitch Class Profile (*HPCP*) approach is employed [19]. For each 'set accented tone' window (the section denoted as 'SAT' in Figure 3), a Short Time Fourier Transform is applied with a frame length of 2048 samples. The local maxima contained within each of the resulting STFT frames are identified using a peak picking algorithm. Following this, the magnitudes of each frequency at each resulting peak location are added to the appropriate chroma bin according to the note of the musical scale to which the frequency most closely corresponds. Only frequencies between $130Hz$ and $3140Hz$ are considered for this approach as $130Hz$ is the frequency of the lowest note on a banjo which is the lowest note likely to be present within an Irish Traditional tune and $3140Hz$ is the third harmonic of the highest note of a standard tin whistle, the highest note likely to be present within an Irish Traditional tune. This gives an appropriately rich description of the frequency content of a given audio frame. This results in a chroma vector of twelve elements each containing the amount of each note which was present in the given 'set accented tone' window.

4.4 Part Length Calculation

Following chroma calculation at each 'set accented tone' location, it is necessary to determine how many 'set accented tones' per part there are in the piece of music. This is required in order to determine the correct groups of chroma vectors to use when

creating the part similarity matrix in Section 4.5. According to the Irish Traditional Music heuristics detailed in [9] there can only be 12, 16 or 24 'set accented tones' per part (*SATpp*) in an Irish Traditional tune. Consequently, each of these three conditions are tested and a confidence score is calculated for each possible *SATpp* value. Following this, the chroma vectors representing the 'set accented tones' are divided into groups according to the *SATpp* value currently being tested. For example, if the current *SATpp* value is equal to 12, the chroma vectors are divided into groups of 12. The resulting groups of chroma vectors now represent potential parts of a tune. Following this, each potential part is compared with every other potential part and a confidence score is calculated based on these part comparisons. The *SATpp* value which results in the highest confidence score is the value used when creating the part similarity matrix in Section 4.5.

Individual chroma vectors are compared using the Euclidean Distance formula given in Equation (1).

$$D(v_1, v_2) = \sqrt{(\sum_{i=1}^{12}(v_1(i) - v_2(i))^2)} \quad (1)$$

where v_1 and v_2 are the two chroma vectors being compared.

Entire parts are compared with one another using Equation (2). The resulting value *S* is low if the two parts being compared are similar, therefore *S* is a measure of the dis-similarity between two parts.

$$S = \frac{\sum_{n=0}^{N-1} D(v_1(n), v_2(n))}{N} \quad (2)$$

where *N* is equal to the *SATpp* value currently being tested.

Finally, a confidence value *C* for the *SATpp* value being tested is calculated according to Equation (3).

$$C = 1/\frac{\sum_{m=0}^{M-1} S_{SATpp}}{M} \quad (3)$$

where *M* is equal to the total number of part comparisons. The *SATpp* value which results in the greatest confidence value *C* is the value used to create the part similarity matrix in Section 4.5.

4.5 Part Similarity Matrix

As detailed in Section 4.4, once the number of 'set accented tones' per part has been determined, the part similarity matrix is created according to this *SATpp* value. The values of *S* (calculated in Section 4.4 using Equation (2)) associated with this particular *SATpp* value are used to create the part similarity matrix. These values of *S* indicate the similarity of each part of length *SATpp* with every other part of length *SATpp* within the music.

Positioning these values into a matrix results in a part similarity matrix of size *P* by *P* where *P* is equal to the total number of parts within the music. An example of a part similarity matrix is shown in Figure 5. The part similarity matrix is used along with unit kernels to determine the structure of the music as described in Section 4.6.

4.6 Kernel Matching

The following section details how unit kernels are matched with the part similarity matrix created in Section 4.5. Firstly, the kernels that are used for matching are described, along with justifications for using these particular kernels. The process of how the unit kernels are matched with the part similarity matrix is then detailed.

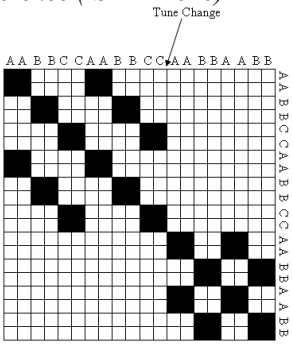


Figure 4. A unit kernel representing the structure of two tunes. The first tune represented has a structure of *AABBCCAABBCC* the second tune represented has a structure of *AABBAABB*. Black represents similar parts and white represents dis-similar parts.

Kernel matching relies on the availability of pre-existing unit kernels which each represent a specific musical structure. A unit kernel is a matrix consisting of ones and zeros which represent the pattern of a musical structure. A total of 24 unit kernels which represent a single tune are used here. Kernels representing the structure of more than a single tune are created by combining the kernels which represent a single tune. An example of a two-tune kernel is shown in Figure 4. There are 24 kernels used to represent the possible structures of one tune. As such combining each one-tune kernel with every other one-tune kernel results in 576 possible combinations for a two-tune kernel. The unit kernels represent the musical structures which are most common within Irish Traditional Music.

The kernels that are used limit the number of possible parts per tune to four. According to [15], tunes containing two, three and four parts make up 97% of the volume of tunes within this genre. Both one-tune kernels *and* two-tune kernels are utilised to give a total number of kernels of 600. The unit kernels are correlated with sections of the part similarity matrix as illustrated in Figure 5. The kernel which yields the highest matching value is the kernel which represents the most likely musical structure present within the given section of the part similarity matrix. The following steps describe the kernel matching technique:

1. At location *(i,i)* of the part similarity matrix, each *K x K* unit kernel is matched with a *K x K* section of the part similarity matrix using inner matrix multiplication. For the first iteration only, *i = 1*.
2. The kernel which results in the highest match value is deemed to represent the structure of that section of the part similarity matrix.
3. The value *i* is updated to be *(i + K)* where *K* is equal to the length of the kernel in step 2.
4. Steps 1-3 are repeated until the entire part similarity matrix has been processed.

The outcome of this kernel matching as outlined in Figure 2 is the location of each tune change within the set of Irish Traditional tunes along with a semantic label for each resulting part.

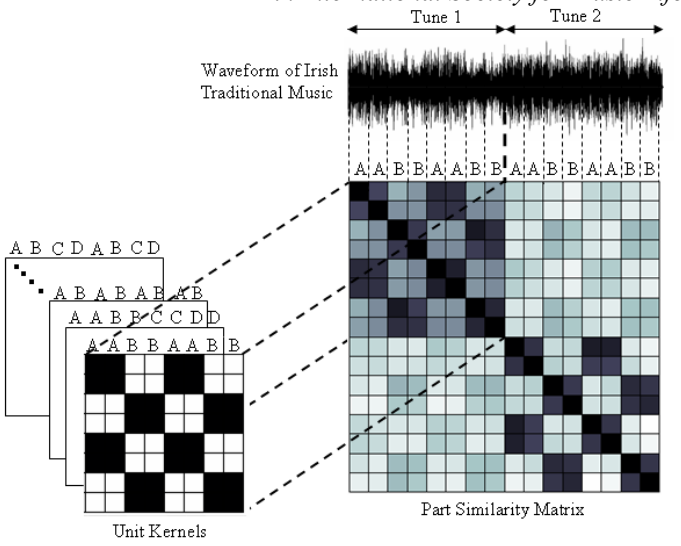


Figure 5. The section of a part similarity matrix with which a unit kernel is correlated. Each cell of the part similarity matrix corresponds to a part of an Irish Traditional tune. This section of the part similarity matrix is correlated with each unit kernel. The unit kernel which results in the highest matching value corresponds to the structure of this section of the part similarity matrix.

The semantic part labels are represented by the concatenation of the descriptions of the unit kernels which were matched to each tune. The locations of each tune change are determined by the beat locations of the start of each successfully matched unit kernel. Results for the approach which has been presented here are detailed in Section 5.

5. RESULTS

The evaluation of this approach was carried out on a hand annotated database of 30 sets of Irish Traditional music. These 30 musical pieces consist of 75 separate Irish traditional tunes with 34 tune changes and a total of 589 parts. The results of detecting the location of tune changes and determining labels for each structural segment were calculated separately. A tune change location was considered correct if the automatically detected tune change locations were within 1 second of the hand annotated tune change locations. Table 1 details the results of labelling the parts of the music and also details the results of locating tune changes within the music. In Table 1, *N* is equal to the number of annotations, *GP* is equal to Good Positives, *FP* is equal to False Positives and *FN* is equal to False Negatives. Results were calculated using three different measures, precision, recall and accuracy. These three measures are defined by Equations (4), (5) and (6).

$$Precision = \frac{GP}{GP + FP} \tag{4}$$

$$Recall = \frac{GP}{GP + FN} \tag{5}$$

$$Accuracy = \frac{N - FP - FN}{N} \tag{6}$$

The results show that the approach can label the parts contained in a piece of Irish Traditional Music with an accuracy of 86% along with a precision value of 90% and a recall value of 94%. Additionally, this approach can correctly identify the location of tune changes within a set of Irish Traditional tunes with an accuracy of 62% along with a precision value of 77% and a recall value of 80%.

	N	*GP*	*FP*	*FN*	*Prec*	*Rec*	*Acc*
Part Labels	589	479	51	31	90%	94%	86%
Tune Change Locations	34	24	7	6	77%	80%	62%

Table 1. Results of the part labelling and locating tune changes produced by the presented approach.

Tolerance	*Prec*	*Rec*
1 second	69.7%	64.79%
2 seconds	87%	81%

Table 2. Results of the approach presented in [3] and [4] toward locating tune changes within a set of Irish Traditional tunes.

The accuracy value is higher for labelling parts than for detecting tune changes. This is because even when a tune change is not accurately detected, the algorithm may still correctly identify subsequent parts contained within the music. This is due to many kernels having common part locations. The high recall values should be noted, this indicates that the algorithm detects most tune changes present in the music.

An approach is presented in [4] which also attempts to calculate the tune change locations within sets of Irish Traditional music. In [4] a tune change location is considered to be correct if the automatically generated tune change locations are within 2 seconds of the equivalent hand annotated tune change locations. The approach presented in [4] is detailed further in [3] where results are also provided for a tolerance of 1 second. The results of detecting tune change locations as detailed in [3] and [4] can be seen in Table 2 for a tolerance window of both 1 second and 2 seconds. The results detailed in Table 1 were obtained from testing on the same database used in [3].

When detecting tune changes within a set the approach detailed in [3] claims a precision value of 69.7% and a recall value of 64.79% for a tolerance of 1 second. The approach presented in Section 4 has improved these values by 7.3% and 15.21% respectively and also does not require the database of Irish Traditional tunes which is utilised in [3]. In [3], to correctly detect the tune changes within a set of Irish Traditional tunes, each tune in the set must also be in a pre-existing database of tunes.

6. CONCLUSIONS

This paper presented an approach toward locating tune changes and providing a semantic labelling of sets of Irish Traditional tunes. This music type consists of sets of tunes, the tunes themselves are made up of parts. This approach utilised certain notes within the music which remain constant despite the presence of musical variation. Chroma was extracted at these specific note locations and was compared to create a part similarity matrix. Unit kernels representing common structures present within Irish Traditional Music were then matched with sections of the part similarity matrix. The unit kernel which resulted in the high-

est match value corresponds to the structure of the music at the given location within the part similarity matrix.

Using chroma at the 'set accented tone' locations within the music significantly reduces the amount of data required to produce a structural segmentation. This reduced representation of the music also filters out musical variation which can affect the process of determining which parts are equivalent. The approach presented here was tested on a database of 30 sets of Irish Traditional tunes. The results of the approach presented here were compared with a similar approach toward detecting tune change locations within a set of Irish Traditional tunes by testing on the same database of Irish Traditional music. When a tolerance of 1 second between automatically detected tune changes and hand annotated tune changes is used, the approach presented here performs significantly better than a previous approach toward the same goal. For this approach, increasing the tolerance window will not result in an increase in performance as tune change location times are calculated using part locations and are not calculated on a scale which is sensitive to increments of less than the length of a part.

The approach presented in this paper relies on correctly calculating the amount of 'set accented tones' per part (*SATpp*). If this value is calculated incorrectly, the resulting part similarity matrix will not accurately reflect the parts which are present within the set of Irish Traditional tunes. Consequently, it would not be possible to identify the correct tune change locations or determine the correct semantic part labelling. This approach also relies on the accuracy of the beat tracker in order to correctly identify the 'set accented tone' locations.

Future work will aim to combine the approach presented here with the approach presented in [4] to identify tune change locations within a set. Firstly, the method detailed in [4] would be used to determine the tune change locations. If there are tunes present within a set that are not present within the pre-existing database used in [4] the tune change locations cannot be calculated using this method. In this case, the approach presented in Section 4 would be used as an alternative, as there is no pre-existing knowledge required of the particular tunes within the set for this approach.

7. REFERENCES

[1] Mark A. Bartsch and Gregory H. Wakefield. To catch a chorus: using chroma-based representations for audio thumbnailing. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, 2001.

[2] Roger B. Dannenberg and Ning Hu. *Discovering Musical Structure in Audio Recordings*. Lecture Notes in Computer Science. Springer Berlin, 2002.

[3] Bryan Duggan. *Machine Annotation of Traditional Irish Dance Music*. PhD thesis, Dublin Institute of Technology, 2009.

[4] Bryan Duggan, Brendan O'Shea, Mikel Gainza, and Padraig Cunningham. Machine annotation of sets of traditional irish dance tunes. In *International Conference on Music Information Retrieval*, Philadelphia, PA, USA, 2008.

[5] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *IEEE Intl Conf. on Multimedia and Expo*, New York, 2000.

[6] Mikel Gainza. On the use of a dynamic hybrid tempo detection model for beat tracking. In *IEEE International Conference on Multimedia and Expo*, Singapore, 2010.

[7] Masataka Goto. A chorus-section detecting method for musical audio signals. In *IEEE Conference on Acoustics, Speech, and Signal Processing*, Hong Kong, 2003.

[8] Min-Hong Jian, Chia Han Lin, and Arbee L.P. Chen. Perceptual analysis for music segmentation. In *Storage and Retrieval Methods and Applications for Multimedia*, San Jose, California, USA, 2003.

[9] Cillian Kelly, Mikel Gainza, David Dorran, and Eugene Coyle. Structural segmentation of music using set accented tones. In *124th Audio Engineering Society Convention*, Amsterdam, The Netherlands, 2008.

[10] Cillian Kelly, Mikel Gainza, David Dorran, and Eugene Coyle. Structural segmentation of irish traditional music using chroma at set accented tone locations. In *127th Audio Engineering Society Convention*, New York, New York, U.S.A., 2009.

[11] S. Lefevre, B. Maillard, and N. Vincent. A two level classifier process for audio segmentation. In *16th International Conference on Pattern Recognition*, Washington DC, USA, 2002.

[12] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 1966.

[13] Beth Logan and Stephen Chu. Music summarization using key phrases. In *International Conference on Audio Speech and Signal Processing*, Istanbul, Turkey, 2000.

[14] Benoit Meudic. Musical pattern extraction: from reptition to musical structure. In *Computer Music Modelling and Retrieval*, Montpellier, France, 2003.

[15] Donncha Seán Ó'Maidín. *A Programmer's Environment for Music Analysis*. PhD thesis, University College Cork, 1995.

[16] Bee Suan Ong and Perfecto Herrera. Semantic segmentation of music audio contents. In *International Computer Music Conference*, Barcelona, Spain, 2005.

[17] Mícháel Ó'Súilleabháin. *Innovation and Tradition in the Music of Tommie Potts*. PhD thesis, Queen's University, 1987.

[18] Steffen Pauws. Musical key extraction from audio. In *International Conference on Music Information Retrievals*, Barcelona, Spain, 2004.

[19] Joan Serra, Emilia Gomez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 2008.

[20] Chris Walshaw. Abc notation - an introduction, http://abcnotation.com/. Accessed March 2010.

[21] Yibin Zhang and Jie Zhou. Audio segmentation based on multi-scale audio classification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Quebec, Canada, 2004.

APPROXIMATE NOTE TRANSCRIPTION FOR THE IMPROVED IDENTIFICATION OF DIFFICULT CHORDS

Matthias Mauch and Simon Dixon
Queen Mary University of London, Centre for Digital Music
{matthias.mauch, simon.dixon}@elec.qmul.ac.uk

ABSTRACT

The automatic detection and transcription of musical chords from audio is an established music computing task. The choice of chord profiles and higher-level time-series modelling have received a lot of attention, resulting in methods with an overall performance of more than 70% in the MIREX Chord Detection task 2009. Research on the front end of chord transcription algorithms has often concentrated on finding good chord templates to fit the chroma features. In this paper we reverse this approach and seek to find chroma features that are more suitable for usage in a musically-motivated model. We do so by performing a prior approximate transcription using an existing technique to solve non-negative least squares problems (NNLS). The resulting NNLS chroma features are tested by using them as an input to an existing state-of-the-art high-level model for chord transcription. We achieve very good results of 80% accuracy using the song collection and metric of the 2009 MIREX Chord Detection tasks. This is a significant increase over the top result (74%) in MIREX 2009. The nature of some chords makes their identification particularly susceptible to confusion between fundamental frequency and partials. We show that the recognition of these diffcult chords in particular is substantially improved by the prior approximate transcription using NNLS.

Keywords: chromagram, chord extraction, chord detection, transcription, non-negative least squares (NNLS).

1. INTRODUCTION

Chords are not only of theoretical interest for the understanding of Western music. Their practical relevance lies in the fact that they can be used for music classification, indexing and retrieval [2] and also directly as playing instructions for jazz and pop musicians. Automatic chord transcription from audio has been the subject of tens of research papers over the past few years. The methods usually rely on the low-level feature called chroma, which is a mapping of the spectrum to the twelve pitch classes C,...,B, in which the pitch height information is discarded. Nevertheless, this feature is often sufficient to recognise chords because chord labels themselves remain the same whatever octave the constituent notes are played in. An exception is the lowest note in a chord, the bass note, whose identity is indeed notated in chord labels. Some research papers have taken advantage of the additional information conveyed by the bass note by introducing special bass chromagrams [18, 12] or prior bass note detection [21].

There is much scope in developing musical models to infer the most likely chord sequence from the chroma features. Many approaches use models of metric position [16], the musical key [8, 21], or combinations thereof [12], as well as musical structure [13], to increase the accuracy of the chord transcription. Although in this work we will also use such a high-level model, our main concern will be the low-level front end.

Many previous approaches to chord transcription have focussed on finding a set of chord profiles, each chord profile being a certain chroma pattern that describes best the chroma vectors arising while the chord is played. It usually includes the imperfections introduced into the chromagram by the upper partials of played notes. The shape of each pattern is either theoretically motivated (e.g. [15]) or learned, usually using (semi-) supervised learning (e.g. [8, 9]). A few approaches to key and chord recognition also emphasise the fundamental frequency component before producing the chromagrams [5, 18] or use a greedy transcription step to improve the correlation of the chroma with true fundamental frequencies [19]. Emphasising fundamental frequencies before mapping the spectrum to chroma is preferable because here all spectral information can be used to determine the fundamental frequencies – *before* discarding the octave information.

However, in order to determine the note activation, the mentioned approaches use relatively simple one-step transforms, a basic form of approximate transcription. A different class of approaches to approximate transcription assumes a more realistic linear generative model in which the spectrum (or a log-frequency spectrum) Y is considered to be approximately represented by a linear combination of note profiles in a dictionary matrix E, weighted by the activation vector x, with $x \geq 0$:

$$Y \approx Ex \tag{1}$$

This model conforms with our physical understanding of how amplitudes of simultaneously played sounds add up [1] .

[1] Like the one-step transforms, the model assumes the absence of si-

Approaches to finding the activation vector x in (1) differ from the one-step transforms in that they involve iterative re-weighting of the note activation values [1]. To our knowledge, such a procedure has not been used to generate chromagrams or otherwise conduct further automatic harmony analysis. Unlike traditional transcription approaches, we are not directly interested in note events, and the sparsity constraints required in [1] need not be taken into account. This allows us to use a standard procedure called non-negative least squares (NNLS), as will be explained in Section 2.

The motivation for this is the observation that the partials of the notes played in chords compromise the correct recognition of chords. The bass note in particular usually has overtones at frequencies where other notes have their fundamental frequencies. Interestingly, for the most common chord type in Western music, the major chord (in root position), this does not pose a serious problem, because the frequencies of the first six partials of the bass note coincide with the chord notes: for example, a C major chord (consisting of C, E and G) in root position has the bass note C, whose fist six partials coincide with frequencies at pitches C, C, G, C, E, G. Hence, using a simple spectral mapping works well for major chords. But even just considering the first inversion of the C major chord (which means that now E is the the bass note), leads to a dramatically different situation: the bass note's first six partials coincide with E, E, B, E, G♯, B – of which B and G♯ are definitely not part of the C major triad. Of course, the problem does not only apply to the bass note, but to all chord notes [2].

This is a problem that can be eliminated by a perfect prior transcription because no partials would interfere with the signal. Section 2 focusses mainly on describing our approach to an approximate transcription using NNLS, and also gives an outline of the high-level model we use. In Section 3 we demonstrate that the problem does indeed exist and show that the transcription capabilities of the NNLS algorithm can improve the recognition of the affected chords. We give a brief discussion of more general implications and future work in Section 4, before presenting our conclusions in Section 5.

2. METHOD

This section is concerned with the technical details of our method. Most importantly, we propose the use of NNLS-based approximate note transcription, prior to the chroma mapping, for improved chord recognition. We call the resulting chroma feature *NNLS chroma*. To obtain these chroma representations, we first calculate a log-frequency spectrogram (Subsection 2.2), pre-process it (Subsection 2.3) and perform approximate transcription using the NNLS algorithm (Subsection 2.4). This transcription is then wrapped to chromagrams and beat-synchronised (Section 2.5). Firstly, however, let us briefly consider the high-level musical model which takes as input the chroma features, and which we use to test the effect of different chromagrams on chord transcription accuracy.

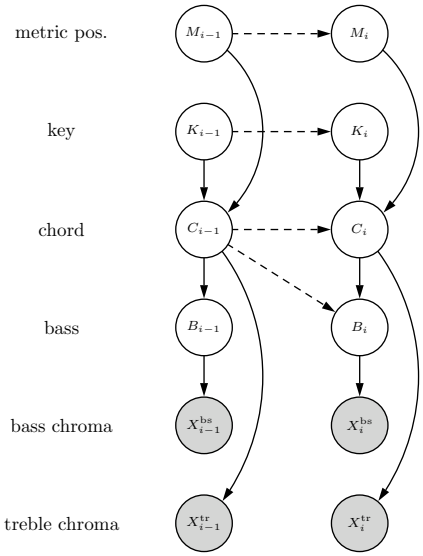


Figure 1: High-level dynamic Bayesian network, represented as two slices corresponding to two generic consecutive beats. Random variables are shown as nodes, of which those shaded grey are observed, and the arrows represent direct dependencies (inter-slice arrows are dashed).

2.1 High-level Probabilistic Model

We use a modification of a dynamic Bayesian network (DBN) for chord recognition proposed in [10], which integrates in a single probabilistic model the hidden states of metric position, key, chord, and bass note, as well as two observed variables: chroma and bass chroma. It is an expert model whose structure is motivated by musical considerations; for example, it enables to model the tendency of the bass note to be present on the first beat of a chord, and the tendency of the chord to change on a strong beat. The chord node distinguishes 121 different states: 12 for each of 10 chord types (major, minor, major in first inversion, major in second inversion, major 6th, dominant 7th, major 7th, minor 7th, diminished and augmented) and one "no chord" state. With respect to the original method, we have made some slight changes in the *no chord* model and the metric position model [3]. The DBN is implemented using Murphy's BNT Toolbox [14], and we infer the jointly most likely state sequence in the Viterbi sense.

2.2 Log-frequency Spectrum

We use the discrete Fourier transform with a frame length of 4096 samples on audio downsampled to 11025 Hz. The DFT length is the shortest that can resolve a full tone in the bass region around MIDI note 44 [4], while using a Ham-

nusoid cancellation.

[2] For example, a major third will create some energy at the major 7th through its third partial.

[3] The *no chord* model has been modified by halving the means of the multivariate Gaussian used to model its chroma, and the metric position model is now fully connected, i.e. the same low probability of 0.0167 is assigned to missing 1, 2 or three beats.

[4] Smaller musical intervals in the bass region occur extremely rarely.

ming window. We generate a spectrogram with a hop size of 2048 frames (≈0.05s).

We map the magnitude spectrum onto bins whose centres are linearly-spaced in log frequency, i.e. they correspond to pitch (e.g. [17]), with bins spaced a third of a semitone apart. The mapping is effectuated using cosine interpolation on both the linear and logarithmic scales: first, the DFT spectrum is upsampled to a highly oversampled frequency representation, and then this intermediate representation is mapped to the desired log-frequency representation. The two operations can be performed as a single matrix multiplication. This calculation is done separately on all frames of a spectrogram, yielding a log-frequency spectrogram $Y = (Y_{k,m})$.

Assuming equal temperament, the global tuning of the piece is now estimated from the spectrogram. Rather than adjusting the dictionary matrix we then update the log-frequency spectrogram via linear interpolation, such that the centre bin of every semitone corresponds to the correct frequency with respect to the estimated tuning [10]. The updated log-frequency spectrogram Y has 256 $1/3$-semitone bins (about 7 octaves), and is hence much smaller than the original spectrogram. The reduced size enables us to model it efficiently as a sum of idealised notes, as will be explained in Subsection 2.4.

2.3 Pre-processing the Log-frequency Spectrum

We use three different kinds of pre-processing on the log-frequency spectrum:

o : original – no pre-processing,

sub : subtraction of the background spectrum [3], and

std : standardisation: subtraction of the background spectrum and division by the running standard deviation.

To estimate the background spectrum we use the running mean $\mu_{k,m}$, which is the mean of a Hamming-windowed, octave-wide neighbourhood (from bin $k-18$ to $k+18$). The values at the edges of the spectrogram, where the full window is not available, are set to the value at the closest bin that is covered. Then, $\mu_{k,m}$ is subtracted from $Y_{k,m}$, and negative values are discarded (method *sub*). Additionally dividing by the respective running standard deviation $\sigma_{k,m}$, leads to a running standardisation (method *std*). This is similar to spectral whitening (e.g. [6]) and serves to discard timbre information. The resulting log-frequency spectrum of both pre-processing methods can be calculated as

$$Y^{\rho}_{k,m} = \begin{cases} \frac{Y_{k,m}-\mu_{k,m}}{\sigma^{\rho}_{k,m}} & \text{if } Y_{k,m}-\mu_{k,m} > 0 \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where $\rho = 0$ or $\rho = 1$ for the cases *sub* and *std*, respectively.

2.4 Note Dictionary and Non-Negative Least Squares

In order to decompose a log-frequency spectral frame into the notes it has been generated from, we need two basic ingredients: a note dictionary E, describing the assumed profile of (idealised) notes, and an inference procedure to determine the note activation patterns that result in the closest match to the spectral frame.

We generate a dictionary of idealised note profiles in the log-frequency domain using a model with geometrically declining overtone amplitudes [5],

$$a_k = s^{k-1} \quad (3)$$

where the parameter $s \in (0,1)$ influences the spectral shape: the smaller the value of s, the weaker the higher partials. Gomez [5] favours the parameter $s = 0.6$ for her chroma generation, in [13] $s = 0.9$ was used. We will test both possibilities, and add a third possibility, where s is linearly spaced (LS) between $s = 0.9$ for the lowest note and $s = 0.6$ for the highest note. This is motivated by the fact that resonant frequencies of musical instruments are fixed, and hence partials of notes with higher fundamental frequency are less likely to correspond to a resonance. In each of the three cases, we create tone patterns over seven octaves, with twelve tones per octave: a set of 84 tone profiles. The fundamental frequencies of these tones range from A0 (at 27.5 Hz) to G♯6 (at approximately 3322 Hz). Every note profile is normalised such that the sum over all the bins equals unity. Together they form a matrix E, in which every column corresponds to one tone.

We assume now that—like in Eqn. (1)—the individual frames of the log-frequency spectrogram Y are generated approximately as a linear combination $Y_{\cdot,m} \approx Ex$ of the 84 tone profiles. The problem is to find a tone activation pattern x that minimises the Euclidian distance

$$||Y_{\cdot,m} - Ex|| \quad (4)$$

between the linear combination and the data, with the constraint $x \geq 0$, i.e. all activations must be non-negative. This is a well-known mathematical problem called the non-negative least squares (NNLS) problem. Lawson and Hanson [7] have proposed an algorithm to find a solution, and since (in our case) the matrix E has full rank and more rows than columns, the solution is also unique. We use MATLAB's implementation of this algorithm. Again, all frames are processed separately, and we finally obtain an NNLS transcription spectrum S in which every column corresponds to one audio frame, and every row to one semitone. Alternatively, we can choose to omit the approximate transcription step and copy the centre bin of every semitone in Y to the corresponding bin of S [17].

2.5 Chroma, Bass Chroma and Beat-synchronisation

The DBN we use to estimate the chord sequence requires two different kinds of chromagram: one general-purpose chromagram that covers all pitches, and one bass-specific chromagram that is restricted to the lower frequencies. We emphasise the respective regions of the semitone spectrum by multiplying by the pitch-domain windows shown in Figure 2, and then map to the twelve pitch classes by summing the values of the respective pitches.

log-freq. spectrum	no NNLS	NNLS $s = 0.6$	NNLS $s = 0.9$	NNLS LS
o	38.6	43.9	43.1	47.5
sub	74.5	74.8	71.5	73.8
std	79.0	**80.0**	76.5	78.6

(a) MIREX metric – correct overlap in %

log-freq. spectrum	no NNLS	NNLS $s = 0.6$	NNLS $s = 0.9$	NNLS LS
o	31.0	35.1	33.9	37.4
sub	58.1	58.2	56.1	57.3
std	61.3	62.7	62.0	**63.3**

(b) metric using all chord types – correct overlap in %

Table 1: Results of the twelve methods in terms of the percentage of correct overlap. Table (a) shows the MIREX metric, which distinguishes only 24 chords and a "no chord" state, Table (b) is shows a finer metric that distinguishes 120 chords and a "no chord" state.

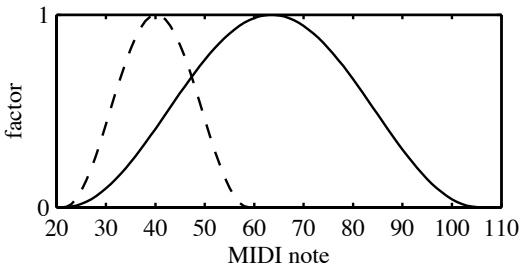


Figure 2: Profiles applied to the log-frequency spectrum before the mapping to the main chroma (solid) and bass chroma (dashed).

Beat-synchronisation is the process of summarising frame-wise features that occur between two beats. We use the beat-tracking algorithm developed by Davies [4], and obtain a single chroma vector for each beat by taking the median (in the time direction) over all the chroma frames between two consecutive beat times. This procedure is applied to both chromagrams, for details refer to [10]. Finally, each beat-synchronous chroma vector is normalised by dividing it by its maximum norm. The chromagrams can now be used as observations in the DBN described in Section 2.1.

3. EXPERIMENTS AND RESULTS

Our test data collection consists of the 210 songs used in the 2009 MIREX Chord Detection task, together with the corresponding ground truth annotations [11]. We run 12 experiments varying two parameters: the preprocessing type (*o*, *sub* or *std*, see Section 2.3), and the kind of NNLS setup used ($s = 0.6$, $s = 0.9$, LS, or direct chroma mapping, see Section 2.4).

3.1 Overall Accuracy

The overall accuracy of the 12 methods in terms of the percentage of correct overlap

$$\frac{\text{duration of correctly annotated chords}}{\text{total duration}} \times 100\%$$

is displayed in Table 1: Table 1a shows results using the MIREX metric which distinguishes only two chord types and the "no chord" label, and 1b shows results using a finer evaluation metric that distinguishes all 121 chord states that the DBN can model; see also [10, Chapter 4].

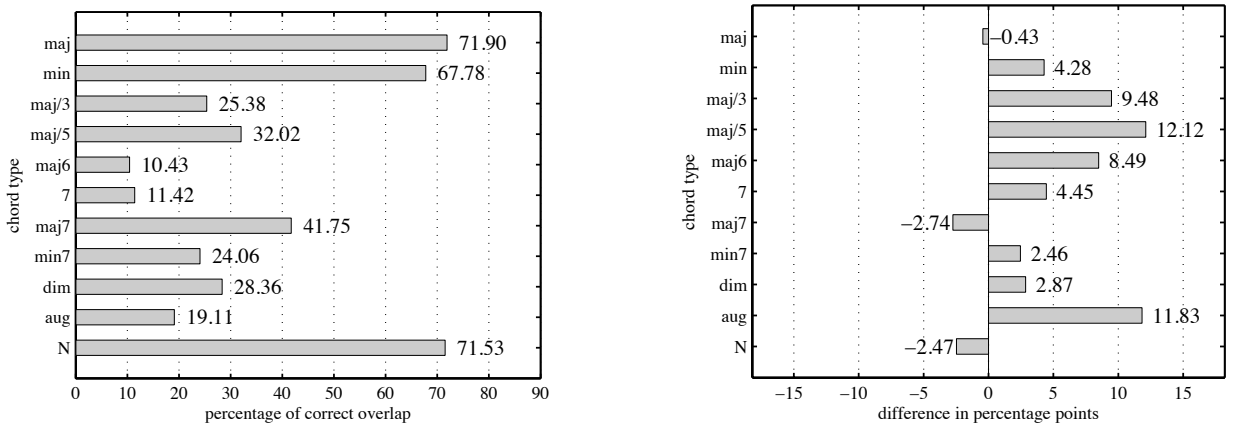
When considering the MIREX metric in Table 1a it is immediately clear that one of the decisive factors has been the spectral standardisation: all four *std* methods clearly outperform the respective analogues with *sub* preprocessing or no preprocessing. We performed a 95% Friedman multiple comparison analysis on the song-wise results of the *std* methods: except for the difference between no NNLS and LS all differences are significant, and in particular the NNLS method using $s = 0.6$ significantly outperforms all other methods, achieving 80% accuracy. With a p-value of 10^{-10} in the Friedman test, this is also a highly significant increase of nearly 6 percentage points over the 74% accuracy achieved by the highest scoring method [20] in the 2009 MIREX tasks.

In Table 1b the results are naturally lower, because a much finer metric is used. Again, the *std* variants perform best, but this time the NNLS chroma with the linearly spaced s has the edge, with 63% accuracy. (Note that this is still higher than three of the scores in the MIREX task evaluated with the MIREX metric.) According to a 95% Friedman multiple comparison test, the difference between the methods *std*-LS and *std*-0.6 is not significant. However, both perform significantly better than the method without NNLS for this evaluation metric which more strongly emphasises the correct transcription of difficult chords.

The reason for the very low performance of the *o* methods without preprocessing is the updated model of the "no chord" state in the DBN. As a result, many chords in noisier songs are transcribed as "no chord". However, this problem does not arise in the *sub* and *std* methods, where the removal of the background spectrum suppresses the noise. In these methods the new, more sensitive "no chord" model enables very good "no chord" detection, as we will see in the following subsection.

3.2 Performance of Individual Chords

Recall that our main goal, as stated in the introduction, is to show an improvement in those chords that have the problem of bass-note induced partials whose frequencies do not coincide with those of the chord notes. Since these chords are rare compared to the most frequent chord type, major, differences in the mean accuracy are relatively small (compare the *std* methods with NNLS, $s = 0.6$, and without in Table 1a). For a good transcription, however, all



(a) *std* method without NNLS

(b) improvement of *std* with NNLS chroma ($s = 0.6$) over baseline *std* method.

Figure 3: Percentage of correct overlap of individual chord types.

chords are important, and not only those that are most frequently used. First of all we want to show that the problem does indeed exist and is likely to be attributed to the presence of harmonics. As a baseline method we choose the best-performing method without NNLS chroma (*std*), whose performance on individual chords is illustrated in Figure 3a. As expected, it performs best on major chords, achieving a recognition rate of 72%. This is rivalled only by the "no chord" label `N` (also 72%), and the minor chords (68%). All other chords perform considerably worse. This difference in performance may of course have reasons other than the bass note harmonics, be it an implicit bias in the model towards simpler chords, or differences in usage between chords. There is, however, compelling evidence for attributing lower performance to the bass note partials, and it can be found in the chords that differ from the major chord in only one detail: the bass note. These are the major chord inversions (denoted `maj/3`, and `maj/5`): while the chord model remains the same otherwise, performance for these chords is around 40 percentage points worse than for the same chord type in root position.

To find out whether the NNLS methods suffer less from this phenomenon, we compare the baseline method discussed above to an NNLS method (*std*, with the chord dictionary parameter $s = 0.6$). The results of the comparison between the baseline method and this NNLS method can be seen in Figure 3b. Recognition rates for almost all chords have improved by a large margin, and we would like to highlight the fact that the recognition of major chords in second inversion (`maj/5`) has increased by 12 percentage points. Other substantial improvements can be found for augmented chords (also 12 percentage points), and major chords in first inversion (9 percentage points). These are all chords in which even the third harmonic of the bass note does not coincide with the chord notes (the first two always do), which further assures us that our hypothesis was correct. Note that, conversely, the recognition of major chords has remained almost stable, and only two chords, major 7th and the "no chord" label, show a slight performance decrease (less than 3 percentage points).

4. DISCUSSION

While the better performance of the difficult chords is easily explainable by approximate transcription, there is some scope in researching why the major 7th chord performed slightly worse in the method using NNLS chroma. Our hypothesis is that the recognition of the major 7th chord actually benefits from the presence of partials: not only does the bass note emphasise the chord notes (as it does in the plain major chord), but the seventh itself is also emphasised by the third harmonic of the third; e.g. in a C major 7th chord (C, E, G, B), the E's third harmonic would emphasise the B. In future work, detailed analyses of which major 7th chords' transcriptions change due to approximate transcription could reveal whether this hypothesis is true.

Our findings provide evidence to support the intuition that the information which is lost by mapping the spectrum to a chroma vector cannot be recovered completely: therefore it seems vital to perform note transcription or calculate a note activation pattern *before* mapping the spectrum to a chroma representation (as we did in this paper) or directly use spectral features as the input to higher-level models, which ultimately may be the more principled solution.

Of course, our approximate NNLS transcription is only one way of approaching the problem. However, if an approximate transcription is known, then chord models and higher-level musical models can be built that do not mix the physical properties of the signal ("spectrum given a note") and the musical properties ("note given a musical context"). Since the components of such models will represent something that actually exists, we expect that training them will lead to a better fit and eventually to better performance.

5. CONCLUSIONS

We have presented a new chroma extraction method using a non-negative least squares (NNLS) algorithm for prior approximate note transcription. Twelve different chroma methods were tested for chord transcription accuracy on a

standard corpus of popular music, using an existing high-level probabilistic model. The NNLS chroma features achieved top results of 80% accuracy that significantly exceed the state of the art by a large margin.

We have shown that the positive influence of the approximate transcription is particularly strong on chords whose harmonic structure causes ambiguities, and whose identification is therefore difficult in approaches without prior approximate transcription. The identification of these difficult chord types was substantially increased by up to twelve percentage points in the methods using NNLS transcription.

6. ACKNOWLEDGEMENTS

This work was funded by the UK Engineering and Physical Sciences Research Council, grant EP/E017614/1.

7. REFERENCES

[1] S. A. Abdallah and M. D. Plumbley. Polyphonic music transcription by non-negative sparse coding of power spectra. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, 2004.

[2] M. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-Based Music Information Retrieval: Current Directions and Future Challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[3] B. Catteau, J.-P. Martens, and M. Leman. A probabilistic framework for audio-based tonal key and chord recognition. In R. Decker and H.-J. Lenz, editors, *Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation*, pages 637–644, 2007.

[4] M. E. P. Davies, M. D. Plumbley, and D. Eck. Towards a musical beat emphasis function. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2009)*, 2009.

[5] E. Gomez. *Tonal Description of Audio Music Signals*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2006.

[6] A. P. Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 216–221, 2006.

[7] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*, chapter 23. Prentice-Hall, 1974.

[8] K. Lee and M. Slaney. Acoustic Chord Transcription and Key Extraction From Audio Using Key-Dependent HMMs Trained on Synthesized Audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):291–301, February 2008.

[9] N. C. Maddage. Automatic structure detection for popular music. *IEEE Multimedia*, 13(1):65–77, 2006.

[10] M. Mauch. *Automatic Chord Transcription from Audio Using Computational Models of Musical Context*. PhD thesis, Queen Mary University of London, 2010.

[11] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler. OMRAS2 metadata project 2009. In *Late-breaking session at the 10th International Conference on Music Information Retrieval (ISMIR 2009)*, 2009.

[12] M. Mauch and S. Dixon. Simultaneous estimation of chords and musical context from audio. *to appear in IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

[13] M. Mauch, K. C. Noland, and S. Dixon. Using musical structure to enhance automatic chord transcription. In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR 2009)*, pages 231–236, 2009.

[14] K. P. Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, 33(2):1024–1034, 2001.

[15] L. Oudre, Y. Grenier, and C. Févotte. Template-based chord recognition: Influence of the chord types. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 153–158, 2009.

[16] H. Papadopoulos and G. Peeters. Simultaneous estimation of chord progression and downbeats from an audio file. In *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, pages 121–124, 2008.

[17] G. Peeters. Chroma-based estimation of musical key from audio-signal analysis. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006.

[18] M. Ryynänen and A. P. Klapuri. Automatic Transcription of Melody, Bass Line, and Chords in Polyphonic Music. *Computer Music Journal*, 32(3):72–86, 2008.

[19] M. Varewyck, J. Pauwels, and J.-P. Martens. A novel chroma representation of polyphonic music based on multiple pitch tracking techniques. In *Proceedings of the 16th ACM International Conference on Multimedia*, pages 667–670, 2008.

[20] A. Weller, D. Ellis, and T. Jebara. Structured prediction models for chord transcription of music audio. In *MIREX Submission Abstracts*. 2009. `http://www.cs.columbia.edu/~jebara/papers/icmla09adrian.pdf`.

[21] T. Yoshioka, T. Kitahara, K. Komatani, T. Ogata, and H. G. Okuno. Automatic chord transcription with concurrent recognition of chord symbols and boundaries. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, pages 100–105, 2004.

CONCURRENT ESTIMATION OF CHORDS AND KEYS FROM AUDIO

Thomas Rocher, Matthias Robine, Pierre Hanna
LaBRI, University of Bordeaux
351 cours de la Libration
33405 Talence Cedex, France
{rocher,robine,hanna}@labri.fr

Laurent Oudre
Institut TELECOM, TELECOM ParisTech
37-39 rue Dareau
75014 Paris, France
oudre@telecom-paristech.fr

ABSTRACT

This paper proposes a new method for local key and chord estimation from audio signals. A harmonic content of the musical piece is first extracted by computing a set of chroma vectors. Correlation with fixed chord and key templates then selects a set of key/chord pairs for every frame. A weighted acyclic harmonic graph is then built with these pairs as vertices, and the use of a musical distance to weigh its edges. Finally, the output sequences of chords and keys are obtained by finding the best path in the graph.

The proposed system allows a mutual and beneficial chord and key estimation. It is evaluated on a corpus composed of Beatles songs for both the local key estimation and chord recognition tasks. Results show that it performs better than state-of-the art chord analysis algorithms while providing a more complete harmonic analysis.

1. INTRODUCTION

Harmony, like rhythm, melody or timbre, is a central aspect of Western music. This paper focuses on chord sequences and key changes, which are strong components of the harmony. Audio chord transcription has been a very active field for the past recent years. In particular, the increasing popularity of Music Information Retrieval (MIR) with applications using mid-level tonal features, has established chord transcription as useful and challenging task.

Among the numerous chord recognition methods, we can distinguish four main types of systems. The first ones can be referred as *template-based methods* [6, 9, 14], since a central information they need to perform the transcription is the definition of the chords they want to detect. Working just like pattern recognition methods, they choose for every frame the chord whose template fits the best the data. The temporal structure of the song is often captured thanks to post-processing methods working either on the sequence of detected chords or on the calculated fitness features.

Other methods rely on musical information (such as rhythm or musical structure) in order to capture a harmonically relevant chord transcription. These *music-based methods* [2, 12], implicitly or explicitly exploit information from music theory in the construction of their systems. In particular, the transitions between chords or the rhythmic structure are often modeled with parameters reflecting musical knowledge, by estimating the likelihood of a given chord being followed by a different chord, for example. Some *data-driven methods* [10, 17], use completely or partially annotated data in order to build a system which fits the audio data. In these methods, all the parameters are evaluated with training. Finally, some systems merge music- and data-based approaches in order to build *hybrid methods* [15, 16], which combine the use of training data and music theory knowledge.

All these methods have the opportunity to compare to each other in the MIREX [4], which is an annual community-based framework for the evaluation of MIR systems and algorithms. In 2009, the results for the audio chord detection were pretty close and the different methods seemed to compete at the same level of accuracy. The aim of this to work is to offer a chord estimation with a comparable level of accuracy, and estimating a sequence of local keys as well as chords.

Fewer works were achieved to estimate musical keys from audio, and the vast majority of them only consider the main key (or global key) of a piece of music [8, 13]. In these works, because only the main key is handled, key changes are ignored (songs having different local keys are either ignored or considered to be in the first local key encountered). Chai [3] presented one of the few studies on key change from audio. In this work, local key tracking was performed by a HMM-based approach, and evaluated on ten classical piano pieces.

The main contribution of this paper relies in the fact that both chord and key can benefit from each other's estimation, as chords bring out information about local key and vice versa. We present a new system estimating simultaneously both chord and key sequences from audio. The proposed method is both *template-based* and *music-based* and no training is required.

We begin to present our work by describing the system used for both key and chord estimation in Section 2. Section 3 presents the experiments performed to evaluate the accuracy of the proposed method. Conclusion and future work follow in Section 4.

2. SYSTEM DESCRIPTION

In this section, we provide the description of the proposed method, which is adapted for audio from the proposed system in [anonymous self-reference]. The overall process is illustrated in Figure 1. The system works in four major steps: (1) chroma vectors are computed from audio signal; (2) a set of harmonic candidates are selected for each frame (Figure 1(a)); (3) a weighted acyclic graph of harmonic candidates is built (Figure 1(b)), (4) the dynamic process takes place (Figure 1(c)) and the final sequence of chords/keys corresponding to the best path is outputted (Figure 1(d)).

An additional step consists in post-filtering the outputted sequence, to correct some analysis errors remaining.

2.1 Chroma Computation

The input audio file of the analysis system proposed is represented as sequences of chromas. This mid-level feature captures the tonal information since it represents the short-time energy related to each pitch class independently of octave [5]. Indeed, information about octave is not necessary for chord and key analysis purposes.

2.1.1 Tuning Issues

The chromas are computed on each frame. One of the main problem when analyzing audio musical piece is the variation in tuning. All the instruments are not always tuned to the same value, and this value often varies in time. Two options are possible. The tuning value may be analyzed, but tuning analysis assumes a stationarity. We choose to avoid this analysis by computing chroma on 36 bins and by shifting chroma at each frame according to possible tuning variations. This way, two chords played with different tuning result in two different 36 bin chromas, but results in almost the same 12 bin chroma [5].

2.1.2 Multi-Scale Approach

Instead of relying exclusively on one chromagram (sequence of chroma vectors over time), the proposed method includes a set of chromagrams. Each one has its own parameters but all share a common multiple hop size, to combine information at the same times during the piece of music. These chromagrams bring out different kinds of information, and may be subject to different treatments. Longer chromas may bring out information for key analysis, and different set of sizes for shorter chromas may fit different tempos and carry out different information useful for chord identification.

2.1.3 Filter

In order to reduce the influence of the noise, transients or sharp edges, we filter the chromagram on several frames [2, 6]. The filtering method used here is the median filtering, which has been widely used in image processing in order to correct random errors.

2.2 Selection of Harmonic Candidates

An harmonic candidate is a pair (C_i, K_i), where C_i (resp. K_i) represents a potential chord (resp. local key) for the ith frame of audio signal. C_i is then considered as a chord candidate (among possible others), and K_i as a key candidate. This section presents the processes allowing to select one or several chord/key pairs as harmonic candidate(s), and discard others.

2.2.1 Chord

The chords studied here are major and minor triads (12 major and 12 minors). Lots of works [6, 9, 14] have used chord templates to determine the likelihood of each of the 24 chords according to a chroma vector. With 12 dimensional vectors, major/minor triadic chord profile may be defined like the following:

$$\text{Major-triad} = (1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0)$$

$$\text{Minor-triad} = (1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0)$$

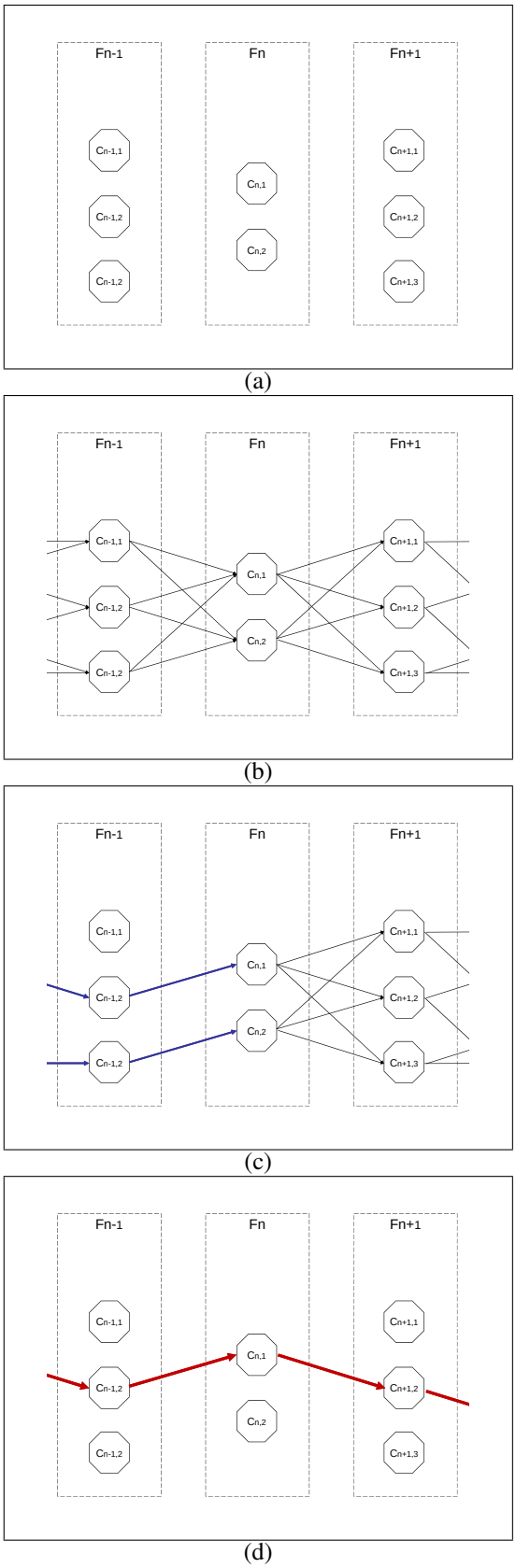


Figure 1. (a) Enumeration of harmonic candidates $C_{i,j}$ for consecutive audio frames F_i. $C_{i,j}$ represents the jth harmonic candidate for frame F_i. Time appears from left to right. (b) Creation of the edges of a weighted acyclic graph. An edge is built from each of the first frame's candidates to each of the second frame. (c) Dynamic process selects an unique path to each candidate of a given frame (here, frame n). (d) Selection of final path. The final chord/key sequences is then outputted from the sequence of chosen harmonic candidates.

All the major (resp. minor) chord templates can be obtained by rotation of the major (resp. minor) profile.

For each of the 24 chord templates, we compute a correlation score by scalar product. The following details the correlation C between a chord template T and a 12 dimensional chroma vector V.

$$C_{T,V} = \sum_{i=1}^{12} (T[i].V[i])$$

The higher the correlation is, the more likely the chord corresponding to the template is played in the considered frame. A direct way to get chord candidates is thus by selecting the chords whose templates get the higher correlation score with the chroma of a given audio frame. In the multi-scale approach as exposed in Section 2.1.2, it is possible to consider different highest correlated chords from different windowed chromas as candidate for the same frame. The different chord candidates may thus be carrying different kind of information.

2.2.2 Key

Key selection is carried out with the same approach as for chords, but with larger time frame as keys have a larger time persistence than chords. The key profiles used are presented in [18]:

$$\text{Major} = (5, 2, 3.5, 2, 4.5, 4, 2, 4.5, 2, 3.5, 1.5, 4)$$

$$\text{Minor} = (5, 2, 3.5, 4.5, 2, 4, 2, 4.5, 3.5, 2, 1.5, 4)$$

As for chord candidate computation, the correlation of each of the 24 keys (12 minors + 12 majors) are computed using a scalar product between shifted key template and chroma vectors.

2.2.3 Harmonic candidates

The harmonic candidates finally enumerated are all the possible combination of previously selected keys and chords. If n chords and m keys are selected for a given audio frame, n x m pairs are enumerated. For example, with C_M and A_m as selected chords and C_M and G_M as compatible keys, the harmonic candidates enumerated would be (C_M, C_M), (C_M, G_M), (A_m, C_M) and (A_m, G_M). A different choice can be made, by considering a compatibility between chords and keys. But an incorrect chord selected may discard the correct key (and vice versa), because the two are not compatible. For this reason, adding a compatibility between chords and keys has led to a decrease of accuracy.

2.3 Weighted Acyclic Harmonic Graph

Once the harmonic candidates are enumerated for two consecutive frames, an edge is built from each of the first frame's candidates to each of the second frame. This edge is weighted by a transition cost between the two chord candidates. This transition cost must take into account both the different selected chords, and the different selected local keys.

We thus choose to use Lerdahl's distance [11] as transition cost. If (C_x, K_x) represents the chord C_x in the key K_x, Lerdahl defines the transition cost from $x = (C_x, K_x)$ to $y = (C_y, K_y)$ as follows:

$$\delta(x \rightarrow y) = i + j + k$$

where i is the distance between K_x and K_y in the circle of fifths, j is the distance between C_x and C_y in the circle of fifths and k is the number of non-common pitch classes in the basic space of y compared to those in the basic space of x (see [11] for more details).

The distance thus provides an integer cost from 0 to 13, and is adequate for a transition cost in the proposed method, since both compatible chords and keys are involved in the cost computation. Nevertheless, this distance offers a small range of possible values. As we need to compare different paths between harmonic candidates, this small range induces a lot of equality scenarios. The Lerdahl's distance is thus slightly modified and the cost between two consecutive candidate is set to $i^\alpha + j^\beta + k$, with i, j and k defined in Section 2.3. We choose $\alpha > 1$ to discourage immediate transitions between distant keys, and encourage progressive key changes, since modulations often involve two keys close to each other in the circle of fifths. For the same reason with chords, we also choose $\beta > 1$. After experiment, α and β have been set to 1.1 and 1.01.

2.4 Finding the Best Path

Once the graph between all the harmonic candidates is formed, the best path has to be found. This task is achieved by dynamic programming [1]. In the graph, from left to right, only one edge to each harmonic candidate is preserved. Several ways to select this edge can be considered. We choose to preserve the edge minimizing the total sum of weights along the path leading to each candidate, as illustrated in Figure 1(c). The number of final paths is the number of harmonic candidates for the last frame. The final selected path is the path minimizing its total cost along its edges. This path is outputted by the program.

2.5 Post-smoothing computation

Among the selected sequence of chord/key, some errors may still be corrected by applying a post-smoothing treatment. For example, if an instrument (or a singer) plays a flattened third (Eb) as a blue note, it may induce a mode error on the selected chord (making C_m as a chord candidate and discarding C_M for the considered frame). The outputted chord sequence may thus present several consecutive frames analyzed as C_M are followed by a single frame analyzed as C_m, and then by another several C_M. A simple post treatment on the outputted sequence of chords may resolve this kind of errors.

3. EXPERIMENTS

This section presents the database used for experiments, the evaluation procedure, and the influence on the different parameters on the system accuracy. Once the best settings determined, we compare the system to a state-of-the-art method for chord estimation, and a direct template-based method for key estimation.

3.1 Database

As both local key and chord ground truth were needed, we choose to evaluate the proposed system on the Beatles audio discography (174 songs) with a 44100 Hz sampling rate. In this database, the average number of chord changes by song is 69, with an average of 7.7 different

chords by song. The average number of different local keys by song is 1.69. Chords transcriptions were checked by Christopher Harte and the MIR community, and keys annotations were provided by the Centre for Digital Music (C4DM). Both sets of transcriptions are available at `http://www.isophonics.net`.

3.2 Evaluation

In the transcriptions, chords have a root note and a type which belongs to a vast dictionary [7]. In this paper, we only focus on the root note (C, C#, D, ..., B) and the mode (maj/min) of chords. All the ground truth chords of the database have thus been mapped to min/maj triads. When the chord has no third and cannot be mapped to a min/maj triad, only the root note is considered and later compared to the corresponding estimated root note. Silences and no-chords (part of a song when no chord is defined) are ignored, as the chord/no-chord detection issue has not yet been addressed in the proposed system. The other components of the evaluation are the same as for the evaluation led in the 2009 MIREX audio chord detection task [1]. The audio signal is divided in frames of approximately 100 ms (4096 audio samples). The estimated chord is compared for each frame to the ground truth at the time corresponding to the center of the frame. The final score for a song is the number of frames where estimated chord matches the ground truth divided by the number of frames analyzed. For the local key evaluation, the procedure is identical. For each frame, the estimated key is compared to the ground truth key at the center of the frame.

3.3 Chord Estimation

Following the multi-scale approach presented in 2.1.2, we need to use different size of chromas vectors for chord estimation. The parameters for the different chroma scales are the following:

- "long" chromas: 32768 samples as window length (approximately 0.8 sec) and 8192 (approximately 0.2 sec) as hop size,
- "medium" chromas: 8192 samples as window length (approximately 0.2 sec) and 8192 (approximately 0.2 sec) as hop size,
- "short" chromas: 4096 samples as window length (approximately 0.1 sec) and 4096 (approximately 0.1 sec) as hop size.

3.3.1 Influence of Filtering

A first experiment has been carried out on long chromas to measure the influence of chroma filtering on chord estimation. For each frame, we set as chord candidate the n highest correlated chords with the maj/min chord templates . Tests go from $n = 1$ to $n = 4$. For each value of n, we compute the ratio of correctness as the number of frames for which the correct chord is among the selected candidates over the total number of frames. This ratio represents the system theoretical maximum accuracy, and is reached if every correct chord candidate is present in the final chord sequence outputted. Obviously, the higher the number of considered chord candidates is, the higher the chance is for one of them to be the correct chord, and thus the higher the ratio of correctness is. In the other hand, more chord candidates considered increase the likelihood of the system to pick a incorrect chord. Finding the balance between these two parameters is thus a real need. Table 1 presents the ratio of correctness and the system score with or without chroma filtering and for different values of n (number of chord candidates). Best chroma filtering setting is achieved by taking into account a window of 9 chromas, centered on the considered chroma.

[1] `http://www.music-ir.org/mirex/2009/index.php/Audio_Chord_Detection_Results`

No filtering

Nb. of chord candidates	1	2	3	4
Ratio of correct. (%)	58.3	71.5	78.6	82.9
System (%)	58.3	**64.9**	64.1	62.2

Filtering

Nb. of chord candidates	1	2	3	4
Ratio of correct. (%)	68.4	79.1	85.5	88.9
System (%)	68.4	**70.0**	64.3	59.3

Table 1. Percentage of correct chord in harmonic candidates and system output accuracy depending on the number of candidates and chroma filtering. Best results are achieved by limiting the number of candidates and filtering chromas.

These first results shows that filtering leads to an improvement of the ratio of correctness, from 6% with four selected candidates (82.9% to 88.9%) to more than 10% with one (58.3% to 68.4%). Filtering thus seems to correct some errors due to chroma vectors, by taking into account information from adjacent frames.

3.3.2 Influence of the Number of Chord Candidates

In Table 1, we notice the drop of the system's performance when the number of selected candidates per frame exceeds two. This can be explained by the close relationship existing among the highest correlated chord candidate of a given chroma vector. Indeed, chord templates of two major and minor chords sharing the same root note often induce a close correlation score for a given chroma. The same goes for any couple of chords close to each other in terms of Lerdahl's distance. In 80% of the frames, top 2 correlated chord candidate have a distance less or equal to 1 on the circle of fifths. Considering different candidates from the same chroma thus does not seem profitable to gain a maximum system accuracy.

3.3.3 Influence of the Multi-Scale Approach

Since a drop of accuracy is noticed when too many candidates from the same chroma are selected as candidates, we propose a new approach by considering top correlated candidates from different sized chromas. Table 2 presents the ratio of correctness as well as the system score depending on the combination of chroma size, and filtering. The general idea is to add highest correlated chord candidates from shorter chromas to the highest chord candidate of a given long chroma. Tri-candidate means the combination of the two best candidates from the two adjacent short chromas centered in a long chroma with the best candidate from the long chroma. Bi-candidate means the combination of the best candidate from the medium chroma centered in a long chroma with the best candidate from the long chroma. Since top correlated chords of two different sized chroma

Tri-candidate (1 long and 2 short chromas)

Filtering	none	long	short	both
Ratio of correct. (%)	69.8	78.2	76.6	79.1
Distinct chords (av.)	1.86	1.95	1.43	1.36
System (%)	64.5	72.2	71.8	**73.7**

Bi-candidate (1 long and 1 medium chromas)

Filtering	none	long	medium	both
Ratio of correct. (%)	65.1	75.1	73.5	76.7
Distinct chords (av.)	1.38	1.46	1.29	1.23
System (%)	62.8	71.6	70.7	**72.8**

Table 2. Percentage of correct chord in harmonic candidates and system score depending on the selection of candidates from different sized chromagrams. Each time, the average number of distinct chord candidates is mentioned. Best results are reached by filtering both long and short chromagrams, and by combining their information. Tri-candidate means the two best candidates from the two adjacent short chromas centered in a long chroma with the best candidate from the long chroma. Bi-candidate means the best candidates from the medium chroma centered in a long chroma with the best candidate from the long chroma.

may be identical, we also show the average number of distinct chord candidates per frame. As for the previous experiment, best filtering is achieved by taking into account a windows of 9 chromas and centered on the considered chroma, whatever its length.

Filtering effective in each case. If filtering is applied to only one scale, the system accuracy and ratio of correctness both benefit a little more of the long chroma filtering than the short (resp. medium) for the tri-candidate (resp. bi-candidate). Nevertheless, the best overall filtering efficiency is reached by combining the filter on both chromas size. Compared to no filter, the double sized-filter leads to an increase of around 10% for the ratio of correctness and the system accuracy, both in the tri-candidate and the bi-candidate cases. The difference between the ratio of correctness and the system score is less important by considering candidates from different chromas than by considering several candidates from the same chroma. With filtering, this difference is of 5.4% (79.1 - 73.7) for the tri-candidate configuration and of 2.9% (76.7 - 72.8) for the bi-candidate (see Table 2), when it is more than 9% (79.1 - 70) with 2 candidates from the same long chromas (see Table 1). A first way to explain this improvement is by considering the decrease of average number of distinct chord candidates, which is always lesser than 2 in both tri-candidate and bi-candidate configuration. This decrease means fewer chord candidates to consider for the system, thus decreasing the likelihood to select an incorrect chord.

Maximum accuracy is reached with tri-candidate configuration, with a system accuracy of 73.7%.

3.3.4 Post-Smoothing Treatment

We decide to apply a post-smoothing treatment to the output of the system with the best settings, which performs a 73.7% accuracy (see Table 2). The post-smoothing filter looks for output chord sequence in the form of $...AABAA...$ (resp. $...AAABCAAA...$) and corrects it in $...AAAAA...$ (resp. $...AAAAAAAA...$). By applying the post-smoothing treatment with the system best previous settings, chord detection reaches a **74.9%** accuracy.

Method	Root	Root and Mode
OGF2 (%)	**78.9**	72.3
Proposed System (%)	77.9	**74.9**

Table 3. Comparison of the proposed method with the OGF2 method, which scored 1st (resp 2nd) in the 2009 MIREX Audio Chord Detection "root estimation" task (resp. root and mode task).

3.3.5 Comparison to a state-of-the-art Method

We compare the best configuration of our system to one of the best methods of the 2009 MIREX Audio Chord Estimation task, evaluated on the same database with the same evaluation procedure. The comparison is made with the OGF2 method, proposed by Oudre et al. Results are shown in Table 3. On the root estimation only, OGF2 is 1% more accurate than the proposed method (78.9% compared to 77.9%). On the root and mode estimation, the proposed system performs better than the OGF2 method and improves by almost 3% the accuracy of the detected chords (74.9% compared to 72.3%). This comparison shows that the proposed method is comparable, and maybe even more accurate than one of the best methods presented at the 2009 MIREX when it comes to chord estimation, and compares the local key sequence as well as the chord sequence.

3.4 Local Key Estimation

Key estimation is performed on the same database than for chord estimation. We compare the key sequence output of the proposed system to a direct template-based method (DTBM). The same settings are used for the two compared methods, as we wish to evaluate the system's contribution. The window size is set to 30 sec approximately. For the proposed method, the number of key candidates per frame is set to 3. Results, shown in Table 4, detail the estimated key error made by the two compared method, by presenting relative and neighbor errors as well as correct key accuracy. Relative keys share the same key signature (for example, C_M and A_m are relative keys of each other). A neighbor key differs from the original key by an accidental. Each key has two neighbors (for instance, C_M has F_M and G_M as neighbors).

The system performs better than the DTBM method by estimating more correct keys (62.4% compared to 57.6%). Moreover, the number of errors due to non related key (different from neighbor or relative) is less important when the analysis is performed by the system (17.3% compared to 21.9%).

3.5 Reciprocal Benefit of Simultaneous Estimation

We present here an evaluation to measure the reciprocal influence of the chord and key simultaneous estimation. We compared the proposed system, which takes into account harmonic candidates (i.e. pairs of chord AND key candidates), to the same system with only chord OR key candidates. When only chord (resp.) are considered, the distance used to weigh edges in the harmonic graph is edited to take only chord (resp. key) into account. Results are shown in Table 5.

Key estimation	Correct	Rel.	Nei.	Oth.
System (%)	**62.4**	2.9	17.4	**17.3**
DTBM (%)	57.6	1.6	18.9	21.9

Table 4. DTBM: Direct Template-Based Method. Keys scores are shown in % and are split among possible errors: correct keys, relative keys (Rel.), neighbor keys (Nei.) and others (Oth.). The system performs the highest accuracy in terms of correct key detected. The number of due to non related keys is also less important with the system analysis than with the DTBM.

Harmonic Candidate	Chord C	•	•	Key K	Both C	K
System (%)	73.1	•	•	57.8	**74.9**	**62.4**

Table 5. System accuracy considering (chord,key), only chord and only key as harmonic candidates. The system performs better in chord and key estimation when taking into account both information simultaneously.

We note that both key and chord estimation are better when the harmonic candidate is the (chord,key) pair. Chord estimation accuracy drops of almost 2% (74.9 compared to 73.1) and key estimation accuracy drops of almost 5% (62.4 compared to 57.8).

4. CONCLUSION AND FUTURE WORK

This paper presents a new method for chord and local key estimation where the analysis of chord sequence and key changes are performed simultaneously. A multi-scale approach for chroma vectors is proposed, and we show an increase in accuracy when the chords are selected from different sized chromas. While the key estimation performs better than a direct template-based method, the chord accuracy shows better results than a state-of-the-art method.

Future work will involve analysis of different chord types, silence and no-chord detection as well weighing the harmonic graph of the proposed method in a probabilistic approach. Applications for MIR using both local key and chord information are also studied. For example, harmonic information may be helpful for estimating the musical structure of pieces since changes of local key generally occur at the beginning of new patterns. Furthermore, we aim at investigating the possible improvements induced by a retrieval system based on all the harmonic information, compared to existing systems that only consider chord progressions.

5. REFERENCES

[1] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[2] J.P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proc. of the International Symposium on Music Information Retrieval (ISMIR)*, pages 304–311, London, UK, 2005.

[3] W. Chai and B. Vercoe. Detection of key change in classical piano music. In *Proc. of the International Symposium on Music Information Retrieval (ISMIR)*, pages 468–473, London, UK, 2005.

[4] J. Stephen Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[5] E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, University Pompeu Fabra, Barcelona, Spain, July 2006.

[6] C. Harte and M. Sandler. Automatic chord identification using a quantised chromagram. In *Proc. of the Audio Engineering Society*, Barcelona, Spain, 2005.

[7] C. Harte, M. Sandler, and A. Samer. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proc. 4th Int. Conf. on Music Information Retrieval (ISMIR), 2005*, pages 66–71, 2005.

[8] O. Izmirli. Audio key finding using low-dimensional spaces. In *Proc. of the International Symposium on Music Information Retrieval (ISMIR)*, pages 127–132, Victoria, Canada, 2006.

[9] K. Lee. Automatic chord recognition from audio using enhanced pitch class profile. In *Proc. of the International Symposium on Music Information Retrieval (ISMIR)*, Victoria, Canada, 2006.

[10] K. Lee and M. Slaney. Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. *IEEE Trans. on Audio, Speech and Language Processing*, 16(2):291–301, 2008.

[11] F. Lerdahl. *Tonal Pitch Space*. Oxford University Press, 2001.

[12] M. Mauch and S. Dixon. Simultaneous estimation of chords and musical context from audio. *IEEE Trans. on Audio, Speech and Language Processing*, 2010.

[13] K. Noland and M. Sandler. Key estimation using a hidden markov model. In *Proc. of the International Symposium on Music Information Retrieval (ISMIR)*, pages 121–126, Victoria, Canada, 2006.

[14] L. Oudre, Y. Grenier, and C. Févotte. Template-based chord recognition : influence of the chord types. In *Proc. of the International Symposium on Music Information Retrieval (ISMIR)*, pages 153–158, Kobe, Japan, 2009.

[15] H. Papadopoulos and G. Peeters. Large-scale study of chord estimation algorithms based on chroma representation and HMM. In *Proc. of the International Workshop on Content-Based Multimedia Indexing*, pages 53–60, Bordeaux, France, 2007.

[16] M.P. Ryynänen and A.P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[17] A. Sheh and D.P.W. Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In *Proc. of the International Symposium on Music Information Retrieval (ISMIR)*, pages 185–191, Baltimore, MD, 2003.

[18] D. Temperley. *The Cognition of Basic Musical Structures*. The MIT Press, 1999.

SOLVING MISHEARD LYRIC SEARCH QUERIES USING A PROBABILISTIC MODEL OF SPEECH SOUNDS

Hussein Hirjee **Daniel G. Brown**
University of Waterloo
Cheriton School of Computer Science
{hahirjee, browndg}@uwaterloo.ca

ABSTRACT

Music listeners often mishear the lyrics to unfamiliar songs heard from public sources, such as the radio. Since standard text search engines will find few relevant results when they are entered as a query, these misheard lyrics require phonetic pattern matching techniques to identify the song. We introduce a probabilistic model of mishearing trained on examples of actual misheard lyrics, and develop a phoneme similarity scoring matrix based on this model. We compare this scoring method to simpler pattern matching algorithms on the task of finding the correct lyric from a collection given a misheard query. The probabilistic method significantly outperforms all other methods, finding 5-8% more correct lyrics within the first five hits than the previous best method.

1. INTRODUCTION

Though most Music Information Research (MIR) work on music query and song identification is driven by audio similarity methods, users often use lyrics to determine the artist and title of a particular song, such as one they have heard on the radio. A common problem occurs when the listener either mishears or misremembers the lyrics of the song, resulting in a query that *sounds* similar to, but is not the same as, the actual words in the song she wants to find.

Furthermore, entering such a misheard lyric query into a search engine often results in many practically identical hits caused by various lyric sites having the exact same versions of songs. For example, a Google search for "Don't walk on guns, burn your friends" (a mishearing of the line "Load up on guns and bring your friends" from Nirvana's "Smells Like Teen Spirit") gets numerous hits to "Shotgun Blues" by Guns N' Roses (Figure 1). A more useful search result would give a ranked list of possible matches to the input query, based on some measure of similarity between the query and text in the songs returned. This goal suggests a similarity scoring measure for speech sounds: which potential target lyrics provide the best matches to a misheard lyric query?

Figure 1. Search for misheard lyrics from "Smells Like Teen Spirit" returning results for Guns N' Roses.

The misheard lyric phenomenon has been recognized for quite some time. Sylvia Wright coined the autological term "Mondegreen" in a 1954 essay. This name refers to the lyric "They hae slain the Earl O' Moray / And laid him on the green," misheard to include the murder of one Lady Mondegreen as well [1]. However, the problem has only recently been tackled in the MIR community.

Ring and Uitenbogerd [2] compared different pattern-matching techniques to find the correct target lyric in a collection given a misheard lyric query. They found that a method based on aligning syllable onsets performed the best at this task, but the increase in performance over simpler methods was not statistically significant. Xu et al. [3] developed an acoustic distance metric based on phoneme confusion errors made by a computer speech recognizer. Using this scoring scheme provided a slight improvement over phoneme edit distance; both phonetic methods significantly outperformed a standard text search engine.

In this paper, we describe a probabilistic model of mishearing based on phonetic confusion data derived from pairs of *actual* misheard and correct lyrics found on misheard lyrics websites. For any pair of phonemes a and b, this model produces a log-odds score giving the likelihood of a being (mis)heard as b. We replicate Ring

and Uitenbogerd's experiments using this model, as well as phonetic edit distance as described in Xu et al.'s work, on misheard lyric queries from the misheard lyrics site KissThisGuy.com. Our statistical method significantly outperforms all other techniques, and finds up to 8% more correct lyrics than phonetic edit distance.

2. RELATED WORK

Ring and Uitenbogerd [2] compared three different pattern-matching techniques for finding the correct lyrics or matches judged to be relevant given a misheard lyric query. The first is a simple Levenshtein edit distance performed on the unmodified text of the lyrics. The second, Editex, groups classes of similar-sounding letters together and does not penalize substitutions of characters within the same class as much as ones not in the same class.

The third algorithm is a modified version of Syllable Alignment Pattern Searching they call SAPS-L [4]. In this method, the text is first transcribed phonetically using a set of simple text-to-phoneme rules based on the surrounding characters of any letter. It is then parsed into syllables, with priority given to consonants starting syllables (onsets). Pattern matching is performed by local alignment where matching syllable onset characters receive a score of +6, mismatching onsets score -2, and other characters score +1 for matches and -1 for mismatches. Onsets paired with non-onset characters score -4, encouraging the algorithm to produce alignments in which syllables are matched before individual phonemes. SAPS is especially promising since it is consistent with psychological models of word recognition in which segmentation attempts are made at the onsets of strong syllables [5].

They found that the phonetic based methods, Editex and SAPS-L, did not outperform the simple edit distance for finding all lyrics judged by assessors to sound similar to a given query misheard lyric but SAPS-L most accurately determined its single correct match. However, due to the size of the test set of misheard lyric queries, they did not establish statistical significance for these results.

In a similar work, Xu et al. [3] first performed an analysis of over 1000 lyric queries from Japanese question and answer websites and determined that 19% of these queries contained misheard lyrics. They then developed an acoustic distance based on phoneme confusion to model the similarity of misheard lyrics to their correct versions. This metric was built by training a speech recognition engine on phonetically balanced Japanese telephone conversations and counting the number of phonemes confused for others by the speech recognizer. They then evaluated different search methods to determine the correct lyric in a corpus of Japanese and English songs given the query misheard lyrics. Phonetic pattern matching methods significantly outperformed Lucene, a standard text search engine. However, their acoustic distance metric only found 2-4% more correct lyrics than a simpler phoneme edit distance, perhaps due to its basis on machine speech recognition. They also implemented an indexed version of the search which reduced the running time by over 85% with less than 5% loss of accuracy.

3. METHOD

3.1 A Scoring Approach

Similar to our method for identifying rhymes in rap lyrics [6], we use a model inspired by protein homology detection techniques, in which proteins are identified as sequences of amino acids. In the BLOSUM (BLOcks of amino acid SUbstitution Matrix) local alignment scoring scheme, pairs of amino acids are assigned log-odds scores based on the likelihood of their being matched in alignments of homologous proteins – those evolving from a shared ancestor [7]. In a BLOSUM matrix M, the score for any two amino acids i and j, is calculated as

$$\mathrm{M}[i,j] = \log_2(\Pr[i,j|\mathrm{H}]/\Pr[i,j|\mathrm{R}]), \qquad (1)$$

where $\Pr[i,j|\mathrm{H}]$ is the likelihood of i being matched to j in an alignment of two homologous proteins, while $\Pr[i,j|\mathrm{R}]$ is the likelihood of them being matched by chance. These likelihoods are based on the co-occurrence frequencies of amino acids in alignments of proteins known to be homologous. A positive score indicates a pair is more likely to co-occur in proteins with common ancestry; a negative score indicates the pair is more likely to co-occur randomly. Pairs of proteins with high-scoring aligned regions are labeled homologous.

In the song lyric domain, we treat lines and phrases as sequences of phonemes and develop a model of mishearing to determine the probability of one phoneme sequence being misheard as another. This requires a pairwise scoring matrix which produces log-odds scores for the likelihood of pairs of phonemes being confused. The score for a pair of phonemes i and j is calculated as in Equation (1), where $\Pr[i,j|\mathrm{H}]$ is the likelihood of i being heard as j, and $\Pr[i,j|\mathrm{R}]$ is the likelihood of i and j being matched by chance.

As for the proteins that give rise to the BLOSUM matrix, these likelihoods are calculated using frequencies of phoneme confusion in actual misheard lyrics. Given a phoneme confusion frequency table F, where $\mathrm{F}_{i,j}$ is the number of times i is heard as j (where j may equal i), the mishearing likelihood is calculated as

$$\Pr[i,j|\mathrm{H}] = \mathrm{F}_{i,j}/\sum_m\sum_n \mathrm{F}_{m,n}. \qquad (2)$$

This corresponds to the proportion of phoneme pairs in which i is heard as j. The match by chance likelihood is calculated as

$$\Pr[i,j|\mathrm{R}] = \mathrm{F}_i \times \mathrm{F}_j/(\sum_m \mathrm{F}_m \times \sum_n \mathrm{F}_n), \qquad (3)$$

where F_i is the total number of times phoneme i appears in the lyrics. This is simply the product of the background frequencies of each phoneme in the pair.

We note that our work is in some ways similar to that of Ristad and Yianilos [8], for learning string edit distance.

3.2 Training Data for the Model

To produce the phoneme confusion frequency table F, we require a training set of misheard lyrics aligned to their correct versions. Our corpus contains query and target pairs from two user-submitted misheard lyrics websites, KissThisGuy.com and AmIRight.com. In both cases, the first phrase in the pair is the song lyric heard by the submitter and the second phrase is the true lyric in the song.

The KissThisGuy.com pairs were provided by HumorBox Entertainment, the parent company of KissThisGuy.com, and consist of 9,527 pairs randomly selected from the database and comprising 10% of the total number of misheard lyrics on the website. The pairs from AmIRight.com were selected from the pages for the top 10 artists (by number of misheard lyrics submitted) on the site and total 11,261 pairs, roughly corresponding to 10% of the misheard lyrics on the site. The artists included are The Beatles, Michael Jackson, Elton John, Nirvana, Red Hot Chili Peppers, Queen, Metallica, Madonna, traditional songs, and Green Day.

3.3 Producing Transcriptions

We first use the Carnegie Mellon University pronouncing dictionary to obtain phonetic transcriptions of the lyrics. The CMU pronouncing dictionary has phonetic transcriptions for over 100,000 words and is tailored specifically for North American English, the language used by the majority of artists in our data [9]. We use the Naval Research Laboratory's text-to-phoneme rules to transcribe any words not found in the dictionary [10].

The transcriptions contain 39 phonemes, consisting of 24 consonants, including affricates such as /tʃ/ and /dʒ/, and 15 vowels, including diphthongs like /ɔɪ/ and /aɪ/ [11]. Additionally, metrical stress is included for the vowels to indicate whether they are part of syllables with primary (1), secondary (2), or no (0) stress. To avoid overfitting due to the relatively small number of secondary stressed syllables in the dictionary, we combine primary and secondary stresses into strong stress to contrast with weak or unstressed syllables. This results in a set of 54 phonemes: 24 consonants and 30 stress-marked vowels.

To better model actual prosody in singing, we reduce the stress in common single-syllable words with less metrical importance such as "a," "and," and "the." To allow for variation in the likelihood of different phonemes being missed (deleted) or misheard without having been sung (inserted), we introduce an additional symbol for gaps in alignment and treat it like any other phoneme. This would let a "softer" approximant such as /r/ get a lesser penalty when missed than a "harder" affricate such as /tʃ/.

3.4 Iterated Training

We perform an iterated alignment method with the lyric pairs to determine the confusion frequencies. In the first phase, phonemes are lined up sequentially starting from the left end of each phrase in the pair. This may seem to be too rough an alignment method, but it results in the highest frequencies for identical phoneme pairs since most of the misheard lyrics contain some correct lyrics within them. For example, "a girl with chestnut hair" being misheard as "a girl with just no hair" from Leonard Cohen's "Dress Rehearsal Rag" would be aligned as

ə g ˈɝl w ɪ θ dʒ ˈʌ s t n oʊ h ˈeɪ r
ə g ˈɝl w ɪ θ tʃ ˈɛ s t n ə t h ˈeɪ r,

with all phonemes matching exactly until the /tʃ/ heard as /dʒ/, then the /ˈɛ/ heard as /ˈʌ/, etc. From these simple alignments, we construct an initial phoneme confusion frequency table F'.

Since gaps do not appear explicitly in any lyrics, we approximate their occurrence by adding gap symbols to the shorter phrase in each pair to ensure the phrases are of the same length. In the example above, we would count one gap, and have it occurring as an /r/ being missed in the F' table. This approximation results in an essentially random initial distribution of gap likelihood across phonemes.

Now, given the initial frequency table, we calculate an initial scoring matrix M' using Equations (1) to (3) above. We then use the scores found in M' to align the pairs in the second phase of training. In this stage, we use dynamic programming to produce the optimum global alignment between each misheard lyric and its corresponding correct version, which may include gaps in each sequence. We then trace back through the alignment and update the phoneme co-occurrences in a new confusion frequency table F. For the example cited above, the new alignment would look like

ə g ˈɝl w ɪ θ dʒ ˈʌ s t n oʊ h ˈeɪ r
ə g ˈɝl w ɪ θ tʃ ˈɛ s t n ə t h ˈeɪ r.

The gap occurs earlier and results in a missed /t/ in the F table. After all the pairs have been processed, we calculate a final scoring matrix M from frequency table F, as above.

3.5 Structure of the Phonetic Confusion Matrix

One interesting property of the phonetic confusion matrix is that, from first principles, we discover perceptual similarities between sounds: if two phonemes a and b have positive scores in our confusion matrix, then they sound similar to the real people who have entered these queries into our database.

Table 1 shows all of the pairs of distinct consonant phonemes a and b such that $M[a, b]$ is positive. These consist mainly of changes in voicing (*e.g.*, /g/ versus /k/) or moving from a fricative to a plosive (*e.g.*, /f/ versus /p/); the only distinct consonant pairs scoring above +1.0 are pairs of sibilants (such as /tʃ/ versus /dʒ/ or /ʒ/ versus /ʃ/). All of these similarities are discovered without knowledge of what sounds similar; they are discovered by the training process itself.

When examining stressed vowel scores in detail, it becomes evident that vowel height is the least salient articulatory feature for listeners to determine from sung words, as most of the confused vowels differ mainly in height. These pairs include /ɑ/ and /ʌ/, /ʌ/ and /ʊ/, /æ/ and /ɛ/, and /ɛ/ and /ɪ/. Other common confusions include vowels differing mainly in length and diphthongs confused with their

Query Phoneme	Target Phoneme
/b/	/f/,/p/,/v/
/tʃ/	/dʒ/,/k/,/ʃ/,/t/,/ʒ/
/f/	/b/,/p/,/θ/
/g/	/dʒ/,/k/
/dʒ/	/tʃ/,/ʃ/,/y/,/ʒ/
/k/	/g/
/ŋ/	/n/
/p/	/b/,/f/,/θ/,/v/
/s/	/z/
/ʃ/	/tʃ/,/dʒ/,/s/,/ʒ/
/θ/	/f/
/z/	/s/,/ʒ/
/ʒ/	/dʒ/,/ʃ/

Table 1. Non-identical consonants with positive scores.

constituent phonemes, such as /ɪ/ with /i/, /ɑ/ with /aʊ/, and /ɔ/ with /ɔʊ/.

When examining differences in gap scores, we find that the phonemes most likely to be missed (deleted) or heard without being sung (inserted) are /r/, /d/, /ŋ/, and /z/. Although the model is trained without any domain knowledge, a semantic explanation is likely for this finding since /d/ and /z/ are often added to words to form past tenses or plurals which could be easily confused. /ŋ/ is often changed to /n/ in verb present progressive tenses in popular music; for example, "runnin' " could be sung for "running." The phonemes least likely to be missed are /ʒ/, /ʃ/, /ɔɪ/, and /ɪ/, probably (with the surprising exception of /ɪ/) due to their relative "length" of sound. Similarly, /ʃ/, /ʊ/, /ɪ/, and /ɝ/ were least likely to be heard without being sung.

3.6 Searching Method

To perform phonetic lyric search with this model, we use matrix M to score semi-local alignments [12] between the query phrase (sequence of phonemes) and all candidate songs in the database. The highest scoring alignment indicates the actual song lyric most likely to be heard as the query, according to our model.

In addition to this phonemic model, we develop a syllable-based model which produces a log-likelihood score for any syllable being (mis)heard as another. For any pair of syllables a and b, we calculate this score as

$$\mathrm{S}[a,b] = \mathrm{align}(a_o, b_o) + \mathrm{M}[a_v, b_v] + \mathrm{align}(a_e, b_e), \quad (4)$$

where a_v is the vowel in syllable a and $\mathrm{M}[a_v, b_v]$ is defined in Equation **??** above. $\mathrm{align}(a_o, b_o)$ is the score for the optimal global alignment between the onset consonants of a and b, and a_e is the ending consonants (or coda) for syllable a.

Searching and training are performed in the same way as with the phonemic method, except that syllables are aligned instead of phonemes. Essentially, this ensures that vowels only match with other vowels and consonants only match with other consonants.

4. EXPERIMENT

To compare the performance of the probabilistic model of mishearing with other pattern matching techniques, we reproduced the experiment of Ring and Uitenbogerd [2] finding the best matches for a query set of misheard lyrics in a collection of full song lyrics containing the correct version of each query.

4.1 Target and Query Sets

We used Ring and Uitenbogerd's collection, comprising a subset of songs from the lyrics site lyrics.astraweb.com containing music from a variety of genres by artists such as Alicia Keys, Big & Rich, The Dave Matthews Band, Queensrÿche, and XTC. After removing duplicates, it contained 2,345 songs with a total of over 486,000 words. This formed our set of targets.

We augmented their original query set of 50 misheard lyrics from AmIRight.com with 96 additional misheard lyrics from the KissThisGuy.com data. These additional queries have corresponding correct lyric phrases that match exactly with a phrase from a single song in the collection. They do not necessarily match the same song the query lyric was misheard from, but only had one unique match in the collection. For example, "you have golden eyes" was heard for "you're as cold as ice" from Foreigner's "Cold As Ice," a song which does not appear in the collection. However, the same line occurs in 10cc's "Green Eyed Monster," which is in the collection. We included at most one query for each song in the collection. In practice, misheard lyric queries may have correct counterparts which appear in multiple songs, potentially making our results less generalizable for large corpora.

4.2 Methods Used in Experiments

We implemented three different pattern-matching algorithms in addition to the probabilistic mishearing models described above: SAPS-L and simple edit distance as the best methods from Ring and Uitenbogerd's paper, and phonemic edit distance to estimate a comparison with Xu et al.'s Acoustic Distance. (The actual scoring matrix used in that work was unavailable.) We removed all test queries from the training set for the probabilistic models.

4.3 Evaluation Metrics

For each method, we found the top 10 best matches for each misheard lyric in our query set and use these results to calculate the Mean Reciprocal Rank (MRR_{10}) as well as the hit rate by rank for the different methods. The MRR_{10} is the average of the reciprocal ranks across all queries, where reciprocal rank is one divided by the rank of the correct lyric if it is in the top ten, and zero otherwise. Thus, if the second returned entry is the correct lyric, we score 0.5 for that query and so on. The hit rate by rank is the cumulative percentage of correct lyrics found at each rank in the results.

Pattern Matching Method	Mean Reciprocal Rank
Probabilistic Phoneme Model	0.774
Phoneme Edit Distance	0.709
Probabilistic Syllable Model	0.702
SAPS-L	0.655
Simple Edit Distance	0.632

Table 2. Mean reciprocal rank after ten results for different search techniques.

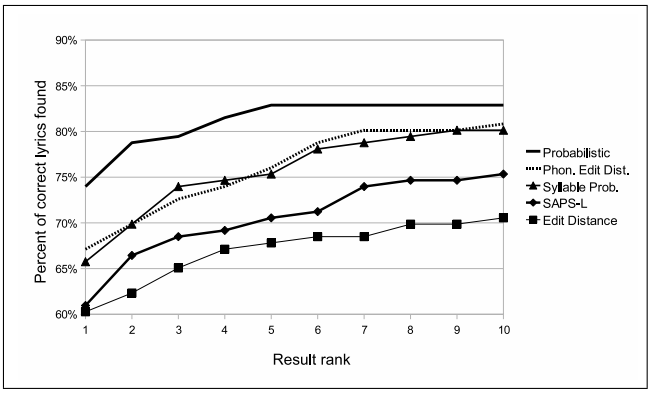


Figure 2. Cumulative percentage of correct lyrics found by rank for different search methods. The probabilistic phoneme model finds 5-8% more correct targets in the first five matches than the next best method. The probabilistic syllable model and phoneme edit distance perform nearly identically, and significantly better than SAPS-L and simple edit distance.

5. RESULTS

The probabilistic model of phoneme mishearing significantly outperformed all other methods in the search task, achieving an MRR of 0.774 and ranking the correct answer for 108 of the 146 queries (74.0%) first. The next best methods were phonemic edit distance and probabilistic syllable alignment, receiving MRRs of 0.709 and 0.702, respectively. Performing a paired t-test on the reciprocal rankings of the probabilistic phoneme model and the phonemic edit distance returned a p-value less than 0.001, strongly indicating that the results were drawn from different distributions. There was no statistically significant difference between the probabilistic syllable model and the phonemic edit distance results. Both these methods were found to significantly outperform SAPS-L, with p-values less than 0.05 on the t-tests. SAPS-L produced an MRR of 0.655, which was marginally better than the simple edit distance's MRR of 0.632. However, the difference between these two was again not found to be statistically significant. The Mean Reciprocal Rank results are shown in Table 2.

The hit rate by rank (Figure 2) further illustrates the effectiveness of the probabilistic phoneme model as it ranks between 5% and 8% more correct lyrics within the top five matches than phonemic edit distance and the probabilistic syllable model. These next two methods appear to perform equally well and considerably better than SAPS-L and edit distance. SAPS-L seems to improve in performance over simple edit distance moving down the ranks, indicating that it may be better at finding less similar matches.

5.1 Analysis of Errors

We also observe that the performance of the probabilistic phoneme model plateaus at a hit rate of 83%. This corresponds to 121 of the 146 misheard lyric queries, and we provide a brief analysis of some of the 25 queries missed.

5.1.1 Differences Among Methods

The phoneme edit distance method did not return any correct lyrics not found by the probabilistic phoneme model. The one query for which SAPS-L returned a hit in the top 10 and the statistical model did not was "spoon aspirator" for "smooth operator," from Sade's song of the same name. In SAPS-L, this was transcribed as "SPoon AsPiRaTor," getting a score of 24 when matched with "Smooth OPeRaTor." The relatively high number of matching syllable onsets (S, P, R, and T) in the short query gave SAPS-L the advantage since it heavily emphasizes onsets. On the other hand, the probabilistic method produced higher scores for results such as "spoon in spoon stir(ring)" and "I'm respirating" due to the high number of exactly matching and similar phonemes.

The probabilistic syllable model also returned a hit for one query for which the phoneme model did not. The misheard lyric in this case was "picture Mona runnin' " heard for "get your motor runnin' ", presumably from Steppenwolf's "Born to be Wild." This was likely due to the parsing of the phonetic transcription so that paired syllables had high scores at both the onset and ending consonants ("Mon" and "mot", "run" and "run"). The top ranking match using the phoneme model was "picture on your button." When the phrases are transcribed without word or syllable boundaries, the only large differences are an inserted /m/ from "Mona" and a missed /b/ from "button."

5.1.2 Common Types of Errors

Though syllable parsing and alignment may have helped for the two misheard lyrics described above, the majority of the queries not returning results tended to be quite dissimilar from their target correct lyrics. Some examples of these include a young child hearing "ooh, Tzadee, I'm in a cheerio" for "we are spirits in the material" from The Police's "Spirits in the Material World;" "Girl, I wanna yodel" for "You're The One That I Want" from Grease; "Apple, dapple, and do" for Prince's "I Would Die 4 U;" and "Swingin' the bat" for the Bee Gees' "Stayin' Alive." In other interesting cases the listener superfluously heard the singer's name within the song lyrics: "Freddie time!" for "and turns the tides" in Queen's My Fairy King, and "Oh, Lionel (Oh Line')" for Lionel Richie's "All Night Long (all night"). Without knowledge of the song artist, it would be hard to consider these similar to their originals.

The other common problem preventing the algorithms

Pattern Matching Method	Correlation
Probabilistic Phoneme Model	0.45
Phoneme Edit Distance	0.54
Probabilistic Syllable Model	0.55
SAPS-L	0.53
Simple Edit Distance	0.51

Table 3. Correlation between misheard query length and reciprocal rank of correct answer returned. The positive correlations indicate that longer queries are more likely to have the correct lyric ranked higher, though this effect is least pronounced for the probabilistic phoneme model.

from finding the correct matches for many misheard lyrics stems from the short length of such queries. Some of these include "chew the bug" for "jitterbug," "can of tuna" for "can't hurt you now," "rhubarb" for "move out", and "wow thing" for "wild thing." While these tend to be fairly similar to their correct counterparts, their short length makes it much easier to find exact partial matches which score highly enough to balance the dissimilar remaining portions. Though the models are trained on mishearing, most misheard lyrics tend to have parts heard correctly, so matching identical phonemes will usually give the highest scores. For all methods, longer queries were more likely to have their correct lyrics found in the top 10, resulting in a positive correlation between the length of the query and the reciprocal rank of the correct result. Table 3 details these correlations for the different algorithms. Note that this correlation is smallest for the probabilistic phoneme model: it is the least fragile in this way.

5.2 Running Time

The current implementation of the search algorithm is an exhaustive dynamic programming search over the entire collection of lyrics, resulting in O(mn) computing complexity per query, where m is the length of the query and n is the size of the collection. This would likely not be feasible in a commercial application due to the long search time required (about 3 seconds per query on a 1.6 GHz laptop). Xu et al. [3] did demonstrate the effectiveness of using *n*-gram indexing to reduce the running time by pre-computing the distances from 90% of all syllable 3-grams in their collection and pruning off the most dissimilar lyrics. However, this is simpler with Japanese pronunciation than English due to the relatively limited number of possible syllables. Determining the effectiveness of English phoneme *n*-gram indexing while balancing speed, accuracy, and memory use remains an open problem.

6. CONCLUSION

We introduce a probabilistic model of mishearing based on phoneme confusion frequencies calculated from alignments of actual misheard lyrics with their correct counterparts. Using this model's likelihood scores to perform phoneme alignment pattern matching, we were better able to find the correct lyric from a collection given a misheard lyric query. Tested on 146 misheard lyric queries with correct target lyrics in a collection of 2,345 songs, the probabilistic phoneme model produces a Mean Reciprocal Rank of 0.774 and finds up to 8% more correct lyrics than the previous best method, phoneme edit distance, which achieves an MRR of 0.709.

7. ACKNOWLEDGEMENTS

We thank Eric Barberio, from HumorBox Entertainment, for supplying us with the KissThisGuy.com queries we have used in this study. Our research is supported by the Natural Sciences and Engineering Research Council of Canada and by an Early Researcher Award from the Province of Ontario to Daniel Brown.

8. REFERENCES

[1] S. Wright: "The Death of Lady Mondegreen," *Harper's Magazine*, Vol. 209 No. 1254 pp. 48-51, 1954.

[2] N. Ring and A. Uitenbogerd: "Finding 'Lucy in Disguise': The Misheard Lyric Matching Problem," *Proceedings of AIRS 2009*, pp. 157–167, 2009.

[3] X. Xu, M. Naito, T. Kato, and H. Kawai: "Robust and Fast Lyric Search Based on Phonetic Confusion Matrix," *Proceedings ISMIR 2009*, 2009.

[4] R. Gong and T. Chan: "Syllable Alignment: A Novel Model for Phonetic String Search," *IEICE Transactions on Information and Systems*, Vol. 89 No. 1 pp. 332–339, 2006.

[5] D. Norris, J.M. McQueen, and A. Cutler: "Competition and Segmentation in Spoken Word Recognition," *Third International Conference on Spoken Language Processing*, 1994.

[6] H. Hirjee and D.G. Brown: "Automatic Detection of Internal and Imperfect Rhymes in Rap Lyrics," *Proceedings ISMIR 2009*, 2009.

[7] S. Henikoff and J.G. Henikoff: "Amino Acid Substitution Matrices from Protein Blocks" *Proceedings of the NAS*, Vol. 89 No. 22 pp. 10915–10919, 1992.

[8] E.S. Ristad and P.N. Yianilos: "Learning string-edit distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20 No. 5 pp.522-532, 1998.

[9] K. Lenzo: *The CMU Pronouncing Dictionary*, 2007 `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`.

[10] H.S. Elovitz, R.W. Johnson, A. McHugh, J.E. Shore: "Automatic translation of English text to phonetics by means of letter-to-sound rules," *Interim Report Naval Research Lab*. Washington, DC., 1976

[11] International Phonetic Association: *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*, Cambridge University Press, 1999.

[12] R. Durbin, S. Eddy, A. Krogh, G. Mitchison: *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, 1999.

COLLABORATIVE FILTERING BASED ON P2P NETWORKS

Noam Koenigstein[1], Gert Lanckriet[2], Brian McFee[3], and Yuval Shavitt[1]

[1]School of Electrical Engineering, Tel Aviv University
[2]Electrical and Computer Engineering, University of California, San Diego
[3]Computer Science and Engineering, University of California, San Diego

ABSTRACT

Peer-to-Peer (P2P) networks are used by millions of people for sharing music files. As these networks become ever more popular, they also serve as an excellent source for Music Information Retrieval (MIR) tasks. This paper reviews the latest MIR studies based on P2P data-sets, and presents a new file sharing data collection system over the Gnutella. We discuss several advantages of P2P based data-sets over some of the more "traditional" data sources, and evaluate the information quality of our data-set in comparison to other data sources (Last.fm, social tags, biography data, and MFCCs). The evaluation is based on an artists similarity task using Partial Order Embedding (POE). We show that a P2P based Collaborative Filtering data-set performs at least as well as "traditional" data-sets, yet maintains some inherent advantages such as scale, availability and additional information features such as ID3 tags and geographical location.

1. INTRODUCTION

The usage of P2P based information for music information retrieval (MIR) tasks is gaining momentum. The process of collecting Collaborative Filtering (CF) data from P2P networks is typically more complex than from more "traditional" sources such as Last.fm or social networks, but there are several advantages that significantly undermine this small impairment.

Barrington et al. [2] compared different approaches for music recommendation with a user study of 185 subjects. They concluded that approaches based on collaborative filtering which essentially capture the "wisdom of the crowds", outperform content-based approaches so long as the data-set used is sufficiently comprehensive. However, when the data-set is insufficient, or the artists are less popular (those in the long tail), we are compelled to use content-based approaches. The scale of a CF data-set is therefore of great importance. Using a crawler of the Gnutella file-sharing network, we were able to record 281,865,501 user-to-song relations of over 1.3 million users in a single 24 hours crawl. Such scales far exceed the "traditional" CF data-sets such as the well-established Last.fm data-set provided by [6] (17,559,530 records from 359,347 users).

Another advantage of P2P data-sets over traditional data-sets is the availability of information, mitigating the need for agreements with website operators and various restrictions they pose on the amount of data collected or its usage. Due to their decentralized nature and open protocols, P2P networks are a source for independent large scale data collection. Anyone who overcomes the initial technological barrier can crawl the network and collect valuable information.

Data-sets based on shared folders typically include ID3 tags that reveal information such as the title, artist, album and track number. Although sometimes these records are absent or conflicting, it is often still possible to restore the correct values. In this paper for example, we used majority voting to decide on the correct artist names for different files. P2P data-sets typically include also the IP addresses of the users. The IP address can be used as a unique user identifier for short time spans, but more importantly, it also allows for geographical classification of users. IP-based geographical classification is highly accurate and can reveal not only the user's country and state, but also the user's city and sometimes even smaller areas like the boroughs of New-York City. Such geographical resolution was used by [14] for identifying emerging local musical artists with high potential for a breakthrough.

Despite all their advantages, P2P networks are quite complex, making the collection of a comprehensive data-set far from being trivial, and in some cases practically unfeasible. First, P2P networks have high user churn, causing users to constantly connect and disconnect from the network, being unavailable for changing periods. Second, users in P2P networks often do not expose their shared data in order to maintain high privacy and security measures, therefore disabling the ability to collect information about their shared folders. Finally, users often delete content after using it, leaving no trace of its usage.

A different complexity involves the usage of meta-data, which was shown to be particulary useful for finding similarity between performing artists [17]. The content on file sharing networks is mostly ripped by individual users for consumption by other users. User based interactions are a desirable property in IR data-sets, however when it comes to meta-data, its the main source for ambiguities and noise. Be it a movie, a song, or any other file type, typically there would be several similar duplications available on the network. The files may be digitally identical, thus having the same hash signature, yet bearing different file names, and meta-data tags. Duplication in meta-data tags typically

caused by spelling mistakes, missing data, and different variations on the correct values. In the Gnutella network for example, only 7-10% of the queries are successful in returning useful content [30]. A common hash signature can facilitate similar files grouping, nonetheless it does not solve the problem of copies that are not digitally identical. The problem of meta-data ambiguities in P2P data-set is addressed in [16].

2. BACKGROUND

MIR studies based on P2P networks belong to one of two categories:

- **Studies Based on Queries:** Queries in a file-sharing network represent the current tastes and interests of users. A query is issued upon a request by a user searching for a specific file, or content relevant to the search string. Query data-sets are time dependent, and because of dynamic IP assignments, it can be difficult to track a single user over time. Therefore, query-based studies tend to focus on temporal trends such as predicting artists success or artists ranking. Queries are generally ineffective for predicting artist similarity and general recommendation systems because the information gathered per user is limited to a short time period, and thus only a few files per user are usually available.

- **Studies Based on Shared Folders:** The content of a user's shared folder accumulates over time. It can be viewed as an integration of a user's taste over an extended period of time. Data-sets derived from shared folders are therefore the preferred choice for similarity tasks such as recommender systems.

2.1 Previous Query-based Studies

In [14], geographically identified P2P queries were used in order to detect emerging musical talents. The detection algorithm is based on the observation that emerging artists, especially rappers, have a discernible stronghold of fans in their hometown area, where they are able to perform and market their music. In a file-sharing network, this is reflected as a spike in the spatial distribution of queries. The algorithm mimics human scouts by looking for performers which exhibit a sharp increase in popularity within a small geographic region.

The algorithm in [14] is effective for predicting the success of emerging artists, but it cannot be applied on well-established artists. Bhattacharjee et al. [4,5] have used P2P activity to predict an album's life cycle and trends on the Billboard's top 200 albums chart. Both papers used the WinMx file-sharing network. In [4], they showed that P2P sharing activity levels provide leading indicators for the direction of movement of albums on the Billboard charts. In [5], a linear regression model was used to show that sharing activity may be used to predict an album's life cycle. More recently, [17] used the C4.5 [22] and BFTree [8, 26] algorithms on queries collected from the Gnutella network in order to predict a song's top rank on the Billboard singles chart.

A different approach for using P2P queries was taken by [13]. Grace et al. [10] noticed that although music sales are losing their role as means for music dissemination, they are still used by the music industry for ranking artist success, e.g., in the Billboard Magazine chart. They therefore suggested using social networks as an alternative ranking system; a suggestion which is problematic due to the ease of manipulating the list and the difficulty of implementation. Koenigstein et al. [13] used Gnutella queries in order to build an alternative to the Billboard song ranking chart. They compared trends in sales and air-play counts, to piracy popularity trends, and showed that piracy popularity of singles by well-established artists, is highly correlated with the Billboard charts.

2.2 Previous Shared Folders Studies

First attempts to use P2P shared folders for artist similarity were presented in [3, 7]. The centralized and somewhat undersized OpenNap network was used in order to generate a similarity measurement that was based on artists co-occurrences in shared folders. The authors compared the P2P information to other similarity measurements such as social tags in [7], and also Gaussian mixtures over MFCCs and playlists co-occurrences in [3]. The evaluation was done against survey data, and similarities were measured by a pre-determined similarity function.

We took the same approach of evaluating data against a human based survey. The evaluation in this paper is based on the Partial Order Embedding (POE) algorithm of [18], which learns an optimized artist similarity space from labeled (partially ordered) examples. The key difference between the evaluation in [3] and the present work is that we report accuracy achievable by an optimized similarity function, whereas [3] relies on a fixed similarity function. The results in Section 4 emphasize the importance of training the embedding before evaluating with a human based survey. The scale of the data-set used here (13.8 million user-to-song relations after processing), is much higher than in [3,7] (400K user-to-song relations after processing), although our experimental results are restricted to a subset for evaluation purposes.

The first working recommender system based on P2P information was demonstrated in [25]. Shared folders data from the Gnutella network was used in order to generate a user-to-artists matrix. The artists were clustered using k-means algorithm, and recommendations were done from the centroid or from the nearest neighbor.

3. DATA COLLECTION METHODOLOGY

The practice of collecting information from file-sharing networks is relatively common in the field of computer communication. P2P measurement techniques fall into five basic categories:

1. **Passive Monitoring:** Monitoring P2P activity by analyzing data from a gateway router.

2. **Participate:** Developing a client software that can capture and log interesting information [13, 14, 17].

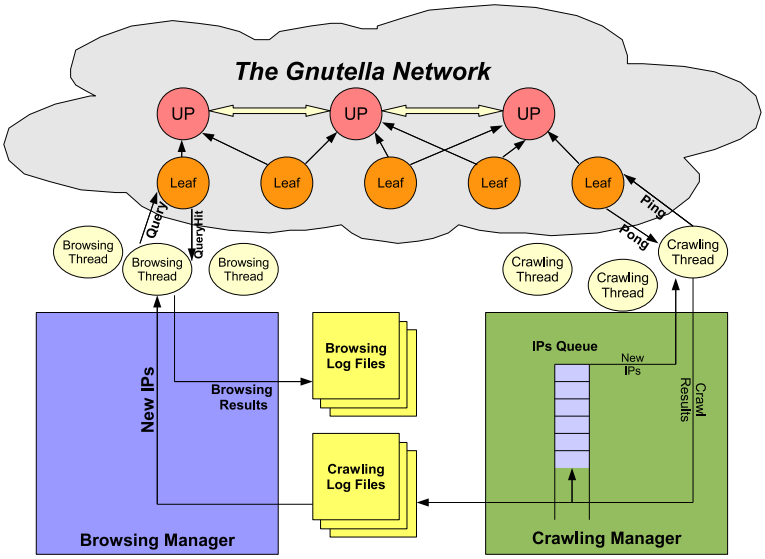


Figure 1. Crawling and Browsing in a Two-Tier Gnutella Segment

3. **Crawl:** Developing a crawler which recursively "walks" the network by asking each peer for a list of its neighbors [15, 16, 25].
4. **Sample:** Sampling a set of peers and gathering static peer properties [4, 5].
5. **Central:** Study information gathered from a central entity in the network [3, 7].

The data collection system described here belongs to the third category. We crawled the Gnutella file-sharing network as described below.

3.1 The Gnutella Network

Gnutella started its operations on March 2000, as the first decentralized file-sharing network. It is arguably the most academically studied file-sharing network [1, 9, 11, 23, 24, 27, 28].In late 2007, it was the most popular file-sharing network on the Internet with an estimated market share of more than 40% [21], serving millions of users.

Modern Gnutella, as well as other popular P2P file-sharing applications, adopted a two-tier topology. In this architecture, a small fraction of nodes, called *ultrapeers*, form an ad-hoc top-level overlay whereas the remaining nodes, called *leaves*, each connect to the overlay through a small number of ultrapeers. Ultrapeers belong to regular users with higher computing and network resources. These nodes route search requests and respond to other users who connected to them. Ultrapeers typically have a high degree (i.e., maintain 30 neighbors) in order to keep a short path lengths between participating peers [28]. We crawl both leaves and ultrapeers in a similar manner.

3.2 Crawling the Network

P2P crawlers operate in a similar way to web crawlers. The crawler treats the network as a graph. The starting points of the crawling operation are taken from an offline initialization list of known hosts. This initialization list must contain some redundancies, because unlike web crawling, the Gnutella nodes might be offline and therefore unresponsive. To maximize the performance of the highly parallelized architecture of the crawler, we used a very large initialization list of 104,767 IP addresses. This allows us to make use of all the crawling clients right at the beginning of the crawling operation [1] .

Figure 1 depicts the crawling and browsing operations in a two-tier Gnutella segment. The crawling process is a breadth-first exploration, where newly discovered leaves and ultrapeers are enqueued in a list of un-crawled addresses (The *IPs Queue*). The parallel crawling threads constantly ask the *Crawling Manager* for new IP addresses from the queue, and send back newly received results. The results are stored in text log files, and new IPs are enqueued in the IPs Queue.

Gnutella's "Ping-Pong" protocol is used by the crawling threads to discover new Gnutella nodes in the network. A node receiving a "Ping" message is expected to respond with one or more "Pong" messages. A "Pong" message includes the address of a connected Gnutella node and information regarding the amount of data it is making available to the network. An incoming Ping message with TTL = 2 and Hops = 0 is a "Crawler Ping" used to scan the network. It should be replied to with Pongs containing information about the node receiving the Ping and all other nodes it is connected to. More details about the the Gnutella protocol can be found in [29].

The crawling of large scale *dynamic* networks, such as file-sharing networks never reaches a full stop. As clients constantly connect and disconnect from the network, the crawler will always discover new IP addresses. We thus use two stopping conditions: A time constraint (typically 1 hour), or reaching a low rate of newly discovered nodes, which indicates the completion of a crawl. In the beginning of a crawl, the rate of newly discovered nodes increases dramatically and typically reaches over 300,000 new clients per minute. As the crawling process proceeds,

[1] Such a large list of IP addresses can be easily generated from the results of a previous crawling operation.

discovery rate slows down until it reaches a few hundreds per minute. At this point, the networks is almost fully covered, and the newly discovered nodes are mostly the ones that have joined the network only after the crawling operation started.

3.3 Browsing Shared Folders

The browsing operation begins shortly after the crawling operation started. Once the first crawling log file is created, the *Browsing Manager* can start assigning IP addresses (taken from the crawling logs) to the browsing threads. The browsing threads send "Query" messages to the Gnutella nodes, and wait for a "QueryHit" message in return. Query messages with TTL=1, hops=0 and Search Criteria="␣␣␣␣" (four spaces) are used to index all files a node is sharing. A node should reply to such queries with all of its shared files. The sharing information is stored by the *Browsing Manager* in the browsing logs. These files are used to generate the CF data.

4. EVALUATION

To evaluate the information content of the present P2P data, we test its performance on an artist similarity prediction task. [18,19] developed the Partial Order Embedding (POE) algorithm for integrating multiple data sources to form an optimized artist similarity space, and applied it to acoustic models (Gaussian mixtures over MFCCs and chroma vectors), semantic models (semantic multinomial auto-tags, and social tags from Last.fm [2]), and text models (biography data) [3]. By applying the same algorithm to collaborative filtering data, we can evaluate the amount of high-level artist similarity information captured by P2P collaborative filtering data, and quantitatively compare it to alternative data sources.

In general, collaborative filtering has been repeatedly demonstrated to be an effective source of information for recommendation tasks (see, e.g., [2, 12]). One may then wonder how one source of collaborative filtering data compares to another. Because [18] did not include collaborative filtering in their experiments, there is no existing baseline to compare against for the artist similarity task. We therefore repeat the experiment with the Last.fm collaborative filtering data provided by [6], allowing us to quantitatively compare P2P data to a more conventional source of collaborative filtering information.

We sampled 100K (7.69%) users out of over 1.3 million Gnutella users recorded on a single 24 hours crawl. We filtered the files that correspond to musical files according to file suffix (.mp3 files). In the entire (1.3 million users) data set, we identified 531,870 different songs. In our 100K users sample, we identified 511K songs, a value that is not much lower than the total number. Ambiguities in artist names due to typos and misspellings were corrected by majority voting. After this step, we had 13,839,361 user-to-song relations, which was the base for a collaborative matrix (ARTISTSxUSERS). The artist names are the same as in the aset400 data set of Ellis and Whitman [7]. These

[2] http://last.fm/

[3] The data from [18] can be found at http://mkl.ucsd.edu/.

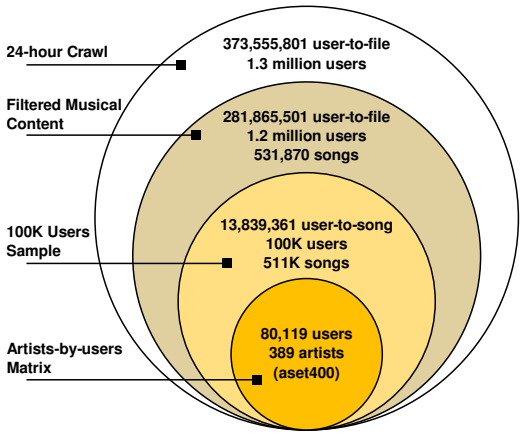


Figure 2. A quantitative summary of the data-set scale after each processing stage

artists were found in the shared folders of 80,119 users. The above numbers are summarized in Figure 2. Our similarity matrix will be available on the authors website by publication time.

4.1 The embedding problem

Formally, the goal in this experiment is to learn an embedding function $g : \mathcal{X} \rightarrow \mathbb{R}^n$, which maps a set of artists $\mathcal{X}$ into Euclidean space. The embedding is trained to reproduce relative comparison measurements (i, j, k), where (i, j) are more similar to each-other (i.e., closer) than (i, k).

Each artist is represented as a vector in some feature space, and the embedding function is parameterized as a linear projection from that feature space to the embedding space. This can be expressed in terms of inner products:

$$g(i) = NK_i,$$

where N is a linear projection matrix to be learned, and K_i is a vector containing the inner product of i's feature vector with each other point in the training set. As described in [18], this readily generalizes to non-linear kernel functions and heterogeneous data sources, but we do not make use of these extensions in the present experiment.

To summarize, given a set of training artists, relative similarity measurements between the artists, and a feature representation of each artist (equivalently, a kernel matrix over the training artists), the algorithm finds a linear projection matrix N which attempts to satisfy the similarity measurements under Euclidean distance calculations:

$$(i, j, k) \Leftrightarrow \|N(K_i - K_j)\| < \|N(K_i - K_k)\|.$$

The matrix N is found by solving a convex optimization problem, which involves three competing terms:

$$\max_W \sum_{i,j} \|K_i - K_j\|_W^2 - \beta \cdot \sum_{ijk} \xi_{ijk} - \gamma \cdot \mathrm{tr}(WK)$$
$$\|K_i - K_j\|_W^2 \doteq (K_i - K_j)^\mathsf{T} W (K_i - K_j),$$

where W is a positive semi-definite matrix which can be factored to recover the projection matrix: $W = N^\mathsf{T} N$.

The first term maximizes the variance of the data in the embedding space, which prevents points from being collapsed onto each-other.

The second term tries to minimize the number of ordering mistakes made by the embedding function. This is accomplished by using a slack variable $\xi_{ijk} \geq 0$ for each triplet constraint (as in support vector machines), allowing for margin violations:

$$\|K_i - K_j\|_W^2 \leq \|K_i - K_k\|_W^2 + 1 - \xi_{ijk}.$$

Finally, the third term limits the complexity of the learned space by penalizing a convex approximation to the rank of the embedding space. For more details about the optimization procedure, see [18].

At test time, similarity queries are presented in a similar form: (q, i, j), where q is previously unseen, and i and j come from the training set. The query artist is mapped into the embedding space by first computing inner products to the training set, resulting in a vector K_q, and then projecting by N: $g(q) = NK_q$. Once in the embedding space, distances are calculated to i and j, and the similarity prediction is counted as correct if the distance to i is smaller than the distance to j.

4.2 From P2P to artist similarity

In order to apply the POE algorithm to collaborative filtering data, we need to define a kernel function between artists in terms of the collaborative filtering matrix. One straightforward choice of kernel function is to simply count the number of users shared between two artists i and j. However, this may suffer from popularity bias if i has many users and j has relatively few. To counteract this, we normalize each artist by the number of users to which it is matched. This gives rise to the kernel function:

$$k(i,j) = \frac{\#\text{users for } i \text{ and } j}{(\#\text{users for } i) \cdot (\#\text{users for } j)}.$$

Equivalently, we can interpret this kernel function as the cosine-similarity between *bag-of-users* representations of artists i and j, i.e., an artist is represented by a binary vector where coordinate z is 1 if user z is present and 0 otherwise. This is similar to the bag-of-words representation commonly used in text applications, and like in text, the dimensionality of the feature representation is much larger than the number of data points (i.e., there are many more users than artists). Consequently, it is more economical to use the kernel matrix representation than to work directly on the feature vectors.

4.3 Results

We reproduced the main experiment of [18], using P2P collaborative filtering data, as well as listener data from Last.fm [6]. We first pruned both data sets down to the 412 artists of aset400 [7]. Of these artists, 23 were missing from P2P, and 5 were missing from Last.fm. Nonetheless, we retain similarity measurements for these artists to maintain comparability with the previously published results.

As in [18], the artists (and corresponding similarity measurements) are split by 10-fold cross-validation, and the training and test procedure is repeated for each fold. We then calculate the accuracy of the learned embeddings, averaged across all folds. Results are presented in Table 1.

Data source	Native	Learned	Restricted
P2P	0.561	0.728	0.741
Last.fm	0.570	0.760	0.763
MFCC	0.535	0.620	
Biography	0.514	0.705	
Tags	0.705	0.776	

Table 1. Test accuracy for artist similarity. *Native* corresponds to similarity measurements taken from the raw kernel matrix, and *learned* corresponds to similarities learned by POE. The *restricted* column reports accuracy achieved by testing only on artists observed in the data (389 artists for P2P and 407 for Last.fm). See Section 4.3 for details.

The accuracy of similarity predictions may be skewed due to testing on artists for which the data source may have no information (i.e., no users shared songs by that artist). To quantify this effect, we also computed accuracy on similarity measurements restricted to include only those artists observed in collaborative filtering data. These results are given in the *restricted* column of Table 1.

Overall, Table 1 indicates that both P2P and Last.fm collaborative filtering data captures a great deal of high-level artist similarity information. Both sources perform comparably to highly detailed social tags (Tags), and both outperform similarity models derived from artist biographies (Biography) or acoustic content (MFCCs) as reported in [18].

In this experiment, the Last.fm data achieves slightly higher accuracy than the P2P data. However the difference is quite small, and might be eliminated by using a larger sample of P2P users (we only used 7.69%). Also note that the results dramatically improve once the embedding is trained. This emphasizes the importance of learning an optimal similarity space, rather than using a pre-determined similarity function as in [3].

5. SUMMARY

We reviewed the latest P2P based MIR studies, and presented a new Gnutella-based data collection system. We evaluated the information content of our P2P data-set on an artist similarity prediction task based on the Partial Order Embedding (POE) presented in [18], and compared it to the "traditional" data sources, such as Last.fm collaborative filtering, tags, and acoustic models. We showed that a P2P based Collaborative Filtering data-set performs comparably to "traditional" data-sets, yet maintains some inherent advantages such as scale, availability and additional information features such as ID3 tags and geographical location.

According to the International Federation of the Phonographic Industry (IFPI) 95% of all music is downloaded in file sharing networks [20]. We expect that as the practice of file-sharing becomes even more widespread, the usage of P2P based data-sets will become increasingly relevant.

6. REFERENCES

[1] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5, 2000.

[2] Luke Barrington, Reid Oda, and Gert Lanckriet. Smarter than genius? human evaluation of music recommender systems. In *International Symposium on Music Information Retrieval*, 2009.

[3] Adam Berenzweig, Beth Logan, Daniel P. W. Ellis, and Brian Whitman. A large-scale evaluation of acoustic and subjective music similarity measures. In *Computer Music Journal*, 2003.

[4] Sudip Bhattacharjee, Ram Gopal, Kaveepan Lertwachara, and James R. Marsden. Whatever happened to payola? an empirical analysis of online music sharing. *Decis. Support Syst.*, 42(1):104–120, 2006.

[5] Sudip Bhattacharjee, Ram D. Gopal, Kaveepan Lertwachara, and James R. Marsden. Using P2P sharing activity to improve business decision making: proof of concept for estimating product life-cycle. *Elec. Commerce Research and Applications*, 4(1):14–20, 2005.

[6] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[7] Daniel P. W. Ellis and Brian Whitman. The quest for ground truth in musical artist similarity. In *International Symposium on Music Information Retrieval*, pages 170–177, 2002.

[8] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression : A statistical view of boosting. *Annals of statistics*, 28(2):337–407, 2000.

[9] Adam Shaked Gish, Yuval Shavitt, and Tomer Tankel. Geographical statistics and characteristics of p2p query strings. In *International Workshop on Peer-to-Peer Systems*, February 2007.

[10] J. Grace, D. Gruhl, K. Haas, M. Nagarajan, C. Robson, and N. Sahoo. Artist ranking through analysis of online community comments. *International World Wide Web Conference*, 2008.

[11] Mihajlo A. Jovanovic. Modeling large-scale peer-to-peer networks and a case study of gnutella. Master's thesis, University of Cincinatti, Cincinatti, OH, USA, 2001.

[12] J. Kim, B. Tomasik, and D. Turnbull. Using artist similarity to propagate semantic information. In *Proc. International Symposium on Music Information Retrieval*, 2009.

[13] Noam Koenigstein and Yuval Shavitt. Song ranking based on piracy in peer-to-peer networks. In *International Symposium on Music Information Retrieval*, Kobe, Japan, October 2009.

[14] Noam Koenigstein, Yuval Shavitt, and Tomer Tankel. Spotting out emerging artists using geo-aware analysis of p2p query strings. In *The 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 937–945, Las Vegas, NV, USA, 2008.

[15] Noam Koenigstein, Yuval Shavitt, Tomer Tankel, Ela Weinsberg, and Udi Weinsberg. A framework for extracting musical similarities from peer-to-peer networks. In *IEEE International Conference on Multimedia and Expo (ICME 2010)*, Singapore, July 2010.

[16] Noam Koenigstein, Yuval Shavitt, Ela Weinsberg, and Udi Weinsberg. On the applicability of peer-to-peer data in music information retrieval research. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, the Netherlands, August 2010.

[17] Noam Koenigstein, Yuval Shavitt, and Noa Zilberman. Predicting billboard success using data-mining in p2p networks. In *ISM '09: Proceedings of the 2009 11th IEEE International Symposium on Multimedia*, December 2009.

[18] Brian McFee and Gert Lanckriet. Heterogeneous embedding for subjective artist similarity. In *International Symposium on Music Information Retrieval*, Kobe, Japan, October 2009.

[19] Brian McFee and Gert R.G. Lanckriet. Partial order embedding with multple kernels. In *Proceedings of the 26th annual International Conference on Machine Learning (ICML)*, pages 721–728, 2009.

[20] IFPI: International Federation of the Phonographic Industry. Digital music report 2009.

[21] Ars Technica Report on P2P File Sharing Client Market Share. `http://arstechnica.com/old/content/2008/04/study-bittorren-sees-big-growth-limewire\-still-1-p2p-app.ars`, 2008.

[22] J. R. Quinlan. Learning with continuous classes. pages 343–348, 1992.

[23] Amir H. Rasti, Daniel Stutzbach, and Reza Rejaie. On the long-term evolution of the two-tier gnutella overlay. In *IEEE Global Internet Symposium*, Barcelona, Spain, April 2006.

[24] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6, 2002.

[25] Yuval Shavitt and Udi Weinsberg. Song clustering using peer-to-peer co-occurrences. In *ISM '09: Proceedings of the 2009 11th IEEE International Symposium on Multimedia*, December 2009.

[26] Haijian Shi. Best-first decision tree learning. Master's thesis, University of Waikato, Hamilton, NZ, 2007. COMP594.

[27] K. Sripanidkulchai. The popularity of gnutella queries and its implications on scalability, February 2001. Featured on O'Reilly's www.openp2p.com website.

[28] Daniel Stutzbach and Reza Rejaie. Characterizing the two-tier gnutella topology. *SIGMETRICS Perform. Eval. Rev.*, 33(1):402–403, 2005.

[29] The Gnutella Protocol Specification v0.41. `http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf`, 2010.

[30] Matei A. Zaharia, Amit Chandel, Stefan Saroiu, and Srinivasan Keshav. Finding content in file-sharing networks when you can't even spell. In *IPTPS*, 2007.

COMBINED AUDIO AND VIDEO ANALYSIS FOR GUITAR CHORD IDENTIFICATION

Alex Hrybyk and Youngmoo Kim
Electrical & Computer Engineering, Drexel University
{ahrybyk, ykim}@drexel.edu

ABSTRACT

This paper presents a multi-modal approach to automatically identifying guitar chords using audio and video of the performer. Chord identification is typically performed by analyzing the audio, using a chroma based feature to extract pitch class information, then identifying the chord with the appropriate label. Even if this method proves perfectly accurate, stringed instruments add extra ambiguity as a single chord or melody may be played in different positions on the fretboard. Preserving this information is important, because it signifies the original fingering, and implied "easiest" way to perform the selection. This chord identification system combines analysis of audio to determine the general *chord scale* (i.e. A major, G minor), and video of the guitarist to determine *chord voicing* (i.e. open, barred, inversion), to accurately identify the guitar chord.

1. INTRODUCTION

The ability of an instrument to produce multiple notes simultaneously, or chords, is a crucial element of that instrument's musical versatility. When trying to automatically identify chords, stringed instruments, such as the guitar, add extra difficulty to the problem, because the same note, chord, or melody can be played at different positions on the fretboard. Figure 1 depicts a musical passage in staff notation, followed by three representations in tablature form (the horizontal lines represent the strings of the guitar, and number is the fret of that string). All of these tablature notations are valid transcriptions, in that they produce the correct fundamental frequencies as the staff notation when performed. However, only one of these positions may correspond to the original, perhaps easiest fingering

Guitar lessons are more accessible now than ever with the rise of streaming Internet video and live interactive lessons. The research presented in this paper has direct applications to these multimedia sources. A system which can automatically transcribe chord diagrams from audio and video lessons between student and teacher would be an invaluable tool to aid in the learning process.

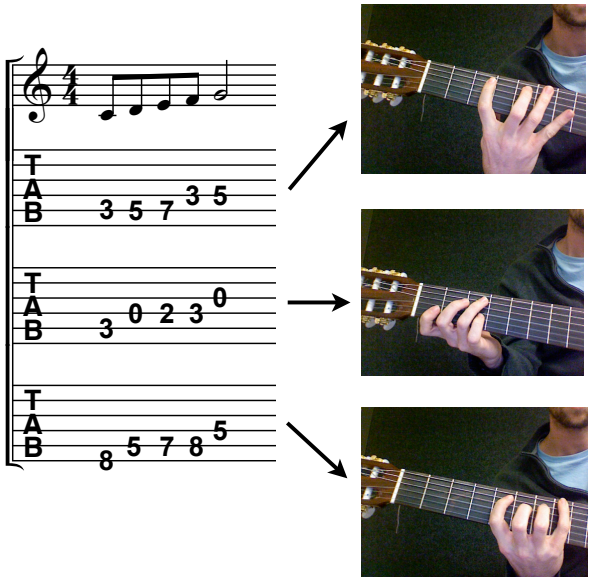


Figure 1. Three voicings of a C major scale in staff and tablature notation, shown in various positions along the guitar fretboard.

Automatic chord identification algorithms have traditionally used the chroma feature introduced by Fujishima [1]. The chroma based approach, though intuitive and easily implemented, presents many problems due to the existence of overtones in the signal. This paper avoids this problem by using a polyphonic pitch estimation method named *Specmurt Analysis* which filters out the overtones in the log-frequency spectrum to yield only a chord's fundamental frequencies [2].

Visual approaches to guitar chord and melody transcription have been attempted. Most of these methods, while accurate, are obtrusive to the guitarist; cameras must be mounted to the guitar [3], or the guitarist must wear colored fingertips to be tracked [4]. The method presented here uses brightly colored dots placed at various points along the guitar's fretboard to be tracked by the camera. These dots, which are unobtrusive to the guitarist, are used as reference points to isolate the fretboard within the image, so that principal components analysis may be used to identify the guitarist's particular voicing of that chord.

The multi-modal guitar chord identification algorithm presented in this paper is as follows: first, using Specmurt Analysis, fundamental frequency information will be re-

trieved and the general *chord scale* identified (i.e. G major, A# minor, etc.). Next, using video analysis, the guitarist's particular *chord voicing* (i.e. open, barred, inversion, etc.) will be identified using principal components analysis (PCA) of the guitarist's fretting hand.

2. RELATED WORK

The chromagram or pitch class profile (PCP) feature has typically been used as the starting point for most chord recognition systems. Fujishima first demonstrated that decomposing the discrete Fourier transform (DFT) of a signal into 12 pitch classes and then using template matching of various known chords produces an accurate representation of a song's chord structure [1].

The main problem with chroma is apparent when using template matching for various chords. For example, a C Major triad would have an ideal chroma vector of $[1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0]$. The existence of overtones in the signal cause the ideal 0's and 1's to fluctuate and create false chord identifications.

Modified versions of the chromagram, such as the Enhanced Pitch Class Profile by Lee have been introduced to ease the effects of overtones in the signal [5]. This method computes the chroma vector from the harmonic product spectrum rather than the DFT, suppressing higher harmonics making the chroma vector more like the ideal binary template. However, this method fails to identify the voicing of the chord, such as a first or second inversion.

Burns et al. developed a visual system for left-hand finger position tracking with respect to a string/fret grid [3]. Their method relies on the circular shape of fingertips, using a circular Hough transform on an image of the left-hand to detect fingertip locations with respect to the underlying fretboard. However, this method requires mounting a camera on the headstock of the guitar, which poses many problems: it can be obtrusive to the guitar player's natural method of playing, and also only captures information about the first five frets of the guitar.

Kerdvibulvech et al. proposed to track the fingering positions of a guitarist relative to the guitar's position in 3D space [4] . This is done by using two cameras to form a 3D model of the fretboard. Finger position was tracked using color detection of bright caps placed on each of the guitarist's fingertips. Again, this can hinder the physical capabilities and creative expression of the guitarist, which should not happen in the transcription process.

3. AUDIO ANALYSIS

When playing a single note, instruments produce natural harmonics (overtones) in addition to the note's fundamental frequency. Therefore, when playing multiple notes, the frequency spectrum of the audio appears cluttered, making detection of the fundamental frequencies (the actual notes) hard to locate. Saito *et al.* have proposed a technique called *Specmurt* analysis, which will be used to extract the notes of a guitar chord from the audio signal [2].

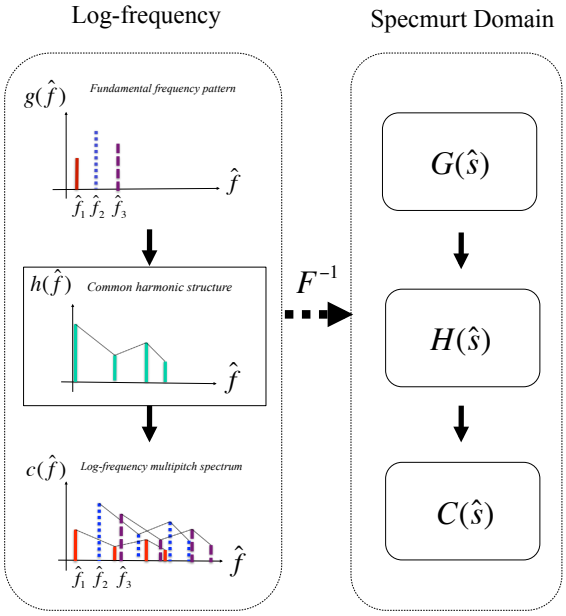


Figure 2. Log-spaced frequency domain $c(\hat{f})$ as a convolution of common harmonic structure $h(\hat{f})$ with fundamental frequency distribution $g(\hat{f})$.

3.1 Specmurt Analysis

Multiple fundamental frequency estimation using Specmurt analysis is performed by inverse filtering the log-scale frequency domain with a common harmonic structure of that instrument [2]. The resulting log-frequency spectrum contains only impulses located at the log-fundamental frequencies.

Harmonics of a fundamental frequency f_0 normally occur at integer multiples of the fundamental, nf_0. Furthermore, if the fundamental frequency changes by some Δf, the change in frequency of its respective higher harmonics will also be $n\Delta f$. By resampling the frequency domain to have a log-scaled axis, this allows the harmonics of a given fundamental to be consistently spaced by $\log n + \log f_0$, independent of fundamental frequency.

$$\hat{f} = \log f \tag{1}$$

3.1.1 Common Harmonic Structure

Using the log-scale frequency axis, we can assume that the harmonic frequencies are located at $\hat{f} + \log 2, \hat{f} + \log 3, ..., \hat{f} + \log n$. When a chord is played on an instrument, each note will presumably contain these same harmonic frequencies, beginning at different $\hat{f}$'s. Therefore, we can assume that the log-scaled multipitch spectrum, $c(\hat{f})$, is a combination of these harmonic structures, shifted and weighted differently per note. Specifically, the resulting log-scale frequency spectrum, $c(\hat{f})$, is equal to the convolution of a common harmonic structure, $h(\hat{f})$, with a fundamental frequency distribution, $g(\hat{f})$.

$$c(\hat{f}) = h(\hat{f}) * g(\hat{f}) \tag{2}$$

The harmonic structure can be written in terms of its log-frequency axis spacing, $\hat{f}_{0n}$, and its harmonic weights, W_n, where $n = 1, 2...N$ harmonics.

$$h(\hat{f}, W) = \sum_{n=1}^{N} W_n \delta(\hat{f} - \hat{f}_{0n}) \quad (3)$$

The harmonic weights will initially be a guess, which will be refined later using an iterative process to minimize the overall error of Specmurt analysis.

3.1.2 Specmurt Domain

In order to determine the desired fundamental frequency distribution, $g(\hat{f})$, one can solve (2) by deconvolving the log-spectrum with the common harmonic structure. An easier way of obtaining $g(\hat{f})$ would utilize the duality of the time/frequency-convolution/multiplication relationship (shown in Figure 2). Therefore, taking the inverse Fourier transform would yield the relationship

$$\mathcal{F}^{-1}\{c(\hat{f})\} = \mathcal{F}^{-1}\{h(\hat{f}) * g(\hat{f})\} \quad (4)$$

$$C(\hat{s}) = H(\hat{s})G(\hat{s}) \quad (5)$$

where $\hat{s}$ is a temporary Specmurt domain variable. Simple algebra followed by a Fourier tranform of $G(\hat{s})$ will yield the resulting fundamental frequency spectrum.

$$G(\hat{s}) = \frac{C(\hat{s})}{H(\hat{s})} \quad (6)$$

$$\mathcal{F}\{G(\hat{s})\} = g(\hat{f}) \quad (7)$$

The squared error after performing Specmurt analysis can be defined as

$$E(W_n) = \int_{-\infty}^{+\infty} \left\{ c(\hat{f}) - h(\hat{f}, W_n) * g(\hat{f}) \right\}^2 d\hat{f} \quad (8)$$

Minimizing the error of Specmurt is done by refining the harmonic weights, W_n, of the harmonic structure. This is done by setting the error's N partial derivatives $\frac{\partial E}{\partial W_n} = 0,\ n = 1...N$, and solving the system of equations for W_n.

The original Specmurt formulation assumed that the first weight, $W_1 = 1$, of the normalized common harmonic structure. After experimentation with various guitar signals, the higher harmonics were sometimes of larger magnitude than the fundamental frequency. By allowing the first harmonic's magnitude to vary, the algorithm was able to better identify fundamental frequencies.

4. VIDEO ANALYSIS

In order to visually identify the performing guitarist's chord voicing, the guitar fretboard must first be located and isolated within the image. However, the guitar can be held in many different orientations relative to the camera, making it difficult to find the location or coordinates of the fretboard in the image plane.

The frets of a guitar are logarithmically spaced to produce the 12 tones of the western scale. The coordinates in the (x, y) plane are plotted in Figure 3, where the x_i coordinates are related by

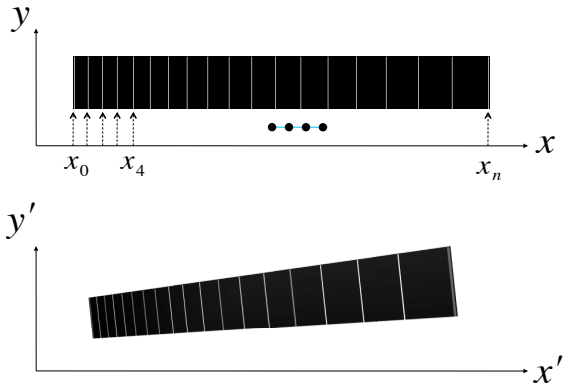


Figure 3. Ideal fretboard (top) with logarithmic x spacing of n frets, and arbitrary neck width in y direction, and seen image (bottom) with warped spacing.

$$x_i = \sum_{k=0}^{i} x_0 \times 2^{\frac{k}{12}} \quad (9)$$

4.1 Homography

Homography is the process of applying a projective linear transformation to a scene (a 2D image or 3D space), to describe how perceived positions of observed objects change when the point of view of the observer (a camera) changes. Homography will be used to determine the correct perspective transformation, i.e. rectify or warp the original image to fit the ideal fretboard spacing in Figure 3. This will make it easy to isolate the fretboard in the image for analysis. The general homography matrix equation

$$w\mathbf{p}' = \mathbf{H}\mathbf{p} \quad (10)$$

states that points in the image, $\mathbf{p}'$ can be expressed as a warping of ideal points $\mathbf{p}$ with a homography matrix $\mathbf{H}$, including a scale factor w. The homography matrix is a transformation matrix between the two images, based on which a one-to-one relationship between the features points $\mathbf{p}'$ and $\mathbf{p}$ [6]. Specifically, the points will have two dimensions, x and y, and will be expressed in terms of a 3x3 homography matrix with elements h_{ij}.

$$w \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \approx \begin{bmatrix} h_{00} & h_{10} & h_{20} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (11)$$

x_i are determined from (9) and y_i are determined as an arbitrary guitar neck width (from the ideal, rectangular fretboard). The corresponding reference points (x_i', y_i') in the image now need to be established, to compute the homography matrix, $\mathbf{H}$.

4.2 Reference Point Tracking

In order to perform the homography rectification concepts in 4.1, the correct reference points in the image must be

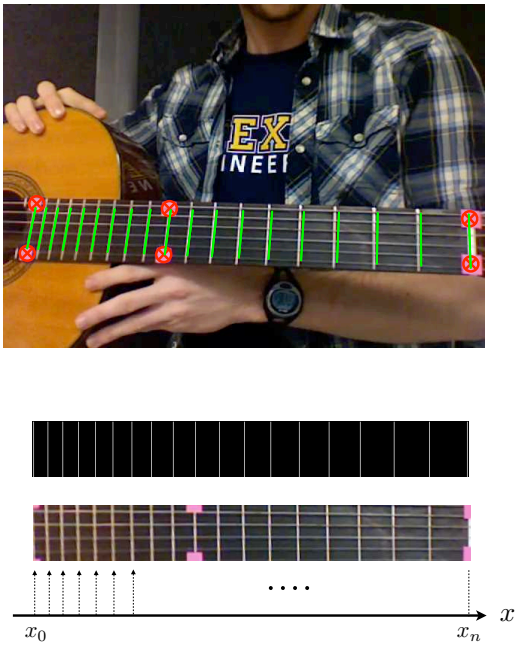


Figure 4. (top) Original image showing tracking points (in red), projected frets (in green) using the homography matrix. (bottom) Ideal fretboard, and subsection of original image after applying homography matrix to each coordinate.

determined. Attempts were made at using an iterative non-linear error minimization method, which proved initially unsuccessful (see later section 6). Instead, distinct bright colored stickers were placed at various fret locations on the neck of the guitar. The coordinates of these points (x'_i, y'_i) were tracked in each frame of video using a simple color masking followed by a k-means clustering algorithm. The small stickers were placed on the neck of the guitar on either side of the metal frets, so as not to interfere with the guitarist's playing and the timbre of the instrument.

A set of (x_i, y_i) and (x'_i, y'_i) now exist, corresponding to the frets of the guitar. The homography matrix is determined by minimizing the mean square error of (11) using these points. Applying the inverse transformation, $\mathbf{H^{-1}}$, to the ideal grid in Figure 3 yields frets that overlay perfectly with the frets in the image (Figure 4). Applying $\mathbf{H}$ to the original image and taking the subsection of coordinates yields the rectified fretboard (Figure 4), whose fret spacings are known from (9). The rectified fretboard is now isolated and in a usable form for PCA.

4.3 Determination of Chord Style

The next goal is to determine which chord voicing, given the subset of voicings that exist for a particular chord. PCA is used to decompose the rectified fretboard in its "eigen-chord" components, and determine the correct chord voicing.

Let the training set of fretboard images be $F_1, F_2...F_M$ which are vectors of length LW for an image with dimensions L by W. An example training set of fretboard images is shown in Figure 5. The average image is $A = \frac{1}{M}\sum_{i=1}^{M} F_i$, and each image with subtracted mean is $\bar{F}_i = F_i - A$. PCA seeks to find the eigenvectors and eigenvalues of the covariance matrix

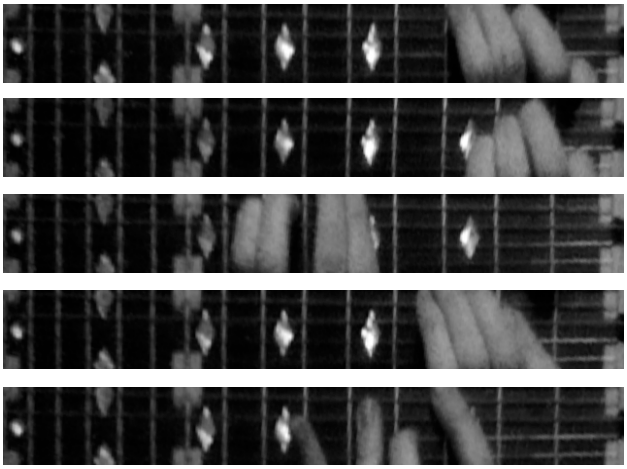


Figure 5. Example fretboard images used for training.

$$\mathbf{C} = \frac{1}{M}\sum_{i=1}^{M} \bar{F}_i \bar{F}_i^T \quad (12)$$

$$= \mathbf{S}\mathbf{S}^T \quad (13)$$

where $\mathbf{S} = [\bar{F}_1, \bar{F}_2...\bar{F}_M]$ is a set of training of images in matrix form. However, $\mathbf{C}$ is of dimension LW; the images used in this experiment are of size 80x640 pixels, and computing 51200 eigenvectors and eigenvalues is computationally intractable. Turk et. al presented a method for solving for the LW eigenvectors by first solving for the eigenvectors of an MxM matrix $\mathbf{S}^T\mathbf{S}$ [7]. The M eigenvectors v_l are used to form the eigenvectors u_l of $\mathbf{C}$.

$$u_l = \sum_{i=1}^{M} v_l \bar{F}_i \qquad l = 1...M \quad (14)$$

A new image $\tilde{F}$ can be reduced to its eigen-chord components, c_k, using the M' eigenvectors which correspond to the larger eigenvalues of $\mathbf{S}^T\mathbf{S}$.

$$c_k = u_k(\tilde{F} - A) \qquad k = 1...M' \quad (15)$$

5. EXPERIMENTAL RESULTS

Three guitarists were asked to perform a sequence of chords from chord diagrams. The chords were a selection drawn from eight scales (major and minor), each in three voicing-dependent positions: open (traditional open stringed), barred, and a 1st inversion, totaling 24 chords all together. The system was evaluated using various combinations of features derived from audio only, video only, and combinations thereof. All experiments were performed using leave-one-out training of audio and video when using PCA.

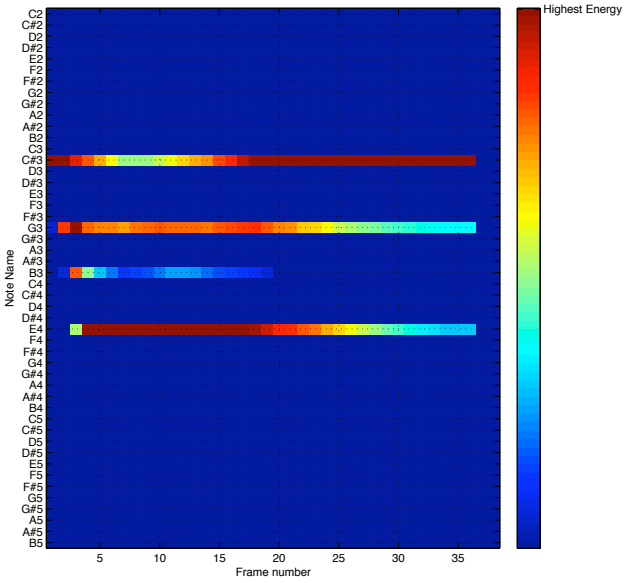


Figure 6. Specmurt piano-roll output of a C#m7♭5 jazz chord.

5.1 Audio Only

The output of Specmurt analysis is a piano-roll vector of size 48, each element corresponding to the energy of a chromatic note from C2 to B5 (4 octaves, 12 notes per octave). An example of a piano-roll vector over multiple time frames is shown in Figure 6.

Two methods were used to calculate the correctness of the chord scale and voicing using this vector. It is known what notes make up each major and minor scale. Therefore, the chord scale was evaluated by summing the energy over all octaves of the notes belonging to that chord - similar to chroma analysis. The chord scale with the highest energy was assumed to be correct, yielding an accuracy of 98.6%.

It is not deterministic, however, as to which chord voicing created a particular set of notes, or chord. For example, both the G major open chord and G major barred chord contain six notes total, all of the same fundamental frequencies, but the notes are rearranged on different strings, and hence use different fingerings. Therefore, a training set using the piano-roll energy vector was developed for each chord scale. Using PCA to identify chord voicing from the piano-roll vector shows some accuracy (62%) but is understandably low, as the difference in note energies may be very fine and inseparable for different voicings with similar notes.

	Audio only	Video only	Combined System
Scale	**98.6**	34.8	**98.6**
Voicing	62.0	**94.4**	**94.4**
Both	61.1	32.8	**93.1**

Table 1. Accuracy results for various combinations of modes of information. Combined accuracy results using Specmurt for scale identification, and video for voicing identification showing highest accuracy.

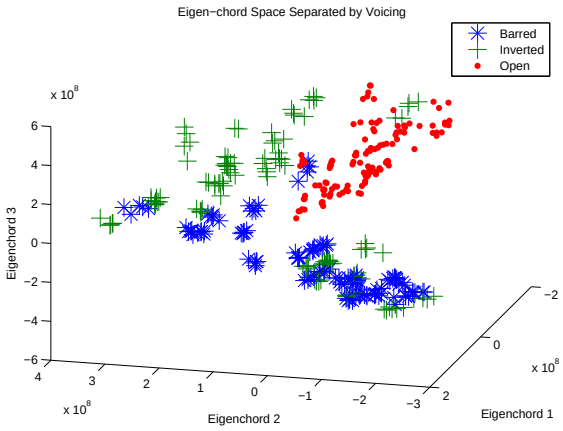


Figure 7. Three voicings from A minor, G major, and C major, after being projected into the chord-space. Various colors and symbols show the how the voicing of chords remain grouped after dimensionality reduction.

5.2 Video Only

A training set of 240 images was used to build the eigenchord space for each test. Frames of video were then projected into the chord-space using three eigen-chords of the training set using (15), and its closest centroid was assumed to be the correct chord.

Chord scale identification using only video performed extremely poorly (34%). This is expected, as the chord scale centroids in the projected chord-space after PCA are somewhat meaningless. For a particular chord scale, many different voicings exist at various points on the fretboard, which is what we hope to separate by using PCA.

For chord voicing however, very high accuracy was achieved (94.4%). Figure 7 shows how various voicings of chords, irrespective of scale, tend to group together due to the similar hand shapes used by the guitarist.

5.3 Combined System

The system which performs the best in terms of correctly identifying the overall chord (scale and voicing) utilizes the strong points of scale and voicing identification within the audio and video results. Since Specmurt analysis yielded extremely high accuracy for determining scale, it was used as a preprocessing step to voicing identification via video.

6. FUTURE WORK

The video and audio components of this guitar chord identification system have the potential to be expanded upon.

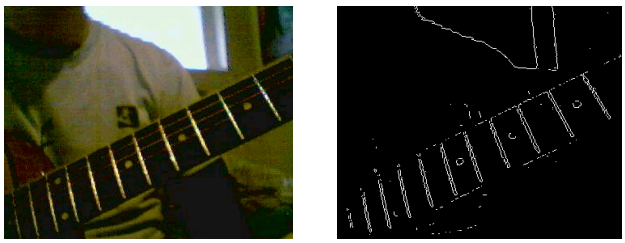


Figure 8. Guitar image (left) and edge-thresholded image (right).

6.1 Automatic Fretboard Registration

Placing colored tracking points along the neck of the guitar presents additional constraints on how the guitar fretboard can be rectified: all the tracking points must be visible in the frame of video, and nothing else in the frame may have similar color. Ideally, we would like to locate the fretboard without these points. By looking at the edge-detected image of a guitar, this produces a fairly accurate representation of where the frets are - the color of the metal frets contrasts heavily with that of the wooden neck, providing edges at frets (Figure 8).

Using the homography concept in 4.1, the points denoted as edges, $\mathbf{p}'$, should be warped using $\mathbf{H^{-1}}$ to align with the ideal fret-grid points $\mathbf{p}$. This is equivalent to minimizing an error function defined as

$$E(\mathbf{H}) = ||\mathbf{p} - \mathbf{H^{-1}}\mathbf{p}'||^2 \quad (16)$$

$$\mathbf{H} = \underset{\mathbf{H}}{\operatorname{argmin}}(E(\mathbf{H})) \quad (17)$$

After experimentation, the error function $E(\mathbf{H})$ is noticeably non-convex, and contains local minima in $\mathbf{H}$. The two fret-grids "align" in alternate orientations which are incorrect, but still minimize the error function. This area of research is being continued with the motive of constraining (16) and (17), such that the error function will always be convex, and converge to a global minimum when the two images are correctly aligned.

6.2 Larger Training Sets

Very high accuracy of video voicing identification (94.4%) was achieved using image data from only three guitarists. A more robust classifier of chord voicings could be created by collecting more data, to account for players who use non-traditional finger orientations for chords. With more data, the accuracy of determining chord scale from video may increase (34.8%), as scales may then form more meaningful distributions in the eigen-chord space.

6.3 Additional Chord Types

This system is very extendable to detect different chord scales besides major and minor. Detection of diminished, augmented, 7th, and other jazz chords are easily implemented with the chroma-style analysis of Specmurt's output, and refined search using the eigen-chord decomposition of the fretboard image.

6.4 Fusing Audio/Video Data

Currently the system uses Specmurt analysis to determine a chord's scale as a pre-processing step to eigen-chord decomposition of the fretboard to determine voicing. This means that any error introduced by Specmurt propagates throughout the rest of the system. Therefore it is desired to jointly estimate the scale and voicing together using audio and video features simultaneously.

7. CONCLUSION

This paper has presented an alternate approach to automatic guitar chord identification using both audio and video of the performer. The accuracy of chord identification increases from 61.1% to 93.1% when using audio for scale identification, and video for voicing. The "eigen-chord" decomposition of fretboard images proved extremely successful in distinguishing between a given chords voicings (normal, barred, inverted) if the chord scale is known (94.4%).

8. REFERENCES

[1] T. Fujishima, "Realtime chord recognition of musical sound: A system using common lisp music." in *Proceedings of the International Computer Music Conference*, 1999.

[2] S. Saito, H. Kameoka, K. Takahashi, T. Nishimoto, and S. Sagayama, "Specmurt analysis of polyphonic music signals," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 3, pp. 639–650, February 2008.

[3] A.-M. Burns and M. M. Wanderley, "Visual methods for the retrieval of guitarist fingering," in *NIME '06: Proceedings of the 2006 conference on New interfaces for musical expression*. Paris, France, France: IRCAM — Centre Pompidou, 2006, pp. 196–199.

[4] C. Kerdvibulvech and H. Saito, "Vision-based guitarist fingering tracking using a bayesian classifier and particle filters," in *PSIVT07*, 2007, pp. 625–638.

[5] K. Lee, "Automatic chord recognition from audio using enhanced pitch class profile," in *Proceedings of the International Computer Music Conference*, 2006.

[6] X. Wang and B. Yang, "Automatic image registration based on natural characteristic points and global homography," in *Computer Science and Software Engineering, 2008 International Conference on*, vol. 5, dec. 2008, pp. 1365 –1370.

[7] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, June 1991, pp. 586 –591.

COMBINING CHROMA FEATURES FOR COVER VERSION IDENTIFICATION

Teppo E. Ahonen
Department of Computer Science, University of Helsinki
teahonen@cs.helsinki.fi

ABSTRACT

We present an approach for cover version identification which is based on combining different discretized features derived from the chromagram vectors extracted from the audio data. For measuring similarity between features, we use a parameter-free quasi-universal similarity metric which utilizes data compression. Evaluation proves that combined feature distances increase the accuracy in cover version identification.

1. INTRODUCTION

Measuring similarity in music is an essential challenge in music information retrieval (MIR). However, the definition of similarity is not trivial. Clearly, pieces of music from the same genre are similar in various features such as orchestration, but the essential similarity of the compositions can vary largely within the genre.

Cover version identification provides a valid, objective way to estimate how well similarity in music can be recognized and measured. Cover versions often differ in various musical features, but can still be distinguished to be different performances of one composition by a human listener. Thus, successful cover version identification yields important information on how similarity in music can be measured and how features affecting the similarity can be represented.

We approach the problem of cover version identification by taking into account several features derived from the chromagram. These features are represented using different kinds of discrete alphabets and the similarity between features is calculated using a similarity metric called normalized compression distance (NCD) [9]. Evaluation shows that when using NCD for cover version identification, better identification accuracy can be obtained by taking several features into account instead of just focusing on a single feature.

Cover version identification is an objective way to estimate the performance of a retrieval system based on musical similarity. Cover versions, especially in popular music, are often intentionally different from the original recordings of the composition. Changes are common in such features as the musical keys, structures, tempos and arrangements. Also, the lyrics can be altered, translated to another language, or completely discarded. It is more difficult to estimate the features which do not change, but usually these are the melodic and harmonic features.

When successful, cover version identification provides a reliable content-based way to measure the essential similarity in different pieces of music. This provides various potential targets of applications for such systems and algorithms, ranging from end users to music researchers.

In recent years, cover version identification has gained a significant amount of interest from the MIR community. Although a relatively short time has passed since the problem of cover version identification was addressed, the problem has been studied extensively and with various different kinds of approaches.

The most important feature in cover version identification is the chromagram. Chromagram, also known as the pitch class profile, is a sequence of 12-dimension vectors which describe the relative energy of each semitone pitch class. As such, chromagram captures important tonal information and represents the harmonic and melodic content of the audio file.

Various different methods for measuring similarity between chromagrams or features derived from chromagrams exist. These include dynamic time warping and other edit distance variants, dot product and cross correlation. For an extensive and comparative review on different cover version identification approaches, we refer to [20].

The MIREX (Music Information Retrieval Evaluation eXchange) is a community-driven effort providing evaluation for different MIR applications. Cover version identification has been a MIREX task since 2006, and through the years, several different approaches have participated in the evaluation and significant improvement in the identification performance can be perceived. In 2009, the best performing cover version identification application performed with a mean of average precision value of 0.75 [1], suggesting that there still are several unsolved problems in cover version identification which need to be addressed until the problem can be declared solved.

We propose an approach that uses a similarity metric called normalized compression distance (NCD) [9] for measuring the similarity between features extracted from the

[1] http://www.music-ir.org/mirex/2009/index.php/Audio_Cover_Song_Identification_Results

audio files. For features, we extract several different representations from the chromagram vectors. As data compression works with discrete symbols, we use several different techniques for quantizing the continuous chroma values. Our starting point is that different representations have more distinguishing power when combined than they have when used alone. Also, we assume that when using NCD the chromagram cannot be quantized into a representation which both contains all the required information and is not too noisy. Thus, different features must be represented and measured on their own.

The rest of this paper is organized as follows. In Section 2, we give a brief tutorial on the concepts and theories behind the normalized compression distance. In Section 3 we describe the chroma features we use for identification. The approach is evaluated in Section 4. Finally, we present conclusions and discussion in Section 5.

2. NORMALIZED COMPRESSION DISTANCE

Normalized compression distance (NCD) is a distance metric that has its roots in information theory. The idea is to measure the information in an object using Kolmogorov complexity, the length in bits of the shortest binary program that produces the object as an output. Based on the Kolmogorov complexity, a universal information distance can be calculated. This distance, called normalized information distance [9], is denoted

$$NID(x,y) = \frac{max\{K(x|y), K(y|x)\}}{max\{K(x), K(y)\}} \quad (1)$$

where $K(x)$ is the Kolmogorov complexity of the string x and $K(x|y)$ is the conditional Kolmogorov complexity, meaning the length of $K(x)$ given the information of y.

However, Kolmogorov complexity is non-computable, and thus the normalized information distance cannot be calculated. However, Kolmogorov complexity can be approximated using any standard lossless data compression algorithm. The better the compression of a string is, the closer the approximation is to the Kolmogorov complexity.

The normalized compression distance approximates the Kolmogorov complexity with the aid of a data compression algorithm. For strings x and y, the NCD is denoted

$$NCD(x,y) = \frac{C(xy) - min\{C(x), C(y)\}}{max\{C(x), C(y)\}}, \quad (2)$$

where $C(x)$ is the length of the string x when compressed using a standard lossless data compression algorithm C and xy is the concatenation of the two strings.

NCD is proven to be robust against noise in the data [8], and studies have proven that observing several common pitfalls of the compression algorithms will help to evade problems when measuring the distances [7]. Especially, PPM-based (Prediction by Partial Matching) compression algorithms have been proven to be resistant against noise [8] and perform well in NCD calculation despite the lengths of the files [7].

Normalized compression distance has been used for several tasks in MIR. In the symbolic domain, there has been research at least in melody classification [16], genre classification [9], composer classification [9] and piano music classification [10]. In the audio domain, NCD has been applied for tasks such as structure-based clustering [3], genre classification [6, 17], cover version identification [1] and query by example [12].

3. CHROMA FEATURES

The chromagram seems to be the only valid feature to be used for cover version identification. For example, the MFCC vectors capture the timbral information of the audio file, but this information has very little help in identifying cover versions. The chromagram is robust against the changes in instrumentation and dynamics, and it captures both melodic and harmonic information from the audio file.

The easiest way to measure similarity between chromagrams using NCD would seem to be converting the chromagram into a sequence of characters and calculating the distance between these. However, we noticed that this approach has several drawbacks. If the alphabet used in sequences is small, the information contained in the chromagram will be too reduced and different sequences will turn out too similar, making distinguishing the sequences challenging. A large alphabet that contains most of the information of the chromagram, on the other hand, will make sequences noisy and lead into insignificant compression and thus into impractical identification. Our solution is to extract various feature sets of the chromagram and measure the similarities between each set.

For obtaining chromagram from the audio file, we use MIRToolbox [15], version 1.3. The window length for the Fourier transform needed in obtaining the chromagram is 0.1858 seconds and the hop factor is 0.875. We use a four-octave range of transformation with a minimum frequency of 55 Hz.

We do not have any tempo estimation and beat averaging over the chromagram frames. This is based on the assumption that unsuccessful tempo estimation might lead to even noisier representations and thus to worse identification results. A similar observation was made in [2], where frame-based identification yielded better results than the tactus-based version. Also, in [1] it was suggested that the shorter chroma sequences produced by the beat averaging may have a negative impact on the NCD values, because the error between $K(x)$ and $C(x)$ minimizes as the file length increases [9].

For compression, we use the PPMZ compression algorithm. The PPMZ is a statistical, more efficient compression algorithm than the more commonly used gzip and bzip2. Thus, it provides a better approximation of the Kolmogorov complexity. This may not, however, lead automatically into better NCD values, as the improvements in compression may be different for the different items in the formula and thus cause the NCD value to move away from the NID value [9].

3.1 Chroma Sequence Labeling

In order to measure similarity successfully with a compression algorithm, the continuous chroma vectors need to be quantized. Out of the several existing quantization methods, the hidden Markov model (HMM) has the advantage of taking into account the temporal statistics. The HMM approach has been studied extensively in converting chroma vectors into a discrete representation, and it is a common method when estimating a chord sequence representation from the harmonic content of the audio. The approach can be described as a process of using the chroma vectors as observations for a HMM whose each state represents a triad chord, training the model with the expectation-maximization (EM) algorithm, and finally obtaining the state transition path using the Viterbi algorithm.

Out of the several different methods, we use the one suggested by Bello and Pickens [4]. This means initializing the state transition parameters according to a double-nested circle of fifths and selecting the mean vectors and the covariance matrices on the basis of musical knowledge. When training the model with the EM algorithm, we train only the state distribution and transition parameters and leave the observation parameters untrained.

The 24-chord estimation provides a robust but slightly noisy representation of the harmonic content of the audio file. When observing the representations we noticed that the estimated chords were occasionally oscillating between major and minor chords of the same root note. This suggests that the third of the chord can harm the sequence labeling. Similar observation can be derived from the MIREX chord detection task where average overlap scores usually become better when the major and minor chords are merged (see for example the results of the MIREX Chord Detection Task 2009 [2]). This led us to an experiment with a 12-state HMM, where the triad of the chord is discarded from the chord templates. In the 12-state HMM, the initial parameters are set in a similar manner as with the 24-state HMM, but with respect to the simpler model and reduced chords. As such, the state sequence provided by the Viterbi algorithm can be seen as a "power chord" representation. Such representation is clearly too reduced and inaccurate to distinguish the versions on their own, but it seems to improve the identification performance when used in parallel with the 24-state HMM representation. In Figures 1 and 2 we display state sequences derived from a single audio file using 24- and 12-state HMMs, respectively.

3.2 Chromagram Flux

In addition to the chromagram vectors themselves, we experimented on whether the distance between subsequent chromagram vectors might have any effect. A somewhat similar approach was presented in [14], where a 12-dimension dynamic chroma vector feature called delta chroma was utilized. The delta chroma describes the degree of chroma changes on all possible intervals.

Here, we do not consider the delta chroma, but instead we calculate the distances between successive chroma vectors. A similar approach was utilized in [21], where correlation between adjacent chroma vectors was used as a feature in identification. We discovered empirically that the Manhattan distance (the city-block distance) had more distinguishing power for our work than Eucledian or cosine distances.

The Manhattan distances between chroma vectors of a musical piece can be seen as a time series. To discretize the time series, we use SAX (Symbolic Aggregate approXimation) [18]. In short, SAX discretizes the continuous values by first reducing their dimensionality using piecewise aggregate approximation and then discretizing the values according to a Gaussian curve. We chose SAX after experimenting with several quantization methods. Also, SAX has been used successfully for quantization when calculating similarity between time series using NCD [13].

Selection of the SAX parameters is not a trivial task. As we want to represent the whole chromagram flux as a string of characters, the sliding window is set to the length of the chromagram. The alphabet size and SAX accuracy parameters are more difficult to choose. We set the alphabet size to four and the number of frames per character to ten. These were chosen empirically, and thus are open to discussion.

3.3 Strongest Tone Sequence

The chromagram represents not only harmonic, but also melodic information contained in the audio file. We tested several methods to have more melodic information from the chromagram to be presented in a format suitable for NCD, but as with the chromagram quantization, different representations proved either to be too noisy or too reducing.

However, a straightforward way to represent some of the mid-level melodic information proved to increase the identification accuracy. We took the index of the strongest pitch class of a chroma vector (for a normalized chroma vector, the pitch class with the value of one), and represented the piece of music as a sequence of the strongest pitch class components. For a less densely orchestrated piece of music, this representation provides some information of the predominant melody of the piece. Even with more dense arrangements, it provides a representation that displays information different from the sequence labeling.

3.4 Transposition

Because cover versions are occasionally performed in a different key, the distance between chroma features can turn out large if key invariance is not addressed, even if the chroma features would otherwise be fairly similar. To obtain key invariance, a possible solution is to calculate distances between all 12 transpositions of the candidate version, but this is time-consuming. Another solution is to transpose the chromagrams into a common key using key estimation, but as with the tempo estimation, key estimation can fall short and lead to even worse identification results. We do not estimate the keys from the chromagrams,

[2] http://www.music-ir.org/mirex/2009/index.php/Audio_Chord_Detection_Results

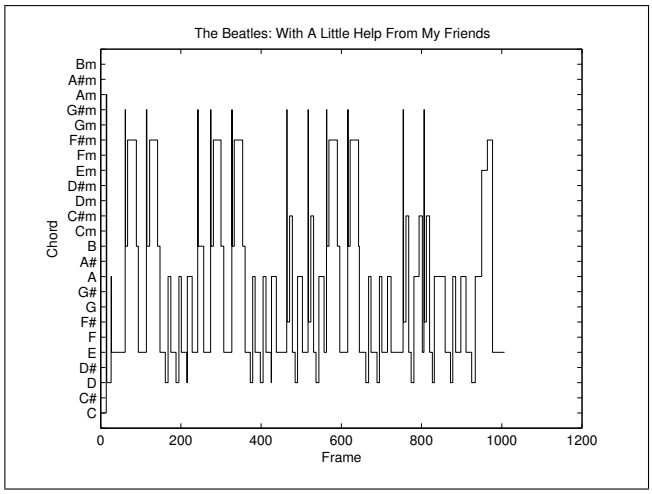


Figure 1. 24-state HMM Viterbi path for *With A Little Help From My Friends* performed by The Beatles.

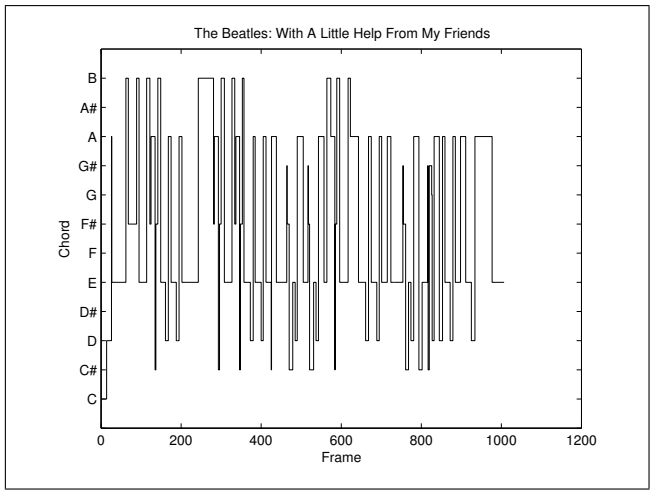


Figure 2. 12-state HMM Viterbi path for *With A Little Help From My Friends* performed by The Beatles.

but instead use Optimal Transposition Index (OTI) [19] to transpose the chroma sequences into a common key. In OTI, the transposition index is selected by first taking the global chroma vectors (by summing and normalizing the chroma vectors) of the two pieces of music. Then, the transposition index is selected by rotating the candidate global chroma vector 12 times and calculating the dot product between each pair of the target and candidate global chroma vectors. The rotation with the highest dot product is selected as the transposition index and the whole chromagram of the candidate is rotated according to the index. Fast and straightforward, OTI has also been proven to provide better identification accuracy than using the key estimation [19]. We apply OTI before any feature extraction.

3.5 Total Distance

After the distances between all the features of the pieces of music are calculated, the total distance for a pair of performances is obtained by simply taking the mean of all the feature distances. The distances could be weighted according to the importance of the features. To reduce the possible bias in the mean values caused by outliers, we also measured the total distance as the median of all measured feature distances.

4. EVALUATION

4.1 Test Data

To evaluate the performance of our approach, we collected a data set of original performances and their cover versions. For each original piece of music we included five cover versions. The data set included 25 such six-song sets and to complete the collection, a total number of 600 unique pieces of music were included, thus making the collection a total of 750 pieces of music with 150 possible queries.

The material was obtained from personal music collections and contains mostly western popular music, but with cover versions ranging from classical music renditions to world music and electronic versions. Apart from studio cover versions by different artists, the data set also includes live versions and a few remixes of the original versions. The complete detailed content of the data set can be requested from the authors.

4.2 Results

We used each of the 150 versions in the dataset as a query. From the output distance matrix, we calculated the total number of identified covers in the top five (TOP-5), the mean of average precisions (MAP) and the mean rank of the first identified cover (RANK). The results, using the mean as the total distance, are depicted in Table 1.

To present the effect of each different feature in the identification, we ran the algorithm for the whole test data set using only selected features of the feature set. The results for different feature sets, using the mean as the total distance, are depicted in Table 2.

The difference between using the mean and median values as the total distance is depicted in Figure 3. Generally, using the median as the total distance provided smaller distances. This suggests that outliers do exist in the feature distances and overall identification could be improved by taking them into account. However, using the mean as the total distance provided slightly better identification accuracy with a TOP-5 rating of 263 against the TOP-5 rating of 243 of the median distance.

4.3 Comparison to the LabROSA Cover Song Detection System

To see how well our approach performs in comparison with another cover version identification approach, we ran our test data with the LabROSA Cover Song Identification software [11]. To our knowledge, this is the only cover version identification application that is freely distributed and available online [3] .

[3] http://labrosa.ee.columbia.edu/projects/coversongs/

Measure	Value	Range
TOP-5	263	[0–750]
MAP	0.410	[0–1]
RANK	4.795	[1–745]

Table 1. Results of the 150 query evaluation.

Features used	TOP-5	MAP
24-state HMM	216	0.356
24- and 12-state HMM	242	0.378
HMMs and Chroma flux	249	0.399
All features	263	0.410

Table 2. The effect of combining different features.

The comparison between the results of our approach and the LabROSA application is depicted in Table 3. The results show that the performance of our application is comparable with the performance of the LabROSA system. However, we are aware that the LabROSA application was introduced several years ago and is possibly not comparable with some of the state-of-the-art approaches. For comparing the performance of our approach with more state-of-the-art approaches, we refer to the future MIREX cover song identification task where our application will be submitted.

5. CONCLUSIONS

We have presented an approach for cover version identification that combines different features derived from the chromagrams extracted from the audio files. To discretize continuous values, several techniques such as HMM and SAX have been used. The similarity between discretized features is calculated using a distance metric called normalized compression distance, which uses data compression to approximate the Kolmogorov complexities of the objects and as such is a quasi-universal, parameter-free similarity metric.

Based on the results, it is evident that the chroma feature combination together with the NCD metric can be used for cover version identification. As our results proved, combining different features and composing the final distance based on the distances between these features provides more accurate identification with the NCD. The algorithm was tested with competent results against a large data set consisting of various different kinds of versions from original performances.

The biggest obstacle for using normalized compression distance for cover version identification is the process of converting continuous features to discrete representations. Extracting features from audio is likely to yield noisy representations, and although NCD has been proved to be resistant against noise [8], it still affects the identification.

Our approach has more emphasis on the harmonic features, and observing the results supports this: pieces of music with distinctive, recognizable harmonic content are easily identified even when arrangements and structures vary. Also, as stated, we comprise the total distance simply as a mean of all distances, but this could be improved by weighting the different distances according to their relevance. Using the median as the total distance also gave a finding of the bias caused by the outliers.

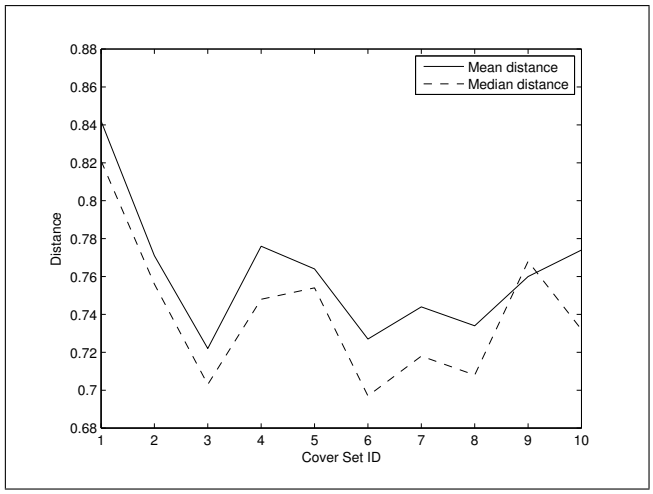


Figure 3. Mean distances for the first identified covers in ten 6-version cover sets using mean and median total distances.

Another issue demanding attention is that the phases of the measuring process each have a wide selection of parameters. Parameter selection is present in every phase of the identification process: from selecting the parameters of the Fourier transform when obtaining the chromagram to the choice of the compression algorithm used for calculating the NCD values. It is unclear if the parameters we have selected are optimal for the identification task and also the possibility of overfitting is evident. Future work addressing the parameter selection is under consideration.

5.1 Remark on Different Versions

As the cover version dataset also included live renditions and remixed versions of the original recordings, we took a closer look at the cases of these versions.

Live versions, either by the performers of the original versions or by a different performer, were in most cases identified very well. We see two reasons for this. First, live versions are often quite similar to original versions, having only slight modifications such as key, tempo or small structural differences (lengthier introductions or solo sections). Second, the live versions are less densely produced and arranged, whereas the studio versions are usually far more orchestrated. This makes the chroma features derived from live versions less noisy, which in turn benefits the similarity measuring. All in all, live version detection can be seen as a somewhat easier case of cover version identification. Thus, developing and testing cover version identification algorithms using predominantly live renditions may lead to slightly biased results.

Remixed versions, on the other hand, were often far more difficult to identify. In many cases, remixed versions

System	TOP-5	MAP
Our approach	263	0.410
LabROSA	256	0.405

Table 3. The results between our approach and LabROSA system.

share only limited elements similar to the original performance, usually combining audio elements of the original performances with completely different, and often electronic, instrumentation. Whereas live versions usually have very little changes in structures and a stripped-down instrumentation, the situation is often completely vice versa with remix versions: the original structure is often completely discarded and the instrumentation is usually even more dense than the original performance. We feel free to say that remix version identification is a far more difficult case of cover version identification. Thus, it would be interesting to see how well cover version identifiers perform when the task is specifically remix version identification. To our knowledge, version identification specialized in remix identification has been done only on a small scale [5].

6. ACKNOWLEDGEMENTS

This work has been supported by the Academy of Finland, grant #129909. The author would like to thank Kjell Lemström and Rainer Typke for their insightful comments on earlier versions of this paper. The author is also grateful to Päivi Suomalainen for her valuable help in collecting and organizing the test data.

7. REFERENCES

[1] T. E. Ahonen. Measuring harmonic similarity using PPM-based compression distance. In *WEMIS'09*, Corfu, Greece, 2009.

[2] J. P. Bello. Audio based cover song retrieval using approximate chord sequences: Testing shifts, gaps, beats and swaps. In *ISMIR'07*, Vienna, Austria, 2007.

[3] J. P. Bello. Grouping recorded music by structural similarity. In *ISMIR'09*, Kobe, Japan, 2009.

[4] J. P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *ISMIR'05*, London, UK, 2005.

[5] M. Casey and M. Slaney. Fast recognition of remixed music audio. In *ICASSP-07*, Hawaii, USA, 2007.

[6] Z. Cataltepe, Y. Yaslan, and A. Sonmez. Music genre classification using MIDI and audio features. *EURASIP Journal on Applied Signal Processing*, 2007(1), 2007.

[7] M. Cebrián, M. Alfonseca, and A. Ortega. Common pitfalls using the normalized compression distance: what to watch out for in a compressor. *Communications in Information and Systems*, 5(4):367–384, 2005.

[8] M. Cebrián, M. Alfonseca, and A. Ortega. The normalized compression distance is resistant against noise. *IEEE Transactions on Information Theory*, 53(5):1895–1900, 2007.

[9] R. Cilibrasi and P. M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[10] R. Cilibrasi, P. M. B. Vitányi, and R. de Wolf. Algorithmic clustring of music based on string compression. *Computer Music Journal*, 28(4):49–67, 2004.

[11] D. P. W. Ellis and G. E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *ICASSP-07*, Hawaii, USA, 2007.

[12] M. Helén and T. Virtanen. A similarity measure for audio query by example based on perceptual coding and compression. In *DAFx-07*, Bordeaux, France, 2007.

[13] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *KDD'04*, Seattle, Washington, USA, 2004.

[14] S. Kim and S. Narayanan. Dynamic chroma feature vectors with applications to cover song identification. In *MMSP'08*, Cairns, Australia, 2008.

[15] O. Lartillot and P. Toiviainen. A MATLAB toolbox for musical feature extraction from audio. In *DAFx-07*, Bordeaux, France, 2007.

[16] M. Li and R. Sleep. Melody classification using a similarity metric based on Kolmogorov complexity. In *SMC'04*, Paris, France, 2004.

[17] M. Li and R. Sleep. Genre classification via an LZ78-based string kernel. In *ISMIR'05*, London, UK, 2005.

[18] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD'03*, San Diego, California, USA, 2003.

[19] J. Serrà, E. Gómez, and P. Herrara. Transposing chroma representations to a common key. In *IEEE CS Conference on The Use of Symbols to Represent Music and Multimedia Objects*, 2008.

[20] J. Serrà, E. Gómez, and P. Herrara. *Audio Cover Song Identification and Similarity: Background, Approaches, Evaluation, and Beyond*. Springer-Verlab Berlin / Heidelberg, 2010.

[21] Y. Yu, M. Crucianu, V. Oria, and L. Chen. Local summarization and multi-level LSH for retrieving multivariant audio tracks. In *MM'09*, Beijing, China, 2009.

COMBINING FEATURES REDUCES HUBNESS IN AUDIO SIMILARITY

Arthur Flexer,[1] **Dominik Schnitzer,**[1,2] **Martin Gasser,**[1] **Tim Pohle**[2]
[1]Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria
[2]Department of Computational Perception
Johannes Kepler University Linz, Austria
arthur.flexer@ofai.at, dominik.schnitzer@ofai.at
martin.gasser@ofai.at, tim.pohle@jku.at

ABSTRACT

In audio based music similarity, a well known effect is the existence of hubs, i.e. songs which appear similar to many other songs without showing any meaningful perceptual similarity. We verify that this effect also exists in very large databases ($>$ 250000 songs) and that it even gets worse with growing size of databases. By combining different aspects of audio similarity we are able to reduce the hub problem while at the same time maintaining a high overall quality of audio similarity.

1. INTRODUCTION

One of the central goals in music information retrieval is the computation of audio similarity. Proper modeling of audio similarity enables a whole range of applications: genre classification, play list generation, music recommendation, etc. The de facto standard approach to computation of audio similarity is timbre similarity based on parameterization of audio using Mel Frequency Cepstrum Coefficients (MFCCs) plus Gaussian mixtures as statistical modeling (see Section 3.1). However, it is also an established fact that this approach suffers from the so-called hub problem [3]: songs which are, according to the audio similarity function, similar to very many other songs without showing any meaningful perceptual similarity to them. The hub problem of course interferes with all applications of audio similarity: hub songs keep appearing unwontedly often in recommendation lists and play lists, they degrade genre classification performance, etc.

Although the phenomenon of hubs is not yet fully understood, a number of results already exist. Aucouturier and Pachet [1] established that hubs are distributed along a scale-free distribution, i.e. non-hub songs are extremely common and large hubs are extremely rare. This is true for MFCCs modelled with different kinds of Gaussian mixtures as well as Hidden Markov Models, irrespective whether parametric Kullback-Leibler divergence or non-parametric histograms plus Euclidean distances are used for computation of similarity. But is also true that hubness is not the property of a song per se since non-parametric and parametric approaches produce very different hubs. It has also been noted that audio recorded from urban soundscapes, different from polyphonic music, does not produce hubs [2] since its spectral content seems to be more homogeneous and therefore probably easier to model. Direct interference with the Gaussian models during or after learning has also been tried (e.g. homogenization of model variances) although with mixed results. Whereas some authors report an increase in hubness [1], others observed the opposite [5]. Using a Hierarchical Dirichlet Process instead of Gaussians for modeling MFCCs seems to avoid the hub problem altogether [6].

Our contribution to the understanding of the hub problem is threefold: (i) since all results on the hub problem so far were achieved on rather small data sets (from $\sim$ 100 to $\sim$ 15000 songs), we first establish that the problem also exists in very large data sets ($>$ 250000 songs); (ii) we show that a non-timbre based parameterization is not prone to hubness; (iii) finally we show how combining timbre based audio similarity with other aspects of audio similarity is able to reduce the hub problem while maintaining a high overall quality of audio similarity.

2. DATA

2.1 Web shop data

For our experiments we used a data set $D(ALL)$ of $S_W = 254398$ song excerpts (30 seconds) from a popular web shop selling music. The freely available preview song excerpts were obtained with an automated web-crawl. All meta information (artist name, album title, song title, genres) is parsed automatically from the hmtl-code. The excerpts are from $U = 18386$ albums from $A = 1700$ artists. From the 280 existing different hierarchical genres, only the $G_W = 22$ general ones on top of the hierarchy are being kept for further analysis (e.g. "Pop/General" is kept but not "Pop/Vocal Pop"). The names of the genres plus percentages of songs belonging to each of the genres are given in Table 1. Please note that every song is allowed to belong to more than one genre, hence the percentages in Table 1 add up to more than 100%. The genre information is identical for all songs on an album. The numbers of

genre labels per albums range from 1 to 8. Our database was set up so that every artist contributes between 6 to 29 albums.

To study the influence of the size of the database on results, we created random non-overlapping splits of the entire data set: $D(1/2)$ - two data sets with mean number of song excerpts $= 127199$, $D(1/20)$ - twenty data sets with mean number of songs excerpts $= 12719.9$, $D(1/100)$ - one hundred data sets with mean number of songs excerpts $= 2543.98$. An artist with all their albums is always a member of a single data set.

Pop	Classical	Broadway
49.79	12.89	7.45
Soundtracks	Christian/Gospel	New Age
1.00	10.20	2.48
Miscellaneous	Opera/Vocal	Alternative Rock
6.11	3.24	27.13
Rock	Rap/Hip-Hop	R&B
51.78	0.98	4.26
Hard Rock/Metal	Classic Rock	Country
15.85	15.95	4.07
Jazz	Children's Music	International
6.98	7.78	9.69
Latin Music	Folk	Dance & DJ
0.54	11.18	5.24
Blues		
11.24		

Table 1. Percentages of songs belonging to the 22 genres with multiple membership allowed for the **web shop data**.

2.2 Music portal data

We also used a smaller data base comprised of the music of an Austrian music portal. The FM4 Soundpark is an internet platform [1] of the Austrian public radio station FM4. This internet platform allows artists to present their music free of any cost in the WWW. All interested parties can download this music free of any charge. This music collection contains about 10000 songs and is organized in a rather coarse genre taxonomy. The artists themselves choose which of the $G_M = 6$ genre labels "Hip Hop, Reggae, Funk, Electronic, Pop and Rock" best describe their music. The artists are allowed to choose one or two of the genre labels. We use a data base of $S_M = 7665$ songs for our experiments. Number of songs and percentages across genres are given in Table 2. Please note that every song is allowed to belong to more than one genre, hence the percentages in Table 2 add up to more than 100%.

[1] http://fm4.orf.at/soundpark

HiHo	Regg	Funk	Elec	Pop	Rock
15.34	4.64	21.87	46.25	34.39	44.03

Table 2. Percentages of songs belonging to genres with multiple membership allowed for the **music portal data**. Genres are Hip Hop, Reggae, Funk, Electronic, Pop and Rock.

3. METHODS

We compare two approaches based on different parameterizations of the data. Whereas Mel Frequency Cepstrum Coefficients (MFCCs) are a quite direct representation of the spectral information of a signal and therefore of the specific "sound" or "timbre" of a song, Fluctuation Patterns (FPs) are a more abstract kind of feature describing the amplitude modulation of the loudness per frequency band.

3.1 Mel Frequency Cepstrum Coefficients and Single Gaussians (G1)

We use the following approach to compute music similarity based on spectral similarity. For a given music collection of songs, it consists of the following steps:

1. for each song, compute MFCCs for short overlapping frames
2. train a single Gaussian (G1) to model each of the songs
3. compute a distance matrix M_{G1} between all songs using the symmetrized Kullback-Leibler divergence between respective G1 models

For the web shop data the 30 seconds song excerpts in mp3-format are recomputed to 22050Hz mono audio signals. For the music portal data, the two minutes from the center of each song are recomputed to 22050Hz mono audio signals. We divide the raw audio data into overlapping frames of short duration and use Mel Frequency Cepstrum Coefficients (MFCC) to represent the spectrum of each frame. MFCCs are a perceptually meaningful and spectrally smoothed representation of audio signals. MFCCs are now a standard technique for computation of spectral similarity in music analysis (see e.g. [7]). The frame size for computation of MFCCs for our experiments was $46.4ms$ (1024 samples), the hop size $23.2ms$ (512 samples). We used the first $d = 25$ MFCCs for all experiments with the web shop data and the first $d = 20$ MFCCs for all experiments with the music portal data.

A single Gaussian (G1) with full covariance represents the MFCCs of each song [8]. For two single Gaussians, $p(x) = \mathcal{N}(x; \mu_p, \Sigma_p)$ and $q(x) = \mathcal{N}(x; \mu_q, \Sigma_q)$, the closed form of the Kullback-Leibler divergence is defined as [14]:

$$KL_N(p\|q) = \frac{1}{2}\left(\log\left(\frac{\det(\Sigma_p)}{\det(\Sigma_q)}\right) + Tr\left(\Sigma_p^{-1}\Sigma_q\right) + (\mu_p - \mu_q)'\,\Sigma_p^{-1}(\mu_q - \mu_p) - d\right) \quad (1)$$

where $Tr(M)$ denotes the trace of the matrix M, $Tr(M) = \Sigma_{i=1..n} m_{i,i}$. The divergence is symmetrized by computing:

$$KL_{sym} = \frac{KL_N(p\|q) + KL_N(q\|p)}{2} \quad (2)$$

3.2 Fluctuation Patterns and Euclidean Distance (FP)

Fluctuation Patterns (FP) [9] [12] describe the amplitude modulation of the loudness per frequency band and are based on ideas developed in [4]. For a given music collection of songs, computation of music similarity based on FPs consists of the following steps:

1. for each song, compute a Fluctuation Pattern (FP)
2. compute a distance matrix M_{FP} between all songs using the Euclidean distance of the FP patterns

Closely following the implementation outlined in [10], an FP is computed by: (i) cutting an MFCC spectrogram into three second segments, (ii) using an FFT to compute amplitude modulation frequencies of loudness (range $0 - 10Hz$) for each segment and frequency band, (iii) weighting the modulation frequencies based on a model of perceived fluctuation strength, (iv) applying filters to emphasize certain patterns and smooth the result. The resulting FP is a 12 (frequency bands according to 12 critical bands of the Bark scale [15]) times 30 (modulation frequencies, ranging from 0 to $10Hz$) matrix for each song. The distance between two FPs i and j is computed as the squared Euclidean distance:

$$D(FP^i, FP^j) = \sum_{k=1}^{12}\sum_{l=1}^{30}(FP^i_{k,l} - FP^j_{k,l})^2 \quad (3)$$

For the web shop data an FP pattern is computed from the full 30 second song excerpts. For the music portal data an FP pattern is computed from the central minute of each song.

4. RESULTS

4.1 Hubs in very large data bases

As a measure of the hubness of a given song we use the so-called n-occurrence [1], i.e. the number of times the songs occurs in the first n nearest neighbors of all the other songs in the data base. Please note that the mean n-occurrence across all songs in a data base is equal to n. Any n-occurrence significantly bigger than n therefore indicates existence of a hub. For every song in the data bases $D(ALL)$, $D(1/2)$, $D(1/20)$ and $D(1/100)$ (see Section 2.1) we computed the first n nearest neighbors for both methods G1 and FP. For method G1, the first n nearest neighbors are the n songs with minimum Kullback Leibler divergence (Equation 2) to the query song. For method FP, the first n nearest neighbors are the songs with minimum Euclidean distance of the FP pattern (Equation 3) to the query song. To compare results for data bases of different sizes S_W, we keep the relation n/S_W constant at 0.001965: e.g. for $D(ALL)$ $S_W = 254398$ and $n = 500$, for $D(1/100)$ $S_W = 2543.98$ and therefore $n = 5$.

data set	n	maxhub	maxhub%	$hub3\%$
D(ALL)	500	29588	11.63	7.75
D(1/2)	250	12094	9.52	7.56
D(1/20)	25	590	4.68	6.13
D(1/100)	5	62	2.49	4.62

Table 3. Hub analysis results for **web shop data** using method **G1**. See Section 4.1 for details.

data set	n	maxhub	maxhub%	$hub3\%$
D(ALL)	500	3386	1.33	1.18
D(1/2)	250	1639	1.29	1.18
D(1/20)	25	137	1.08	1.12
D(1/100)	5	25	1.02	1.22

Table 4. Hub analysis results for **web shop data** using method **FP**. See Section 4.1 for details.

The results given in Tables 3 and 4 show mean values over 100 ($D(1/100)$), 20 ($D(1/20)$), 2 ($D(1/2)$) data sets or the respective single result for the full data set $D(ALL)$. We give the number of nearest neighbors n, the absolute number of the maximum n-occurrence $maxhub$ (i.e. the biggest hub), the percentage of songs in whose nearest neighbor lists this biggest hub can be found $maxhub\% = maxhub/S_W$ and the percentage of hubs $hub3\%$ (i.e. the percentage of songs of which the n-occurrence is more than three times n).

When looking at the results for method G1 (Table 3) it is clear that hubs do exist even for very large data bases. As a matter of fact, the hub problem increases significantly with the size of the data base. Whereas for the small data sets $D(1/100)$ on average the biggest hub is in the neighbor lists of 2.49% of all songs, the biggest hub for $D(ALL)$ is a neighbor to 11.63% of all songs. The number of hubs increases from an average 4.62% of all songs in $D(1/100)$ to 7.75% in $D(ALL)$. To sum up, there are more and bigger hubs in larger data bases when using method G1 for computation of audio similarity.

The results for method FP in Table 4 show a quite different picture. The size of the biggest hub is much smaller and the number of hubs is also much reduced. There is also very little influence of the size of the data bases on the results. We like to conclude that method FP is not as prone to hubness as method G1.

w_{G1}	w_{FP}	maxhub	maxhub%	$hub3\%$	$hub10\%$	$hub15\%$	$hub20\%$	acc
1.0	**0.0**	**879**	**11.47**	**8.05**	**0.94**	**0.40**	**0.22**	**48.47**
0.9	0.1	598	7.80	8.15	0.86	0.35	0.09	49.84
0.8	0.2	445	5.81	8.23	0.80	0.23	0.08	49.47
0.7	0.3	342	4.46	8.11	0.72	0.16	0.05	48.44
0.6	**0.4**	**352**	**4.59**	**8.06**	**0.57**	**0.09**	**0.01**	**47.80**
0.5	0.5	344	4.49	8.04	0.51	0.07	0.01	46.58
0.4	0.6	334	4.36	7.91	0.31	0.04	0.01	45.73
0.3	0.7	315	4.11	7.80	0.21	0.01	0.01	44.93
0.2	0.8	247	3.22	7.21	0.17	0.01	0.0	43.94
0.1	0.9	215	2.81	6.72	0.04	0.0	0.0	42.82
0.0	**1.0**	**145**	**1.89**	**5.38**	**0.0**	**0.0**	**0.0**	**38.45**

Table 5. Hub analysis result for **music portal data** using combinations of **G1** and **FP**. Results for using G1 or FP alone as well as for a moderate combination are in bold face. See Section 4.2 for details.

4.2 Reducing hubs by combining G1 and FP

Recent advances in computing audio similarity rely on combining timbre-based approaches (MFCCs plus Gaussian models) with a range of other features derived from audio. In particular, combinations of timbre and, among other features, fluctuation patterns or variants thereof have proven sucessfull [11, 13]. Such a combination approach was able to rank first at the 2009 MIREX "Audio Music Similarity and Retrieval"-contest [2]. Since our method based on fluctuation patterns is less prone to hubness than the timbre based approach, we tried to combine distances obtained with methods G1 and FP. It is our hypothesis that such a combination could reduce hubness and at the same time preserve the good quality of timbre based methods in terms of audio similarity.

Following previous approaches towards combination of features [10, 11] we first normalize the distance matrices M_{G1} and M_{FP} by subtracting the respective overall means and dividing by the standard deviations:

$$\bar{M}_{G1} = \frac{M_{G1} - \mu_{G1}}{s_{G1}} \quad \bar{M}_{FP} = \frac{M_{FP} - \mu_{FP}}{s_{FP}} \qquad (4)$$

We combine the normalized distance matrices linearly using weights w_{G1} and w_{FP}:

$$\bar{M}_C = w_{G1}\bar{M}_{G1} + w_{FP}\bar{M}_{FP} \qquad (5)$$

To evaluate the quality of audio similarity achieved by combining methods G1 and FP we computed the genre classification performance. We used nearest neighbor classification as a classifier. For every song in the data base we computed the first nearest neighbor using the distance matrix $\bar{M}_C$. The first nearest neighbor to a query song is the song with minimum distance according to $\bar{M}_C$. To estimate genre classification accuracy, the genre label of a query song s_{query} and its first nearest neighbor s_{nn} were compared. The accuracy is defined as:

$$acc(s_{query}, s_{nn}) = \frac{|g_{query} \cap g_{nn}|}{|g_{query} \cup g_{nn}|} \times 100 \qquad (6)$$

[2] http://www.music-ir.org/mirex/2009/

with g_{query} (g_{nn}) being a set of all genre labels for the query song (nearest neighbor song) and $|.|$ counting the number of members in a set. Therefore accuracy is defined as the number of shared genre labels divided by the set size of the union of sets g_{query} and g_{nn} times 100. The latter is done to acount for nearest neighbor songs with two genre labels as compared to only one genre label. The range of values for accuracy is between 0 and 100. All genre classification results are averaged over ten fold cross validations.

We ran a series of experiments using the music portal data base (see Section 2.2) and a number of different weight combinations w_{G1} and w_{FP}. To measure the hubness of a given song we use n-occurrence with n equal 15. The results given in Table 5 show: the weights w_{G1} and w_{FP}, the absolute number of the maximum n-occurrence $maxhub$ (i.e. the biggest hub), the percentage of songs in whose nearest neighbor lists this biggest hub can be found $maxhub\%$, the percentage of hubs $hub3|10|15|20\%$ (i.e. the percentage of songs of which the n-occurrence is more than $3|10|15|20$ times n) and the genre classification accuracy acc.

It is evident that with the weight w_{FP} for method FP growing, the hubs become smaller and less in number but the genre classification accuracy also degrades. Whereas using method G1 alone (i.e. $w_{G1} = 1.0$ and $w_{FP} = 0.0$) yields a maximum hub of size 879 that is in the nearest neighbor lists of 11.47% of all songs, a moderate combination using weights $w_{G1} = 0.6$ and $w_{FP} = 0.4$ diminishes the biggest hub to a size of 352. This reduced hub is now a member of only 4.59% of the nearest neighbor lists. Also the number of especially large hubs decreases: e.g. the percentage of songs of which the n-occurrence is more than 20 times n ($hub20\%$) drops from 0.22% to 0.01% (in absolute numbers from 17 to 1); the number of more moderate sized hubs ($hub10\%$) is still about halfed (from 0.94% to 0.57%, or from 72 to 44 in absolute numbers). Such a moderate combination does not impair the overall quality of audio similarity as measured with genre classification accuracy: it is at 47.80% which is at the level of using method G1 alone yielding 48.47%. The baseline accuracy achieved by always guessing the most probable

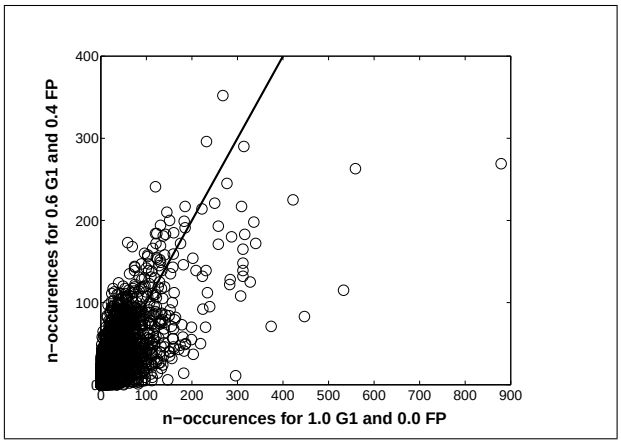


Figure 1. n-occurrences of using method G1 alone (x-axis) vs. n-occurrences using a moderate combination of G1 and FP (y-axis, $w_{G1} = 0.6$ and $w_{FP} = 0.4$) for **music portal data**. The diagonal line indicates songs for which the n-occurence does not change.

genre "Electronic" (see Table 2) is 29.11%. Always guessing the two most probable genres "Electronic" and "Rock" yields 36.46%.

In Figure 1 we have plotted the n-occurrences of using method G1 alone (i.e. $w_{G1} = 1.0$ and $w_{FP} = 0.0$) versus the n-occurrences of the moderate combination using weights $w_{G1} = 0.6$ and $w_{FP} = 0.4$. This is done for all songs in the music portal data base. The n-occurrence of every song beneath the diagonal line is reduced by using the combination. All large hubs with an n-occurrence bigger than 300 are clearly reduced. The same is true for the majority of hubs with n-occurrences between 200 and 300.

5. CONCLUSION

We were able to show that the so-called hub problem in audio based music similarity indeed does exist in very large data bases and therefore is not an artefact of using limited amounts of data. As a matter of fact, the relative amount and size of hubs is even growing with the size of the data base. On the same very large web shop data base we were able to show that a non-timbre based parameterization of audio similarity (fluctuation patterns) is by far not as prone to hubness as the standard approach of using Mel Frequency Cepstrum Coefficients (MFCCs) plus Gaussian modeling. Extending recent successful work on combining different features to compute overall audio similarity, we were able to show that this not only maintains a high quality of audio similarity but also decisively reduces the hub problem.

The combination result has so far only been shown on the smaller music portal data base, but there is no reason why this should not hold for the larger web shop data. Only limitations in computer run time led us to first evaluate the combination approach on the smaller data set. We are not claiming that our specific combination of features is the best general route towards audio similarity. But we are convinced that going beyond pure timbre-based similarity is able to achieve two goals simultaneously: high quality audio similarity and avoiding the hub problem.

6. ACKNOWLEDGEMENTS

This research is supported by the Austrian Science Fund (FWF, grants L511-N15 and P21247) and the Vienna Science and Technology Fund (WWTF, project "Audiominer").

7. REFERENCES

[1] Aucouturier J.-J., Pachet F.: A scale-free distribution of false positives for a large class of audio similarity measures, *Pattern Recognition*, Vol. 41(1), pp. 272-284, 2007.

[2] Aucouturier J.-J., Defreville B., Pachet F.: The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music, *Journal of the Acoustical Society of America*, 122 (2), 881-891, 2007.

[3] Aucouturier, J.-J., Pachet F.: Improving Timbre Similarity: How high is the sky?, *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[4] Fruehwirt M., Rauber A.: Self-Organizing Maps for Content-Based Music Clustering, *Proceedings of the Twelth Italian Workshop on Neural Nets*, IIAS, 2001.

[5] Godfrey M.T., Chordia P.: Hubs and Homogeneity: Improving Content-Based Music Modeling, *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, Philiadelphia, USA, 2008.

[6] Hoffman M., Blei D., Cook P.: Content-Based Musical Similarity Computation Using the Hierarchical Dirichlet Process, *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, Philiadelphia, USA, 2008.

[7] Logan B.: Mel Frequency Cepstral Coefficients for Music Modeling, *Proceedings of the International Symposium on Music Information Retrieval (ISMIR'00)*, Plymouth, Massachusetts, USA,2000.

[8] Mandel M.I., Ellis D.P.W.: Song-Level Features and Support Vector Machines for Music Classification, *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, 2005.

[9] Pampalk E.: Islands of Music: Analysis, Organization, and Visualization of Music Archives, MSc Thesis, Technical University of Vienna, 2001.

[10] Pampalk E.: Computational Models of Music Similarity and their Application to Music Information Retrieval, Vienna University of Technology, Austria, Doctoral Thesis, 2006.

[11] Pampalk E., Flexer A., Widmer G.: Improvements of Audio-Based Music Similarity and Genre Classification, *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, September 11-15., 2005.

[12] Pampalk E., Rauber A., Merkl D.: Content-based organization and visualization of music archives, *Proceedings of the 10th ACM International Conference on Multimedia*, Juan les Pins, France, pp. 570-579, 2002.

[13] Pohle T., Schnitzer D., Schedl M., Knees P., Widmer G.: On rhythm and general music similarity, *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR'09)*, Kobe, Japan, 2009.

[14] Penny W.D.: Kullback-Leibler Divergences of Normal, Gamma, Dirichlet and Wishart Densities, Wellcome Department of Cognitive Neurology, 2001.

[15] Zwicker E., Fastl H.: *Psychoaccoustics, Facts and Models, Springer Series of Information Sciences*, Volume 22, 2nd edition, 1999.

COMPUTATIONAL ANALYSIS OF MUSICAL INFLUENCE: A MUSICOLOGICAL CASE STUDY USING MIR TOOLS

Nick Collins
Department of Informatics, University of Sussex, Falmer, Brighton, BN1 9QJ, UK
N.Collins@sussex.ac.uk

ABSTRACT

Are there new insights through computational methods to the thorny problem of plotting the flow of musical influence? This project, motivated by a musicological study of early synth pop, applies MIR tools as an aid to the investigator. Web scraping and web services provide one angle, sourcing data from allmusic.com, and utilising python APIs for last.fm, EchoNest, and MusicBrainz. Charts of influence are constructed in GraphViz combining artist similarity and dates. Content based music similarity is the second approach, based around a core collection of synth pop albums. The prospect for new musical analyses are discussed with respect to these techniques.

1. INTRODUCTION

Musicians have always been aware of issues of musical influence, from the lists of influences set out in adverts for new band members, the intensive relationship of teachers and pupils in many traditions, to composers consciously admitting their predecessors through interviews, personal journals, and in some cases unconscious or deliberate quotations. Whilst it is convenient to focus on grand examples in a 'genius' model of musical history, all eras of music have had a host of active musicians, though no era more than today's hyper-warren of content creators. Chopin's letters, for example, are littered with references to other active pianist-composers of the day, most of whom are no longer household names, yet Chopin writes 'I shall not be an imitation of Kalkbrenner: *he* has not the power to extinguish my perhaps too audacious but noble wish and intention to create for myself a new world' [9, p. 103]. The literature on human creativity is of note here in exploring the processes of human invention within the engine of culture [5, 15]. Musicologists have re-cast traditional concerns over influence to questions of 'inter-textuality', and the degree to which any musical work can be seen as distinct from social and musical currents [16]. Influence is intimately connected to the continuous negotiation of musical style as it transforms over time; the gradual formation of genres is implicit in much discussion of the philosophy of stylistic categories in music [1, 10], and related to similar questions in biology concerning speciation events and memetics [4, 6].

Automated methods for the analysis of musical similarity provide a new angle on relationships between works, whether comparing individual pieces or within larger corpora. For example, the data-driven analyses explored by David Cope across MIDI files [3] are primarily used for synthesis, but can also help to explore the links between composers. Symbolic analysis tools in MIR parallel such movements in algorithmic composition: McKay and Fujinaga [11] discuss the application of their jSymbolic feature extractor and Autonomous Classification Engine machine learning tool to such projects as comparison between a Chopin nocturne and Mendelssohn piano trio, or distinguishing de Machaut and Palestrina. Charles Smith has carried out perhaps the largest musicological study of influence amongst classical composers by a series of measures applied over library resources, and presents it in a website describing the 'Classical Music Universe'. [1]

There are many MIR studies which have analyzed the current state of public opinion on artist similarity, for purposes of tracking popularity and making recommendations. Zadel and Fujinaga [19] combine cultural meta-data from Amazon with a metric of similarity based on Google search counts to generate a network of related artists through web services. Fields et al. [8] scraped MySpace pages, tracing the recursive (to sixth degree of separation) network of friends and evaluating musical similarity through audio content analysis of their sound examples. They mention influence as one potential link between artists, but do not unpack it from collaboration or general similarity. Park et al. [13] also study the network structure of artists, by scraping online music databases such as allmusic.com, but concentrate on collaboration or 'expert' annotated similarity rather than any explicit tie to dates. Again tackling MySpace, Beuscart and Couronné [2] namecheck influence in their title, but mean it as a general measure of recommendation amongst cliques of artists rather than as a formative influence on creative output.

Thus, although the topics of similarity and genre remain central tenets of much music information retrieval work, the role of dates as markers of the flow of influence is not so widely discussed. This paper makes dates a central part of a musicological investigation. The applicability of MIR

[1] http://people.wku.edu/charles.smith/music/index2.htm

tools to studies of influence both via online meta-data parsing and content based analysis applications is explored. In the latter content analysis, a tight knit set of synth pop albums from the years 1977-1981 are put under the microscope, providing a real challenge for discrimination and a microcosm of gradual stylistic change.

Section 2 introduces the context of synth pop as well as the central node, Depeche Mode (DM). Section 3 presents web scraping and web service exploration of the network of artists around DM, with a technique to automatically extract dates for artists using the MusicBrainz web service. Network diagrams are constructed through python programs and GraphViz. Section 4 tackles the influence question using a set of synth pop albums from the era up to four years before the first DM release, looking for automatic recognition of possible leads on influence through musical similarity (marsyas is the tool of choice here). Results and future extensions are discussed.

2. SYNTH POP AND DEPECHE MODE

The cost of analog synthesizers decreased in the 1970s, until an all synthesizer band was a viable proposition for musicians starting out in the post punk era [14]. Although there are always earlier precedents, and synths had been long known in popular music through such phenomena as mass selling Moog albums, prog rock keyboardists, and krautrock, a real concentration of synth led bands emerged in the later 1970s into a position of mainstream chart success. The Second Invasion of the US by British bands on the back of MTV featured a plethora of synthesized sounds, and the 1980s saw even greater availability of electronic equipment as digital technology stole the show. Although some 'New Romatic' bands such as Duran Duran had only a single keyboardist, the more central examples of synth pop tend to feature all synthesizer backing, including sequencers and drum machine in place of acoustic drummers, after the Kraftwerk model; but like all supposed categories, inbetween cases exist.

Depeche Mode were by no means the first synth pop band, nor the first with popular market appeal; both Kraftwerk as an all electronic band, and Gary Numan as an individual who featured synthesizers, had had greater commercial success than their first album was to achieve. Yet in longer term commercial and artistic success, impact and influence, DM are still of great importance, and a fascinating subject of study in terms of tracking influence. They have touched multiple putative genres, from early teen synth pop, through darker industrial sampling, to electronic tinged stadium rock, [2] and inspired divergent artists (as one example, see covers compilations such as *For the Masses* (1998) or the Swedish synth pop tribute *I Sometimes Wish I Was Famous* (1991)). Early DM is also of note in that Vince Clarke was the chief songwriter, rather than Martin L. Gore, and such complications bring home the challenges of tracking a band's inspirations and influence through extended careers. [3] In another example of the complications, as bands progress through multiple albums, they work with many people, often bringing in younger producers who emerged in historically later scenes (in DM's case, such as Flood, or Mark Bell). The network of musicians who influenced, and who were influenced by, Depeche Mode are examined in a web data analysis, and work historically closely prior to the album Speak & Spell is the focus for the audio content analysis.

Statements by band members past and present provide insight into the formative influences of the band. For example, in Miller's biography [12], the band admit early influences including Gary Numan (p.21), OMD and the track 'Almost' (p.23), The Human League and particularly 'Being Boiled' (p.483), Kraftwerk (p.25) and John Foxx (p.26). DM gigged early on with the post-Foxx incarnation of Ultravox, and their label owner and first producer was the British DIY synth pioneer Daniel Miller; Mute Records artists would remain a central touchpoint as the band developed. Such references are further discussed below.

3. WEB SCRAPING AND WEB SERVICES FOR THE ANALYSIS OF INFLUENCE

Although a musicologist might construct their own model of musical influence from analyzing primary and secondary sources such as original releases, reviews and interviews, the wealth of online commentary and databases provides a further strand of evidence for systematic musicology to exploit. Although there can be issues with the verifiability of information, the much remarked problems of reliability of meta-data in MIR [7], it can still be healthy to admit web content as part of the arsenal of the musicologist. This paper examines the use of web scraping and web services to collect alternative viewpoints on the influences upon and influence of a particular central band. Although the techniques may be applied to any starting point, Depeche Mode are chosen in particular for this study.

The following APIs and websites were investigated:

- allmusic.com artist information explicitly contains entries for 'Influenced By' and 'Followers'
- The EchoNest API has convenience methods to obtain biographic data, lists of similar artists, and a measure of 'familiarity' for a given artist.
- The MusicBrainz metadatabase has an API which allows interrogation of artist releases and dates.
- The last.fm API can return a list of similar artists amongst further functionality

Programs were written in python to utilise the APIs, and for web scraping.

Three tactics could generate graphs of related artists with direction of edges determined by date, using recursive construction. In the first case a similarity measure from a

[2] Arguably, after DM's best selling *Violator* (1990), even converging with U2 for 1991's *Achtung Baby*, as U2 chased the contemporary sound of electronic dance music's commercial break through

[3] A similar radical change of personnel is seen for example in The Human League's development in 1980.

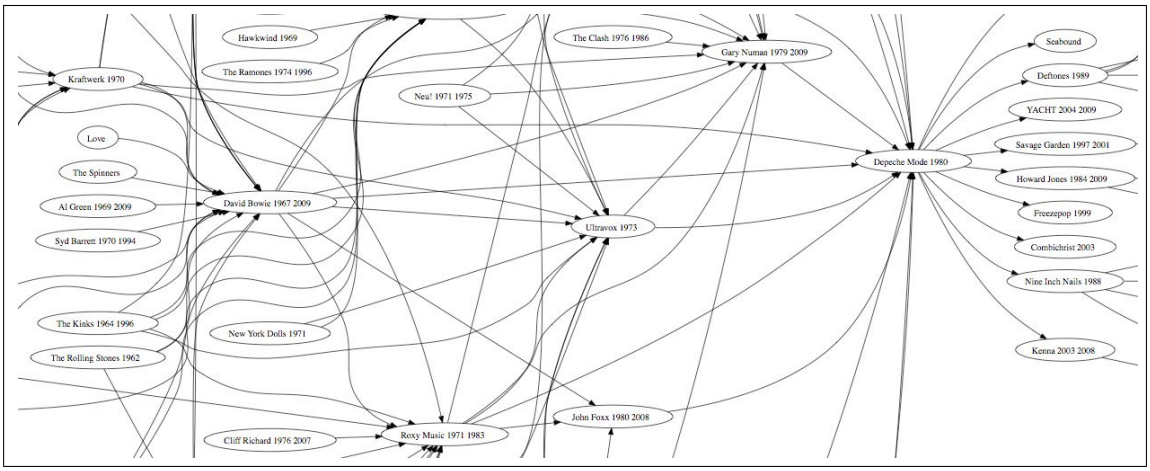


Figure 1. Excerpt of a graph of influences based around Depeche Mode, filtered by familiarity ratings of at least 0.6 per artist according to EchoNest. Note the errors and omissions in dating information, for example the start date for Cliff Richard, who appears via a supposed second order influence from Roxy Music. The overall graph, even relaxing the familiarity filtering, is much richer for precedents than for successors, perhaps reflecting the balance of music history and journalism with regard to the present day.

starting artist was used, and dates imposed as a way of determining directivity (most simply, earliest date of activity of a given artist, ignoring career overlaps). In the second, the allmusic.com site was scraped for the pre-marked Influenced By and Followers lists, which gave the direction of arrows without needing dates (however, dates were still annotated with artist names in this case as a helpful guide; similarity ratings between artists could also rate strength of connection). The third method is for a musicologist to provide a list of artists for which they are interested in interconnections; they can then try various similarity measures to weight connections, and dates can be automatically determined where they are not already known. [4]

Because the computational search should be as automated as possible if larger networks are to be generated, musicbrainz.org provided the ability to hunt for start and end dates of artists (the musicologist can always further corroborate dates later if any promising links are revealed). This was actually one of the hardest coding problems to solve, because for the start date MusicBrainz returns the date of birth of an individual artist, but the date of formation for a group. Code was written for individuals and for cases where there was no returned valid start or end date, to hunt through associated album and single release dates, taking minima and maxima. The API often failed to respond when called too often, but the program kept trying with increasing gaps between calls until connection was re-established or it failed ten times in a row. All dates were stored in a local database to avoid the slow dependency on MusicBrainz, only checking artists if they were new to the database or no date had yet been established in previous attempts (the musicologist can also in principle overwrite dates if they discover more reliable data). [5]

The final large networks of artists could be directly plotted, but facilities were also added to cluster by the year an artist began their recording career, [6] and to exclude artists falling below a certain threshold of familiarity with respect to the EchoNest measure. Figure 1 shows an excerpt of a graph generated via the allmusic database method, recursing only up to second order connections, and annotating dates of artists via MusicBrainz.

It was definitely of benefit to spend time with the technology and with online opinion as a method of immersion into the subject. Results however must be interpreted with caution; in particular, the allmusic.com annotations for synth pop artists did not expose some expected links (for example, Depeche Mode as an influence on Alphaville, the Pet Shop Boys, or Goldfrapp, to name but three, though Camouflage and Nine Inch Nails did appear as successors; The Human League were not listed as an influence despite DM's own documented confession). They did however point to a few other possible leads worth pursuing. Some second order links, such as Elvis Presley and Chuck Berry were admitted by Martin L.Gore as his earliest listening in a recent interview [7], though the centrality of such figures, particularly with respect to the Beatles hub, is a little too obvious and a likely side effect of dominant nodes in artist networks [13].

It was found in practice that similarity of artists as a singular term often proved insufficient, in that it did not adequately respect musical characteristics over social. For example, Pandora, which in any case admits no API, lists similar artists to DM as The Cure, New Order, Duran Duran, The Smiths and Tears for Fears. There is one justifi-

[4] As a proviso to this process, different data sources and similarity measures reflect different construction principles from 'expert' annotation (allmusic) to community consensus (MusicBrainz), and the analyst should keep this in mind.

[5] As pointed out by a reviewer, an alternative model here may be to exploit DBpedia and LinkedData for the Semantic Web.

[6] All artists are in development off the commercial radar for a long time, and formative influences not necessarily via mass released recordings; but the underlying assumption to keep this project manageable is that a commercial release reveals the potential to influence a large number of followers.

[7] http://www.bbc.co.uk/programmes/b00jn4fl

cation as UK bands all active circa 1983 or so, but in terms of tracing the history of synth pop, there are a few overlaps and some mishits (The Smiths, for example). Data was also not always reliable; EchoNest listed Duran Duran Duran, the breakcore artist, instead of Duran Duran; arguably there is more electronic sound in the former, though it is probably an erroneous appearance in this context.

4. CONTENT BASED ANALYSIS

The other angle of approach in this project was to examine the actual audio recordings for similarity between artists. Armed with associated date information for tracks, networks of prior art can be constructed. Though there is not always causal proof that the authors of Speak and Spell would have heard and actively internalised particular tracks by prior artists (though see comments above in section 2 concerning admitted influences such as The Human League), it is possible to speculate, guided by such an investigation, and hope in general that the search provides a pillar in accumulating evidence for a particular linkage.

Table 2 contains a list of 37 albums or compilations, corresponding to 364 tracks, selected as the target data and space for musicological ground truth. The choice of albums reflects our own analysis of possible formative influences, with a bias to British acts, and covers the years 1977-1981, during which electronic instrumentation was breaking through to mass use in popular music (there are many earlier precedents, but the scope of inquiry was arranged around the post punk years transitioning to the early 80s). Depeche Mode were gigging in 1980, and released their first singles in 1981 leading up to the Speak & Spell album that October. There are many other artists and releases of potential relevance, both outside and within the restricted dates, but the core set is large enough to provide ample scope for musicological investigation and pose a significant challenge for MIR technology. [8] For digital convenience, despite originals being chiefly released on LP (CDs arrived in 1982), all data was sourced from purchased CD recordings, since these provide a guaranteed professional transition from master tapes. A few were re-masters (as noted in the table), with possible changes in overall compression and loudness, but since this did not substantially impact on human listening, in the ideal computer analysis should be able to cope (the timbral features used here did not include amplitude measurements as comparators). Any bonus tracks not readily available in the original era of release were excluded, which typically meant removing any tracks not on an original LP. Release dates were cross-referenced from online sources such as allmusic and discogs.com as well as liner notes and textbooks.

Our primary interest was to analyze relevant recordings that might show a strong similarity to tracks on Speak & Spell, and thus see if computer analysis could spot any links of influence. A secondary interest was the analysis of early synth pop's properties in general. There were various opinions and discoveries here from conventional listening, but the computer offered an alternative perspective.

Marsyas [17] and weka [18] provided the tools of choice for audio feature extraction, similarity measurement, and machine learning. The 44.1kHz 16 bit audio recordings were each passed through marsyas' bextract algorithm to obtain single vector averaged MFCC and spectral features over one minute 30 second sections taken from the middle of each track (window size and hop size 1024 samples). These obtain a long exposure timbral summary vector (64 dimensions) for each track. Similarity values between all individual tracks could then be created. A python script was written to order similar songs from a given starting song across the database. The nearest and furthest neighbours from each track on Speak &Spell were listed; Table 1 gives example results for the nearest and furthest ten tracks to the second DM single 'New Life'.

Score	Artist	Album	Track
1.047	DM	Dreaming Of Me	Speak & Spell
1.116	Gary Numan	Replicas	We Have A Technical
1.133	Ultravox	Systems Of Romance	Just For A Moment
1.150	Gary Numan	The Pleasure Principle	Random
1.152	Gary Numan	Telekon	I'm An Agent
1.153	OMD	Orchestral Manoeuvres In The Dark	Red Frame White Light
1.155	Ultravox	Vienna	Vienna
1.211	Ultravox	Vienna	Mr. X
1.221	Gary Numan	Replicas	Replicas
1.234	YMO	Solid State Surviver	Day Tripper
...	...	...	...
2.275	YMO	Yellow Magic Orchestra	Computer Game Theme From The Invader
2.320	Devo	The Essentials	Girl U Want
2.324	Cabaret Voltaire	The Original Sound Of Sheffield	Do The Mussolini (Headkick)
2.339	OMD	Architecture & Morality	Architecture And Morality
2.373	Human League	Reproduction	Blind Youth
2.490	Cabaret Voltaire	The Original Sound Of Sheffield	Baader Meinhof
2.593	John Foxx	Metamatic	Plaza
2.620	Human Leagure	Reproduction	Medley Austerity Girl One
2.623	YMO	Yellow Magic Orchestra	Computer Game Theme From The Circus
3.151	Human League	Dare	The Sound Of The Crowd

Table 1. Maximally similar and dissimilar tracks to 'New Life' by Depeche Mode within the database

Some results were not so surprising; both other DM singles from the first album are close by (Just Can't Get Enough comes in at 18th closest). Further away, the Baader Meinhof track is dark and unsettling and not rhythmic. The low bit arcade timbre of the YMO computer game themes are unique amongst materials here. John Foxx's Plaza features a prominent flanging effect. On the other hand, in musical terms the many distant up tempo Human League tracks, or the close appearance of Vienna are somewhat suspicious. The Sound of the Crowd persistently came far from all tracks on Speak and Spell, perhaps due to the loudly mixed vocal and particular synth percussion sounds.

[8] Possibilities for extensions just with artists active in this period include Telex, Joy Division/New Order, Throbbing Gristle, Jean-Michel Jarre and Wendy Carlos to name a fraction.

The album annotated feature data also underwent machine learning algorithm investigation, by training classifiers to differentiate artists' releases. The musicological interest is to find points of failure of discrimination as insight into potential timbral/musical overlaps and thus through information on dates, promising leads on the flow of influence, Confusion matrices help to indicate this. Under 10-fold validation, the best results were 31.8% correctly classified instances, for a Support Vector Machine (SVM) classifier; related to some other classes, Speak & Spell fared badly, with 2/12 tracks accurately labelled (precision 0.133, recall 0.167) and confusions for example with Reproduction by the Human League and Penthouse and Pavements by Heaven 17. Setting aside concerns over the perceptual relevance of the timbral features, it is challenging to ask for all 37 albums to be well differentiated on the basis of this data set (averaging 10 songs per label). As a more reasonable test, the data was labelled by the ten groupings shown by the horizontal lines in Table 2 (keeping Speak & Spell as a class of its own), obtaining 77% accurate classifications with an SVM. The confusion matrix for the DM album then showed 11 out of 12 songs accurately classified, and one mislabelled as by Gary Numan.

Classification by year was also explored, despite concerns over the hard histogram boundaries; classification accuracy of 55% was obtained, confirming somewhat the closely linked artists in this set (classification by half year periods dropped to 26%). Out of interest, I also tested how well recent artist La Roux's eponymous 2009 album was differentiated from the original synth pop sources to which it might be argued to pay substantial homage; in actual fact, when offered as a sixth category in the year based analysis 9/12 tracks were correctly identified; closely similar tracks were mainly drawn from the same album. Whether this is best traced to female vocals, to recent production and mastering trends, is a subject for further investigation.

Any machine identified link must be followed up by human analysis before imbuing significance. It is clear in the audio content analysis that gross timbral features are the basis of comparison, not details of materials at the human perceptual timeframe. These timbral links might indicate similar equipment or studio facilities more than links of inspiration. Nonetheless, it is valuable to make a start here in applying such MIR tools, on the understanding that through intensive research efforts in computational auditory models, systems can only improve. It was clear in this project that the machine tools were utilising different criteria. For example, a similarity was observed in motifs between a section two minutes into the Fad Gadget track 'Ricky's Hand' and DM's 'Photographic' but this was not closely borne out in machine results (similarity was right in the middle of all tracks, at 165th most similar); this is probably due to the smearing effects of the average concealing that particular section.

None of the leads presented by content analysis are themselves a smoking gun of influence when date is taken into account. It is preferable to seek further corroboration of any potential influence, and the status of conscious tribute or unconscious plagiarism will never be amenable to audio analysis methods alone. For the artists concerned, a single listen to a live gig or work in progress in the studio, obtaining a promotional copy ahead of public release, might all have provided undocumented links; this study was restricted to release dates, not recording dates. Certain off album tracks were excluded, for example songs only played at gigs ('Television' in the case of DM), or particular B sides ('Ice Machine', Shout'),[9] all of which might turn out to be potential connections.

5. CONCLUSIONS

This paper investigated the question of influences from two technological approaches, the first online information seeking, and the second content analysis over a database of relevant audio. Such a study can point to new leads that musicologists may not have immediately heard or imagined. Although there is some danger of getting back what is already known, in the main the great advantage has been the stimulus of exploring the materials from a new perspective. There may be more links between tracks in a close knit era of releases than the musicologist can comfortably track, and computer assistance certainly helps direct inquiry, prompting possibilities of connection, even as the human ear currently remains the best final judge.

Much future work remains, from further web data sourcing tools, to more developed schemes for content analysis. For the former, similarity through co-occurence is a profitable tool, and any similarity network can be mediated through the automatic artist date database. For the latter, more developed similarity methods may compare subsections and simultaneous voices within songs, perhaps with beat or chord synchronous features. A musicologist may wish to focus on particular attributes, choosing weights for rhythmic, timbral, harmonic, melodic features and more. For different pairs of songs, links might be posited based on different parameters, and a more developed analysis system would flag up significant similarities with respect to a number of different weighting schemes. The methods investigated herein do not track the career of artists stage by stage, nor cope with any complex inter-linked developments. A solution may combine content based methods and accurate dating of releases.

6. REFERENCES

[1] J. J. Aucouturier and F. Pachet. Representing musical genre: A state of the art. *Journal of New Music Research*, 32(1):83–93, 2003.

[2] J.-S. Beuscart and T. Couronné. The distribution of online reputation: Audience and influence of musicians on myspace. In *Third International AAAI Conference on Weblogs and Social Media*, pages 187–190. Association for the Advancement of Artificial Intelligence, 2009.

[3] David Cope. *Virtual Music : Computer Synthesis of Musical Style*. MIT Press, Cambridge, MA, 2001.

[4] R. Dawkins. *The Selfish Gene*. Oxford University Press, New York, NY, 1975.

[9] The opening of Ice Machine bears a resemblance to elements of The Word Before Last on the Human League's Reproduction album.

Artist	Album	Original Release Date	Notes
Kraftwerk	Trans-Europe Express	May 1977	
Kraftwerk	The Man Machine	May 1978	
Kraftwerk	Computer World	May 1981	2009 remaster
David Bowie	Low	January 1977	Remaster 1999, Brian Eno production of synth instrumentals in particular of note
David Bowie	Heroes	October 1977	Remaster 1999, Second of Berlin triology; few synths
David Bowie	Lodger	May 1979	Remaster 1999, very few synths, last of Eno/Bowie collaboration
Devo	The Essentials: Devo	1978-1981	Singles from 2002 collection
Giorgio Moroder	From Here to Eternity	July 1977	Lays down a template for electronic dance music
Donna Summer	selected tracks	1977-1979	Giorgio Moroder/Pete Bellotte production with prominent synth instrumentation, from The Dance Collection (1987), Once Upon a Time (1977) and Bad Girls (1979), including the single I Feel Love (1977)
Sparks	No. 1 in Heaven	September 1979	Giorgio Moroder produced
Human League	Reproduction	October 1979	remaster 2003
Human League	Travelogue	May 1980	remaster 2003
Human League	Dare!	20 October 1981	Martin Rushent produced, after split in original Human League line-up
Heaven 17	Penthouse and Pavement	September 1981	Some members of original Human League
Cabaret Voltaire	The Original Sound of Sheffield '78/'82. Best Of;	1978-1981	2002 remaster
Ultravox	Ultravox!	25 February 1977	more guitar band at the earlier stages
Ultravox	Ha!Ha!Ha!	14 October 1977	synthesizer creeping in
Ultravox	Systems of Romance	8 September 1978	Conny Plank producer, last with John Foxx
Ultravox	Vienna	11 July 1980	Conny Plank producer
Ultravox	Rage in Eden	7 September 1981	Conny Plank producer
John Foxx	Metamatic	17 January 1980	
John Foxx	The Garden	25 September 1981	
Yellow Magic Orchestra	Yellow Magic Orchestra	25 November 1978	even includes 8 bit computer game music
Yellow Magic Orchestra	Solid State Survivor	25 September 1979	more pioneering Japanese synth pop
Visage	Visage	November 1980	Containing 'Fade to Grey'
Buggles	The Age of Plastic	Jan 1980	Containing 'Video Killed the Radio Star'
Orchestral Manoeuvres in the Dark (OMD)	eponymous debut	22 February 1980	2003 remaster
OMD	Organisation	24 October 1980	2003 remaster
OMD	Architecture and Morality	8 November 1981	2003 remaster
Gary Numan and Tubeway Army	Replicas	April 1979	Numan grabs the pop centre ground
Gary Numan	The Pleasure Principle	September 1979	
Gary Numan	Telekon	5 September 1980	
Gary Numan	Dance	September 1981	Chart impact wanes
Silicon Teens	Music for Parties	1 September 1980	Daniel Miller's prototype synth pop group
Fad Gadget	Fireside Favourites	1 September 1980	
Fad Gadget	The Best Of Fad Gadget	1980-1981	tracks up to 1981
Depeche Mode	Speak & Spell	October 5 1981(first single Feb 20, 1981)	2006 remaster

Table 2. Table of recordings under comparison

[5] Irène Deliège and Geraint A. Wiggins, editors. *Musical Creativity: Multidisciplinary Research in Theory and Practice*. Psychology Press, London, 2006.

[6] D. C. Dennett. *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. Simon and Schuster, 1996.

[7] J. Stephen Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37:295–340, 2003.

[8] Ben Fields, Michael Casey, Kurt Jacobson, and Mark Sandler. Do you sound like your friends? Exploring artist similarity via artist social network relationships and audio signal processing. In *Proceedings of the International Computer Music Conference (ICMC)*, 2008.

[9] Arthur Hedley, editor. *Selected Correspondence of Fryderyk Chopin*. William Heinemann Ltd, London, 1962.

[10] Scott Johnson. The counterpoint of species. In John Zorn, editor, *Arcana: Musicians on Music*, pages 18–58. Granary Books, Inc., New York, NY, 2000.

[11] C. McKay and I. Fujinaga. Style-independent computer-assisted exploratory analysis of large music collections. *Journal of Interdisciplinary Music Studies*, 1(1):63–85, 2007.

[12] Jonathan Miller. *Stripped: Depeche Mode*. Omnibus Press, London, 2003.

[13] Juyong Park, Oscar Celma, Markus Koppenberger, Pedro Cano, and Javier M. Buldú. The social network of contemporary popular musicians. *International Journal of Bifurcation and Chaos (IJBC)*, 17:2281 – 2288, 2007.

[14] Simon Reynolds. *Rip It Up and Start Again*. Faber and Faber Limited, London, 2005.

[15] Robert J. Sternberg, editor. *Handbook of Creativity*. Cambridge University Press, Cambridge, 1999.

[16] Michael Talbot, editor. *The Musical Work: Reality or Invention?* University of Liverpool Press, Liverpool, 2000.

[17] George Tzanetakis and Perry Cook. Marsyas: a framework for audio analysis. *Organised Sound*, 4:169–175, 2000.

[18] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques (2nd Ed)*. Morgan Kaufmann Publishers, San Francisco, 2005.

[19] Mark Zadel and Ichiro Fujinaga. Web services for music information retrieval. In *Proceedings of the International Symposium on Music Information Retrieval*, Barcelona, Spain, 2004.

Crowdsourcing Music Similarity Judgments using Mechanical Turk

Jin Ha Lee
University of Washington
jinhalee@uw.edu

ABSTRACT

Collecting human judgments for music similarity evaluation has always been a difficult and time consuming task. This paper explores the viability of Amazon Mechanical Turk (MTurk) for collecting human judgments for audio music similarity evaluation tasks. We compared the similarity judgments collected from Evalutron6000 (E6K) and MTurk using the Music Information Retrieval Evaluation eXchange 2009 Audio Music Similarity and Retrieval task dataset. Our data show that the results are highly comparable, and MTurk may be a useful method for collecting subjective ground truth data. Furthermore, there are several benefits to using MTurk over the traditional E6K infrastructure. We conclude that using MTurk is a practical alternative of music similarity when it is used with some precautions.

1. INTRODUCTION

A constant source of frustration for designers and developers of music information retrieval systems is finding users to generate ground truth for evaluation. This is particularly true in music similarity tasks where algorithms are attempting to model some aspect of human intuition or understanding and predict the similarity among a set of songs. Getting humans to verify the results of these algorithms is tedious as a modest collection of several hundred tracks can require tens of thousands of pair-wise comparisons which potentially need to be evaluated.

Our motivation for this study is to explore the usefulness of Amazon Mechanical Turk (MTurk) (http://mturk.com) for collecting the human judgments necessary for evaluating music similarity tasks like the Audio Music Similarity (AMS) and Symbolic Melodic Similarity (SMS) tasks in the Music Information Retrieval Evaluation eXchange (MIREX). In this paper, we compare the similarity judgments obtained from MTurk and Evalutron6000 (E6K) on the same data set used in the MIREX 2009 AMS task. We also compare how these judgments affect the ultimate evaluation outcomes as published by the International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) in the annual MIREX evaluation. Additionally, we were interested in exploring how MTurk could be used to supplement or replace E6K in future music similarity evaluations, opening the possibility for continuous evaluation without incurring the overhead of a full MIREX/E6K-based evaluation.

2. BACKGROUND

The AMS and SMS tasks were carried out as part of MIREX using the E6K infrastructure. Both tasks rely on human judgments of music similarity as ground truth for evaluation of algorithm performance. Every year the IMIRSEL group at the University of Illinois seeks volunteers from the ISMIR community to complete a set of similarity judgments. In addition to the MIREX AMS and SMS tasks, a number of studies have looked at the human judgments of music similarity; to name a few, Aucouturier & Pachet [2], Ellis et al. [5], Berenzweigh et al. [4], Timmers [13], Schubert & Stevens [11], and Novello & McKinney [10].

In these studies, the human judgments were collected by a web survey or by recruiting a number of subjects including musicians and non-experts. Two typical methods were used. In some studies, the users were presented a set of three song excerpts (triads) and were asked to choose the most similar and most dissimilar of the three possible pairs. In other studies, the users were presented with pairs of song excerpts and were asked to rate the similarity between the pairs. Regardless of which method was used, collecting human similarity judgments has always been a challenging, expensive, and time-consuming process. Searching for a better model for obtaining human similarity judgments is especially important considering the fact that the general trend in recent MIREX AMS submissions is to submit multiple variations of an algorithm; there were a total of 15 submissions from 9 participants in 2009 compared to 6 submissions from 5 participants in 2006. There is also a trend towards larger datasets, and evaluating more queries [9].

2.1 Evalutron6000 (E6K)

IMIRSEL collect similarity judgments from human graders using E6K which is in the form of a web-based survey. The graders are supposed to be music experts since they are volunteers from the ISMIR community who have backgrounds in music or music-related research. Collecting human judgments is a long and arduous process every

year since the organizers must rely on volunteer labor. Every year it takes days to weeks to complete the evaluation. Table 1 shows the number of days it took to collect the human similarity judgments for the AMS and SMS tasks in past MIREX cycles.

	AMS	SMS
2006	15 days	18 days
2007	8 days	4 days
2009	14 days	n/a

Table 1. Number of days needed to collect human similarity judgments in past MIREX cycles.

2.2 Amazon Mechanical Turk

Amazon Mechanical Turk (MTurk) is a service which allows people to leverage human-computational power at scale to complete large numbers of tasks requiring human-intervention, cheaply and efficiently. Requesters upload tasks to the service, where they are matched with willing workers. Payment is mediated by Amazon, with a small per-task fee charged to the requester.

Tasks in MTurk are called HITs (Human Intelligence Tasks). Requesters define their HITs using an HTML-based template language and a data source which is used to populate the templates and generate the individual HITs. The requester also offers a payment amount and time limit for each HIT, and can limit who is able to complete the HITs, such as requiring workers to have a minimum percentage of previously accepted HITs.

Requesters can create a qualification test and require workers to have to pass it before being eligible to work on their HITs. Upon completion of a HIT, the requester can review the work and approve or reject payment on HITs individually. Furthermore, workers can be blocked by the requester which would reject all their un-approved HITs and prevent them from completing and submitting additional HITs for that same task.

Workers in MTurk call themselves Turkers. There are over 200,000 Turkers, from all parts of the globe. Ipeirotis [6] conducted a survey of 1,000 Turkers in February 2010. In total, Turkers represented 66 different countries, with 46.8% from the United States, and 34% from India. Among US workers, most (65.6%) are women; however, among Indian workers, most (70%) are men. 62.8% of respondents had a Bachelor's degree or higher.

MTurk has an API through which HITs can be created and results approved and downloaded, making it readily integratable into automated processes. For example, MTurk is successfully used to complete tasks such as filtering user-generated web content, searching through satellite photos for missing aircraft [3], and is gaining traction as a resource in research. Within the SIGIR community, MTurk has been proposed for use in generating relevance judgments in TREC-like evaluations [1]. Alonso and Mizzaro [1] compared MTurk to TREC experts and found the results from MTurk to be comparable to TREC's expert-generated ground truth data. They even claim Turkers found several errors in the TREC data.

Snow, et al. [12] have explored using MTurk for generating ground truth data for several kinds of Natural Language Processing tasks, including determining valence and affect in text, and assigning similarity scores. They found it possible to obtain results on par with those of domain experts.

Kittur et al [8] used MTurk to rate the quality of Wikipedia articles. Among their experiments, they found a naïve approach of simply asking Turkers to rate an article led to inconsistent responses which did not correlate strongly with ratings given by experts. However, when they redesigned the HITs to include several verifiable questions which could be used to filter out "bad" responses, the results improved significantly. They argue that the verification questions serve two purposes: first, they allow the requester to assess the quality of the response; and second, they signal to the Turkers that their responses are being scrutinized.

3. RESEARCH QUESTIONS & STUDY DESIGN

In this paper, we explore two main research questions:

I) How do music similarity judgments obtained from Mechanical Turk compare to those collected from music experts in the Evalutron6000?; and

II) How do evaluation outcomes for tasks like MIREX's Audio Music Similarity evaluation differ when based on similarity judgments collected from Mechanical Turk as compared to Evalutron6000?

To study these questions, we replicated the E6K similarity assessment and subsequent evaluation of the AMS task in the 2009 MIREX evaluation using MTurk. We obtained the query-candidate (QC) results lists from the IMIRSEL lab, consisting of 100 queries, and the top 5 candidates per query returned from the 15 participating algorithms in MIREX 2009. There were a total of 6,732 unique QC pairs which needed to be judged.

In order to keep the amount of work in each HIT reasonable, we limited the number of similarity judgments per HIT to 15 QC pairs, and all QC pairs in a HIT shared the same query. Among the 15 QC pairs in a HIT, two candidates were included for checking the quality of the ratings. One was an identity check and it asked the Turker to rate the similarity of the query to itself. The Turker should indicate that this candidate is "Very Similar (VS)" to the query as they are identical. This was also done in E6K in 2009; however, we were unable to locate those data published for comparison.

The other quality check was a consistency check; the same candidate was included twice in a single HIT, once towards the beginning, and again towards the end of the list of candidates. The expectation here was that the Turker should provide the same response for both in-

stances since they are the same candidate. Excluding these two QC pairs for checking the quality of the results, there were 13 unique QC pairs in an individual HIT. The quality checks were mixed among the other candidates and were not specially demarcated in any way.

The list of all candidates for each query was broken down into multiple HITs containing 13 unique QC pairs. In the event that the last HIT contained less than 13 candidates, the list was padded to 13 with additional candidates selected from that query's other HITs. These padded judgments were not used in the evaluation. Each HIT was completed by single Turker, with the possibility that a single query could be evaluated by multiple Turkers. This was different from MIREX 2009 AMS where a single grader was responsible for judging all candidates for a single query, but is similar to MIREX 2006 AMS, where candidate lists were divided among multiple graders.

A total of 583 HITs were created, and we offered $0.20 per completed HIT. Instructions similar to what are given in E6K were given to Turkers as shown in Figure 1. Figure 2 shows a partial screenshot of MTurk's evaluation page. We tried to reproduce the E6K interface as much as possible. The Turkers were asked to rate the similarity on the E6K BROAD scale (Not Similar, Somewhat Similar, and Very Similar) and were not asked to provide a FINE score (0-10) in order to simplify the task. Additionally, we used the Yahoo! Media Player in the MTurk version of the interface, rather than the E6K player because it was much simpler to use and has much better cross-browser compatibility.

In addition to the HITs described above, we created 4 more HITs in order to see how much variability there was among the Turkers' responses. In these HITs, we took one candidate from each query, put 15 QC pairs into each HIT and had 3 different Turkers complete each HIT. This gave us multiple ratings for the same QC pairs, and allowed us to test the inter-rater agreement. We paid $0.20 for each of these 12 HITs. The total cost for all 595 HITs, including Amazon's administrative fees, was $130.90. Ultimately, we paid less than $0.02 per usable judgment, for a rate of approximately 53 judgments per US dollar.

4. DATA & OBSERVATIONS

In total we collected 15,705 similarity judgments from 1,047 submitted HITs, plus 180 additional judgments created to test agreement among the Turkers. Of the 1,047 HITs submitted, we approved 583 (55.7%), and rejected 464 (44.3%). We rejected HITs which were missing responses, those which were completed too quickly (less than 45 seconds), those in which Turkers failed to assign a Very Similar score to the identity case of a query compared to itself, and those in which Turkers assigned two different scores to the same candidate repeated in the list. Accounting for the rejected HITs, the integrity-check judgments, and for list padding, we ended up with 6,732 unique judgments. Even having to discard almost half the judgments, we were still able to obtain the needed results in less than 12 hours, an order of magnitude faster than the average E6K cycle (see Table 1).

How similar are these songs?

Listen to the following pairs of song clips, the 'query' is the same for all 15 pairs. Evaluate how 'musically similar' each candidate is to the given query. You will be presented with songs from a number of different music genres. Please assign the scores according to what you find 'sounds' similar and do not take into account whether you like the music or not. Provide your best estimation of the similarity for each pair. You should listen to a reasonable portion of every candidate before making your judgment. Answers which are incomplete or missing responses will be rejected. Answers which do not appear to contain honest judgments will be rejected.

Assign a similarity rating using the following 3-point scale:

- Not Similar
- Somewhat Similar
- Very Similar

Figure 1. Instructions given to Turkers are based on the instructions given to E6K graders.

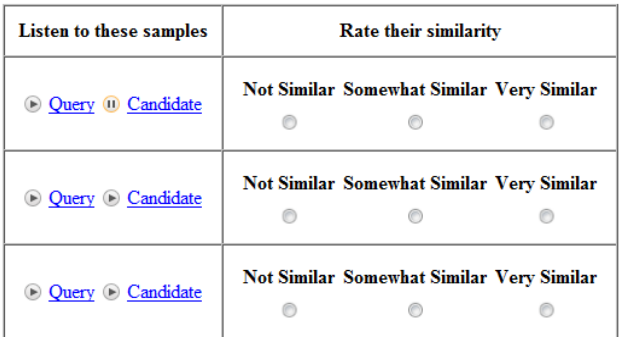


Figure 2. Partial screenshot of an MTurk HIT.

Research Question I: *How do music similarity judgments obtained from Mechanical Turk compare to those collected from music experts in the Evalutron6000?*

The similarity scores derived from MTurk are different from those obtained from E6K, but they are not entirely incomparable. We compared the 6,732 similarity judgments obtained from MTurk to the 6,732 judgments obtained via E6K in AMS 2009 for the same set of QC pairs. We measured the percent-agreement between the MTurk results and the E6K results, and found that 54.6% of the pair-wise ratings were the same. Agreement increases to 72.4% when we consider similarity as a binary decision (Very Similar & Somewhat Similar vs. Not Similar). To our knowledge, E6K has not been used to do multiple evaluations of the same data set in this way, so we do not have a basis for comparison. However, the relatively low agreement between MTurk and E6K does underscore the subjective nature of similarity ratings and similarity-based tasks in general.

The two sets of similarity judgments (MTurk & E6K) have a Pearson's correlation of r=0.495, which while not particularly strong, is comparable to the correlation

(r=0.433) found by Snow [12] between NLP experts and Turkers in an affect annotation study.

Looking at the data in aggregate, Figure 3 shows the similarity judgments derived from MTurk tended to skew towards Not Similar (NS=3,605), where as E6K graders tended to assign similarity scores more uniformly across the categories. In AMS 2009, as a requirement of participation each team had to provide an E6K volunteer to help with the judging for each algorithm they submitted. These volunteers might have had some stake in the outcome of the evaluation which might explain the greater proportion of VS scores in 2009 compared to 2006 and MTurk where the ratings were generated by independent volunteers. However, Figure 3 also shows the distribution of scores from previous MIREX cycles, and while the underlying queries and candidates differ across the years, the distribution of scores from MTurk is not dissimilar to other distributions from previous E6K results.

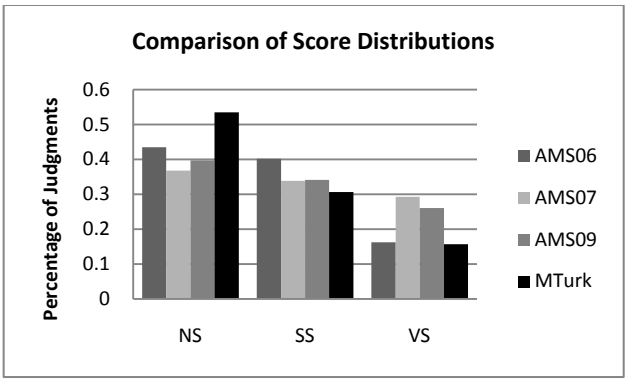


Figure 3. Distributions of scores from the 3 years of MIREX AMS evaluation using E6K compared to the distribution of scores derived from MTurk.

In order to further investigate the similarity of MTurk judgments to E6K judgments, we collected multiple similarity judgments over the same set of QC pairs from several different Turkers. Specifically, we randomly selected one candidate from each candidate list for each of the 60 queries used in MIREX 2006 AMS, and created 4 lists of 15 QC pairs to be evaluated by three different Turkers. This setup is very similar to how E6K was configured in 2006 (multiple graders rating portions of candidate lists), and given we were working with a subset of the 2006 data, we are able to compare the inter-Turker agreement to that found by Jones, et al. [7]. While we did not test SMS in MTurk, we have provided the data for comparison.

Table 2 shows the agreement using the 3-level and 2-level analysis used in [7]. The overall distribution of scores is fairly similar to the AMS 2006 results; the proportions of QC pairs across the various levels of agreement are comparable between the two data sources. The percentage of cases of total agreement are the same, and there is some shifting of cases between partial and total disagreement with slightly more cases of total disagreement (VS,SS,NS) among Turkers. This may be due to the nature of the QC pairs sampled for this evaluation which is plausible given the small sample size, or it may be inherent given what [7] describes the vague definition of "music similarity".

3-level	**SMS 2006**		**AMS 2006**		**MTURK**	
VS,VS,VS	114	12.6%	61	3.7%	4	6.7%
SS,SS,SS	38	4.3%	137	8.4%	1	1.7%
NS,NS,NS	263	29.1%	293	18.0%	13	21.7%
Triples	**415**	**45.9%**	**491**	**30.1%**	**18**	**30.0%**
VS,VS,*	24	2.7%	150	9.2%	3	5.0%
SS,SS,*	158	17.5%	469	28.8%	18	30.0%
NS,NS,*	288	31.8%	404	24.8%	11	18.3%
Doubles	**470**	**51.9%**	**1023**	**62.8%**	**32**	**53.3%**
VS,SS,NS	20	2.2%	115	7.1%	10	16.7%
2-level	**SMS 2006**		**AMS 2006**		**MTURK**	
S,S,S	188	20.8%	494	30.3%	19	31.7%
NS,NS,NS	263	29.1%	293	18.0%	13	21.7%
Triples	**451**	**49.8%**	**787**	**48.3%**	**32**	**53.3%**
S,S,N	166	18.3%	438	26.9%	17	28.3%
N,N,S	288	31.8%	404	24.8%	11	18.3%
Doubles	**454**	**50.2%**	**842**	**51.7%**	**28**	**46.7%**

Table 2. Comparison of disagreement among Turkers and E6K graders from MIREX 2006 AMS evaluation.

When we examine the results using a binary similarity measure (SS+VS against NS), we see greater similarity between the E6K graders and the Turkers. The distributions across the levels of agreement are nearly identical between the two sets. Jones [7] also found greater consensus when considering similarity on a binary scale, and suggest that the binary metric might be sufficient for the task of evaluation.

Research Question II: *How do evaluation outcomes for tasks like MIREX's Audio Music Similarity evaluation differ when based on similarity judgments collected from Mechanical Turk as compared to Evalutron6000?*

Given the similarity judgments derived from MTurk appear to be different from those generated via E6K, we wished to see if those differences have any substantial bearing on MIREX evaluations. It is possible that the individual ratings differ, but still produce similar outcomes in comparing the performance of individual algorithms in the audio music similarity task. Conversely, the differences may in fact be substantive and produce significantly different end results.

MIREX evaluates the performance of similarity algorithms using Friedman test with repeated-measures. Figure 4 shows a graphical depiction of the results of the MIREX 2009 AMS Friedman evaluation, comparing the average rankings among the different algorithms. There are clearly two distinct groupings to the data: ANO, BSWH1, BSWH2, CL2, GT, LR, PS1, PS2, SH1, SH2; and BF1, BF2, CL1, ME1, ME2. One way to interpret the figure is that all algorithms in one group are significantly different from all algorithms in the other group, but within the groups the algorithms are not all significantly dif-

ferent from each other. So, while PS2 does appear to lie slightly outside the rest of the larger group, it is not significantly different from all members of that group (it overlaps partially with PS1).

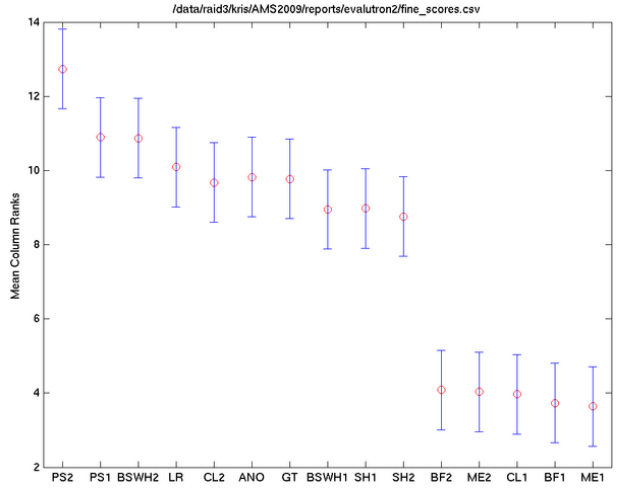


Figure 4. Friedman rank comparison for MIREX 2009 AMS based on judgments from E6K (from [9]).

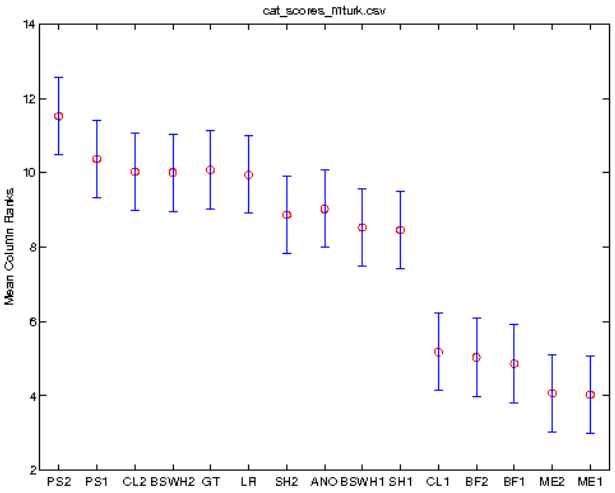


Figure 5. Friedman rank comparison for MIREX 2009 AMS based on judgments from MTurk.

Figure 5 shows the Friedman evaluation results performed using the similarity judgments derived from MTurk. As we can see, the two main groupings are still evident, with clearly significant differences in the performance of the algorithms between the groups. However, the gap between the groups is narrower, and the data are more compact. Furthermore, the global ordering of the algorithms has changed. Notwithstanding these differences, the overall results are remarkably similar.

The main significant differences between the groupings are still evident and significant differences are still preserved among most of the algorithm pairs. Table 3 summarizes the differences between the E6K and MTurk results. Out of the 105 possible pair-wise comparisons among the 15 algorithms submitted to MIREX 2009 AMS, only six (5.7%) algorithm pairings were determined to be significantly different based on E6K judgments and not found to be significantly different based on MTurk judgments. No algorithms which were not significantly different under E6K were found to be significantly different under MTurk. This discrepancy is not substantially different compared to the Friedman results computed using the E6K BROAD scores and FINE scores in AMS 2009. The Friedman test based on the FINE scores rates 3 (2.9%) algorithm-pairs differently than the BROAD score results [9].

Algorithm 1	Algorithm 2	Significant in E6K?	Significant in MTurk?
BSWH2	SH1	TRUE	FALSE
PS1	SH1	TRUE	FALSE
PS1	SH2	TRUE	FALSE
PS2	CL2	TRUE	FALSE
PS2	GT	TRUE	FALSE
PS2	LR	TRUE	FALSE

Table 3. Excerpt from Friedman table of differences in significance between E6K and MTurk.

5. IMPLICATIONS FOR MIR EVALUATION

Overall, we were quite impressed with the quality of the data we were able to obtain from MTurk. We can identify many benefits to using MTurk in MIR evaluation, some of which include:

1. The tasks were inexpensive to submit, costing approximately USD$0.02 per judgment, although other researchers have obtained quality results for far less payment (c.f., [1],[12]).
2. Evaluation was fast, taking less than 12 hours to complete. It took over 2 weeks to obtain the same number of judgments using E6K.
3. MTurk has a scriptable API, meaning it can be used for "on-demand" evaluation. This is especially attractive given the developments of the MIREX Do-It-Yourself infrastructure.
4. No judging fatigue. MTurk provides a nearly endless supply of willing labor, and it is not feasible to expect the ISMIR community to continuously provide input for use in MIREX DIY evaluations. Several Turkers sent us messages saying they found the HITs more fun than most other HITs on MTurk, one even expressed a willingness to work for free.
5. Using MTurk for similarity judgments avoids any conflict of interest inherent in asking participants or their labmates to evaluate the results.
6. MTurk provides a mechanism for compensating volunteers for their time and effort.
7. MTurk is considered "exempt" by human subjects review as it falls under the description of a web survey, or as paid workers.
8. Using MTurk does not preclude restricting participation in the evaluation to only ISMIR members. It is possible to require Turkers obtain a qualification prior to working on any HITs. Qualification credentials could include membership in the ISMIR Society.

9. MTurk is very stable. It is built on Amazon's cloud infrastructure and is very robust.

However, there are several limitations which need to be kept in mind when using MTurk, including:

1. HITs need to contain validation questions which allow you to check responses for quality and consistency. Early in our exploration of MTurk we created HITs without validation questions and the data we collected was highly inconsistent.
2. The instructions we provided also needed clarification over several tests. We found it helpful to spell out the precise conditions why a HIT would be rejected in the instructions. Likewise, you need to provide clear explanations why HITs were rejected.
3. It does cost money to use MTurk. While each HIT is cheap, in aggregate the price adds up quickly. Amazon's overhead is either $0.005 or 10% per HIT, whichever is greater. However, the total costs are small compared to the value of E6K volunteers' time.
4. Some university human subjects review boards might not be very familiar with MTurk and might be unsure how to handle it. This could slow applications.

6. CONCLUSIONS

Our data show that while the specific judgments may differ, using MTurk produces comparable results to using E6K for collecting human similarity judgments. The differences are not dissimilar to the findings of other studies of MTurk and do not significantly alter the MIREX evaluation outcomes, indicating that MTurk may be a used as a reliable source of similarity judgments for audio-based music similarity comparisons. Overall, the differences between MTurk and E6K judgments resulted in a 5.7% difference in the ultimate outcome of the Friedman test comparing the 15 algorithms submitted to AMS 2009.

We are excited by the possibilities MTurk offers to the development of future evaluation infrastructure, like the MIREX DIY, but are most excited by what MTurk can do for the community as a whole. There are many types of data which could be collected from Turkers, and it remains to be seen how well MTurk is suited for collecting those data. In future work we would like to explore other data types; for example, music mood labels, music tagging, onset annotation work, key-identification, humming or singing, tapping, transcribing, etc.

7. ACKNOWLEDGEMENTS

We would like to thank IMIRSEL for providing us with access to the MIREX data and helping us to reproduce the MIREX analyses.

8. REFERENCES

[1] O. Alonso and S. Mizzaro: "Can we get rid of TREC assessors? Using Mechanical Turk for relevance assessment," *Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation*, pp. 15-16, 2009.

[2] J-J. Aucouturier and F. Pachet: "Music Similarity Measures: What's the Use?" *Proceedings of the 3rd International Society of Music Information Retrieval (ISMIR) Conference,* pp. 157-163, 2002.

[3] D. Axe; "Geeks Spot Fossett?" Wired, Retrieved from http://www.wired.com/dangerroom/2007/09/geeks-spot-foss/comment-page-3/, 2007.

[4] A. Berenzweig, B. Logan, D. P. W. Ellis, and B. Whitman: "A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures," *Proceedings of the 4th ISMIR,* pp. 99–105, 2003.

[5] D. P. W. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence: "The Quest For Ground Truth in Musical Artist Similarity," *Proceedings of the 3rd ISMIR Conference,* pp. 170-177, 2002.

[6] P. G. Ipeirotis: "Demographics of Mechanical Turk," CeDER Working Papers, Retrieved from http://hdl.handle.net/2451/29585, 2010.

[7] M. C. Jones, J. S. Downie, and A. F. Ehmann: "Human Similarity Judgments: Implications for the Design of Formal Evaluations," *Proceedings of the 8th ISMIR,* pp. 539-542, 2007.

[8] A. Kittur, E. H. Chi, and B. Suh: "Crowdsourcing User Studies With Mechanical Turk," *CHI 2008 Proceedings,* pp. 453-456, 2008.

[9] MIREX 2009 Wiki, Retrieved from http://music-ir.org/mirex/2009/, 2009.

[10] A. Novello and M. F. McKinney: "Assessment of Perceptual Music Similarity," *Proceedings of the 8th ISMIR,* pp. 111-112, 2007.

[11] E. Schubert and C. Stevens: "The Effect of Implied Harmony, Contour and Musical Expertise on Judgments of Similarity of Familiar Melodies," *Journal of New Music Research,* Vol. 35, No. 2, pp. 161-174, 2006.

[12] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng: "Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks," *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* pp. 254-263, 2008.

[13] R. Timmers: "Predicting the Similarity Between Expressive Performances of Music From Measurements of Tempo and Dynamics," *Journal of the Acoustical Society of America,* Vol. 117, No. 1, pp. 391-399, 2005.

DECOMPOSITION INTO AUTONOMOUS AND COMPARABLE BLOCKS : A STRUCTURAL DESCRIPTION OF MUSIC PIECES

Frédéric BIMBOT [1] frederic.bimbot@irisa.fr
Olivier LE BLOUCH [2] olivier.le_blouch@inria.fr
Gabriel SARGENT [3] gabriel.sargent@irisa.fr
Emmanuel VINCENT [2] emmanuel.vincent@inria.fr

METISS - Speech and Audio Research Group

(1) IRISA, CNRS - UMR 6074 - (2) INRIA, Rennes Bretagne Atlantique – (3) Université de Rennes 1
Campus Universitaire de Beaulieu, 35042 RENNES cedex, France

ABSTRACT

The structure of a music piece is a concept which is often referred to in various areas of music sciences and technologies, but for which there is no commonly agreed definition. This raises a methodological issue in MIR, when designing and evaluating automatic structure inference algorithms. It also strongly limits the possibility to produce consistent large-scale annotation datasets in a cooperative manner.

This article proposes an approach called *decomposition into autonomous and comparable blocks*, based on principles inspired from structuralism and generativism. It specifies a methodology for producing music structure annotation by human listeners based on simple criteria and resorting solely to the listening experience of the annotator.

We show on a development set that the proposed approach can provide a reasonable level of concordance across annotators and we introduce a set of annotations on the RWC database, intended to be released to the MIR community.

1. INTRODUCTION

1.1 Motivations

The automatic inference of musical structure is a research area of growing interest [1-6], which is illustrated for instance by the creation in 2009 of a task called *structural segmentation* in the MIREX community [7], or the existence of a specific research topic called *music structuring and summarization* in the QUAERO project (started 2008) [8].

Inference of musical structure has multiple applications, such as fast browsing of musical contents, automatic music summarization, chorus detection, unsupervised mash-ups, music thumb-nailing, etc… but also, more fundamentally, it offers great potential for improving the acoustic and musicological modeling of a piece of music with the help of structural information such as the relative position of events within structural elements or the exploitation of recurring similarities across them.

Musical structure deals with the description of the formal organization of music pieces. However, several conceptions of musical structure coexist and there is no widely accepted definition. This raises a methodological issue when the question arises of evaluating and comparing automatic algorithms on a common "ground-truth" (see, in particular [9])

This article presents an attempt to provide an operational definition and to specify an annotation procedure for producing a structural description of music pieces that can be obtained quasi-univocally and in a reproducible way by several human annotators.

The concepts and the methodology proposed in this article are intended to be applied to what we will call *conventional music*, which covers a large proportion of current western popular music and also a large subset of classical music. However, we keep in mind that some other types of music (in particular, contemporaneous music) are much less suited to the proposed approach.

1.2 Objectives

The concepts and methodology presented in this work aim at specifying a process for annotating musical structure, with the following requirements :

- based on the *listening experience* of the annotator (and not on his/her *musicological expertise*)
- *unrelated* to any particular algorithm or application
- *independent* of any *musical role* assigned to structural elements
- *reproducible* across annotators
- *applicable* to a wide variety of music genres

At the current stage of our work, we have focused on the issue of locating structural elements (i.e. segmentation) and we postpone to a later step the question of how to label these elements.

We first present, in section 2, the fundamentals of our approach, then we describe, in section 3, the proposed annotation process. We provide in section 4, an evaluation of the consistency of the methodology on a development set and we introduce to the MIR community a set of annotations on the RWC [10] Pop data set, based on the proposed approach.

1.3 Preliminary definitions

In the rest of this paper, we consider that a piece of music is characterized by *3 reference properties*, which may be constant or vary over time :

- tonality/modality (reference key and scale)
- tempo (speed / pace of the piece)
- timbre (instrumentation / audio texture)

We also consider that a piece of music shows *4 levels of temporal organization :*

- rhythm (relative duration and accentuation of notes)
- harmony (chord progression)
- melody (pitch intervals between successive notes)
- lyrics (linguistic content)

These levels of description form *7 musical layers* which we consider independently.

2. FUNDAMENTALS

2.1 Framework

The proposed approach relies on concepts inspired from *structuralism*, a school of thoughts initiated by Ferdinand de Saussure in the field of Linguistics [11] and later extended to other domains, in particular to some areas of music semiotics. Our approach also borrows ideas from *generative theory* as explored by Lerdahl and Jackendoff [12].

In this context, we consider a music piece as the layout of a number of constitutive elements governed by a specific assembling process, called *syntagmatic process*. The constitutive elements are related to one another through *paradigmatic relationships* which manifest themselves as a set of *equivalence classes* (i.e. two elements belong to the same subset if and only if the relation holds between them). The entire scheme forms a *system* in the structuralist sense, namely an "entity of internal dependencies", according to Hjelmslev's definition [13].

The piece of music thus appears as a particular observation produced by this system and the problem of musical structure inference consists in determining the constitutive elements of the piece (i.e. *segmentation* task or, more generally speaking, *decomposition*) and to assign equivalence classes to each of them (*labeling* or *tagging* task).

As a consequence, specifying a type of musical structure requires the definition of :

1. the nature and properties of the constitutive elements
2. the assembling process used to combine them
3. the equivalence relation(s) that are referred to, so as to relate them to one another.

2.2 Working assumptions

In the present work, the constitutive elements are assumed to be common to the 4 levels of temporal organization. They are limited in time and are assembled mainly by concatenation. They are called *blocks*.

A block is defined as an *autonomous* segment (see 3.2). It is specified by a *starting instant*, a *duration* and a *size* (the concept of size is also detailed in 3.2). A block can be decomposed into a *stem* (which is itself autonomous) and one or several *affixes*.

Several equivalence relations can be considered and combined to qualify *comparability* between blocks, in particular: isometry (same size), interchangeability (possibility to swap), similarity in one or several layers, etc…

Thus, we approach music structure description as the process of decomposing the music piece into autonomous and comparable blocks. As a consequence, we elude the various hierarchical levels of music structure and focus on the segmental macro-organization of music pieces.

Blocks share similarities with musical phrases but are not strictly identifiable to them. The concept of blocks also has some connections with the notion of *periode* introduced by Riemann (see [14]) and that of *grouping structure* as developed in [12].

3. SPECIFICATIONS

In this section, we introduce a number of criteria which are used to specify as univocally as possible the structural decomposition of a music piece. We attempt to formulate these criteria without resorting to absolute acoustic properties of the musical segments nor to their musical role in the piece (chorus, verse, etc…), so as to remain as independent as possible from musical genre.

3.1 Musical consistency w.r.t. simple transformations

We base the decomposition process on the assumption that an annotator is able to decide on the *musical consistency* of a passage resulting from a local transformation of the music piece. More specifically, the listener is supposed to be able to judge if simple operations such as the suppression, insertion, substitution or repetition of a given musical segment within the music piece creates (or not) a morphological singularity or a syntagmatic disruption with respect to the original piece.

This approach assumes that the structural organization of the music piece is governed by underlying processes which the listener is able to infer (even though he/she may not be able to formulate them) and which he/she can refer to, to decide on the musical consistency of the transformed piece. In particular, the listener will be strongly inclined to consider that musical consistency is preserved by a transformation when a similar passage is found somewhere else in the piece, or *could be* found without creating any feeling of heterogeneity with the rest of the piece.

Clearly, this definition is partly subjective but we believe that it provides non-expert human listeners with an operational criterion that requires from them some familiarity with the genre of the music piece but does not demand a sharp expertise in musicology. Note that, in the same way, a human listener is generally able to tell whether a sentence in his/her native language is grammatical or not, even if he is not a linguist.

3.2 Properties of blocks

A block is defined as a musical segment which is *autonomous*, i.e. which fulfils one of the two following properties : either it is *independent* (i.e. it is perceived as self-sufficient when played on its own) or it is *iterable* (it can be looped and the result is musically consistent).

Moreover, blocks within a musical piece have the property of being *suppressible*, i.e. they can be removed from the piece without altering its musical consistency. This test is used to identify the most likely block boundaries. However, suppressibility is a necessary but not a sufficient condition to qualify a block.

It is also worth noting that blocks are not necessarily homogeneous : reference properties may evolve within a block (change of tonality, tempo modifications, appearance/disappearance of instruments or voice).

The *size* of a musical block is expressed as the number of times a listener would *snap* his fingers to accompany the music, at a rate which is as close as possible to 1 bps (beat per second). Conventionally, block boundaries are synchronized with the first beat of a bar. Occasionally, unusual situations may arise, such as blocks having a fractional size or for which the listener is unable to decide what the size is.

Blocks can contain *affixes*, i.e. portions which can be suppressed, yielding a reduced block which remains musically consistent with the rest of the piece. The various types of affixes are : *prefixes*, *suffixes* and *infixes* (the latter can be non-connex). A block can therefore be described as a *stem* combined with zero, one or several affixes. In general, affixes are short and not autonomous.

3.3 Equivalence relations and comparability

Several paradigmatic relationships between blocks can be considered :

- *isometry* : blocks of the same size (absolute isometry) or blocks reducible to stems of the same size (stem isometry).
- *interchangeability* : blocks that can be swapped within a music piece without altering its musical consistency.
- *similarity* : blocks identical across some of their musical layers (over the whole blocks or their stems only).
- *isomorphy* : blocks that can be obtained from each other by a transformation of their reference properties.

As mentioned earlier, these equivalence relations are resorted to in order to determine on what basis blocks are judged comparable with one another.

3.4 Structural pulsation and regularity

To specify further the decomposition into blocks, it is hypothesized that the structure of (most) music pieces is rather cyclic and therefore follows some form of regularity characterized by a small set of distinct block sizes.

We thus suppose that the music piece is built upon *structural pulsation periods* which take, in order of preference :

- One value *(type I)*
- Two values *(type II)*
- A limited set of values observed in a quasi-regular sequence, called *structural pattern (type III)*
- A limited set of values, showing no structural pattern *(type IV)*
- A large variety of distinct values *(type V)*

We also designate as *type 0* (or undeterminable) a piece for which it turns out to be impossible (for the listener) to locate clear boundaries of autonomous segments. Blocks whose size complies with one value of the structural pulsation periods are called *regular* blocks.

The regularity property induces decompositions which tend to yield comparable blocks within a given piece.

Figure 1 depicts a block-wise structural decomposition in the case of a *type I* structure (top) and illustrates several configurations of non-regular blocks (bottom) and their corresponding notations :

- Truncated block : block resulting from the suppression of a fragment inside a regular block
- Extended block : block obtained by the insertion of an affix into a regular block
- Bridging block : irregular block, usually of a smaller size, and which is often isolated at the beginning of the piece, at the end or in-between regular parts.
- Tiling : partially overlapping blocks (on all levels of organization simultaneously), as is the case for instance in canon singing or fugue-like compositions.

3.5 PIC minimization and target duration

The various properties introduced in the previous paragraphs still do not necessarily elicit a unique structural decomposition. Several possibilities may remain, generally based on structural pulsation periods which are multiples or sub-multiples of each other.

These situations can be decided between by specifying a target duration for regular blocks, the value of which we derive from the minimization of the *predominant informative context*.

Figure 2 illustrates a structural decomposition as a paradigmatic representation showing the correspondence between homologous parts across distinct blocks.

If this decomposition is exploited to predict the musical properties of the music piece on a short time interval, the most relevant portions of the piece for this purpose are, on the one hand, the neighboring portions belonging to the same block (horizontally) and the homologous portions across the other blocks (vertically).

This is what we call the Predominant Informative Context (or PIC). It is distinct for each small portion of the music piece and it is solely determined by the structure. It consti-

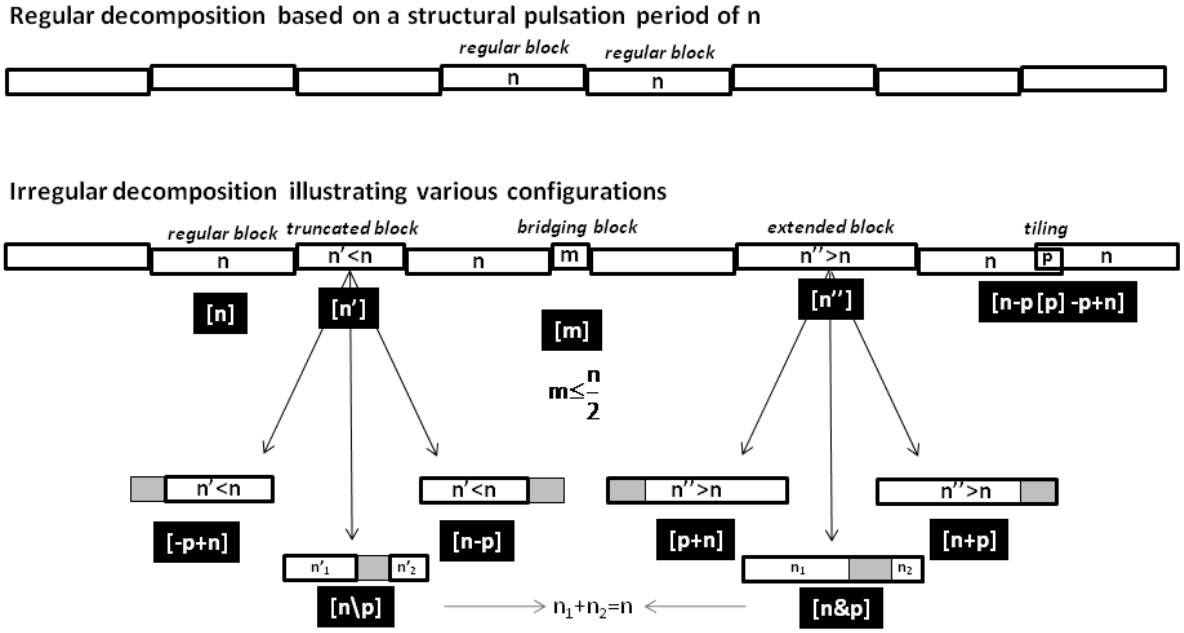


Figure 1 : *Decomposition into structural blocks (syntagmatic point of view) and the corresponding notation of block sizes.*

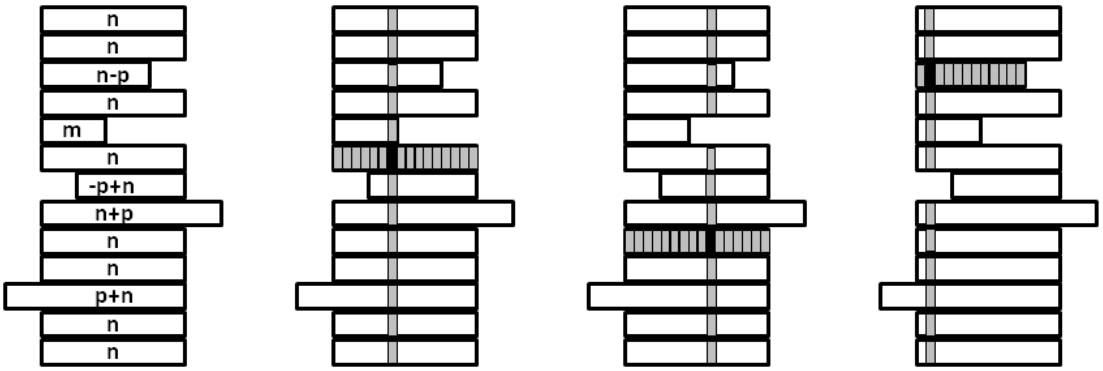


Figure 2 : *Paradigmatic representation of structural blocks and visualisation of the PIC.*
Any small portion (in black) presents privileged internal dependencies with the gray parts in the piece, namely other portions within the same structural block (horizontally) and homologous portions across other blocks (vertically)

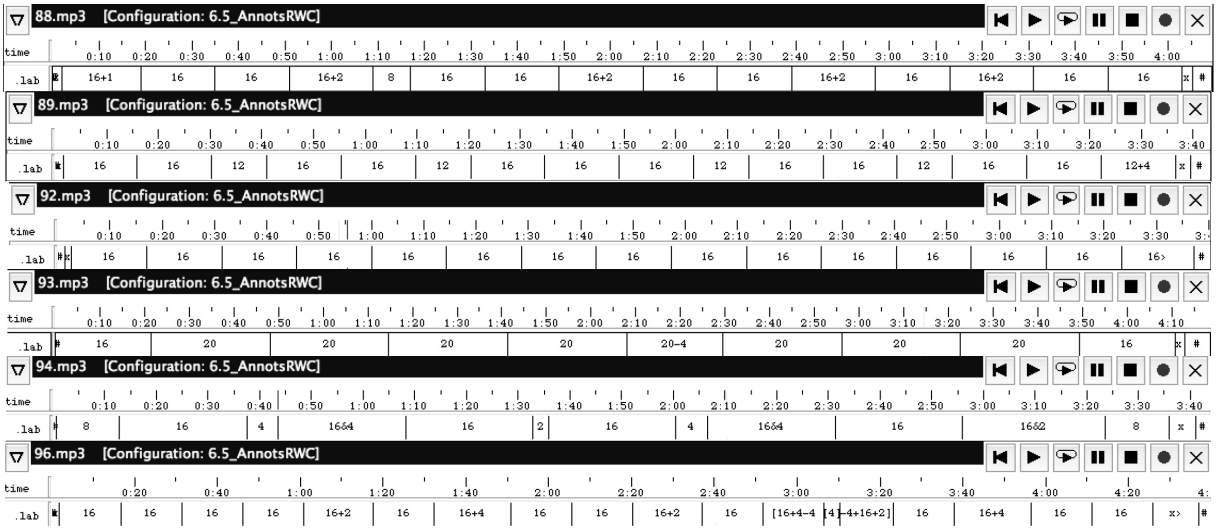


Figure 3 : *Example of block-wise decompositions obtained on 6 songs from the RWC Pop set (displayed with Wavesurfer).*

tutes the predominant source of information within the entity of internal dependencies mentioned in section 2.1.

If the total length of the music piece is equal to N and if the typical block length is equal to n, then, the number of blocks in the piece is in the order of N/n and the total coverage of the PIC is given by :

$$C = n + (N/n) - 2 \qquad (1)$$

which is minimal when $n = \sqrt{N}$.

Taking the second as a unit, and on the basis of music pieces with a typical length of 4 minutes (i.e. $N = 240$ seconds), the value of n which minimizes C is approximately equal 15.5 seconds.

In the present work, we retain the value of 15 s as the *target duration* for blocks, which leads to the following additional criterion : at least one of the structural pulsation periods must have a duration as close as possible to 15 s, on a logarithmic scale. This criterion tends to provide structural decompositions which result from a balanced contribution of the paradigmatic and syntagmatic axes.

More generally speaking a relative weight λ can be applied to the two terms in equation (1), leading to a *PIC function* which writes :

$$C(\lambda) = n + \lambda(N/n) - (\lambda+1) \qquad (2)$$

and whose minimization (in n) induces decompositions based on an adjustable balance between the syntagmatic and the paradigmatic axes.

The constraint resulting from a target duration criterion enables the disambiguation of situations such as several identical medium-size segments in sequence and it provides blocks which are more adapted to comparisons *across* music pieces.

3.6 Subsidiary criteria

In some residual situations, several competing decompositions may locally be compatible with a given structural pulsation period while fulfilling satisfactorily all other criteria. For instance, sequences of an odd number of suppressible segments which have a size equal to half of the structural pulsation period.

In these circumstances, the following *subsidiary criteria* are considered :

- group preferably in a same block short neighboring segments of this passage which are most similar
- favor decompositions that yield the largest possible number of blocks which are interchangeable with other blocks outside this passage.

3.7 Procedure (summary)

The box hereafter summarizes the annotation procedure resulting from the criteria presented above.

Once the process finalized, the annotator fills up a short report summing up the type of structure, the degree of difficulty, the level of confidence and any relevant information pertaining to the annotation of that piece.

1) Identify a plausible value n (or set of values $\{n_i\}$) for the structural pulsation period(s), from the parts of the piece which are structured with strong regularity. Choose in priority values as close as possible to the target duration.
2) Locate suppressible blocks of size n, whether they are regular or can be derived straightforwardly from regular blocks. Also search for tiling at this stage.
3) Continue the decomposition by resorting to less regular (suppressible) blocks considering in priority block sizes that are sub-multiple of n.
4) Assess the regularity of the decomposition, and find out to which type (0, I, II, III, IV or V) the decomposition tends to belong. The local structure around the beginning and the end of the piece should be given a lower importance and so should it be for affixes.
5) Consider other options for the value(s) of the structural pulsation period(s) and find out whether they would lead to a simpler decomposition.
6) Refine the decomposition by resolving remaining ambiguities with the help of the subsidiary criteria.

Figure 3 provides example of block-wise decompositions obtained on 6 songs from the Pop set of the RWC database.

4. EVALUATION AND DISSEMINATION

4.1 Evaluation protocol

In order to validate the annotation procedure proposed in this paper, we have measured the concordance between several annotators on a same annotation task.

Four annotators are provided with a development set of 20 songs in their audio version, the list of which was determined by IRCAM, in the context of task 6.5 of the QUAERO Project (Table 2).

The concordance between annotators is evaluated by taking them pair-wise and computing for each piece the F-measure between their annotations (with a tolerance of ± 0.75 s between segment boundaries) and averaging the F-measure over all 20 pieces.

Among the four annotators, none is a musicologist nor a professional musician. However, it is important to mention that they are the 4 co-authors of this paper, which may induce some methodological bias, which needs to be taken into account in interpreting the experimental results.

Annotator	N°1	N°2	N°3	N°4
N°1	-	88.9	95.7	92.9
N°2	88.9	-	88.7	88.7
N°3	95.7	88.7	-	92.8
N°4	92.9	88.7	92.8	-

Table 1 : *Pre-final concordance between annotators evaluated as the F-measure (%) between annotations averaged over 20 pieces of music. The mean value is 91.3 %.*

Scores presented in Table 1 correspond to what we call pre-final concordance between annotators, i.e. results of a round of annotation carried out after the annotation procedure was

specified in its main lines, but before the *subsidiary criteria* of section 3.6 were introduced.

Figure 4 details the distribution of concordance scores across pieces. The median score is 95.8 % and 2 pieces are responsible for almost 4% absolute error rate.

The subsidiary criteria were added in a last stage to resolve most of the residual ambiguities and a consensual annotation was produced for 19 of the 20 pieces, while the 20th piece (#11 in Table 2) was considered as type 0 (i.e. impossibility to define reliable block boundaries).

01	Pink Floyd	Brain Damage
02	Queen	Lazing On A Sunday Afternoon
03	DJ Cam	Mad Blunted Jazz
04	Outkast	Return Of The G
05	ACDC	You Shook Me All Night Long
06	Eric Clapton	Old Love
07	Stan Getz & J. Gilberto	O Pato
08	Enya	Caribbean Blue
09	Mickael Jackson	Off The Wall
10	Bass America Collection	Planet
11	Plastikman	Fuk
12	Shack	Natalies Party
13	Sean Kingston	Take You There
14	Lil Mama	Shawty Get Loose
15	Abba	Waterloo
16	Eiffel 65	Blue (Da Ba Dee)
17	Meat Loaf	I'd Do Anything For You
18	Kaoma	Lambada
19	Vangelis	Conquest Of Paradise
20	Nirvana	Smells Like Teen Spirit

Table 2 : *List of music pieces used in the development set*

4.2 Dissemination

In a later phase, we annotated the "Pop" subset (100 songs) of the RWC database [10], with the goal to make the result available to the MIR community via MIREX, and on :

`http://musicdata.gforge.inria.fr`

Out of the 100 songs, 77 appear to be of type I, 20 of type II and 3 of type III (none of the other types). The average number of blocks per song is 15.54 (minimum 9, maximum 22). A vast majority (67 %) of blocks are regular, 15 % are derivations of regular blocks and 18 % are irregular. The average duration of regular blocks is 17.11 seconds. Table 3 details the distribution of sizes across blocks.

5. CONCLUSIONS

In this paper, we have specified and described thoroughly a consistent procedure for the description and the manual annotation of music structure, intended to be usable by non-expert human listeners, and which does not resort to absolute acoustic properties, nor to the analysis of the musical role of the constitutive elements. We hope that the proposed methodology will be experimented and refined by other groups and its usability widely established.

We are currently working on the definition of a procedure based on a multi-dimensional analysis, for assigning labels to the structural blocks, accounting for internal similarities and contrasts within the music piece.

Size	*Raw (A)*	*Stems (B)*	*Regular (C)*
4	1.6 %	1.6 %	0.0 %
8	10.5 %	9.7 %	3.6 %
12	2.6 %	2.4 %	1.1 %
14	1.1 %	0.1 %	0.0 %
16	59.2 %	72.3 %	87.3 %
18	5.6 %	0.1 %	0.0 %
20	4.9 %	1.4 %	1.4 %
24	1.8 %	1.3 %	1.5 %
32	2.2 %	2.7 %	3.6 %
Other	10.5 %	8.4 %	1.5 %

Table 3 : *Distribution of block sizes (in snaps) for raw blocks (A), stems (B) and regular blocks only (C). RWC – Pop database.*

6. REFERENCES

[1] Peeters, G. "Deriving Musical Structures from Signal Analysis for Music Audio Summary Generation: Sequence and State Approach", *in Lecture Notes in Computer Science*, Springer-Verlag, 2004.

[2] Abdallah, S. Noland, K., Sandler, M., Casey, M., and Rhodes, C. "Theory and evaluation of a Bayesian music structure extractor", *in Proc. ISMIR*, London, UK, 2005.

[3] Goto, M. "A Chorus Section Detection Method for Musical Audio Signals and Its Application to a Music Listening Station", *IEEE Trans. on Audio, Speech, and Language Processing*, 2006.

[4] Paulus, J. and Klapuri, A. "Music structure analysis by finding repeated parts", *in Proc. AMCMM*, Santa Barbara, California, USA, 2006.

[5] Peeters, G. "Sequence Representation of Music Structure Using Higher-Order Similarity Matrix and Maximum-Likelihood Approach", *in Proc. ISMIR*, Vienna, Austria, 2007.

[6] Levy, M., Sandler, M. "Structural Segmentation of musical audio by constrained clustering", *IEEE Transactions on Audio, Speech and Language Processing*, 2008.

[7] MIREX 2009 : http://www.music-ir.org/mirex/2009

[8] QUAERO Project : http://www.quaero.org

[9] Geoffroy Peeters and Emmanuel Deruty : Is Music Structure Annotation Multi-Dimensional ? A Proposal For Robust Local Music Annotation. LSAS, Graz (Austria) 2009.

[10] RWC : http://staff.aist.go.jp/m.goto/RWC-MDB

[11] F. de Saussure : Cours de Linguistique Générale. 1916.

[12] Fred Lerdahl & Ray Jackendoff : A Generative Theory of Tonal Music, MIT Press, 1983, reprinted 1996.

[13] Louis Hjelmslev : Essays in Linguistics (1959).

[14] Ian Bent with William Drabkin : Analysis. The New Grove Handbooks in Music. Macmillan Publishers Ltd, London, 1987, reprinted 1998. pp. 90-93.

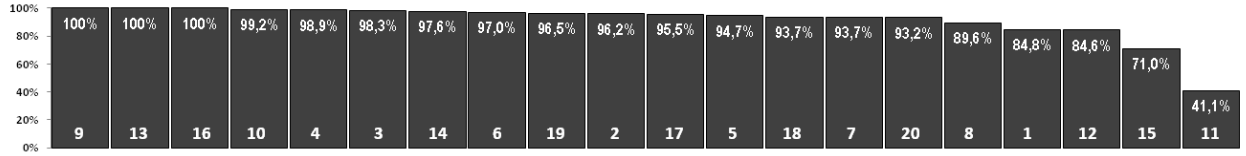


Figure 4 : *Inter-annotator concordance sorted in descending order for the 20 music pieces in the development set.*

DISCOVERING METADATA INCONSISTENCIES

Bruno Angeles
CIRMMT
Schulich School of Music
McGill University
bruno.angeles@mail.mcgill.ca

Cory McKay
CIRMMT
Schulich School of Music
McGill University
cory.mckay@mail.mcgill.ca

Ichiro Fujinaga
CIRMMT
Schulich School of Music
McGill University
ich@music.mcgill.ca

ABSTRACT

This paper describes the use of fingerprinting-based querying in identifying metadata inconsistencies in music libraries, as well as the updates to the jMusicMeta-Manager software in order to perform the analysis. Test results are presented for both the Codaich database and a generic library of unprocessed metadata. Statistics were computed in order to evaluate the differences between a manually-maintained library and an unprocessed collection when comparing metadata with values on a MusicBrainz server queried by fingerprinting.

1. INTRODUCTION

1.1 Purpose

Metadata is useful in organizing information, but in large collections of data it can be tedious to keep that information consistent. Whereas decision making in previous environments such as traditional libraries was limited to a small number of highly trained and meticulous people, the democratization of music brought about by the digital age poses new challenges in terms of metadata maintenance, as music can now be obtained from diverse and potentially noisy sources.

The contributors to many popular metadata repositories tend to be much less meticulous, and may have limited expertise. The research presented here proposes a combination of metadata management software, acoustic fingerprinting, and the querying of a metadata database in order to discover possible errors and inconsistencies in a local music library.

Metadata entries are compared between a library of manually-maintained files and a metadata repository, as well as between a collection of unprocessed metadata and the same repository, in order to highlight the possible differences between the two.

1.2 Metadata and its Value

Metadata is information about data. In music, it is information related to recordings and their electronic version (such as the performer, recording studio, or lyrics), although it can also be event information about artists or other attributes not immediately linked to a recorded piece of music. Corthaut et al. present 21 semantically related clusters of metadata [1], covering a wide range of information that illustrates the variety of metadata that can be found in music. Lai and Fujinaga [6] suggest more than 170 metadata elements organized into five types, in research pertaining to metadata for phonograph recordings. Casey et al. [2] distinguish factual from cultural metadata. The most widely used implementation of musical metadata is the ID3 format associated with MP3 files [5].

The main problem with metadata is its inconsistency. The fact that it is stored in databases containing thousands, if not millions, of entries often means that the data is supplied by several people who may have different approaches. Spelling mistakes may go unnoticed for a long time, and information such as an artist's name might be spelled in different but equally valid ways. Additionally, several metadata labels—most notably *genre*—are highly subjective.

When maintaining large databases of music, valid and consistent metadata facilitates the retrieval and classification of recordings, be it for Music Information Retrieval (MIR) purposes or simply for playback.

1.3 Metadata Repositories and Maintenance Software

Metadata repositories are databases that include information about recordings. When they are supported by an Application Programming Interface (API), they provide users with a convenient way of accessing the stored metadata. Existing music metadata repositories include MusicBrainz, Discogs, Last.fm, and Allmusic, to name just a few [3].

Several software solutions exist that provide ways to access, use, and adjust musical metadata. These include MusicBrainz Picard, MediaMonkey, jMusicMeta-Manager, and Mp3tag. The first three applications support fingerprinting in some form, but Mp3tag does not. GNAT [10] allows querying by metadata or fingerprinting to explore aggregated semantic Web data. jMusicMetaManager is the only application that performs extensive automated internal textual metadata error detection, and it also produces numerous useful summary statistics not made available by the alternatives.

1.4 Acoustic Fingerprinting

Acoustic fingerprinting is a procedure in which audio recordings are automatically analyzed and deterministically associated with a key that consumes considerably less space than the original recording. The purpose of using the key in our context is to retrieve metadata for a given recording using only the audio

information, which is more reliable than using the often highly noisy metadata packaged with recordings.

Among other attributes, fingerprinting algorithms are distinguished by their execution speed; their robustness to noise and to various types of filtering; and their transparency to encoding formats and associated compression schemes [1].

The fingerprinting service used in this paper is that of MusicIP (now known as AmpliFIND). It is based on Portable Unique Identifier (PUID) codes [9]. These are computed using the GenPUID piece of software. The PUID format was chosen for its association with the MusicBrainz API.

1.5 Method

This research uses jMusicMetaManager [7] (a Java application for maintaining metadata), Codaich [7] (a database of music with manually-maintained metadata), a reference library of music labeled with unprocessed metadata, and a local MusicBrainz server at McGill University's Music Technology Area. Reports were generated in jMusicMetaManager, an application for music metadata maintenance which was improved as part of this project by the addition of fingerprinting-based querying. This was done in order to find the percentage of metadata that was identical between the manually-maintained metadata and that found on the MusicBrainz server of metadata. In addition to comparing the *artist*, *album*, and *title* fields, a statistic was computed indicating how often all three of these specific fields matched between the local library and the metadata server, a statistic that we refer to as "identical metadata." Raimond et al. [10] present a similar method, with the ultimate objective of accessing information on the Semantic Web.

An unprocessed test collection, consisting of music files obtained from file sharing services, was used in order to provide a comparison between unmaintained and manually-maintained metadata. This unprocessed library is referred to as the "reference library" in this paper.

2. JMUSICMETAMANAGER

jMusicMetaManager [7] is a piece of software designed to automatically detect metadata inconsistencies and errors in musical collections, as well as generate descriptive profiling statistics about such collections. The software is part of the jMIR [8] music information retrieval software suite, which also includes audio, MIDI, and cultural feature extractors; metalearning machine learning software; and research datasets. jMusicMetaManager is, like all of the jMIR software, free, open-source, and designed to be easy to use.

One of the important problems that jMusicMetaManager deals with is the inconsistencies and redundancies caused by multiple spellings that are often found for entries that should be identical. For example, uncorrected occurrences of both "Lynyrd Skynyrd" and "Leonard Skinard" or of the multiple valid spellings of composers such as "Stravinsky" would be problematic for an artist identification system that would incorrectly perceive them as different artists.

At its simplest level, jMusicMetaManager calculates the Levenshtein (edit) distance between each pair of entries for a given field. A threshold is then used to determine whether two entries are likely to, in fact, correspond to the same true value. This threshold is dynamically weighted by the length of the strings. This is done separately once each for the *artist*, *composer*, *title*, *album*, and *genre* fields. In the case of titles, recording length is also considered, as two recordings might correctly have the same title but be performed entirely differently

This approach, while helpful, is too simplistic to detect the full range of problems that one finds in practice. Additional pre-processing was therefore implemented and additional post-modification distances were calculated. This was done in order to reduce the edit distance of strings that probably refer to the same thing, thus making it easier to detect the corresponding inconsistency. For example:

- Occurences of "The " were removed (e.g., "The Police" should match "Police").
- Occurrences of " and " were replaced with " & "
- Personal titles were converted to abbreviations (e.g., "Doctor John" to "Dr. John").
- Instances of "in'" were replaced with "ing" (e.g., "Breakin' Down" to "Breaking Down").
- Punctuation and brackets were removed (e.g., "R.E.M." to "REM").
- Track numbers from the beginnings of titles and extra spaces were removed.

In all, jMusicMetaManager can perform 23 pre-processing operations. Furthermore, an additional type of processing can be performed where word orders are rearranged (e.g., "Ella Fitzgerald" should match "Fitzgerald, Ella," and "Django Reinhardt & Stéphane Grappelli" should match "Stéphane Grappelli & Django Reinhardt"). Word subsets can also be considered (e.g., "Duke Ellington" might match "Duke Ellington & His Orchestra").

jMusicMetaManager also automatically generates a variety of HTML-formatted statistical reports about music collections, including multiple data summary views and analyses of co-occurrences between artists, composers, albums, and genres. This allows one to easily acquire and publish HTML collection profiles. A total of 39 different HTML reports can be automatically generated to help profile and publish musical datasets.

Users often need a graphical interface for viewing and editing a database's metadata. It was therefore decided to link jMusicMetaManager to the Apple iTunes software, which is not only free, well-designed, and commonly used, but also includes an easily parsed XML-based file format. iTunes, in addition, has the important advantage that it saves metadata modifications directly to the ID3 tags of MP3s as well as to its own files, which means that the recordings can easily be disassociated from iTunes if needed. iTunes can also access Gracenote's metadata automatically, which can then be cleaned with jMusicMetaManager.

jMusicMetaManager can extract metadata from iTunes XML files as well as directly from MP3 ID3 tags. Since Music Information Retrieval systems do not typically read these formats, jMusicMetaManager can also be used to generate ground-truth data formatted in ACE XML or Weka ARFF formats.

3. CODAICH

Codaich is a curated audio research dataset that is also part of jMIR [8]. It is constantly growing, and is now significantly larger than its original size of 20,849 recordings. The version used for the experiments described in this paper contains 32,328 recordings.

There is music by nearly 3,000 artists in Codaich, with 57 different musical genres represented. The dataset can be divided into four broad genres of Jazz, Popular, Classical, and (the somewhat problematic) World, henceforth referred to as "genre groups." These recordings are labeled with 19 metadata fields.

The metadata of the original version of Codaich was meticulously cleaned, both by hand and with jMusic-MetaManager. Care has been taken to maintain this very high level of metadata quality as the dataset has grown. The metadata for the original version of Codaich is available at the jMIR web site (http://jmir.sourceforge.net), and the metadata of the most recent version can be obtained in iTunes XML form by contacting the authors.

4. THE REFERENCE LIBRARY

In order to provide context, it was decided that a benchmark was needed against which the metadata consistency between Codaich and MusicBrainz could be compared. This was the motivation behind assembling the reference library, a combination of files downloaded from torrent-based networks and files that were obtained before the emergence of such systems. In the former case, files were downloaded as entire albums, while the rest of the reference library consists of recordings that were downloaded individually. The reference library consists of 1363 recordings by 446 artists, with 70 musical genres represented.

Since the reference library contained many music files with no ID3 metadata, but did hold some information in the files' names, metadata was created in such cases based on file names.

5. METHOD

5.1 Overview of the Experiments

Experiments were conducted to determine whether or not manually-maintained Codaich musical metadata showed a different level of consistency with MusicBrainz's information than the unprocessed reference library, for a fixed number of metadata fields.

The first step of the experiments consisted of obtaining PUID codes for each recording in each of the two libraries. The PUID information was stored in an XML file for later parsing by jMusicMetaManager.

In jMusicMetaManager, all the recordings in Codaich and the reference library were matched with entries in the XML file of PUID codes. PUID-based querying was performed on the MusicBrainz server, and a report of matching fields was generated for the chosen metadata fields.

Similar research done by Raimond et al. [10] presents GNAT, a Python application that supports PUID-based fingerprinting for track identification on a personal music library. The authors suggest accessing information pertinent to the user through the Semantic Web by querying, while we analyze the rate of consistency between the two datasets.

5.2 Changes to jMusicMetaManager

Running jMusicMetaManager on a large library of music files revealed that the application was not able to read the ID3 tags of files using releases 1, 2, 3, and 4 of the ID3v2 protocol. This was due to the choice of metadata API, *java_mp3*, used in jMusicMetamanager. Replacing *java_mp3* with the *jaudiotagger* API allowed us to read those formats.

Fingerprinting-based querying was added to jMusicMetaManager to enhance its capabilities. MusicBrainz's official (although no longer in active development) Java API was used (it is known as *libmusicbrainz-java*), since it allows querying by PUID, and a new corresponding report was added to jMusicMetaManager.

To expedite the querying process, threaded querying was implemented. This was applied to a copy of the MusicBrainz database hosted on a server at McGill University, something that was important in overcoming the one-query-per-second limitation of the public MusicBrainz server.

Genre group	Number of identified recordings	Identical artist	Identical album	Identical title	Identical artist, album, and title
Classical	1,476	3%	2%	6%	0%
Jazz	3,179	70%	25%	**64%**	12%
Popular	16,206	**84%**	**52%**	61%	**32%**
World	1,640	58%	29%	46%	11%

Table 1. Querying results for Codaich. Percentages represent the number of entries that were identical to those in MusicBrainz. The top results per statistic are identified in bold.

Genre group	Number of identified recordings	Identical artist	Identical album	Identical title	Identical artist, album, and title
Classical	285	17%	0%	5%	0%
Jazz	181	43%	14%	39%	4%
Popular	481	**79%**	**19%**	**51%**	**10%**
World	115	57%	12%	41%	3%

Table 2. Querying results for the reference library. Percentages represent the number of entries that were identical to those in MusicBrainz. The top results per statistic are identified in bold.

Genre group	Identical artist	Identical album	Identical title	Identical artist, album, and title
Classical	-14%	2%	1%	0%
Jazz	**27%**	11%	**25%**	8%
Popular	5%	**33%**	11%	**22%**
World	2%	17%	6%	9%

5.3 Reporting and Statistics

Not all files listed in the XML file of PUID codes were successfully identified by the MusicBrainz server (and, of course, MusicBrainz identification does not guarantee correctness). Several files list *unanalyzable* or *pending* as their status, while other extracted PUID codes did n return any result at all. Only identified recordings are used in this paper's statistics. The ratios of files that were not processed in each collection are specified in the following sections.

A case-insensitive string comparison was used in order to determine whether or not the *artist*, *album*, and *title* fields were identical on the MusicBrainz server and in the files' metadata.

6. RESULTS

Reports were generated for both Codaich and the reference library. The former database is maintained manually and is assumed to contain very few metadata errors and inconsistencies, while the latter contains many metadata problems due to the wide range of contributors and their varied interest and methods in maintaining metadata.

6.1 Codaich Results

Of the 32,328 songs in Codaich, 22,501 (70%) were identified on the MusicBrainz server using PUID values, 44 files were assigned a status of *unanalyzable* by GenPUID, and 84 were assigned the label *pending*. Of the remaining files, 9,645 (30%) had a PUID value but resulted in no hit on the MusicBrainz server.

Table 1 shows the metadata consistency between Codaich and MusicBrainz.

6.2 Reference Library Results

Of the 1,363 songs in the reference library, 1,062 (78%) were identified on the MusicBrainz server using PUID values. 5 files were assigned a status of *unanalyzable* by GenPUID, and 18 were assigned the label *pending*. Of the remaining files, 274 (20%) had a PUID value but resulted in no hit on the MusicBrainz server.

Table 2 shows the metadata consistency between the reference library and MusicBrainz.

6.3 Comparison of Codaich and the Reference Library

Table 3 illustrates the difference between the entries of Table 2 and Table 1. Positive values indicate a higher rate of matching metadata between MusicBrainz and Codaich

than between MusicBrainz and the reference library, while negative values mean the opposite. Although the first two tables are based on different libraries, the values of their difference provide us with a rough estimate of the quality difference between metadata in unprocessed music files collected from the internet and a curated library.

7. DISCUSSION

7.1 Global Observations

A comparison of Table 1 and Table 2 shows that the strongest agreement with MusicBrainz for the *artist* and *album* fields, as well as for the "identical metadata" statistic, is obtained for Popular music. This supports the argument that the main drivers of community-based metadata services are musical genres with which the most people are familiar, particularly among the techno-logically-savvy younger generations who may be more likely to contribute to metadata libraries.

With respect to titles, however, there is a greater level of MusicBrainz consistency in the manually-maintained library for Jazz recordings than for Popular recordings (albeit only 3% more, and the MusicBrainz title consistency is greatest for Popular music in the reference library). This may be due to better knowledge of Jazz on the part of Codaich's curator relative to the general public.

A potential cause of the relatively low percentages of Table 2 is the fact that part of the reference library consists of files that used ID3v1 tags instead of ID3v2 ones. ID3v1 tags limit the size of the *title*, *artist*, and *album* fields to 30 characters [5], which could cause mismatches in the case of longer entries of the MusicBrainz server compared to the limited ones of the ID3v1 tags.

7.2 Differences Between a Curated Database and an Unprocessed Collection

The largest changes between the two collections, as can be seen from Table 3, were obtained (in decreasing order) for the *album* field of Popular music (33%), the *artist* field of Jazz (27%), the *title* field of Jazz (25%), and all three chosen fields for Popular music (22%). In 14 out of 16 statistics, the MusicBrainz metadata matches the curated database's information more often than it does the unprocessed collection. We now focus on the two fields that were more MusicBrainz-consistent for the reference library than for Codaich. For Classical music, the consistency of the *artist* field for the reference library (17%) is much higher than that for Codaich (3%).

We were surprised to notice that the *artist* field results appeared to be "worsened" by the manual maintenance of metadata in this way. This can perhaps be explained by noting that Codaich fills the *artist* field with the name of the performer, while reserving composer names for the *composer* field. Most metadata on the MusicBrainz server and in the reference library, in contrast, lists the composer's name in the *artist* field, ignoring the *composer* field, possibly due to inherent limitations of the choice of underlying database schema.

The second statistic that is not higher in Table 1 than in Table 2 is "identical metadata" (artist, album, and title) for Classical music. In both cases, this statistic has a value of 0%, meaning that none of the Classical files considered had values matching the MusicBrainz entries for all three of these fields.

Indeed, the MusicBrainz consistency for Classical Music was very low, even for individual fields. Associating metadata with Classical music is challenging, as one must make decisions such as choosing how to convert names from the Cyrillic alphabet to Latin characters[1], choosing who the artist is, choosing whether to include long subtitles in album names, choosing whether to include the key and opus number in the title, etc. It is important to note that different ways of writing metadata may be perfectly valid in these situations, but multiple valid values can cause retrieval problems.

7.3 Classical/World Music vs. Jazz/Popular Music

Tables 1 and 2 allow us to distinguish two sets of genre groups, based on the frequency of matching metadata between the local files and the MusicBrainz server: Jazz and Popular music in one group, World and Classical music in the other.

Indeed, the highest matches are most often obtained for Jazz and Popular music, while the lowest results are in general obtained for Classical and World music. Popular and Classical music have different characteristics and use different fields [4], which leads to complications when applying a uniform metadata structure to different genres.

Classical music has by far the lowest MusicBrainz agreement in both music collections. World music has results between those of Classical music and Popular/Jazz music. The fact that Popular (and to a certain extent Jazz) music uses clearly-defined *artist*, *album*, and *title* tags facilitates the matching of such information on a web server.

7.4 Matching Results and Correct Results

Human maintenance of metadata has the expected effect of making metadata consistent across a library. Let us consider the case of Classical music. The low rate of consistency of matching artist names between Codaich and MusicBrainz might lead us to think that manual maintenance had a negative effect on the metadata, but in reality the Codaich metadata is arguably better than the

[1] Although this happens in other genres, it could be argued that conversion between languages is statistically more likely in Classical and World music.

MusicBrainz metadata. The ID3v2 protocols support a *composer* field, and should be used for that purpose, as done in Codaich.

It was interesting to observe in Codaich's file-by-file report of metadata comparisons that certain files that were assumed to belong to the same album were listed as belonging to different ones in the web service's fetched metadata. Consider the case of recordings that appear in different contexts such as movie soundtracks, compilations, and regular studio albums. Certain users may want to keep these recordings identified as being associated with the original studio release, while others will prefer to associate them with the other releases on which they appear, both points of view being valid. Such an issue would be avoided by allowing multi-value metadata fields.

8. CONCLUSIONS

Our results indicate that manually-maintained music files tend to have a greater level of metadata consistency with MusicBrainz than do unprocessed files. This does not, however, mean that the web service's metadata contains the correct values. We also noted that the matching rates of metadata vary across the analyzed genre groups. Differences in metadata between a manually-maintained database and a metadata database such as MusicBrainz may be due to a variety of reasons.

Combining jMusicMetaManager with fingerprinting querying can facilitate the cleanup of local collections of music. The matching of metadata between local files and a metadata server is particularly useful in the case of Popular music and Jazz, recent genres for which metadata fields are more easily attributed than for Classical and World music. With this new querying feature, jMusic-MetaManager is useful in more situations and unique in its reporting capability. We have also seen that, although at first sight the manual maintenance of metadata revealed some lowering of matches with the MusicBrainz server, it was justified by the proper use of attributes by the human curator. This can be seen as an illustration of the unreliability of collaboratively-maintained databases such as MusicBrainz for musical genres that do not benefit from the same public exposure as Popular music and Jazz.

In light of this potential unreliability, the use of metadata management software such as jMusicMeta-Manager is recommended in order to detect potential errors and inconsistencies in local libraries of music, as it can detect problems without reference to external sources of potentially unreliable information.

Having discussed the possibility that multiple values of a metadata field may all be valid in certain cases, we stress the need for multidimensional musical metadata.

Through our analysis of statistical results, we confirm the pertinence of the manual maintenance of metadata, and explain the reasons behind minor unexpected results.

jMusicMetaManager already computes metrics that can be used to detect metadata inconsistencies. As future work, integrating such features in the comparison of local and remote metadata would be helpful in that a threshold of comparison could allow the user to identify metadata that is similar enough. We expect an improvement in Classical music retrieval with such a change.

Finally, a similar experiment could be performed by manually correcting a given library of music while keeping the unprocessed version as reference.

9. REFERENCES

[1] Cano, P., E. Battle, T. Kalker, and J. Haitsma. 2002. A Review of Algorithms for Audio Fingerprinting. *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, 169–73.

[2] Casey, M., R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. 2008. Content-Based Music Information Retrieval. *Proceedings of the IEEE*. 668–96, 2008.

[3] Corthaut, N., S. Govaerts, K. Verbert, and E. Duval. 2008. Connecting the dots: Music metadata generation, schemas and applications," *Proceedings of the International Conference on Music Information Retrieval*. 249–54.

[4] Datta, D. 2002. Managing metadata, *Proceedings of the International Conference on Music Information Retrieval*. 249–51.

[5] ID3.org. 2003. Home: ID3.org. http://www.id3.org. Accessed 21 March 2010.

[6] Lai, C., and I. Fujinaga. 2006. Data dictionary: Metadata for phonograph records. *Proceedings of the International Conference on Music Information Retrieval.* 1–6.

[7] McKay, C., D. McEnnis, and I. Fujinaga. 2006. A large publicly accessible prototype audio database for music research," *Proceedings of the International Conference on Music Information Retrieval*. 160–3.

[8] McKay, C., and I. Fujinaga. 2009. jMIR: Tools for automatic music classification," *Proceedings of the International Computer Music Conference*. 65–8.

[9] MusicBrainz. 2009. How PUIDs work: MusicBrainz Wiki. http://wiki.musicbrainz.org/ HowPUIDsWork. Accessed 21 March 2010.

[10] Raimond, Y., C. Sutton, and M. Sandler. 2008. Automatic interlinking of music datasets on the semantic web. *Linked Data on the Web Workshop (LDOW2008).*

DISCOVERY OF CONTRAPUNTAL PATTERNS

Darrell Conklin
Department of Computer Science and AI
Universidad del País Vasco
San Sebastián, Spain
IKERBASQUE, Basque Foundation for Science
Bilbao, Spain
darrell_conklin@ehu.es

Mathieu Bergeron
CIRMMT
McGill University
Montreal, Canada
bergeron.mcgill@gmail.com

ABSTRACT

This paper develops and applies a new method for the discovery of polyphonic patterns. The method supports the representation of abstract relations that are formed between notes that overlap in time without being simultaneous. Such relations are central to understanding species counterpoint. The method consists of an application of the vertical viewpoint technique, which relies on a vertical slicing of the musical score. It is applied to two-voice contrapuntal textures extracted from the Bach chorale harmonizations. Results show that the new method is powerful enough to represent and discover distinctive modules of species counterpoint, including remarkably the suspension principle of fourth species counterpoint. In addition, by focusing on two voices in particular and setting them against all other possible voice pairs, the method can elicit patterns that illustrate well the unique treatment of the voices under investigation, e.g. the inner and outer voices. The results are promising and indicate that the method is suitable for computational musicology research.

1. INTRODUCTION

Polyphonic music presents challenges for music information retrieval, and the representation and discovery of patterns that recur in a polyphonic corpus remains an open and interesting problem. The discovery of monophonic patterns in music could be considered a mature area, with several powerful methods proposed and effectively applied to this task.

In polyphonic music, monophonic patterns can naturally be discovered using such methods provided that voice separation is first performed [12]. However, the discovery of polyphonic patterns – those containing two or more overlapping voices – is largely an open problem and there remain few approaches in the literature. The difficulty of this problem arises primarily from the presence of temporal relations between notes that occur in different voices, without being simultaneous. These relations are central to the understanding of counterpoint [9] and occur in all but the simplest note against note (first species) counterpoint. The discovery of polyphonic patterns is a new area with few existing approaches.

A difficulty with polyphony is that most polyphonic music cannot be easily separated into a regular number of voices [2]. In piano music, for example, voices can appear and disappear throughout a piece, and it would be mistaken to parse the music into a persistent number of voices. This paper uses the voiced Bach chorale harmonizations, and later some discussion is provided regarding the application of the method to unvoiced polyphony.

The simplest approach to find patterns in a polyphonic corpus is based on the conversion into chord symbol sequences, followed by the application of monophonic pattern discovery algorithms [1, 5]. However, accurate automatic chord labelling and segmentation is an unsolved problem, and the preparation of large corpora is difficult to achieve by hand.

Vertical approaches [4] are applied to voiced polyphonic textures first converted to a homophonic form. This is done by fully expanding a polyphonic score by adding artificial notes whenever a new onset time is encountered. This expanded score is then sliced, features computed for each slice in the sequence, and the resulting sequence mined using a monophonic pattern discovery approach. A similar method using bit-string approaches has been described by Chiu et al. [3]. The problem with these vertical approaches is in the lack of flexible temporal relations between components of a slice, which must all have the same onset time.

Geometric approaches [10] can be applied to voiced or unvoiced polyphony and can reveal patterns not contiguous on the music surface. The geometric approach however has two drawbacks for discovering contrapuntal patterns. Since geometric patterns must conserve exact inter-onset intervals in the time dimension, they cannot represent general local temporal relations among pattern components. Furthermore the expensive time complexity of the method leads to intractability for large pieces or corpora.

Polyphonic patterns are ideally represented as relational networks. Nodes represent events and edges represent relations between events. For example, consider the score

fragment and small relational network presented in Figure 1(i,ii). This pattern matches four notes (a, b, c, d) in the following relations: a m7 dissonance (a, c) formed by the bottom voice (c) approached by a downwards leap, resolving to a M3 consonance (b, d) by a stepwise motion down in the top voice (b), and a leap up in the bottom voice (d). In addition, an asterisk $*$ indicates that (c) is preceded by a note that the pattern does not represent explicitly, but rather implicitly via the notion of melodic motion. Note that while (a, b) and (c, d) follow each other in the same voice, critically important in this pattern are the different temporal relations between (a, c) (c starting while a is sounding) and (b, d) (they start together).

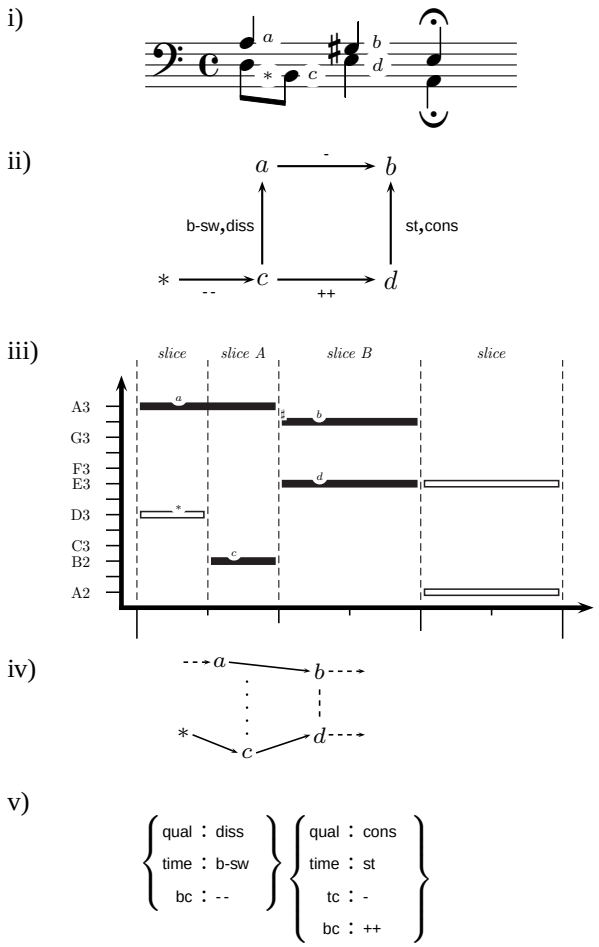


Figure 1. Two-voice counterpoint pattern: i) instance in final bar of BWV 257; ii) relational pattern; iii) piano-roll notation of instance showing the slicing process; iv) a schematic representation of the pattern; v) an equivalent feature set pattern.

Figure 1(iv) shows a schematic representation that we have developed as a visually appealing notation for a full relational network over a limited number of relations. Horizontal lines indicate voicing. The placement of vertical lines indicates temporal relations, and their form indicates consonance (dashed) or dissonance (dotted) between the temporally related notes. The slope of horizontal lines is proportional to the melodic contour it indicates (i.e. a steeper slope indicates a melodic leap). When the contour is not defined by the pattern, the horizontal line is dashed. Again, an asterisk $*$ indicates a note that must implicitly be present for the pattern to match, but is not explicitly matched by a pattern component.

This paper is concerned with discovering patterns like the one presented in Figure 1 in a large corpus of polyphonic music. We consider a precise notion of polyphonic pattern, one that captures musical intervals between notes that overlap in time, including between notes that are not simultaneous. Our approach is computationally very efficient as it translates the relational discovery problem to a form where sequential pattern discovery can be applied.

The paper is structured as follows. First, the new $\mathcal{VVP}$ (Vertical ViewPoint) method for polyphonic pattern discovery will be described. Results will be highlighted and translated to schematic relational networks for clarity. Finally, the strengths and weaknesses of $\mathcal{VVP}$ are discussed.

2. METHOD

The vertical viewpoint approach as originally presented [4] applies to homophony and cannot express temporal relations that might hold between components of a vertical slice, as would be required for the pattern developed in Figure 1. However, the approach can be extended resulting in what we call the $\mathcal{VVP}$ formalism. This is done by a) adding the ability to handle local temporal relations within slices; and b) describing properties of slices using flexible sets of features.

The first step was to employ event *continuations*, which specially identify those events with onsets prior to the onset of the slice. This method was inspired by the approach of Dubnov et al. [7] who developed a real-time method for prediction of continuations for polyphonic improvisations. Figure 1(iii) illustrates slices where continuations are used to retain local temporal relations. For example, the slice labelled A contains two notes: B2 in the bottom voice (c) and a continued event A3 in the top voice (a).

The second step was to employ *feature sets* [6] to describe successive slices in a piece. In this paper, four features are used to describe slices:

time describing the temporal relations between the two voices, taking the values st (both voices start together), b-sw (bottom voice starts while the top voice is sounding) and t-sw (top voice starts while the bottom voice is sounding). These temporal relations are derived by inspecting the continuation features of events within slices;

qual of the harmonic interval (non-compound diatonic interval between lower and higher pitches in the slice), taking the values cons (intervals P1, m3, M3, P5, m6, M6) and diss (all other intervals);

bc, tc describing the melodic contour of either the top voice (tc) or bottom voice (bc), both taking the possible values leap up: ++; leap down: --; step up: +; step down: -; unison: =.

Figure 1(v) illustrates a $\mathcal{VVP}$ pattern, using the above features. The pattern is a sequence of two features sets which expresses exactly the relational network in Figure 1(ii). It matches the score fragment in Figure 1(i) and in general all pairs of contiguous slices that contain the features presented.

To apply this representation in pattern discovery, a corpus is sliced at every unique onset time, every slice is saturated with the above features, then a pattern discovery algorithm finds all patterns that are distinctive (with a score above a threshold) and maximally general [5]. The algorithm for finding these patterns is a depth-first search of a subsumption space, that iteratively refines patterns at each search node to make them more specific. This is described in more detail in [5].

Regarding the scoring of patterns, it is well-known that discovered patterns should not be ranked simply by their count, because a general pattern will naturally occur more frequently than a more specific one, regardless of the corpus. Patterns are scored by an odds ratio, dividing their corpus probability by their background probability:

$$\Delta(P) \stackrel{\text{def}}{=} \frac{p(P|\oplus)}{p(P|\ominus)}, \quad (1)$$

where P is a pattern and the probabilities $p(P|\oplus)$ and $p(P|\ominus)$ are computed as follows. The probability of pattern P in the corpus is $p(P|\oplus) = c^{\oplus}(P)/n^{\oplus}$, where $c^{\oplus}(P)$ is the total slice count of pattern P in the corpus and $n^{\oplus}$ is the number of slices in the corpus. To evaluate the background probability $p(P|\ominus)$, two distinct methods may be used: the *anticorpus* method, and the the *null model* method.

An *anticorpus* is a set of pieces that is set up specifically to contrast with the analysis corpus. When an explicit anticorpus is used, the background probability is simply the anticorpus probability of the pattern:

$$p(P|\ominus) \stackrel{\text{def}}{=} c^{\ominus}(P)/n^{\ominus}, \quad (2)$$

where $c^{\ominus}(P)$ is the total slice count of the pattern P in the anticorpus, and $n^{\ominus}$ is the number of slices in the anticorpus.

In cases where an explicit anticorpus is not used, the *null model* method may be employed instead to rank patterns. The null model probability of a pattern P comprised of feature sets $c_1, \ldots, c_n$ is defined as:

$$p(P|\ominus) \stackrel{\text{def}}{=} \prod_{i=1}^{n} \prod_{f \in c_i} c^{\oplus}(f)/n^{\oplus}, \quad (3)$$

where $c^{\oplus}(f)$ is the total count (number of slices) of the feature f in the corpus. It estimates how probable a pattern is by multiplying the probabilities of its constituent features as they are encountered in the corpus. For example, the b-sw and t-sw temporal relations might occur less frequently than the st temporal relation. The null model probability of a pattern containing either b-sw or t-sw would then be lower, allowing for such a pattern to be interesting even if it has a lower total count than a similar pattern with a st temporal relation.

3. RESULTS

This section presents the results of $\mathcal{VVP}$ on pairs of voices extracted from 185 J.S. Bach chorale harmonizations in Humdrum format. Importantly, this format has diatonic spelling for all pitches, facilitating the computation of consonance and dissonance relations. Voices were extracted and recombined using the Humdrum toolset, and the appropriate filtering applied to insure well-formed slices (e.g. removing null lines to avoid slices where no event starts). All 6 possible ordered voice pairs were extracted from each piece: soprano-alto (SA), soprano-tenor (ST), soprano-bass (SB), alto-tenor (AT), alto-bass (AB), and tenor-bass (TB). This provides a total of $185 \times 6 = 1110$ possible two-voice pairs.

Two experiments were performed: one where an anticorpus was used to reveal the most interesting patterns and one where a null model was used. In both experiments, discovered pattern sets were filtered to retain only those patterns that contained at least a temporal (time) and harmonic interval quality feature (qual) in all components of a pattern. The method was calibrated to discover patterns with a score of at least 3 (Equation 1): with 3 times higher probability in the corpus as compared to the anticorpus or null model background probabilities (this calibration has worked well for other pattern discovery experiments in music [5]). In addition, only patterns with a total count of at least 100 were considered. This is a simple way to isolate patterns that are deemed to be reasonably frequent (the suspension pattern reported in Section 3.2, for example, has a total count of 450).

3.1 Anticorpus

In this experiment, all 6 voice pairs (SA, ST, SB, AT, AB, TB) were in turn used as the corpus, with the remaining 5 as the anticorpus. Thus each corpus comprises 185 two-voice polyphonic extracts, and each anticorpus $185 \times 5 = 925$ extracts. Patterns are ranked by using the anticorpus method (Equation 2) to compute background probabilities. Results for all experiments are concatenated, sorted from high to low $\Delta(P)$ (Equation 1), and the most distinctive 3 patterns each from the AT and SB corpora were retained (Table 1). The inner voices (AT) and outer voices (SB) were selected for illustration as they were previously discussed in the literature [8], but the method returns results for every voice pair. Below each pattern, Table 1 shows its total count and score as a number pair. The three patterns in each group are sorted by decreasing score $\Delta(P)$: note how a pattern can have higher score despite having a lower total count.

The first pattern of Table 1 is roughly 5 times more likely to occur between SB (outer voices) than any other voice pair. As mentioned by Huron [8], outer voices are more perceptually salient. They are hence expected to exhibit a good contrapuntal quality. Not surprinsingly, the patterns discovered by $\mathcal{VVP}$ as characteristic of SB are all consistent with counterpoint. For example, the first and third patterns introduce a dissonance over a b-sw temporal

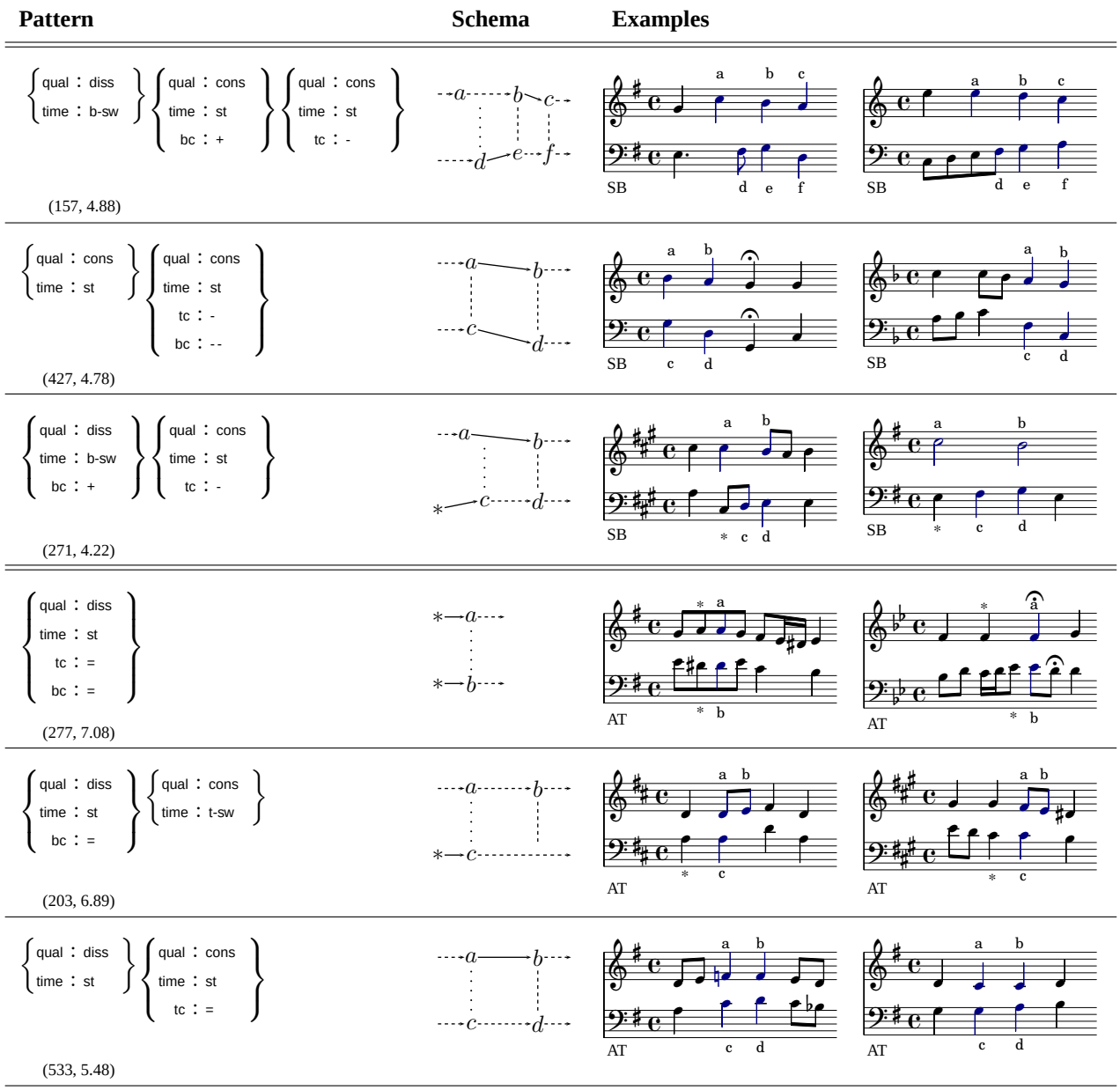


Table 1. Patterns discovered by computing background probabilities using an anticorpus. Top: distinctive of SB; bottom: distinctive of AT.

relation on a weak beat and resolve it straight away through stepwise melodic motion.

By opposition, one can expect the inner voices (AT), which are less perceptually salient, to be used more freely to create harmonies, with less considerations for the presence of dissonance. The last three patterns of Table 1 all characterize the inner voices and are consistent with this idea. All patterns introduce a dissonance over a st temporal relation. Pattern 5 clearly demonstrates this: a dissonances of a perfect fourth, usually forbidden between outer voices, here freely occurs in both examples.

In addition to the results presented in Table 1, many of the most interesting patterns discovered by $\mathcal{VVP}$ refer to either SB or AT pairs (data not shown). This suggests that the inner and outer voices are the two voice pairs that are the easiest to characterize, possibly because they differ the most from other voice pairs in terms of their contrapuntal quality.

3.2 Null model

In this experiment, we consider all voice pairs together (a corpus of 1110 two-voice pairs). Patterns are ranked by using a null model (Equation 3) to compute background probabilities. The seven most distinctive patterns are presented in Table 2. Again, the total count and score of each pattern is shown as a number pair.

The first pattern exhibits consecutive leaps. On its own, a leap is less likely than a step. Similarly, a sequence of leaps will be deemed unlikely by the null model. However, the first pattern contains such a sequence and does occur significantly in the chorales: 11 times more than ex-

pected. This can be interpreted as a composition rule: a sequence of leaps is acceptable, given that it occurs over two consonances, over two b-sw temporal relations, and changes contour. Note how the second instance of pattern 1 presents a rhythmic augmentation: the abstract pattern makes no constraints on rhythmic features.

The second, fourth and fifth pattern evoke second species counterpoint, while the sixth evokes first species. Pattern 3 evokes third species counterpoint, with a four note against one texture, alternating with dissonance and consonance. In one instance the bass line is rising: in the other it falls. In pattern 2, similarly, the direction of motion of the tenor line is inverted between the two instances shown. Again, this illustrates the abstraction power of the feature set pattern representation.

Pattern 7 is characteristic of the fourth species, with voices overlapping via successive b-sw and t-sw temporal relations. Remarkably, this is a pattern precisely describing the suspension pattern of fourth species counterpoint, including the melodic contour of a step down for the resolution of the dissonance [9]. The two instances presented are examples of the 4-3 and 7-6 suspension. Note how the high level of abstraction of this pattern (refraining from specifying exact intervals, and lower voice movement for the suspension resolution) is necessary to represent the concept of a suspension.

4. DISCUSSION

This paper presented the $\mathcal{VVP}$ method for the discovery of relational patterns in two-voice counterpoint. The method is based on a monophonic pattern discovery algorithm [5], and extends earlier results [4] through the use of continuations of events across slices. The $\mathcal{VVP}$ approach is fast as it transforms a relational data mining problem into a simpler sequential one, an example of representation change often employed in machine learning. Nevertheless, the representation is flexible due to the abstraction of slices by feature sets.

In the experiments presented here, the representation allows the concise expression of many contrapuntal patterns. This demonstrates that the approach is powerful enough to discover polyphonic patterns of theoretical significance. The patterns reported here are not particularly surprising, but their discovery is nonetheless promising: further exploration with the method could help discover significant patterns that are yet unknown to music theory. This exploration is however outside the scope of the current paper.

Similar patterns to the those presented here have been studied in the context of supervised learning [11], where the patterns were identified beforehand and the task was to learn rules explaining the structure of the patterns. However, only simple note against note counterpoint was studied.

The application of the method in this paper led to patterns that were very abstract, with highly divergent instances on the musical surface. It was shown, however, that such abstraction is necessary to capture the concept of a fourth species suspension pattern in its full generality. Furthermore, using only three simple background features (temporal relations, harmonic interval quality, and melodic contour), the method was able to discover the suspension in a corpus of Bach chorale harmonizations.

If less abstraction is desired, more background features can be added, for example those referring to melodic interval and durations of notes. For this study, a calibration of the method using the most basic relations of counterpoint was desired. This has been successful and future work will focus on further pattern representation aspects.

Though applied to two-voice textures here, the method easily extends to more voices. Further temporal relations would need to be added to accommodate the additional temporal relations formed; for example for pattern discovery in 3 voices (e.g., SAT), three overlap relations (SA, ST, AT) and three corresponding simultaneity relations would be needed: in general a number of such features quadratic in the number of notes in a slice. In addition, three different harmonic interval relations would be necessary, or alternatively the notion of harmonic interval quality could somehow be extended to triads.

A limitation of this approach, as presented here, is that it applies to voiced polyphony, or corpora where voices have already been extracted from an unvoiced polyphonic texture. There are several possible ways to apply the method to raw unvoiced textures. Temporal relations between slice components may simply be discarded, and any "static" viewpoint (such as chord type, mode) of a slice may be used. Alternatively, the presence of a temporal relation between slice components can be considered but with voice leading relations (e.g., melodic contour) omitted. Finally, if both temporal and voice-leading relations are desired, one might replace the notions of top voice and bottom voice with the highest note and lowest note in the slice, reminiscent of the simple skyline approach to voice segregation. With feature sets, one could even include melodic relationships between second highest pitches (if present), third highest pitches (if present), lowest (possible bass line) and so on. The presence or absence of such features would also indicate texture density, hence potentially expressing patterns occurring over changes of textures.

5. ACKNOWLEDGEMENTS

Darrell Conklin is supported by IKERBASQUE, Basque Foundation for Science, Bilbao, Spain. Mathieu Bergeron is supported by Le Fonds québécois de la recherche sur la nature et les technologies, Québec, Canada.

6. REFERENCES

[1] A. Anglade and S. Dixon. Characterisation of harmony with inductive logic programming. In *International Conference on Music Information Retrieval (ISMIR)*, pages 63–68, Philadelphia, USA, 2008.

[2] E. Cambouropoulos. Voice and stream: Perceptual and computational modeling of voice separation. *Music Perception*, 26(1):75–94, 2008.

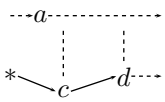
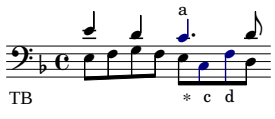
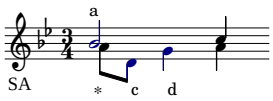
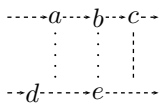
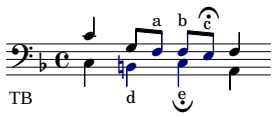
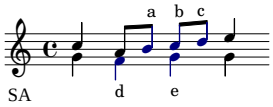
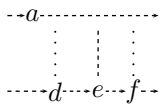
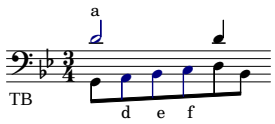
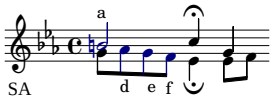
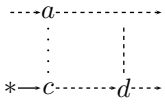
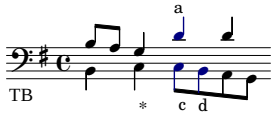
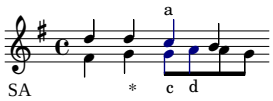
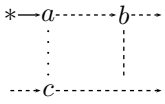
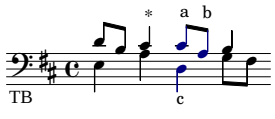
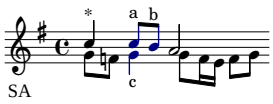
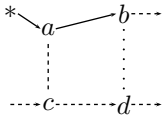
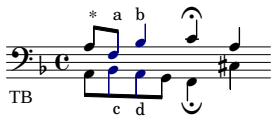
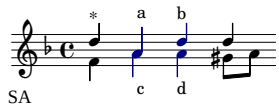
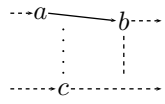
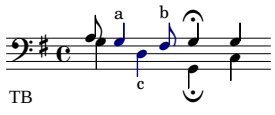
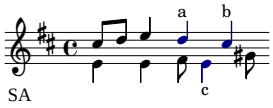
Pattern	Schema	Examples
{qual : cons, time : b-sw, bc : --} {qual : cons, time : b-sw, bc : ++} (152, 11.69)		 
{qual : diss, time : t-sw} {qual : diss, time : st} {qual : cons, time : t-sw} (158, 11.68)		 
{qual : diss, time : b-sw} {qual : cons, time : b-sw} {qual : diss, time : b-sw} (154, 10.11)		 
{qual : diss, time : st, bc : =} {qual : cons, time : b-sw} (1128, 9.69)		 
{qual : diss, time : st, tc : =} {qual : cons, time : t-sw} (804, 9.00)		 
{qual : cons, time : st, tc : --} {qual : diss, time : st, tc : ++} (116, 8.89)		 
{qual : diss, time : b-sw} {qual : cons, time : t-sw, tc : -} (450, 7.73)		 

Table 2. Patterns discovered by computing background probabilities using a null model.

[3] S-C. Chiu, M-K. Shan, J-L. Huang, and H-F. Li. Mining polyphonic repeating patterns from music data using bit-string based approaches. In *ICME 2009*, pages 1170–1173, 2009.

[4] D. Conklin. Representation and discovery of vertical patterns in music. In *Music and Artificial Intelligence: Lecture Notes in Artificial Intelligence*, number 2445, pages 32–42. Springer-Verlag, 2002.

[5] D. Conklin. Discovery of distinctive patterns in music. *Intelligent Data Analysis*, 14(5), 2010. To appear.

[6] D. Conklin and M. Bergeron. Representation and discovery of feature set patterns in music. *Computer Music Journal*, 32(1):60–70, 2008.

[7] S. Dubnov, G. Assayag, O. Lartillot, and G. Bejerano. Using machine-learning methods for musical style modeling. *IEEE Computer*, 36(10):73–80, 2003.

[8] D. Huron. Voice denumerability in polyphonic music of homogeneous timbres. *Music Perception*, 6(4):361–382, 1989.

[9] K. Kennan. *Counterpoint*. Prentice-Hall, 1959.

[10] D. Meredith, K. Lemström, and G. A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4):321–345, 2002.

[11] U. Pompe, I. Kononenko, and T. Makše. An application of ILP in a musical database: Learning to compose the two-voice counterpoint. In *Proceedings of the MLnet Familiarization Workshop on Data Mining with Inductive Logic Programing*, pages 1–11, Heraklion, Crete, 1996.

[12] P. E. Utgoff and P. B. Kirlin. Detecting motives and recurring patterns in polyphonic music. In *ICMC*, pages 487–494, New Orleans, Lousianna, USA., 2006.

EIGENVECTOR-BASED RELATIONAL MOTIF DISCOVERY

Alberto Pinto
Università degli Studi di Milano
Dipartimento di Informatica e Comunicazione
Via Comelico 39/41, I-20135 Milano, Italy
pinto@dico.unimi.it

ABSTRACT

The development of novel analytical tools to investigate the structure of music works is central in current music information retrieval research. In particular, music summarization aims at finding the most representative parts of a music piece (motifs) that can be exploited for an efficient music database indexing system. Here we present a novel approach for motif discovery in music pieces based on an eigenvector method. Scores are segmented into a network of bars and then ranked depending on their centrality. Bars with higher centrality are more likely to be relevant for music summarization. Results on the corpus of J.S.Bach's 2-part Inventions demonstrate the effectiveness of the method and suggest that different musical metrics might be more suitable than others for different applications.

1. INTRODUCTION

Listening to music and perceiving its structure is a relatively easy task for humans, even for listeners without formal musical training. However, building computational models to simulate this process is a hard problem. On the other hand, the problem of automatically identifying relevant characteristic motifs and efficiently store and retrieve the digital content has become an important issue as digital collections are increasing in number and size more or less everywhere.

Notwithstanding the conspicuousness of the literature, current approaches seem to rely just on the *repetition* paradigm [20] [8], assigning higher scores to recurring equivalent melodic and harmonic patterns [11]. Recently reported approaches to melodic clustering based on string compression [10], motivic topologies [18], graph distance [21] and paradigmatic analysis [19] have been used to select relevant subsequences among highly repeated ones by heuristic criteria [15] [1]. However, this approach is not completely satisfying as the repetition paradigm can provide just a first approximation of the perceptual ranking mechanism [3] and produces too many false positives sharing the same repetition rates.

Moreover, the repetition paradigm, in order to be applied, needs by no means a precise definition of *"varied repetition"*, a concept not easy to define. Of course, it has to include standard music transformation, but it is very difficult to adopt a simple two-valued logic (this is a repetition and this is not) in this context, where a more fuzzy approach seems to better address such a problem.

Sometime repetitions may even lead to evident mistakes, as it might happen that highly repeated patterns turn to be totally irrelevant from a musicological point of view. In fact cases occur where the most repeated pattern in the whole composition is an ornament, like a trill. This is to show that the repetition paradigm is not sufficient in itself to identify relevant themes but it needs some heuristics to select among relevant and irrelevant patterns.

Here we present an alternative ranking method based on connections instead of repetitions. We show that a distance distribution on a graph of note subsequences induced by music similarity measures generates a ranking real eigenvector whose components reflect the actual relevance of motives. False positives of the repetition paradigm turned out to be less connected nodes of the graph due to their higher degree of dissimilarity with relevant motives.

Our results show how higher indexes of connection, or "centrality", are more likely to perform better than higher repetition rates in motif discovery, with no additional assumptions on the particular nature of the sequence or the adopted similarity measure.

2. RELATED WORKS

Music segmentation is usually realized through musicological analysis by human experts and, at the moment, automatic segmentation is a difficult task without human intervention. The supposed music themes have often to undergo a hand-made musicological evaluation, aimed at recognizing their expected relevance and completeness of results. As a matter of fact, an automatic process could extract a musical theme which is too long, or too short, or simply irrelevant. Thats why a human feedback is still required in order to obtain high-quality results.

We present here an overview of current approaches based on different musical assumptions. We start this section with a general overview of the literature. Then we intro-

duce harmony related approaches, with a focus to reductionistic ones. Finally we introduce topology-based models, which share much more similarities than others with our approach. All those methods makes use of the repetition paradigm.

2.1 General approaches

Lartillot [15] [16] defined a musical pattern discovery system motivated by human listening strategies. Pitch intervals are used together with duration ratios to recognize identical or similar note pairs, which in turn are combined to construct similar patterns. Pattern selection is guided by paradigmatic aspects and overlaps of segments are allowed.

Cambouropoulos [6], on the other hand, proposed methods to divide given musical pieces into mostly non-overlapping segments. A prominence value is calculated for each melody based on the number of exact occurrences of non-overlapping melodies. Prominence values of melodies are used to determine the boundaries of the segments [7]. He also developed methods to recognize variations of filling and thinning (through note insertion and deletion) into the original melody. Cambouropoulos and Widmer [9] proposed methods to construct melodic clusters depending on the melodic and rhythmic features of the given segments. Basically, similarities of these features up to a particular threshold are used to determine the clusters. High computational costs of this method make applications to long pieces difficult.

2.2 Tonal harmony-based approaches

Tonal harmony based approaches exploit particular harmonic patterns (such as tonic-subdominant-dominant-tonic), melodic movements (e.g. sensible-tonic), and some rhythmical punctuation features (pauses, long-duration notes, ...) for a definition of a commonly accepted semantic in many ages and cultures.

These approaches typically lead towards score reductions (see Figure 1), made possible by taking advantage of additional musicological information related to the piece and assigning different level of relevance to the notes of a melody. For example one may choose to assign higher importance to the stressed notes inside a bar [22]. In other words, the goal of comparing two melodic sequences is achieved by reducing musical information into some "primitive types" and comparing the reduced fragments by means of suitable metrics.

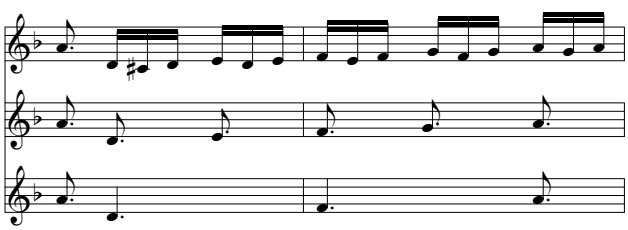


Figure 1. J.S. Bach, BWV 1080: Score reductions.

A very interesting reductionistic approach to music analysis has been attempted by Fred Lerdahl and Ray Jackendoff. Lerdahl and Jakendoff [17] research was oriented towards a formal description of the musical intuitions of a listener who is experienced in a musical idiom. Their purpose was the development of a formal grammar which could be used to analyze any tonal composition.

The study of these mechanisms allows the construction of a grammar able to describe the fundamental rules followed by human mind in the recognition of the underlying structures of a musical piece.

2.3 Topological approaches

Mazzola and Buteau [5] proposed a general theoretical framework for the paradigmatic analysis of the melodic structures. The main idea is that a paradigmatic approach can be turned into a topological approach. They consider not only consecutive tone sequences, but allow any subset of the ambient melody to carry a melodic shape (such as rigid shape, diastematic shape, etc.). The mathematical construction is very complex and, as for the motif selection process, it relies on the repetition paradigm.

The method proposed by Adiloglu, Noll and Obermayer in [1] does not take into account the harmonic structure of a piece and is based just on similarities of melodies and on the concept of similarity neighborhood. Melodies are considered as pure pitch sequences, excluding rests and rhythmical information.

A monophonic piece is considered to be a single melody M, i.e. they reduce the piece to its melodic surface. Similarly, a polyphonic piece is considered to be the list $M = (M_i)_{i=1,\ldots,N}$ of its voices M_i. The next step is to model a number of different melodic transformations, such as transpositions, inversions and retrogradations and to provide an effective similarity measure based on cross-correlation between melodic fragments that takes into account these transformations. They utilize a mathematical distance measure to recognize melodic similarity and the equivalence classes that makes use of the concept of *neighbourhood* to define a set of similar melodies.

Following the repetition paradigm stated by Cambouropoulos in [7] they define a prominence value to each melody based on the number of occurrences, and on the length of the melody. The only difference is that they allow also melody overlapping. In the end, the significance of a melody m of length n within a given piece M is the normalized cardinality of the similarity neighbourhood set of the given melody. If two melodies appear equal number of times, the longer melody is more significant than the shorter one.

In [1] the complete collection of the Two-part Inventions by J. S. Bach is used to evaluate the method, and this will be also our choice in section 4.

3. THE RELATIONAL MODEL

As stated in Section 2, current methods rely on the repetition paradigm. Our point of view can be synthesized in the following points:

1. We consider a music piece as a network graph of segments,

2. we do take into account both melodic and rhythmical structures of segments

3. we do not consider harmony, as it is too much related to tonality.

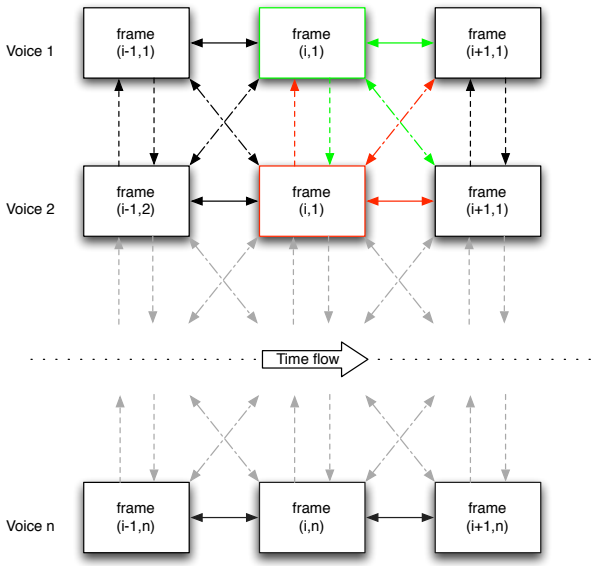


Figure 2. A representation of the (first-order) network of frames.

A single frame may represent, for instance, a bar or a specific voice within a bar like in Fig. 2, but also more general segments of the piece. We do not take into account here the problem of windowing as the method is basically independent from any specific segmentation of the piece. What we provide here is a different point of view which, like the repetition paradigm, can be applied in principle to any specific segmentation.

3.1 Score representation

The natural consequence is that a music piece can be looked at like a complete graph K_n. In graph theory, a complete graph is a simple graph where an edge connects every pair of distinct vertices. The complete graph on n vertices has $n(n-1)/2$ edges and is a regular graph of degree $n-1$.

In this representation, score segments correspond to graph nodes and the similarity between couples of segments correspond to edge weights. This approach can be better explained if we think to a score like a network graph of "pages", so we can establish a parallelism between score segments ranking and the World Wide Web ranking process as originally depicted in [4] by S. Brin and L. Page.

As stated before, the problem of windowing is partly overcome in the network concept as it does not strongly affect the model. In fact by using undersized windows we normally get just more detailed results. In our experiments (see Section 4) we decided to adopt a one-bar length window, as we considered metric information relevant to music segmentation, avoiding any form of overlapping. In fact it turned out that if the metric information is taken into account, overlapping windows are not relevant in a relational model, as they can lead to inaccurate motif discovery, due to overrates given to highly self-similar segments.

3.2 Metric weights

As stated above, graph nodes correspond to score segments. The next issue is the definition of a suitable concept of distance between segments. This should be apparently at the very heart of the method, and in a sense it is. Every time there is a similarity concept the question is: *which kind of similarity?* There are so many different concepts of music similarity (perceptual, structural, melodic, rhythmical, and so on) that is not possible to provide a unique definition.

The variety of segmentations reflects to a large extent the variety of musical similarity concepts, and that is the reason why it is correct to have this parameter here. Nevertheless, as stated in Section 4 the model is rather robust respect to metric changes.

In general, we can just say that the set of segments can be endowed with a notion of distance

$$d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$$

between pairs of segments and turns this set into a (possibly metric) space $(\mathcal{S}, d)$. A natural choice for point sets of a metric space is the Hausdorff metric [13] but any other distance discovered to be useful in music perception, like EMD/PTD [23], can be chosen as well.

Here we assume d to be:

1. real,

2. non-negative,

3. symmetric and

4. such that $d(s,s) = 0, \forall s \in \mathcal{S}$

As a matter of fact, most musically relevant perceptual distances do not satisfy all metric axioms [23]. Therefore no further property, like the identity of indiscernibles or the triangle inequality, is assumed.

Given two segments s_1 and s_2, for the experiments we adopted the two following simple metrics:

$$d_1(s_1, s_2) = \sqrt{\sum_{|s|} |[s_1]_{12} - [s_2]_{12}|} \qquad (1)$$

$$d_2(s_1, s_2) = \sqrt{\sum_{|s|} (s_1' - s_2')^2} \qquad (2)$$

where s' is the derivative operator on the sequence s, $|s|$ is the length of s and $[s]_{12}$ is the sequence s where each entry has been chosen in the interval $[0, 11]$.

d_1 is a first-order metric that takes into account just octave transpositions of melodies. In fact, pitch classes out of the range $[0, 11]$ are folded back into the same interval,

so melodies which differ for one or more octaves belong to the same congruence class modulo 12 semitones. d_2 is a second-order metric that takes into account arbitrary transpositions and inversions of a melody. No other assumptions on possible variations have been made, so that an equivalence class of melodies is composed just of transpositions and inversions of the same melody like in [1].

Both distances can be applied to single voice sequences but also to multiple voice sequences, given that a suitable representation has been provided. For instance, in a two voice piece, with voices v_1 and v_2, one can consider the difference vector $v = v_1 - v_2$ as a good representation of a specific segment, and then apply d_1 or d_2 to this new object. The advantage of using this differential representation is that it is invariant respect to transpositions and inversions of the two voices so that, for instance, it makes also d_1 invariant respect to transpositions and inversions, and not just to octave shifts.

By exploiting those distance concepts, it is possible to endow the edges of the complete graph with metric weights in order to compute the weights of nodes in terms of the main eigenvector, as we are going to show in the following Sections.

3.3 Matrix representation and ranking eigenvector

The adopted algebraic representation of the 'score graph' K is the adjacency matrix $A(K)$. This is a nonnegative matrix as its entries are the distance values between the different segments in which the score has been divided into. Perron-Frobenius theory for nonnegative matrices grants that, if $A \in M_{n\times n}$ and $A \geq 0$, then there is an eigenvector $x \in \mathbb{R}^n$ with $x \geq 0$ and and $\sum_{i=1}^{n} x_i = 1$, called the *Perron vector* of A [14].

This result has a natural interpretation in the theory of finite Markov chains, where it is the matrix-theoretic equivalent of the convergence of a finite Markov chain, formulated in terms of the transition matrix of the chain [2].

The Perron vector can be viewed as a probability distribution of presence of a 'random listener' on a particular segment of a musical piece. This listener recalls with probability $d(s_i, s_j)$ segment s_j from segment s_i, following the 'links' represented by the values of the similarity function.

3.4 The algorithm

Let $d : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ denote a distance function on $\mathcal{S}$, like those defined in Section 3.2, which assigns each pair of segments s_i and s_j a distance $d(s_i, s_j)$. We can describe the algorithm through the following steps:

1. Form the distance matrix $A = [a_{i,j}]$ such that $a_{i,j} = d(s_i, s_j)$;

2. Form the affinity matrix $W = [w_{i,j}]$ defined by

$$w_{i,j} = \exp\left(-\frac{a_{i,j}^2}{2\sigma^2}\right) \qquad (3)$$

   where σ is a parameter that can be chosen experimentally. A possible choice is the standard deviation of the similarity values within the considered network graph;

3. Compute the leading eigenvector $x = [x_i]$ of W and rank each segment s_i according to the component x_i of x.

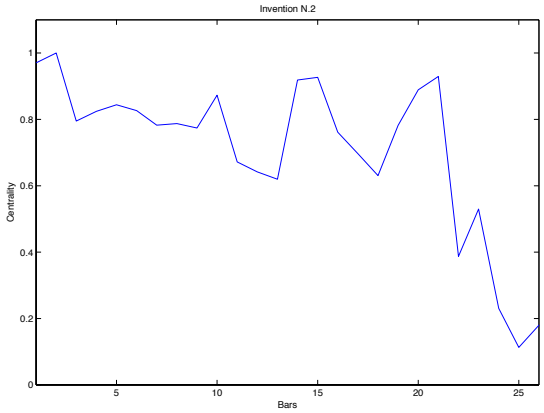


Figure 3. Normalized eigenvector profile for bars in BWV 773. Higher values correspond to higher centrality (see also Table 2). The metric space is $(\mathcal{S}, d_1)$.

4. EXPERIMENTAL RESULTS

In order to evaluate the relevance of the results of the proposed method we need a suitable data collection together with a commonly acceptable ground truth for that collection. Following [1], Johann Sebastian Bach's *Two-part Inventions* has been our choice. For this collection, a complete ground truth is provided by musicological analysis and it can be found for example in [12] and [24].

The first choice we had to make was the segment size. Many experiments has been conducted but, as stated before, it turned out that reductions of the segment size (for example from two bars to one bar) did not sensibly affect the results. So experiments have been performed with a one-bar long window. Experiments have been performed also to verify the suitability of an overlapping technique but we did not observed any improvement in the results.

Second, we implemented the functional metrics described in Section 3.3. By performing the experiments, we observed a few variations in the first two ranked levels, and this means that top ranked bars tend to be more "stable" respect to metric changes. Thus we can say that the method is rather robust, as far as these metrics are concerned. In the synthesis reported in Table 2 we considered just the top ranked segments, i.e. corresponding to the two (different) highest values of x components.

When compared to musicological analysis [1] [12] [24] it is evident that the centrality-based model outperforms the repetition-based model, providing also more significative information. Segments with higher rank in the relational model represent always relevant bars of the score, even if they may be different by using different metrics. This means that relevant bars contain a main motif or characterizing sequences. It is not the same for the model based

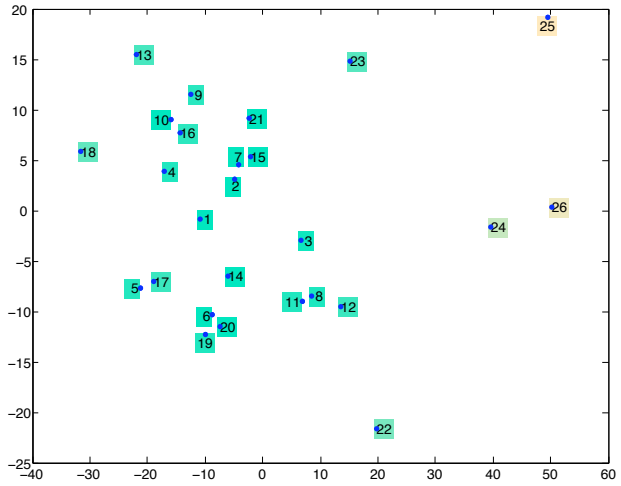


Figure 4. 2D projection of the metric space $(\mathcal{S}, d_1)$ for BWV 773. Bars with higher centrality values (darker labels) tend to occupy the central region of the graph.

Model	Precision (%)
Repetition	43
d_1	77
d_2	95

Table 1. Precision results for the three models applied to J. S. Bach's Inventions.

on repetitions: here the relevancy really depends just on the number of repetitions, so it can happen that a trill turns to be more relevant than the rest of the piece just because its repetition rate is higher than that of the other bars.

Bar ranking is in principle not affected by the repetition rate of patterns and higher importance is equally given to higher and lower repetition rates. Of course, superpositions of the two methods may happen too. On the other hand, cases exist for which no repetition occurs and, consequently, the repetition paradigm is not applicable in principle, unless defining ad hoc neighborhood concepts for each piece. In these cases, motif centrality can provide significant results.

In Figure 3 the components of the main eigenvector for BWV 773, representing the degree of centrality of each bar, have been plotted against bar numbers. This provides an immediate representation of the "importance" of each bar within the whole piece. Bars with higher values are more likely to contain a main motif of the piece. In particular, for BWV 773, bars 1 and 2 actually contain the main motif.

Figure 4 shows a two-dimensional projection of the 26-dimensional metric space for BWV 773 obtained through a dimensionality reduction algorithm. From this picture it is evident how the top ranked results (1, 2) occupy the central region of the graph and have darker labels, as the darkness is directly proportional to the correspondent component of the main eigenvector, and thus to the centrality, in the sense of graph theory, of the correspondent segment.

Table 1 presents a synthesis of the results shown in Table 2 in terms of the precision of the three methods. As for the computational complexity, suitable linear eigensolvers are available, and they can be easily applied, especially in case of very long pieces.

5. CONCLUSIONS

We presented a new approach for motif discovery in music pieces based on an eigenvector method. Scores are segmented into a network of bars and then ranked depending on their graph centrality. Bars with higher centrality are more likely to be musically relevant and can be exploited for music summarization. Experiments performed on the collection of J.S.Bach's 2-parts Inventions show the effectiveness of our method.

Besides music information retrieval, we expect this approach to find applications in music theory, perception and visualization. For instance, one could investigate how particular mathematical entities (e.g. spectra) relate to particular musical issues (e.g. genre, authorship).

Second, one could investigate how different metrics d relate to different concepts of melodic and harmonic similarity; in this context, the inverse problem of finding metrics d induced by a priori eigenvectors (coming from a hand-made musicological analysis) could provide interesting insights into music similarity perception.

Third, it is also possible to compare different music pieces from a structural point of view by comparing their associated eigenvectors.

Finally, the method could be extended to the audio domain, for instance to organize large audio collections, where heuristic methods can be hardly applied and it is usually difficult or even impossible to separate different voices and/or musical instruments.

6. REFERENCES

[1] K. Adiloglu, T. Noll, and K. Obermayer. A paradigmatic approach to extract the melodic structure of a musical piece. *Journal of New Music Research*, 35(3):221–236, 2006.

[2] T. Bedford, M.S. Keane, and C. Series. *Ergodic theory, symbolic dynamics, and hyperbolic spaces*. Oxford University Press New York, 1991.

[3] R. Bod. Memory-Based Models of Melodic Analysis: Challenging the Gestalt Principles. *Journal of New Music Research*, 31(1):27–36, 2002.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[5] C. Buteau and G. Mazzola. From Contour Similarity to Motivic Topologies. *Musicae Scientiae*, 4(2):125–149, 2000.

[6] E. Cambouropoulos. Extracting'Significant'Patterns from Musical Strings: Some Interesting Problems. *Presente aux London String Days*, 2000, 2000.

N^o	Catalog	Repeated bars	d_1	d_2
1	BWV 772	$\emptyset$	16, 18, 17, 3	16, 18
2	BWV 773	2, 3, 24, 25	2, 1, 21, 15	3, 25, 2, 24
3	BWV 774	2, 20, 42, 50, 55	54, 41, 19, 48	2, 20, 42, 50, 55
4	BWV 775	1, 2, 5, 6, 19, 20, 21, 44, 45	6, 2, 29, 22	1, 5, 44, 2, 6, 45
5	BWV 776	2, 3, 28, 29	2, 16, 17, 27	2, 28, 3, 29
6	BWV 777	5, 25, 42, 63, 84, 105	56, 21, 6, 20	5, 25, 42, 63, 84, 105
7	BWV 778	$\emptyset$	10, 8, 5, 1	7, 4
8	BWV 779	$\emptyset$	1, 6, 20	6, 26
9	BWV 780	1, 2, 3, 4, 29, 30, 31, 32	31, 14, 1, 2	1, 3, 29, 2
10	BWV 781	20, 21, 22, 23, 27, 29, 31	1, 17, 27, 32	1, 27, 29, 31
11	BWV 782	$\emptyset$	17, 4, 14, 6	6, 17
12	BWV 783	$\emptyset$	5, 6, 17, 16	9, 5
13	BWV 784	$\emptyset$	18, 2, 1, 21	8, 1
14	BWV 785	12, 14	8, 6, 7, 16	12, 14, 16
15	BWV 786	$\emptyset$	7, 10, 14, 9	12, 4

Table 2. Experimental results for the repetition paradigm (using both metrics d_1 and d_2) and the relational paradigm. Gray numbers represents irrelevant bars.

[7] E. Cambouropoulos. Musical pattern extraction for melodic segmentation. *Proceedings of the ESCOM Conference 2003*, 2003.

[8] E. Cambouropoulos, M. Crochemore, C. Iliopoulos, L. Mouchard, and Y. Pinzon. Algorithms for computing approximate repetitions in musical sequences. *International Journal of Computer Mathematics*, 79(11):1135–1148, 2002.

[9] E. Cambouropoulos and G. Widmer. Automated motivic analysis via melodic clustering. *Journal of New Music Research*, 29(4):303–318, 2000.

[10] R. Cilibrasi, P. Vitányi, and R. de Wolf. Algorithmic Clustering of Music Based on String Compression. *Computer Music Journal*, 28(4):49–67, 2004.

[11] T. Crawford, C.S. Iliopoulos, and R. Raman. String Matching Techniques for Musical Similarity and Melodic Recognition. *Computing in Musicology*, 11:73–100, 1998.

[12] E. Derr. The Two-Part Inventions: Bach's Composers' Vademecum. *Music Theory Spectrum*, 3:26–48, 1981.

[13] P. Di Lorenzo and G. Di Maio. The Hausdorff Metric in the Melody Space: A New Approach to Melodic Similarity. In *Ninth International Conference on Music Perception and Cognition*, 2006.

[14] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[15] O. Lartillot. Discovering musical patterns through perceptive heuristics. *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pages 89–96, 2003.

[16] Olivier Lartillot. A musical pattern discovery system founded on a modeling of listening strategies. *Comput. Music J.*, 28(3):53–67, 2004.

[17] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, Cambridge, Massachusetts, 1996.

[18] G. Mazzola and S. Müller. *The Topos of Music: Geometric Logic of Concepts, Theory, and Performance.* Birkhäuser, 2002.

[19] A. Nestke. Paradigmatic Motivic Analysis. *Perspectives in Mathematical and Computational Music Theory, Osnabrück Series on Music and Computation*, pages 343–365, 2004.

[20] A. Pienimaki. Indexing Music Databases Using Automatic Extraction of Frequent Phrases. *Proceedings of the International Conference on Music Information Retrieval*, pages 25–30, 2002.

[21] Alberto Pinto, Reinier van Leuken, Fatih Demirci, Frans Wiering, and Remco C. Veltkamp. Indexing music collections through graph spectra. In *Proceedings of the ISMIR 2007 Conference*, Vienna, September 2007.

[22] E. Selfridge-Field. Towards a Measure of Cognitive Distance in Melodic Similarity. *Computing in Musicology*, 13:93–111, 2004.

[23] Rainer Typke, Frans Wiering, and Remco C. Veltkamp. Transportation distances and human perception of melodic similarity. *Musicae Scientiae, Discussion Forum 4A, 2007 (special issue on similarity perception in listening to music), p. 153-182.*

[24] P.F. Williams. *JS Bach*. Cambridge University Press.

EVALUATING THE GENRE CLASSIFICATION PERFORMANCE OF LYRICAL FEATURES RELATIVE TO AUDIO, SYMBOLIC AND CULTURAL FEATURES

Cory McKay, John Ashley Burgoyne, Jason Hockman, Jordan B. L. Smith, Gabriel Vigliensoni and Ichiro Fujinaga
Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT)
McGill University, Montréal, Québec, Canada
[cory.mckay,jason.hockman,jordan.smith2]@mail.mcgill.ca,
[ashley,gabriel,ich]@music.mcgill.ca

ABSTRACT

This paper describes experimental research investigating the genre classification utility of combining features extracted from lyrical, audio, symbolic and cultural sources of musical information. It was found that cultural features consisting of information extracted from both web searches and mined listener tags were particularly effective, with the result that classification accuracies were achieved that compare favorably with the current state of the art of musical genre classification. It was also found that features extracted from lyrics were less effective than the other feature types. Finally, it was found that, with some exceptions, combining feature types does improve classification performance. The new lyricFetcher and jLyrics software are also presented as tools that can be used as a framework for developing more effective classification methodologies based on lyrics in the future.

1. INTRODUCTION

Automatic music classification is an important area of music information retrieval (MIR) research. Areas such as classification by genre, mood, artist and user tag have all received significant attention in the MIR literature. Classification is typically performed by training machine learning algorithms on features extracted from audio recordings, symbolic data or cultural information mined from the Internet. An interest in features extracted from textual transcriptions of lyrics has also become increasingly evident recently.

Most research to date has involved experiments involving one or, at most, two of these four types of data. This leaves unanswered questions as to whether improvements in classification performance might be achieved by combining features extracted from various combinations of these four musical data sources, especially with respect to the relatively new area of classification based on lyrics.

The first goal of the research presented here is to investigate this issue through a series of genre classification experiments on each possible subset combination of features extracted from lyrical, audio, symbolic and cultural data. Genre classification in particular is chosen because it is a well-established area of inquiry in the MIR literature that can be particularly difficult to perform well, and as such provides a good general basis for evaluation.

The second goal of this paper is to present software for mining lyrics from the Internet and for extracting features from them. There is not yet an established research toolset for performing these tasks, and the lyricFetcher and jLyrics software described here are intended to fill this gap.

2. PREVIOUS RESEARCH

2.1 Mining Lyrics from the Web

There are many web sites providing access to lyric transcriptions, including industry-approved pay services (e.g., Gracenote Lyrics), specialized lyric-scraping services (e.g., EvilLyrics, iWeb Scraping and Web Data Extraction), and other sites that amalgamate user contributions. The main difficulties encountered when automatically mining lyrics are associated with high variability in display formatting and content. Many sites also attempt to obscure lyrical content in the page source because of copyright concerns. There have been several attempts to extract and align lyrics from multiple sources automatically using dynamic programming [2,6], but these have encountered difficulties due to varying search results.

LyricsFly is one site that promises well-formatted lyrics and simplified searches accessible via a published API. Lyrics are provided in a convenient XML format, and multiple versions of songs are accessible. LyricWiki once provided a public API as well, but has since discontinued this service due to copyright concerns. Its content is still accessible via web browsing, however.

2.2 Extracting Classification Features from Lyrics

Logan et al. [10] and Mahedero et al. [11] provide important early contributions on analyzing lyrics using a

variety of techniques drawn from natural language processing, including topic modelling, to quantify musical similarity. Maxwell [12] also uses a large and varied feature set extracted from lyrics to rank similarity.

Mayer et al. [13] provide a particularly helpful examination of the classificatory power of various lyrical features with respect to genre. Kleedorfer et al. [5] and Wei et al. [18] present strategies for identifying topics, which can be adapted for use as classification features. Hirjee and Brown [3] present a sophisticated tool for extracting rhymes from lyrics, with a focus on hip-hop styles.

Some research has been performed on combining lyrical features with audio features in the context of artist, genre and mood classification [4,7,8,13]. Brochu and Freitas [1] have done research on combining lyrical features with features extracted from symbolic music.

2.3 jMIR

jMIR [14] is a suite of software tools and other resources developed for use in automatic music classification research. It was used to perform all of the experiments described in this paper. jMIR includes the following components:

- **jAudio:** An audio feature extractor.
- **jSymbolic:** A symbolic feature extractor.
- **jWebMiner 2.0:** A cultural feature extractor.
- **ACE 2.0:** A metalearning-based classifier.
- **jMusicMetaManager:** Software for managing and detecting errors in musical datasets.
- **jMIRUtilities:** Performs infrastructural tasks.
- **ACE XML:** Standardized MIR file formats.
- **Codaich, Bodhidharma MIDI and SAC:** Musical research datasets.

The jMIR software is all implemented in Java, which has advantages with respect to platform-independence. All jMIR components are open-source and are distributed free of charge at jmir.sourceforge.net.

2.4 Comparing the Performance of Feature Types

This paper expands upon the research described in [15], which experimentally investigated the classification utility of combining features extracted from audio, symbolic and cultural sources of musical information using an earlier version of jMIR. It was found that combining feature types did indeed substantially improve classification performance, in terms of both overall classification accuracy and the seriousness of those misclassifications that did occur.

To the best of the authors' knowledge, [15] is the only previous study involving cultural, symbolic and audio data. There have, however, been many important studies involving features extracted from pairs of musical data types, including [9] and [19]. Section 2.2 highlights additional work involving lyrics.

3. THE SLAC DATASET

The new SLAC (Symbolic Lyrical Audio Cultural) dataset is an expansion of the SAC dataset [15] that now includes lyrics. The purpose of this dataset is to facilitate experiments comparing the relative performance of features extracted from different types of musical data.

SAC consists of 250 MP3 recordings, 250 matching MIDI recordings and identifying metadata for each recording. This metadata is stored in an iTunes XML file that can be parsed by software such as jWebMiner in order to extract cultural features from the web.

SLAC adds lyrics to all of the non-instrumental musical pieces in SAC. These lyrics were mined from the Internet, as described in Section 4.

SLAC is divided into 10 genres, with 25 pieces of music per genre. These 10 genres consist of 5 pairs of similar genres, as shown in Figure 1. This arrangement makes it possible to perform 5-class genre classification experiments as well as 10-class experiments simply by combining each pair of related genres into one class, thus providing an indication of how well systems perform on both small and moderately sized genre taxonomies.

Blues: Modern Blues *and* Traditional Blues
Classical: Baroque *and* Romantic
Jazz: Bop *and* Swing
Rap: Hardcore Rap *and* Pop Rap
Rock: Alternative Rock *and* Metal

Figure 1: The ten genres found in the SLAC dataset and the five super-genres that they can be paired into.

SLAC includes some instrumental music. This complicates classification based on lyrics, as lyrics provide no way to distinguish one instrumental piece from another. Nonetheless, the inclusion of some instrumental music is necessary to evaluate classification performance properly, as one must simulate the music that classification systems will encounter in practice, including instrumental music.

4. MINING LYRICS WITH LYRICFETCHER

A new lyrics mining script called *lyricFetcher* was implemented in Ruby to automatically harvest lyrics from LyricWiki and LyricsFly. These two repositories were chosen for their large sizes and because of the simplicity of querying their collections: LyricsFly provides a simple API and LyricWiki offers a standardized URL naming scheme that is relatively easy to mine.

Once provided with a list of artist names and song titles to search for, lyricFetcher obtains lyrics in three steps: first, a query is made to the lyrics source; second, the lyrics themselves are extracted from the result; and third, lyrical content is cleaned and standardized in post-processing, an important step given the variability in for-

matting of user-contributed lyrics. In particular, raw retrieved lyrics are often abridged by providing a label for the first occurrence of a section (e.g., "chorus," "hook," "refrain," etc.) and repeating only this label when the section reoccurs. lyricFetcher automatically searches for and expands such sections. Common keywords added to the lyrical transcriptions, such as "verse," are also removed.

lyricFetcher was used to mine LyricWiki and LyricsFly for the lyrics to the recordings in Codaich and SLAC. These lyrics were used in the experiments described below in Section 6. Lyrics were manually retrieved from other web sources for the 20 pieces out of the 160 non-instrumental pieces in SLAC for which lyrics could not be harvested automatically from LyricWiki and LyricsFly.

5. EXTRACTING FEATURES FROM SLAC

5.1 Lyrical Features Extracted

A large number of features were implemented and extracted based on a survey of previous work and on original ideas: *AutomatedReadabilityIndex, AverageSyllablesPerWord, ContainsWords, FleshKincaidGradeLevel, FleshReadingEase, FunctionWordFrequencies, LetterBigramComponents, LetterFrequencies, LettersPerWordAverage, LettersPerWordVariance, LinesPerSegmentAverage,*[1] *LinesPerSegmentVariance, NumberOfLines, NumberOfSegments, NumberOfWords, PartOfSpeechFrequencies,*[2] *PunctuationFrequencies, RateOfMisspelling, SentenceCount, SentenceLengthAverage, TopicMembershipProbabilities,*[3] *VocabularyRichness, VocuabularySize, WordProfileMatch, WordsPerLineAverage* and *WordsPerLineVariance*. Descriptions of these features are provided at jmir.sourceforge.net/index_jLyrics.html.

5.2 The jLyrics Feature Extractor

A new Java-based feature extraction framework called *jLyrics* was implemented as part of this research. Like the existing jMIR feature extractors, it is designed to serve as an easy-to-use feature extraction application as well as an extensible framework for developing new features. It has the usual jMIR advantages in this respect [14], including a modular architecture, automatic resolution of feature dependencies and the option of saving feature values in several file formats. Many of the features described in Section 5.1 were implemented directly in jLyrics, although some features based on third-party libraries remain to be ported to the Java framework.

In addition to extracting features jLyrics can, given sets of lyrics belonging to a class, generate profiling reports indicating ranked lists of the most commonly used words in each class. These profiles can be used to "train" *WordProfileMatch* features to measure how well novel lyrics match each class' profile. Lyrics mined with lyricFetcher for the music in Codaich (with all pieces in SLAC filtered out) were used to do just this, in preparation for the experiments described in Section 6.

5.3 Audio, Symbolic and Cultural Feature Extraction

jMIR, as described in Section 2.3 and [14], was used to extract audio, symbolic and cultural features from SLAC. Of particular interest, the new jWebMiner 2.0 [17] software was used to extract cultural features based on both Yahoo! co-occurrence page counts and Last.FM user tags, as opposed to the older jWebMiner 1.0 used in [15], which only extracted features based on web searches. A newer version of ACE, ACE 2.0, was also used.

6. EXPERIMENTAL PROCEDURE

The first step of the experiment was to extract feature values from SLAC, as described in Section 5. This resulted in a set of 26 features (A) extracted from the audio version of each piece, 101 features (S) extracted from the MIDI version of each piece, 26 features (L) extracted from the lyrics for each piece and 20 features (C) extracted from the Internet based on the identifying metadata for each piece.[4] These four types of features were then grouped into all 15 possible subset combinations using jMIRUtilites. These feature groups are identified using the codes indicated in Table 1.

Feature Types	Identifying Code
Symbolic	S
Lyrical	L
Audio	A
Cultural	C
Symbolic + Lyrical	SL
Symbolic + Audio	SA
Symbolic + Cultural	SC
Lyrical + Audio	LA
Lyrical + Cultural	LC
Audio + Cultural	AC
Symbolic + Lyrical + Audio	SLA
Symbolic + Lyrical + Cultural	SLC
Symbolic + Audio + Cultural	SAC
Lyrical + Audio + Cultural	LAC
Symbolic + Lyrical + Audio + Cultural	SLAC

Table 1: The identifying codes for the feature type groups used in each of the experiments.

[1] A "segment" is a unit of text separated by line breaks.
[2] Extracted using the Stanford parts-of-speech tagger [18].
[3] Trained on Codaich (with SLAC instances filtered out) using latent Dirichlet allocation [8].

[4] The jMIR feature extractors are each capable of extracting more features than this, but were set to omit unpromising features in order to save processing time. Also, many of the features that were extracted are in fact feature vectors consisting of multiple values.

Features	5-Genre Accuracy (%)	10-Genre Accuracy (%)
S	85	66
L	69	43
A	84	68
C	100	86
SL	89	70
SA	95	74
SC	99	89
LA	88	66
LC	100	81
AC	100	85
SLA	93	77
SLC	99	84
SAC	100	89
LAC	99	83
SLAC	99	85

Table 2: Classification accuracies for each of the experiments. Feature codes are identified in Table 1. All values are averages across cross-validation folds.

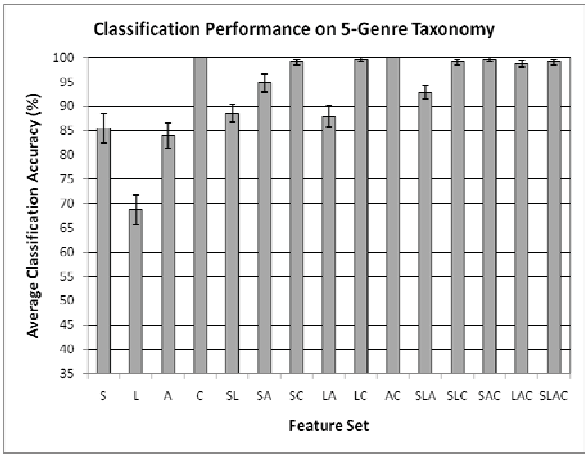


Figure 2: Results of the 5-genre experiments, as detailed in Table 2. Feature set codes are defined in Table 1.

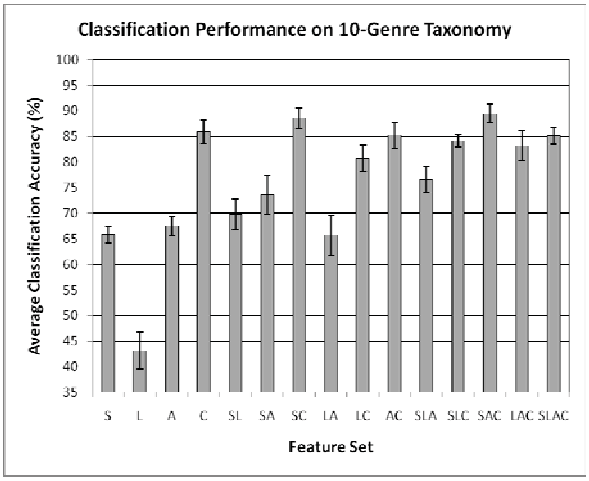


Figure 3: Results of the 10-genre experiments, as detailed in Table 2. Feature set codes are defined in Table 1.

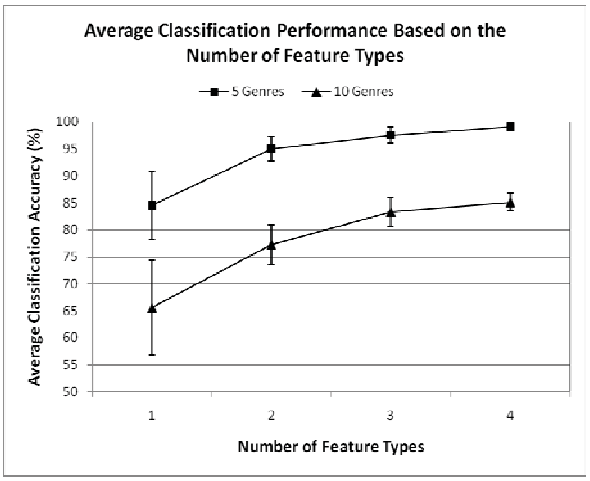


Figure 4: Classification accuracies averaged for all groups of, respectively, 1, 2, 3 and 4 feature types.

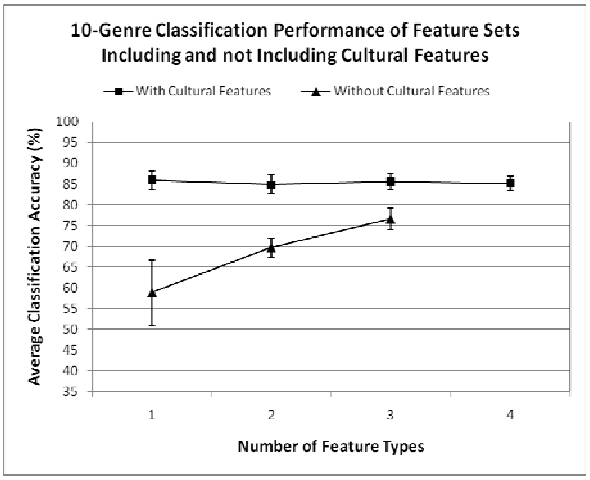


Figure 5: Average accuracies for feature groups including cultural features (C), compared to groups without C.

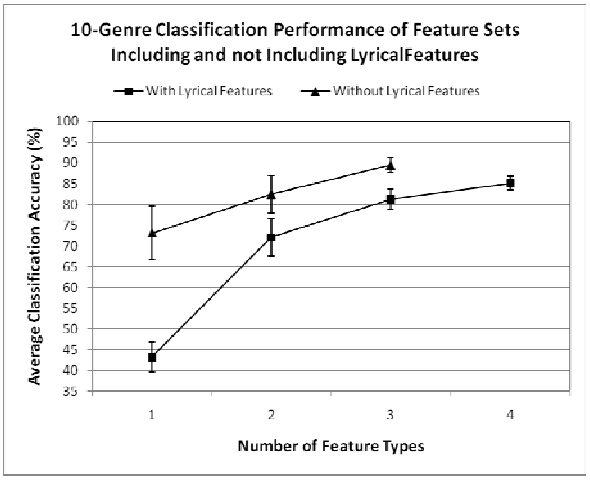


Figure 6: Average accuracies for feature groups including lyrical features (L), compared to groups without L.

jMIR ACE 2.0 was then used to classify each of these 15 feature sets by genre in 30 separate 10-fold cross-validation metalearning-based experiments, [5] such that each of the 15 feature sets was processed once in a 5-genre experiment and once in a 10-genre experiment. The results of these experiments are shown in Table 2. Figures 2 and 3 also represent this information graphically for the 5- and 10-genre taxonomies, respectively. The error bars on all figures represent standard error (i.e., the standard deviation of the cross-validation accuracies divided by the square root of the number of measurements).

7. RESULTS AND DISCUSSION

7.1 Overall Classification Performance

Overall, excellent classification accuracies were obtained with jMIR, with peak performances of 100% on the 5-genre taxonomy and 89% on the 10-genre taxonomy. For the purpose of comparison, the MIREX (www.music-ir.org/mirex/2010/) contests provide the best benchmarking references available. The highest MIREX symbolic genre classification performance to date is 84%, attained on a 9-genre ontology, and all six audio genre classification evaluations to date on genre ontologies larger than six classes have failed to achieve success rates above 80%. Although it is inappropriate to compare results obtained on different datasets directly, this does cast the results obtained here with jMIR in a favourable light.

7.2 Effect on Accuracy of Combining Feature Types

The next thing to consider was, now that lyrical features were included and the new jWebMiner 2.0 cultural features were used, whether combining different feature types still improved classification performance, as was the case in [15]. Figure 4 demonstrates the results of averaging together the classification accuracies of all feature groups with the same number of feature types (i.e., S, L, A and C; SL, SA, SC, LA, LC and AC; etc.), with a separate curve for each of the two genre taxonomies. It can be seen that, on average, classification accuracy did indeed increase with the number of feature types available.

It thus appears, at least upon first consideration, that combining features from different types of data does tend to improve performance. A closer examination of Table 2 shows that this was only true on average, however, as there were some cases where combining feature groups actually decreased performance (e.g., LC performed less well than C in the 10-genre experiments). Furthermore, an examination of Figure 5, described below, suggests that there was no advantage to combining cultural features in general with any other feature types.

7.3 Effectiveness of Cultural Features

Figure 5 shows, for the 10-class taxonomy, the average performance of all feature groups of the same size that contain cultural features, compared with the average performance of all feature groups of the same size that do not contain cultural features. The experimental results as a whole demonstrate that, for both taxonomies, cultural features significantly outperformed all other feature types.[6]

This dominance of cultural features was not apparent in [15], which only used cultural features derived from web searches. As described in [17], the new jWebMiner 2.0 combines these features with additional tag-based features extracted from Last.FM. This is likely responsible for the much higher performance of cultural features in this study relative to the results from [15].

7.4 Effectiveness of Lyrical Features

Figure 6 shows, for the 10-class taxonomy, the average performance of all feature groups of the same size that contain lyrical features, compared with the average performance of all feature groups of the same size that do not contain lyrical features. The results indicate that lyrical features were significantly less effective than the other feature types.[7] It is notable, however, that combining lyrical features with other feature types did, in some but not all cases, improve performance relative to the features operating individually. This is true for SL and SLA in both the 5- and 10-genre experiments. Furthermore, it is important to emphasize that 90 of the SLAC recordings were instrumental (although these recordings were strongly correlated with the Jazz and Classical genres).

8. CONCLUSIONS

This paper introduces the lyricFetcher and jLyrics tools for, respectively, mining lyrics from the web and extracting features from them. These tools are available for use in other research projects, and jLyrics in particular is designed to provide an easily extensible framework for implementing, testing and extracting new features.

With respect to the experiments described in this paper, excellent overall classification accuracies were obtained relative to the current state of the art of genre clas-

[5] A validation partition was reserved for each of the 30 experiments in order to guard against overfitting. Any experiment that resulted in an average cross-validation success rate that was higher than the validation performance with statistical significance was redone. It should also be noted that ACE includes dimensionality reduction functionality, so training was actually performed with automatically chosen subsets of the available features in order to avoid the "curse of dimensionality."

[6] Based on a Wilcoxon signed-rank test with a significance level of 0.05.

[7] Based on a Wilcoxon signed-rank test with a significance level of 0.05.

sification. In particular, the jWebMiner 2.0 cultural features based on both web searches and listener tags extracted from Last.FM were especially effective. It was also found that combining different feature types improved performance on average if cultural features were unavailable, but was not necessary if cultural features were available. With respect to lyrical features, it was found that combining them with other types of features did, in certain cases, improve classification performance. Overall, however, lyrical features performed poorly relative to the other feature types.

The disappointing performance of the lyrical features was probably due in part to noisiness in the mined lyrical transcriptions, including inconsistent annotation practices, occasional errors and the inclusion of non-standardized markup in XML and other formats. The relatively low performance of lyrics was likely also partly due to inherent limitations with respect to classifying instrumental music, as well as to the general-purpose text mining orientation of the lyrical features used. This highlights the need for continued research on more specialized music-oriented lyrical features, and on still better lyric mining and cleaning methodologies. Both of these could potentially lead to significantly improved performance by lyrical features.

9. REFERENCES

[1] Brochu, E., and N. de Freitas. 2003. "Name that song!": A probabilistic approach to querying music and text. In *Advances in Neural Information Processing Systems* 15, 1505–12. Cambridge, MA: MIT Press.

[2] Geleijnse, G., and J. Korst. 2006. Efficient lyrics extraction from the web. *Proc. of the Int. Conference on Music Information Retrieval*. 371–2.

[3] Hirjee, H., and D. G. Brown. 2009. Automatic detection of internal and imperfect rhymes in rap lyrics. *Proc. of the Int. Society for Music Information Retrieval Conference*. 711–6.

[4] Hu, X, J. S. Downie, and A. F. Ehman. 2009. Lyric text mining in music mood classification. *Proc. of the Int. Society for Music Information Retrieval Conference*. 411–6.

[5] Kleedorfer, F., P. Knees, and T. Pohle. 2008. Oh oh oh whoah! Towards automatic topic detection in song lyrics. *Proc. of the Int. Conference on Music Information Retrieval*. 287–92.

[6] Knees, P., M. Schedl, and G. Widmer. 2005. Multiple lyrics alignment: Automatic retrieval of song lyrics. *Proc. of the Int. Conference on Music Information Retrieval. 564–*9.

[7] Laurier, C., J. Grivolla, and P. Herrera. 2008. Multimodal music mood classification using audio and lyrics. *Proc. of the Int. Conference on Machine Learning and Applications.* 688–93.

[8] Li, T., and M. Ogihara. 2004. Semi-supervised learning from different information sources. *Knowledge and Information Systems* 7 (3): 289–309.

[9] Lidy, T., A. Rauber, A. Pertusa, and J. M. Iñesta. 2007. Improving genre classification by combination of audio and symbolic descriptors using a transcription system. *Proc. of the Int. Conference on Music Information Retrieval*. 61–6.

[10] Logan, B., A. Kositsky, and P. Moreno. 2004. Semantic analysis of song lyrics. *Proc. of the IEEE Int. Conference on Multimedia and Expo*. 827–30.

[11] Mahedero, J. P. G., Á. Martínez, and P. Cano. 2005. Natural language processing of lyrics. *Proc. of the ACM Int. Conference on Multimedia.* 475–8.

[12] Maxwell, T. 2007 Exploring the Music Genre: Lyric Clustering with Heterogeneous Features. *M.Sc. thesis,* University of Edinburgh.

[13] Mayer, R., R. Neumayer, and A. Rauber. 2008. Combination of audio and lyrics features for genre classification in digital audio collections. *Proc. of the ACM Int. Conference on Multimedia*. 159–68.

[14] McKay, C. 2010. Automatic music classification with jMIR. *Ph.D. Dissertation*. McGill University, Canada.

[15] McKay, C., and I. Fujinaga. 2008. Combining features extracted from audio, symbolic and cultural Sources. *Proc. of the Int. Conference on Music Information Retrieval*. 597–602.

[16] Neumayer, R., and A. Rauber. 2007. Integration of text and audio features for genre classification in music information retrieval. *Proc. of the European Conference on IR Research*. 724–7.

[17] Vigliensoni, G., C. McKay, and I. Fujinaga. 2010. Using jWebMiner 2.0 to improve music classification performance by combining different types of features mined from the web. Accepted for publication at the *Int. Society for Music Information Retrieval Conference*.

[18] Wei, B., C. Zhang, and M. Ogihara. 2007. Keyword generation for lyrics. *Proc. of the Int. Conference on Music Information Retrieval*. 121–2.

[19] Whitman, B., and P. Smaragdis. 2002. Combining musical and cultural features for intelligent style detection. *Proc. of the Int. Symposium on Music Information Retrieval*. 47–52.

EVALUATION OF A SCORE-INFORMED SOURCE SEPARATION SYSTEM

Joachim Ganseman, Paul Scheunders
IBBT - Visielab
Department of Physics, University of Antwerp
2000 Antwerp, Belgium

Gautham J. Mysore, Jonathan S. Abel
CCRMA
Department of Music, Stanford University
Stanford, California 94305, USA

ABSTRACT

In this work, we investigate a method for score-informed source separation using Probabilistic Latent Component Analysis (PLCA). We present extensive test results that give an indication of the performance of the method, its strengths and weaknesses. For this purpose, we created a test database that has been made available to the public, in order to encourage comparisons with alternative methods.

1. INTRODUCTION

Source separation is a difficult problem that has been a topic of research for several decades. It is desirable to make use of any available information about the problem to constrain it in a meaningful way. Musical scores provide a great deal of information about a piece of music. We therefore use this information to guide a source separation algorithm based on PLCA.

PLCA [1] is a technique that is used to decompose magnitude spectrograms into a sum of outer products of spectral and temporal components. It is a statistical interpretation of Non-Negative Matrix Factorization (NMF) [2]. The statistical framework allows for a structured approach to incorporate prior distributions.

Extraction of a single source out of a sound mixture by modeling a user guidance as a prior distribution was presented in [3]. In our previous work [4], we based ourselves on that approach and extended it to a complete source separation system informed by musical scores, finally demonstrating it by separating sources in a single real-world recording.

We perform source separation by decomposing the spectrogram of a given sound mixture using PLCA, and then performing reconstructions of groups of components that correspond to a single source. Before using PLCA on the sound mixture, we first decompose synthesized versions of those parts of musical scores that correspond to the sources that we wish to separate (also using PLCA). The temporal and spectral components obtained by these decompositions of synthesized sounds are then used as prior distributions while decomposing the real sound mixture.

In this work we make a detailed evaluation of such source separation system and its overall performance. To the best of our knowledge, a comprehensive and extensive dataset to use as ground truth for such problem does not exist, mainly because we also require the corresponding scores as additional information to the source separation system. We therefore construct a test set of our own, mimicking realistic conditions as well as possible even though it is synthetic. This also allows us to make detailed evaluations of how the results are affected by common performance practices, like changes in tempo or synchronization. To get objective quality measurements of this method, we use the metrics defined in the BSS_EVAL framework [5], which are widely adopted in related literature.

2. SCORE-INFORMED SOURCE SEPARATION WITH PLCA

We're not the first to propose source separation based on score information. A method based on sinusoidal modeling has been proposed by Li [6], and Woodruff [7] used scores as information source for the separation of stereo recordings. Our PLCA-based system for score-informed source separation is set up as shown in fig. 1 :

- The complete score gets synthesized;
- Dynamic Time Warping (DTW) matches the spectrogram of the sound mixture to that of the score;
- The resulting path is used to match single parts or sections from the score to the mix;
- Components for each of the parts to extract are learned using PLCA on separately synthesized parts;
- These components are used as prior distributions in the subsequent PLCA decomposition of the mix;
- With the learned components 'fitted' to the mix, we can now resynthesize only those components from the mix that we want.

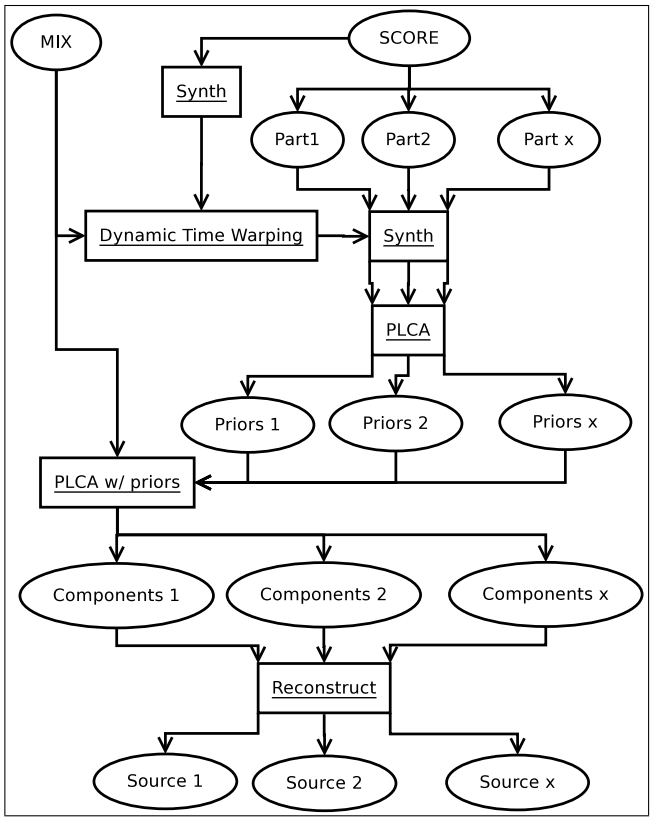


Figure 1. Architecture of the score-informed PLCA-based source separation system

The PLCA method that we adopt does not presuppose any structure, instead it learns the best representation for a spectrogram through an EM (expectation-maximization) algorithm. Both temporal and spectral components can assume any shape. The dictionary of spectral and temporal components resulting from decomposition of the synthesized score parts is only used to initialize the subsequent PLCA decomposition of the sound mixture. The EM-iterations decomposing this mixture optimize those spectral and temporal components further in order to make them explain the sources in the sound mixture. A drawback of PLCA is that it operates on magnitude spectrograms and does not take into account the phase, which easily leads to some audible distortion in the resynthesized sounds.

We implemented our system largely in Matlab, with the DTW routine provided from [8]. We always work on mono audio. The method does not work in real-time - on a modern dual-core 3.0GHz computer with 4GB RAM memory, processing a 1 minute sound file (44100Hz samplerate) takes about 3 to 4 minutes of calculation time with high quality settings. The DTW subroutine has a memory complexity that is quadratic in spectrogram size, due to the calculation of a complete similarity matrix between the spectrograms of the sound mixture and the synthesized score. Alternatives to DTW exist and could be used, there is e.g. prior work on aligning MIDI with audio without computing a complete rendering of the MIDI file (also available through [8]).

It needs to be noted that the spectral and temporal components of the synthesized score parts are initialized with random data. Starting from these random probability distributions, the EM-algorithm then iteratively estimates better candidates that fit the data. The resulting estimates of the components from the score data will be slightly different on each run. This will in turn affect the subsequent PLCA analysis of the real data and its path towards convergence. We will quantify this in more detail later, but it is important to keep in mind that all measurements presented in this paper are subject to a certain error margin that is a direct result of this random initialization.

3. TEST SETUP

In order to do large scale comprehensive testing of this method, we need a database of real sources and their scores which we can mix together and then try to separate. To the best of our knowledge, a carefully crafted database for research purposes containing separate sources and their scores for a wide range of instruments and/or styles does not yet exist [9]. For source separation, evaluation databases with multitrack recordings are available (e.g. [10]), but usually they don't come with scores, MIDI files, or any other symbolic information.

We decided to create our own database, generating short random MIDI tunes, and then running them through different synthesizers. In testing, one of the synthesized sounds then can take on on the role of 'real performance', while the other is used as 'synthesized score'. To better simulate real performance, we generated several versions of each file with tempos regularly changing, up to half or double the speed of the original. This also allows the database to be used to test alignment algorithms. The resulting dataset is available online [1].

We generated a set of 10 second sound files using PortSMF [11]. The files were synthesized once using Timidity++ [12] with the FluidR3 GM soundfont on Linux, and once with the standard built-in MIDI player capabilities in Windows XP (DirectSound, saved to file using WinAmp [13]). Each file contains on average 20 note onsets, spread randomly over 10 seconds. We reduced our test set to 20 commonly used instruments, both acoustic and electric. This was done in part because of a lot of the sounds standardized in General MIDI are rarely found in scores (helicopter sounds, gunshots), and to keep the size of the resulting data manageable. With 380 possible duos with different instruments out of 20 instruments, it allowed us to run repeated experiments on all of these combinations.

This original test set of 20 sounds was expanded by introducing timing variations in the MIDI files. Several sets of related files were generated, in which the tempo in each

[1] https://ccrma.stanford.edu/~jga/ismir2010/ismir2010.html

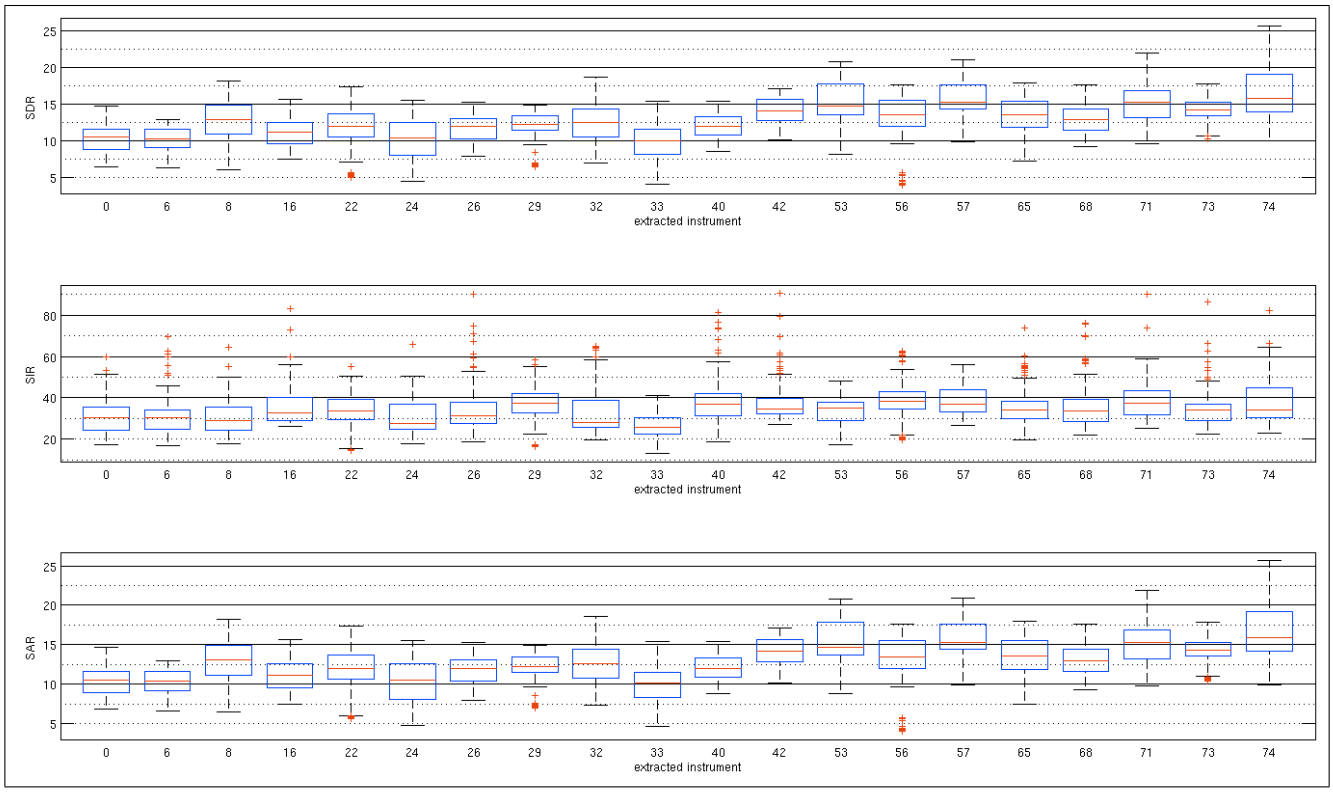


Figure 2. Overall source extraction scores per instrument, mixed with any other instrument. On the x-axis, the MIDI Program Change number. In this and all other figures, standard Matlab style boxplots are used.

file was changed 5 times - this to be able to test the effects of the method used to align symbolic data and recordings, which is part of the system. 2 distinctly different tempo curves were defined, and for each of these 2 curves, 5 new renditions were made for every original source. The first of these 5 would have the tempo changed by up to 10%, either slower and faster, while the last would allow deviations from the original tempo up to 50% . Thus we have a dataset of 20 original sources, and for each original file also 10 files with all different variations in tempo.

We acknowledge that there are a couple of drawbacks with this dataset. The first one that the files are randomly generated, while in most popular and classical music, harmonic structure makes separation more difficult due to overlapping harmonic components. The second that even using two different soundbanks to synthesize can result in the two synthesized versions of a single file to be more similar to eachother than they might be similar to a real recording. We found however that the timbres of the two soundbanks used differ quite significantly. As for the random generation: not using real data frees us from dealing with copyright issues, and generating it randomly allowed us to quickly obtain a large and comprehensive body of test files, not presupposing any structure or style.

In the following sections, we use the files generated on Windows as sources for the 'performance' sound mixture, and the files rendered on Linux as 'scores' from which we obtain priors. The BSS_EVAL toolbox [5] calculates 3 metrics on the separated sources given the original data. The Signal-to-Interference Ratio (SIR) measures the inclusion of unwanted other sources in an extracted source; the Signal-to-Artefacts Ratio (SAR) measures artefacts like musical noise; the Signal-to-Distortion Ratio (SDR) measures both the interference and artefacts.

4. MEASUREMENTS ON IDEAL DATA

4.1 Error margin on the results

As mentioned previously, due to randomness in the initialization, separation results might differ with every run, and so might the SDR, SIR and SAR scores. To properly quantify what we are dealing with, we ran the system with standard parameters that give decent results (sampling rate of 44100Hz, 2048-point FFT with 75% overlap, 50 components per source, 50 iterations) 10 times on each of the possible 380 instrument duos in the test set. This was done

	min std	max std	mean std	median std
SDR	0.062	1.55	0.48	0.41
SIR	0.056	14.21	1.89	1.20
SAR	0.061	1.55	0.47	0.41

Table 1. Reliability of the results: statistics on the standard deviation of SDR, SIR and SAR scores of 10 runs of the algorithm with the same parameters on 380 data pairs.

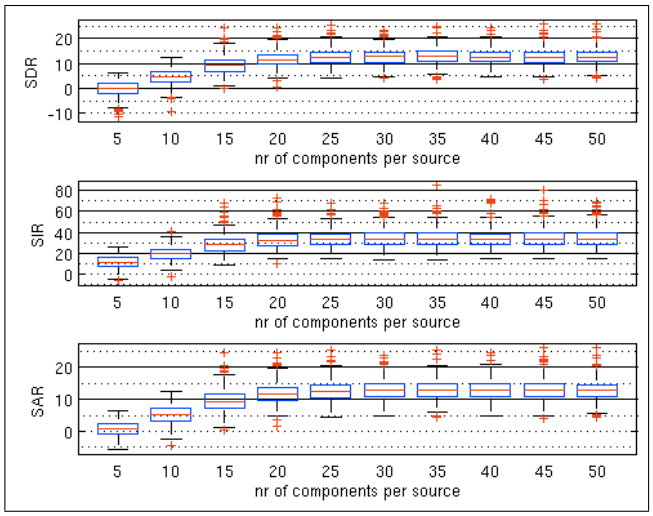


Figure 3. SDR, SIR and SAR vs. number of components used per source

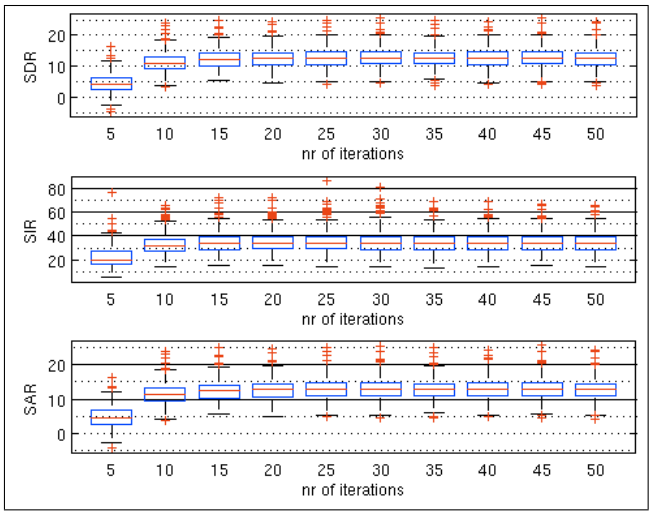


Figure 4. SDR, SIR and SAR vs. number of iterations in EM algorithm

on 'ideal' data, where the score would exactly line up with the sound mixture and no DTW was needed, so we compute the effect of the random initialization only. SDR, SIR and SAR values tend to be pretty consistent in between runs on the same pair, except in a few rare combinations where there is a lot of variance in the results. Even in these cases, the mean values of the scores are still within normal range.

The numbers in table 1 show that the mean standard deviation of calculated SDR and SAR scores stays below 0.5 dB, while there can be highly variable results in some SIR scores. Incidences of high variance in SIR score seem unrelated to each other, and almost every instrument had some combination with another instrument where SIR scores would be very variable in between runs of the algorithm. For evaluation purposes, SDR and SAR scores seem to be better suited to pay attention to.

Some instruments seem to be easier to extract from mixes with any other instrument, than others. Fig. 2 gives an idea of the 'overall easiness' with which an instrument can be extracted from a mix.

4.2 Components and iterations

The algorithm's running time will increase linearly with the amount of components. Generally, increasing the number of components that are available per source increases the ability to model the priors accurately, and thus also the overall separation results. We ran a small test of the effect of the number of components across all couples of sources. The effectiveness of the number of components is almost identical for every instrument, so we can generally plot the number of components versus the outcome of the metrics, which is shown in fig 3.

With on average 20 notes in each source, there is a huge climb in improvements up to 20 components after which the scores level off. There is some small improvement after this, but not drastic. We haven't run complete tests with significantly larger amounts of components, but from a couple of single tries we find that overfitting becomes an issue when the amount of components is chosen too large. Superfluous components of a single source risk to start modeling parts of other sources, which degrades separation performance again.

The number of iterations of the EM algorithm does not suffer from this - since the likelihood of subsequent EM iterations is monotonically increasing, more is always better. The only constraint here is how much time we're willing to spend on those iterations. We can see that the convergence towards a good solution is obtained rather fast: independent of instrument, above 25 iterations there is hardly any improvement of the scores (fig. 4).

4.3 Other parameters

The PLCA routine decomposes a magnitude spectrogram, and thus the properties of that spectrogram also play a role in the end result. Conducting a few small tests, we were able to conclude that the larger the FFT size, the better the results generally are. In subsequent tests, we used 2048-point FFTs. The overlap should be kept above 67.5% ; 75% is a safe value. Binary masking (assigning each spectrogram time-frequency bin to a single source instead of dividing it among different sources) significantly improves SIR scores, at the cost of a slight decrease in SDR and SAR scores.

It is possible to cut the spectrogram into 'timeslices' of variable length. Certainly when there are possibilities for parallelization, or when due to system limitations the spectrogram size needs to be kept to a minimum, it might be interesting to run the analysis on each slice separately.

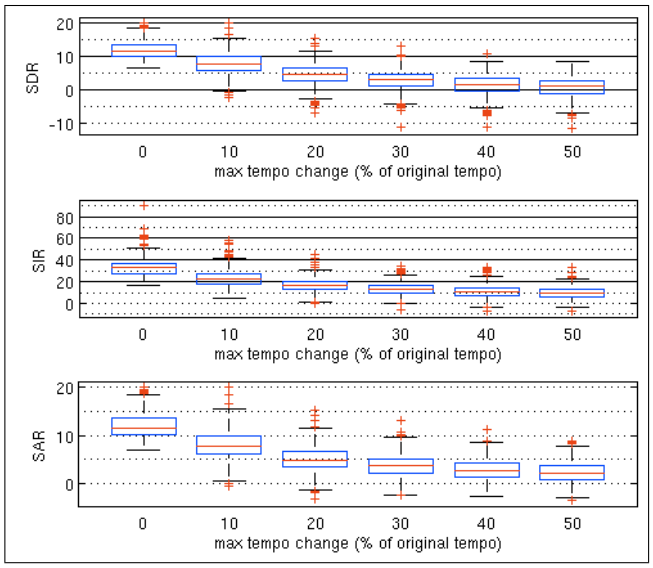


Figure 5. SDR, SIR and SAR vs. tempo deviation from reference, all sources with the same deviation

This makes that the spectral components, and their temporal counterparts, can change from slice to slice. Due to the random initialization of the components, they are likely not related to the components in other slices at all, and each slice will have its components defined such that they represent the data in that slice optimally. During resynthesis, small artefacts can be introduced on the slice borders due to these changes in basis vectors that occur. Our tests indicated that, though a decline in scores remains small and only noticeable when slices were smaller than a second long, it is a good idea to have the length of a slice as large as possible. How this relates to the number of needed components or iterations remains to be studied in the future.

5. THE EFFECT OF DYNAMIC TIME WARPING

5.1 Quantifying the role of DTW

Whereas in the previous section we discussed metrics on ideally aligned data, this is not likely to occur in real life. Performers use their artistic freedom to make the notes on paper into a compelling concert. One of the main means of doing so are local changes in tempo. To cope with this, a DTW routine is attached at the beginning of the system. It serves to line up the score with the real performance.

In figure 5 the performance of the algorithm on ideally aligned sound mixtures (0% deviation from the score tempo) is compared to performance on mixtures with tempo deviations, where alignment is needed. The sources that were used were divided into 5 segments that each had a different tempo assigned to them, in such a way that the tempo was in every file partly below the reference tempo, and partly above. The amount of change has a pretty high influence on the effectiveness of the subsequent source separation.

Logically, the timing of an entire score applies to the individual instrument parts too. We provide the output from the DTW routine to a phase vocoder that dilates or compresses each of the synthesized parts in time, so that they match up with the the performance mixture. This is a quick and practical solution to make sure that in the following PLCA analysis steps, the temporal and spectral components of the performance mixture and their associated priors obtained from the synthesized score parts, have the same dimensions.

Both errors in alignment and the subsequent stretching of synthesized score parts introduce errors in the priors, which affect successful analysis. From the data in fig. 5, we conclude that heavy time warping and subsequent stretching of the spectrum puts the quality of the results at severe risk. The DTW routine and phase vocoder that we used [8] were chosen because they were readily available to plug into our code. It is however a bottleneck in our system. In future work, alternative methods to align scores with recordings are worth looking into [14]. If using DTW, in practical applications the possibility to manually correct or at least smoothen the time alignment should be available.

In tests where source files with different tempo curves were used in a single sound mixture (in order to simulate performers that are out of sync with each other), very similar results were observed. In such a case the time alignment is likely to contains errors for at least one of the sources, since notes that should be played together according to the score, are not necessarily played together in the mixture. We can conclude that the application of DTW and subsequent time dilating and compressing of the synthesized data with a phase vocoder can cause a considerable stir in the computed priors, to such an extent that in the subsequent decomposition of the mix, it becomes very difficult to get decent separation results.

5.2 Adaptations and alternatives

The DTW plus phase vocoder routine is the weak link in the complete process, and we ventured on to do a couple of experiments adapting that part of our system. Inspired by recent work by Dannenberg et al [15] we substituted the spectrograms used in the DTW routine by chromagrams, using code obtained from the same source [8]. The results are practically equal to those in fig. 5 . Just like in the case of DTW with spectrograms, some (manual) post-processing on the results of the DTW routine is likely to improve the test results.

We also undertook a small experiment skipping the use of a phase vocoder to stretch the spectrograms of the scores, instead only resampling the temporal vectors using piecewise cubic Hermite polynomials to maintain nonnegativity. It turns out that the mean SDR and SAR scores plummet, and the standard deviation increases drastically, resulting in a small but not negligible number of test results

that are actually better than what could be attained previously. Also, the SIR values stay remarkably high, and even at large tempo deviations. However, overall the system became highly unreliable and unfit for general use.

Given that the parameters of the PLCA routine can be chosen optimally and that their effects are relatively well-known, most of the future effort in improving this score-informed separation system should clearly go into better and more accurate alignment and matching of the scores to the real performance data. Also more varied data and use cases need to be considered - here, we only worked on mixes of 2 instruments, and did not include common performance errors like wrongly struck notes. Several approaches to solve this problem, or parts of it, exist or are being worked on [14], and can contribute to a solution.

For alignment of scores with recordings, we have some future work set out, replacing the DTW and phase vocoder with methods more fit for our particular setup. In hindsight, with symbolic data and performance recordings available, we would very likely be better off applying a method that directly aligns symbolic information with a single spectrogram, to then modify the timing of the symbolic data, only then to synthesize it and compute priors from. For any future developments, we now have an extensive dataset to quickly evaluate the system.

6. CONCLUSIONS

In this paper we quantified the performance of a recently developed score-informed source separation framework based on PLCA. Several parameter options were explored and we paid special attention to the effect of DTW. The use of metrics that are prevalent in literature allows for future comparison with competing methods. We synthesized our own test dataset covering a wide range of instruments, using different synthesizers to mimick the difference between real-world data and scores, and mimicking some performance characteristics by introducing tempo changes. This dataset has been made freely available to the general public, and we exemplified its usability for extensive testing of alignment and source separation algorithms.

7. REFERENCES

[1] P. Smaragdis, B. Raj and M.V. Shashanka: "Supervised and Semi-Supervised Separation of Sounds from Single-Channel Mixtures," *Proc. of the 7th International Conference on Independent Component Analysis and Signal Separation*, London, UK, September 2007.

[2] D. Lee and H.S. Seung: "Algorithms for Non-negative Matrix Factorization," *Proc. of the 2000 Conference on Advances in Neural Information Processing Systems*, MIT Press. pp. 556562.

[3] P. Smaragdis and G. Mysore: "Separation by 'humming': User-guided sound extraction from monophonic mixtures," *Proc. of IEEE Workshop on Applications Signal Processing to Audio and Acoustics* , New Paltz, NY, October 2009.

[4] J. Ganseman, G. Mysore, P. Scheunders and J. Abel: "Source separation by score synthesis," *Proc. of the International Computer Music Conference*, New York, NY, June 2010.

[5] C. Févotte, R. Gribonval and E. Vincent: "BSS_EVAL, A toolbox for performance measurement in (blind) source separation," *Available at http://bass-db.gforge.inria.fr/bss_eval/*, accessed May 27, 2010.

[6] Y. Li, J. Woodruff and D. L. Wang: "Monaural musical sound separation using pitch and common amplitude modulation," *IEEE Trans. Audio, Speech and Language Processing*, vol. 17, no. 7, pp. 1361-1371, 2009.

[7] J. Woodruff, B. Pardo and R. B. Dannenberg: "Remixing Stereo Music with Score-informed Source Separation," *Proc. of the 7th International Conference on Music Information Retrieval*, Victoria, Canada, October 2006.

[8] D. Ellis: "Matlab audio processing examples," *Available at http://labrosa.ee.columbia.edu/matlab/* accessed May 27, 2010.

[9] A. Grecu: "Challenges in Evaluating Musical Instrument Sound Separation Algorithms," *Proc. 9th International Student Workshop on Data Analysis (WDA2009)*, Certovica, Slovakia, July 2009, pp. 3-9.

[10] E. Vincent, R. Gribonval, C. Févotte et al,. "BASS-dB: the Blind Audio Source Separation evaluation database." *Available at http://www.irisa.fr/metiss/BASS-dB/* , accessed May 27, 2010.

[11] R. B. Dannenberg and contributors: "PortSMF, part of PortMedia" *Available at http://portmedia .sourceforge.net/*, accessed May 27, 2010.

[12] Masanao Izumo and contributors: "Timidity++," *Available at http://timidity.sourceforge.net* , accessed May 27, 2010.

[13] NullSoft, Inc: "Winamp," *Available at http://www.winamp.com*, accessed May 27, 2010.

[14] R. B. Dannenberg and C. Raphael: "Music Score Alignment and Computer Accompaniment," *Communications of the ACM*, vol. 49, no. 8 (August 2006), pp. 38-43.

[15] R. B. Dannenberg and G. S. Williams: "Audio-to-Score Alignment In the Audacity Audio Editor," *Late Breaking Demo session, 9th International Conference on Music Information Retrieval*, Philadelphia, USA, September 2008.

EVIDENCE FOR PIANIST-SPECIFIC RUBATO STYLE IN CHOPIN NOCTURNES

Miguel Molina-Solana
Dpt. Computer Science and AI
University of Granada, Spain
miguelmolina at ugr.es

Maarten Grachten
IPEM - Dept. of Musicology
Ghent University, Belgium

Gerhard Widmer
Dpt. of Computational Perception
Johannes Kepler Univ., Austria

ABSTRACT

The performance of music usually involves a great deal of interpretation by the musician. In classical music, the final ritardando is a good example of the expressive aspect of music performance. Even though expressive timing data is expected to have a strong component that is determined by the piece itself, in this paper we investigate to what degree individual performance style has an effect on the timing of final ritardandi. The particular approach taken here uses Friberg and Sundberg's kinematic rubato model in order to characterize performed ritardandi. Using a machine-learning classifier, we carry out a pianist identification task to assess the suitability of the data for characterizing the individual playing style of pianists. The results indicate that in spite of an extremely reduced data representation, when cancelling the piece-specific aspects, pianists can often be identified with accuracy above baseline. This fact suggests the existence of a performer-specific style of playing ritardandi.

1. INTRODUCTION

Performance of music involves a great deal of interpretation by the musician. This is particularly true of piano music from the Romantic period, where performances are characterized by large fluctuations of tempo and dynamics. In music performance research it is generally acknowledged that, although widely used, the mechanical performance (with a constant tempo throughout the piece) is not an adequate norm when studying expressive timing, since it is not the way a performance should naturally sound.

As an alternative, models of expressive timing could be used, as argued in [18]. However, only few models exist that deal with expressive timing in general [2, 16]. Due to the complexity and heterogeneity of expressive timing, most models only describe specific phenomena, such as the timing of grace notes [15] or the final ritardando.

Precisely, the final ritardando —the slowing down toward the end of a musical performance to conclude the piece gracefully— is one of the clearest manifestations of expressive timing in music. Several models have been proposed [3, 14] in the related literature to account for its specific shape. Those models generally come in the form of a mathematical function that describes how the tempo of the performance changes with score position.

In a previous empirical study by Grachten *et al.* [4] on the performance of final ritardandi, a kinematic model [3] was fitted to a set of performances. Even though some systematic differences were found between pianists, in general the model parameters tend to reflect primarily aspects of the piece rather than the individual style of the pianist (i.e. expressive timing data is expected to have a strong component that is determined by piece-specific aspects).

This fact is relevant in a recurrent discussion in the field of musicology, about which factor (the piece or the performer) mostly influences a performance [9]. Some experts argue that the performance should be preceded of a thorough study of the piece; while others indicate that the personal feeling of music is the first and main point to be considered. Works supporting both views can be found in [12]. A study by Lindström *et al.* [7] involving a questionnaire, showed that music students consider both the structure of the piece and the feelings of the performer as relevant in a performance.

The current paper extends that previous work by Grachten *et al.*, by investigating whether or not canceling piece-specific aspects leads to a better performer characterization. Musicologically speaking, the validation of this hypothesis implies that performers' signatures do exist in music interpretation regardless of the particular piece. We present a study of how final ritardandi in piano works can be used for identifying the pianist performing the piece. Our proposal consists in applying a model to timing data, normalizing the fitted model parameters per piece and searching for performer-specific patterns.

Performer characterization and identification [8, 13] is a challenging task since not only the performances of the same piece by several performers are compared, but also the performance of different pieces by the same performer. Opposed to performer identification (where performers are supposed to have distinctive ways of performing) is piece identification —which requires the structure of the piece to imply a particular expressive behavior, regardless of the performer.

A further implication of this work would be that, when

an estimation can be made of the prototypical performance based on the musical score, this estimation could be a useful reference for judging the characteristics of performances. This knowledge can also allow the artificial interpretation of musical works by a computer in expressive and realistic ways [17].

This paper is organized as follows: Section 2 describes the dataset used for this study, including the original timing data and the model we fit them to. Section 3 deals with the data processing procedure. Results of the pianist classification task are presented and discussed in Section 4, while Section 5 states conclusions and future work.

2. DATA

The data used in this paper come from measurements of timing data of musical performances taken from commercial CD recordings of Chopin's Nocturnes. This collection has been chosen since these pieces exemplify classical piano music from the Romantic period, a genre that is characterized by the prominent role of expressive interpretation in terms of tempo and dynamics. Furthermore, Chopin's Nocturnes is a well-known repertoire, performed by many pianists, and thus facilitating large scale studies.

As explained before, models of expressive timing are generally focused in a certain phenomenon. In our study, we will focus on the final ritardando of the pieces. Hence, we select those Nocturnes whose final passages have a relatively high note density and are more or less homogeneous in terms of rhythm. With these constraints we avoid the need to estimate a tempo curve from only few interonset intervals, and reduce the impact of rhythmic particularities on the tempo curve.

In particular, we used ritardandi from the following pieces: Op. 9 nr. 3, Op. 15 nr. 1, Op. 15 nr. 2, Op. 27 nr. 1, Op. 27 nr. 2 and Op. 48 nr. 1. In two cases (Op. 9 nr. 3 and Op. 48 nr. 1), the final passage consists of two clearly separated parts, being both of them performed individually with a ritardando. These ritardandi were treated separately — namely *rit1* and *rit2*. So that, we have 8 different ritardandi for our study.

The data were obtained in a semi-automated manner, using a software tool [10] for automatic transcription of the audio recordings. From these transcriptions, the segments corresponding to the final ritardandi were then extracted and corrected manually by means of *Sonic Visualiser*, a software tool for audio annotation and analysis [1].

The dataset in this paper is a subset of that used in previous work [4], as we are only considering those pianists from whom all eight recordings are available. Table 1 shows the names of these pianists and the year of their recordings. Hence, the dataset for the current study contains a total amount of 136 ritardandi from 17 different pianists.

Arrau (1978)	Falvai (1997)	Pires (1996)
Ashkenazy (1985)	Harasiewicz (1961)	Pollini (2005)
Barenboim (1981)	Hewitt (2003)	Rubinstein (1965)
Biret (1991)	Leonskaja (1992)	Tsong (1978)
d'Ascoli (2005)	Mertanen (2001)	Woodward (2006)
Engerer (1993)	Ohlsson (1979)	

Table 1. Performer and year of the recordings analyzed in the experiments

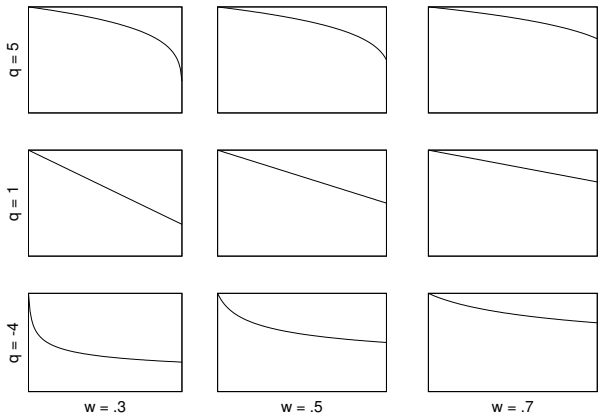


Figure 1. Examples of tempo curves generated by the model using different values of parameters w and q. In each plot, the x and y axis represent score position and tempo respectively, both in arbitrary units.

2.1 Friberg & Sundberg's kinematic model

As mentioned in Section 1, we wish to establish to what degree the specific form of the final ritardando in a musical performance is dependent on the identity of the performing pianist. We address this question by fitting a model to the data, and investigating the relation between the piece/pianist identity and the parameter values of the fitted model. To such a task, we employ the kinematic model by Friberg & Sundberg [3].

This model is based on the hypothesized analogy of musical tempo and physical motion, and is derived from a study of the motion of runners when slowing down. From a variety of decelerations by various runners, the decelerations judged by a jury to be most aesthetically pleasing turned out to be those where the deceleration force is held roughly constant. This observation was implying that velocity was proportional to square root function of time, and to a cubic root function of position. Equating physical position to score position, Friberg and Sundberg used this velocity function as a model for tempo in musical ritardandi. Thus, the model describes the tempo $v(x)$ of a ritardando as a function of score position x:

$$v(x) = (1 + (w^q - 1)x)^{1/q} \quad (1)$$

The parameter q is added to account for variation in curvature, as the function is not necessarily a cubic root of position. The parameter w represents the final tempo, and was added since the tempo in music cannot reach zero. The model can be fitted to ritardandi performed by particular pianists by means of its parameters.

Parameters w and q generate different plots of tempo curves (see Figure 1). Values of $q > 1$ lead to convex tempo curves, whereas values of $q < 1$ lead to concave

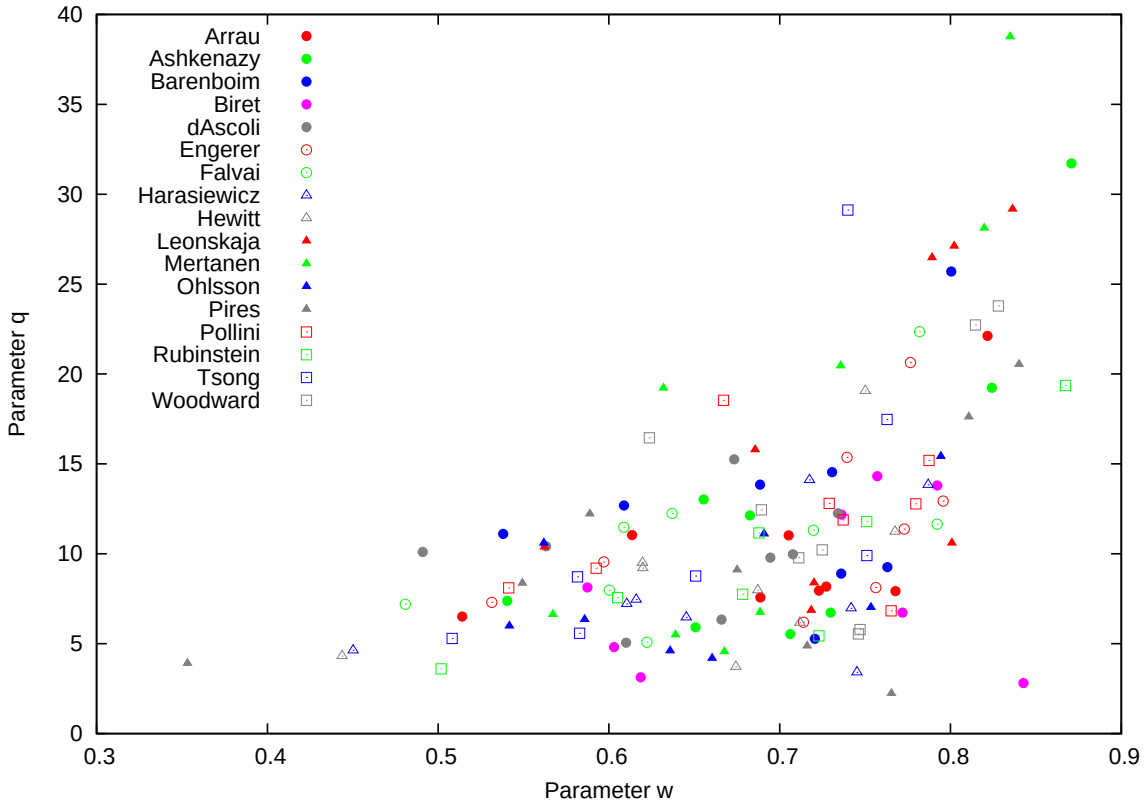


Figure 2. Original data representation in the w-q plane

curves. The parameter w determines the vertical end position of the curve.

Even though this kind of models are incomplete as they ignore several musical characteristics [6], the kinematic model described above was reported to predict the evolution of tempo during the final ritardando quite accurately, when matched to empirical data [3]. An additional advantage of this model is its simplicity, both conceptually (it contains few parameters) and computationally (it is easy to implement).

The model is designed to work with normalized score position and tempo. More specifically, the ritardando is assumed to span the score positions in the range [0,1], and the initial tempo is defined to be 1. Although in most cases there is a ritardando instruction written in the score, the ritardando may start slightly before or after this instruction. When normalizing, we must assure that normalized position 0 coincide with the actual start of the ritardando. A manual inspection of the data showed that the starting position of the ritardandi strongly tended to coincide among pianists. For each piece, the predominant starting position was determined and the normalization of score positions was done accordingly.

The model is fitted to the data by non-linear least-squares fitting through the Levenberg-Marquardt algorithm [1], using the implementation from *gnuplot*. The model fitting is applied to each performance individually, so for each combination of pianist and piece, three values are obtained: w, q and the root mean square of the error after fitting (serving this value as a *goodness-of-fit* measure).

At this point, we can represent each particular ritardando in the corpus as a combination of those two attributes: w and q. In Figure 2 [2], the values obtained from fitting are displayed as a scatter plot on the two-dimensional attribute space q versus w. The whole dataset —136 instances— is shown in this plot. Each point location correspond to a certain curve with parameters w and q. We refer the reader to Figure 1 to visualize the shape of different combination of parameters.

As can be seen from Figure 2, there are no clusters that can be easily identified from this representation. Hence, the performer identification task using these original data is expected to have a low success rate.

3. METHOD

In Section 1, we already mentioned that the expressive timing data is expected (as stated in [4]) to have a strong component that is determined by piece-specific aspects such as rhythmical structure and harmony. In order to focus on pianist-specific aspects of timing, it would be helpful to remove this piece-specific component.

Let X be the set of all instances (i.e. ritardando performances) in our dataset. Each instance $x \in X$ is a duple (w, q). Given a ritardando i, X_i is the subset of X that

[1] The fitting must be done by numerical approximation since the model is non-linear in the parameters w and q

[2] this figure is best viewed in color

contains those instances $x \in X$ corresponding to that particular ritardando.

In order to remove the piece-specific components, we propose to apply a linear transformation to the 2-attribute representation of ritardandi. This transformation consists in calculating the performance norm for a given piece and subtracting it from the actual examples of that piece. To do so, we first group the instances according to the piece they belong. We then calculate the centroid of each group (e.g. mean value between all these instances) and move it to the origin, moving consequently all the instances within that group.

We are aware that modelling the performance norm of a given ritardando as the mean of the performances of that ritardando is not the only option and probably not the best one. In fact, which performance is the best and which one is the most representative is still an open problem with no clear results. Moreover, several performance norms can be equally valid for the same score. In spite of these difficulties, we chose to use the mean to represent the performance norm, for its simplicity and for the lack of an obvious alternative.

Two approaches were then devised in order to calculate that performance norm. In the first one, the mean performance curve is calculate as a unweighted mean of the attributes w and q (see Equation 2); whereas in the second one, *fit* serves to weight the mean (see Equation 3).

In the first approach, the performance norm for a given ritardando i can be calculated as:

$$norm_i = \frac{\sum_{x_i \in X_i} x_i}{|X_i|} \tag{2}$$

In the second approach, it is calculated as a weighted mean, where fit_i stands for the *fit* value of instance x_i:

$$norm_i = \frac{\sum_{x_i \in X_i} x_i fit_i}{\sum fit_i} \tag{3}$$

In either case, all instances x_i are then transformed into x'_i by subtracting the corresponding performance norm:

$$x'_i = x_i - norm_i \tag{4}$$

X' would be then the dataset that contains all x'. After this transformation, all x' contain mainly information about the performer of the ritardando, as we have removed the common component of the performances per piece.

4. EXPERIMENTATION

In order to verify whether pianists have a personal way of playing ritardandi, independent of the piece they play, we have designed a classification experiment with different conditions, in which performers are identified by their ritardandi. The ritardandi are represented by the fitted model parameters. In one condition, the data instances are the set X, i.e. the fitted model parameters are used as such, without modification. In the second and third conditions, the piece-specific component in every performance is subtracted (data set X'). The second condition uses the unweighted average as the performance norm, the third condition uses the weighted average.

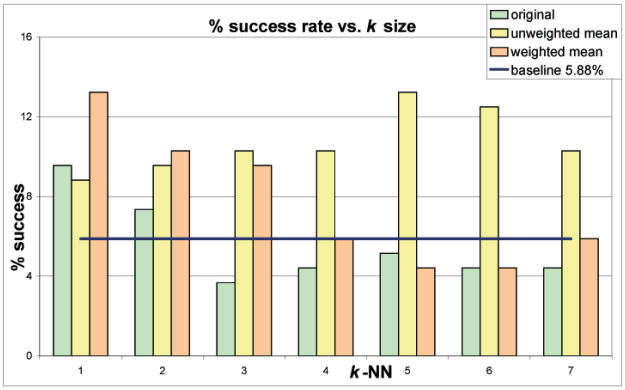


Figure 3. % success rate in the performer identification task using the whole dataset, with different k-NN classifiers. Baseline value (5.88%) from random classification is also shown

Note that accurate performer identification in this setup is unlikely. Firstly the current setting, in which the number of classes (17) is much higher than the number of instances per class (8), is rather austere as a classification problem. Secondly, the representation of the performer's rubato by a model with two parameters is very constrained, and is unlikely to capture all (if any) of the performer's individual rubato style. Nevertheless, by comparing results between the different conditions, we hope to determine the presence of individual performer style independent of piece.

As previously explained, the training instances (ritardandi of a particular piece performed by a particular pianist) consist of two attributes (w and q) that describe the shape of the ritardando in terms of timing. Those attributes come from matching the original timing data with the kinematic model previously cited.

The pianist classification task is executed as follows. We employ k-NN (Nearest Neighbor) classification, with $k \in \{1, \ldots, 7\}$. The target concept is the pianist in all the cases, and two attributes (w and q) are used. For validation, we employ leave-one-out cross-validation over a dataset of 136 instances (see Section 2). The experiments are carried out by using the Weka framework [5].

Figure 3 shows the results for the previously described setups, employing a range of k-NN classifiers with different values of $k \in \{1, \ldots, 7\}$. We also carry out the classification task using the original data (without the transformation) that were shown in Figure 2, in order to compare the effect of the transformation.

The first conclusion we can extract from the results is that the success rate is practically always better when transforming the data than when not. In other words, by removing the (predominant) piece-specific component, it gets easier to recognize performers. This is particularly interesting as it provides evidence for the existence of a performer-specific style of playing ritardandi, which was our initial

hypothesis.

Note however, that the success rate is not so good to allow this representation for being a suitable estimation of the performer of a piece, even in the best case. A model with only two parameters cannot comprise the complexity of a performer expressive fingerprint. Although improving performer identification is an interesting problem, that is not the point of this work.

As can be seen, employing a weighted mean of w and q for calculating the performance norm of a piece —being *fit* the weight— leads to better results when k is small (i.e. $k < 3$). However, this approach, which is methodologically the most valid, does not make a remarkable difference with respect to the original data for larger values of k.

An interesting and unexpected result is that the transformation with the unweighted mean (see equation 2), gives better results for medium-large k values. The lower results for smaller k could be explained by the fact that instances with a low fit (which are actually noisy data), interfere with the nearest-neighbor classification process. The better results for higher k suggest that in the wider neighborhood of the instance to be classified, the instances of the correct target dominate —and thus that the noise due to low fit is only limited.

Note also that this approach is more stable with respect to the size of k than the original or the weighted ones. It also outperforms the random classification baseline —that is 5.88% with 17 classes— for all the different values of k.

Further experiments show that those are the trends for those two different transformation of the data. Employing the weighted mean leads to the highest accuracy using a 1-NN classifier, but it quickly degrades as k is increased. On the other hand, an unweighted mean leads to more stable results, with the maximum reached with an intermediate number of neighbors.

Although (as expected with many classes, few instances and a simplistic model) the classification results are not satisfactory from the perspective of performer identification, the improvement that transforming the data (by removing piece-specific aspects) gives in classification results, suggests that there is a performer-specific aspect of rubato timing. Even more, it can be located specifically in the curvature and depth of the rubato (w and q parameters).

5. CONCLUSIONS AND FUTURE WORK

Ritardandi in musical performances are good examples of the expressive interpretation of the score by the pianist. However, in addition to personal style, ritardando performances tend to be substantially determined by the musical context they appeared in. Because of this fact, we propose in this paper a procedure for canceling these piece-specific aspects and focus on the personal style of pianists.

To do so, we make use of collected timing variations during ritardando in the performances of Chopin Nocturnes by famous pianists. We obtain a two-attributes (w,q) representation of each ritardando, by fitting Friberg and Sundberg's kinematic model to the data.

A performer identification task was carried out using k-Nearest Neighbor classification on, comparing the (w,q) representation to another condition in which average w and q values per piece are subtracted from each (w,q) pair.

The results indicate that in even in this reduced representation of ritardandi, pianists can often be identified by the tempo curve of the ritardandi above baseline accuracy. More importantly, removing the piece-specific component in the w and q values leads to better performer identification.

This suggests that even very global features of ritardandi, such as its depth (w) and curvature (q), carry some performer-specific information. We expect that a more detailed representation of the timing variation of ritardandi performances will reveal more of the individual style of pianists.

A more detailed analysis of the results is necessary to answer further questions. For instance, do all pianists have a quantifiable individual style or only some? Also, there is a need for alternative models of rubato (such as the model proposed by Repp [11]), to represent and study ritardandi in more detail.

Finally, we intend to relate our empirical findings with the musicological issue of the factors affecting music performances. Experiments supporting whether or not the structure of the piece and the feelings of the performer are present in renditions could be of interest to musicologists.

6. ACKNOWLEDGMENTS

This research is supported by the Austrian Research Fund FWF under grants P19349 and Z159 ('Wittgenstein Award'). M. Molina-Solana is supported by the Spanish Ministry of Education (FPU grant AP2007-02119).

7. REFERENCES

[1] Chris Cannam, Christian Landone, Mark Sandler, and Juan Pablo Bello. The sonic visualiser: A visualisation platform for semantic descriptors from musical signals. In *Proc. Seventh International Conference on Music Information Retrieval (ISMIR 2006)*, Victoria, Canada, October 8-12 2006.

[2] Anders Friberg. Generative rules for music performance: A formal description of a rule system. *Computer Music Journal*, 15(2):56–71, 1991.

[3] Anders Friberg and Johan Sundberg. Does musical performance allude to locomotion? A model of final ritardandi derived from measurements of stopping runners. *Journal of the Acoustical Society of America*, 105(3):1469–1484, 1999.

[4] Maarten Grachten and Gerhard Widmer. The kinematic rubato model as a means of studying final ritards across pieces and pianists. In *Proc. Sixth Sound and Music Computing Conference (SMC 2009)*, pages 173–178, Porto, Portugal, July 23-25 2009.

[5] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[6] Henkjan Honing. When a good fit is not good enough: a case study on the final ritard. In *Proc. Eighth International Conference on Music Perception & Cognition (ICMPC8)*, pages 510–513, Evanston, IL, USA, August 3-7 2004.

[7] Erik Lindström, Patrik N. Juslin, Roberto Bresin, and Aaron Williamon. "expressivity comes from within your soul": A questionnaire study of music students' perspectives on expressivity. *Research Studies in Music Education*, 20:23–47, 2003.

[8] Miguel Molina-Solana, Josep Lluis Arcos, and Emilia Gomez. Using expressive trends for identifying violin performers. In *Proc. Ninth Int. Conf. on Music Information Retrieval (ISMIR2008)*, pages 495–500, 2008.

[9] Miguel Molina-Solana and Maarten Grachten. Nature versus culture in ritardando performances. In *Proc. Sixth Conference on Interdisciplinary Musicology (CIM10)*, Sheffield, United Kingdom, July 23-24 2010.

[10] Bernhard Niedermayer. Non-negative matrix division for the automatic transcription of polyphonic music. In *Proc. Ninth International Conference on Music Information Retrieval (ISMIR 2008)*, Philadelphia, USA, September 14-18 2008.

[11] Bruno H. Repp. Diversity and commonality in music performance - An analysis of timing microstructure in Schumann's "Träumerei". *Journal of the Acoustical Society of America*, 92(5):2546–2568, 1992.

[12] John Rink, editor. *The Practice of Performance: Studies in Musical Interpretation*. Cambridge University Press, 1996.

[13] Efstathios Stamatatos and Gerhard Widmer. Automatic identification of music performers with learning ensembles. *Artificial Intelligence*, 165(1):37–56, 2005.

[14] Johan Sundberg and Violet Verrillo. On the anatomy of the retard: A study of timing in music. *Journal of the Acoustical Society of America*, 68(3):772–779, 1980.

[15] Renee Timmers, Richard Ashley, Peter Desain, Henkjan Honing, and W. Luke Windsor. Timing of ornaments in the theme of Beethoven's Paisiello Variations: Empirical data and a model. *Music Perception*, 20(1):3–33, 2002.

[16] Neil P. Todd. A computational model of rubato. *Contemporary Music Review*, 3(1):69–88, 1989.

[17] Gerhard Widmer, Sebastian Flossmann, and Maarten Grachten. YQX plays Chopin. *AI Magazine*, 30(3):35–48, 2009.

[18] W. Luke Windsor and E.F. Clarke. Expressive timing and dynamics in real and artificial musical performances: Using and algorithm as an analytical tool. *Music Perception*, 15(2):127–152, 1997.

FAST vs SLOW: LEARNING TEMPO OCTAVES FROM USER DATA

Jason A. Hockman and Ichiro Fujinaga
Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT)
Distributed Digital Music Archives and Libraries (DDMAL)
Music Technology Area, Schulich School of Music
McGill University, Canada
jason.hockman@mail.mcgill.ca, ich@music.mcgill.ca

ABSTRACT

The widespread use of beat- and tempo-tracking methods in music information retrieval tasks has been marginalized due to undesirable sporadic results from these algorithms. While sensorimotor and listening studies have demonstrated the subjectivity and variability inherent to human performance of this task, MIR applications such as recommendation require more reliable output than available from present tempo estimation models. In this paper, we present a initial investigation of tempo assessment based on the simple classification of whether the music is *fast* or *slow*. Through three experiments, we provide performance results of our method across two datasets, and demonstrate its usefulness in the pursuit of a reliable global tempo estimation.

1. INTRODUCTION

Within the last ten years, beat tracking and tempo induction methods have been significantly improved. Several state-of-the-art methods [1–3] are now capable of identifying and providing reliable beat calculations through difficult passages marked by features such as expressive timing or competing rhythms. However, the usefulness of such methods for information retrieval tasks has been limited due to the unpredictable behavior of these algorithms. While many studies demonstrate musical beat localization for humans to be variable and highly subjective [4–8], MIR applications such as recommendation and harmonic description require more reliable tempo estimates. The most frequent error in this context is the so-called *octave error*, or the halving or doubling of the perceived tempo caused by attributing the driving beat level to a metrical level other than the most predominant pulse.

Identification of the most appropriate tempo octave has been shown to be a difficult problem, as demonstrated in the discrepancy between beat tracking evaluations in which a single tempo octave and multiple tempo octaves are accepted [2, 3, 9, 16]. As metronomic values are not absolute, they are not well-suited for defining the perceived relative speed of a piece of music. Unfortunately, if a user of a recommendation system were to request slow music labeled 60 BPM, and received music more commonly associated with 120 BPM, they would not be satisfied! This paper presents a novel approach to this problem, by identifying *fast* or *slow* music without the use of a beat tracker, and demonstrates the usefulness of this categorization in selecting the appropriate tempo octave of a given piece of music.

1.1 Background

The selection of tempo octave is most commonly achieved as an embedded step within the framework of the beat- or tempo-tracking task. The general procedure used in most audio tempo-tracking algorithms is comprised of three steps. First, the audio signal undergoes a process of reduction, which simplifies the signal by accentuating prominent signal information such as transients. Second, periodicity analysis is performed on the simplified signal, to extract possible beat periods (i.e., the duration between beat events). Third, the algorithm identifies which period is most likely, and assigns this value as the *tactus*, or the most influential beat, which typically controls the local timing of a musical piece.

The majority of recent efforts in beat tracking have centered on this third step, mostly through attempts to incorporate musical knowledge. Musical knowledge is, in this sense, information of any complexity that is provided to the model that allows it to focus on a particular subset of candidates within the wide variety of possible solutions. This knowledge may take on several forms, from a simple limiting of values to desired candidates, to conditional dependencies between metrical levels and prior decisions. The need for such knowledge comes from the ambiguity faced in analyzing the output of periodicity functions of real signals, which may include intra-measure timing variations (e.g., the swing factor in jazz music), syncopation, and/or global tempo shifts. Inspection of the output of periodicity functions during most musical signals will demonstrate several peaks including both octave-related (e.g., half- or double-time periods) resonances as well as other peaks due to rhythmic complexity and noise; these peaks often overshadow the otherwise steady period.

Therefore, a selection of the tactus based on output energy of a periodicity function alone at each frame will result in a highly unsteady tempo output for many music sources.

To address the tempo octave problem, Goto and Muraoka [10] limit the possible period values to those periods whose tempi are within only one octave.

As an alternative to placing strict boundaries on tempo values, both Ellis [1] and Klapuri et al. [2] weigh the output of their periodicity functions with log-Gaussian distributions originally proposed by Parncutt [6]. The motivation behind this approach is to model tactus preferences exhibited during listening tests [5, 6], and it is intended to provide emphasis to tempi positioned around the mean of the distribution.

Davies and Plumbley [3] use a variable-state method that alternates weighting functions based on the observed variation of the autocorrelation output. The purpose of this method is to model the uncertainty of the listening process upon initial contact with the stimulus, and then to constrain the possible values based on prior observations.

Klapuri et al. [2] use a hidden Markov model to extract the temporal evolution of a hidden metrical sequence exhibited in the output of a comb filterbank. The joint-state estimates of the present tactus, tatum, and meter periods are achieved through a first-order Markov process, in which the present filterbank output and transition probabilities between periods are used to generate a probabilistic determination of the present state. Selection of bar-length periodicities and tatum help to reduce incorrect tactus attribution. The strength of this model lies in its ability to reinforce a metrical framework within sections displaying less prevalent metrical observations.

In a method conceptually similar to our own, Xiao et al. [11] use a Gaussian mixture model to capture the timbral characteristics of a given tempo through the association of Mel-frequency cepstral coefficients (MFCCs) to discrete BPM values. While this method was demonstrated to reduce the occurrence of octave errors for the beat tracker presented in [1], its reliance on a discrete BPM values as class labels requires a large amount of ground truth that is difficult to produce due to human subjectivity during data collection.

1.2 Motivation

With the exception of [11], the above methods rely on some form of limiting or weighting curve applied to the output of the periodicity function (e.g., autocorrelation and comb filterbank) to reduce the effects of alternate tempo octaves, but these curves are based on BPM responses which are highly variable due to the subjectivity of the task.

What can actually be inferred about a piece of music from a BPM value? Given that humans choose different levels at which to tap when synchronizing with music, is it plausible that a BPM measure would provide us with information about the speed of a piece? Certainly within a single tempo octave the BPM scale can be very informative, but the plurality of acceptable BPM values across tempo octaves makes an inter-octave comparison of musical rates less reliable.

In addition, other than [11], all above methods rely exclusively on periodicity functions and relatively few features for determination of BPM and thus tempo octave. Our method relies instead on the assumption that the difference between fast and slow music manifests itself across multiple features.

1.3 Organization of this paper

Section 2 briefly outlines our technique for the determination of a piece of music as fast or slow. Section 3 presents both experimentation and results for our method, as well as the application of our method to tempo-tracking. Section 4 presents discussion, and Section 5 provides conclusions and future work.

2. METHOD

To address the problem of tempo octave estimation, we present a classification-based approach that does not rely on discrete BPM values. Alternatively, the proposed method performs a binary classification using broad categories of human response to the pace of music: *fast* and *slow*. There are several benefits to the proposed classification scheme. Unlike solving for a discrete BPM value, music classification as fast or slow is a binary classification problem that offers higher accuracy than present multi-class solutions (e.g., discrete BPM values). Evaluation methodology and interpretation is greatly simplified without acceptance of multiple metrical levels. In addition, ground truth—in this case class labels created through listener response to music—is more readily available for this particular problem.

The proposed technique has two immediate applications: first, as a feature within another retrieval task, and second, as a component within a tempo-tracker that guides the algorithm to the more appropriate of two tempo ranges. While the taxonomy of fast or slow is not precisely analogous to a specific BPM range, we propose that the tempo range can roughly be divided in half to accommodate two tempo octaves. With a training set approximately covering several musical styles in both fast and slow categories, a mapping may be achieved between these two taxonomies. Our assumption is that labelling a song as slow is indicative of the existence of prevalent acoustic characteristics that have led to a selection of the lower tempo octave, while a classification of fast is indicative of features that prompted a rate of synchronization within the faster tempo octave.

2.1 Data collection

To generate our datasets, we created a data harvester [1] built on the Last.fm and YouTube APIs. Our initial intention was to extract features and train our classifiers based on audio for songs that were relevant to the fast and slow tags on Last.fm. Because audio content is for the most part not available on Last.fm, we opted instead to generate a list of artist and track names associated with either fast or slow

[1] available at: http://www.music.mcgill.ca/∼hockman/other/mashup

tags, and use each artist-track combination in this list as search terms for videos on YouTube.

An initial list of artist and track names was created by mining Last.fm for the most popular tracks related to the query tags. Additional tracks were then appended to this list through a search for similar tracks that also displayed these tags. If the video matching the query was available, an audio track was automatically extracted from the video. Each file was then manually verified to be a version of the artist-track combination. The specific size and makeup of the dataset varied with the experiment being performed (as explained in Section 3).

2.2 Feature extraction

The success of our classification relies chiefly on our feature set, which has been generated using jAudio [12], a Java-based feature extraction tool from the jMIR software package [13]. [2]

Each of the tempo estimation methods discussed in Section 1.1 generates an *onset detection function* (also known as a *driving signal*) by analyzing either a single feature or relatively few features, and tracks these over the course of overlapping windows; the aim being to highlight significant local signal characteristics, such as fast attack transients, while attenuating steady-state components.

Alternatively, our approach uses a significantly larger feature set, and characterizes features across entire tracks. We suspect that the perception of acoustic cues differs for songs heard as fast and slow, and that these cues are related to pitch, loudness, and timbre. We therefore extract a large number of features in hopes of exploiting regularities within these three musical attributes. Each audio track is first converted into a normalized 8 kHz single-channel .wav file. For each audio file, we assess over 80 overall features, including spectral centroid, rolloff, flux, variability, peak-based spectral smoothness, zero crossings, MFCCs, LPC, and Method of Moments, along with the aggregates [14] of several of these features, e.g., derivative, running mean, and standard deviation.

2.3 Classification

Classification is performed using jMIR's Autonomous Classification Engine (ACE) software [15]. Provided feature vectors as created in Section 2.2 and a classifications file containing a list of labels directly from user data corresponding to each audio track as in Section 2.1, ACE performs classification with a variety of machine learning classification algorithms. Our experiments focused on the following six classifiers available in ACE:

- Unweighted k-Nearest Neighbor, with k=1 (k-NN)
- Support Vector Machines (SVM)
- Naive Bayes
- C4.5 Decision Trees (C4.5)
- AdaBoost seeded with C4.5 (AdaBoost)
- Bagging seeded with C4.5 (Bagging)

[2] available at: http://jmir.sourceforge.net

3. EXPERIMENTS

The goal for our experiments was to measure how well the above machine learning algorithms can identify fast and slow songs. To evaluate our method, we compared the output of several classifiers tested on two separate datasets. In all, we conducted three experiments: the first two deal specifically with identifying the best classification algorithm for determining fast or slow tempo, and the third compares our method against an existing tempo-tracking algorithm modified to output fast or slow values.

3.1 Experiment 1: Fast vs. slow

For the first of these experiments, we tested the feasibility of our approach using a dataset comprised of audio that users of Last.fm have tagged as fast or slow. The dataset was constructed as explained in Section 2.1, using search terms restricted to *fast* and *slow*. The total size of this dataset was 397 full-length audio tracks, comprised of 109 fast songs and 288 slow songs. Features were extracted as described in Section 2.2. Success rates are based on averages of five runs of three-fold cross-validation performed on the dataset with each classifier. Overall averages are displayed in Table 1.

Classifier	Avg. Success
k-NN (k=1)	97.48
SVM	99.37
Naive Bayes	98.24
C4.5	99.18
AdaBoost w/ C4.5	**99.44**
Bagging w/ C4.5	99.12

Table 1. 3-fold cross-validation results for Experiment 1. Values are presented in percentages for k-NN, SVM, Naive Bayes, C4.5, AdaBoost, and Bagging classifiers.

The best performing classifier was AdaBoost, closely followed by SVMs, C4.5, and Bagging. From the high success rates of these learners, we may infer the effectiveness of training exclusively with global features, as well as the lack of need for a periodicity function.

We can identify two weaknesses in our approach for this experiment, both related to genre. First, we did not attempt to control the influence of genre across tempo classes; it is plausible that relatively few genres comprise a large portion of the dataset, ultimately simplifying the classification task to one of basic genre classification (e.g., ambient vs. punk). Without genre labels we cannot reliably isolate the effect of genre from the determination of fast or slow music within our dataset.

Second, the fast and slow tags may have been made with respect to genre, and we cannot assume the motivation behind the use of these tags. While one listener might use these tags to describe the pace of a piece in relation to other music of many genres, others might use the same tags to describe its pace in relation to a specific genre. This could potentially be an issue if the two tag meanings were not

consistent. For example, a slower Drum and Bass track could conceivably be tagged as slow within the genre, or fast in comparison with other genres.

3.2 Experiment 2: Intra-genre fast vs. slow

Following the results of our previous experiment in Section 3.1, we designed an experiment to ensure that the classifiers were not simply classifying genres. For this experiment, a new dataset was created. An ideal dataset would have comprised of fast and slow versions of each song, eliminating any differences cause by genre that were not related to tempo. As we neither have such music, nor tags to describe it as fast or slow, we instead used our data harvester to find fast and slow music within each genre. For search tags we first looked for tempo-genre pairs in the form of fast x and slow x, where x is a genre taken from a list of over 1500 genres. [3] For a tempo-genre tag pair to be considered as search terms, each tag was required to return a tracklist result with no less than five audio tracks for each genre. Once the list of tracks was established, they were downloaded as in the first experiment.

For this particular search, we found the distribution of tracklist results between fast and slow genres highly unbalanced. Many of the returned tempo-genre pairs (fast x and slow x) had a large number of files in one category and close to the minimum in the other. We therefore selected the five most evenly distributed genres (Country, Jazz, Rap, R&B, and Rock). Our desired dataset was comprised of at least thirty tracks in each tempo-genre class. As the number of tracks retrieved in each category did not meet our expectations, we decided to increase the size of the dataset by mining YouTube directly using the tempo-genre terms as queries for playlists. Our final dataset for this experiment was comprised of 831 verified full-length audio tracks, as shown in Table 2, and the complete list of the songs is available online. [4]

	Country	Jazz	Rap	RnB	Rock	Totals
Fast	33	112	63	76	111	395
Slow	66	103	78	120	69	436
Totals	99	215	141	196	180	831

Table 2. Dataset 2 breakdown by genre and tempo class.

We then tested our classification method within each of the five genres using three-fold cross-validation, as in the previous experiment. Results in Table 3 demonstrate the capability of each of the five classifiers in this task. Even the worst performer, the naive Bayesian classifier, scored above 93%. The top performers for each of the genres were either C4.5 or AdaBoost seeded with C4.5. The best classifier across all genres was again AdaBoost seeded with C4.5, and the most difficult genre tested across each classifier was Rap.

Next, as in Section 3.1 we evaluated each classifier's ability to determine fast or slow across the entire dataset, without genre separation. Results for this test are presented in Table 4. The top performing classifier was AdaBoost, and success rates were only minimally affected by the absence of genre specification. We can therefore conclude that the classifiers were able to learn fast and slow characteristics of music without prior knowledge of musical genre.

Genre	k-NN	SVM	Naive	C4.5	Ada	Bag
Cntry	94.83	97.26	92.51	**98.48**	97.95	97.46
Jazz	95.81	98.49	92.78	98.01	**99.30**	99.07
Rap	90.28	96.98	93.10	98.24	**99.29**	99.11
R&B	89.04	95.16	93.98	**98.47**	98.21	98.08
Rock	92.92	95.71	93.32	99.17	**99.28**	97.93
Avg.	92.58	96.72	93.14	98.47	**98.80**	98.33

Table 3. 3-fold cross-validation results for intra-genre tests in Experiment 2. Values are presented in percentages for k-NN, SVM, Naive Bayes (Naive), C4.5, AdaBoost (Ada), and Bagging (Bag) for each genre: Country (Cntry), Jazz, Rap, R&B, and Rock.

Classifier	Avg. Success
k-NN (k=1)	95.97
SVM	96.42
Naive Bayes	90.94
C4.5	95.10
AdaBoost w/ C4.5	**96.81**
Bagging w/ C4.5	96.45

Table 4. 3-fold cross-validation results (in percentages) for six classifiers tested across entire dataset (i.e., without genre separation) in Experiment 2.

3.3 Experiment 3: Applications in tempo-tracking

A third experiment was undertaken to compare the presented method to another method capable of fast and slow determination. This comparison was achieved using the results of the top performing classifier from Section 3.2 and the binarized output of a beat tracker [16] modified to provide a single tempo for each track in the second dataset. For each song n, the beat tracker calculates the derivative Δ of beats θ_n and outputs a single BPM value Γ_n as:

$$\Gamma_n = 60/median(\Delta\theta_n). \quad (1)$$

An obstacle in the comparison between the two approaches is the selection of a boundary λ between fast and slow BPM values output by the tempo tracker. A plausible approach to scoring the output would be to identify a mean tempo for the dataset. However, as we lack ground truth BPM values for this dataset, we were unable to generate an average tempo at which to divide the tempo range. We therefore instead tested a set of integer tempo values $\{50, ..., 150\}$ for λ, defining the optimal divisor as the tempo that provided the best results for the tempo tracker.

[3] http://en.wikipedia.org/wiki/List_of_music_genres

[4] http://www.music.mcgill.ca/~hockman/projects/fastSlow/dataset.zip

Table 5 shows the results of this experiment, with the best performing divisor between fast and slow, $\lambda = 93$ BPM.

Method	Success Rate
Classification (AdaBoost)	**96.81**
Tempo tracking, $\lambda = 93$ BPM	61.85

Table 5. Results for Experiment 3 (in percentages). Results for the classifier (AdaBoost) were generated using 3-fold cross-validation. Tempo tracker output was binarized using λ = 93 BPM as a tempo range divisor.

The discrepancy between results of the two approaches led us to attempt to improve the tempo tracker output using a genre-specific average tempo for each song in the dataset, as we felt that using fixed BPM value λ was unfairly scoring the tempo tracker. For these values, we used average genre tempi calculated from the BPM List [5], a hand-annotated database of 20,000 BPM labels for popular Western music listed by genre. Unfortunately, decomposition by genre did not improve results.

The success rates for the tempo tracker in this experiment should *not* be taken to be indicative of the algorithm's overall performance, as the intention of the tracker is not to define musical pace as either fast or slow, but rather to replicate the perceptual phenomenon of synchronization with a heard piece of music.

4. DISCUSSION

Through the three experiments performed in Section 3, classification of songs as either fast or slow has been shown to be a robust method of determining the overall pace of music. We have achieved above 96% accuracy for two separate datasets and demonstrated its effectiveness in this task over another existing methodology. The high success rate of the presented method suggests its reliability as an independent feature within several MIR tasks. In addition to using classification labels as features themselves, the method could also be used to improve lower-level metrical analysis such as tempo-tracking algorithms by selectively correcting misclassified tempo-tracking octave errors by simply using the classification results.

Our method differs considerably from existing approaches to the problem of tempo octave selection. First, we are currently using only two classes of possible output, as opposed to discretized BPM values. To achieve these class labels, we use machine learning algorithms trained on global features, calculated by aggregating windowed features for each training instance. In addition, we are using a large number of such features to describe each audio track in our dataset. A key difference that sets our method apart from all existing methods is that no periodicity calculation is attempted; we instead rely only on global features and statistics.

[5] http://www.bpmlist.com/

The two datasets used in the course of this study were created through the use of Last.fm and YouTube APIs, and were specifically created based on listener responses to audio. The composition of generated datasets is essential to the training of our classifiers, as the contents will define the ability of our classifiers to differentiate between the two classes. In review of our first experiment, we were concerned that our classification results were artificially high because our first dataset was constructed by downloading tracks associated with fast and slow tags, and that tracks associated with these tags were possibly leading to a division based on musical genre. We therefore constructed a second dataset for the following experiment, which contained examples of fast and slow music within each genre, reducing the effect of musical genre separation. Results of this experiment demonstrated that the classification approach could not only separate fast and slow music within each genre, but within the entire dataset as well.

A weakness of this approach lies in the ambiguity of responses to particular pieces of music. For example, songs in certain genres, such as Hip Hop, intentionally juxtapose a fast lyrical layer with slower percussion and bass loops (e.g., Bone Thugs'n'Harmony, Twista). In these scenarios, a number of listeners tagged some of these songs as fast, possibly referring to the unusually fast rate of lyrics, while other listeners tagged tracks in the same style as slow, possibly focusing on those characteristics that define the genre standards—namely the percussion and bass lines.

A second issue is the variable number of annotations per training file. On Last.fm, more popular songs are likely to have more instances of listeners using fast or slow tags, and thus improving tag reliability. In the present study, we have combined user data from Last.fm with playlist results from YouTube without regard to the number of listeners agreeing with each tag. While this did not cause difficulty for our experiments, perhaps an optimal method might be to directly label more music with Last.fm tags or even to perform structured listening tests.

5. CONCLUSIONS

We believe estimation of tempo octaves within music to be a perceptual phenomenon that can be learned through use of the presented classification model. In this paper we have outlined the training of such a model using a large number of global features related to the overall pitch, timbre, and loudness of an audio track. Through the use of the proposed fast or slow classification, we believe that it is possible to improve the usefulness of tempo-tracking models within applications requiring a reliable single tempo value.

In our future work, we would like to perform further evaluation of our method with several datasets of varied content. Specifically, we would like to test our method using an artificial dataset containing fast and slow versions of songs with the exact same spectral content. Such a dataset could be created through the use of any commercial

sequencer using MIDI files to control synthesizer and sampler output. Evaluation on significantly larger datasets would also be of interest. A difficulty here might lie in the collection of ground truth for training. Towards this end, listening tests may be useful as an alternative source.

We also plan to investigate the applicability of the proposed method in the task of beat tracking. An obstacle in this area is that the proposed method defines *entire* songs. As we cannot assume that segments of the audio contain acoustic features that motivated the class labels (i.e., fast or slow) of the entire file, each segment would need to be classified independently, which would require manually labeled segments for training. Informal tests, however, suggest only a slight decrease in performance with audio segments of shorter durations, e.g., 10 seconds.

Finally, we intend to explore alternative strategies for incorporating our approach into tempo- and beat-tracking methods towards improved performance of these algorithms.

6. ACKNOWLEDGEMENT

The authors would like to thank the Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT) for their generous support, M. Davies for providing source code for his algorithm, and C. McKay, A. Hankinson, and J. Thompson for their technical assistance.

7. REFERENCES

[1] D. Ellis: "Beat tracking with dynamic programming," http://www.music-ir.org/mirex/2006/mirex/abstracts/2006/TE_BT_ellis.pdf (accessed March 1, 2009), 2006.

[2] A. Klapuri, A. Eronen, and J. Astola: "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Speech and Audio Processing*, Vol. 14, No. 1, pp. 342–355, 2006.

[3] M. Davies and M. Plumbley: "Context-dependent beat tracking of musical audio," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 15, No. 3, pp. 1009–1020, 2007.

[4] S. Dixon and W. Goebl: "Pinpointing the beat: Tapping to expressive performances," *Proceedings of the 7th International Conference on Music Perception and Cognition*, pp. 617–620, 2002.

[5] P. Fraisse: "Rhythm and tempo," in *The Psychology of Music*, ed. D. Deutsch, Academic Press, Orlando, Florida, pp. 649–680, 1982.

[6] R. Parncutt: "A perceptual model of pulse salience and metrical accent in musical rhythms," *Music Perception*, Vol. 11, No. 4, pp. 409–464, 1994.

[7] J. Snyder and C. Krumhansl: "Tapping to ragtime: Cues to pulse finding," *Music Perception*, Vol. 18, No. 4, pp. 455–489, 2001.

[8] P. Toiviainen and J. Snyder: "Tapping to Bach: Resonance-based modeling of pulse," *Music Perception*, Vol. 21, No. 1, pp. 43–80, 2003.

[9] M. McKinney, D. Moelants, M. Davies, and A. Klapuri: "Evaluation of Audio Beat Tracking and Music Tempo Algorithms," *Journal of New Music Research*, Vol. 36, No. 1, pp. 1–16, 2007.

[10] M. Goto and Y. Muraoka: "A real-time beat tracking system for audio signals," *Proceedings of the 1995 International Computer Music Conference*, pp. 171–174, 1995.

[11] L. Xiao, A. Tian, W. Li, and J. Zhou: "Using a statistical model to capture the association between timbre and perceived tempo," *Proceedings of the 9th International Conference on Music Information Retrieval*, pp. 659–669, 2006.

[12] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle: "jAudio: A feature extraction library," *Proceedings of the 6th International Conference on Music Information Retrieval*, pp. 600–603, 2005.

[13] C. McKay and I. Fujinaga: "jMIR: Tools for automatic music classification," *Proceedings of the 6th International Society for Music Information Retrieval Conference*, pp. 65–68, 2009.

[14] D. McEnnis, C. McKay, and I. Fujinaga: "jAudio: Additions and improvements," *Proceedings of the 7th International Conference on Music Information Retrieval*, pp. 385–386, 2009.

[15] C. McKay, R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga: "ACE: A framework for optimizing music classification," *Proceedings of the 6th International Conference on Music Information Retrieval*, pp. 42–49, 2005.

[16] M. Davies, N. Degara, and M. Plumbley: "Evaluation methods for musical audio beat tracking algorithms," Technical Report C4DM-TR-09-06, Queen Mary University of London, Centre for Digital Music, 2009.

GEOSHUFFLE: LOCATION-AWARE, CONTENT-BASED MUSIC BROWSING USING SELF-ORGANIZING TAG CLOUDS

Scott Miller, Paul Reimer
University of Victoria
Electrical and Computer Engineering
smiller@ece.uvic.ca

Steven Ness, George Tzanetakis
University of Victoria
Computer Science
gtzan@cs.uvic.ca

ABSTRACT

In the past few years the computational capabilities of mobile phones have been constantly increasing. Frequently these smartphones are also used as portable music players. In this paper we describe GeoShuffle – a prototype system for content-based music browsing and exploration that targets such devices. One of the most interesting aspects of these portable devices is the inclusion of positioning capabilities based on GPS. GeoShuffle adds location-based and time-based context to a user's listening preferences. Playlists are dynamically generated based on the location of the user, path and historical preferences.

Browsing large music collections having thousands of tracks is challenging. The most common method of interaction is using long lists of textual metadata such as artist name or genre. Current smartphones are characterized by small screen real-estate which limits the amount of textual information that can be displayed. We propose self-organizing tag clouds, a 2D tag cloud representation that is based on an underlying self-organizing map calculated using automatically extracted audio features. To evalute the system the Magnatagatune database is utilized. The evaluation indicates that location and time context can improve the quality of music recommendation and that self-organizing tag clouds provide faster browsing and are more engaging than text-based tag clouds.

1. INTRODUCTION

Portable mobile phones with strong multimedia capabilities and computational power are rapidly gaining popularity. As these devices frequently also function as portable digital music players it is important to investigate how music information retrieval systems can be adapted to the unique challenges and opportunities they present. In this paper we describe GeoShuffle a music browsing application designed to address the challenge of limited screen real estate and to take advantage of the opportunity of location information that smart phones provide.

Automatic music recommendation is an active topic of research. Such systems can be based on collaborative filtering, expert annotations, folksonomies, automatic content analysis and any of their combinations. However, all these approaches suffer from the limitation that their results are the same irrespective of the listening context. The preferences of a listener change depending on where they are and what they are doing. For example the music a student would like recommended when studying might be different from the music desired when riding the bus.

Location-aware devices based on technologies such as GPS are common. We propose that the quality of automatically generated playlists can be improved by taking into account this newly available location data. This information can be used to determine a user's listening habits while in transit to common destinations, as people often have daily routines such as return trips to work, school, social activities, and so on. It provides context to a user's listening preferences beyond general ratings. A user providing a rating to a song does not provide context about the conditions under which a user would enjoys listening to that song. For example, a high-energy song that a user rates highly may never be desired when the user wants to relax.

Another unique characteristic of smart phones is their limited screen real-estate. The size of personal digital audio collections is steadily increasing. Effective interaction with these large audio collections poses significant challenges to traditional user interfaces. Music management software typically allow users to select artist, genres or individual tracks by browsing long sortable lists of text. This mode of interaction, although adequate for small music collections, becomes increasingly problematic as collections become larger especially when screen estate is limited. A variety of alternative ways of browsing music collections have been proposed mostly in academic contexts. They typically rely on a combination of audio signal analysis to automatically extract features followed by visualization techniques to map the feature space to a 2D or 3D representation for browsing and navigation.

Tag clouds provide both an overview of the information space as well as direct search support that is particularly suited for mobile phones with small touch screens. In this paper, we present content-aware self-organizing tag clouds a technique that attempts to support querying, browsing, and summarization using the familiar information model

of a tag cloud while taking into account automatic content analysis information as well as location based information.

2. RELATED WORK

Although there is existing work in location-based applications and automatic/semi-automatic playlist generation there seems to be a lack of published material on location-aware playlist generation. With respect to intelligent playlist creation, Flexer et al. have proposed using audio similarity based on Mel Frequency Cepstrum Coefficients (MFCC) and Gaussian models to create a similarity matrix and select songs that blend from and into a user-selected start and end track in a playlist [1]. Pampalk et al. have proposed using user behaviour based on track skipping to determine what artists, genres, rhythms, etc., the user prefers to pass-over [5]. With respect to location-aware playlist creation most existing work simply associates particular pieces of music with specific locations [7].

The current generation of mobile phones feature decent sized displays that also include touch functionality. Interfaces for managing large audio collections based on long lists of scrollable text are not particularly convenient in such displays. An alternative that has mostly been explored in research literature is the use of content-based visualizations of music collections [4].

Tagging systems allow users to add keywords, or tags, to resources without relying on a controlled vocabulary and have become ubiquitous in web-based systems. Tags are aggregated from many users forming "folksonomies" which, although not as accurate as well-designed ontologies, have the advantage of reflecting how users perceive the data and how their vocabulary and perception evolve over time. Tagging is simple and does not require a lot of thinking. Tags form an essential part of personalized internet radio and music community websites such as Last.fm [1]. Tag clouds are the most common way of visualizing tags. They are two-dimensional stylized visual representations of a list of words where the more prominent words are typically assigned a larger font. They are useful for quickly giving users the gist of a set of words. Tag clouds are in common usage on a number of different social networks such as Flickr [2] but trace their origins back at least 90 years to Soviet Constructivist art [16].

There has been considerable research in recent years into the design, use and effectiveness of tag clouds. A historical look at tag clouds is presented in Viegas and Wattenburg [16], which looks at the development of tag clouds since their inception a decade ago, and speculates about their development in the future. In the paper "Seeing things in clouds" [2], an extensive evaluation of different types of visual features in tag clouds, including font size, font weight, intensity, number of characters and area were investigated. Tag navigation in general has been examined in detail with particular focus on "Last.FM", an online social community for music [10]. A context aware browser for mobile devices that uses tag clouds is presented in Mizzaro et al. [11].

Islands of Music [12] is a a content-based visualization of music collections that uses Self-Organizing Maps (SOM) to generate a two-dimensional representation of a collection of music. MusiCream [8] is an interface that allows users to interact with a music collection using a dynamic visualization interface. MusicRainbow [13] is a similar system that uses web-based labelling and audio similarity to visualize music collections. Examples of visualizations for music discovery in commercial and research systems can be found in the Visualizing Music blog [3].

3. SYSTEM DESCRIPTION

Our proposed system takes as input the user's location, the current playing and associated metadata as well as content-based similarity information between all tracks in a user collection. This information is stored in a database for organization and retrieval. The system processes these inputs to generate location-based information such as common paths and make automatic recommendations based on them. Semantic information related to the generated playlists such as track names, artists, genres, tags, playlists are rendered based on self-organizing tag clouds that are computed based on automatically extracted audio features.

3.1 Location and Path Logging

We introduce the following terms to describe location information: **Paths** consist of a start and end location and a collection of **Path Segments** which consist of a start, end, bearing and segment speed. The **Path Segments** are determined by a list of **Location Points** which are instantaneous snapshots of what song is playing and where. This includes a track's metadata (artist, album, title, etc.), current coordinate and time, and whether a song started or skipped.

As a user's location or music changes and location points are generated, the system interpolates the user's current line-of-travel in real-time and generates a path segment consisting of a line between start and end coordinates. These path segments are then associated to a path from the start location of the first path segment to the end location of the last path segment. These paths can then be profiled by counting the songs that are played or skipped, the most listened to genres or tempos, etc.; therefore, as the user builds up a path history, it can be used to generate a more accurate representation of the user's listening tastes.

One of the challenges of determining path segments is that location estimates vary in accuracy and are sampled irregularly. In addition a user following the same path in different days (for example taking the bus to school) will not have exactly the same set of location points. Therefore we have developed an algorithm for determining determining path segments from a running list of location points. The basic idea is to first determine the bearing between the first two location points in a path segment. Subsequently

[1] http://www.last.fm

[2] http://www.flickr.com

[3] http://visualizingmusic.com/

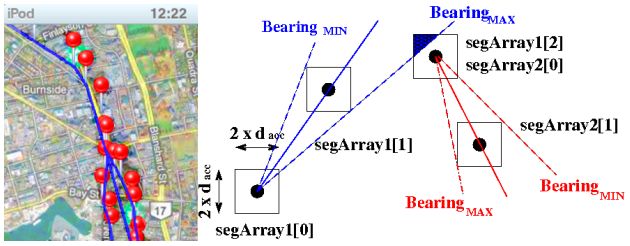


Figure 1. Visualization of paths and location points on a map and schematic of path finding algorithm

the bearing between the start point of the segment and subsequent points is determined. If the new point has the same bearing as the original pair, the new point becomes the end to the segment. This continues until a coordinate yields a bearing of the current segment's path. This basic algorithm works when travelling in very straight lines, and with very accurate positioning hardware, but in real world usage will generate segments between almost every pair of points, as any deviation in bearing will result in a new segment being generated.

In order to account for the accuracy of the positioning system, an algorithm was devised to allow for variation in the absolute location based on the intrinsic accuracy of the mobile device. Each absolute position is reported as a box bounded by the accuracy of the device. Consequently, any points in the bounding box are considered the same absolute coordinate. The same bounding box is used in calculating the bearing for path segments.

These located segments are combined from a start location to an end location in order to generate a path. Figure 1 shows a schematic diagram of the algorithm and a map with paths and location points overlayed. Currently, a path is started when the first change in a user's location is sensed. A path is ended when a user stays at a location for more than 15 minutes. Basic equations for finding distances based on decimal degree coordinates for latitudes and longitudes, and for finding the bearing between two coordinates are based on the WGS84 world representation (currently used by GPS systems).

3.2 Audio Feature Extraction and Recommendations

The goal of audio feature extraction is to represent each track as a vector of features that characterize musical content. First low-level features such as the Spectral Centroid, Rolloff, Flux and the Mel-Frequency Cepstral Coefficients (MFCC) are computed approximately every 20 milliseconds. To capture the feature dynamics we compute a running mean and standard deviation over the past M frames (the so-called "texture window" typically around 1 second). The result is a feature vector of 32 dimensions at the same rate as the original 16D feature vector. The sequence of feature vectors is collapsed into a single feature vector representing the entire audio clip by taking again the mean and standard deviation across the 30 seconds (of the sequence of dynamics features), resulting in the final 64D feature vector per audio clip. A more detailed description of the features and their motivation can be found in Tzanetakis and Cook [15]. For the calculation of the self-organizing map described in the next section all features are normalized so that the minimum of each feature across the music collection is 0 and the maximum value is 1. This feature set has shown state-of-the-art performance in audio retrieval and classification tasks for example in the Music Information Retrieval Evaluation Exchange (MIREX) 2008 and was computed using the free Marsyas audio processing framework [4]. Most audio feature sets proposed exhibit similar performance so we expect that any audio feature front end can be used.

Based on a distance matrix calculated between all pairs of tracks, 3 different recommendation algorithms are implemented. In the naive similarity case, a random seed-song is selected, and playlists of the ten most similar songs (based on pre-calculated Euclidean distances) were created. If the user skipped a song, a new seed is selected and a new playlist is generated along with it. In the similary-with-history case, a profile is constructed based on songs the user listened to at the same time and day of the week to recommend similar songs. A seed song is selected based on tracks that the user enjoyed at similar times (current time $+/-$ an hour) in the past and their three nearest neighbours. If a user skipped a track, a new seed based on their history is selected and a new playlist is generated. Using location information, the system predicted a path that the user is taking and selects a seed from a similar track that was listened to on that path previously. Finally we provide interactive control to the specificity of the generated playlists using the accelerometers included in more mobile devices. Shaking the device at varying levels results in selecting seeds scuh that recommendations are more similar if the shake is light and less similar if it is heavy.

3.3 Self-Organizing Maps

For creating the visualization layout we utilized the self-organizing map (SOM) which is a type of neural network used to map a high dimensional feature space to a lower dimensional representation while preserving the topology of the high dimensional space. This facilitates both similarity quantization and visualization simultaneously. The SOM was first documented in 1982 by T. Kohonen, and since then, it has been applied to a wide variety of diverse clustering tasks [14]. In our system the SOM is used to map the audio features (64-dimensions) corresponding to each track to two discrete coordinates on a grid.

The traditional SOM consists of a 2D grid of neural nodes each containing a n-dimensional vector, $\mathbf{x}(\mathbf{t})$ of data. The goal of learning in the SOM is to cause different neighbouring parts of the network to respond similarly to certain input patterns. The network must be fed a large number of example vectors that represent, as closely as possible, the kinds of vectors expected during mapping. The data associated with each node is initialized to small random values before training. During training, a series of n-dimensional

[4] http://marsyas.info

vectors of sample data are added to the map. The "winning" node of the map known as the *best matching unit* (BMU) is found by computing the distance between the added training vector and each of the nodes in the SOM. This distance is calculated according to some pre-defined distance metric which in our case is the standard Euclidean distance on the normalized feature vectors.

Once the winning node has been defined, it and its surrounding nodes reorganize their vector data to more closely resemble the added training sample. The training utilizes competitive learning. The weights of the BMU and neurons close to it in the SOM lattice are adjusted towards the input vector. The magnitude of the change decreases with time and with distance from the BMU. The time-varying learning rate and neighborhood function allow the SOM to gradually converge and form clusters.

3.4 Self-Organizing Tag Clouds

The technique of self-organizing tag clouds can be viewed as a fusion of concepts from text-based visualization interfaces and more abstract content-aware visualization interfaces. We use the term tag loosely to denote any metadata associated with a track such as genre, artist or year of release. Traditional systems based on long lists of sortable text such as iTunes provide little support for browsing, discovery and summarization. An alternative is visualization interfaces that are based on automatic analysis of musical content. By mapping the music collection onto a 2D or 3D representation they enable quick browsing and navigation especially in the case of music that is not known to the user or that has not been tagged.

Tag-clouds provide a simple, familiar interface that partly overcomes these limitations. For example they support both direct searching as well as browsing and navigation. However they come with their own problems. In order for a tag to assist search or browsing it is necessary for the user to have some notion of its meaning. For example a specialized term such as indie pop might be completely unfamiliar to a particular listener while at the same time essential to another. This problem becomes even more acute using the more generalized notion of tags that includes information such as artist or album. As one of the goals for an effective interface of music collection browsing is the discovery of new music by artists not known to the listener, this is an important disadvantage. Simple tag clouds do not provide the user with any information about the connections and similarity relations between tags. A final problem with any system based solely on tag information is that there is no way to access music tracks that have not been tagged (the so-called "cold start" problem). By contrast content-based visualizations allow any track to be accessed and do not require familiarity with the music explored.

We describe a new method for organizing music tag clouds that makes a persistent map taking into account the musical similarity between songs. Figure 2 shows an example of a self-organized tag cloud. Each label (artist, genre, tag) is associated with a set of tracks that have been annotated with it. As the tracks have been mapped to feature vectors and subsequently to 2D grid coordinates by the SOM, each tag is associated with a set of 2D grid coordinates. The SOM process ensures that neighboring points (tracks) will have similar high-dimensional audio features and therefore similar musical content. The tags are placed on the centroids of their corresponding set of 2D grid coordinates. Their placement reflects the underlying musical content but results in visual overlap between them.

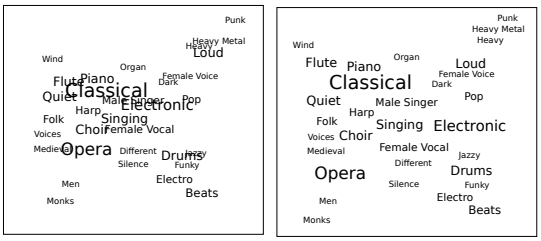


Figure 2. Self-Organizing tag cloud before and after mass-spring layout algorithm

This initial layout contains many overlapping words, so the position of each tag is repositioned using a mass, spring and damper force-based algorithm for drawing [6]. In our implementation each tag is anchored to its original position using a spring and an electrostatic-like force is applied between every pair of tags that is proportional to the inverse of their squared distance. Therefore tags that are close and overlapping will be pushed away while still trying to remain close to their original location. An additional wall force term was added to keep all tags within the designated window. The font size for each tag was determined by counting the number of instances of that tag.

There are some interesting characteristics of the resulting visualization that we would like to highlight. The first is that tags that are not correlated with the acoustical content will correspond to tracks spread across the underlying self-organizing map and therefore their placement will be in the center. For example in Figure 2 the tags Male Singer, Singing and Female Vocal are near the center as they have a large variety of tracks that have been annotated with them. In contrast more specialized tags such as Heavy Metal or Monks are more localized. The second important characteristic is that faceted browsing is naturally supported. For example an artist name, that the user might not be familiar with, located near the left corner will correspond to the tag Monks. Finally a track for which there are no tag annotations will still be placed on the underlying self-organizing map and that way receive an implicit visual automatic tag annotation addressing to some extent the cold-start problem.

3.5 Implementation

The feature extraction, music similarity calculation and self-organizing map training are performed using the Marsyas audio processing framework. Our current prototype application GeoShuffle has been implemented for Apple Inc.'s iPhone or iPod Touch devices. The application dynamically generates music playlists that can be played in the

default iPhone/iPod Touch music player based on location, path of travel, historical information and content similarity. To provide feedback to the user on their preferences by path, as well as to test the accuracy of the application, a Google Map generated map has been embedded into the application (see Figure 1). This map supports annotations in the form of paths or absolute location points. The device's positioning system provides real-time updates on the user's absolute position. This allows the user to visually trace their daily commutes and inspect their musical taste over each path.

4. EVALUATION

Evaluating a complex system and user interface such as the one described in this paper is challenging due to its subjective nature. We focus on two aspects of our work: 1) the use of self-organizing tag clouds as a way to explore large music collections that combines text and content information without requiring large displays 2) the use of location information to improve music recommendation.

For evaluation purposes we used a subset of the Magnatagatune dataset consisting of 1141 tracks with each artist represented by at most 3 tracks. This was chosen as a large enough dataset to have considerable variability while at the same time being manageable in the limited storage of the iPod Touch used for development. There are 341 artists represented and also 14 top-level genre labels. In addition to the regular meta-data information such as artist and genre, also includes tags derived from the Tagatune Game with a purpose [9]. The dataset has been made available to the scientific community for use in research.

For evaluating the self-organizing tag clouds, 14 participants were recruited from graduate Computer Science students. Three were female and 11 were male. All subjects had normal or corrected-to-normal vision, enjoyed listening to music and were experienced computer users. None of the participants had previous knowledge of the Magnatune dataset. The user study consisted of a 5-point system usability survey (SUS) [3].

The survey consisted of six questions, each rated on a five point scale, where "1" was labelled "Strongly disagree" and "5" was labelled "Strongly agree". The 6 questions were: 1) I thought the application was easy to use, 2) I needed to learn a lot before I could accomplish tasks with the application, 3) I think people would need technical support to learn how to use the application, 4) I think most people would learn to use the application very quickly, 5) Overall, accomplishing tasks using the self-organizing tag cloud was easy 6) Overall, accomplishing tasks using the self-organizing tag cloud was fun

Results from survey are detailed in Table 1. On average users rated Question 4 highest, which indicated that they thought most other people would be able to learn the application quickly. This question also had the lowest variance. In Table 1 we detail all the responses from the participants. We can see that two participants chose the middle check box, six chose the next one to the right, and six chose the checkbox labelled "Strongly agree".

Table 1. System Usability Survey

Question	1	2	3	4	5	Mean	Std
1	0	1	3	8	2	3.79	0.8
2	5	7	1	1	0	1.86	0.86
3	5	3	3	1	2	2.43	1.45
4	0	0	2	6	6	4.29	0.73
5	0	2	1	4	7	4.14	1.1
6	0	2	0	6	6	4.14	1.03

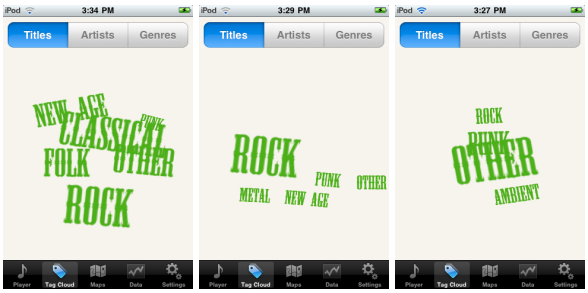


Figure 3. Screen shot of playlist visualization using the Self-Organizing Tag Cloud

In a similar vein, participants also rated questions 5 and 6 highly, although notably, two participants rated this question as one box to the right of "Strongly Disagree". This shows that certain users found our interface easy to use and fit in well with their expectations of an interface to explore music collections, but for other users it did not. For Question 2, the average response was 1.85, which implies that on average, users strongly disagree that they would have to learn a lot before accomplishing tasks with this application. It is important to include negative examples on such a user study to ensure that participants are not just choosing answers to questions randomly; this question performs this control function.

For evaluating the location-aware music recommendation component it was necessary to collect data over an extended period of usage. Usage data was collected from only one subject. The subject used the system over a period of three weeks through their daily routine. GeoShuffle logged their musical preference over the time period and generated sets of user paths (consisting of an origin, destination, and linear path segments). The device switched between four modes of recommendation without the user's knowledge (random, similarity, similarity with history, similarity with location-awareness) and logged which tracks were skipped throughout operation. These results were then used to determine the amount of user skips in each mode of recommendation without biasing the data.

Self-organizing tag clouds can also be used to visualize text information associated with a playlist. Figure 3 shows the self-organizing tag cloud text associated with three playlists (from left to right: random, similarity and path). The figure clearly shows the increase in specificity and the content distribution of the recommended playlists.

Table 2. Number of skips and genres present in playlists created with different generators

	Skips / Track Played	Genres in Playlist
Random	4.3	12
Similarity	1.7	7
+ History	1.2	3
+ Path	0.3	10

Table 2 shows the analysis of skipping behavior between different configurations of the system. We assume that playlists that result in less skipping are better and show the results as average number of skips per track played. The baseline of 4.3 corresponds to randomly selecting songs from the collection in similar fashion to the iPod shuffle. The similarity configuration returns tracks that are similar to all the tracks played in the logging period. The history configuration in addition to similarity takes into account the time of the day. The last configuration also takes into account information about paths taken during the day and is the only one that requires portable devices with location information. As can be seen there is a significant reduction in the number of skips when taking into account location information.

5. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we describe our investigations in designing an interface for content-aware music browsing, discovery and recommendation that is designed based on the unique characteristics of modern smartphones. We propose using location information to improve the quality of music recommendations and introduce self-organizing tag clouds: a visualization of metadata information such as genres, artists, tags and playlists that takes into account automatically extracted musical content information. The specificity of the music recommendation algorithm can be interactively controlled using the accelerometers. The resulting interface is particularly suited for small screen real-estate and touchscreens. Our evaluation indicates that self-organizing tag clouds are an effective and fun way of exploring music collections and that location information can improve the quality of music recommendations.

There are many directions for future work. We plan to explore visualizing tag-based similarities as edges between tags with proportional thickness. Another interesting direction is the addition of social networking and collaboration features such as sharing playlists for particular paths or comparison of collections between different users. Several of the user study participants suggested using the same interface for personalized tag annotation. Finally we plan to conduct a wider ethnographic study where self-organizing tag clouds and location-based recommendation are used in personal music collections.

6. REFERENCES

[1] G. Widmer A. Flexer D. Schnitzer, M. Gasser. Playlist generation using start and end songs. In *Int. Conf. on Music Information Retrieval (ISMIR)*, Philadelphia, USA, 2008.

[2] S. Bateman, C. Gutwin, and M. Nacenta. Seeing things in the clouds: the effect of visual features on tag cloud selections. In *Proc. ACM conf. on Hypertext and Hypermedia*, pages 193–202, 2008.

[3] J. Brooke. Sus: a "quick and dirty" usability scale. In *Usability Evaluation in Industry*. 1996.

[4] M. Cooper, J. Foote, E. Pampalk, and G Tzanetakis. Visualization in audio-based music information retrieval. *Computer Music Journal*, 30(2):42–62, 2006.

[5] G. Widmer E. Pampalk, T. Pohle. Dynamic playlist generation based on skipping behavior. In *The International Society for Music Information Retrieval (ISMIR 2005)*, London, UK, 2005.

[6] J. Ellson, E.R. Gansner, E. Koutsofios, S.C. North, and G. Woodhull. Graphviz - open source graph drawing tools. *Graph Drawing*, pages 483–484, 2001.

[7] K. Eustice and et al. The smart party: A personalized location-aware multimedia experience. In *Consumer Communications and Networking Conference (CCNC 2008)*, Las Vegas, USA, 2008.

[8] M. Goto and T. Goto. Musicream: New music playback interface for streaming, sticking, sorting, and recalling musical pieces. In *Proc. Int. Conf. on Music Information Retrieval (ISMIR)*, 2005.

[9] E. Law and L. von Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In *Proc. CHI 2009*, pages 1197–1206, 2009.

[10] C.S. Mesnage and M.J. Carman. Tag navigation. In *Proc. Int. Workshop on Social software engineering and applications*, pages 29–32, 2009.

[11] S. Mizzaro, E. Nazzi, and L. Vassena. Collaborative annotation for context-aware retrieval. In *Proc. of the WSDM '09 Workshop on Exploiting Semantic Annotations in Information Retrieval*, pages 42–45, 2009.

[12] E. Pampalk, S. Dixon, and G. Widmer. Exploring music collections by browsing different views. In *Proc. Int. Conf. on Music Information Retrieval (ISMIR)*, 2003.

[13] E. Pampalk and M. Goto. Musicrainbow: A new user interface to discover artists using audio-based similarity and web-based labeling. In *Proc. Int. Conf. on Music Information Retrieval (ISMIR)*, 2006.

[14] Kohonen. T. *Self-Organizing Maps*. 1995.

[15] G. Tzanetakis and P. Cook. Musical Genre Classification of Audio Signals. *IEEE Trans. on Speech and Audio Processing*, 10(5), July 2002.

[16] F.B. Viégas and M. Wattenberg. Timelines : Tag clouds and the case for vernacular visualization. *interactions*, 15(4):49–52, 2008.

HANDLING REPEATS AND JUMPS IN SCORE-PERFORMANCE SYNCHRONIZATION

Christian Fremerey
Bonn University
Computer Science
Bonn, Germany
fremerey@iai.uni-bonn.de

Meinard Müller
Saarland University and
MPI Informatik
Saarbrücken, Germany
meinard@mpi-inf.mpg.de

Michael Clausen
Bonn University
Computer Science
Bonn, Germany
clausen@iai.uni-bonn.de

ABSTRACT

Given a score representation and a recorded performance of the same piece of music, the task of score-performance synchronization is to temporally align musical sections such as bars specified by the score to temporal sections in the performance. Most of the previous approaches assume that the score and the performance to be synchronized globally agree with regard to the overall musical structure. In practice, however, this assumption is often violated. For example, a performer may deviate from the score by ignoring a repeat or introducing an additional repeat that is not written in the score. In this paper, we introduce a synchronization approach that can cope with such structural differences. As main technical contribution, we describe a novel variant of dynamic time warping (DTW), referred to as *JumpDTW*, which allows for handling jumps and repeats in the alignment. Our approach is evaluated for the practically relevant case of synchronizing score data obtained from scanned sheet music via optical music recognition to corresponding audio recordings. Our experiments based on Beethoven piano sonatas show that JumpDTW can robustly identify and handle most of the occurring jumps and repeats leading to an overall alignment accuracy of over 99% on the bar-level.

1. INTRODUCTION

Given a score and a performance of the same piece of music, a common task of music information retrieval consists of synchronizing note events or musical sections given by the score representation with time positions or temporal sections of the performance. A useful example application of such a synchronization is to allow users to navigate in a recorded performance of a piece of music by selecting locations of interest from the visual sheet music representation of the synchronized score and simultaneously playback the performance while highlighting the current playback position in the sheet music [1].

Scores and performances can be given in many different forms and formats. For example, scores can be given as scans of printed sheet music, vector graphics generated by a computer typesetting software, optical music recognition results, symbolic score formats such as MusicXML, Humdrum, or Lilypond, or as MIDI files. Performances are usually given as audio recordings or in form of MIDI files generated by electronic instruments. When aligning score and performance representations, challenging problems arise when the two representations reveal differences in their global overall structures. For example, a performer may ignore a repeat that is written in the score or may introduce an extra repeat that is not written in the score (e.g. an additional verse). Furthermore, a performance may include parts that are not written in score at all (e.g., a cadenza or solo part) or may skip certain parts of an underlying score. Structural differences between scores and performances have been encountered in previous work on on-line score following such as [2–4]. In this scenario, the scores and performances that are synchronized are usually monophonic. The most popular approach for this scenario is to use hidden Markov models (HMM) in combination with a training process to determine model parameter that suit the given type of data. For the case of off-line synchronization of polyphonic scores and performances, dynamic time warping (DTW) in combination with chroma features has become a popular approach [5, 6] because it can deliver similar accuracy than HMMs but without the need for creating and training models. Furthermore, efficient multi-scale implementations can easily be realized for this approach [7]. An overview on on-line and off-line score-performance synchronization approaches is found in [8]. In previous work on off-line score-performance synchronization, a basic assumption usually is that there are no structural differences between the two versions to be aligned. In [6], the authors point out that classical DTW can bypass additional segments such as repeated verses, at least to some extent. Raphael [9] remarks in his work that structural differences such as repeats are a common problem in score-performance synchronization. Content-based comparison of scores and performances also plays an important role in retrieval scenarios [10–12]. As pointed out in [12], retrieval methods may also be used to determine the structural differences between a score and a performance. Further related work has focused on performances only, either in the scenario of general partial music synchronization [13] or structural analysis of performances [14, 15].

In this paper, we describe a novel approach that allows for synchronizing score and performance data in the presence of structural differences. The main motivation for our work originates from a problem of high practical relevance arising in the data acquisition and processing pipeline of a digital music library [1]. Here, the score data is typically obtained by first scanning the given printed sheet music

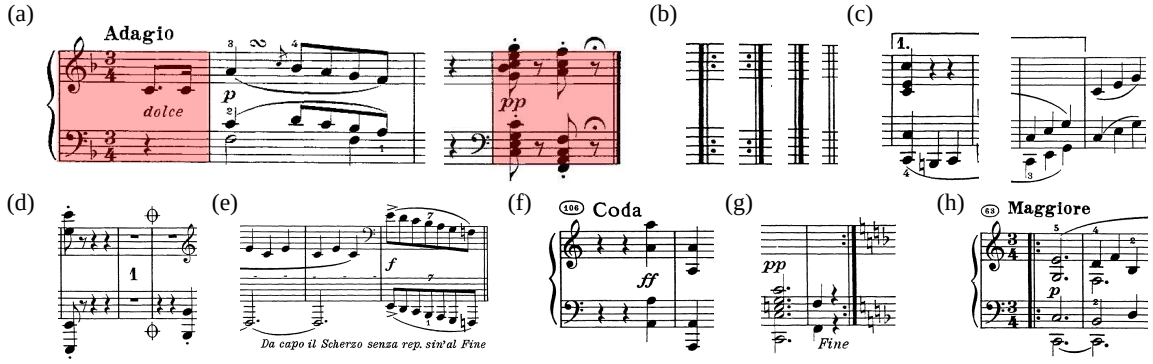


Figure 1. Examples for several types of block boundary indicators: (a) beginning and end of movement/song, (b) double bar lines with and without repeat signs, (c) brackets for alternative endings, (d) segno marker, (e) textual jump directive, (f) coda, (g) fine, (h) title heading of new musical section

material and then by converting the digitized images into a symbolic score representation using optical music recognition (OMR). In this process, repeat and jump directives that are written in the printed sheet music (as shown in Fig. 1) are often not recognized reliably by the OMR software. Besides the reasons given above, such missing directives are a major source for structural differences between the resulting score representation and a given audio recording. As the main technical contribution of this paper, we introduce a novel variant of dynamic time warping (DTW), which we refer to as *JumpDTW*. The main idea of our approach is to estimate the repeats and jumps that make the score match the performance and to calculate the actual score-performance alignment within a joint optimization procedure based on a content-based comparison of the score and audio data. The task tackled in this paper is related to the task of computing a possibly large partial alignment of two data streams [13, 16]. However, in contrast to these approaches, our goal is to somehow unfold the score representation to best explain the performance. Furthermore, we assume that the jumps and repeats only occur on musically meaningful positions by exploiting additional structural information given by the score. To this end, the score is searched for structural elements such as double bar lines to divide the score into blocks, see Fig. 1. Then repeats and jumps are allowed only at block boundaries but never inside blocks.

The remainder of this paper is organized as follows. In Sect. 2, we formalize the task of handling repeats and jumps in score-performance synchronization. In Sect. 3, we describe our novel *JumpDTW* algorithm in detail and indicate several extensions. Finally, in Sect. 4, we present experiments performed on a test dataset consisting of piano sonatas by Beethoven and conclude in Sect. 5 with a discussion of future work.

2. PROBLEM MODELING

We now assume that we are given one sheet music representation and one performance in form of an audio recording of the same piece of music. After processing the sheet music via OMR, one obtains a symbolic representation referred to as *score* representation. The score is naturally divided into sections that are delimited by either bar lines or the left or right boundary of a grand staff. Even though these sections may differ from the musical bars as they are usually counted in Western sheet music notation, in this paper, we simply refer to each such section as *bar*.

Let $\mathcal{B}$ denote the set of bars appearing in the score and let $K = |\mathcal{B}|$ be the number of bars. Ordering the set of bars by their visual occurrence in the sheet music (canonically ordered by the page number, line number, and left to right within a line), one obtains a sequence $\sigma = (\sigma_1, \ldots, \sigma_K)$, $\sigma_k \in \mathcal{B}$, $k \in [1 : K]$, which we refer to as *score bar sequence*. Note that the score bar sequence does not account for jump and repeat directives, see Fig. 2. Depending on the context, we use the term *bar* to denote either an element of $\mathcal{B}$, the region in the sheet music image that represents the bar, the musical content of the bar, or one of possibly many occurrences of the bar in the performance.

As discussed before, sheet music may contain jump and repeat directives such as repeat signs, alternative endings, dacapos or segnos, see Fig. 1. Because of these directives, the given performance often deviates from the score bar sequence σ. The musician may even choose to ignore or add some of the displayed repeats or may introduce shortcuts. This leads to a possibly different sequence $\pi = (\pi_1, \ldots, \pi_J)$, $\pi_j \in \mathcal{B}$, $j \in [1 : J]$, which we call *performance bar sequence*, see Fig. 2. Note that in the scenario discussed in this paper, the performance bar sequence π is unknown. One application of the approach introduced in the remainder of this paper is to determine this sequence π.

To relate the score bar sequence σ and the performance bar sequence π, intuitively, the score bar sequence, which represents the source material, has to be suitably *unfolded* to best explain the performance. Here, the *unfolding* typically appears at the jump and repeat directives indicated by the sheet music. Making use of this fact, the problem of unfolding sequences of bars can be reduced to the easier task of unfolding much shorter sequences of so-called *blocks* which are obtained by concatenating suitable subsequences of bars during which no repeats or jumps are expected to occur. To this end, the score is searched for *block boundary indicators* that indicate bars in the score that might serve as source or target for jumps and repeats. Examples of these indicators are depicted in Fig. 1.

Let $k_0 = 0 < k_1 < \ldots < k_{I-1} < k_I = K$ be boundary indices corresponding to the jump and repeat directives. Then, we define the block

$$\beta_i = (\sigma_{k_{i-1}+1}, \ldots, \sigma_{k_i}) \quad (1)$$

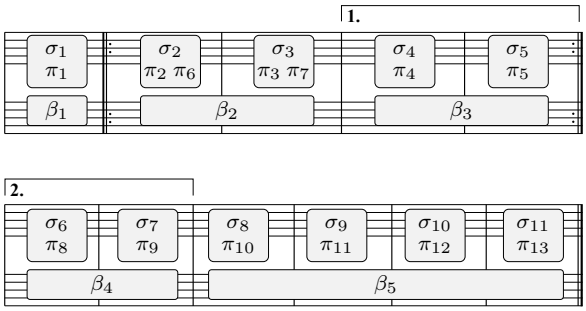


Figure 2. Illustration of the score bar sequence σ, the performance bar sequence π and the score block sequence β.

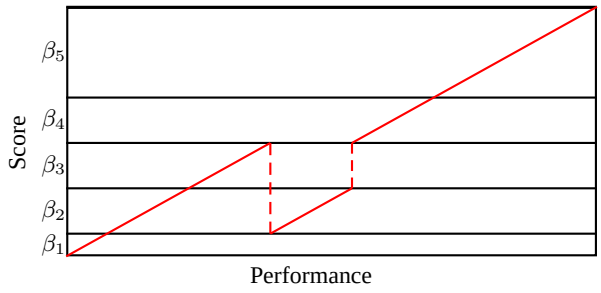


Figure 3. Visualization of a score-audio synchronization result with score block sequence $b = (1, 2, 3, 2, 4, 5)$ for the score and performance bar sequences shown in Fig. 2. The red line indicates an alignment path with jumps.

of length $|\beta_i| = k_i - k_{i-1}$ for $i \in [1 : I]$. The resulting *score block sequence* $\beta := (\beta_1, \ldots, \beta_I)$ is a partition of σ, see Fig. 2. Now, the task of finding the performance bar sequence π is reduced to finding a sequence of block indices $b = (b_1, \ldots, b_G)$, $b_g \in [1 : I]$, $g \in [1 : G]$, such that $(\beta_{b_1}, \ldots, \beta_{b_G})$ is as close as possible to the performance bar sequence π. The task of finding such a sequence b is discussed in the next section. For an example, we refer to Fig. 3. Note that, depending on the context, we will later use the term *block* not only to denote elements of the score block sequence β, but also to refer to elements of the block index sequence b.

3. PARTIAL SYNCHRONIZATION WITH JUMPS

Most procedures for score-performance synchronization first convert the two data streams to be aligned into suitable feature representations. Then, based on a local cost measure that allows for comparing features, a global alignment path between the feature sequences is computed using dynamic time warping (DTW). This procedure only works well if the score and the performance are in global correspondence and do not differ in their overall structure.

To account for structural differences as occurring in our scenario, we extend the classical DTW approach to enable jumps in the alignment path. Our idea of allowing jumps is inspired by the way a piece of music is often modeled using a Hidden Markov Model (HMM). Here, the note events of a score are modeled by states which are left-to-right connected to enforce that the music can only move forward but not backward. To account for possible repeats and jumps at certain block boundaries, one then simply adds further connections that connect states representing possible jump sources to states representing possible jump targets. After a short review of classical DTW (Sect. 3.1), we show how the jump directives can be incorporated (Sect. 3.2) and then indicate further DTW variants (Sect. 3.3).

3.1 Classical DTW

Introducing some notation, we now summarize the classical DTW approach using a slight reformulation. Let $x = (x_1, \ldots, x_N)$ and $y = (y_1, \ldots, y_M)$ be the feature sequences obtained from the score and performance representation, respectively. Furthermore, let c denote the local cost measure used to compare two features. Then the *local cost matrix* C of dimension $N \times M$ is defined by

$$C(n, m) := c(x_n, y_m) \tag{2}$$

for $(n, m) \in Z$, where $Z := [1 : N] \times [1 : M]$ is referred to as the set of *cells*. A (global) *alignment path* between x and y is a sequence $p = (p_1, \ldots, p_L)$ with $p_\ell = (n_\ell, m_\ell) \in Z$ for $\ell \in [1 : L]$ satisfying the boundary condition $p_1 = (1, 1)$ and $p_L = (N, M)$ and the step condition $p_\ell - p_{\ell-1} \in \Sigma$ for $\ell \in [2 : L]$ Here, $\Sigma := \{(1, 0), (0, 1), (1, 1)\}$ denotes the set of possible steps. The cost of the path p is defined by $\sum_{\ell=1}^{L} C(p_\ell)$. An *optimal alignment path* is defined to be an alignment path having minimal cost over all possible alignment paths.

An optimal alignment path can be computed using dynamic time warping (DTW). First, for a given cell $(n, m) \in Z$, one defines the set $Z_{n,m}$ of possible *predecessors* by

$$Z_{n,m} := \{(n, m) - z \mid z \in \Sigma\} \cap Z. \tag{3}$$

Then, one computes an *accumulated cost matrix* D of dimension $N \times M$. First, one sets $D(1, 1) := C(1, 1)$ and then recursively defines

$$D(n, m) := C(n, m) + \min\big\{D(z) \mid z \in Z_{n,m}\big\} \tag{4}$$

for $(n, m) \in Z \setminus \{(1, 1)\}$. The value $D(N, M)$ represents the cost of an optimal alignment path. Such an optimal path can be constructed based on a simple back tracking algorithm using D. For details, we refer to [17].

3.2 JumpDTW

To account for structural differences between the score and the performance caused by repeats and jumps, we now extend the concept of an alignment path and the classical DTW approach. Recall that we assume that the jumps occur from ends to beginnings of the blocks β_i, $i \in [1 : I]$. With regard to the feature representation $x = (x_1, \ldots, x_N)$ of the score, we assume that the beginning of β_i corresponds to index $s_i \in [1 : N]$ and the end to index $t_i \in [1 : N]$, where $s_i < t_i$. Furthermore, we assume that the beginning of block β_{i+1} immediately follows the end of block β_i, i.e., $s_{i+1} = t_i + 1$. Let $S := \{s_i \mid i \in [1 : I]\}$ and $T := \{t_i \mid i \in [1 : I]\}$.

Next, an *alignment path with jumps* with respect to the sets S and T is defined to be a sequence $p = (p_1, \ldots, p_L)$ with $p_\ell = (n_\ell, m_\ell) \in Z$ for $\ell \in [1 : L]$ satisfying the boundary condition as before. However, this time we modify the step condition by requiring that either $p_\ell - p_{\ell-1} \in \Sigma$ (as before) or

$$m_{\ell-1} = m_\ell - 1 \ \wedge \ n_{\ell-1} \in T \ \wedge \ n_\ell \in S. \tag{5}$$

In other words, besides the regular steps, we also permit jumps in the first coordinate (corresponding to the score) from the end of any block (given by T) to the beginning of any other block (given by S), see also Fig. 3.

We now introduce a modified DTW version, referred to as *JumpDTW*, that allows for computing an optimal alignment path with jumps. Recall that, in classical DTW, the set $Z_{n,m}$ of possible predecessor cells encodes all cells from which one can reach the cell (n,m) by applying a single step from Σ, see (3). The main idea of our modification is to add further predecessor cells that model possible jumps between the block boundaries. To this end, we extend all sets $Z_{n,m}$ for $n \in S$ by setting

$$\tilde{Z}_{n,m} := Z_{n,m} \cup \big(\{(t, m-1) \,|\, t \in T\} \cap Z\big). \quad (6)$$

Furthermore, we set $\tilde{Z}_{n,m} := Z_{n,m}$ for all other $n \in [1:N] \setminus S$. Intuitively, the additional predecessor cells in $\tilde{Z}_{n,m} \setminus Z_{n,m}$ permit jumps from the end of any block to the beginning of any other block. As in the classical case, one then computes an accumulated cost matrix simply by replacing the sets $Z_{n,m}$ by the sets $\tilde{Z}_{n,m}$ obtaining a matrix $\tilde{D}$. More precisely, we set $\tilde{D}(1,1) = C(1,1)$ and

$$\tilde{D}(n,m) := C(n,m) + \min\big\{\tilde{D}(z) \,|\, z \in \tilde{Z}_{n,m}\big\} \quad (7)$$

for $(n,m) \in Z \setminus \{(1,1)\}$. Note that for a given $(n,m) \neq (1,1)$, the set $\tilde{Z}_{n,m}$ only contains cells of the form $(n-1,m)$ or $(k,m-1)$ for some $k \in [1:N]$. In other words, $\tilde{Z}_{n,m}$ only contains cells that lie below or to the left of the current cell (n,m) when the axes are chosen as in Fig. 3. Therefore, $\tilde{D}$ can still be computed recursively in a column-wise fashion. The matrix entry $\tilde{D}(N,M)$ yields the cost of an optimal alignment path with jumps. As for the classical case, such an optimal path can then be constructed based on a simple back tracking algorithm using $\tilde{D}$.

From an optimal warping path with jumps one can derive the underlying sequence of block indices $b = (b_1, \ldots, b_G)$, $b_g \in [1:I]$, $g \in [1:G]$, in a canonical way. Starting with the first block, one either enters the subsequent block via a step from Σ or enters a different block via a jump. For example, in the case of a jump from $p_{\ell-1} = (t_j, m-1)$ to $p_\ell = (s_i, m)$ for some $\ell \in [2:L]$, one obtains $b_{g-1} = j$ and $b_g = i$ for some $g \in [2:G]$, see also Fig. 3 for an illustration. Having determined the sequence of block indices b, one can easily derive the performance sequence π by expanding blocks to bars.

3.3 Further DTW Variants

Because of the boundary condition, an alignment path starts at $p_1 = (1,1)$ and ends at $p_L = (N,M)$. Therefore, the score block sequence b is also restricted to start with the first block $b_1 = 1$ and to end with the last block $b_G = K$. In practice, however, a performance may end with a different block. For example, this happens in the presence of a "dacapo", where the piece ends at a block marked with the keyword "fine." To account for this possibility, one can easily modify the JumpDTW algorithm. Instead of looking at the entry $\tilde{D}(N,M)$, one simply has to determine the index

$$n^* := \operatorname{argmin}\big\{\tilde{D}(n,M) \,|\, n \in T\big\}. \quad (8)$$

Then, the alignment path with jumps is computed via backtracking starting with the cell (n^*, M) instead of (N,M). Similarly, one can relax the condition that one has to start with the first block, see [17] for details. Note that further constraints on the jumps can easily be handled by suitably modifying the sets $\tilde{Z}_{n,m}$ of predecessor cells. For example, to restrict the jump possibilities for a given block β_i, one simply restricts the set T to a suitable subset $T' \subset T$ and then uses $\tilde{Z}_{s_i,m} := Z_{s_i,m} \cup \big(\{(t, m-1) \,|\, t \in T'\} \cap Z\big)$.

4. EXPERIMENTS

To evaluate the usefulness of JumpDTW in a practically relevant application, experiments are conducted on the first 15 piano sonatas by Beethoven including a total of 54 individual movements. The score data is obtained from OMR results of a printed sheet music edition, and the performances are given as audio CD recordings. Since the score data does not include any tempo information, a mean tempo is estimated for each movement using the number of bars and the duration of the corresponding performance. For each movement, the score bar sequence σ is known and the score block sequence β is obtained using block boundary indicators extracted from the score. Note that this may include block boundary indicators where actually no jump or repeat occur in the performance. The performance bar sequence π is given as ground truth and is used to derive a ground truth block index sequence b with respect to β. For our test data set, the total number of score blocks appearing in the sequences β of the 54 movements is 242. The total number of score bars is 8832. Note that, because of repeats and jumps, a score block may occur more than once in the performance. Therefore, the total number of blocks appearing in the sequences b is 305 which corresponds to a total of 11836 bars being played in the performance. The total duration of the performance amounts to 312 minutes.

JumpDTW is performed on the data using β as described in Section 3.2. From the resulting warping path with jumps, an output block index sequence $b' = (b'_1, \ldots b'_{G'})$ is obtained. In the optimal case, this block index sequence b' would be equal to the ground truth block index sequence b. Table 1 shows the results of comparing b' to b using several different evaluation measures. Each row shows the results for different sets of b' obtained using a different JumpDTW variant. Each entry in the table summarizes the results for all 54 movements. The first row, tagged `nojumps`, represents the results when using classical DTW as described in Sect. 3.1, which serves as bottom line in our evaluation. The second row, tagged `s1_plain`, represents the basic JumpDTW algorithm as described in Section 3.2 including the relaxed boundary condition for dacapo/fine cases as described in 3.3.

The numbers plotted in the first six columns are based on a direct comparison of the sequences b' and b and measure how many blocks (abbreviated as `blk`) or performance bars (`bar`) match between the two sequences (`mch`), have been erroneously inserted into b' (`ins`), or have been erroneously omitted in b' (`omt`) with respect to the ground truth b. To this end, we calculate an optimum alignment between the two block index sequences using a variant of the edit distance that only allows insertions and deletions (but not replacements). To find an alignment between the two block index sequences that is optimal with respect to the amount of inserted and omit-

	mch blk % (#)	ins blk % (#)	omt blk % (#)	mch bar % (#)	ins bar % (#)	omt bar % (#)	prf % (#)
nojumps	70.2 (214)	0.3 (1)	29.8 (91)	74.6 (8831)	0.0 (1)	25.4 (3005)	69.8 (8258)
s1_plain	93.4 (285)	9.2 (28)	6.6 (20)	99.2 (11740)	0.9 (105)	0.8 (96)	98.5 (11661)
s2_add_special_states	93.4 (285)	5.6 (17)	6.6 (20)	99.3 (11759)	0.7 (82)	0.7 (77)	98.8 (11692)
s3_penalize_0.5_100	94.4 (288)	9.5 (29)	5.6 (17)	99.4 (11767)	0.4 (51)	0.6 (69)	99.1 (11725)

Table 1. Evaluation results for classical DTW and different variants of JumpDTW.

ted bars (instead of blocks), each block index entry in the sequences is weighted by the length of the corresponding score block. Each entry in Table 1 is given as a percentage with respect to the total number of blocks/bars in the performance followed by the absolute number in parentheses. For example, the entry $70.2(214)$ in the first row and column means that 214 blocks of b' have a matching counterpart in b, which is $214/305 = 70.2\%$ of the total number of blocks in b. Similarly, the entry $74.6(8831)$ for matching bars means that the 214 matching blocks have a total length of 8831 bars, which is $8831/11836 = 74.6\%$ of the total length of b in bars.

A further evaluation measure (`prf`), which is plotted in the last column of Table 1, expresses the alignment accuracy on the bar-level. This measure is motivated by the application of visually presenting sheet music that is linked on a bar-wise level to a given recorded audio performance. For this application, we want to measure for how many of the performance bars the alignment computed via JumpDTW is suitably accurate. To this end, the ground truth block index sequence b is used to create a feature sequence x from the score data that matches the repeats and jumps of the performance. Then, this feature sequence is synchronized to a feature sequence y obtained from the performance using classical DTW. From the output warping path, we derive a bar-wise score-performance synchronization that maps each performance bar $\pi_j \in \pi$ to a temporal region with center time d_j in the performance, see Fig. 4. Furthermore, this synchronization delivers a mapping $\phi : [0, D] \to [1 : K]$, with D being the duration of the performance, that for each time positions $d \in [0, D]$ in the performance returns an index $k \in [1 : K]$ indicating that bar σ_k is played at time d, see also Fig. 4. Since, from manual inspection, the synchronization results obtained when using the ground truth block index sequence b are known to be suitably accurate on a bar-wise level, they are used as a reference for finding deviations in the synchronization results obtained using b'. For each performance bar π_j, we take the bar center time d_j and input it into the mapping ϕ' obtained from the synchronization results using b'. The performance bar is counted as correctly matched if $\phi'(d_j) = \phi(d_j)$, which means that in the synchronization obtained using b', the time position d_j points to the same bar σ_k as in the reference synchronization. Unlike the mere number of matched bars listed in the column `mch bar`, this measure takes into account the extra confusion that is caused in the synchronization by erroneously inserted or omitted bars.

From the results using classical DTW (`nojumps`) one can see that about 70–75% of the blocks and bars of the performance are covered by the plain score bar sequence. The remaining 25–30% are repeats that are omitted in this sequence. The synchronization-based measure indicates a similar result: 69.8% of the center time positions of the bars in the performance were aligned to the correct bar in the score. These results are improved significantly,

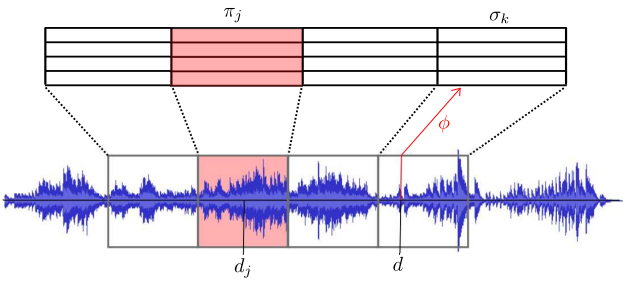


Figure 4. Illustration of a bar-wise score-performance synchronization. Each performance bar π_j is synchronized to a temporal region of a performance with bar center time d_j. Furthermore, a mapping ϕ can be derived that for a given time position d in the performance outputs the index k of the corresponding score bar σ_k.

when using JumpDTW (`s1_plain`). Here, 93.4% of the blocks and 99.2% of the bars are matched correctly. In the synchronization-based measure, 98.5% of the performed bars match the reference synchronization. Even though 28 blocks have been erroneously inserted and 20 blocks have been omitted, this amounts to only 105 inserted bars and 96 omitted bars, revealing that the mean length of inserted and omitted blocks is only about 4.2 bars.

Manual inspection of the results for the individual movements reveals that in many cases an extra block is inserted at the beginning or the end of the sequence to cover for silence at the beginning or end of the performance. In one case, this even leads to the last block of the sequence being confused with an incorrect one. To encounter this issue, we extend the JumpDTW algorithm by adding special states to the score representation that model silence at the beginning or end of the performance. The results for this modification are listed in the line labeled `s2_add_special_states` and show slightly improved numbers. An in-depth analysis of the results shows that this modification solved all of the previously mentioned problems caused by initial or trailing silence in the performance. Furthermore, it turned out that 130 of the $82 + 77 = 159$ inserted and omitted bars occur in just 3 of the 54 movements. The performance of the first movement of "Sonata 8, Op. 13, *Pathetique*" contains extreme tempo changes with slow sections of roughly 20 BPM (beats per minute) alternating with fast sections of about 300 BPM. This results in a large difference between the estimated mean tempo of the score and the tempo of the slow sections in the performance. The JumpDTW algorithm reacts by erroneously inserting more or less random blocks to cover the unexpectedly slow sections of the performance. A different kind of problem occurs in "Sonata 12, Op. 26, Andante con variazioni". Here, the second block is a variation of the first block that has virtually the same harmonic progression. The JumpDTW erroneously treats this second block in the performance as a repeat of the first block in the score. This behavior is not very surprising considering that

the content-based comparison of score and performance is somewhat noisy and for the chroma-based features used, sections with the same harmonic progression are almost indistinguishable. In "Sonata 13, Op. 27 No. 1, Andante–Allegro" it is again a significant change in the tempo that causes a problem. Here, a repeat of a block (length = 9 bars) of the faster Allegro section is omitted by JumpDTW, which is provoked by the estimated tempo of the score being significantly slower than the tempo of the corresponding section of the performance. For all of the remaining movements, only blocks of length 2 or lower are inserted or omitted.

To encounter the problems discussed above, we further extend the JumpDTW approach by introducing a penalty cost for performing jumps in the warping path that is added to the accumulated cost. The cost value is set to $0.5 \cdot \frac{N}{100}$, with N being the length of the score feature sequence. The particular formula is motivated by the idea of choosing a cost value that is close to the cost of matching $1/100$-th of the score to a section of the performance that is not considered similar. Since in our implementation, we use normalized chroma features with a cosine measure for the local cost, a local cost value of 0.5 is already considered not similar. The results for this modification are listed in the row `s3_penalize_0.5_100`. A closer analysis shows that adding the penalty solves the confusion for the "Andante con Variazioni" and lowers the amount of inserted bars for the slow sections of the "*Pathetique*", which leads to a better overall result. However, the penalty also causes degradation for many of the other movements because short blocks for alternative endings are no longer skipped. Tuning the penalty cost to higher or lower values did not improve the situation. An increased penalty led to an increased amount of erroneously skipped short blocks while a decreased penalty no longer solved the confusion for the two movements discussed above.

5. CONCLUSIONS

In this paper, we have formally modeled the task of score-performance synchronization in the presence of structural differences induced by jumps and repeats. To handle such differences, we introduced a novel DTW variant referred to as JumpDTW. The results of the experiments presented in Section 4 show that the JumpDTW approach can successfully align about 99% of the bars played in the performance on the given test dataset with less than 1% of bars being omitted and less than 1% of extra bars being inserted. This positive result suggests that the approach may be useful for the large-scale automatic alignment of OMR data and audio recordings in a digital music library scenario.

Introducing penalty cost for performing jumps did fix some problems occuring on the test dataset but also caused additional errors. Further improvements of our approach are needed in situations where one has large differences (more than a factor of two) in the estimated tempo of the score and the tempo of the actual performance. Also, when using chroma features, blocks that reveal a similar harmonic progression are prone to confusion. Here, combinations with other feature types may help to resolve this problem. Note that, besides the segmentation of the score data into blocks, the JumpDTW approach completely relies on content-based comparison of notes and acoustic data. If further structural information from the score can be incorporated, as for example tempo directives or jumps and repeats as suggested by the notation, many of the remaining issues and inaccuracies might be solved. Besides this, another direction of future work may be to incorporate the case of cadenzas, where the performance contains sections that are not written in the score.

Acknowledgement. This work was supported by the German Research Foundation (DFG, CL 64/6-1) and by the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University. We would like to thank the anonymous reviewers for their very helpful comments and suggestions.

6. REFERENCES

[1] D. Damm, C. Fremerey, F. Kurth, M. Müller, and M. Clausen. Multimodal presentation and browsing of music. In *Proc. ICMI*, pp. 205–208, Chania, Crete, Greece, 2008.

[2] M.E. Tekin, C. Anagnostopoulou, and Y. Tomita. Towards an intelligent score following system: Handling of mistakes and jumps encountered during piano practicing. In *Computer Music Modeling and Retrieval*, pp. 211–219, 2005.

[3] B. Pardo and W. Birmingham. Modeling form for on-line following of musical performances. In *Proc. National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania, 2005.

[4] A. Arzt, G. Widmer, and S. Dixon. Automatic page turning for musicians via real-time machine listening. In *Proc. ECAI*, Patras, Greece, 2008.

[5] R. Dannenberg and N. Hu. Polyphonic audio matching for score following and intelligent audio editors. In *Proc. ICMI*, pp. 27–34, San Francisco, USA, 2003.

[6] R. Turetsky and D. Ellis. Ground-truth transcriptions of real music from force-aligned MIDI syntheses. In *Proc. ISMIR*, pages 135–141, Baltimore, Maryland, USA, 2003.

[7] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. In *Intelligent Data Analysis*, 11(5):561–580, 2007.

[8] R. Dannenberg and C. Raphael. Music score alignment and computer accompaniment. *Comm. ACM, Special Issue: Music Information Retrieval*, 49(8):38–43, 2006.

[9] C. Raphael. A hybrid graphical model for aligning polyphonic audio with musical scores. In *Proc. ISMIR*, Barcelona, Spain, 2004.

[10] J. Pickens, J. P. Bello, G. Monti, T. Crawford, M. Dovey, M. Sandler, and D. Byrd. Polyphonic score retrieval using polyphonic audio. In *Proc. ISMIR*, Paris, France, 2002.

[11] I. Suyoto, A. Uitdenbogerd, and F. Scholer. Searching musical audio using symbolic queries. *IEEE TASLP*, 16(2):372–381, 2008.

[12] C. Fremerey, M. Clausen, M. Müller, and S. Ewert. Sheet music-audio identification. In *Proc. ISMIR*, pp. 645–650, Kobe, Japan, 2009.

[13] M. Müller and D. Appelt. Path-constrained partial music synchronization. In *Proc. ICASSP*, pp. 65–68, Las Vegas, Nevada, USA, 2008.

[14] M. Goto. A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE TASLP*, 14(5):1783–1794, 2006.

[15] M. Levy and M. Sandler. Structural segmentation of musical audio by constrained clustering. *IEEE TASLP*, 16(2):318–326, 2008.

[16] J. Serrà, E. Gómez, P. Herrera, and X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE TASLP*, 16:1138–1151, 2008.

[17] M. Müller. *Information Retrieval for Music and Motion*. Springer, 2007.

HIERARCHICAL CO-CLUSTERING OF MUSIC ARTISTS AND TAGS

Jingxuan Li
School of Computer Science
Florida International University
Miami, FL USA
jli003@cs.fiu.edu

Tao Li
School of Computer Science
Florida International University
Miami, FL USA
taoli@cs.fiu.edu

Mitsunori Ogihara
Department of Computer Science
University of Miami
Coral Gables, FL USA
ogihara@cs.miami.edu

ABSTRACT

The user-assigned tag is a growingly important research topic in MIR. Noticing that some tags are more specific versions of others, this paper studies the problem of organizing tags into a hierarchical structure by taking into account the fact that the corresponding artists are organized into a hierarchy based on genre and style. A novel clustering algorithm, *Hierarchical Co-clustering Algorithm (HCC)*, is proposed as a solution. Unlike traditional hierarchical clustering algorithms that deal with homogeneous data only, the proposed algorithm simultaneously organizes two distinct data types into hierarchies. HCC is additionally able to receive constraints that state certain objects "must-be-together" or "should-be-together" and build clusters so as to satisfying the constraints.

HCC may lead to better and deeper understandings of relationship between artists and tags assigned to them. An experiment finds that by trying to hierarchically cluster the two types of data better clusters are obtained for both. It is also shown that HCC is able to incorporate instance-level constraints on artists and/or tags to improve the clustering process.

1. INTRODUCTION

The user-defined tags are becoming an essential component in web databases and social network services. The tags assigned to events and data objects as a whole represent how they are received by the community and provide keys to other users accessing them. In music information retrieval some recent papers study how to incorporate tags effectively for fundamental data retrieval tasks such as clustering, recommendation, and classification (see, e.g., [4, 19, 21, 22]).

An important characteristic of the tags is that sometimes tags are extensions of others and thus more specific than those they extend, e.g., "Soft Metal" extending "Metal", "Dance Pop" extending "Pop", and "Extremely Provocative" extending "Provocative". Since there is no limit in the length of tags, a tag can be an extension of another one, which is an extension of yet another one. This suggests that the music tags can be not only clustered as has been done before but *hierarchically* clustered.

Many approaches have been developed to produce hierarchical organizations of words and of documents, and so organizing tags into a hierarchy is a problem that is already-solved. However, we observe that the artists, to which tags are assigned, too can be organized into a hierarchical structure based on their prominent genres and styles. In fact, these hierarchies are much related to each other, since style labels often appear as tags. This leads to the questions of whether an attempt to build simultaneously hierarchical organizations of tags and of artists will lead to better organizations of both and of whether such organizations can be effectively and efficiently built.

In data mining the problem of developing hierarchical organization of data is referred to as *hierarchical clustering* and the problem of clustering two data types is referred to as *co-clustering*. While co-clustering essentially aims at simultaneously clustering rows and columns of a matrix, where the rows and the columns correspond to separate data types (e.g., terms and documents), hierarchical clustering aims at building a tree-like structure of the rows based on the columns a tree-like structure of the columns based on the rows. While both organizations have their own advantages, such as natural facilitation of data navigation and browsing in hierarchical clustering [6], few algorithms simultaneously build both [2].

In this paper, we develop a novel method, called HCC, for simultaneously clustering two data types, and we use HCC for building hierarchical co-clusters of tags and styles. HCC is designed based on the approaches in [10, 14]. HCC is essentially agglomerative hierarchical clustering: it starts with singleton clusters and then repeatedly merging two nearest clusters into one until there remains only one cluster. However, it may *merge groups from different data types at any point*. In our case, this means that at each step of the merging process, HCC can merge a subset of the artists with a subset of the tags based on their internal heterogeneity. In practice, one sometimes observes that a group of artists and a group of tags are exclusively correlated with each other (i.e., not correlated with any other artists or tags). HCC aims at, in such a situation, merging them into a single group at the earliest possible stage.

Our hope is that such artist-tag mixed clusters will be used for better retrieval when both artists and tags are specified in a query.

Figure 1 shows a sample output dendrogram of HCC while Figure 2 shows a sample output dendrogram of a traditional hierarchical clustering method. We show that such mixed-data-type hierarchical clusters can be generated by HCC and empirically better clusters are generated by concurrent use of two data types. Furthermore, we show that HCC can be extended to incorporate instance-level constraints that specify certain tags must be or must not be together or certain artists must be or must not be together for better organization.

The rest of the paper is organized as follows: Section 2 discusses the related work; Section 3 describes the details of HCC and the techniques for incorporating instance-level constraints; Section 4 presents experimental results; and finally Section 5 provides our conclusions.

2. RELATED WORK

Hierarchical Clustering is generation of tree-like cluster structures without user supervision. Hierarchical clustering algorithms organize input data either bottom-up (agglomerative) or top-down (divisive) [20]. **Co-clustering** refers to clustering of more than one data type. Dhillon [7] proposes bipartite spectral graph partitioning approaches to co-cluster words and documents. Long et al. [15] proposed a general principled model, called Relation Summary Network, for co-cluster heterogeneous data presented as a k-partite graph. While hierarchical clustering deals with only one type of data and the organization that co-clustering produces consists of just one level, **Hierarchical Co-clustering** aims at simultaneously construction of two or more hierarchies [12, 13].

Recently much work has been done on the use of background information in the form of instance level must-link and cannot-link constraints. This topic is referred to as **Constrained Clustering**. Here a must-link constraint enforces that two instances must be placed in the same cluster and a cannot-link constraint enforces that two instances must not be placed in the same cluster. Most of these constraint-based algorithms are developed for partitional clustering (e.g, K-means clustering, spectral clustering, and non-negative matrix factorizations) [1], and little has been done on utilizing constraints for hierarchical clustering.

3. HIERARCHICAL CO-CLUSTERING (HCC)

3.1 Problem Formulation

Suppose we are given a set of m artists $\boldsymbol{A}$=$\{a_1, a_2, \ldots, a_m\}$, and a set of n unique tags that are assigned to the music of these artists $\boldsymbol{T}$=$\{t_1, t_2, \ldots, t_n\}$. Suppose we are also given an $m \times n$ artist-tag relationship matrix $X = (x_{ij}) \in \mathbb{R}^{m \times n}$, such that x_{ij} represents the relationship between the i-th artist in $\boldsymbol{A}$ and the j-th tag in $\boldsymbol{T}$. Our goal is to simultaneously generate a hierarchical clustering of $\boldsymbol{A}$ and of $\boldsymbol{T}$ based on matrix X.

3.2 HCC

Like agglomerative hierarchical clustering algorithms, HCC starts with singleton clusters and then successively merges the two nearest clusters until only one cluster is left. However, unlike traditional algorithms, it may unify classes from two different data types. This means that the cluster left at the end consists of all the rows and columns and so if there are m rows and n columns exist, HCC executes $m+n-1$ rounds. The output of HCC is thus a single tree where the leaves are the rows and the columns of the input matrix, where nodes having both rows and columns as descendants may appear at any non-leaf level. Note that, in Figure 1, at the third layer the artist A3 - Led Zeppelin is joined with the tag B2 - Classic rock.

The algorithm of HCC is presented in Algorithm 1. The

Algorithm 1 HCC Algorithm Description

```
Create an empty hierarchy H
List ← Objects in A + Objects in B
N ← size[A] + size[B]
Add List to H as the bottom layer

for i = 0 to N − 1 do
  p, q = PickUpTwoNodes(List)
  o = Merge(p, q)
  Remove p, q from List and add o to List
  Add List to H as the next layer
end for
```

central part in the design of Algorithm 1 is the method PickUpTwoNodes, which is for selecting two nodes (corresponding to two clusters) to merge. For the purpose of creating groups consisting of two different data types, we use cluster heterogeneity measurement, denoted by CH. Given a group C consisting of r rows, P, and s columns, Q, we define $CH(C)$ as

$$CH(C) = \frac{1}{rs} \sum_{i \in P, j \in Q} (x_{ij} - \mu)^2, \qquad (1)$$

where μ is the average of entries over rows P and columns Q; i.e., $\mu = \frac{1}{rs} \sum_{i \in P, j \in Q} x_{ij}$. For a merger, we choose the two nodes whose merging would result in the least increase in the total cluster heterogeneity [10].

3.3 Incorporating Instance-level Constraints

In practice, one may observe pairs of artists that should be clustered into the same cluster. Similarly, one may observe pairs of tags that must be always in the same tag cluster. These observations are represented as the aforementioned "must-link" and "cannot-link" constraints. We design HCC so as to incorporate such constraints.

There are two issues in incorporating these constraints. One is how to use them for grouping data points of the same type; i.e., how to use artist constraints for grouping artists and tag constraints for grouping tags. The other is how to transfer constraints on one data type to the other data type. To address the first issue, we use Dunn's Index

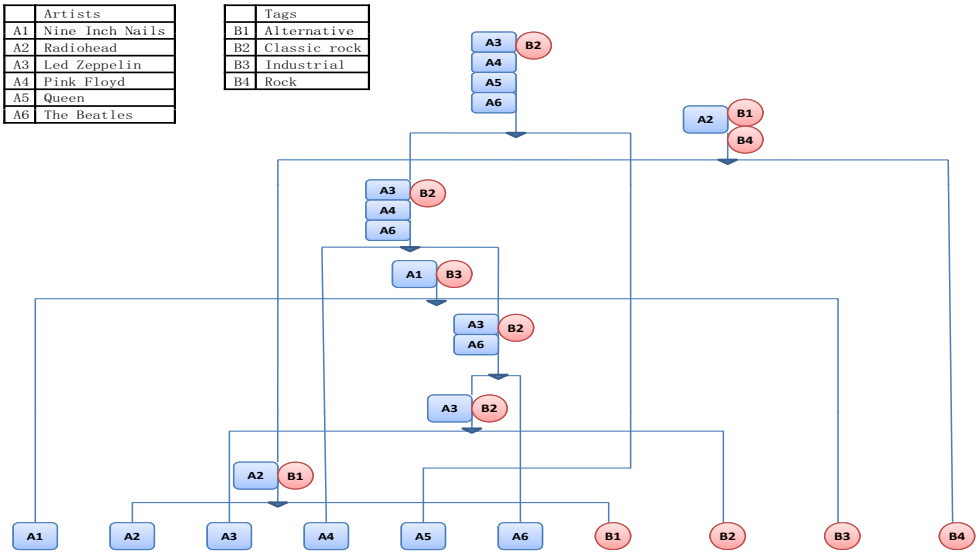


Figure 1. Part of HCC dendrogram. Rectangles represent artists and ellipses represent tags assigned to these artists. The nodes containing both rectangles and ellipses are clusters containing both an artist and a tag.

to determine the best layer for cutting the HCC-generated dendrogram and then apply the constrained K-Means to incorporate the constraints of the same data type. To address the second issue, we use an alternating exchange algorithm.

3.3.1 Best Layer

Since HCC produces a list of clustering results and each clustering corresponds to one layer of the dendrogram, we use Dunn's Validity Index [9] to measure and compare these clusterings. This validity measure is based on the idea that good clustering produces well-separated compact clusters. Given a clustering layer consisting of r clusters $c_1, \ldots, c_r$, Dunn's Index is given by:

$$D = \frac{\min_{1 \le i < j \le r} d(c_i, c_j)}{\max_{1 \le k \le r} d'_k}, \tag{2}$$

where $d(c_i, c_j))$ is the inter-cluster distance between the i-th and the j-th clusters and d'_k is the intra-cluster distance of the k-th cluster. Generally, the larger Dunn's Index, the better the clustering.

After determining the best layer to cut the dendrogram, we can easily make use of the constraints of the same data type. In particular, we perform constrained K-Means on the best layer with the parameter K set to the number of clusters in that layer. For this purpose, we use the MPCK-Means algorithm in [3].

3.3.2 Alternating Exchange

Here we show how to transfer the constraints between different data types. Specifically, at the best layer of the dendrogram generated by HCC, if some artist (or tag) data points of certain node are being re-assigned to another node at the same layer after using the instance-level constraints, we can use an alternating exchange algorithm [11] to improve tag (or artist) clustering. The objective function of clustering can be written as [11]:

$$Z = \sum_{k=1}^{r} \sum_{l=1}^{m} \sum_{i \in A_k} \sum_{j \in T_l} (x_{ij} - w_{kl})^2, \tag{3}$$

with

$$w_{kl} = \frac{1}{a_k t_l} \sum_{i \in A_k} \sum_{j \in T_l} x_{ij}. \tag{4}$$

Here r is the number of type A clusters, m is the number of type T clusters, A_k is the k-th cluster contains data points of type A, T_l is the l-th cluster contains data points of type T, a_k and t_l respectively denote data points of type A and T. As before, x_{ij} is the value representing the relationship between the i-th type-A data point and the j-th type-T data point.

To transfer constraints from tags to artists, we do the following: Suppose we have just obtained a clustering of artists, C_A, and a clustering of tags, C_T, by cutting the HCC dendrogram using Dunn's index, as described before. We first incorporate into these clusterings the tag constraints using the techniques described in Section 3.3.1 thereby obtain an improved tag clustering, C'_T. Then we execute the greedy algorithm shown in Algorithm 2 to make changes on artist class assignments. The greedy algorithm is aimed at minimizing the quantity Z in (3) and in each round one artist is moved from the current cluster to another if that move decreases the value of Z. Transferring constraints backward (i.e., from artists to tags) could be done by simplying switch the role of tags and artists. In our implemenation, we transfer only from tags to artists.

Algorithm 2 Alternating Exchange Algorithm

Input: clusterings C_A and C'_T, and normalized A-T matrix X, where C'_T is obtained by using the MPCK-Means on the output of HCC with respect to tag constraints.

while There is an artist whose relocation from the current cluster to another decreases the value of Z **do**
  pick an artist-destination pair that maximizes the decrease and relocate the artist to the destination
end while
Output the resulting artist clustering C'_A

4. EXPERIMENT

4.1 Data Set

We use the data set in [22] consisting of 403 artists. For each artist, tags and styles are collected from Last.fm (http://www.last.fm). There are 8,529 unique tags and 358 unique style labels. Note that an artist may receive the same tag more than once. By counting the number of assignments by the same tag, each artist is represented by a 8,529-dimensional integer vector. We scale these tag vectors so that the total of the 8,529 entries is equal to a fixed constant. We will use X to denote the artist-tag frequency matrix thus generated.

As to the style labels, each artist belongs to at least one style and each style contains at least one artist. We generate an artist-style incident matrix from the data, so that the entry at coordinate (i, j) is 1 if the i-th artist has the j-th style label and 0 otherwise.

4.2 Hierarchies Generated from HCC

We use HCC to generate a dendrogram of the artists and the tags. Figure 1 is part of the dendrogram generated by HCC in our experiment. In the dendrogram, each leaf represents one artist or one tag, each internal node contains subsets of artists and tags, and the top layer is the cluster contains all artists and tags. Because many people assign a tag "Industrial" to artist *Nine Inch Nails*, "Industrial" and *Nine Inch Nails* are clustered together. The novelty here is that artists and tags are jointly organized into a hierarchical structure. Once such a hierarchical organization has been generated, an artist can be described by the tags that appear in its cluster. The more representative are the tags for certain artists, the larger possibility for them to be clustered together.

We compare the HCC-generated dendrogram with one generated by single linkage hierarchical clustering (SLHC). This is the standard hierarchical clustering method and thus serves as our baseline. Since SLHC can cluster only one type of data, we provide SLHC with the normalized artist-tag matrix by viewing each row as the feature vector of the corresponding artist and produce hierarchical clustering of artists. The artist dendrogram generated by SLHC is shown in Figure 2. To evaluate and compare these two artist dendrograms, we utilize CoPhenetic

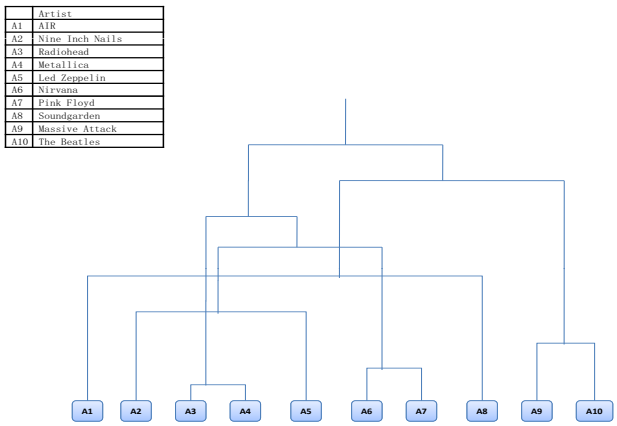


Figure 2. Part of the dendrogram generated by SLHC.

Correlation Coefficient (CPCC) [17] as evaluation measure. Intuitively CPCC measures how faithfully a dendrogram preserves the pairwise distances between the original data points. CoPhenetic Correlation Coefficient (CPCC) is given as:

$$\frac{\sum_{i<j}(d(i,j)-d)(t(i,j)-t)}{\sqrt{(\sum_{i<j}(d(i,j)-d)^2)(\sum_{i<j}(t(i,j)-t)^2)}} \quad (5)$$

Here $d(i, j)$ and $t(i, j)$ are respectively the ordinary Euclidean distance and the dendrogrammatic distance between the i-th and the j-th data points, and d and t are their respective averages. The CPCC for HCC was 3.71 while that for SLHC was 3.69, and so we can say that our HCC method generates faithful dendrogram with reasonable clustering performance on artist-tag dataset. Through the coupled dendrogram, one can observe the relationship between artists and tags, also make use of the tags within the same cluster as some artists to explain why these artists are clustered together.

4.3 Clustering Performance Comparisons

We also evaluate the artist clustering performance of HCC, by comparing it with three co-clustering algorithms including Information-Theoretic Co-clustering (ITCC) [8], Euclidean Co-clustering (ECC), and Minimum Residue Co-clustering (MRC) [5] on the artist-tag dataset.

Using style labels we obtain artist clusters and cluster labels. We first cluster the styles using KMeans clustering based on the artist-style matrix (that is, clustering of the columns, where each column is the 403-dimensional $0/1$ vector that shows assignments of the style corresponding to the column to the 403 artists). We then treat each cluster as a label and assign to each artist one label in the following manner:

- If all the styles assigned to an artist a belongs to a single cluster, we use that cluster as the label of a. Otherwise, choose the cluster with the largest number of styles assigned to a. If there is a tie, choose the one with the larger total number of styles, and if that doesn't break the tie, break it arbitrarily.

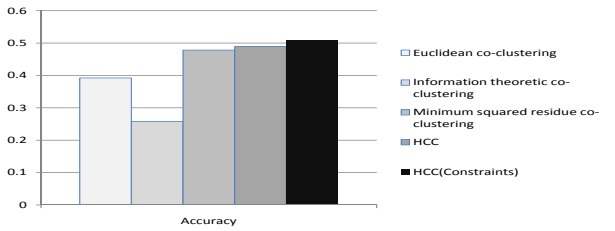


Figure 3. Accuracy of various clustering methods. HCC(constraints) represents HCC with 10 artist constraints.

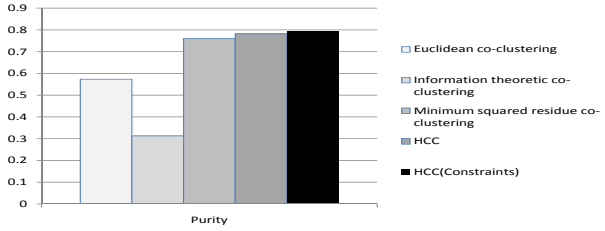


Figure 4. Purity of various different clustering methods. HCC(constraints) represents HCC with 10 artist constraints.

We use these labels as our ground truth class labels in the clustering performance measurements presented below.

4.3.1 Evaluation Measures

We use Accuracy, Normalized Mutual information (NMI), Purity, and Adjusted Rand Index (ARI) as performance measures. These measures have been widely used in clustering evaluation and we hope they would provide insights on the performance of our HCC method. For all these measures, the higher the value, the better the clustering.

Suppose we are given clusters $C_1, \ldots, C_k$ of size $c_1, \ldots, c_k$, respectively and we are comparing this clustering against the ground-truth clustering $E_1, \ldots, E_k$ of size $e_1, \ldots, e_k$. Let n be the total number of data points and for all i and j, let μ_{ij} denote the number of data points in both C_i and E_j.

Accuracy measures the extent to which each cluster contains the entities from corresponding class and is given by:

$$Accuracy = \max_{\pi} \frac{\sum_{i,\pi(i)} \mu_{i\pi(i)}}{n}, \tag{6}$$

where π ranges all permutations of $1, \ldots, k$. **Purity** measures the extent to which a cluster contains entities of a single class and is given by:

$$Purity = \frac{1}{n} \sum_{i=1}^{k} \mu_{i\rho(i)}, \tag{7}$$

where $\rho(i)$ is the j that maximizes μ_{ij}. **Adjusted Rand Index** is the corrected-for-chance version of Rand Index, and measures the similarity between two clusterings [16]. It is given by:

$$ARI = \frac{a - \frac{2bc}{n(n-1)}}{\frac{b+c}{2} - \frac{2bc}{n(n-1)}}. \tag{8}$$

Here $a = \sum_{i,j} \frac{\mu_{ij}(\mu_{ij}-1)}{2}$, $b = \sum_i \frac{c_i(c_i-1)}{2}$, and $c = \sum_j \frac{e_j(e_j-1)}{2}$. **NMI** is the normalized version of mutual information and measures how much information the two clusterings share [18] and is given by:

$$NMI = \frac{\sum_{i,j} \mu_{ij} \log(\frac{n\mu_{ij}}{c_i e_j})}{\sqrt{(\sum_i c_i \log \frac{c_i}{n})(\sum_j e_j \log \frac{e_j}{n})}}. \tag{9}$$

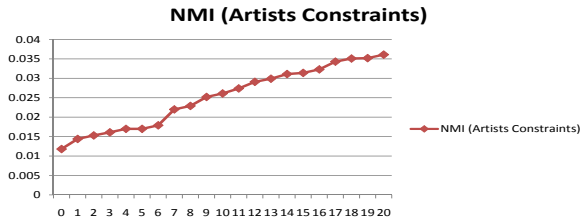


Figure 5. NMI on HCC with artists constraints range from 0 - 20.

4.3.2 Experimental Results

As we mentioned in Section 3.3.1, Dunn's Index can be used to find the best layer of the dendrogram generated by HCC. After computing Dunn's Index on the clustering of each layer, it is found that there are 11 clusters in the best layer. Since we have already obtained the best layer, the clustering of this layer is compared against Co-clustering algorithms. This clustering is based on artist data points, we applied co-Clustering algorithms for clustering artists.

Figures 3 and Figure 4 show the experiment results on the clustering methods using accuracy and purity as the performance measures, respectively. The results in both figures demonstrate that our HCC method outperforms the co-clustering methods. Similar behaviors can be observed when using ARI and NMI measures. Due to space limitation, we do not include the figures for ARI or NMI. Figures 3 and 4 also show that using the artist constraints improves the clustering performance.

We also evaluate NMI on HCC with increasing number of constraints. The result in Figure 5 shows that the artist clustering performance improves with the increasing number of artist constraints. In other words, the artist constraints improves the clustering performance of HCC. Figure 6 shows that artist clustering performance improves as the number of tag constraints increases.

5. CONCLUSION

In this paper, we propose a novel clustering method, HCC, to hierarchically cluster artists and tags simultaneously. With the dendrogram generated by HCC one can have a picture of all artists and tags, so as to find the relationship between the artists and tags within the same cluster. Furthermore, we perform experiments on artist-tag dataset, the results show that HCC outperforms its competitors, provid-

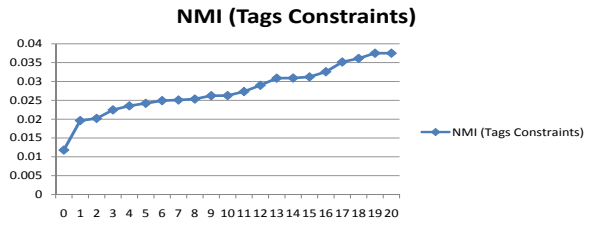


Figure 6. NMI on HCC with tags constraints range from 0 - 20.

ing reasonable dendrograms with clusterings in each layer. In the future, we would try out HCC on larger datasets to further confirm its ability in MIR area.

6. ACKNOWLEDGMENT

The work is partially supported by NSF grants IIS-0546280, CCF-0939179, and CCF-0958490 and an NIH Grant 1-RC2-HG005668-01.

7. REFERENCES

[1] S. Basu, I. Davdison, K. Wagstaff (eds): "Constrained Clustering: Advances in Algorithms, Theory, and Applications," Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, 2008.

[2] P. Berkhin: "A survey of clustering data mining techniques," *Grouping Multidimensional Data*, pp. 25–71, Springer, Heidelberg, 2006.

[3] M. Bilenko, S. Basu, and R. J. Mooney: "Integrating constraints and metric learning in semi-supervised clustering," in *Proc. 21st International Conference on Machine Learning*, pp. 81–88, 2004.

[4] K. Bosteels, E. Pampalk, and E. E. Kerre: "Music retrieval based on social tags: A case study," in *Proc. 9th International Conference on Music Information Retrieval*, 2008.

[5] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra: "Minimum sum-squared residue co-clustering of gene expression data," in *Proc. 4th SIAM International Conference on Data Mining*, pp. 114–125, 2004.

[6] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey: "Scatter/gather: a cluster-based approach to browsing large document collections," in *Proc. 15th ACM SIGIR Conference*, pp. 318–329, 1992.

[7] I. S. Dhillon: "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 269–274, 2001.

[8] I. S. Dhillon, S. Mallela, and D. S. Modha: "Information-theoretic co-clustering," in *Proc. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 89–98, 2003.

[9] J. C. Dunn: "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *J. Cybernetics*, 3(3):32–57, 1974.

[10] T. Eckes and P. Orlik: "An error variance approach to two-mode hierarchical clustering," *J. Classification*, 10(1):51–74, 1993.

[11] W. Gaul and M. Schader: "A new algorithm for two-mode clustering," in *Data analysis and information systems*, pp. 15–23, Springer, Heidelberg, 1996.

[12] M. Hosseini and H. Abolhassani: "Hierarchical co-clustering for web queries and selected urls,". in *Proc. Web Information Systems Engineering (WISE)*, pp. 653-662, 2007.

[13] D. Ienco, R.G. Pensa, and R. Meo: "Parameter-Free Hierarchical Co-clustering by n-Ary Splits,". in *Proc. Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pp. 580–595, 2009.

[14] J. Li and T. Li: "HCC: A Hierarchical Co-Clustering Algorithm," in *Proc. 33rd ACM SIGIR Conference*, 2010.

[15] B. Long, X. Wu, Z. M. Zhang, and P. S. Yu: "Unsupervised learning on k-partite graphs," in *Proc. 12th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 317–326, 2006.

[16] G. W. Milligan and M. C. Cooper: "A study of the comparability of external criteria for hierarchical cluster analysis," *Multivariate Behavioral Research*, 21(4):441–458, 1986.

[17] R. R. Sokal and F. J. Rohlf: "The comparison of dendrograms by objective methods," *Taxon*, 11:33–40, 1962.

[18] A. Strehl and J. Ghosh: "Cluster ensembles - a knowledge reuse framework for combining multiple partitions," *J. Machine Learning Research*, 3:583–617, 2003.

[19] P. Symeonidis, M. M. Ruxanda, A. Nanopoulos, and Y. Manolopoulos: "Ternary semantic analysis of social tags for personalized music recommendation," in *Proc. 9th International Conference on Music Information Retrieval*, pp. 219–224, 2008.

[20] P. Tan, M. Steinbach, and V. Kumar: "Introduction to Data Mining," Addison-Wesley, 2005.

[21] D. Turnbull, L. Barrington, and G. Lanckriet: "Five approaches to collecting tags for music," in *Proc. 9th International Conference on Music Information Retrieval*, pp. 225–230, 2008.

[22] F. Wang, X. Wang, B. Shao, T. Li, and M. Ogihara: "Tag integrated multi-label music style classification with hypergraphs," in *Proc. 10th International Society for Music Information Retrieval*, pp. 363–368, 2009.

MUSIC EMOTION RECOGNITION: A STATE OF THE ART REVIEW

Youngmoo E. Kim, Erik M. Schmidt, Raymond Migneco, Brandon G. Morton
Patrick Richardson, Jeffrey Scott, Jacquelin A. Speck, and Douglas Turnbull[†]
Electrical and Computer Engineering, Drexel University
Computer Science, Ithaca College[†]
{ykim, eschmidt, rmigneco, bmorton,
patrickr, jjscott, jspeck}@drexel.edu
turnbull@ithaca.edu[†]

ABSTRACT

This paper surveys the state of the art in automatic emotion recognition in music. Music is oftentimes referred to as a "language of emotion" [1], and it is natural for us to categorize music in terms of its emotional associations. Myriad features, such as harmony, timbre, interpretation, and lyrics affect emotion, and the mood of a piece may also change over its duration. But in developing automated systems to organize music in terms of emotional content, we are faced with a problem that oftentimes lacks a well-defined answer; there may be considerable disagreement regarding the perception and interpretation of the emotions of a song or ambiguity within the piece itself. When compared to other music information retrieval tasks (e.g., genre identification), the identification of musical mood is still in its early stages, though it has received increasing attention in recent years. In this paper we explore a wide range of research in music emotion recognition, particularly focusing on methods that use contextual text information (e.g., websites, tags, and lyrics) and content-based approaches, as well as systems combining multiple feature domains.

1. INTRODUCTION

With the explosion of vast and easily-accessible digital music libraries over the past decade, there has been a rapid expansion of music information retrieval research towards automated systems for searching and organizing music and related data. Some common search and retrieval categories, such as artist or genre, are more easily quantified to a "correct" (or generally agreed-upon) answer and have received greater attention in music information retrieval research. But music itself is the expression of emotions, which can be highly subjective and difficult to quantify. Automatic recognition of emotions (or mood) [1] in music is still in its early stages, though it has received increasing attention in recent years. Determining the emotional content of music audio computationally is, by nature, a cross-disciplinary endeavor spanning not only signal processing and machine learning, but also requiring an understanding of auditory perception, psychology, and music theory.

Computational systems for music mood recognition may be based upon a model of emotion, although such representations remain an active topic of psychology research. Categorical and parametric models are supported through substantial prior research with human subjects, and these models will be described in further detail in the sections that follow. Both models are used in Music-IR systems, but the collection of "ground truth" emotion labels, regardless of the representation being used, remains a particularly challenging problem. A variety of efforts have been made towards efficient label collection, spanning a wide range of potential solutions, such as listener surveys, social tags, and data collection games. A review of methods for emotion data collection for music is also a subject of this paper.

The annual Music Information Research Evaluation eXchange (MIREX) is a community-based framework for formally evaluating Music-IR systems and algorithms [2], which included audio music mood classification as a task for the first time in 2007 [3]. The highest performing systems in this category demonstrate improvement each year using solely acoustic features (note that several of the systems were designed for genre classification and then appropriated to the mood classification task, as well). But emotion is not completely encapsulated within the audio alone (social context, for example, plays a prominent role), so approaches incorporating music metadata, such as tags and lyrics, are also reviewed here in detail.

For this state-of-the-art review of automatic emotion recognition in music, we first discuss some of the psychological research used in forming models of emotion, and then detail computational representations for emotion data. We present a general framework for emotion recognition that is subsequently applied to the different feature domains. We conclude with an overview of systems that combine multiple modalities of features.

[1] Emotion and mood are used interchangeably in the literature and in this paper.

2. PSYCHOLOGY RESEARCH ON EMOTION

Over the past half-century, there have been several important developments spanning multiple approaches for qualifying and quantifying emotions related to music. Such inquiry began well before the widespread availability of music recordings as a means of clinically repeatable musical stimuli (using musical scores), but recordings are the overwhelmingly dominant form of stimulus used in modern research studies of emotion. Although scores can provide a wealth of relevant information, score-reading ability is not universal, and our focus in this section and the overall paper shall be limited to music experienced through audition.

2.1 Perceptual considerations

When performing any measurement of emotion, from direct biophysical indicators to qualitative self-reports, one must also consider the source of emotion being measured. Many studies, using categorical or scalar/vector measurements, indicate the important distinction between one's perception of the emotion(s) *expressed* by music and the emotion(s) *induced* by music [4,5]. Both the emotional response and its report are subject to confound. Early studies of psychological response to environment, which consider the emotional weight of music both as a focal and distracting stimulus, found affective response to music can also be sensitive to the environment and contexts of listening [6]. Juslin and Luakka, in studying the distinctions between perceptions and inductions of emotion, have demonstrated that *both* can can be subject to not only the social context of the listening experience (such as audience and venue), but also personal motivation (i.e., music used for relaxation, stimulation, etc.) [5]. In the remainder of this paper, we will focus on systems that attempt to discern the emotion expressed, rather than induced, by music.

2.2 Perception of emotion across cultures

Cross-cultural studies of musical power suggest that there may be universal psychophysical and emotional cues that transcend language and acculturation [7]. Comparisons of tonal characteristics between Western 12-tone and Indian 24-tone music suggest certain universal mood-targeted melodic cues [8]. In a recent ethnomusicology study of people with no exposure to Western music (or culture), Mafa natives of Cameroon, categorized music examples into three categories of emotion in the same way as Westerners [9].

2.3 Representations of emotion

Music-IR systems tend to use either categorical descriptions or parametric models of emotion for classification or recognition. Each representation is supported by a large body of supporting psychology research.

2.3.1 Categorical psychometrics

Categorical approaches involve finding and organizing some set of emotional descriptors (tags) based on their relevance to some music in question. One of the earliest studies by Hevner, published in 1936, initially used 66 adjectives, which were then arranged into 8 groups [10]. While the adjectives used and their specific grouping and hierarchy have remained scrutinized and even disputed, many categorical studies conducted since Hevner's indicate such tagging can be intuitive and consistent, regardless of the listener's musical training [11, 12].

In a recent sequence of music-listening studies Zenter *et al.* reduced a set of 801 "general" emotional terms into a subset metric of 146 terms specific for music mood rating. Their studies, which involved rating music-specificity of words and testing words in lab and concert settings with casual and genre-aficionado listeners, revealed that the interpretation of these mood words varies between different genres of music [13].

The recent MIREX evaluations for automatic music mood classification have categorized songs into one of five mood clusters, shown in Table 1. The five categories were derived by performing clustering on a co-occurrence matrix of mood labels for popular music from the All Music Guide [2] [3].

Clusters	Mood Adjectives
Cluster 1	passionate, rousing, confident, boisterous, rowdy
Cluster 2	rollicking, cheerful, fun, sweet, amiable/good natured
Cluster 3	literate, poignant, wistful, bittersweet, autumnal, brooding
Cluster 4	humorous, silly, campy, quirky, whimsical, witty, wry
Cluster 5	aggressive, fiery, tense/anxious, intense, volatile, visceral

Table 1. Mood adjectives used in the MIREX Audio Mood Classification task [3].

2.3.2 Scalar/dimensional psychometrics

Other research suggests that mood can be scaled and measured by a continuum of descriptors or simple multi-dimensional metrics. Seminal work by Russell and Thayer in studying dimensions of arousal established a foundation upon which sets of mood descriptors may be organized into low-dimensional models. Most noted is the two-dimensional *Valence-Arousal* (V-A) space (See Figure 1), where emotions exist on a plane along independent axes of arousal (intensity), ranging high-to-low, and valence (an appraisal of polarity), ranging positive-to-negative [4]. The validity of this two-dimensional representation of emotions for a wide range of music has been confirmed in multiple studies [11, 14].

Some studies have expanded this approach to develop three-dimensional spatial metrics for comparative analysis of musical excerpts, although the semantic nature of the third dimension is subject to speculation and disagreement [17]. Other investigations of the V-A model itself

[2] All Music Guide: http://www.allmusic.com

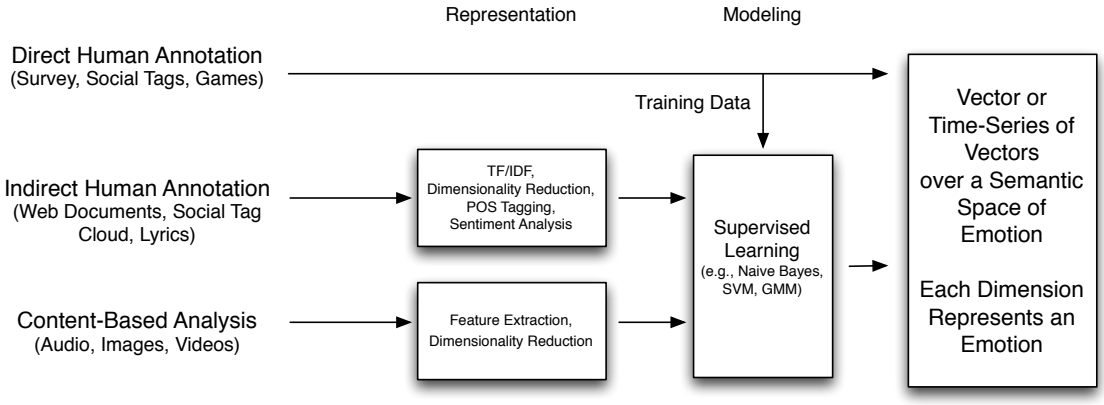


Figure 2. Overall model of emotion classification systems.

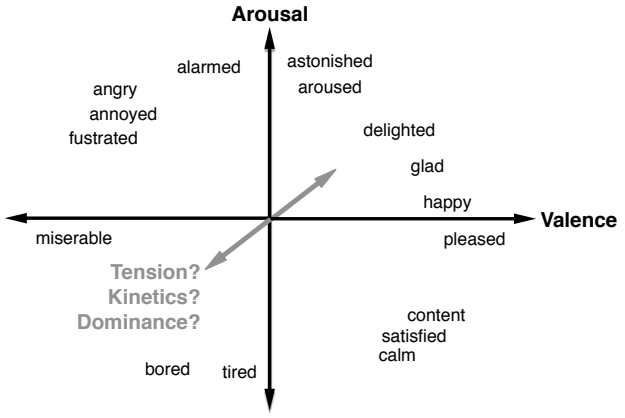


Figure 1. The Valence-Arousal space, labeled by Russell's direct circular projection of adjectives [4]. Includes semantic of projected third affect dimensions: "tension" [15], "kinetics" [16], "dominance" [6].

suggest evidence for separate channels of arousal (as originally proposed by Thayer) that are not elements of valence [18].

A related, but categorical, assessment tool for self-reported affect is the Positive and Negative Affect Schedule (PANAS), which asserts that all discrete emotions (and their associated labels) exist as incidences of positive or negative affect, similar to valence [19, 20]. In this case, however, positive and negative are treated as separate categories as opposed to the parametric approach of V-A.

3. FRAMEWORK FOR EMOTION RECOGNITION

Emotion recognition can be viewed as a multiclass-multilabel classification or regression problem where we try to annotate each music piece with a set of emotions. A *music piece* might be an entire song, a section of a song (e.g., chorus, verse), a fixed-length clip (e.g., 30-second song snipet), or a short-term segment (e.g., 1 second).

We will attempt to represent mood as either a single multi-dimensional vector or a time-series of vectors over a semantic space of emotions. That is, each dimension of a vector represents a single emotion (e.g., angry) or a bi-polar pair of emotions (e.g., positive/negative). The value of a dimension encodes the strength-of-semantic-association between the piece and the emotion. This is sometimes represented with a binary label to denote the presence or absence of the emotion, but more often represented as a real-valued score (e.g., Likert scale value, probability estimate). We will represent emotion as a time-series of vectors if, for example, we are attempting to track changes in emotional content over the duration of a piece.

We can estimate values of the emotion vector for a music piece in a number of ways using various forms of data. First, we can ask human listeners to evaluate the relevance of an emotion for a piece (see Section 4). This can be done, for example, using a survey, a social tagging mechanism, or an annotation game. We can also analyze forms of contextual meta-data in text form (see Section 5). This may include text-mining web-documents (e.g., artist biographies, album reviews) or a large collection of social tags (referred to as a *tag cloud*), and analyzing lyrics using natural language processing (e.g., sentiment analysis). We can also analyze the audio content using both signal processing and supervised machine learning to automatically annotate music pieces with emotions (see Section 6). Content-based methods can also be used to analyze other related forms of multimedia data such as music videos and promotional photographs [21]. Furthermore, multiple data sources, for example lyrics and audio, may be combined to determine the emotional content of music (see Section 7).

4. HUMAN ANNOTATION

A survey is a straightforward technique for collecting information about emotional content in music. All Music Guide has devoted considerable amounts of money, time and human resources to annotate their music databases with high-quality emotion tags. As such, they are unlikely to fully share this data with the Music-IR research community. To remedy this problem, Turnbull *et al.* collected the CAL500 data set of annotated music [22]. This data set contains one song from 500 unique artists each of which

have been manually annotated by a minimum of three non-expert reviewers using a vocabulary of 174 tags, of which 18 relate to different emotions. Trohidis *et al.* have also created a publicly available data set consisting of 593 songs each of which have been annotated using 6 emotions by 3 expert listeners [23].

A second approach to directly collect emotion annotations from human listeners involves social tagging. For example, Last.fm [3] is a music discovery website that allows users to contribute *social* tags through a text box in their audio player interface. By the beginning of 2007, their large base of 20 million monthly users have built up an unstructured vocabulary of 960,000 free-text tags and used it to annotate millions of songs [24]. Unlike AMG, Last.fm makes much of this data available to the public through their public APIs. While this data is a useful resource for the Music-IR community, Lamere and Celma point out that there are a number of problems with social tags: sparsity due to the cold-start problem and popularity bias, ad-hoc labeling techniques, multiple spellings of tags, malicious tagging, etc. [25]

4.1 Annotation Games

Traditional methods of data collection, such as the hiring of subjects, can be flawed, since labeling tasks are time-consuming, tedious, and expensive [26]. Recently, a significant amount of attention has been placed on the use of collaborative online games to collect such ground truth labels for difficult problems, so-called "Games With a Purpose". Several such games have been been proposed for the collection of music data, such as *MajorMiner* [27], *Listen Game* [28], and *TagATune* [29]. These implementations have primarily focused on the collection of descriptive labels for a relatively short audio clip. Screenshots of a few, select games are shown in Figure 3.

Figure 3. Examples of "Games With A Purpose" for ground truth data collection of musical data. Top left: *TagATune*. Right: *MoodSwings*. Bottom left: *Herd It*.

MoodSwings is another game for online collaborative annotation of emotions based on the arousal-valence model [30, 31]. In this game, players position their cursor within the V-A space while competing (and collaborating) with a partner player to annotate 30-second music clips where scoring is determined by the overlap between players' cursors (encouraging consensus and discouraging nonsense labels). Using a similar parametric representation, Bachorik *et al.* concluded that most music listeners require 8 seconds to evaluate the mood of a song, a delay that should be considered when collecting such time-varying annotations [32]. *Herd It* combines multiple types of music annotation games, including valence-arousal annotation of clips, descriptive labeling, and music trivia [33].

[3] Last.fm: http://www.last.fm

5. CONTEXTUAL TEXT INFORMATION

In this section, we discuss web documents, social tag clouds and lyrics as forms of textual information that can be analyzed in order to derive an emotional representation of music. Analysis of these data sources involves using techniques from both text-mining and natural language processing.

5.1 Web-Documents

Artist biographies, album reviews, and song reviews are rich sources of information about music. There are a number of research-based Music-IR systems that collect such documents from the Internet by querying search engines [34], monitoring MP3 blogs [35], or crawling a music website [36]. In all cases, Levy and Sandler point out that such web mined corpora can be *noisy* since some of the retrieved webpages will be irrelevant, and in addition, much of the text content on relevant webpages will be useless [37].

Most of the proposed web mining systems use a set of one or more documents associated with a song and convert them into a single document vector (e.g., Term Frequency-Inverse Document Frequency (TF-IDF) representation) [38, 39]. This *vector space* representation is then useful for a number of Music-IR tasks such as calculating music similarity [39] and indexing content for a text-based music retrieval system [38]. More recently, Knees *et al.* have proposed a promising new web mining technique called *relevance scoring* as an alternative to the vector space approaches [34].

5.2 Social Tags

Social tags have been used to accomplish such Music-IR tasks as genre and artist classification [40] as well as assessment of musical mood. Some tags such as "happy" and "sad" are clearly useful for emotion recognition and can be applied directly in information retrieval systems. Research has also shown that other tags, such as those related to genre and instrumentation, can also be useful for this task. Using ground truth mood labels from AMG, Bischoff *et al.* used social tag features from Last.fm to perform emotion classification based on MIREX mood categories as well as the V-A model [41]. They experimented with SVM, Logistic Regression, Random Forest, GMM, K-NN, Decision Trees, and Naive Bayes Multinomial classifiers, with

the Naive Bayes Multinomial classifier outperforming all other methods.

Other research involving the analysis of social tags has focused on clustering tags into distinct emotions and validating psychometric models. Making each tag a unique class would yield an unmanageable number of dimensions and fails to take into account the similarity of many terms used to describe musical moods. For example, the terms "bright", "joyful", "merry" and "cheerful" describe similar variants of happiness. Similarly, the tokens "gloomy", "mournful", "melancholy" and "depressing" are all related to sadness [10]. Recent efforts have demonstrated that favorable classification results can be obtained by grouping like descriptors into similarity clusters [14].

A number of approaches exist to arrange tags together into homogeneous groups. Manual clustering involves grouping of tags into pre-established mood categories, but given the size and variety of existing tag databases, this approach is not scalable. A straightforward automated clustering method, derived from the TF-IDF metric used often in text mining, looks for co-occurrences within the mood tags and forms clusters until no more co-occurrences are present. The co-occurrence method compares a threshold to the ratio of the number of songs associated with two tags to the minimum number of songs associated with either individual tag [42].

Another established method for automatically clustering labels is Latent Semantic Analysis (LSA), a natural language processing technique that reduces a term-document matrix to a lower rank approximation [43]. The term-document matrix in this case is a sparse matrix which describes the number of times each song is tagged with a given label. For a data set with several thousand songs and over 100 possible mood tags, the term-document matrix generated will have very high dimensionality. After some modifications, performing a singular value decomposition (SVD) on the modified term-document matrix yields the left and right singular vectors that represent the distance between terms and documents respectively. Initial work by Levy and Sandler applied a variation called Correspondence Analysis to a collection of Last.fm social tags to derive a semantic space for over 24,000 unique tags spanning 5700 tracks.

Tags can also be grouped by computing the cosine distance between each tag vector and using an unsupervised clustering method, such as Expectation Maximization (EM), to combine terms. In recent work, Laurier *et al.*, using a cost function to minimize the number of clusters to best represent over 400,000 unique tags, found that just four clusters yielded optimal clustering of the mood space [14]. The resulting clusters are somewhat aligned with Russell and Thayer's V-A model. Furthermore, both Levy & Sandler and Laurier *et al.* demonstrate that application of a self organizing map (SOM) algorithm to their derived semantic mood spaces yields a two-dimensional representation of mood consistent with the V-A model.

5.3 Emotion recognition from lyrics

In comparison to tag-based approaches, relatively little research has pursued the use of lyrics as the sole feature for emotion recognition (although lyrics have been used as features for artist similarity determination [44]). Lyric-based approaches are particularly difficult because feature extraction and schemes for emotional labeling of lyrics are non-trivial, especially when considering the complexities involved with disambiguating affect from text. Lyrics have also been used in combination with other features, work that is detailed in Section 7.

5.3.1 Lyrics feature selection

Establishing "ground-truth" labels describing the emotion of interconnected words is a significant challenge in lyric-based emotion recognition tasks. Mehrabian and Thayer proposed that environmental stimuli are linked to behavioral responses by emotional responses described by pleasure (valence), arousal and dominance (PAD) [6]. To this end, Bradley developed the Affective Norms for English Words (ANEW), which consists of a large set of words labeled with PAD values. A large number of subjects were used to label the words by indicating how the word made them feel in terms of relative happiness, excitement and situational control, which correspond to the pleasure, arousal and dominance dimensions, respectively. A distribution of the pleasure and arousal labels for words in ANEW show that they are well-distributed according to the V-A model [45]. Hu *et al.* used Bradley's ANEW to develop a translation called Affective Norms for Chinese Words (ANCW), operating under the assumption that the translated words carry the same affective meaning as their English counterparts [46].

Such affective dictionaries do not take into account multi-word structure. For lyrics features, most approaches employ a Bag of Words (BOW) approach, accounting for frequency of word usage across the corpus (e.g., TF-IDF), but not the specific order of words. One initial approach by Chen *et al.* utilized vector space model (VSM) features that consisted of all the words comprising the lyric [47]. However, more recently Xia *et al.* refined the feature vectors by only including sentiment and sentiment-related words, which they refer to as a *sentiment*-VSM (s-VSM) [48]. The focus on sentiment-related words is intended to capture the effect of modifying terms strengthening or weakening the primary sentiments of the lyrics and further reduces feature dimensionality.

5.3.2 Systems for emotion recognition from lyrics

Meyers' *Lyricator* system provides an emotional score for a song based on its lyrical content for the purpose of mood-based music exploration [49]. Feature extraction for Lyricator consists of obtaining PAD labels for the words comprising a songs lyric. Songs receive an overall emotional score in one of the four quadrants of the P-A model based on a summation of the PAD values for all the words in the lyric. While this approach is straightforward, it is not a machine learning system, nor does it make use of natural

language processing (NLP) to disambiguate emotion from lyrics.

Vector space approaches by Xia and Chen utilize support vector machine (SVM) classifiers for training and testing data [47, 48]. Xia's corpus consists of 2600 Chinese pop songs, 60% of which are hand-labeled as "light-hearted" and the remainder labeled as "heavy-hearted". Their sentiment-VSM feature set scores above 73% in terms of precision, recall, and F-1 measure.

Hu *et al.* utilize a fuzzy clustering technique to determine the main emotion from a lyric. The clusters are then weighted using grammatical information, which specify confidence and weights for individual sentences based on factors such as tense and inter-sentence relationships. The cluster with the greatest overall weight is considered the main emotion of the song and is characterized by its mean V-A values into one of the four quadrants of the V-A model. They demonstrate that their system outperforms the baseline Lyricator system in multiple categories [46].

Y. Yang explored the use of bi-gram BOW features, using pairs of words to examine the effects of negation terms (e.g., "not happy" differs from "happy") and Probabilistic LSA (PLSA) to model song topics using word frequencies [50]. Bi-gram BOW features demonstrated negligible increases in classification valence, but PLSA proved to be much more robust to a reduction of training set size.

6. CONTENT-BASED AUDIO ANALYSIS

Clearly, many human assessments of musical mood are derived from the audio itself (after all, tags are most often generated by people listening to the music). Contextual information for a music piece may be incomplete or missing entirely (i.e., for newly composed music). Given the rapid expansion of digital music libraries, including commercial databases with millions of songs, it is clear that manual annotation methods will not efficiently scale. Thus, the appeal of content-based systems is obvious and the recognition of emotions from audio has been a longstanding goal for the Music-IR research community (the corresponding MIREX task focused on systems driven by music audio).

6.1 Acoustic Features

Emotions can be influenced by such attributes as tempo, timbre, harmony, and loudness (to name only a few), and much prior work in Music-IR has been directed towards the development of informative acoustic features. Although some research has focused on searching for the most informative features for emotion classification, no dominant single feature has emerged. An overview of the most common acoustic features used for mood recognition is given in Table 2.

In searching for the most informative emotion and expressive features to extract from audio, Mion and De Poli investigated a system for feature selection and demonstrated it on an initial set of single-dimensional features, including intensity and spectral shape as well as several music-theoretic features [16]. Their system used sequential feature selection (SFS), followed by principal component analysis (PCA) on the subset to identify and remove redundant feature dimensions. The focus of their research, however, was monophonic instrument classification across nine classes spanning emotion and expression, as opposed to musical mixtures. Of the 17 tested features the most informative overall were found to be roughness, notes per second, attack time, and peak sound level.

Type	Features
Dynamics	RMS energy
Timbre	MFCCs, spectral shape, spectral contrast
Harmony	Roughness, harmonic change, key clarity, majorness
Register	Chromagram, chroma centroid and deviation
Rhythm	Rhythm strength, regularity, tempo, beat histograms
Articulation	Event density, attack slope, attack time

Table 2. Common acoustic feature types for emotion classification.

MacDorman *et al.* examined the ability of multiple acoustic features (sonogram, spectral histogram, periodicity histogram, fluctuation pattern, and mel-frequency cepstral coefficients–MFCCs [51, 52]) to predict pleasure and arousal ratings of music excerpts. They found all of these features to be better at predicting arousal than pleasure, and the best prediction results when all five features were used together [53].

Schmidt *et al.* investigated the use of multiple acoustic feature domains for music clips both in terms of individual performance as well as in combination in a feature fusion system [54, 55]. Their feature collection consisted of MFCCs, chroma [56], statistical spectrum descriptors (including centroid, flux, and rolloff, depicted in Figure 4) [57], and octave-based spectral contrast [58]. The highest performing individual features were spectral contrast and MFCCs, but again the best overall results were achieved using combinations of features.

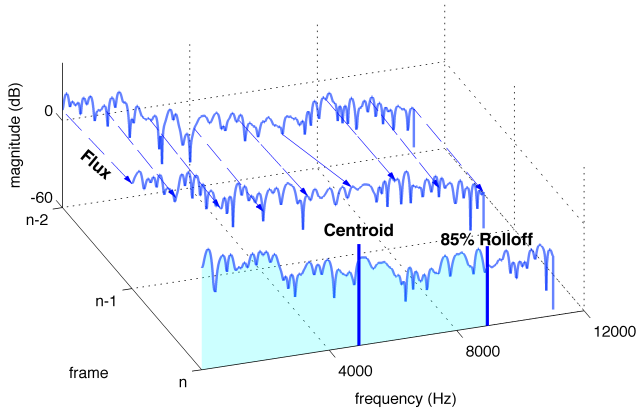


Figure 4. Graphical depiction of statistical spectrum descriptors.

Eerola *et al.*, also the developers of the open-source feature extraction code, *MIRtoolbox*,[4] have developed a specific subset of informative audio features for emotion. These features are aggregated from a wide range of domains including dynamics, timbre, harmony, register, rhythm, and articulation [15]. Other recent approaches have adopted a generalized approach to feature extraction, compiling multiple feature sets (resulting in high dimensional spaces) and employing dimensionality reduction techniques [50,59]. Regardless of feature fusion or dimensionality reduction methods, the most successful systems combine multiple acoustic feature types.

6.2 Audio-based systems

Audio content-based methods for emotion recognition use either a categorical or parametric model of emotion. The former results in a classification task and the latter in a regression task, with all recent systems employing one of these methods.

6.2.1 Categorical emotion classification

In one of the first publications on this topic, Li and Ogihara used acoustic features related to timbre, rhythm, and pitch to train support vector machines (SVMs) to classify music into one of 13 mood categories [60]. Using a hand-labeled library of 499 music clips (30-seconds each) from a variety of genres spanning ambient, classical, fusion, and jazz, they achieved an accuracy of 45%.

Lu *et al.* pursued mood detection and tracking using a similar variety of acoustic features including intensity, timbre, and rhythm [59]. Their classifier used Gaussian Mixture Models (GMMs) for the four principal mood quadrants on the V-A representation. The system was trained using a set of 800 classical music clips (from a data set of 250 pieces), each 20 seconds in duration, hand labeled to one of the 4 quadrants. Their system achieved an overall accuracy of 85%, although it is also unclear how the multiple clips extracted from the same recording were distributed between training and testing sets.

Proposing a guided scheme for music recommendation, Mandel *et al.* developed active learning systems, an approach that can provide recommendations based upon any musical context defined by the user [61]. To perform a playlist retrieval, the user would present the system with a set of "seed songs," or songs representing the class of playlist desired. The system uses this data, combined with verification data from the user, to construct a binary SVM classifier using MFCC features. When tested on 72 distinct moods from AMG labels, the system achieved a peak performance of 45.2%.

Skowronek *et al.* developed binary classifiers for each of 12 non-exclusive mood categories using a data set of 1059 song excerpts. Using features based on temporal modulation, tempo and rhythm, chroma and key information, and occurrences of percussive sound events they trained quadratic discriminant functions for each mood, with accuracy ranging from 77% (carefree-playful) to 91% (calming-soothing), depending on the category [62].

As mentioned in the introduction, MIREX first included audio music mood classification as a task in 2007 [3]. In 2007, Tzanetakis achieved the highest percentage correct (61.5%), using only MFCC, and spectral shape, centroid, and rolloff features with an SVM classifier [63]. The highest performing system in 2008 by Peeters demonstrated some improvement (63.7%) by introducing a much larger feature corpus including, MFCCs, Spectral Crest/Spectral Flatness, as well as a variety of chroma based measurements [64]. The system uses a GMM approach to classification, but first employs Inertia Ratio Maximization with Feature Space Projection (IRMFSP) to select the most informative 40 features for each task (in this case mood), and performs Linear Discriminant Analysis (LDA) for dimensionality reduction. In 2009, Cao and Li submitted a system that was a top performer in several categories, including mood classification (65.7%) [65]. Their system employs a "super vector" of low-level acoustic features, and employs a Gaussian Super Vector followed by Support Vector Machine (GSV-SVM). It's worth noting that the best performers in each of the three years of the evaluation were general systems designed to perform multiple MIREX tasks.

6.2.2 Parametric emotion regression

Recent work in music emotion prediction from audio has suggested that parametric regression approaches can outperform labeled classifications using equivalent features. Targeting the prediction of V-A coordinates from audio, Yang *et al.* introduced the use of regression for mapping high-dimensional acoustic features to the two-dimensional space [50]. Support vector regression (SVR) [66] and a variety of ensemble boosting algorithms, including AdaBoost.RT [67], were applied to the regression problem, and one ground-truth V-A label was collected for each of 195 music clips. As this work focused primarily on labeling and regression techniques, features were extracted using publicly available extraction tools such as PsySound [68] and Marsyas [69], totaling 114 feature dimensions. To reduce the data to a tractable number of dimensions principal component analysis (PCA) was applied prior to regression. This system achieves an R^2 (coefficient of determination) score of 0.58 for arousal and 0.28 for valence.

Schmidt *et al.* and Han *et al.* each began their investigation with a quantized representation of the V-A space and employed SVMs for classification [54, 70]. Citing unsatisfactory results (with Schmidt obtaining 50.2% on a four-way classification of V-A quadrants and Han obtaining 33% accuracy in an 11-class problem), both research teams moved to regression-based approaches. Han reformulated the problem using regression, mapping the projected results into the original mood categories, employing SVR and Gaussian Mixture Model (GMM) regression methods. Using 11 quantized categories with GMM regression they obtain a peak performance of 95% correct classification.

[4] MIRtoolbox: http://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox

Eerola *et al.* introduced the use of a three-dimensional emotion model for labeling music; fully committing themselves to regression [15]. In their work they investigated multiple regression approaches including Partial Least-Squares (PLS) regression, an approach that considers correlation between label dimensions. They achieve R^2 performance of 0.72, 0.85, and 0.79 for valence, activity, and tension, respectively, using PLS and also report peak R^2 prediction rates for 5 basic emotion classes (angry, scary, happy, sad, and tender) as ranging from 0.58 to 0.74.

Noting that quantization by quadrant is inconsistent with the continuous nature of their collected V-A labels, Schmidt *et al.* also approached the problem using both SVR and Multiple Linear Regression (MLR). Their highest performing system obtained 13.7% average error distance in the normalized V-A space [54]. In more recent work, Schmidt *et al.* have introduced the idea of modeling the collected human response labels to music in the V-A space as a parametrized stochastic distribution, noting that the labels for most popular music segments of a reasonably small size can be well represented by a single two-dimensional Gaussian [55]. They first perform parameter estimation in order to determine the ground truth parameters, $\mathcal{N}(\mu, \Sigma)$ and then employ MLR, PLS, and SVR to develop parameter prediction models. An example regression is shown in Figure 5, which employs only spectral contrast features and MLR.

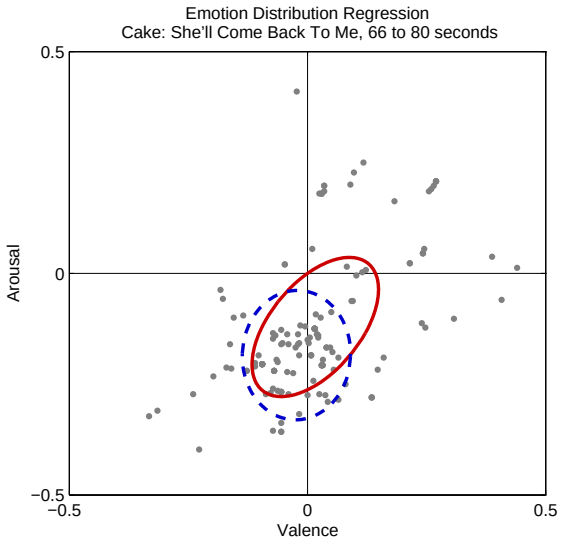


Figure 5. Collected V-A labels and distribution projections resulting from regression analysis. V-A labels: second-by-second labels per song (gray •), Σ of labels (solid red ellipse) and Σ of MLR projection from acoustic features (dashed blue ellipse) [55].

6.3 Emotion Recognition Over Time

As few other Music-IR tasks are subject to dynamic (time varying) "ground truth", it can be argued that accounting for the time varying nature of music is perhaps more important for emotion recognition than most other tasks. Because of this variation, systems relying on a single mood label to refer to an entire song or lengthy clip are subject to high classification uncertainty. Lu *et al.* pursued mood tracking across the four principal V-A quadrants, detecting mood changes at 1 second resolution. They report precision and recall for mood boundary detection at 84.1% and 81.5%, respectively on a corpus of 9 movements from classical works [59].

Using second-by-second V-A labels collected via the *MoodSwings* game, Schmidt *et al.* also investigated tracking the emotional content of music over time [54, 55]. Their time-varying analyses remain formulated as a regression to develop a mapping from short-time high-dimensional acoustic features to time-localized emotion space coordinates. Figure 6 shows the V-A projections of a song clip obtained from spectral contrast features and MLR prediction [54]. In this example, a 15-second clip has been broken down into three five-second segments (projections) demonstrating the overall movement in the V-A space. A version of their time-varying regression system is implemented back into *MoodSwings* as a simulated "AI" partner for single-player games.

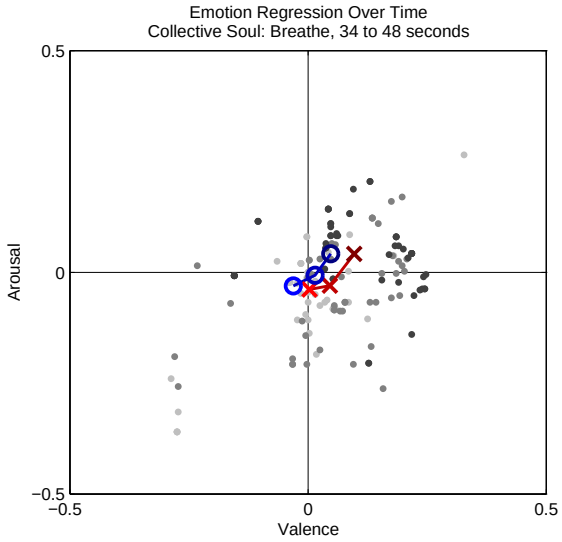


Figure 6. V-A labels and projections over time for a 15s segment (markers become darker over time): second-by-second labels per song (gray •), mean of the collected labels over 5-second intervals (red X), and projection from acoustic features in 5-second intervals (blue O) [54].

7. COMBINING MULTIPLE FEATURE DOMAINS

It is clear that some aspects of music data (e.g., social factors, such as "Christmas music", or songs considered to be "one hit wonders") are not revealed in the audio, and results using only acoustic (spectral) features for Music-IR tasks have led many to believe there is an upper limit to performance using these features alone [71]. This has led to a growing amount of research towards combining multiple feature domains to improve recognition in Music-IR systems. The earliest attempts at classification of emotion using multi-modal approaches (combining features from disparate domains) were applied to speech,

using combined analysis of audio and facial expressions [72–74]. These methods have inspired other multi-modal approaches to several other Music-IR classification tasks, such as genre recognition, but such combined approaches to classifying the emotional content of music have emerged only within the past few years.

7.1 Combining Audio & Lyrics

In spite of the importance of audio, some musical genres (e.g., "Christmas songs") are much easier to detect using text. This was the motivation for Neumayer's work utilizing multiple combinations of audio and textual features for musical genre classification [75]. Similarly, many of the studies described below are motivated by the idea that some emotions conveyed by music are better detected using a combination of audio and lyrics. Some systems report relatively modest performance gains, but it is often in tasks where baseline performance using audio features alone has been high. The most compelling results show, in certain cases, improvement over audio features alone, demonstrating that information contained within the two feature modalities can be highly complementary.

7.1.1 Systems combining audio & lyrics

The first system to employ both audio and lyrics for emotion classification (D. Yang and Lee) used lyric text and a wide range of audio features, such as beats per minute and 12 low-level MPEG-7 descriptors (e.g., spectral centroid, rolloff, and flux), on a set of 145 30-second song clips [76]. Each clip was hand-labeled with one of 11 emotional categories, based on PANAS labels [19, 20]. While strong correlations were found between particular lyrics and emotional categories (hostility, sadness, and guilt), the addition of lyrics increased classification performance by only 2.1% (82.8% vs. 80.7% using only audio) on their relatively small data set.

A more recent system (Y. Yang *et al.*) combined lyrics and audio features using a database of 1240 Chinese pop songs [50]. The songs were hand-labeled according to one of the four valence-arousal quadrants of the Russell-Thayer model, and acoustic features (MFCC and spectral descriptors) were computed for a 30-second segment drawn from the middle of each song. Lyrics (assumably from the entire song) were text-analyzed using the BOW approach. This work compared three different methods of combining audio and text features, obtaining the best results by first using audio and text separately to classify arousal and valence, respectively, using SVMs and then merging the results to determine a full V-A classification. On an 80/20 training-testing split of the data with 1000-fold cross-validation, acoustic features alone resulted in a baseline of 46.6% correct classification, while combining audio and lyrics yielded 57.1% accuracy (a relative performance increase of 21%).

Other recent work by Laurier, Grivolla, and Herrera explored the use of audio and multiple lyrics features to classify emotions of songs in the four-quadrant V-A space [77]. Songs were labeled using Last.fm tags and filtered using the lexical database WordNet-Affect to remove synonyms (with subsequent validation by human listeners). The corpus consisted of 1000 songs, with equal distribution across the four quadrants, and a combination of timbral, rhythmic, tonal, and temporal features were computed for each song. Three different methods, lyric similarity, LSA, and Language Model Differences (LMD) were investigated for deriving lyrics features. LMD compares the difference in frequency of terms between the language models of various mood categories and is used here to select the 100 most discriminative terms, resulting in significantly higher classification performance vs. LSA. The audio and text features were combined into a single vector for joint classification, improving performance over audio features alone in two quadrants by 5%: "happy" (81.5% to 86.8%) and "sad" (87.7% to 92.8%). The other quadrants were essentially unchanged, but already had high classification accuracy using only audio features: "relaxed" (91.4% to 91.7%) and "angry" (98.1% to 98.3%).

Hu *et al.* also combined audio and lyrics for emotion recognition on a relatively large database of nearly 3000 songs [78]. Eighteen emotion classes were formed based on social tags from Last.fm, using WordNet-Affect to remove tags irrelevant to emotion recognition and significant refinement by human judges. For lyrics features, this system uses a BOW approach with TF-IDF weighting. The BOW stemmed (prefixes and suffixes removed), TF-IDF weighted (BSTI) features are directly concatenated with 63 spectrum-derived audio features for training an SVM classifier. BSTI features were selected using different methods, including the LMD selection method (described in the previous system [77]), varying the number of feature dimensions to identify optimal performance. Interestingly, using BSTI (lyrics) features alone outperforms audio alone for 12 of the 18 mood classes. Audio + lyric approaches demonstrate the highest performance in recognizing 13 of the 18 mood classes (audio alone is highest for 3 classes, "happy", "upbeat", and "desire", while lyrics alone perform best for "grief" and "exciting"). Audio with LMD feature selection (63 dimensions, equivalent to the number of audio features) performed the highest in 5 categories ("calm", "sad", "anger", "confident", and "earnest") and improved performance in half of the cases where audio alone outperforms lyrics and vice versa, demonstrating the utility of combining features.

Recently, Schuller *et al.* investigated using audio features, lyrics, and metadata to automatically label music in a discretized version of the V-A space [79]. For a database of 2648 pop songs, each was rated by four listeners who selected one of 5 discrete labels. Their classification task, is ultimately reduced to two independent three-class problems. Their best performing system made use of feature selection algorithms and label filtering, achieving 64.1% and 60.9% on valence and arousal, respectively.

7.2 Combining Audio & Tags

Music-IR researchers have also focused on multimodal approaches incorporating tags and low-level audio features

for classification tasks. Turnbull *et al.* explore the problem of tag classification by combining semantic information from web documents, social tags and audio analysis on the CAL500 data set [80]. They compare a number of algorithms (e.g., Calibrated Score Averaging, RankBoost, Kernel Combination SVM) and find that multimodal approaches significantly outperform unimodal approaches.

Bischoff *et al.* combine social tag information and audio content-based analysis specifically for the task of emotion recognition [41]. For each of 4,737 songs, they collect social tags from Last.fm and generate a 240-dimensional audio feature vectors (including MFCCs, chroma features, and other spectral features). They then train a naive Bayes classifier for the social tags and an SVM for the audio feature vectors, combining them using a simple weighted combination approach. In one experiment, this approach is used to predict one of the five MIREX mood clusters [3]. In the second experiment, the approach is used to predict one of the four quadrants of the V-A mood space. Ground truth for each track is based on a manually-determined (ad-hoc) mapping between one of 178 mood tags to a MIREX mood cluster and to a V-A quadrant. In both experiments, the multimodal approach demonstrates better performance than either the tag-based and audio-based approaches alone.

7.3 Combining Audio & Images

Analysis of audio features of music in combination with associated images (album covers, artist photos, etc.) for Music-IR tasks has very recently sparked research interest. Dunker *et al.* investigated methods to combine music and image domains for mood classification [81]. Their work investigates both classifying audio paired with images in a multi-modal media tagging approach as well as trying to pair audio and images that have been tagged separately. More recently, Lībeks and Turnbull analyze promotional photographs of artists using content-based image annotation [21]. Although they label these artists with genre tags (provided by Last.fm), it would be straightforward to adapt their approach to use emotion tags.

8. CONCLUSION

Recognizing musical mood remains a challenging problem primarily due to the inherent ambiguities of human emotions. Though research on this topic is not as mature as some other Music-IR tasks, it is clear that rapid progress is being made. In the past 5 years, the performance of automated systems for music emotion recognition using a wide range of annotated and content-based features (and multimodal feature combinations) have advanced significantly. As with many Music-IR tasks open problems remain at all levels, from emotional representations and annotation methods to feature selection and machine learning.

While significant advances have been made, the most accurate systems thus far achieve predictions through large-scale machine learning algorithms operating on vast feature sets, sometimes spanning multiple domains, applied to relatively short musical selections. Oftentimes, this approach reveals little in terms of the underlying forces driving the perception of musical emotion (e.g., varying contributions of features) and, in particular, how emotions in music change over time. In the future, we anticipate further collaborations between Music-IR researchers, psychologists, and neuroscientists, which may lead to a greater understanding of not only mood within music, but human emotions in general. Furthermore, it is clear that individuals perceive emotions within music differently. Given the multiple existing approaches for modeling the ambiguities of musical mood, a truly personalized system would likely need to incorporate some level of individual profiling to adjust its predictions.

This paper has provided a broad survey of the state of the art, highlighting many promising directions for further research. As attention to this problem increases, it is our hope that the progress of this research will continue to accelerate in the near future.

9. REFERENCES

[1] C. C. Pratt, *Music as the language of emotion*. The Library of Congress, December 1950.

[2] J. S. Downie, "The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research," *Acoustical Science and Technology*, vol. 29, no. 4, pp. 247–255, 2008.

[3] X. Hu, J. Downie, C. Laurier, M. Bay, and A. Ehmann, "The 2007 MIREX audio mood classification task: Lessons learned," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Philadelphia, PA, 2008.

[4] J. A. Russell, "A circumspect model of affect," *Journal of Psychology and Social Psychology*, vol. 39, no. 6, p. 1161, 1980.

[5] P. Juslin and P. Luakka, "Expression, perception, and induction of musical emotions: A review and questionnaire study of everyday listening," *Journal of New Music Research*, vol. 33, no. 3, p. 217, 2004.

[6] A. Mehrabian and J. A. Russell, *An Approach to Environmental Psychology*. MIT Press, 1974.

[7] C. McKay, "Emotion and music: Inherent responses and the importance of empirical cross-cultural research," Course Paper. McGill University, 2002.

[8] L.-L. Balkwill and W. F. Thompson, "A cross-cultural investigation of the perception of emotion in music," *Music Perception*, vol. 17, no. 1, pp. 43–64, 1999.

[9] T. Fritz, S. Jentschke, N. Gosselin, D. Sammler, I. Peretz, R. Turner, A. D. Friederici, and S. Koelsch, "Universal recognition of three basic emotions in music," *Current Biology*, vol. 19, no. 7, pp. 573 – 576, 2009.

[10] K. Hevner, "Experimental studies of the elements of expression in music," *American Journal of Psychology*, vol. 48, no. 2, pp. 246–267, 1936.

[11] P. Juslin and J. Sloboda, *Music and Emotion: theory and research*. Oxford Univ. Press, 2001.

[12] E. Schubert, "Update of the Hevner adjective checklist," *Perceptual and Motor Skills*, vol. 96, pp. 1117–1122, 2003.

[13] M. Zentner, D. Grandjean, and K. R. Scherer, "Emotions evoked by the sound of music: Characterization, classification, and measurement." *Emotion*, vol. 8, p. 494, 2008.

[14] C. Laurier, M. Sordo, J. Serra, and P. Herrera, "Music mood representation from social tags," in *Proc. of the Intl. Society for Music Information Conf.*, Kobe, Japan, 2009.

[15] T. Eerola, O. Lartillot, and P. Toiviainen, "Prediction of multidimensional emotional ratings in music from audio using multivariate regression models," in *Proc. of the Intl. Society for Music Information Conf.*, Kobe, Japan, 2009.

[16] L. Mion and G. D. Poli, "Score-independent audio features for description of music expression," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 2, pp. 458–466, 2008.

[17] E. Bigand, "Multidimensional scaling of emotional responses to music: The effect of musical expertise and of the duration of the excerpts," *Cognition and Emotion*, vol. 19, no. 8, p. 1113, 2005.

[18] U. Schimmack and R. Reisenzein, "Experiencing activation: energetic arousal and tense arousal are not mixtures of valence and activation," *Emotion*, vol. 2, no. 4, p. 412, 2002.

[19] D. Watson and L. Clark, *PANAS-X: Manual for the Positive and Negative Affect Schedule*, expanded form ed., University of Iowa, 1994.

[20] A. Tellegen, D. Watson, and L. A. Clark, "On the dimensional and hierarchical structure of affect," *Psychological Science*, vol. 10, no. 4, pp. 297–303, 1999.

[21] J. Lībeks and D. Turnbull, "Exploring artist image using content-based analysis of promotional photos," in *Proc. of the Int. Computer Music Conf.*, 2010.

[22] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic annotation and retrieval of music and sound effects," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 2, 2008.

[23] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotion," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Philadelphia, PA, 2008.

[24] F. Miller, M. Stiksel, and R. Jones, "Last.fm in numbers," *Last.fm press material*, February 2008.

[25] P. Lamere and O. Celma, "Music recommendation tutorial notes," ISMIR Tutorial, September 2007.

[26] L. von Ahn, "Games with a purpose," *Computer*, vol. 39, no. 6, pp. 92–94, 2006.

[27] M. I. Mandel and D. P. W. Ellis, "A web-based game for collecting music metadata," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Vienna, Austria, 2007, pp. 365–366.

[28] D. Turnbull, R. Liu, L. Barrington, and G. Lanckriet, "A game-based approach for collecting semantic annotations of music," in *Proc. Intl. Conf. on Music Information Retrieval*, Vienna, Austria, 2007, pp. 535–538.

[29] E. L. M. Law, L. von Ahn, R. B. Dannenberg, and M. Crawford, "TagATune: A game for music and sound annotation," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Vienna, Austria, 2007.

[30] Y. Kim, E. Schmidt, and L. Emelle, "Moodswings: A collaborative game for music mood label collection," in *Proc. Intl. Conf. on Music Information Retrieval*, Philadelphia, PA, September 2008.

[31] B. G. Morton, J. A. Speck, E. M. Schmidt, and Y. E. Kim, "Improving music emotion labeling using human computation," in *HCOMP 2010: Proc. of the ACM SIGKDD Workshop on Human Computation*, Washington, D.C., 2010.

[32] J. P. Bachorik, M. Bangert, P. Loui, K. Larke, J. Berger, R. Rowe, and G. Schlaug, "Emotion in motion: Investigating the time-course of emotional judgments of musical stimuli," *Music Perception*, vol. 26, no. 4, pp. 355–364, April 2009.

[33] L. Barrington, D. Turnbull, D. O'Malley, and G. Lanckriet, "User-centered design of a social game to tag music," *ACM KDD Workshop on Human Computation*, 2009.

[34] P. Knees, T. Pohle, M. Schedl, D. Schnitzer, and K. Seyerlehner, *A Document-Centered Approach to a Natural Language Music Search Engine*. Springer Berlin / Heidelberg, 2008, pp. 627–631.

[35] O. Celma, P. Cano, and P. Herrera, "Search sounds: An audio crawler focused on weblogs," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Victoria, Canada, 2006.

[36] B. Whitman and D. Ellis, "Automatic record reviews," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Barcelona, Spain, 2004, pp. 470–477.

[37] M. Levy and M. Sandler, "A semantic space for music derived from social tags," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Vienna, Austria, 2007.

[38] P. Knees, T. Pohle, M. Schedl, and G. Widmer, "A music search engine built upon audio-based and web-based similarity measures," in *ACM SIGIR*, 2007.

[39] B. Whitman, "Combining musical and cultural features for intelligent style detection," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Paris, France, 2002, pp. 47–52.

[40] P. Knees, E. Pampalk, and G. Widmer, "Artist classification with web-based data," in *Proc. of the Intl. Symposium on Music Information Retrieval*, Barcelona, Spain, 2004, pp. 517–524.

[41] K. Bischoff, C. S. Firan, R. Paiu, W. Nejdl, C. Laurier, and M. Sordo, "Music mood and theme classification-a hybrid approach," in *Proc. of the Intl. Society for Music Information Retrieval Conf.*, Kobe, Japan, 2009.

[42] K. Bischoff, C. Firan, W. Nejdl, and R. Paiu, "How do you feel about 'Dancing Queen'?: Deriving mood & theme annotations from user tags," in *Proc. of the ACM/IEEE-CS Joint Conf. on Digital Libraries*, Austin, TX, Jan 2009.

[43] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, pp. 391–407, 1990.

[44] B. Logan, A. Kositsky, and P. Moreno, "Semantic analysis of song lyrics," in *Proc. IEEE Intl. Conf. on Multimedia and Expo*, Taipei, Taiwan, June 27-30 2004.

[45] M. M. Bradley and P. J. Lang, "Affective norms for English words (ANEW)," The NIMH Center for the Study of Emotion and Attention,University of Florida, Tech. Rep., 1999.

[46] Y. Hu, X. Chen, and D. Yang, "Lyric-based song emotion detection with affective lexicon and fuzzy clustering method," in *Proc. of the Intl. Society for Music Information Conf.*, Kobe, Japan, 2009.

[47] R. H. Chen, Z. L. Xu, Z. X. Zhang, and F. Z. Luo, "Content based music emotion analysis and recognition," in *Proc. of the Intl. Workshop on Computer Music and Audio Technology*, 2006.

[48] Y. Xia, L. Wang, K. Wong, and M. Xu, "Sentiment vector space model for lyric-based song sentiment classification," in *Proc. of the Association for Computational Linguistics*. Columbus, Ohio, U.S.A: ACL-08, 2008, pp. 133–136.

[49] O. C. Meyers, "A mood-based music classification and exploration system," Master's thesis, Massachusetts Institute of Technology, June 2007.

[50] Y.-H. Yang, Y.-C. Lin, H.-T. Cheng, I.-B. Liao, Y.-C. Ho, and H. Chen, *Advances in Multimedia Information Processing - PCM 2008*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, December 2008, ch. 8, pp. 70–79.

[51] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357– 366, 1980.

[52] B. Logan, "Mel frequency cepstral coefficients for music modeling," in *Proc. of the Intl. Symposium on Music Information Retrieval*, Plymouth, MA, September 2000.

[53] K. F. MacDorman, S. Ough, and C.-C. Ho, "Automatic emotion prediction of song excerpts: Index construction, algorithm design, and empirical comparison," *Journal of New Music Research*, vol. 36, no. 4, pp. 281–299, 2007.

[54] E. M. Schmidt, D. Turnbull, and Y. E. Kim, "Feature selection for content-based, time-varying musical emotion regression," in *MIR '10: Proc. of the Intl. Conf. on Multimedia Information Retrieval*, Philadelphia, PA, 2010, pp. 267–274.

[55] E. M. Schmidt and Y. E. Kim, "Prediction of time-varying musical mood distributions from audio," in *Proc. of the Int. Society for Music Information Conf.*, Utrecht, Netherlands, August 2010.

[56] T. Fujishima, "Realtime chord recognition of musical sound: A system using Common Lisp music." in *Proc. of the Intl. Computer Music Conf.*, 1999.

[57] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293– 302, 2002.

[58] D. Jiang, L. Lu, H. Zhang, J. Tao, and L. Cai, "Music type classification by spectral contrast feature," in *Proc. Intl. Conf. on Multimedia and Expo*, vol. 1, 2002, pp. 113–116.

[59] L. Lu, D. Liu, and H. J. Zhang, "Automatic mood detection and tracking of music audio signals," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 1, pp. 5–18, 2006.

[60] T. Li and M. Ogihara, "Detecting emotion in music," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Baltimore, MD, October 2003.

[61] M. I. Mandel, G. E. Poliner, and D. P. W. Ellis, "Support vector machine active learning for music retrieval," *Multimedia Systems*, vol. 12, no. 1, pp. 3–13, Aug 2006.

[62] J. Skowronek, M. McKinney, and S. van de Par, "A demonstrator for automatic music mood estimation," in *Proc. Intl. Conf. on Music Information Retrieval*, Vienna, Austria, 2007.

[63] G. Tzanetakis, "Marsyas submissions to MIREX 2007," MIREX 2007.

[64] G. Peeters, "A generic training and classification system for MIREX08 classification tasks: Audio music mood, audio genre, audio artist and audio tag," MIREX 2008.

[65] C. Cao and M. Li, "Thinkit's submissions for MIREX2009 audio music classification and similarity tasks." ISMIR, MIREX 2009.

[66] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[67] D. Shrestha and D. Solomatine, "Experiments with AdaBoost.RT, an improved boosting scheme for regression," *Neural Computation*, vol. 18, no. 7, pp. 1678–1710, 2006.

[68] D. Cabrera, S. Ferguson, and E. Schubert, "Psysound3: software for acoustical and psychoacoustical analysis of sound recordings," in *Proc. of the Intl. Conf. on Auditory Display*, Montreal, Canada, June 26-29 2007, pp. 356–363.

[69] G. Tzanetakis and P. Cook, "Marsyas: a framework for audio analysis," *Organized Sound*, vol. 4, no. 3, pp. 169–175, 1999.

[70] B. Han, S. Rho, R. B. Dannenberg, and E. Hwang, "SMERS: Music emotion recognition using support vector regression," in *Proc. of the Intl. Society for Music Information Conf.*, Kobe, Japan, 2009.

[71] J.-J. Aucouturier and F. Pachet, "Improving timbre similarity: How high is the sky?" *Journal of Negative Results in Speech and Audio Sciences*, vol. 1, no. 1, 2004.

[72] J. Cohn and G. Katz, "Bimodal expression of emotion by face and voice," in *ACM Intl. Multimedia Conf.*, 1998.

[73] L. Silva and P. Ng, "Bimodal emotion recognition," in *Proc. of the IEEE Intl. Conf. on Automatic Face and Gesture Recognition*, 2000, pp. 332–335.

[74] Z. Zeng, M. Pantic, G. Roisman, and T. Huang, "A survey of affect recognition methods: Audio, visual, and spontaneous expressions," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 1, 2009, pp. 39–58.

[75] R. Neumayer and A. Rauber, "Integration of text and audio features for genre classification in music information retrieval," in *Proc. of the European Conf. on Information Retrieval*, 2007.

[76] D. Yang and W. Lee, "Disambiguating music emotion using software agents," in *Proc. of the Intl. Conf. on Music Information Retrieval*. Barcelona, Spain: Universitat Pompeu Fabra, October 2004.

[77] C. Laurier, J. Grivolla, and P. Herrera, "Multimodal music mood classification using audio and lyrics," in *Proc. of the Intl. Conf. on Machine Learning and Applications*. Universitat Pompeu Fabra, 2008, pp. 1–6.

[78] X. Hu, J. S. Downie, and A. F. Ehmann, "Lyric text mining in music mood classification," in *Proc. of the Intl. Society for Music Information Retrieval Conf.*, Kobe, Japan, 2009.

[79] B. Schuller, J. Dorfner, and G. Rigoll, "Determination of nonprototypical valence and arousal in popular music: Features and performances," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2010, pp. 1–20, Jan 2010.

[80] D. Turnbull, L. Barrington, M.Yazdani, and G. Lanckriet, "Combining audio content and social context for semantic music discovery," *ACM SIGIR*, 2009.

[81] P. Dunker, S. Nowak, A. Begau, and C. Lanz, "Content-based mood classification for photos and music: A generic multimodal classification framework and evaluation approach," in *Proc. of the Intl. Conf. on Multimedia Information Retrieval*. ACM, 2008.

LOOKING THROUGH THE "GLASS CEILING": A CONCEPTUAL FRAMEWORK FOR THE PROBLEMS OF SPECTRAL SIMILARITY

Ioannis Karydis
Dept. of Informatics
Ionian University
Greece
karydis@ionio.gr

Miloš Radovanović
Faculty of Science
University of Novi Sad
Serbia
radacha@dmi.rs

Alexandros Nanopoulos
Inst. of Computer Science
University of Hildesheim
Germany
nanopoulos@ismll.de

Mirjana Ivanović
Faculty of Science
University of Novi Sad
Serbia
mira@dmi.rs

ABSTRACT

Spectral similarity measures have been shown to exhibit good performance in several Music Information Retrieval (MIR) applications. They are also known, however, to possess several undesirable properties, namely allowing the existence of hub songs (songs which frequently appear in nearest neighbor lists of other songs), "orphans" (songs which practically never appear), and difficulties in distinguishing the farthest from the nearest neighbor due to the concentration effect caused by high dimensionality of data space. In this paper we develop a conceptual framework that allows connecting all three undesired properties. We show that hubs and "orphans" are *expected* to appear in high-dimensional data spaces, and relate the cause of their appearance with the concentration property of distance / similarity measures. We verify our conclusions on real music data, examining groups of frames generated by Gaussian Mixture Models (GMMs), considering two similarity measures: Earth Mover's Distance (EMD) in combination with Kullback-Leibler (KL) divergence, and Monte Carlo (MC) sampling. The proposed framework can be useful to MIR researchers to address problems of spectral similarity, understand their fundamental origins, and thus be able to develop more robust methods for their remedy.

1. INTRODUCTION

The notion of audio-based music similarity is generally considered to be complex, subjective, and context dependent [13]. However, spectral similarity measures [2,10] are receiving a growing interest and have been shown to exhibit good performance in several Music Information Retrieval (MIR) applications [14]. These measures describe aspects related to timbre and model the "global sound" of a musical signal based on features called Mel Frequency Cepstrum Coefficients (MFCCs).

Despite the advantages of spectral similarity measures, related research has also reported a number of undesired properties, summarized as follows:

- The existence of hub songs (also called "always similar") [2], which are close neighbors of many other pieces to which they hold no perceptual similarity, thus increasing the rate of false positives.
- The existence of "orphans" (also called "never similar") [12], which are songs that rarely become close neighbors of any other piece (despite possibly having perceptual similarity to a large number of songs), increasing therefore the rate of true negatives.
- Songs are represented in a feature space whose dimensionality is determined by the number of features (MFCCs). As the dimensionality grows, it is becoming hard to identify meaningful nearest neighbors, since all songs tend to be at nearly the same distance from each other. This property was identified in other research areas [1,5,8], but was also examined in the contexts of MIR [6] and audio speech data (based on MFCCs) [18].

These undesired properties constitute some of the main causes for the empirically demonstrated upper limit for the performance of spectral similarity measures, referred to as the "glass ceiling" [2]. Recent research has focused mostly on the amelioration of hubness (the attribute of being a hub song), by proposing techniques for normalizing the distances between songs in a way that reduces the influence of hubs [11,14,15], whereas other works [9,17,19] developed measures that try to avoid hubness.

Our motivation is to develop a conceptual framework that allows for relating all three aforementioned undesired properties, and explains the mechanisms that create them. Aucouturier and Pachet [3] have focused on the analysis of hubness and concluded that the creation of *homogenized* models (i.e., models that ignore the least likely mixture components) are responsible for creating hubs. Despite this detailed conclusion, our emphasis is to disclose a more fundamental reason that causes all three undesired properties, which is the high dimensionality of the feature space that originates from the need to use multiple MFCCs in order to capture the "global sound".

A conjecture about the role of high dimensionality has been stated by Berenzweig in his thesis [4]. This conjecture was drawn from two synthetic data sets that follow multivariate Gaussian distributions. In particular, a main conclusion of this thesis [4, page 99] was: "First, hubness

seems to be a natural consequence of the curse of dimensionality, at least for the points distributed according to a Gaussian in a space up to 32 dimensions. In high dimensions these points tend to be spread around the shell of the space with very few points near the center; this implies that any points that do happen to remain near the center will be extreme hubs." However, this work neither generalized the conclusion to real audio music data, nor even to other settings besides simple synthetic data. More importantly, it did not provide a clear explanation of the mechanism that creates hubness, leaving this question unresolved [4, question 1 on page 99]. A more thorough examination of hubness has been performed by Radovanović et al. [16], wherein using real vector-space data the authors relate hubness with the *intrinsic* dimensionality of data, and show that in (intrinsically) high-dimensional data sets hubs tend to appear in the proximity of cluster centers. However, [16] focused primarily on general vector spaces and l_p norms, with the results not directly applicable for MIR purposes.

In this paper, we propose a conceptual framework to provide a clear explanation of the mechanism that creates hubness and show that hubs are *expected* to appear in high-dimensional spaces (i.e., they are not points that just *happen* to remain near the center). Moreover, the framework helps to understand the connection between all three undesired properties: hubs, "orphans" and the problem of finding meaningful neighbors. Also, our conclusions are verified with real audio music data. The proposed conceptual framework can be useful to MIR researchers to address the problems of spectral similarity in relation to each other, understand their fundamental reasons, and thus be able to develop more robust methods for their remedy.

In the rest of this article, Section 2 reviews related work. Section 3 presents the proposed framework, whereas Section 4 provides empirical evidence for verifying the conclusions of the proposed framework in the MIR domain. Finally, Section 5 concludes the paper.

2. RELATED WORK

Research by Aucouturier and Pachet [3] focuses on the nature and causes of hub songs. They propose methods to detect hubs and infer that hubs are distributed along a scale-free distribution. Moreover, in their work they deduce that hubs neither exist due to the spectral features, nor are they a property of a feature representation or a given modeling strategy but rather tend to occur with any type of model that uses agglomeration of multiple frames of a sound texture. Furthermore, they establish that hubness is not a characteristic of certain songs, as different algorithms distribute hubs differently in a database. In addition, they also establish that the class of algorithms studied is irrelevant to hubs which appear only for data with a given amount of heterogeneity. Finally, they conclude that hubness can be localized to certain parts of the distribution of a song.

Berenzweig [4] offers insight as to the understanding of hubs and arrives to the conclusion that their existence is a natural result of the curse of dimensionality. Additionally, in his work, the possibility of hubs being derived from similarity to a universal background is proven invalid through experimentation, that is by showing that the discriminating power of specific frames is not ameliorated by weighting based on their shared information.

As mentioned in Section 1, unlike Aucouturier and Pachet [3] our motivation is to provide high dimensionality as a more fundamental reason for hubness, and for the other two undesired properties (see Section 1) as well. Differentiating also from the work of Berenzweig [4], we develop a thorough conceptual framework that links all three properties and clearly explains the mechanisms through which they originate.

Other related work includes techniques to ameliorate or try to avoid hubness [9, 11, 14, 15, 17, 19]. We hope that our proposed framework will assist in this direction, by allowing MIR researchers to further analyze the causes of all examined undesired properties (not just hubness), and develop solutions that take into account all of them.

3. PROPOSED CONCEPTUAL FRAMEWORK

We commence the description of the proposed conceptual framework by demonstrating the property of *concentration* [8] that is exhibited by spectral similarity measures due to the high dimensionality of their feature space. Next, we examine how the generation of hubs and "orphans" can be explained as a consequence of high dimensionality, in relation with the concentration phenomenon. The conclusions (in this and the following section) are verified with real data. Since our description involves some empirical measurements, we start by detailing the employed settings.

3.1 Settings for Empirical Measurements

We focus on two characteristic spectral similarity measures that have been widely used in related research. The first is proposed by Logan and Salomon [10], and uses Earth Mover's Distance (EMD) in combination with Kullback-Leibler (KL) divergence to compute the distance between groups of frames generated by a GMM approach. The second is proposed by Aucouturier and Pachet [2], and uses Monte Carlo (MC) sampling to measure the similarities of GMMs. Henceforth, the first measure is denoted as EMD-KL, whereas the second as GMM-MC. We based our implementation of both measures on the MA toolbox [12].

The main parameter examined for both measures is the number of MFCCs, denoted as d, which corresponds to the dimensionality of the feature space, since each frame of the audio signal is mapped to a point in a d-dimensional space.

The default value for the number of clusters used in Gaussian mixture modeling performed by GMM-MC and EMD-KL is equal to one, since measures like G1C [12] have demonstrated the efficacy of this option. In our experiments we also examined other values in order to verify that this factor does not have any impact on our conclusions. For brevity we therefore present results only for the default number of clusters. Regarding other parameters (sampling rate, frame size, etc.), empirical evidence in related work [3], and our experiments, indicates that they are not related with the examined issues. For this reason, we

keep the remaining parameters at the default values from the MA toolbox, which correspond to commonly used values in most related works. The sampling frequency of the input wav file is 11025, the FFT window size is 512 samples and the FFT window hop size is 256 samples. Finally, we used the MIREX'04 audio collection for the reasons that it is widely used by the MIR community, has been involved in all related work (e.g., [3]), and is publicly available, allowing reproducibility of the presented results.

3.2 The Property of Concentration

The concentration property of a distance (similarity) measure refers to the tendency of all points in a high-dimensional feature space to be almost equally distant from (similar to) each other. Concentration has been studied in vector spaces for Euclidean distance and other l_p norms (including fractional distances) [1, 8], and was also analyzed in the MIR context [6], but not explicitly for the spectral similarity measures we are focusing on in this work.

To ease comprehension, we first consider iid Gaussian random data with d-dimensional points, the components of which are independently drawn from $\mathcal{N}(0, 1)$. Figure 1 illustrates the concentration of Euclidean distance that incurs with high dimensionality. In particular, the figure depicts from top to bottom: the maximal observed value, mean value plus one standard deviation, the mean value, mean value minus one standard deviation, and minimal observed value of distances of all data points to the origin. It can be seen that the mean value steadily increases with increasing dimensionality, while the standard deviation remains constant, and that the observed minimal and maximal values become constrained to a narrow interval as dimensionality increases. This means that distances of all points to the origin (i.e., the norms) become very similar to each other as dimensionality increases, with the same behavior also extending to all pairwise distances within a data set [8], thus making it harder to distinguish between the farthest and the nearest neighbor in high dimensions [1]. It is important to note that this property of distances was proven to hold for *any* iid random data distribution [8].

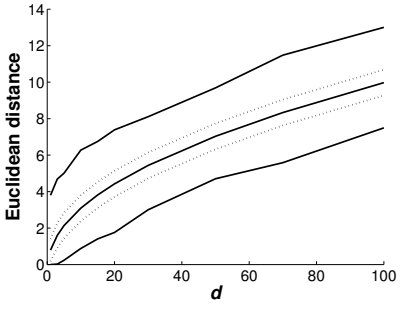


Figure 1. Concentration of Euclidean distance for iid Gaussian random data ($n = 2000$ points).

Concentration is usually expressed as the ratio between some measure of spread, like the standard deviation, and some measure of magnitude, like the mean value, of distances of all points in a data set to some arbitrary reference point [1, 8]. If this ratio converges to 0 as dimensionality goes to infinity, it is said that distances concentrate. Regarding the aforementioned synthetic data set, if the origin is selected as the reference point, Fig. 2 illustrates the ratio between the standard deviation (σ_{dist}) and mean value (μ_{dist}) of distances of all points to the origin, showing that it tends to 0 as dimensionality increases. Moreover, theoretical results by François et al. [8] indicate that the same asymptotic behavior holds with any other point selected as the reference, and also extends to all pairwise distances in a data set. In the case of Euclidean distance, the mean value μ_{dist} behaves asymptotically as $\sqrt{d}$, whereas the standard deviation σ_{dist} remains asymptotically constant.

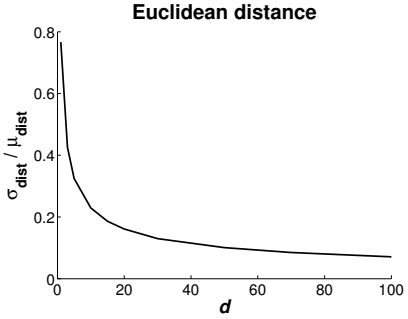


Figure 2. Ratio between the standard deviation and mean value of the distribution of distances to the origin, for iid Gaussian random data.

The property of concentration can be used to explain the generation of hubs and "orphans" as follows. Existing theoretical and empirical results [1, 5] specify that concentration can be viewed as causing points in a high-dimensional data set to approximately lie on the surface of a hypersphere centered at an arbitrary point. In addition, further results [7, 8], as illustrated in Fig. 1, indicate that the distribution of distances to any reference point has a finite variance for any given d. If the data distribution center is taken to be the reference point (as, coincidentally, is the case in the previously used random data example), it can be said that it is *expected* for points closer to the data center to exist in high dimensions, since for any finite d the standard deviation of the distribution of distances to the data set center is finite (in this case, constant). However, in higher dimensions, the points closer to the center have the tendency to become closer, on average, to all other points, thus having increased probability of becoming hubs by being near neighbors of many remaining points. On the other hand, it is also expected to have a non-negligible number of points farther from the data set center. Consequently, these points, which are the "orphans", become farther from all other points and more difficult to be near neighbors of any other point. The aforementioned connection between concentration and hubs/"orphans" has been initially proposed by Radovanović et al. [16] using experimentation on iid uniform random data, in contrast to Gaussian random data which pertains to real musical data utilized in this paper. Still, it is important to note that based on the results in [8], the reasoning followed in [16] can be applied to *any* iid random data distribution.

In order to clarify the mechanism through which the "centrality" of a point close to the data center, i.e. its proximity to all other points, becomes amplified in high dimensions, let us return to the iid Gaussian random data example and observe as reference points (instead of the origin) two points with the following properties: point x_0, which is at the expected distance from the data center for a given dimensionality d, and point x_2, which is two standard deviations closer to the center than x_0.[1] Next, we compute the distributions of distances of x_0 and x_2 to all other points, and denote the means of these distributions μ_{x_0} and μ_{x_2}, respectively. Figure 3 plots the difference between μ_{x_0} and μ_{x_2}. It can be seen that this difference increases with increasing dimensionality, meaning that x_2 becomes closer, on average, to all other points, solely by virtue of increasing dimensionality. According to the results by François et al. [8], verified by our empirical measurements, both μ_{x_0} and μ_{x_2} asymptotically behave as $\sqrt{d}$. However, convergence does not occur at the same *speed*, giving rise to the differences shown in Fig. 3 which ultimately result in the emergence of hubs. Similar arguments hold when, for example, point x_2 is taken to be two standard deviations *farther* from the center than x_0, explaining the formation of "orphans."

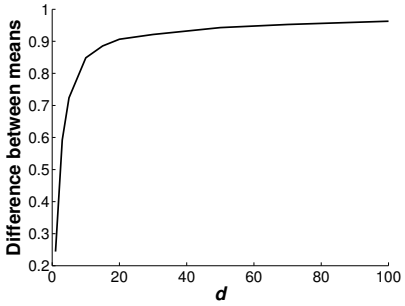


Figure 3. Difference between means of distance distributions to points at analogous positions wrt the data center, for iid Gaussian random data.

3.3 Concentration in Real Audio Data

The preceding discussion suggests that concentration can help explain the generation of hubs and "orphans" for audio music data and spectral similarity measures. However, the aforementioned conclusions were drawn with respect to distances between single points, whereas spectral similarities are computed between Gaussian Mixture Models (GMMs). Moreover, in this case of spectral similarity, we can only consider pairwise distances and not a point of reference like the center. Therefore, to examine the concentration of spectral similarity, we perform the following measurement. We vary the dimensionality of the feature space (number of MFCCs). For each examined dimensionality, we define for each song in the examined collection its *neighbor-range* by computing the difference between the distances to its farthest and nearest neighbor. To characterize the distribution of neighbor-range for each dimensionality, as explained before, we follow the approach of related work [8] and compute the ratio between the standard deviation σ_{range} and the mean μ_{range} of the neighbor-ranges of all songs.

Figure 4 illustrates this ratio as a function of dimensionality, for the two examined spectral similarity measures. The fact that the examined ratio reduces with increasing dimensionality indicates that the neighbor-range is narrowing as dimensionality increases, making it more difficult to separate the closest from the farthest neighbor. Asymptotically, as dimensionality tends to infinity, the examined ratio is expected to become equal to zero, denoting that the closest and the farthest neighbor of any song will tend to coincide. This means that asymptotically all points tend to become equidistant. However, for the high but finite dimensionality values used in MIR applications, the standard deviation in the examined ratio will be small but nonzero, causing an analogous amplification of "centrality" to that described above.

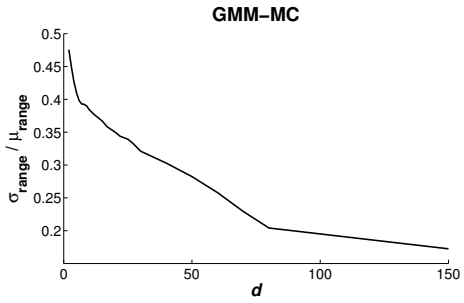


Figure 4. Ratio of standard deviation and mean value of neighbor-range as a function of dimensionality.

4. DIMENSIONALITY: ROLE & IMPACT ON MIR

The proposed framework allows for explaining the emergence of hubs and "orphans" principally as a consequence of high dimensionality of the feature space, in relation with the concentration it incurs. In this section we will verify with the examined real audio music data that the high dimensionality of the feature space creates the hubs and "orphans" according to the mechanism described in the previous section. Additionally, we examine the resulting impact of high dimensionality on MIR specific applications.

4.1 Verifying the Role of Dimensionality

Let $N_k(x)$ denote the number of *k-occurrences* of each song x in a collection, i.e., the number of times x occurs among the k nearest neighbors of other songs. Following the approach in [16], we express the asymmetry of N_k (i.e., the skewness) using the standardized third moment:

$$S_{N_k} = \mathrm{E}(N_k - \mu_{N_k})^3 / \sigma_{N_k}^3 ,$$

where μ_{N_k} and σ_{N_k} are the mean and standard deviation of N_k.[2] For the examined real audio data, and for the two

[1] Roughly speaking, for every d point x_0 has the same "probability" of occurrence with regards to the distance from the data distribution center, and the same can be said for x_2.

[2] An S_{N_k} value of 0 signifies that the distribution of N_k is symmetrical, positive (negative) values indicate skewness to the right (left).

similarity measures GMM-MC and EMD-KL, Figures 5 and 6, respectively, depict skewness as a function of dimensionality (three different values of k are examined in each case). As dimensionality increases, the increase of skewness, for all values of k, indicates that the distribution of N_k becomes considerably skewed to the right, resulting in the emergence of *hubs*, i.e., points which appear in many more k-nearest neighbor lists than other points.

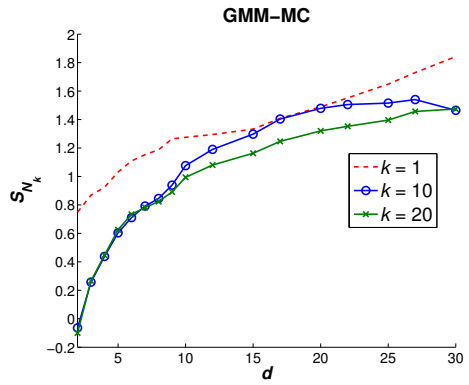


Figure 5. Skewness of N_k as a function of dimensionality for the GMM-MC measure.

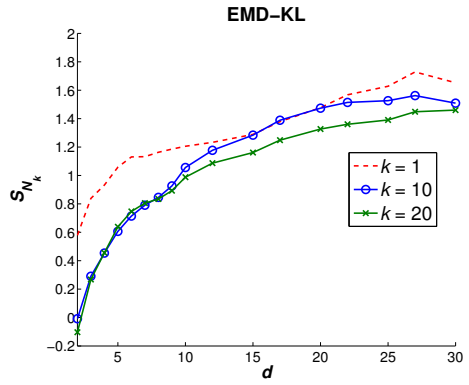


Figure 6. Skewness of N_k as a function of dimensionality for the EMD-KL measure.

Regarding "orphans," Figure 7 depicts, for both examined measures, the number of songs with N_k equal to zero as a function of dimensionality (k was set to 10). As expected, with increasing dimensionality, the number of such songs increases, demonstrating the relation of "orphans" with dimensionality.

All the above results show that dimensionality is the fundamental reason for the emergence of hubs and "orphans." In the next section we examine the relevance of this result to the objectives of MIR.

4.2 Impact on MIR

In order to study how our main conclusion, concerning the role of high dimensionality, affects MIR applications, we rely on external labels, such as the genre, to characterize the similarity between songs. This assumption is being widely applied in MIR (e.g., in MIREX contests), since results in related work [13] indicate that this assumption is reasonable.

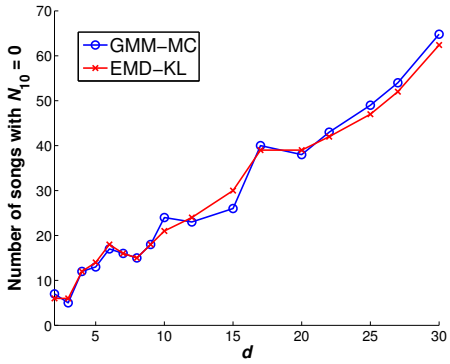


Figure 7. Number of songs with N_k equal to 0 ($k = 10$).

In this context, we measure the impact of hubs by examining the number of times they mismatch in label with the songs to which they are close neighbors. In this sense, we can measure how much of the total error (in terms of label mismatches) can be attributed to hubs. Figure 8 depicts, as a function of dimensionality, the fraction of total error due to the 10% of the strongest hubs, i.e., songs with the largest N_k values (k was set to 1). For low dimensionality values, when according to previous measurements hubs are not strong, their responsibility for the total error is much smaller compared to the case of larger dimensionality. It is worth to note that, for the largest examined dimensionality value, the strongest hubs are responsible for about 90% of the total error.

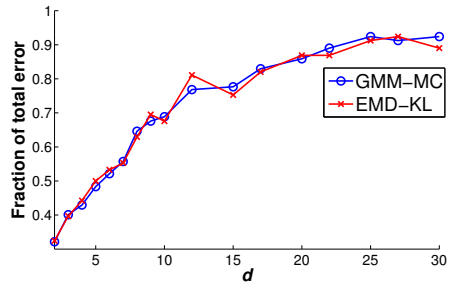


Figure 8. Fraction of total error due to 10% of the strongest hubs ($k = 1$).

5. CONCLUSIONS

In this paper we propose a conceptual framework that relates the known shortcomings of spectral similarity measures for music data: the existence of hubs, "orphans" and the distance concentration phenomenon, with the high dimensionality of underlying data space. The framework presents a unifying view of the three examined problems of music similarity measures, offering an explanation of their fundamental origins, which will hopefully help MIR researchers develop robust methods for their remedy.

The issue of high dimensionality is significant for spectral similarity measures because small dimensionality usually leads to poor discriminating capability, while high dimensionality produces the described negative effects pertaining to hubs, "orphans," and distance concentration.

In future work we will take into consideration more theoretical aspects of the hub/"orphan" properties and similarity concentration, providing a sounder theoretical backing for the described relationships. Furthermore, it would be interesting to examine why some approaches [9, 17, 19] tend to produce less hubs, by relating to the intrinsic dimensionality of the space they produce (i.e., to consider the proposed framework in explaining these approaches and understanding their reported properties). We also plan to conduct an extended experimental evaluation involving more similarity measures and data collections, giving more precise quantification of relationships between high dimensionality and the aforementioned properties of (spectral) similarity measures. Finally, we will develop novel mitigation methods for the problems induced by the existence of excessive hubs and "orphans". In particular, we will examine machine learning methods that apply correction to spectral similarity measures in order to take into account that retrieval error may not be distributed uniformly (as exemplified in Section 4.2), thus focusing on hubs as the main source of error, and in order to enable "orphans" to participate more prominently as nearest neighbors.

Acknowledgments. Alexandros Nanopoulos gratefully acknowledges the partial co-funding of his work through the EC FP7 project *MyMedia* (www.mymediaproject.org/) under the grant agreement no. 215006.

6. REFERENCES

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Proc. 8th Int. Conf. on Database Theory (ICDT)*, pages 420–434, 2001.

[2] J.-J. Aucouturier and F. Pachet. Improving timbre similarity: How high's the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[3] J.-J. Aucouturier and F. Pachet. A scale-free distribution of false positives for a large class of audio similarity measures. *Pattern Recognition*, 41(1):272–284, 2008.

[4] A. Berenzweig. *Anchors and Hubs in Audio-based Music Similarity*. PhD thesis, Columbia University, New York, USA, 2007.

[5] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proc. 7th Int. Conf. on Database Theory (ICDT)*, pages 217–235, 1999.

[6] M. Casey, C. Rhodes, and M. Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech and Language Processing*, 16(5):1015–1028, 2008.

[7] P. Demartines. *Analyse de Données par Réseaux de Neurones Auto-Organisés*. PhD thesis, Institut Nat'l Polytechnique de Grenoble, Grenoble, France, 1994.

[8] D. François, V. Wertz, and M. Verleysen. The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):873–886, 2007.

[9] M. Hoffman, D. M. Blei, and P. R. Cook. Content-based musical similarity computation using the hierarchical dirichlet process. In *Proc. 9th Int. Conf. on Music Information Retrieval (ISMIR)*, pages 349–354, 2008.

[10] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)*, pages 952–955, 2001.

[11] E. Pampalk. Speeding up music similarity. In *1st Annual Music Information Retrieval Evaluation eXchange (MIREX)*, 2005.

[12] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, Vienna, Austria, 2006.

[13] E. Pampalk, A. Flexer, and G. Widmer. Improvements of audio-based music similarity and genre classification. In *Proc. 6th Int. Conf. on Music Information Retrieval (ISMIR)*, pages 628–633, 2005.

[14] T. Pohle. Post processing music similarity computations. In *2nd Annual Music Information Retrieval Evaluation eXchange (MIREX)*, 2006.

[15] T. Pohle, M. Schedl, P. Knees, and G. Widmer. Automatically adapting the structure of audio similarity spaces. In *Proc. 1st Workshop on Learning the Semantics of Audio Signals (LSAS)*, pages 76–86, 2006.

[16] M. Radovanović, A. Nanopoulos, and M. Ivanović. Nearest neighbors in high-dimensional data: The emergence and influence of hubs. In *Proc. 26th Int. Conf. on Machine Learning (IMCL)*, pages 865–872, 2009.

[17] K. Seyerlehner, G. Widmer, and P. Knees. Frame level audio similarity - a codebook approach. In *Proc. 11th Int. Conf. on Digital Audio Effects (DAFx)*, pages 349–356, 2008.

[18] Z. Wang, W. Dong, W. Josephson, Q. Lv, M. Charikar, and K. Li. Sizing sketches: A rank-based analysis for similarity search. In *Proc. ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems*, pages 157–168, 2007.

[19] K. West, S. Cox, and P. Lamere. Incorporating machine-learning into music similarity estimation. In *Proc. 1st ACM Workshop on Audio and Music Computing Multimedia*, pages 89–96, 2006.

ON THE APPLICABILITY OF PEER-TO-PEER DATA IN MUSIC INFORMATION RETRIEVAL RESEARCH

Noam Koenigstein[1], Yuval Shavitt[1], Ela Weinsberg[2], and Udi Weinsberg[1]

[1]School of Electrical Engineering, Tel-Aviv University
[2]Dept. of Industrial Engineering, Tel-Aviv University

ABSTRACT

Peer-to-Peer (p2p) networks are being increasingly adopted as an invaluable resource for various music information retrieval (MIR) tasks, including music similarity, recommendation and trend prediction. However, these networks are usually extremely large and noisy, which raises doubts regarding the ability to actually extract sufficiently accurate information.

This paper evaluates the applicability of using data originating from p2p networks for MIR research, focusing on partial crawling, inherent noise and localization of songs and search queries. These aspects are quantified using songs collected from the Gnutella p2p network. We show that the power-law nature of the network makes it relatively easy to capture an accurate view of the main-streams using relatively little effort. However, some applications, like trend prediction, mandate collection of the data from the "long tail", hence a much more exhaustive crawl is needed. Furthermore, we present techniques for overcoming noise originating from user generated content and for filtering non informative data, while minimizing information loss.

1. INTRODUCTION

Peer-to-Peer (p2p) networks are being increasingly adopted as an invaluable resource for various music information retrieval (MIR) tasks [11], including music and user similarity [3, 5, 15], recommendation [16], ranking [9, 14], and even trend prediction [10, 12]. Various information can be extracted from a p2p network, including files shared by users, search queries, and spatial and temporal changes that take place in the network.

This type of information is traditionally extracted from server-based services, such as Last.FM and Yahoo! Music services. Web based services have the potential to provide a complete view of their data, either by commercial agreements or by crawling using a centralized interface. However, while p2p networks have practically unbounded growth potential, web-based services are often limited in size. This limitation is problematic for collaborative filtering techniques, that were shown to out-perform content based approaches, given that the dataset used is sufficiently comprehensive [2].

Another advantage of p2p datasets over traditional datasets is the availability of information, mitigating the need for agreements with website operators and various restrictions they pose on the data usage. Due to their decentralized nature and open protocols, p2p networks are a source for independent large scale data collection.

Despite all their advantages, p2p networks are quite complex, making the collection of a comprehensive dataset far from being trivial, and in some cases practically unfeasible. First, p2p networks have high user churn, causing users to constantly connect and disconnect from the network, being unavailable for changing periods. Second, users in p2p networks often do not expose their shared data in order to maintain high privacy and security measures, therefore disabling the ability to collect information about their shared folders. Finally, users often delete shared files to save space making it invisible to a crawl being performed after the deletion.

It is yet unknown to what extent data that is collected from large-scale p2p networks actually represents sufficiently accurate information in general, and particularly from a MIR point of view. The objective of this work is to bridge this gap by analyzing the efficiency and extent of crawling required for obtaining accurate information for various MIR tasks. We focus on sufficient sampling in a sparse domain with a long tail of content distribution.

In order to understand how well the crawl captures the underlying network, we perform an empirical study of the utility of an exhaustive crawl relative to a partial crawl. When discussing shared files, a partial crawl means that not all users are reached, resulting in not all songs being collected. Additionally, in the context of search queries, only a portion of the queries are collected since it is practically impossible to collect all queries in a fully distributed p2p network.

We find that some of the graphs modeling p2p network data exhibit a power-law [1] distribution. This distribution indicates that collecting the majority of popular files and extracting accurate information for the main-streams, is relatively easy. By collecting the high degree nodes, which are easily reached, one may extract an abundance of information regarding the core of the network. On the other hand, reaching more exotic niches or following small changes in trendy hits mandates a more through crawl with significantly higher collection effort, as the collection process must visit the long "tail" of the distribution. Furthermore, we observe the existence of geographic locality

of both files and queries, indicating that applications that are geographic aware (like trend prediction [10]), mandate sampling from different geographic locations.

2. MEASUREMENT INFRASTRUCTURE

This section details the architecture of the measurement system developed to crawl the Gnutella [13] network and collect queries in a distributed manner. Although the exact details are adapted to comply to the Gnutella architecture and protocols, similar techniques can be applied to other p2p networks. As such is Apollo [17], an efficient framework for crawling the BitTorrent p2p network, which uses a centralized server that collects trackers, enabling it to reach related peers and extract files that peer hold.

2.1 Crawling and Browsing Shared Files

Our crawler traverses the network as a graph, similar to the method used by web crawlers. The crawler employs a highly parallel technique by spawning numerous threads that attempt connecting to a set of provided IP addresses. Gnutella nodes implement a "Ping-Pong" protocol [18] used for discovering nodes, where a node that receives a "Ping" request replies with information about additional nodes that it is connected to. The resulting IP addresses are fed to the worker threads for further crawling.

Crawling *dynamic* p2p networks never reaches a complete stop, as clients constantly connect and disconnect from the network, and the crawler keeps discovering new IP address. This means that an "exhaustive" crawl is a matter of definition, i.e., deciding when to stop the crawling process. We use two stop conditions that define how exhaustive the crawl will be: (a) a time constraint, and (b) reaching a low rate of newly discovered nodes.

At the early stages of a crawl with an initial set of roughly 100k target node IP addresses, the rate of newly discovered nodes increases dramatically and can typically reach over 300,000 new clients per minute. As the crawling process proceeds, discovery rate slows down until it reaches a few hundreds per minute. At this point, the network is almost fully covered, and the newly discovered nodes are mostly the ones that have joined the network only after the crawling operation started, whereas some of the crawled nodes already left the network.

The browsing operation closely follows the crawling results and operates in parallel. The browsing threads collect active node IP addresses reported from the crawler, and use a "Query" message [18] to retrieve information about the files that a node shares. Notice that some nodes ignore these queries due to privacy or bandwidth considerations.

Although we do not download any of the files, the task of browsing millions of shared folders is bandwidth intensive, and requires high bandwidth Internet access. Our deployed system uses a 1 Gbit/s network card connected to two 2.5 Gbit/s STM-16 lines. Despite our fast connection, browsing takes about 24 hours, whereas crawling ends after roughly 1 hour. More details on our crawler can be found in [8].

2.2 Collection of Queries

The process of query collection is highly dependant on the search paradigm that the p2p protocol employs. Fully distributed searches, like in Gnutella, propagate search strings between peers. While it is possible to capture a large quantity of queries by deploying several hundred "listening" nodes, it is not trivial to determine the queries origin (required for geographical location). The basic problem in identifying the origin of captured queries is that queries do not in general carry their origin IP address. Most peers are "hidden" behind a firewall, hence it is impossible to send the results directly to them. Instead, proxy peers that have routable IP address (in Gnutella – Ultrapeers) are used to convey the information for firewalled peers.

In cases where geographic query analysis is required, this usage of ultrapeers causes a difficulty to match a peer to its geographic location, since the correlation between an ultrapeer geographic location and its attached peers is low [7, 10]. The authors suggest a method to determine queries origin IP, based on the number of hops they traversed. Our geographical resolution is based on a similar technique. More details can be found in [7].

Alternatively, some networks, e.g. BitTorrent, employ a centralized search engine, which is operated by web servers. Users search for content using a web interface, find "trackers" and use them to find active peers that hold the requested files. This technique greatly simplifies the data collection effort. However, it mandates cooperation of web site operators, which are often reluctant to share information on their users.

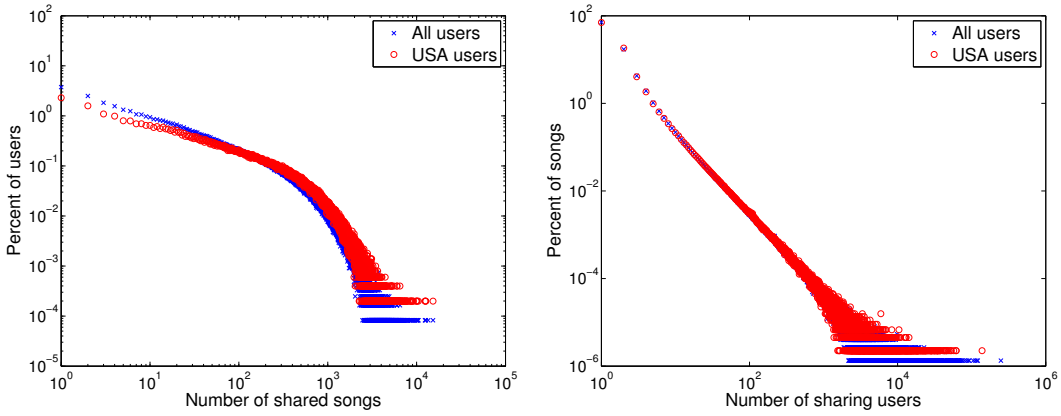
3. SONG DISTRIBUTION

We start by looking at the distribution of songs per users, considering all users in the dataset, and only users that are located in the US. For this end, we consider only music files shared by users, namely files ending with .mp3, .wav, .mid and .wma.

Figure 1(a) shows that all users and US-only users exhibit a power-law [1] distribution, with a very strong cutoff around the middle of the plot. This indicates that the vast majority of users share less than 300 songs, whereas only several thousands of users share more than 1k songs. Notice that only a few users share more than 10k music files, while over 45k users share only a single song.

These two extremes present different aspects of "noise". The few "heavy sharers" are not informative, while the latter simply contribute to a very long tail that is hardly insightful. In collaborative filtering for example, users that share only one song, contribute no similarity relations, while users that share songs from thousands of artists, are likely to "pollute" the dataset with false relations, since they appear to "like everything".

Next, we look a the popularity distribution of songs, by counting the number of different users that share each song. Figure 1(b) shows a clear power-law distribution containing a long tail, which is attributed to popular songs that are shared by many users. The percentage of popular songs shared by many users is slightly lower in the US, yet the two distributions mostly overlap. There are a few extremely popular songs shared by more than 10k users,



(a) Shared songs (b) Song popularity

Figure 1. Distributions of shared songs and song popularity

while the vast majority of the songs are shared by less than 1k users. Considering that there are over 1.2 million users in the dataset, songs that are shared by less than 1k users are quite borderline for being considered "popular".

The figure also shows that there are many songs that are shared by less than 100 users, which means that reaching them, or recording their relations to other songs, requires an extensive crawl. These songs surely do not represent any significant main-stream artist or genre, but for the purpose of detecting hypes or finding small communities with very specific preferences, reaching these users and collecting these songs might be important.

Given these distributions, we wish to evaluate the number of new songs that are discovered as more users are being crawled. Two difficulties arise regarding this analysis. The first is the way to identify that two files are indeed the same song, and for this end, either the file hash or the metadata can be utilized. Using the file hash is straightforward, as every file in the p2p network has a file hash, taken over its content. However, there can be many slightly different copies of the same file, each with a different hash, mostly due to different disc ripping software or originating song. On the other hand, metadata is often missing and contains different spelling mistakes, hence it can also result in incorrect identification of similar songs.

Therefore, we used both file hash and metadata techniques for identification of unique songs. First, we just use the file hash as the song id, and when hashes are exactly the same, we consider them as the same song. When using metadata, we consider only songs that have both "title" (name of the song) and "artist" tags, and use their concatenation as the song id.

The second difficulty is that many songs appear only once in the dataset. These are mostly attributed to faulty music file (not necessarily songs) that were uploaded by users and are of no interest to other users, rendering these files is useless for most MIR tasks. Therefore, we first counted the number of occurrences of each song, once using file hash and then using metadata, and removed all the songs that have only a single appearance in the dataset.

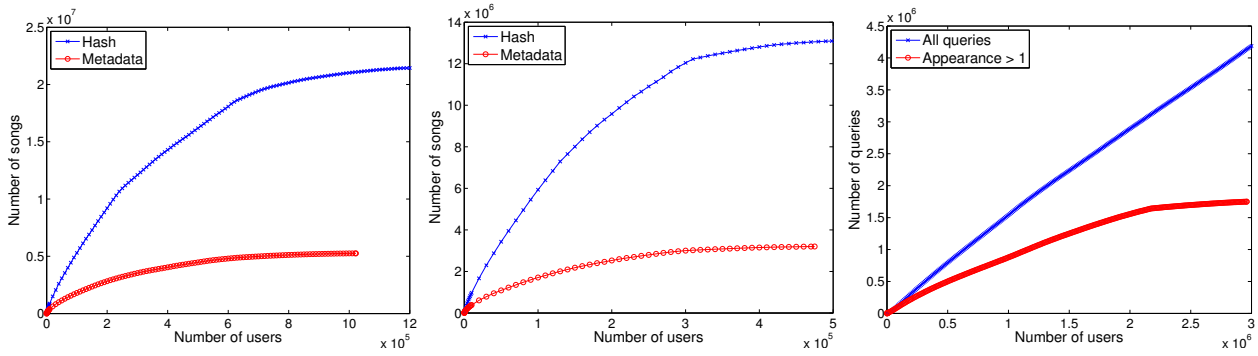
Figure 2 shows the number of unique songs per number of crawled users, showing all users and US-based users. The order of users was randomly selected to reduce spatial bias. Both figures show a converging trend, indicating that the utility of crawling many users decreases. Furthermore, the convergence witnessed when using metadata seems faster than when using file hashes, indicating that file hashes are more noisy than the metadata. Alternatively, this can be attributed to the observation that roughly 75% of the songs did not have both title and artist tags present, hence were removed from the analysis. This contributes to the reduction of "noise" resulting in a more stable and quickly converging set of songs.

The convergence observed when crawling only US-based users (56% of the users) seems slower than when crawling all users. Looking back at the distribution of songs per users (Figure 1(a)) shows that US users tend to have more songs, i.e., higher percentage of users have more than 200 shared songs. This explains the slower convergence, since the probability that a user will contribute previously unseen songs is higher. The number of songs seen in US-based shared folders is only half of the entire world wide collection. However the usage of metadata over hash for songs identification seem to be as effective as in the general case, since the percentage of noise reduction remains the same.

4. SONG CONNECTIVITY

Item-based recommendation systems require an estimation of the distance between songs. This task is often performed using expensive content-based similarity. However, song similarity can be efficiently extracted from p2p networks, by transforming the bipartite graph that connects users to songs into a 1-mode song-similarity graph, where the weight of a link w_{ij} between two songs i and j is the number of users that have both songs in their shared folders.

In this analysis we wish to obtain a stable similarity graph, therefore we do more processing to identify unique songs. Similar to the previous analysis, all the songs that have hash value that appeared only once are removed. We then group together all file hashes that relate to identical metadata value (artist and title). At this stage we have grouped together different digital versions of the same song. Accounting for spelling mistakes is achieved by grouping together artist and title values that have a small edit distance [19] (counting insert, delete and substitute). The dis-



(a) All users (b) US-based users (c) Unique queries

Figure 2. Number of unique songs (using file hash and metadata) and unique queries vs. number of users crawled

tance threshold is determined by a function of the string's length. Representative metadata values are chosen using majority voting. Finally, after this aggregation, all songs that have less than 7 occurrences are removed. This value is a tradeoff between filtering and memory consumption, taking only 3bits of memory for each song.

This unification of songs reduced the number of unique songs from over 21 million when using hashes and 5 million when using metadata to 530k songs, meaning only 2.5% of the songs using hash and roughly 10% of the songs using metadata. Although this technique can slightly overfilter, it successfully overcomes the low signal-to-noise ratio that inherently exists in the p2p network, primarily due to user generated content.

We further perform filtering of "weak" song-to-song relations, to remove noise as the one witnessed in the presence of extremely "heavy sharers". During the collection of songs we only include links that appear in at least 16 different users, a values which was again selected as a tradeoff between filtering and memory consumption. Then, we kept for each file, only the top 40% links (ordered by descending similarity value) and not less than 10. Notice that this filter also removes malicious and spam songs from the graph, assuming that these are not downloaded by too many users. After the removal of these "weak" links, roughly 20 million undirected links remain in the graph.

4.1 Degree Distribution

Intuitively, since some popular songs are shared by many users while many songs are shared by only a few users, it is more likely for a song to be co-shared with a popular song, hence increasing the connectivity of the popular song in the similarity graph. This type of connectivity results in a power-law degree distribution, which results in high degrees of the few popular songs and lower degree of many less-popular songs. An important feature of such power-law distributions is the ability to efficiently capture many of the underlying graph properties, by sampling a partial view of the overall network.

On the other hand, when the "tail" of the power-law is long, meaning many songs have very low connectivity, the crawling effort and required resources are significantly higher. The value of the data that exists in the tail greatly depends on the application [4]. Most applications do consider such "rare" files as noise; in that case, their added value is marginal.

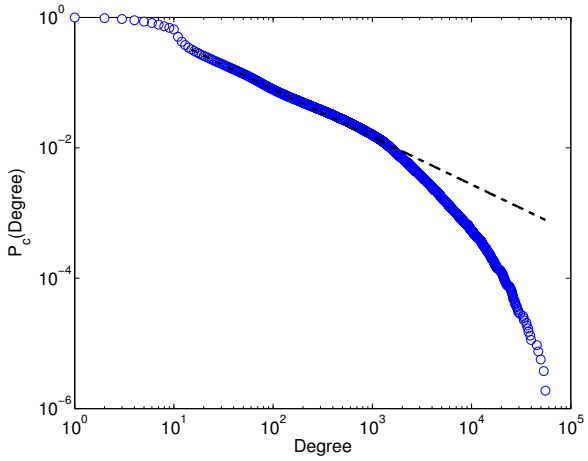


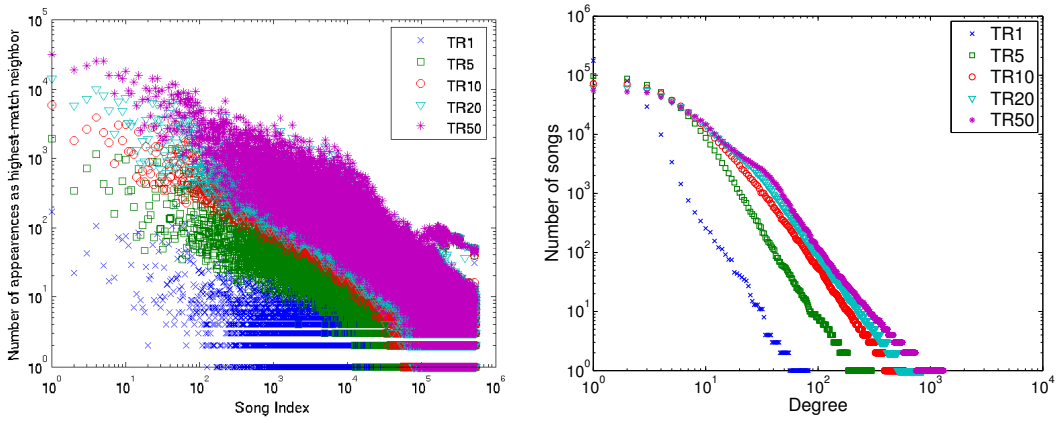
Figure 3. Cumulative degree distribution of the song similarity graph

Several previous studies on p2p networks [6, 7] show that graphs that model various p2p networks exhibit power-law distributions. As can be seen in Figure 4.1 shows the cumulative song degree distribution in the similarity graph, exhibiting a power-law with a strong cut-off. This power-law distribution suggests that there are relatively a few songs with very high connectivity and many songs with low connectivity.

4.2 Partial Sampling

We wish to verify that partial sampling does not significantly alter the distribution of the similarity graph. We first normalize the similarity value between any two songs so it reflects their popularity. Hence, the new similarity is $\widehat{w}_{ij} = w_{ij}/\sqrt{P_i \cdot P_j}$, where w_{ij} is the link weight between songs i and j, and P_i, P_j are their corresponding overall number of occurrences (popularity).

We then create a new graph, denoted by TRN, which contains, for each file, only the top N neighbors, ordered by non increasing normalized similarity. This extends the basic filters since it uses the *normalized* similarity values, thus capturing the relative popularity of adjacent files. This filter is analogous to the effect of a partial sampling in the



(a) Top relations (b) Degree distribution

Figure 4. Effect of sampling on song similarity distribution

p2p network, where many users are simply not reached during the crawling phase. In this case, the crawl "skips" many of the weak relations between songs, while keeping only the strong ones that appear in many users. We therefore, wish to evaluate the way the similarity graph is affected by partial sampling.

The number of times each song appears as the nearest neighbor for different values of N is presented in Figure 4(a). The figure shows that for N=1,5 the distributions are significantly different, whereas for $N \geq 10$ the distributions almost overlap. Similar results can be seen when looking at the degree distribution depicted in Figure 4(b). The figure shows that while for N=1 the distribution is extremely sparse, reaching $N \geq 10$ results in an almost identical distribution with slightly higher node degrees.

The above results indicate that obtaining partial information on the network is sufficient for generating a comprehensive similarity graph, as the utility of having a more complete view of the network quickly decreases. This is attributed to the fact that the songs that are most affected from this partial crawl are the high-degree songs (best noticed in Figure 4(b)). Since many links are gone, songs that did not have too many links to begin with, are hardly affected, while songs that had many links "lose" a lot of them. However, when enough links remain (a sufficient number of users that share these songs are crawled), these songs retain their high degree relative to the other songs.

5. QUERY COLLECTION

Collection of queries is often a much more complicated task than crawling the shared folders. Hence, we seek to quantify the utility of collecting queries from an increasing number of users, similar to the way we did for unique songs. For this end, we collected almost 4.5 million queries from over 3 million users during a week in February 2007. Notice that these queries are not related only to music, however analysis of keywords used for searching the Gnutella network shows that almost 70% of the queries are music related [10].

Figure 2(c) depicts the number of unique queries per number of crawled users, using all the queries, and using only queries that appeared more than once. The figure shows that when all the queries are considered, there is no convergence, meaning that each additional user contributes some new queries. However, when we consider only queries that appeared more than once, there is a clear convergence, and the overall number of unique queries goes down to less than 2 million. We therefore, learn that the diversity in search terms is mostly attributed to very "rare" strings that originate from single users, whereas the majority of the common queries are frequently repeating amongst the different users, hence can be more easily reached.

Queries were shown to be highly correlated with geographic location [7], which is rather intuitive considering the cultural and language differences between countries. In order to quantify the implications of localized query collection, we compared the top-1000 string queries performed by users in different countries, and define the correlation as the total number of matching strings.

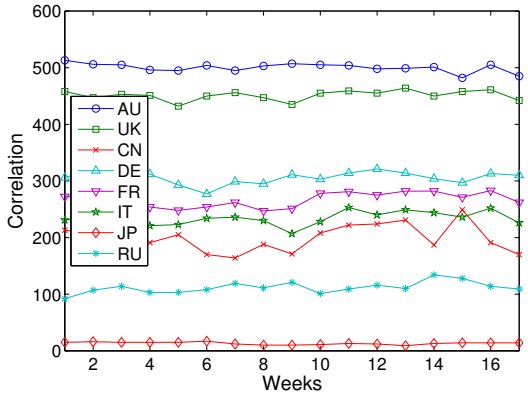


Figure 5. Correlation between top-1000 search queries between the US and different countries over time

Figure 5 depicts the correlation factor between the US and other countries over a period of 17 weeks in early 2007. The figure shows that, as expected, the English speaking countries (Australia and United Kingdom) have much higher correlation with the US than the non-English speaking countries. Japan appears to have the lowest overall

correlation, with less than 20 matching queries. Interestingly, the correlation is quite consistent over the entire period, showing profound differences between the Anglosphere and the non-English speaking countries. Putting aside the musical anthropology aspects of these results, this analysis indicates that when performing targeted research, it is sufficient to focus on a bounded geographical region or country. However, conclusions drawn using queries collected in a specific region should be carefully examined before assuming them on other geographical locations.

6. DISCUSSION AND CONCLUSION

In the presence of an increasing demand for large scale datasets in MIR research, this paper investigates the different considerations in using a p2p based dataset. Several difficulties are considered – the inability to crawl all users and collect information on all songs, the complexities in intercepting all search queries and the inherent noise of user generated content.

Content distribution in a p2p networks typically exhibits a power-law, hence collecting the majority of songs is rather easy. Partial crawling is shown to have much less impact on the availability of main-stream content than on specific "niches". On the other hand, when popularity is considered, partial sampling is more likely to effect the popular songs. Although their relative popularity decreases, song-to-song relations remain intact.

Spatial analysis reveals that p2p networks are highly localized, with profound differences in songs and queries between geographical regions. This can help induce localized research regarding musical trends and preferences, but mandates careful consideration before inferring conclusion drawn from local samples.

File sharing networks were shown to have low signal-to-noise ratio, mandating careful data processing when compared to "traditional" datasets (e.g., website). In order to improve the ability to extract insightful information from the data, we suggest removing songs that appear only once in the dataset, and users that share too many songs, therefore, removing the extremes that are not insightful and may "pollute" the dataset. Furthermore, we present methods for song identification that help merge similar songs, further improving the signal-to-noise ratio. This extensive filtering can be applied to reduce redundant records and false relations, but may result in loss of data, which can be of interest to some MIR tasks, such as popularity predictions.

Overall, p2p networks provide an abundance of information that can be utilized in MIR research. Main-stream data can be easily collected from p2p networks, while having all the benefits over standard website data. However, when seeking to harness the power of the long tail, where p2p networks have a significant advantage, careful analysis is key for sufficient noise reduction while maintaining relevant information.

Acknowledgment. This research was supported in part by a grant from the Israel Science Foundation (ISF) center of excellence program (grant number 1685/07).

7. REFERENCES

[1] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *SCIENCE*, 286:509 – 512, 1999.

[2] Luke Barrington, Reid Oda, and Gert Lanckriet. Smarter than genius? human evaluation of music recommender systems. In *ISMIR*, 2009.

[3] Adam Berenzweig, Beth Logan, Daniel P. W. Ellis, and Brian Whitman. A large-scale evaluation of acoustic and subjective music similarity measures. In *Computer Music Journal*, 2003.

[4] Oscar Celma and Pedro Cano. From hits to niches? or how popular artists can bias music recommendation and discovery. In *2nd Workshop on Large-Scale Recommender Systems*, 2008.

[5] Daniel P. W. Ellis and Brian Whitman. The quest for ground truth in musical artist similarity. In *ISMIR*, 2002.

[6] F. Le Fessant, A. M. Kermarrec, and L. Massoulie. Clustering in peer-to-peer file sharing workloads. In *IPTPS*, 2004.

[7] Adam Shaked Gish, Yuval Shavitt, and Tomer Tankel. Geographical statistics and characteristics of p2p query strings. In *IPTPS*, 2007.

[8] Noam Koenigstein, Gert Lanckriet, Brian McFee, and Yuval Shavitt. Collaborative filtering based on p2p networks. In *ISMIR*, Utrecht, the Netherlands, 2010.

[9] Noam Koenigstein and Yuval Shavitt. Song ranking based on piracy in peer-to-peer networks. In *ISMIR*, 2009.

[10] Noam Koenigstein, Yuval Shavitt, and Tomer Tankel. Spotting out emerging artists using geo-aware analysis of p2p query strings. In *KDD*, 2008.

[11] Noam Koenigstein, Yuval Shavitt, Ela Weinsberg, Udi Weinsberg, and Tomer Tankel. A framework for extracting musical similarities from peer-to-peer networks. In *AdMIRe*, Singapore, July 2010.

[12] Noam Koenigstein, Yuval Shavitt, and Noa Zilberman. Predicting billboard success using data-mining in p2p networks. In *AdMIRe*, 2009.

[13] Matei Ripeanu. Peer-to-peer architecture case study: Gnutella network, 2001.

[14] Markus Schedl, Tim Pohle, Noam Koenigstein, and Peter Knees. What's Hot? Estimating Country-Specific Artist Popularity. In *ISMIR*, Utrecht, the Netherlands, August 2010.

[15] Yuval Shavitt, Ela Weinsberg, and Udi Weinsberg. Estimating peer similarity using distance of shared files. In *IPTPS*, 2010.

[16] Yuval Shavitt and Udi Weinsberg. Song clustering using peer-to-peer co-occurrences. In *AdMIRe*, 2009.

[17] Georgos Siganos, Josep Pujol, and Pablo Rodriguez. Monitoring the Bittorrent monitors: A bird's eye view. In *PAM*, 2009.

[18] The Gnutella Protocol Specification v0.41. `http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf`, 2010.

[19] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, 1974.

A CARTESIAN ENSEMBLE OF FEATURE SUBSPACE CLASSIFIERS FOR MUSIC CATEGORIZATION

T. Lidy, R. Mayer, A. Rauber
Vienna University of Technology, Austria
Department of Software Technology
and Interactive Systems

P. J. Ponce de León, A. Pertusa, J. M. Iñesta
University of Alicante, Spain
Departamento de Lenguajes y
Sistemas Informáticos

ABSTRACT

We present a cartesian ensemble classification system that is based on the principle of late fusion and feature subspaces. These feature subspaces describe different aspects of the same data set. The framework is built on the Weka machine learning toolkit and able to combine arbitrary feature sets and learning schemes. In our scenario, we use it for the ensemble classification of multiple feature sets from the audio and symbolic domains. We present an extensive set of experiments in the context of music genre classification, based on numerous Music IR benchmark datasets, and evaluate a set of combination/voting rules. The results show that the approach is superior to the best choice of a single algorithm on a single feature set. Moreover, it also releases the user from making this choice explicitly.

1. INTRODUCTION AND RELATED WORK

Classification of music into different categories is an important task for retrieval and organization of music libraries. Previous studies reported about a glass ceiling reached using timbral audio features for music classification [1]. Our approach is based on the assumption that a diversity of music descriptors and a diversity of machine learning algorithms are able to make further improvements. We created an ensemble learning system with these two dimensions (feature sets, learning schemes) as input and train models for each combination of those two input dimensions. We call our approach a *cartesian ensemble system*.

Our original motivation has been to combine multiple approaches from the music information retrieval (MIR) domain in order to improve (the reliability of) genre classification results based on the assumption that the various music descriptors are complementary [12]. In our previous work we combined spectrum-based audio features that cover timbral and rhythmic aspects of the sound with symbolic descriptors, based on note and chord sequence statistics. A polyphonic transcription system has been presented as the "missing link" that transcribes audio data into a symbolic notation. In this approach the combination of multiple features from the audio and symbolic domains was performed by a concatenation of feature vectors, jointly used as input to a classification algorithm *(early fusion)*. In a previous comparison of employing MIR algorithms on Western vs. ethnic music [10] we included a *time decomposition* approach, which was already a first ensemble-like approach, applying one learning scheme on multiple input features from different segments of a piece of music and using four different combination (voting) rules to make the final prediction.

The Autonomous Classification Engine ACE [13], by contrast, is a general framework for *model selection*. In machine learning, model selection is the task of selecting one classification model from a pool of models. ACE trains a range of classifiers, with different parameters, and feature selection methods, and then selects the most fitting ones for the current task at hand. ACE is built on top of Weka [20] and thus provides the ensemble techniques implemented in the toolkit, most prominently boosting and bagging, but is not capable of handling *feature subspaces*, or weighted methods as the ones described in Section 3.

The combination of different segments extracted from the same song is studied in [2]. The approach is based on grouping and aggregating non-overlapping blocks of consecutive frames into segments. The segments are then classified individually and the results are aggregated for a song by majority voting. Three different ensemble methods and their applicability to music are investigated in [7]. The first method is based on a *one against all* scheme, i.e. for each class, a classifier is trained on the class and its complement. A second method is based on building a classifier for each pairwise combination of classes. The third method investigates in training different classifiers on different subsets of the feature space. In all methods, the final class label is determined by the probabilities of the individual classifiers.

The approach presented in this paper is a *cartesian ensemble classification* system, which trains a matrix of models built from the combination of a range of individual feature sets and a number of classification algorithms. Our system builds on the Weka machine learning toolkit [20] in an open and flexible way. In contrast to ACE no preselection of classification algorithms has been made – any classification algorithm available can be used with arbitrary parameters in the ensemble. Further, an arbitrary number of feature files can be used. We provide a number of com-

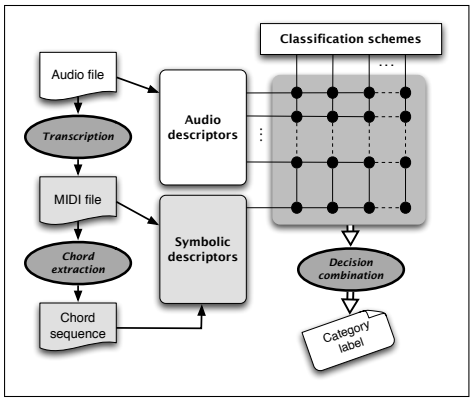


Figure 1. Framework of the cartesian ensemble system

bination and voting rules, which are employed to obtain the final prediction of the classifier ensemble. Our framework is not limited to MIR applications. With regard to our original motivation and our research background, however, we focus on the scenario of music classification into genre categories, in order to show the applicability of the system and the progress in our domain.

The overall scheme of our proposed ensemble classification system is shown in Figure 1. It includes our scenario of a music classification system that processes different descriptors from the audio and symbolic domains (c.f. Section 2). Audio feature extraction algorithms are applied directly to the audio signal data. There is an intermediate step for the symbolic descriptors: A polyphonic transcription system converts the audio information into a symbolic notation (i.e. MIDI files). A chord inference algorithm is applied to provide information about the polyphonic structure of the note stream. Finally, a symbolic feature extractor is applied on the resulting representation. The feature extraction stage provides multiple viewpoints on music objects, called *feature subspaces*. There are several ways of combining them for building a music classification system. *Early fusion* concatenates all feature subspaces to produce so called *superinstances*, including all features at hand. Then a suitable classification scheme is used to learn categories from such data. This approach was used in our previous work [12]. *Late fusion* combines classifier outcomes rather than features. This is the approach employed in our proposed framework.

Section 3 describes the general architecture of our ensemble framework. In Section 4, we evaluate our approach on numerous well-known reference music datasets and show the applicability of the approach. It includes also preliminary research on the use of audio segmentation for generating extended feature subspaces. Finally, Section 5 provides conclusions and an outlook on future work.

2. MUSIC DESCRIPTION

We use two sources of input to our ensemble music classification approach: audio features extracted from audio files and symbolic music descriptors derived from MIDI files that are generated from audio files through a transcription system. We employ features that proved well in our previous works [5, 10–12], also in order to be able to compare progress and results of the new ensemble approach with previous findings. We emphasize, however, that arbitrary feature sets can be used with our classifier ensemble approach presented in Section 3.

2.1 Audio Features

All the following descriptors are extracted from a spectral representation of an audio signal, partitioned into segments of 6 sec. Features are extracted segment-wise, and then aggregated for a piece of music computing the median (RP, RH) or mean (SSD, MVD) from features of multiple segments. We describe the feature extraction algorithms very briefly, please refer to the references for further details.

Rhythm Pattern (RP) The feature extraction process for a Rhythm Pattern is composed of two stages. First, the specific loudness sensation on 24 critical frequency bands is computed through a Short Time FFT, grouping the resulting frequency bands to the Bark scale, and successive transformation into the Decibel, Phon and Sone scales. This results in a psycho-acoustically modified Sonogram representation that reflects human loudness sensation. In the second step, a discrete Fourier transform is applied to this Sonogram, resulting in a spectrum of loudness amplitude modulation per modulation frequency for each critical band. After additional weighting and smoothing steps, a Rhythm Pattern exhibits magnitude of modulation for 60 modulation frequencies on the 24 critical bands [11].

Rhythm Histogram (RH) A Rhythm Histogram (RH) aggregates the modulation amplitude values of the critical bands computed in a Rhythm Pattern and is a descriptor for general rhythmic characteristics in a piece of audio [11].

Statistical Spectrum Descriptor (SSD) The first part of the algorithm, the computation of specific loudness sensation, is equal to the Rhythm Pattern algorithm. Subsequently at set of statistical values [1] are calculated for each individual critical band. SSDs describe fluctuations on the critical bands and capture both timbral and rhythmic information very well [11].

Modulation Frequency Variance Descriptor (MVD) This descriptor measures variations in the critical bands for a specific modulation frequency of the Rhythm Pattern matrix, representing the amplitudes of 60 modulation frequencies on 24 critical bands. The MVD vector is computed by taking statistics [1] for each modulation frequency over the 24 bands [10, 12].

Temporal Features (TRH, TSSD) Feature sets are frequently computed on a per segment basis and do not incorporate time series aspects. We introduced therefore TRH and TSSD features that include a temporal dimension describing variations over time.

For TRH, statistical measures [1] are computed over the individual Rhythm Histograms extracted from the individual 6-second segments in a piece of audio. Thus, change and variation of rhythmic aspects in time are captured.

[1] mean, median, variance, skewness, kurtosis, min and max

TSSD analogously capture timbral variations and changes over time in the spectrum on the critical frequency bands. Hence, a change of rhythmics, instruments, voices, etc. over time is reflected by this feature set [10].

2.2 Transcription from Audio to MIDI

A multiple fundamental frequency (f_0) estimation method is used to convert the audio files to MIDI files. This is a joint estimation approach, which experimentally obtained a high accuracy with a low computational cost. It extends a previous work [16] by adding information about neighboring frames to get a smooth temporal estimation.It does not separate instruments, therefore producing single track MIDI files without any timbral information.

2.3 Symbolic Features

A set of statistical descriptors is extracted directly from transcribed notes. This set is based on the features described in [5], well suited for monophonic classical/jazz classification, and on features described in [17], used for melody track selection in MIDI files. Overall statistics, such as the average number of notes per beat, the occupation rate (non-silence periods with respect to song length) and polyphony rate (proportion of sounding note periods with more than one note active simultaneously) are computed. Further, note pitches, pitch intervals, note durations, silence durations, Inter Onset Intervals (IOI) and non-diatonic notes are analyzed; each property is described by min and max values, range, average, standard deviation, and a normality distribution estimator. Other features include the number of distinct pitch intervals, pitch interval mode, and an estimation of the number of syncopations in the song.

Most of these features are somewhat 'melody-oriented' (e.g., interval-based features). In order to capture relevant information about the polyphonic structure of the transcription, a chord sequence is extracted from it, using the algorithm from Pardo and Birmingham [14], and subsequently analyzed. The different kinds of chord extracted are: major triad, major 7th, dominant 7th, dominant suspended 7th, dominant 7th (sharp 5th), dominant 7th (flat 5th), minor 7th, half diminished and fully diminished chords. The relative frequencies of these chords in a chord sequence are computed as symbolic features. A total of 61 statistical descriptors are therefore provided to the system as a symbolic feature subspace.

3. CARTESIAN ENSEMBLE SYSTEM

Our approach is name a *cartesian ensemble* because the set of models used as base classifiers is the cartesian product of D feature subspaces by C classification schemes. A model is build by training classification scheme c_i on feature subspace d_j. This produces a total of $D \times C$ base models as the ensemble. The aim of this approach is to obtain a sufficiently *diverse* ensemble of models that will guarantee, up to a certain degree, an improvement of the ensemble accuracy over the best single model trained. Moreover, the ensemble abstracts from the selection of a particular classifier and feature set to use for a particular problem. Selecting sufficiently different schemes (different classification paradigms, methods,...) the ensemble provide results that are at least comparable to the best single scheme.

Model diversity is a key design factor for building effective classifier ensembles [9]. This has been empirically shown to improve the accuracy of an ensemble over its base models when they are numerous enough. For selecting the most diverse models within the ensemble the *Pareto-optimal* selection strategy is applied in order to discard models not diverse or not accurate enough.

When a new music instance is presented to the trained ensemble, predictions are made by selected models, which are then combined to produce a single category prediction outcome. A number of decision *combination* (or label fusion) *rules*, can be used for this final prediction.

The cartesian ensemble system is built on the Weka toolkit [20]. The ensemble is a Weka classifier itself, so it can be plugged into any system using this toolkit.

3.1 Pareto-optimal Classifier Selection

This strategy for selecting the best set of models is based on finding the Pareto-optimal set of models by rating them in pairs, according to two measures [9]. The first one is the *inter-rater agreement* diversity measure κ, defined on the coincidence matrix M of the two models. The entry $m_{r,s}$ is the proportion of the dataset, which model h_i labels as L_r and model h_j labels as L_s. The agreement between both classifiers is given by

$$\kappa_{ij} = \frac{\sum_k m_{kk} - \text{ABC}}{1 - \text{ABC}} \tag{1}$$

where ABC is *agreement-by-chance*

$$\text{ABC} = \sum_r (\sum_s m_{r,s})(\sum_s m_{s,r}) \tag{2}$$

The second one is the pair average error, computed by

$$e_{ij} = 1 - \frac{\alpha_i + \alpha_j}{2} \tag{3}$$

where α_i and α_j are the estimated accuracy of the two models, computed as described in Section 3.3. The Pareto-optimal set contains all non-dominated pairs. A pair of classifiers is non-dominated iff there is no other pair that is better than it on both criteria.

3.2 Combination Rules

The combination rules implemented in the system are both weighted and unweighted majority voting rules. A summary of weighted and unweighted combination rules is presented in Table 1, where $P(L_k|\mathbf{x}_i)$ is the posterior probability of instance $\mathbf{x}$ to belong to category L_k, given by model h_i. $\mathbf{x}_i$ is what h_i knows about $\mathbf{x}$, i. e., feature values that correspond to the feature subspace h_i was trained on. Unweighted combination rules are described in [8], and used through their implementation in Weka.

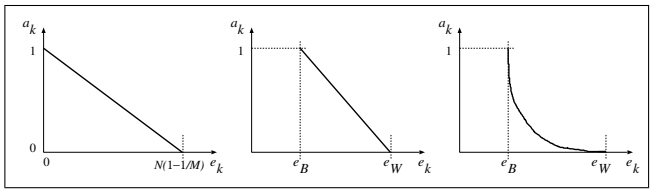


Figure 2. Model weight computation: RSWV (left), BWWV (center), QBWWV (right), giving the model authority a_k as a function of the estimated number of errors e_k made by model h_k on a validation set. N is the number of instances in the set, M is the number of class labels.

All weighted rules multiply model decisions by weights and select the label L_k that gets the maximum score. Model weights are based on the estimated accuracy α_i of the trained models. The *authority* a_i of each model h_i is established as a function of α_i, normalized, and used as its weight ω_i. Weighted methods discussed in [6] have been used in this work. SVW computes weights as described. Weight functions for rules RSWV, BWWV and QBWWV are shown in Figure 2. There, e_B is the lowest estimated number of errors made by any model in the ensemble on a given validation dataset, and e_W is the highest estimated number of errors made by any of those classifiers. WMV is a theoretically optimal weighted vote rule described in [9], where model weights are set proportionally to $\log(\alpha_i/(1-\alpha_i))$.

Table 1. Summary of combination rules.

Rule mnemonic	Description
Unweighted rules	
MAJ	Majority vote rule
AVG	Average of $P(L_k\|\mathbf{x}_i)$
MAX	Maximum of $P(L_k\|\mathbf{x}_i)$
MED	Median of $P(L_k\|\mathbf{x}_i)$
Weighted rules	
SWV	Simple Weighted Vote
RSWV	Rescaled Simple Weighted Vote
BWWV	Best-Worst Weighted Vote
QBWWV	Quadratic Best-Worst Weighted Vote
WMV	Weighted Majority Vote

3.3 Inner/Outer Cross Validation

The classification results presented below are estimated by cross-validating the ensemble. The accuracy of individual ensemble models (α_i), used to compute model weights for combining their outputs, is also estimated through cross-validation. In order to avoid using test data for the ensemble for single model accuracy estimation, an *inner cross-validation*, relying only on ensemble training data, is performed. The number of folds for the ensemble (outer) and the single models (inner) cross-validation are parameters.

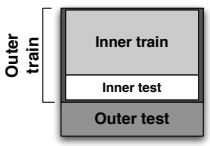


Figure 3. Inner and outer cross-validation scheme.

4. EVALUATION

We performed an extensive evaluation of our ensemble approach on a range of well-known MIR benchmark datasets in order to show both the feasibility and generality of our approach. Classification results are presented as accuracy values with standard deviations.

4.1 Datasets

A dataset overview is given in Table 2. Either full songs or 30 second excerpts were available. 9GDB is originally a MIDI collection, but was synthesized to wav for our experiments and re-transcribed to MIDI to obtain symbolic features. For all other collections audio files were transcribed to MIDI. The GTZAN collection was assembled and used in experiments by G. Tzanetakis [19]. The ISMIRgenre and ISMIRryhthm collections were compiled for the genre and rhythm classification tasks, respectively, of the ISMIR 2004 Audio Description contest [3] and used frequently thereafter by Music IR researchers. ISMIRgenre consists of 6 popular music genres and ISMIRryhthm comprises 8 Latin and Ballroom dances. The Latin Music Database comprises 10 Latin music genres [18]. The African collection is a sub-set of 1024 instances of the audio archive of the Royal Museum of Central-Africain Belgium, digitized in the course of the DEKKMMA project [4]. Various meta-data categories are available for this set, including 27 different *functions*, 11 different *instrument families*, 11 different *countries* and 40 *ethnic groups* [10]. The number of files varies according to number of meta-data available in each category.

Table 2. Datasets used in experiments

dataset	files	genres	file length	ref.
9GDB	856	9	full	[15]
GTZAN	1000	10	30 sec	[19]
ISMIRgenre	1458	6	full	[3]
ISMIRrhythm	698	8	30 sec	[3]
LatinMusic	3225	10	full	[18]
Africa	1024	var.	full	[4]

4.2 Classification Schemes and System Parameters

For our experiments, we set the system to perform 10-fold outer cross-validation and 3-fold inner cross-validation. As for the classification schemes, a selection of classifiers from the *Weka* toolkit has been made, aiming at choosing schemes from different machine learning paradigms. We chose Naïve Bayes, Nearest Neighbor (IB1 [2]) with Euclidean distance, 3-NN with Manhattan distance (IBk), the RIPPER rule learner (JRip), the C4.5 (J48) decision tree learner, the REPTree, a fast decision tree learner, Random Forest, a forest of random trees, and three Support Vector Machines, the first with a linear kernel, the second with a quadratic one and the third with the Puk kernel, a Pearson VII function-based universal kernel with parameter values $C = 4$, $\omega = 3.2$, $\sigma = 13$. Please consult [20] for further reference on these methods.

[2] Weka names for these classifiers in parenthesis.

4.3 Ensemble Classification Results

Table 3. Best results on individual classification of feature sets and classifiers on different datasets

Dataset	Classifier	Featureset	Accuracy
9GDB	SVM-Puk	TSSD	78.15
GTZAN	SVM-lin	SSD	72.60
ISMIRgenre	SVM-quad	TSSD	81.28
ISMIRrhythm	SVM-lin	RP	87.97
LatinMusic	SVM-Puk	TSSD	89.46
Africa/country	SMO-quad	SSD	86.29
Africa/ethnic group	SVM-lin	TSSD	81.10
Africa/function	1-NN	SSD	51.06
Africa/instrument	SVM-Puk	TSSD	69.90

To have a baseline for the cartesian ensemble, we trained all the classification schemes described in Section 4.2 on all the feature sets described in Section 2, i.e. one model for each cell in the cartesian set $D \times C$. Table 3 gives an extract of the accuracies achieved with these single models – due to space limitation, only the best combination of an algorithm and a feature set are given. It can be observed that there is no clear trend, neither for a classifier, nor a feature set. While SVMs clearly dominate the results, the choice of the kernel is not obvious, and results can vary by several percent points. Also the feature sets do not show a clear trend – in approximately half of the cases, TSSDs are the best set to use, while also SSD and RP features sometimes yield clearly better results. These results nourish the hypothesis that ensemble classifiers may provide means to release the user from the difficult choice of the proper feature set and classifier combination.

The accuracy results for the classifier ensembles are shown in Table 4, with the best single classifier as our assumed baseline to improve on. Note that achieving the baseline result would require to know the best combination of feature set and classifier in advance. On each of the datasets, we can observe higher classification accuracies with the ensembles than with the baseline. The improvements are three percent points on average. The highest gains are on the GTZAN dataset, with five percent points, while the improvements on the ISMIRrhythm dataset are of 1.14 percent point. However, the baseline on this dataset is already very high, at approx. 88%.

Out of the nine classification tasks, the QBWWV rule was five times the best, followed by WMV which is three times the best performing rule. AVG and BWWV are both once the highest ranked combination rule. In the tasks where QBWWV is not the rule with the highest accuracy, the relative difference to the top rule is minimal – the largest margin is 0.7 percent points, or 0.86% relative difference.

4.4 Segmentation Ensemble Approach

A logical next step for ensemble classification is the use of individual features from different segments of an audio file as an input to classification. We conducted an experiment segmenting each audio file into 3 equal-sized segments, and extracting individual features from each of those segments. Note that for audio collections with 30 second excerpts we did not do this for TSSD and TRH features, as there would be no temporal variation within a segment, given the feature algorithm's segment-window-length of 6 seconds (c.f. Sec. 2.1). In those cases we used TSSD and TRH features from the full song, as in the previous experiments. Also the symbolic features were used from full songs. Our hypothesis was that with more (detailed) information about the audio content, results would be improved in the ensemble setting. However, results of this segmentation approach were in general inferior compared to using features aggregated over entire songs, as seen from the bottom two lines of Table 4. As the performance decrease was independent of the combination rule applied, we included only the results of the two best combination rules (QBWWV and WMV) for space reasons.

Even though the results of this first experiment did not improve the ensemble approach, we will further pursue this strategy and refine it in multiple ways: First, we will extend the segmentation also to symbolic features. Then we will conduct research on different classifier model combination strategies. Instead of a combination of all classifier/feature set models into one ensemble, a two-tier approach is envisaged, where a decision is made by an ensemble of features from different segments first and then the decisions of multiple different feature sets and classifiers are combined on a second level. Further future work will be the experimentation with different degrees of segmentation of an audio file. Moreover, instead of using equally sized segments, a structural audio segmentation algorithm for segmentation into chorus, verse etc. could be used for semantic segmentation, aiming at an enhanced diversity of the features and the knowledge of the content.

5. CONCLUSIONS

In this paper, we presented a framework for automatic classification of music data. Our system builds ensembles of classifiers in two ways – first, several different algorithms (and parameter variations) are used, and secondly, a set of different features, describing different aspects of the same dataset. We have demonstrated the power of this approach on the classification task for six different datasets and achieved improvements on the classification accuracies in each single task. When comparing the results of the ensemble to the single feature sets, we could observe that there is no clear trend on which classification algorithm, and which feature set to use for a specific dataset. The advantage of the ensemble approach is that the user is released from this task. The ensemble approach delivers superior results through adding a reasonable amount of feature sets and classifiers. Even though we did not discover a combination rule that always outperforms all the others, relying on the QBWWV rule seems feasible.

Future work will include an even wider set of experiments on more datasets, also involving other modalities such as song lyrics. Another area is the above mentioned ensemble of different segments from the same song.

Table 4. Results of the ensemble classification on different datasets (Standard deviations are given in parentheses). The lower section of the table shows the results of the segmentation approach.

Rule	9GDB	GTZAN	ISMIR genre	ISMIR rhythm	Latin Music	Africa country	Africa ethnic group	Africa function	Africa instrument
Single best	78.15 (2.25)	72.60 (3.92)	81.28 (3.13)	87.97 (4.28)	89.46 (1.62)	86.29 (2.30)	81.10 (2.41)	51.06 (6.63)	69.90 (4.69)
MAJ	79.56 (4.78)	72.60 (3.31)	77.78 (2.15)	88.25 (5.08)	89.33 (1.55)	85.31 (4.04)	71.86 (3.41)	37.37 (7.36)	59.63 (5.79)
MAX	60.05 (6.67)	44.00 (6.60)	60.97 (6.71)	54.87 (8.95)	50.64 (2.06)	77.67 (9.16)	73.16 (6.40)	40.38 (7.10)	61.32 (5.88)
MED	74.30 (4.32)	55.90 (3.84)	72.02 (2.74)	77.79 (4.27)	73.64 (2.37)	83.84 (3.77)	70.71 (3.62)	39.49 (5.22)	60.34 (4.67)
AVG	**81.66** (3.96)	68.40 (2.37)	79.70 (3.35)	86.82 (4.29)	86.85 (1.96)	87.66 (2.28)	78.21 (3.50)	53.73 (5.35)	70.60 (3.82)
SWV	81.31 (3.32)	77.10 (3.98)	78.33 (2.48)	88.97 (5.39)	92.00 (1.34)	86.97 (2.98)	75.47 (3.62)	46.83 (4.44)	67.09 (3.99)
RSWV	80.96 (3.26)	77.40 (4.22)	79.22 (2.38)	88.97 (4.94)	92.25 (1.16)	87.17 (2.77)	75.47 (3.62)	48.39 (5.63)	68.35 (4.22)
BWWV	81.54 (3.17)	77.40 (4.22)	82.03 (1.83)	**89.11** (4.62)	92.25 (1.16)	88.34 (2.22)	79.37 (3.95)	52.61 (5.76)	72.71 (3.47)
QBWWV	80.96 (2.94)	**77.50** (4.30)	**84.02** (1.50)	88.97 (3.86)	**92.71** (0.99)	**89.03** (1.63)	82.68 (3.18)	**54.84** (6.29)	72.86 (3.52)
WMV	80.84 (2.90)	76.10 (4.20)	**84.02** (2.02)	87.97 (3.92)	92.59 (1.29)	88.93 (1.76)	**82.97** (3.30)	51.28 (6.93)	**73.00** (4.25)
QBWWV	81.31 (2.78)	76.80 (3.33)	76.95 (3.28)	88.25 (4.39)	91.66 (1.17)	88.44 (2.75)	78.35 (4.08)	50.95 (6.62)	71.03 (3.99)
WMV	80.49 (2.40)	74.50 (4.53)	81.48 (3.01)	87.68 (3.74)	91.56 (1.29)	88.05 (2.12)	80.23 (3.35)	44.83 (4.54)	72.29 (4.45)

6. REFERENCES

[1] J.-J. Aucouturier and F. Pachet. Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[2] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kegl. Aggregate features and Adaboost for music classification. *Machine Learning*, 65:473–484, 2006.

[3] P. Cano, E. Gómez, F. Gouyon, P. Herrera, M. Koppenberger, B. Ong, X. Serra, S. Streich, and N. Wack. ISMIR 2004 audio description contest. Technical Report MTG-TR-2006-02, Pompeu Fabra University, 2006.

[4] O. Cornelis, R. De Caluwe, G. De Tré, A. Hallez, M. Leman, T. Matthé, D. Moelants, and J. Gansemans. Digitisation of the ethnomusicological sound archive of the royal museum for central africa (belgium). *International Association of Sound and Audiovisual Archives Journal*, 26:35–43, 2005.

[5] P.J. Ponce de León and J. M. I nesta. A pattern recognition approach for music style identification using shallow statistical descriptors. *IEEE Trans. on Systems Man and Cybernetics C*, 37(2):248–257, 2007.

[6] F. Moreno-Seco; J. M. Iñesta; P. Ponce de León; L. Micó. Comparison of classifier fusion methods for classification in pattern recognition tasks. *Lecture Notes in Computer Science*, 4109:705–713, 2006.

[7] M. Grimaldi, P. Cunningham, and A. Kokaram. An evaluation of alternative feature selection strategies and ensemble techniques for classifying music. In *Proc. Workshop on Multimedia Discovery and Mining*, 2003.

[8] J. Kittler, M. Hatef, R. P.W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[9] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[10] T. Lidy, C. N. Silla Jr., O. Cornelis, F. Gouyon, A. Rauber, C. A. A. Kaestner, and A. L. Koerich. On the suitability of state-of-the-art music information retrieval methods for analyzing, categorizing, structuring and accessing non-western and ethnic music collections. *Signal Processing*, 90(4):1032 – 1048, 2010.

[11] T. Lidy and A. Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proc. ISMIR*, London, UK, 2005.

[12] T. Lidy, A. Rauber, A. Pertusa, and J.M. Iñesta. Improving genre classification by combination of audio and symbolic descriptors using a transcription system. In *Proc. ISMIR*, Vienna, Austria, 2007.

[13] C. McKay, R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga. Ace: A framework for optimizing music classification. In *Proc. ISMIR*, London, UK, 2005.

[14] B. Pardo and W. P. Birmingham. Algorithms for chordal analysis. *Comput. Music J.*, 26:27–49, 2002.

[15] C. Perez-Sancho, D. Rizo, and J. M. Iñesta. Genre classification using chords and stochastic language models. *Connection Science*, 21(2 & 3):145–159, May 2009.

[16] A. Pertusa and J. M. Iñesta. Multiple fundamental frequency estimation using Gaussian smoothness. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Las Vegas, USA, 2008.

[17] D. Rizo, P.J. Ponce de León, C. Pérez-Sancho, A. Pertusa, and J.M. Iñesta. A pattern recognition approach for melody track selection in midi files. In *Proc. ISMIR*, Victoria, Canada, 2006.

[18] C. N. Silla Jr., A. L. Koerich, and C. A. A. Kaestner. The latin music database. In *Proc. ISMIR*, Philadelphia, USA, 2008.

[19] G. Tzanetakis. *Manipulation, Analysis and Retrieval Systems for Audio Signals*. PhD thesis, Computer Science Department, Princeton University, 2002.

[20] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

IMPROVING THE GENERATION OF GROUND TRUTHS BASED ON PARTIALLY ORDERED LISTS

Julián Urbano, Mónica Marrero, Diego Martín and Juan Lloréns
University Carlos III of Madrid
Department of Computer Science
{jurbano, mmarrero, dmandres, llorens}@inf.uc3m.es

ABSTRACT

Ground truths based on partially ordered lists have been used for some years now to evaluate the effectiveness of Music Information Retrieval systems, especially in tasks related to symbolic melodic similarity. However, there has been practically no meta-evaluation to measure or improve the correctness of these evaluations. In this paper we revise the methodology used to generate these ground truths and disclose some issues that need to be addressed. In particular, we focus on the arrangement and aggregation of the relevant results, and show that it is not possible to ensure lists completely consistent. We develop a measure of consistency based on Average Dynamic Recall and propose several alternatives to arrange the lists, all of which prove to be more consistent than the original method. The results of the MIREX 2005 evaluation are revisited using these alternative ground truths.

1. INTRODUCTION

Information Retrieval (IR) is known for having evolved as a highly experimental discipline. New techniques appear every year, and it is necessary to perform an exhaustive and methodological evaluation to figure out which of these techniques really mean a step forward in the field. These evaluations have been carried out since the late 50's in what has come to be known as the Cranfield paradigm. Given a fixed document collection, IR systems provide their results for certain information needs. Then, they are evaluated against the so called ground truths, which contain information about the documents that should ideally be retrieved by a system. Usually, these ground truths take the form of a matrix, containing the relevance, assessed by humans, for each document to an information need (traditional values are "irrelevant", "relevant" and "highly relevant").

These evaluations have been carried out mostly in Text Information Retrieval, with the TREC conferences as its flagship [1]. Music Information Retrieval (MIR), on the other hand, is a relatively young discipline, and this kind of evaluations has been somewhat scarce until the arrival of MIREX in 2005 as a first attempt to perform TREC-like evaluations in the musical domain [2]. Music IR differs from Text IR in many aspects [3], making the construction and maintenance of such test collections very difficult. In particular, it is unclear what relevance level to assign to a document for a given information need.

In the case of melodic similarity, some studies indicate that relevance is continuous [4]. Single melodic changes such as moving a note up or down in pitch, or extending or shortening its duration, are not perceived to change the overall melody. Nonetheless, the relationship with the original melody is gradually weaker as more changes are applied to it. There does not seem to be common criteria to split the degree of relevance into different levels, so assessments with a fixed scale seem inappropriate.

Ground truths based on partially ordered lists attempted to handle this problem with relevance assessment by the beginning of 2005 [5]. Instead of having documents with a fixed relevance level, these ground truths are lists with ordered groups of documents. The earlier a group appears in the list, the more relevant its documents are, and documents within the same group are assumed to be equally relevant. That way, the ideal retrieval should return these documents in order of relevance, although permutations within the same group are allowed. Because traditional effectiveness measures such as precision or recall need relevance assessments with a fixed scale, a new measure, called Average Dynamic Recall (ADR) [6], was developed also in 2005 to evaluate retrieval systems with ground truths based on partially ordered lists.

The first edition of MIREX had a task for symbolic melodic similarity [7], where 11 ground truths based on partially ordered lists were used along with ADR to evaluate state-of-the-art retrieval systems. Similar methods were used in the 2006 and 2007 editions, as well as in private evaluations external to MIREX, such as [8] [9] [10] and [11]. However, we are not aware of any meta-evaluation work addressing the correctness or improvement of these ground truths. Indeed, a thorough examination shows that the lists have some inconsistencies as to the arrangement and aggregation of documents in groups.

The paper is organized as follows. In Section 2 we review the methodology followed to create these ground truths. Section 3 unveils some inconsistencies and shows that it is not possible to ensure fully-consistent lists. In Section 4 we propose several alternatives to set up the groups, and present a measure to quantify consistency. Section 5 shows the results of the alternatives proposed and revise the MIREX 2005 evaluation using them. The paper ends with conclusions and lines for future work.

2. CURRENT METHODOLOGY

The original method to create ground truths based on partially ordered lists, as described in [5], was used with the RISM A/II collection [12], which at the time contained about half a million musical incipits. The methodology followed may be divided in four steps: filtering, ranking, arranging and aggregating.

First, several features were calculated for each document (musical incipits in this case), such as pitch range, interval histogram or motive repetitions. Filtering by these features, the initial collection was gradually narrowed down to under 300 incipits per query. Then, clearly irrelevant incipits were manually excluded, and several melodic similarity algorithms were used to add supposedly relevant incipits. Second, and once the lists had about 50 candidate incipits each, 35 experts ranked them in terms of melodic similarity to the corresponding query: the more similar a candidate was to the query, the higher it had to be ranked in the list. Incipits that seemed very different from the query could be left unranked. Third, incipits were arranged according to the median of their rank sample. If two incipits had the same median rank, the means were used to resolve the tie. Therefore, the incipits that on average were ranked higher by the experts appeared with higher ranks in the ordered list. Fourth, incipits whose rank samples were similar were aggregated within a group, so as to indicate that they were similarly relevant to the query. Thus, a retrieval system could return them with their rank swapped and still be considered correct. The Mann-Whitney U test (also known as Wilcoxon Rank Sum test, see Appendix) [13], was used to tell whether two incipits had similar ranks or not.

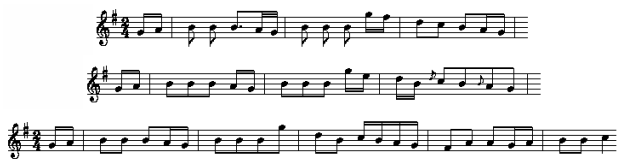


Figure 1. First three results for query 000.111.706-1.1.1. Top to bottom: same incipit as the query, incipit 000.113.506-1.1.1 and incipit 000.116.073-1.1.1.

The ground truths generated have some odd results, as already noted in [5] and [9]. For example, in the list for the query incipit 270.000.749-1.19.1, the first result is the same as the query; the second one (incipit 270.000.746-1.41.1) is written with a different clef, but otherwise identical to the query; and the third result (incipit 270.000.748-1.19.1) is the same as the first half of the query. Although the experts were told to disregard these kinds of changes in the melody, these three results ended up in different groups, indicating that their relevances to the query were significantly different. Also, incipits with virtually the same changes in the melody were sometimes placed in different groups, as it occurs with incipits 000.113.506-1.1.1 and 000.116.073-1.1.1 with respect to the query 000.111.706-1.1.1 (see Figure 1).

These rare results seem to be caused by the second step of the methodology, when experts ranked the results. Though important, we will not focus on them in this paper. There are other problems with this kind of ground truths that have not been addressed yet and lead to inconsistent result lists and incorrect evaluation. These inconsistencies arise at steps three and four, and they are the ones we address here.

3. INCONSISTENCIES DUE TO ARRANGEMENT AND AGGREGATION

We thoroughly examined the 11 ground truth lists used in the evaluation of the symbolic melodic similarity task in MIREX 2005 (*Eval05* for short), and found that there are pairs of incipits contained in the same group of relevance although there is a significant difference between the ranks the experts gave them (i.e. an intra-group inconsistency). For example, incipits 453.001.547-1.1.3 and 451.509.336-1.1.1, for query 190.011.224-1.1.1, are in the same group (see Figure 2), but their difference is significant. That means that if a retrieval system returned them in reverse order it would be considered correct, despite the experts clearly ranked them differently. On the other hand, incipits for which no significant difference could be found form part of different groups (i.e. an inter-group inconsistency). Incipits 700.000.686-1.1.1 and 450.034.972-1.1.1 for the previous query are an example (see Figure 2). Similarly, if a retrieval system returned them in reverse order, it would not be considered correct, despite no difference was found in the experts rankings.

These inconsistencies appear throughout the lists, and they are caused by the initial arrangement and the aggregation function used in the third and fourth steps.

3.1 Arrangement

In the third step of the methodology, incipits are arranged according to the median and mean ranks they were given by the experts. Because the Mann-Whitney U test is used later on to find statistically significant differences between the incipits' ranks, using central-tendency measures such as the median and the mean might not be appropriate to arrange the results, because they do not account for the dispersion in the samples.

Although rare, this phenomenon may happen: we examined the 11 ground truths of the *Eval05* collection and found it. For example, incipits 850.014.902-1.1.1 and 451.002.538-1.1.1 are ranked 20th and 22nd, respectively, for query 400.065.784-1.1.1. Their sample median ranks are 12 and 12.5, so the first one is ranked higher. However, a 1-tailed Mann-Whitney U test shows that it is highly probable for the true medians to be ordered the other way around, so the second incipit should be ranked higher than the first one.

3.2 Aggregation

In the fourth step of the methodology, incipits are aggregated in groups according to their relevance to the query. The rationale originally used by the aggregation function is as follows: traverse from top to bottom the list of incipits already arranged by median and mean, and begin a

new relevance group if the pivot incipit is significantly different from all incipits in the current group [5]. Therefore, it will probably allow significantly different incipits in the beginning and the end of the same group, just because they are not different from a third one. A new group will begin only when an incipit is very significantly different from all the previous ones, so the group is likely to grow a lot. We looked for this kind of inconsistency in the 11 lists of the *Eval05* collection, and found that out of the total 509 ordered pairs of incipits in the same relevance group, 178 (35%) are significantly different. All these intra-group inconsistent pairs translate into incorrect evaluation when allowing an incipit to appear earlier in the results list just for being misplaced in the same group as another one ranked a little higher.

There can also be cases where the aggregation function places an incipit in a new group, but the next one is not significantly different from some others in the group just finished. This next incipit should be in the previous group, but that is not possible since it has been already closed because of the previous one. For example, incipit 453.001.547-1.1.3 started group 4 for the query 190.011.224-1.1.1, because it was different from all incipits in group 3. However, the incipit 700.000.686-1.1.1, in group 3, is not significantly different from incipit 450.034.972-1.1.1, which is in group 4 (see Figure 2). All these inter-group inconsistent pairs also translate to incorrect evaluation when not permitting an incipit to appear earlier in the results for being misplaced in a later group started by another incipit that was sufficiently different.

3.3 Fully Consistent Lists

Inconsistencies come from two different sources: the initial arrangement by median and mean may arrange pairs of documents in the wrong order, and the aggregation function may combine significantly different incipits or set apart similar ones. The aggregation function can mitigate these problems, but there is a more profound problem: hypothesis testing is not transitive.

Let *X*, *Y* and *Z* be the rank samples given to three different incipits. The Mann-Whitney U test may suggest that the median of *Y* is less than the median of *Z*, and that the median of *X* is less than the median of *Y*. Still, it may suggest that the medians of *X* and *Z* are not different (i.e. $X < Y$, $Y < Z$ but $X = Z$). Although this might seem completely paradoxical, it actually happens, for example in the ground truth list for query 400.065.784-1.1.1. Let *X*, *Y* and *Z* be the rank samples of incipits 702.002.512-1.1.1, 804.002.648-1.1.2 and 450.021.643-1.1.1, ranked 6th, 8th and 16th respectively. A 1-tailed Mann-Whitney U test shows that the median of *X* is significantly smaller than the one of *Y* (p-value = 0.238) and that the median rank of *Y* is significantly smaller than the one of *Z* (p-value = 0.239), but the median rank of *X* does not seem significantly smaller than the one of *Z* (p-value = 0.272). Although p-values this large would not usually be accepted to reject a null hypothesis, they are valid in our case, since the significance level originally used was 0.25 [5].

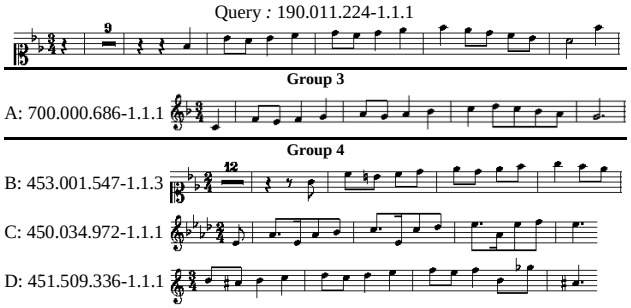


Figure 2. Excerpt of the ground truth for query 190.011.224-1.1.1. According to the experts: $B \neq D$ (intra-group inconsistency), $A = C$ (inter-group inconsistency), $A \neq B$ and $B = C$ (2-tailed non-transitivity as $A = C$).

Therefore, it is not possible to ensure fully consistent lists with this method. In the example above, incipit *X* should be in a group ranked higher than *Y*, which should be in a group ranked higher than *Z*. However, *X* and *Z* should be in the same group, which is clearly impossible. Similar cases can be found with 2-tailed tests, such as the example in Figure 2 ($X \neq Y$, $X = Z$ but $Y = Z$).

4. ALTERNATIVE AGGREGATION FUNCTIONS

The number of intra- and inter-group inconsistencies depend on the aggregation function used. A function too permissive, like the original one, leads to larger groups with more likelihood of intra-group inconsistencies, but a function too restrictive leads to smaller groups with more likelihood of inter-group inconsistencies. The aggregation function should minimize these problems and generate lists as consistent as possible.

We consider three different rationales to be followed:

- *All*: a new group is started if the pivot incipit is significantly different from every incipit in the current group. This should lead to larger groups.
- *Any*: a new group is started if the pivot incipit is significantly different from any incipit in the current group. This should lead to smaller groups.
- *Prev*: a new group is started if the pivot incipit is significantly different from the previous one.

At this point, we have to consider whether 2-tailed or 1-tailed tests should be used. Originally, 2-tailed tests were used, looking for 2-way differences in median ranks. But, because the incipits are already sorted by median and mean, we believe the tests should be 1-tailed, looking for 1-way differences in ranks. After arranging incipits in step three, we may assume that an incipit appearing after another one has a rank either lower or equal, but not higher. In these situations, 1-tailed tests are more powerful than their 2-tailed counterparts, so it is more probable for them to find a difference between two samples if there really is one (see Appendix).

Therefore, we obtain six different functions, combining each of the three rationales with each of the two statistical tests. We call these functions *All-2*, *All-1*, *Any-2*, *Any-1*, *Prev-2* and *Prev-1*. Note that *All-2* is the function originally used by Typke et.al. [5], while the other five are proposed in this paper.

4.1 Measure of list consistency

To evaluate the 5 alternative functions presented above, and compare them with the original one, we developed a measure of consistency based on Average Dynamic Recall [6]. ADR is the main effectiveness measure used to evaluate retrieval systems against ground truths based on partially ordered lists, so we followed its same idea to measure their consistency, and hence the correctness of the evaluation itself. ADR measures the average recall over the first *n* documents, where *n* is the number of documents in the ground truth. At each point, the set of relevant documents allowed comprises all previous documents in the list plus all those in the same group as the pivot, because they are supposed to be equally relevant.

With a ground truth list like *⟨(A, B), (C), (D, E, F)⟩*, and a retrieval list such as *⟨B, C, A, G, H, D⟩*. ADR would be calculated as in Table 1. In the first two positions, either document *A* or *B* is considered correct because they are in the same relevance group, so both of them can be expected. At position 3, both *A* and *B* are expected because they appear before in the list, and only *C* is added when expanding the second group. However, when position 4 is reached, every document in the third group may be expected. Recall is calculated at each position, and the overall ADR is the mean average of these recalls, 0.753 in this case.

Position	Retrieved	Expected	Correct	Recall
1	**B**	A,**B**	1	1
2	**B**,C	A,**B**	1	0.5
3	**B**,**C**,**A**	**A**,**B**,**C**	3	1
4	**B**,**C**,**A**,G	**A**,**B**,**C**,D,E,F	3	0.75
5	**B**,**C**,**A**,G,H	**A**,**B**,**C**,D,E,F	3	0.6
6	**B**,**C**,**A**,G,H,**D**	**A**,**B**,**C**,**D**,E,F	4	0.667

Table 1. Example of ADR calculation.

To measure the consistency, the list is traversed from top to bottom, expanding the group corresponding to the pivot incipit. At each position, it is calculated the percentage of incipits expanded that are actually correct according to the experts rankings. At the end, the mean of those percentages is calculated. Therefore, a final value of 1 means that every expansion is correct and hence the list is fully-consistent. A value of 0 means that every expansion is incorrect. The pivot incipit is never considered for the calculation, because it will always be correctly expanded.

There are two types of incorrect expansion: false positives (i.e. an incipit is included in the set of expected, but it is significantly different from the pivot) and false negatives (i.e. an incipit is not included in the set of expected, but it is not significantly different from the pivot). Note that false positives correspond to intra-group inconsistencies, and false negatives correspond to inter-group inconsistencies. In the example above, imagine *A* and *C* are not significantly different, but *D* and *F* are. In that case, the expansion at position 1 is missing incipit *C* (a false negative due to an inter-group inconsistency between groups 1 and 2). Also, the expansion at position 4 would incorrectly include incipit *F* (a false positive due to an intra-group inconsistency in group 3). Note that at position 2, *C* would not be correctly expanded, as it is still significantly different from *B*. Note also that position 6 is not considered, as there is actually no expansion at the end of the list. In this case, the overall list consistency would be 0.86 (see Table 2).

Position	Correct expansion	Actual expansion	% of correct expansions
1	B,**C**	B	0.5
2	A	A	1
3	A,B	A,B	1
4	A,B,C,E	A,B,C,E,**F**	0.8
5	A,B,C,D,F	A,B,C,D,F	1

Table 2. Example of list consistency calculation.

As before, we can measure the inconsistencies using both 2-tailed and 1-tailed tests. In the former, two incipits are expected to be in the same expanded set if a 2-tailed Mann-Whitney U test is not rejected. In the latter, they are expected to be in the same expanded set if none of the two 1-tailed tests is rejected (i.e. the true median rank of one incipit seems to be neither less nor greater than the other's). Note that both the 2-tailed and the 1-tailed measures account for inconsistencies originated by the aggregation function but only the 1-tailed version accounts for inconsistencies due to the simple arrangement by median and mean. We call these two measures ADR-2 and ADR-1 consistency.

Because of the non-transitivity problem, lists are not expected to have an overall consistency of 1. However, it could be maximized by changing the aggregation function, thus improving the correctness of the evaluation.

5. RESULTS

The five alternative aggregation functions proposed back in Section 4 were used to re-generate the 11 lists in the *Eval05* collection and compare them with the original function *All-2*. We used the ADR-1 consistency measure to calculate the overall consistency of each list. The results are in Figure 3 and in Table 3.

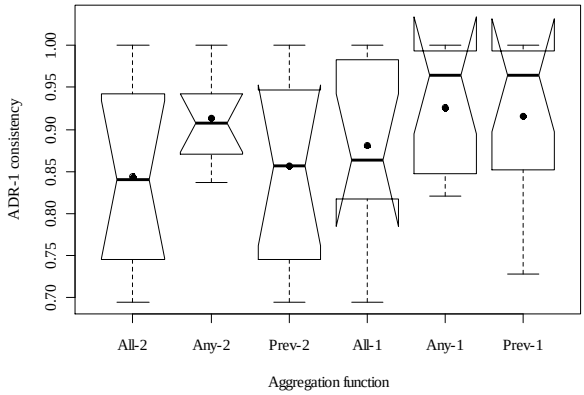


Figure 3. ADR-1 consistency for the six aggregation functions. Solid circles indicate the mean value. Notches mark the 95% confidence interval around the median.

As can be seen, the original function, *All-2*, is outperformed by all of the five alternatives proposed. *All-2* leads to an average consistency of 0.844, which is the

smallest of the six. However, *Prev-2* and *All-1* are not significantly better according to a 1-tailed t-test at the 0.10 significance level. Moreover, the *All* functions lead to results with more variability, while the *Any* functions are more stable in terms of consistency. These results indicate that if the lists were generated with the *Any-2*, *Any-1* or *Prev-1* aggregation functions, they would be more consistent, and so would be the evaluation with them.

Interestingly, the relative order for each of the three 2-tailed and 1-tailed functions is maintained. That is, the *All* functions perform the worst, followed by *Prev* and *Any*, which perform the best both in terms of average consistency and variability.

Our guess back in Section 3.2 was that the larger the sizes of the relevance groups are, the more inconsistent the lists are too. To examine this, we calculated the mean number of incipits per group for each of the 11 resultant lists. Figure 4 and Table 3 show the results.

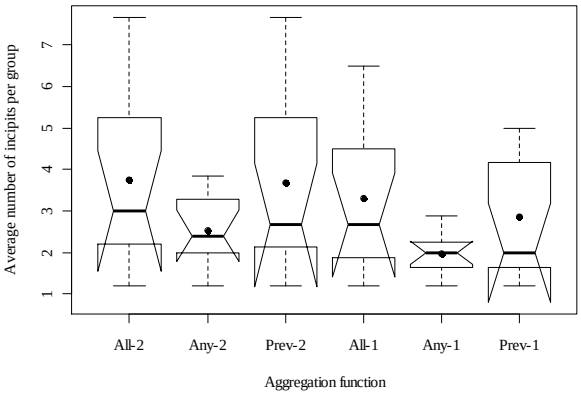


Figure 4. Mean number of incipits per group for each aggregation function. Solid circles indicate the mean value. Notches mark the 95% confidence interval around the median.

As expected, the *All* functions lead to larger groups, because an incipit goes to a new group only if it is different from all the previous ones. On the other hand, the *Any* functions generate smaller groups, because only one difference needs to be found to place the incipit in a new group. Similarly, the *Any-2* function leads to significantly smaller groups than *All-2* at the 0.10 significance level, and *Any-1* is significantly smaller at the 0.05 level.

Aggregation function	ADR-1 consistency	Incipits per group	Pearson's r
All-2	0.844	3.752	-0.892***
Any-2	0.913**	2.539*	-0.862***
Prev-2	0.857	3.683	-0.937***
All-1	0.881	3.297	-0.954***
Any-1	0.926**	1.981**	-0.749***
Prev-1	0.916*	2.858	-0.939***

Table 3. Summary of results. * for significant difference at the 0.10 level, ** at the 0.05 level and *** at the 0.01 level.

Following these results, there seems to be a direct relationship between the size of the groups and the overall consistency of the lists. We checked this by calculating the Pearson's r correlation coefficient between the two variables and, as expected, there is a strong negative correlation, indicating that the size of the groups affects the consistency of the lists (see Table 3). This is why the *All* functions perform worse and the *Any* functions perform better: the *All* versions generate larger groups. Doing so, they allow for many incorrect expansions in the form of false positives due to intra-group inconsistencies.

5.1 MIREX 2005 Results Revisited

In the 2005 edition of the MIREX evaluations, there was a task for symbolic melodic similarity that used 11 ground truths based on partially ordered lists (what so far we have called the *Eval05* collection). In particular, 7 different systems were evaluated.

We calculated the ADR score of each system with the lists generated by the five alternative aggregation functions (see Table 4). Every alternative evaluation produces worse results than the original, except for *Prev-2*, which leads to the same scores. Indeed, every system performed worse for every alternative set of ground truths, with reductions in ADR score of up to 12%.

System	All-2	Any-2	Prev-2	All-1	Any-1	Prev-1
GAM	**0.66**	0.59	**0.66**	0.624	0.583	0.605
O	0.65	**0.607**	0.65	**0.643**	0.593	**0.639**
US	0.642	0.604	0.642	0.639	**0.594**	0.628
TWV	0.571	0.558	0.571	0.566	0.556	0.564
L(P3)	0.558	0.52	0.558	0.54	0.515	0.534
L(DP)	0.543	0.503	0.543	0.511	0.494	0.506
FM	0.518	0.498	0.518	0.507	0.483	0.507
τ	-	0.81	1	0.81	0.714	0.714

Table 4. ADR results of the systems that participated in MIREX 2005 with the lists resulting from the alternative aggregation functions. GAM = Grachten, Arcos and Mántaras; O = Orio; US = Uitdenbogerd and Suyoto; TWV = Typke, Wiering and Veltkamp; L(P3) = Lemström (P3), L(DP) = Lemström (DP); FM = Frieler and Müllensiefen. Best scores appear in bold face.

More importantly, the relative order of the systems, in terms of their mean ADR score, is also modified. For example, with the original lists GAM was the best system, followed by O and US. With the *Any-2* lists, O is ranked first, before US and GAM. However, with the *Any-1* lists the order is reversed: US, O and GAM. We calculated Kendall's τ correlation coefficient to measure the differences in the ranking of systems (see Table 4). A value of 1 means that two rankings are equal, and a value of -1 means that they are reversed. Except for *Prev-2*, which produces the same results as *All-2*, the correlation coefficients tell us that the resulting rankings are different.

6. CONCLUSIONS AND FUTURE WORK

With their appearance in 2005, ground truths based on partially ordered lists represented a big leap towards the scientific evaluation of Music Information Retrieval systems, particularly for melodic similarity tasks. They have been widely accepted and used by the community, both in MIREX and other private evaluations.

We have revised the methodology used to generate these lists, unveiling some unaddressed problems. We have shown that the lists generated have inconsistencies, and propose several alternatives to minimize them. Using ADR-1 consistency, we have shown that our alternatives

lead to better results. We have also seen how would have changed the evaluation of the symbolic melodic similarity task in MIREX 2005, showing that the absolute effectiveness figures would have changed notably, and the ranking of systems would have been different too.

More meta-evaluation work in this line has to be carried out to improve the evaluation in MIR. In this paper we have focused on the last two steps of the methodology, analyzing the evaluation collection used in MIREX 2005. Other test collections should be analyzed, and the first two steps of the methodology should be studied as well because they are known to produce odd results too. One of the reasons may be the subjectivity on the judgments that the loose definition of the task can lead to, as already noted in [2] and [3]. More precise definitions of the information need sought by these tasks would surely lead to more coherent judgments by the experts.

One point that has not been discussed in the literature either is the significance level used by the aggregation function, which was 0.25 for the original lists. Our measure of consistency also works with a significance level to decide whether incipits are correctly arranged or not, and though they should probably be the same, we should study what value is more appropriate in both cases.

Finally, the lists generated with the alternative aggregation functions show diverse characteristics, mainly in terms of group sizes and differences among incipits in the same group. Other effectiveness measures, besides ADR, could be proposed to exploit these characteristics, while accounting for the unavoidable inconsistencies.

ACKNOWLEDGEMENTS

We thank Carlos Gómez, Rainier Typke and the IMIRSEL group, especially Mert Bay and Stephen Downie, for providing us with the MIREX evaluation data. We also thank William Frakes and Gabriella Belli for their insight regarding statistics and hypothesis testing.

APPENDIX. THE MANN-WHITNEY U TEST

The Mann-Whitney U test [13], or Wilcoxon Rank-Sum test, is a non-parametric statistical test to assess whether the true medians of two independent samples, say *X* and *Y*, are significantly different or not. Consider *X* as the sample of ranks of 600.258.342-1.1.2 for query 600.053.481-1.1.1, and *Y* the ranks given to incipit 850.020.721-1.1.1. The test statistic *U* is calculated as:

$$U=|X|\cdot|Y|+\frac{|Y|(|Y|+1)}{2}-\sum_{i=1}^{|Y|}\text{rank}(y_i)$$

where *rank(y_i)* is the rank that the *i-th* number of *Y* would have in the set $X \cup Y$. In our example, $U = 131$. The critical value is calculated depending on the alternative hypothesis H_1. For a 2-tailed test, H_1 would be that the true medians are different, but if a 1-tailed is chosen instead H_1 would be that the true median of *X* is less than the true median of *Y* (or the other way around). In the 2-tailed case, the rejection region is spread around both sides of the critical value, while in the 1-tailed case it is only in one side. Therefore, the 2-tailed case accounts for 2-way differences ($X > Y$ or $X < Y$), while the 1-tailed case looks only for 1-way differences ($X < Y$ in our case).

With a significance level of 0.25, the critical value for the 2-tailed test is $U_2 = 121$, while for the 1-tailed test it is $U_1 = 136$. Thus, the 1-tailed null hypothesis would be rejected because $U < U_1$, but the 2-tailed would not because $U > U_2$. In this case, the 2-tailed test fails to detect that the medians are, in fact, different. Because the 1-tailed test looks for a signed difference, it is more powerful and rejects the null hypothesis ($H_0 = X \leq Y$ in our example).

REFERENCES

[1] E.M. Voorhees and D.K. Harman, *TREC: Experiment and Evaluation in Information Retrieval*, MIT Press, 2005.

[2] J.S. Downie, A.F. Ehmann, M. Bay, and M.C. Jones, "The Music Information Retrieval Evaluation eXchange: Some Observations and Insights," *Advances in Music Information Retrieval*, W.R. Zbigniew and A.A. Wieczorkowska, Springer, 2010, pp. 93-115.

[3] J.S. Downie, "The Scientific Evaluation of Music Information Retrieval Systems: Foundations and Future," *Computer Music Journal*, vol. 28, no. 2, 2004, pp. 12-23.

[4] E. Selfridge-Field, "Conceptual and Representational Issues in Melodic Comparison," *Computing in Musicology*, vol. 11, 1998, pp. 3-64.

[5] R. Typke, M. den Hoed, J. de Nooijer, F. Wiering, and R.C. Veltkamp, "A Ground Truth For Half A Million Musical Incipits," *Journal of Digital Information Management*, vol. 3, no. 1, 2005, pp. 34-39.

[6] R. Typke, R.C. Veltkamp, and F. Wiering, "A Measure for Evaluating Retrieval Techniques based on Partially Ordered Ground Truth Lists," *IEEE International Conference on Multimedia and Expo*, 2006, pp. 1793-1796.

[7] J.S. Downie, K. West, A.F. Ehmann, and E. Vincent, "The 2005 Music Information Retrieval Evaluation Exchange (MIREX 2005): Preliminary Overview," *International Conference on Music Information Retrieval*, 2005, pp. 320-323.

[8] M. Grachten, J. Arcos, and R. López, "A Case Based Approach to Expressivity-Aware Tempo Transformation," *Machine Learning*, vol. 65, no. 2, 2006, pp. 411-437.

[9] P. Hanna, P. Ferraro, and M. Robine, "On Optimizing the Editing Algorithms for Evaluating Similarity Between Monophonic Musical Sequences," *Journal of New Music Research*, vol. 36, no. 4, 2007, pp. 267-279.

[10] J. Urbano, J. Lloréns, J. Morato, and S. Sánchez-Cuadrado, "Using the Shape of Music to Compute the Similarity between Symbolic Musical Pieces," *International Conference on Computer Music Modeling and Retrieval*, 2010.

[11] A. Pinto and P. Tagliolato, "A Generalized Graph-Spectral Approach to Melodic Modeling and Retrieval," *International ACM Conference on Multimedia Information Retrieval*, 2008, pp. 89-96.

[12] K. Saur Verlag, "Répertoire International des Sources Musicales (RISM). Serie A/II, Manuscrits Musicaux après 1600," 2002.

[13] H.B. Mann and D.R. Whitney, "On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other," *Annals of Mathematical Statistics*, vol. 18, no. 1, 1947, pp. 50-60.

IBT: A REAL-TIME TEMPO AND BEAT TRACKING SYSTEM

João Lobato Oliveira[1,2] **Fabien Gouyon**[1] **Luis Gustavo Martins**[3] **Luis Paulo Reis**[2]
[1]Institute for Systems and Computer Engineering of Porto (INESC Porto), Porto, Portugal
[2]Artificial Intelligence and Computer Science Laboratory (LIACC), FEUP, Porto, Portugal
[3]Research Center for Science and Technology in Art (CITAR), UCP, Porto, Portugal
{jmso,fgouyon}@inescporto.pt lgustavomartins@gmail.com lpreis@fe.up.pt

ABSTRACT

This paper describes a tempo induction and beat tracking system based on the efficient strategy (initially introduced in the BeatRoot system [Dixon S., "Automatic extraction of tempo and beat from expressive performances." Journal of New Music Research, 30(1):39-58, 2001]) of competing agents processing musical input sequentially and considering parallel hypotheses regarding tempo and beats. In this paper, we propose to extend this strategy to the *causal* processing of *continuous* input data. The main reasons for this are threefold: providing more robustness to potentially noisy input data, permitting the parallel consideration of a number of low-level frame-based features as input, and opening the way to real-time uses of the system (as e.g. for a mobile robotic platform).

The system is implemented in C++, permitting faster than real-time processing of audio data. It is integrated in the MARSYAS framework, and is therefore available under GPL for users and/or researchers.

Detailed evaluation of the causal and non-causal versions of the system on common benchmark datasets show performances reaching those of state-of-the-art beat trackers. We propose a series of lines for future work based on careful analysis of the results.

1. INTRODUCTION

Computational tracking of musical beats from audio signal is a very important feature to automated music analysis. In the context of Music Information Retrieval applications, such as e.g. automatic genre classification, music similarity computation, *autotagging*, or query-by-example, recent literature indicates that audio descriptors of higher level of abstraction are needed [1]. It is a relatively safe bet to say that reliable beat trackers will be helpful in this endeavour.

Recent evaluations of existing beat tracking systems (see e.g. MIREX [1]) show that, although progresses have undeniably been achieved in the last years, there is still room for improvement. Many open directions to beat tracking research are also detailed in a recent and very thorough evaluation in [6]. Particularly, there is to date, to our knowledge, no real-time and open-source audio beat tracker available.

Very many papers in the literature address the problem of tempo induction and beat tracking of audio signals. Providing a review of existing systems and algorithms is out of the scope of this paper. Interested readers are referred to [8] for a review of rhythm description systems.

It is however important to mention the main functional aspects commonly found in beat tracking algorithms. A generic description includes the following computing blocks: (1) Audio feature extraction, (2) Induction (or "Pretracking" herein), and (3) Beat Tracking *per se*.

It is also interesting to notice that recent systems, e.g. [12], [7], [3], [14], tend to implement beat tracking as a repeated induction process, in which tempo and beats are computed on consecutive windows of signal (usually a few seconds, where it is usually considered that the tempo is constant), with overlap, and in which estimating tempo evolution and beat positions is done by connecting observations between windows. We argue that a problem with this approach is a potential computational overload, the intrinsic difficulty to adapt these tracking strategies to causal and real-time scenarios, as well as lack of continuity between windows. Instead, we propose to follow the tracking strategy initially proposed in the system BeatRoot [4], where competing agents process musical input data sequentially and consider parallel hypotheses regarding tempo and beats.

We propose to differ from BeatRoot's strategy by implementing a *causal* decision process over competing agents (instead of taking decisions after the whole data has been analysed). Further, we extend the algorithm to the processing of *continuous* input data. Our aim is to provide more robustness to potentially noisy input data, and opening the way to (faster than) real-time uses of the system (as e.g. for a mobile robotic platform). The system is implemented in C++ and the source code is available as GPL. Although this paper does not provide experiments with respect to the usefulness of diverse low-level features as input to tracking beats [9] [2], it should be noted that a particularity of the proposed architecture is precisely to be open to such experiments. Another difference with BeatRoot lies in an attempt to not bias results towards faster metrical levels.

[1] http://www.music-ir.org/mirex/2009/

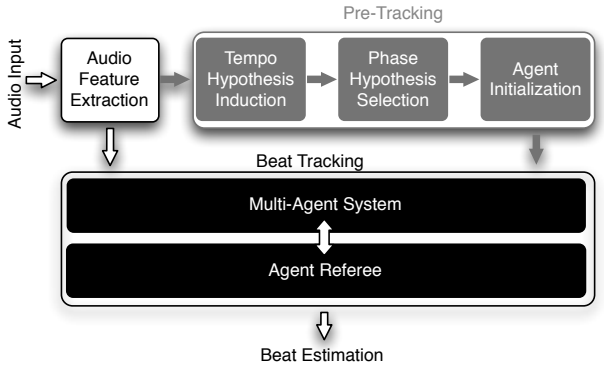


Figure 1. IBT block diagram.

In section 2, we describe one-by-one the functional blocks of IBT, [2] a tempo induction and beat tracking algorithm in the line of BeatRoot [4]. The algorithm follows a modular workflow composed by: (1) an audio feature extraction module, "parsing" the audio data into a continuous [3] feature sequence assumed to convey the predominant information relevant to rhythmic analysis; followed by (2) a pre-tracking module, which outputs initial hypotheses regarding possible beat periods and phases; followed by (3) a beat tracking module, which propagates hypotheses, proceeds to their online creation, killing and ranking, and outputs beats on-the-fly (see Figure 1). Section 3 reports on a thorough evaluation of the system. Section 4 proposes some practical hints for those intending to use the system, and/or make changes to its code. Section 5 discusses the system performances and proposes lines for future work.

2. SYSTEM DESCRIPTION

2.1 Audio Feature Extraction

According to recent comparative studies evaluating alternative onset detection functions [5] and the accuracy of several low-level features applied to beat tracking purposes [9], we selected the spectral flux as the audio feature over which all further processing will be done.

Our implementation follows that proposed in [5]. Particular parameters are: Hamming window, window size of 1024 samples (23.2ms at a sampling rate of $F_s = 44100Hz$), and 50% overlap.

In order to smooth the onset detection function and reduce false detections, a low-pass Butterworth filter is applied on the extracted spectral flux values. As a way to avoid phase distortion the spectral flux values in the induction window are filtered in both the forward and reverse directions, resulting in a precisely zero-phase distortion.

2.2 Pre-tracking

The system is initialized on an induction window, set to a length of 5s. The following sections (until Section 2.3) report on computations done on the induction window only. During the processing of that bit of data, the system does not output beats. At the end of that pre-processing step, hypotheses regarding periods, phases and scores (P_i, ϕ_i, S_i) of a number of beat agents are passed along to the beat tracking module.

The length of the induction window is a high-level parameter that the user can define.

2.2.1 Period Hypotheses Induction

The first step in the pre-tracking stage is to compute a continuous periodicity function, based on the spectral flux autocorrelation, along time-lags τ :

$$A(\tau) = \sum_{n=0}^{m} SF(n)SF(n+\tau), \qquad (1)$$

where $SF(n)$ is the (smoothed) spectral flux for frame n, and m is the induction window size (in frames).

The periodicity function is then parsed by an adaptive peak-picking algorithm to retrieve N global maxima, whose time-lags constitute the initial set of period hypotheses P_i :

$$\begin{cases} P_i = \arg\max_i(A(\tau)), i = 1, ..., N \\ A(\tau) > \delta * \frac{rms(A(\tau))}{M} \end{cases}, \qquad (2)$$

where δ is a fixed threshold parameter, empirically set to 0.75, and *M* is the chosen tempo range, defined to $[50, 250]$ BPM (i.e. periods of 240 ms to 1.2 s), at a 6ms granularity.

2.2.2 Phase Hypotheses Selection

For each one of the period hypothesis P_i, a number of phase hypotheses ϕ_i^j (where j is the index of the alternative hypotheses for the i-th period hypothesis) are considered among detected onsets (detection is done on the induction window only, and computed as proposed in [5]).

For each period hypothesis, we generate an isochronous sequence of beats (a "beat train template") of constant period for each possible phase ϕ_i, with the same length as the induction window.

Using a simplified tracking procedure (see Section 2.3), considering a constant tempo and phase, we then select the beat train template that best matches the detected onsets and retrieve its corresponding phase [10].

At this point, we have computed a set of period and phase hypotheses, (P_i, ϕ_i). The next step is to compute a score for each hypothesis and to rank them.

2.2.3 Agents Setup

A raw score S_i^{raw} is given to each (P_i, ϕ_i) hypothesis, corresponding to the sum of time deviations between elements of the chosen beat train template and local maximum in the *spectral flux* (see eq. (10)).

Scores are then updated via the consideration of possible metrical relationships between each pair of period hypotheses n_{ij}. As proposed in [4], we define a score S_i^{rel} that favors candidates whose periods are in integer relationships:

$$S_i^{rel} = 10 * S_i^{raw} + \sum_{\substack{j=0 \\ j \neq i}}^{N} r(n_{ij}) * S_j^{raw} \qquad (3)$$

[2] Standing for INESC Porto Beat Tracker.

[3] i.e. sampled, with typical sampling rate in the tenth of msec

$$r(n) = \begin{cases} 6-n, 1 \leq n \leq 4 \\ 1, 5 \leq n \leq 8 \\ 0, otherwise \end{cases} \quad (4)$$

Finally, we define the final scores, S_i, as follows:

$$S_i = S_i^{rel} * max(S^{raw}) \quad (5)$$

The estimated hypotheses (P_i, ϕ_i, S_i) can now be used to initialize a set of N beat agents, which will start their beat tracking activity, as described in the following sections.

2.3 Beat Tracking

Following the pre-tracking stage described in the previous sections, the process of on-line beat tracking will consist on the supervision of the incoming spectral flux values, constantly handling any tempo/timing variations, while keeping a good balance between reactiveness (speed of response to system changes) and inertia (stability of the system). As illustrated in Figure 1, this process is handled by a multi-agent system mediated by a central referee.

2.3.1 Agents Operation

Initialized using the pre-tracking (P_i, ϕ_i, S_i) hypotheses, an initial set of N beat agents will start to propagate, in a causal manner, predictions based on incoming data, by representing alternative hypotheses regarding beat positions and tempo. Each prediction is evaluated with respect to its deviation (i.e. error) to the local maximum in the observed data, within a two-level tolerance window. This is a stage where the system differs significantly from BeatRoot: although the tolerance windows are akin to [4], processing continuous data is necessarily different here than onsets; and the generation of new agents also differs as we include more than one new hypothesis, accounting more specifically for tempo and/or timing deviations.

This two-level tolerance window consists in an *inner* tolerance region, $T_{in} \in [T_{in}^l, T_{in}^r]$, $T_{in}^l = T_{in}^r = 46.4$ ms, for handling short period and phase deviations, and an asymetric *outer* tolerance region, $T_{out} \in [T_{out}^l, T_{in}^l[\cup]T_{in}^r, T_{out}^r]$, with a left margin $T_{out}^l = 0.2 * P_i$ and a right margin $T_{out}^r = 0.4 * P_i$, see Figure 2. This allows to contemplate eventual sudden changes in tempo expression (the asymmetry reflects the higher tendency for tempo reductions than increases).

Consequently, two alternative scenarios arise. A first scenario corresponds to a local maximum found inside the *inner* tolerance window. In such case, the agent's period and phase are compensated by a fraction of that error:

$$\begin{cases} P_i = P_i + 0.25 * error \\ \phi^i = \phi^i + P_i + 0.25 * error \end{cases}, \exists\, m \in T_{in}. \quad (6)$$

A second scenario considers bigger deviations, with local maxima in the *outer* tolerance window. On this condition the agent under analysis keeps its period and phase but, in order to cope for potential sudden variations of tempo and/or timing, it generates three children $\{C_1, C_2, C_3\}$ to follow three alternative hypotheses, considering alternative possible deviations of its own current hypothesis: timing (phase), tempo (period), or timing and tempo:

$$C_1 : \begin{cases} P_C^1 = P_i \\ \phi_C^1 = \phi^i + P_i + error \end{cases}, \exists\, m \in T_{out}, \quad (7)$$

$$C_2 : \begin{cases} P_C^2 = P_i + error \\ \phi_C^2 = \phi^i + P_i + error \end{cases}, \exists\, m \in T_{out}, \quad (8)$$

$$C_3 : \begin{cases} P_C^3 = P_i + 0.5 * error \\ \phi_C^3 = \phi^i + P_i + 0.5 * error \end{cases}, \exists\, m \in T_{out}. \quad (9)$$

To keep the competitiveness, these new agents inherit a portion (80% in the current implementation) of their father current score.

Ultimately, alternative possible situations may terminate an agent operation, at any analysis frame: *replacement*, *redundancy*, *obsolescence*, or *loss*. An agent is killed if it is currently the worst agent in a pool of agents that has reached a maximum number (limited to 30 agents), and if its score is lower than a newly created agent. In order to increase the algorithm efficiency, an agent is killed if it is duplicating the work of another agent whose score is bigger (their periods do not differ by more than 11.6ms and their phases no more than 23.2ms). An agent is also terminated if the difference between its score and the best agent's is higher than 80% of the best score. Finally, an agent may be also killed if it seems to be "lost," suggested by a high number (i.e. 8) of consecutive beats predictions outside its inner tolerance window.

2.3.2 Agent Referee

In order to determine the best agent at each data frame, a central *Agent Referee* keeps a running evaluation of all agents at all times. This is conducted by scoring the beat predictions of each agent with respect to its goodness-of-fit to incoming data.

The following evaluation function, Δs, is applied around each beat prediction b_p, which evaluates distance between beat prediction and the local maximum *m* inside either the *inner* or the *outer* window (see Figure 2):

$$\begin{cases} \Delta s = \left(1 - \frac{|error|}{T_{out}^r}\right).\left(\frac{P_i}{P_m}\right).SF(m), \exists\, m \in T_{in} \\ \Delta s = -\left(\frac{|error|}{T_{out}^r}\right).\left(\frac{P_i}{P_m}\right).SF(m), \exists\, m \in T_{out}, \end{cases} \quad (10)$$

where P_m is the maximum admitted period, in frames. The $\frac{P_i}{P_m}$ fraction is used to normalize the score function by the period as a way to deflate faster tempi hypotheses, which would otherwise tend to get higher scores due to a higher number of beat predictions. Note also the fact an agent score can undergo positive as well as negative updates.

2.3.3 Non-Causal Version

Whereas causal processing retrieves the beats of the *current* best agent, at any time-frame, in the non-causal version only the *last* best agent is considered. For such, every agents keep an history of their beat predictions, attached to the one inherited form their relatives, and transmit it to

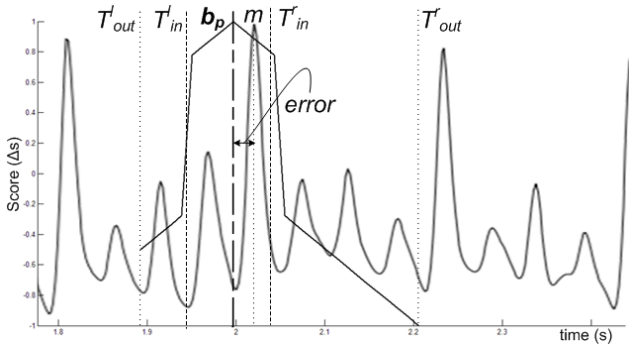


Figure 2. Score function around a beat prediction, b_p, with $P_i = 120BPM$. Example of local maxima m found in the considered inner tolerance window T_{in}.

future generations. Distinctively to the former, this process distinguishes the family of agents whose cumulative score prevails for the whole piece.

In the non-causal version, after pre-tracking and intial agents setup, the analysis "jumps back in time," and beat tracking is performed from the beginning of the signal.

3. EVALUATION

In this section we report on performance evaluation of the proposed algorithm with respect to 2 tasks: tempo estimation and beat tracking. In order to ease comparison to current state-of-the-art systems, we use current benchmark datasets and evaluation measures.

3.1 Datasets

IBT was evaluated using two distinct datasets. For measuring global tempo estimation performance, we use the ISMIR 2004 Tempo Induction Contest data [11]. It consists on 3199 tempo-annotated instances, divided in three categories: *Ballroom*, *Loops*, and *Songs*.

For the beat tracking evaluation, we use 1360 beat-labeled musical pieces (previous use of this dataset is reported in [9] and [6]).

3.2 Evaluation Measures

The system estimation of tempo is evaluated via the two metrics proposed in [11]: *a1* (estimations are considered correct only if they are equal to the annotated tempo) and *a2* (correct estimations also include related metrical levels at 2, 3, $\frac{1}{2}$, and $\frac{1}{3}$ of the ground-truth). Both metrics allow a 4% tolerance window.

Beat-tracking performances are measured via the *P-score* [13], with a 20% tolerance around median Inter-Beat-Interval (IBI) annotations (as in MIREX 2006 Audio Beat Tracking Contest and [6]).

In order to evaluate IBT's robustness to noise distortions, we also applied a number of signal degradations: downsampling, GSM encoding/decoding, fitering, volume adjustment, addition of reverb and white noise (see [11] for more details).

3.3 Global Tempo Estimation

Table 1 presents accuracies obtained for global tempo estimation, with regular and distorted data. The global tempo was measured as the median IBI of final beat predictions, i.e. after beat tracking the whole piece. We also report accuracies obtained before tracking, at the output of the pre-tracking stage, where we select the period hypothesis with highest rank.

Condition	Ballroom		Loops		Songs		Overall	
	a1	a2	a1	a2	a1	a2	a1	a2
IBT(c)	48	83	41	73	30	73	40	76
IBT(c) dist.	44	76	40	72	31	66	38	71
IBT(nc)	49	90	37	76	36	82	41	83
IBT(nc) dist.	48	82	37	74	34	74	40	77
pre-tracking	42	75	40	74	29	71	37	73

Table 1. Global tempo estimation accuracies, in %; "(c)" and "(nc)" stand for the causal and non-causal versions of the system, repectively; "dist." indicates distorted data (see 3.2).

Condition	Metrical relation to annotation (theor. max)					
	1:1(100)	2:1(50)	1:2(50)	3:1(33)	1:3(33)	all
IBT(c)	74(*558*)	47(*282*)	46(*201*)	40(*14*)	27(*9*)	57
IBT(nc)	81(*613*)	46(*266*)	45(*238*)	40(*15*)	24(*10*)	61
1beat(c)	80(*544*)	49(*354*)	40(*109*)	39(*13*)	26(*7*)	59
1beat(nc)	88(*618*)	47(*321*)	42(*153*)	38(*18*)	26(*6*)	64
2beats(c)	79(*1087*)	53(*20*)	28(*2*)	— (*0*)	— (*0*)	72
2beats(nc)	82(*1080*)	47(*44*)	37(*13*)	— (*0*)	— (*0*)	74
dist.(c)	73(*547*)	46(*247*)	45(*164*)	39(*13*)	20(*9*)	55
dist.(nc)	81(*599*)	45(*255*)	44(*196*)	37(*17*)	22(*9*)	59
dind.(c)	72(*511*)	48(*369*)	45(*128*)	38(*22*)	23(*5*)	55
dind.(nc)	81(*582*)	47(*347*)	44(*168*)	37(*15*)	29(*4*)	60
BeatRoot	81(*613*)	48(*535*)	44(*7*)	34(*41*)	N/A	60
BR 2beats	80(*1245*)	45(*10*)	32(*4*)	33(*3*)	N/A	77

Table 2. Beat tracking *P-scores* by metrical relation with the ground-truth, under different conditions; "(c)" and "(nc)" stand for the causal and non-causal versions of the system, repectively; "dist." indicates distorted data (see 3.2); "dind." stands for "dumb" induction (see text). The first line of the table indicates the metrical relation found between the algorithm output and the ground-truth and the corresponding theoretical maximum *P-Score*. The format of other lines is as follows: {P-Score (*number of excerpts tracked at each metrical level)*}.

3.4 Beat Tracking

Table 2 provides results of beat tracking experiments under diverse conditions. The first two lines show results of the causal and non-causal system under regular conditions.

The next four lines refer to beat tracking with biased initialization, either giving the annotated first beat, or giving the first two beats. This help evaluating the performance of beat tracking *per se*, independently of the performance of tempo induction and phase estimation. It is also convenient in order to compare with BeatRoot performances [6].

Results were grouped with respect to metrical relations between the system outputs and the ground-truth annotations. This provides useful information regarding the system tracking performance regardless of it having choosen the "correct" metrical level. (Note that given the evaluation metrics used (the *P-score*), the theoretical performance maximum is different for different metrical relations between output and annotations.)

All results were generated with the same default parameters concerning reactiveness vs. stability of the system. In terms of computational time, IBT took around 11% of the dataset length to process it non-causally, and about 10% to do it causally. (The tests were run on a Core2Duo 2.8 GHz Windows Vista (32-bit) machine.)

4. PRACTICAL USE

IBT was developed in C++ and is freely available, under GPL licensing, in MARSYAS (`http://marsyas.info/`). (At the date of writing, revision 3827.) The algorithm includes three main modes of operation, executable with the following commands:

$ ./ibt input.mp3 (causal mode (default));
$ ./ibt -mic (live mode (microphone captured data));
$ ./ibt -nc input.mp3 (non-causal mode);
$ ./ibt -a input.mp3 (play audio w/ clicks on beats).

4.1 Important parameters

The presented evaluation was run with default parameters, empirically chosen to conciliate reactivity and stability of the system. Values of diverse parameters can be increased to obtain a more reactive system: the margins of tolerance windows (*LFT_OUTTER_MARGIN, RGT_OUTTER_MARGIN, INNER_MARGIN*); portion of an agent current score transmitted to its children, (*CHILDREN_SCORE_FACTOR*); and children correction factors (*CHILDX_FACTOR*).

5. DISCUSSION AND FUTURE WORK

5.1 On tempo estimation

Table 1 shows that the non-causal version of IBT performs comparatively to the best algorithms tested in the ISMIR 2004 contest [11]. The non-causal version shows slightly worse results, but still remains in the best third of the algorithms. Overall it is fair to say that the system finds either the correct tempo or make (somehow acceptable) errors of metrical level in 80% of the cases.

Comparable tempo estimation results are observed on the second dataset (Table 2). Careful evaluation of the first and second lines shows that the tempi of a total of 1064 excerpts and 1142 excerpts (i.e. 78% and 83%) are correct or correspond to metrical level errors, in the causal and non-causal versions, respectively.

The last line of Table 1 also shows that tempo estimation is more reliable after tracking the whole excerpt than at the output of the induction stage. For instance, non-causal beat tracking outperforms pre-tracking by around 10 points. Also, tempo induction seems to work worse on the *Loops* dataset. This is due to the fact that many of these excerpts are very short (many are in fact shorter than our induction window length —5s).

It is also interesting to notice that, after tracking the whole piece, tempo estimation results obtained with "dumb" induction are sensibly similar —albeit an apparent decrease of the number of correct metrical levels found— to those obtained with a more informed induction process (82% vs. 83%, respectively, in the non-causal case, considering all acceptable metrical levels).

Tempo estimation is quite robust to distortions of the audio signal, although the accuracy loss is still about 5 points. This is a clear advantage with respect to systems that process discrete lists of onsets instead of continuous features, such as BeatRoot (see [11] for a detailed comparison — note that BeatRoot's results are relative to a previous version of the software and that recent changes in the onset detection function are likely to have improved them).

These findings seem to indicate that, although tempo induction in IBT reaches good levels, comparable to the state-of-the-art, further increase in accuracy will certainly be obtained if future work is dedicated to improving the induction process. Previous findings indicate that worthwhile lines of work include research on the amount of data needed for induction, reliability of the estimation, improved robustness to noise, and the possibility to trigger induction on different parts of the data, depending on a monitoring of the tracking process self-evaluation.

We can see on Table 1 that accuracy with a2 is much better than with a1 (36-37 points overall difference). Table 2 also shows that, as BeatRoot, IBT tracks a significant number of excerpts at the "wrong" metrical level. However, at the difference with BeatRoot, these excerpts are more uniformally distributed among lower and higher levels. This is the direct effect of the period normalization factor found in the scoring function, eq. (8). These findings indicate that more work should be done on the issue of finding the "correct" metrical level, which may be contemplated by the scoring function itself. In that respect, results from [12] on a1 indicate that a promising direction lies in beat tracking at several metrical levels simultaneously.

5.2 On beat tracking

Table 2 permits us to focus on the tracking performance of the system, independently of its performance in finding the correct tempo. The first two lines of the first column shows us that when IBT finds the correct tempo, it tracks beats correctly in 74% of the cases, the non-causal version does it slightly better: 81%. This is the same performance as BeatRoot. Tracking performances when IBT follows beats on a different metrical level than the annotations are

also similar to BeatRoot.

Careful listening and visualization to tracking errors produced by the causal vs. the non-causal system shows, as should be expected, that the former is more prone to interchanges between phase and metrical levels, compromising continuity.

When one correct beat is given as input to IBT, performance increases up to 88% in the non-causal case. This is a good result, although it also suggests that a number of errors are made at the stage of phase selection (2.2.2) during pre-tracking. Here again, as argued in the previous section, this suggests than future work should be dedicated to improving the induction phase. When two correct beats are given, global performance increases, although performance at the correct metrical level suffers a slight decrease, due to the fact that this figure is computed on significantly more data (i.e. IBT finds more correct metrical levels).

With regards to robustness to signal distortions, it seems that as with tempo estimation, the use of continuous features instead of discrete onsets results in higher robustness. However, IBT performance still decreases about 2 points on average with respect to clean data, calling for future work related to more robust feature extraction.

When the tracking module is given "dumb" period hypotheses, tracking results are only marginally lower than when the period is inferred with a more informed method. This shows that the system has the desirable property to not depend too heavily on correct estimation of the tempo and to recover from errors. Future work should be dedicated to evaluating the speed at which the system recovers from errors, and experiments should be dedicated to fine-tuning system parameters towards the best trade-off between reactiveness to changes and error recovery, on one side, and stability on the other side.

6. SUMMARY

This paper presents IBT, an agent-based tempo and beat tracking system that causally (and non-causally) processes incoming values of a continuous audio feature (e.g. onset detection function). Benchmarks on causal and non-causal versions reveal competitive results, under alternative conditions. In particular, the proposed algorithm produces equivalent beat tracking results to those of BeatRoot, and accurately estimates tempo at the level of state-of-the-art algorithms. A special care has been put on designing a system usable for real-time processing, with good noise robustness, and with no bias towards particular metrical levels. IBT is open-source and freely available with MARSYAS. Promising paths for future work include: tempo induction improvements; informed alternates of the induction and tracking phases; beat tracking at several metrical levels simultaneously; use of several input features.

7. ACKNOWLEDGEMENTS

This work was funded by a PhD scholarship endorsed by FCT, with ref. SFRH/BD/43704/2008, and was supported by QREN's project Palco 3.0, led by Palco Principal.

8. REFERENCES

[1] J.-J. Aucouturier. Sounds like teen spirit: Computational insights into the grounding of everyday musical terms. In J. Minett and W. Wang, editors, *Language, Evolution and the Brain, Frontiers in Linguistics Series*. Taipei: Academia Sinica Press, 2009.

[2] M. Davies and M. Plumbley. Comparing mid-level representations for audio based beat tracking. In *Proceedings of the DMRN Summer Conference*, 2005.

[3] M. Davies and M. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):1009–1020, 2007.

[4] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 2001.

[5] S. Dixon. Onset detection revisited. In *in Proceedings of the 9th International Conference on Digital Audio Effects*, pages 133–13, Montreal, Canada, 2006.

[6] S. Dixon. Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–50, 2007.

[7] D. P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[8] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 29(1):34–54, 2005.

[9] F. Gouyon, S. Dixon, and G. Widmer. Evaluating low-level features for beat classification and tracking. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2007.

[10] F. Gouyon, P. Herrera, and P. Cano. Pulse-dependent analyses of percussive music. In *AES 22nd International Conference on Virtual, Synthetic and Entertainment Audio*, 2002.

[11] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech and Language Processing*, 14(5):1832–1844, 2006.

[12] A. Klapuri, A. Eronen, and J. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):342–355, 2006.

[13] M. F. McKinney, D. Moelants, M. Davies, and A. Klapuri. Evaluation of audio beat tracking and music tempo. *Journal New Music Research*, 36(1):1–6, 2007.

[14] G. Peeters. Template-based estimation of time-varying tempo. *EURASIP, Journal on Applied Signal Processing (Special Issue on Music Information Retrieval Based on Signal Processing)*, 36(1):51–60, 2007.

IMPROVING AUTO-TAGGING BY MODELING SEMANTIC CO-OCCURRENCES

Riccardo Miotto
University of Padova
miottori@dei.unipd.it

Luke Barrington
UC San Diego
lukeinusa@gmail.com

Gert Lanckriet
UC San Diego
gert@ece.ucsd.edu

ABSTRACT

Automatic taggers describe music in terms of a multinomial distribution over relevant semantic concepts. This paper presents a framework for improving automatic tagging of music content by modeling contextual relationships between these semantic concepts. The framework extends existing auto-tagging methods by adding a Dirichlet mixture to model the contextual co-occurrences between semantic multinomials. Experimental results show that adding context improves automatic annotation and retrieval of music and demonstrate that the Dirichlet mixture is an appropriate model for capturing co-occurrences between semantics.

1. INTRODUCTION

A central goal of music information retrieval (MIR) is to create systems that can efficiently and effectively retrieve songs from massive music collections. A potential solution to this challenge is to describe songs with a collection of manually annotated meaningful words (tags) and to perform retrieval based on these text descriptions. Commercial recommendation systems such Last.fm [1] and Pandora [2] extensively use this semantic similarity approach to create recommendation lists. Tags are useful because they contextualize a song by describing human emotions, personal style, geographic origins, spiritual foundations, historical period, or particular uses of the song.

1.1 Auto-Tagging

The continuous growth of music collections is making manual human annotation of every song infeasible. In response, several scalable approaches have been proposed for labeling music with semantics including social tagging [6], web mining [5] or tag propagation from similar songs [12], each with advantages and disadvantages [14]. In particular, MIR researchers have proposed content-based "auto-taggers" – methods that analyze acoustic waveforms and automatically assign meaningful words to songs. Much of this work has been inspired by related methods for automatic image annotation [13].

One of the the first proposed approaches used Gaussian Mixture Models (GMM) computed over the audio features of the training examples to represent a vocabulary of words [15]. An alternative model, the Codeword Bernoulli Average (CBA) [4] attempted to predict the probability that a tag applies to a song based on a vector quantized representation of the audio signal. Regardless of the model used, the output of an auto-tagger is a vector of tag probabilities which may be interpreted as a *semantic multinomial* (SMN), a distribution that characterizes relevance of each tag to a song. Semantic multinomials capture patterns in a song's waveform that represent high-level properties such as genres, emotions or instrumentation.

1.2 Tag co-occurrence

Auto-tagging models aim to capture statistically regular patterns in the audio content and associate these patterns with descriptive semantics. In general, these models treat each tag *independently*, ignoring the *context* that derives from associations between tags. Indeed, while some semantic associations in music are inspired by direct auditory cues (e.g., hearing a "violin"), others are inferred through contextual relationships (e.g., inferring "cello" and "bassoon", when listening to "orchestral classic music"). This gives rise to statistically significant co-occurrence patterns of semantic concepts in the training data (e.g., many "rock" songs also tagged as "loud"), and thus in the SMNs. We suggest that actively capturing correlations in SMNs can improve the semantic description of a song.

Two situations cause tags to co-occur in semantic multinomial distributions. The first is when a tag accidentally co-occurs with another concept. Accidental co-occurrences could be due to many reasons, ranging from poor posterior probability estimates arising from auto-tagger errors, to the unavoidable ambiguous interpretation of music, such as confusing "trumpet" and "trombone". The second type of tag co-occurrence results from feature vectors that truly describe multiple musical concepts. For example, a "cello" piece is very likely to have feature vectors that also fit tags such as "classical music" or "violin". While only co-occurrences of the second type are indicative of *true* contextual relationships, SMN distributions derived from acoustic content exhibit both types of co-occurrences.

[1] http://www.last.fm
[2] http://www.pandora.com

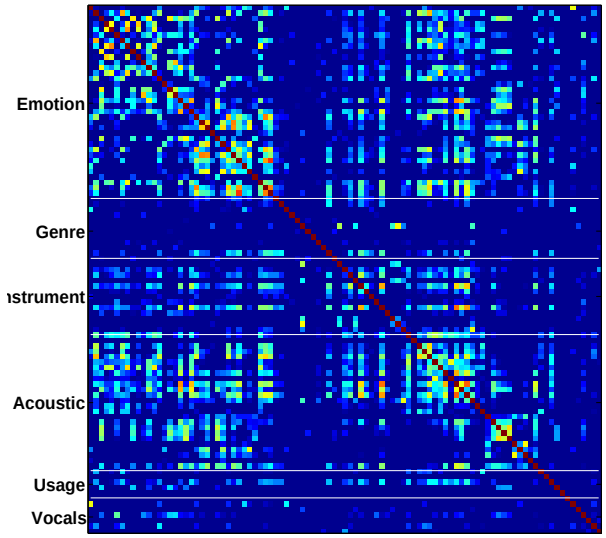


Figure 1. Co-occurrence patterns for *CAL500*; redder points imply high correlation between tags.

To understand the extent of tag co-occurrences, we examine the Computer Audition Lab 500 (CAL500) dataset, used later in our experiments (see Section 4 for more details). Figure 1 depicts the pairwise correlation matrix between CAL500 tags. Correlation values have been computed through an application of Jaccard's Coefficients [8],

$$n_{ij} = \frac{P(w_i \cap w_j)}{P(w_i) + P(w_j) - P(w_i \cap w_j)}, \qquad (1)$$

which provide a measure of the strength of the association between the general words w_i and w_j, normalized by the total number of times the two words appear. The n_{ij} coefficients range between 0 and 1, with $n_{ij} > 0$ if the tags are not mutually exclusive (i.e., if they occur together in some songs). In Figure 1, redder parts represent tag pairs that are highly correlated (i.e., where n_{ij} is large). As can be seen, correlation is present in many tags and it is particularly prevalent in the "Emotion" and "Acoustic" categories whereas tags categorized as "Genre" display few correlation patterns.

The co-occurrence patterns illustrated in Figure 1 are not explicitly captured by auto-taggers that model acoustics independently for each tag. Although SMNs capture patterns at the song level that are predictive of semantic tags, each dimension of the semantic space (i.e., each tag) is assumed to be independent from all others. Exploiting these regular co-occurrences - giving the semantics context - could provide a better semantic description of music.

This suggests an extension of auto-tagging models by adding one additional layer of semantic representation that explicitly captures tag co-occurrences. We began by modeling the probability distribution of tags given audio features, placing each song in a semantic space. Now, by modeling a probability distribution of the SMNs derived from each song - a distribution over distributions - we can obtain a richer semantic description. We refer to these representations as *contextual models*.

1.3 Modeling Context

In this paper, we present a novel approach to automatically tagging music with descriptive words by thinking of each semantic concept as defining a broader *context* that causes multiple, related tags to co-occur in the description of a song. For each tag, we learn a Dirichlet mixture (DM) to model the distribution of the SMNs derived from all training songs for that tag. This DM-based "contextual tag model" is inspired by similar work on modeling the semantics of images [11] where it was proposed as a framework for combining object-centric and scene-centric methods to model contextual relationships between visual concepts. The DM can robustly infer contextually meaningful co-occurrence patterns between tags in semantic multinomials, while removing accidental co-occurrences that might be present in some of the individual song-level SMNs.

2. RELATED WORK

Some recent work in music information retrieval has exploited tag correlation and context. Yang et al. [16] formulate tag detection as an ordinal regression problem to explicitly take advantage of the ordinal relationship between concepts. Moreover, they proposed to leverage the co-occurrence patterns of tags for context fusion and employ tag selection to remove irrelevant or noisy tags. Unlike our approach, the latter is a single-level model, incorporating the tag correlation during the training of each individual detector. Ness et al. [10] propose a hierarchy of two linear SVMs where the first classifier highlighted the audio patterns and output a vector of tag affinities (analogous to a SMN), and the second layer modeled the contextual relationships between tags. Modeling context was also proposed in [7] where a second stage used a learning and correlation reweighing scheme to boost the result of tag detection, and, earlier, in [1] where authors used a decision tree to refine the result of individual detectors.

Our approach using the DM to model context is appropriate for two reasons. First, the DM is a *generative* model that is learned from only positive training examples i.e., songs which have been positively associated with a semantic tag. Unlike discriminative models (e.g., SVMs, boosting, decision trees) which also require negative examples, generative models can accommodate weakly labeled training data where the absence of an association between a song and a tag does not guarantee that no such association exists. Second, the Dirichlet is a distribution over parameters of the multinomial distribution, making it a probabilistically appropriate model of semantic multinomials derived from auto-taggers.

3. AUTO-TAGGING WITH DIRICHLET MIXTURES

We start by briefly defining the problem and by reviewing the song-level auto-tagging system described in [15].

3.1 Problem formulation

The task of semantic annotation and retrieval can be seen as a supervised multiclass, multilabel classification problem, where each class is a word w_i from a vocabulary $\mathcal{V} = \{w_1, ..., w_{|\mathcal{V}|}\}$ of unique tags, and each song is labeled with multiple words. A song is represented as a series of audio content features, $\mathcal{X} = \{x_1, ..., x_T\}$, where x_t represents a vector of features, and T is related to the length of the audio content; the goal is to find the words $w_i \in \mathcal{V}$ which best describe a given song. Each song can then be represented as an annotation vector $\pi = (\pi_1, ...\pi_{|\mathcal{V}|})$, where $\pi_i > 0$ if w_i has a positive semantic association with the song and $\pi_i = 0$ otherwise. The coefficients π_i represent the strength of semantic association between the song and word w_i and are termed *semantic weights* [15] or *affinity values* [10].

3.2 Defining a semantic space

Various auto-tagging methods have been proposed for deriving the semantic weights from acoustic features including hierarchical Gaussian mixture models [15], support vector machines [2, 10], codeword Bernoulli averaging [4] and boosting [7]. Any of these auto-taggers may be used to produce semantic multinomials — a set of semantic weights — that describe songs, a process that is illustrated on the left of Figure 2. In this work, we use the hierarchical GMM approach and briefly review it hereafter but refer the reader to [15] for the details of this model.

For each word w_i in the vocabulary, we train a tag-level probability distribution over the audio feature space, e.g. $P_{X|W}(x|w_i)$ for $i = 1, \ldots, |\mathcal{V}|$. The most relevant tags for a song $\mathcal{X}$ are the words with highest posterior probability, computed using Bayes' rule:

$$\pi_i = P_{W|X}(w_i|\mathcal{X}) = \frac{P_{X|W}(\mathcal{X}|w_i)\, P_W(w_i)}{P_X(\mathcal{X})}\,, \qquad (2)$$

where $P_W(w_i)$ is the prior of the i^{th} word. We assume an uniform prior, e.g., $P_W(w_i) = 1/|\mathcal{V}|$ for $i = 1, \ldots, |\mathcal{V}|$. We compute the song prior as $p(\mathcal{X}) = \sum_{i=1}^{|\mathcal{V}|} p(\mathcal{X}|w_i)p(w_i)$. We follow [15] in estimating the likelihood term in Equation 2, $P_{X|W}(\mathcal{X}|w_i)$, with the geometric average of the individual feature likelihoods of all the songs positively associated with word w_i:

$$P_{X|W}(\mathcal{X}|w_i) = \prod_{t=1}^{T} \left(P_{X|W}(x_t|w_i)\right)^{\frac{1}{T}}\,, \qquad (3)$$

where the distribution $P_{X|W}(x|w_i)$ is modeled as a mixture of Gaussians. The $P_{X|W}(x|w_i)$ distributions capture the patterns of audio content that are predictive of each word w_i.

Given an unseen test song, represented by a set of audio feature vectors $\mathcal{X}$, we compute the posterior probabilities for the presence of concept $w_i \in \mathcal{V}$ from Equation 2. Collecting the posterior probabilities of each word results in an annotation vector describing the song, $\pi = \{\pi_1, ..., \pi_{|\mathcal{V}|}\}$, where π_i denotes the posterior word probability $P_{W|X}(w_i|\mathcal{X})$. With appropriate normalization ($s.t. \sum_i \pi_i = 1$), this vector can be conceived of as a *semantic multinomial* (SMN) which lies on a probability simplex defined as a *semantic space*. The semantic multinomial is analogous to a *document vector* of word counts, often used in natural language processing [8], and it captures all the semantic information about the song.

3.3 A model to learn context

To capture the common patterns in the SMNs and model co-occurrences between tags, we learn *contextual tag models* in the semantic space from the SMNs of the all songs in a training set that have been labeled with each tag. This contextual modeling stage is illustrated on the right of Figure 2. Just as we modeled acoustic feature vectors as samples from a mixture of Gaussians, we consider that semantic multinomials π are drawn from a mixture of Dirichlet distributions over the semantic space [11]:

$$P_{\Pi|W}(\pi|w; \Omega^w) = \sum_k \beta_k^w \ \mathrm{Dir}(\pi|\alpha_k^w)\,, \qquad (4)$$

The contextual model for the word w is characterized by a vector of parameters $\Omega^w = \{\beta_k^w, \alpha_k^w\}$, where β_k is a probability mass function ($\sum_k \beta_k^w = 1$), $\mathrm{Dir}(\pi; \alpha)$ a Dirichlet distribution of parameter $\alpha = \{\alpha_1, ..., \alpha_{|V|}\}$,

$$\mathrm{Dir}(\pi|\alpha) = \frac{\Gamma(\sum_{i=1}^{|V|} \alpha_i)}{\prod_{i=1}^{|V|} \Gamma(\alpha_i)} \prod_{i=1}^{|V|} (\pi_i)^{\alpha_i - 1}\,, \qquad (5)$$

and $\Gamma(.)$ the Gamma function.

The parameters Ω^w are learned from the SMNs π_n of all the songs annotated with word w. Note that the contextual models $P_{\Pi|W}(\pi|w)$ play, in the semantic space, a similar role to the models $P_{X|W}(\mathcal{X}|w)$ in the acoustic feature space.

The learning process for the Dirichlet mixture model relies on the maximum likelihood estimation, via the generalized expectation-maximization (GEM) algorithm. GEM is an extension of the standard EM algorithm, applicable when the M-step of the latter is intractable. The E-step computes the expected values of the component probability distribution β_k, whereas the generalized M-step estimates the parameters α_k. Rather than solving for the parameters of maximum likelihood, each M-step simply produces an estimate of the likelihood which is higher than that available in the previous iteration. This is known to be sufficient for EM convergence [3]. Parameter estimation is achieved through an application of the Newton-Raphson algorithm [9].

Given an unseen test song described by the SMN $\pi = \{\pi_1, ..., \pi_{|\mathcal{V}|}\}$, the assignment of a word, w_i, results from a Bayes decision rule based on the posterior word probabilities in the context space:

$$P_{W|\Pi}(w_i|\pi) = \frac{P_{\Pi|W}(\pi|w_i)P_W(w_i)}{P_\Pi(\pi)}\,. \qquad (6)$$

Again we assume a uniform word prior probability $P_W(w_i)$. Collecting all the posterior probabilities

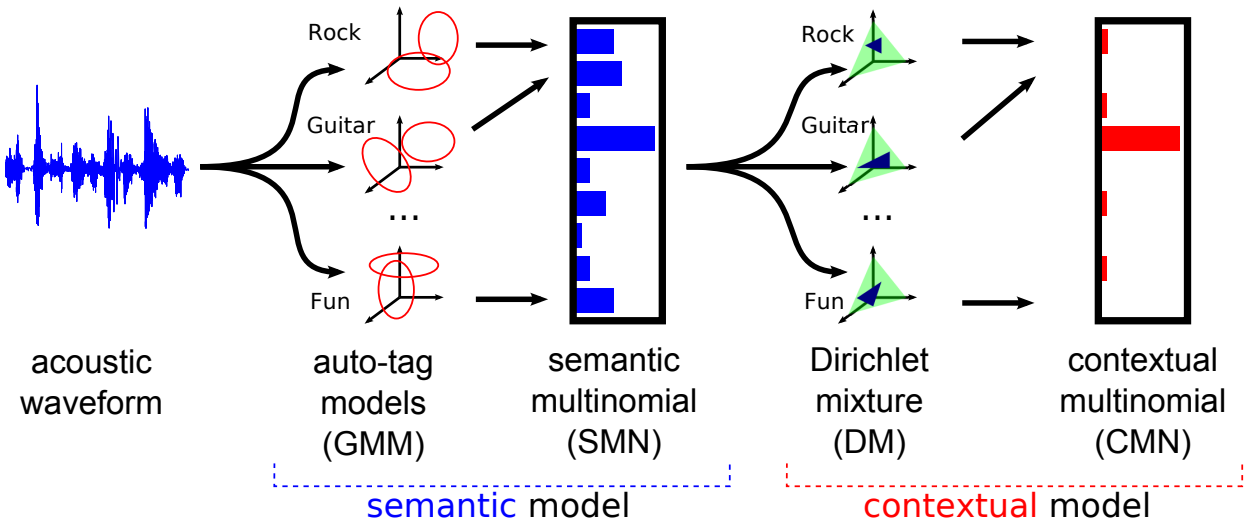


Figure 2. Overview of the system: the Dirichlet Mixture models context by considering co-occurrences patterns between auto-tags lying in a semantic space.

$P_{W|\Pi}(w_i|\pi) = \theta_i$ and normalizing ($s.t. \sum_i \theta_i = 1$), we build the vector $\theta = (\theta_1, ..., \theta_{|\mathcal{V}|})$, denoted as the *contextual multinomial* (CMN) distribution of a song. Similar to the semantic space defined in Section 3.2, CMN vectors lies in a *contextual space* (see Figure 2).

4. EXPERIMENTAL RESULTS

In this section, we demonstrate the impact of contextual models, and in particular the DM, on automatically tagging music with meaningful words.

4.1 CAL500 Dataset

The Computer Audition Lab 500 (CAL500) [15] dataset comprises 502 songs by 502 different artists. Each song has been annotated by at least 3 humans using a vocabulary composed of 174 tags from 6 different semantic categories, representing both objective and subjective concepts.

The songs are described by Mel-Frequency Cepstral Coefficient (MFCC) feature vectors; each MFCC vector summarizes the spectral content of 23ms windows of a song. Our experiments use 39-dimensional MFCC-Delta feature vectors, composed by appending the first and second instantaneous derivatives to the 13-component MFCCs.

A first analysis of the dataset demonstrates an imbalance in the distribution of tags: while frequent tags can have more than 300 positive examples, some others have less than 10 ones. This is not a big problem when training auto-taggers since each song is described by a large number of features vectors. However, the resulting set of SMNs describing songs is much smaller than the number of feature vectors and thus, we require more songs to adequately train the contextual models. For this reason, our evaluation considers only the tags with more than 30 examples, aiming to have at least $20-25$ examples in the training set with the remainder in the test set. This reduces the CAL500 vocabulary to 97 tags: 11 genres, 14 instruments, 25 acoustic qualities, 6 vocal characteristics, 35 emotions and 6 usages.

To provide sufficient data to train the DM, we extract multiple SMNs from each song, each derived from clips lasting 3 seconds. We find empirically that, unlike images which generally depict only a few semantic concepts (i.e., their SMNs have a few peaks that dominate all other tags), even a short music clip can be reasonably tagged with many words and the resulting SMNs tend to be much more uniform. For this reason, when learning DM models, we threshold the SMNs, retaining at most the ten largest affinity values and setting all other dimensions to zero.

4.2 Annotation and Retrieval

We evaluate auto-tagging performance on both annotation and retrieval tasks. In the *annotation* task, we use Equation 6 to label each test song with the ten most likely tags. Performance is measured using mean per-tag precision, recall and F-score. Per-tag precision is the probability that a tag used by the model is correctly applied to a song. Per-tag recall is the probability that the model annotates all the tags that should apply to a song. F-score is the harmonic mean of precision and recall, and is a single measure of overall annotation performance.

In the *retrieval* task, we rank-order all songs according to their relevance to a query tag. The retrieval goal is to have highly relevant songs at the top of the ranking list as this is the most crucial requirement in a music retrieval system. We consider the mean average precision (MAP) and the *precision at k* (k = 3, 5, 10). For completeness, we also report the area under the receiver operating characteristic curve (AROC) as a measure of the quality of the complete ranking [8].

Evaluation was performed using 5-fold cross validation, with 400 songs in the training set, and 100 in the test set. The folds were built such that each song appeared in the test set exactly once. The results reported in Table 1 display the annotation and retrieval metrics, averaged over all tags in the vocabulary.

		Annotation			Retrieval				
		Precision	Recall	F-Score	P3	P5	P10	MAP	AROC
Semantic	CBA	0.361	0.212	0.267	0.463	0.458	0.440	0.425	0.691
	GMM	0.405	0.202	0.269	0.456	0.455	0.441	0.433	**0.698**
Context	SVM	0.380	0.230	0.286	0.512	0.487	0.449	0.434	0.687
	DM	**0.441**	**0.232**	**0.303**	**0.519**	**0.501**	**0.470**	**0.443**	0.697
Upper Bound		0.716	0.471	0.568	1.000	0.993	0.942	1.000	1.000
Random		0.231	0.101	0.140	0.255	0.249	0.250	0.277	0.504

Table 1. Performance of different auto-taggers: the Codeword Bernoulli Average (CBA) and Gaussian Mixture Models (GMM) consider semantics alone whereas the Support Vector Machine (SVM) and Dirichlet Mixture (DM) models learn contextual relationships between the semantic multinomials produced by the GMM. All experiments were performed on the same songs represented by the same set of features. "Random" is a baseline that annotates and ranks songs randomly. "Upper Bound" uses the optimal labeling for each evaluation metric and shows the upper limit on what any system could achieve.

4.3 Contextual improvement

The proposed contextual modeling approach is compared to some recent state of the art auto-tagging approaches: the GMM model [15] alone (i.e., without context) and the CBA model [4]. For the CBA model, each song is represented as a histogram over a codebook of 500 vector-quantized MFCCs. For each fold we trained the codebook models only on the songs in the training set. All the code was provided by the authors of [4].

We see in Table 1 that there is significant benefit from modeling context on almost all annotation and retrieval metrics. In particular, the precision-at-k metrics demonstrate improvements at the top of the ranked retrieval list but not throughout list (based on AROC). It can be argued that precision-at-k metrics consider the part of the ranked list which is most interesting for users of a semantic music retrieval engine.

4.4 DM as a model of context

The center rows of Table 1 compare the DM approach for modeling semantic co-occurrences to a Support Vector Machine (SVM). As with the DM, we trained a contextual SVM for each tag using the semantic multinomials as the input feature vector. Using SVM as a model of context was first proposed in [10] although their approach differs in the features used (median MFCC texture windows) and in the semantic model (SVM), so our results do not present a direct comparison with [10]. Our goal is simply to compare the DM and SVM as models of contextual relationships. The context SVM does not benefit from pre-processing the SMNs (results not shown), thus SVMs are trained on all the original semantic values. Table 1 shows that DM generally improves on all the metrics and never performs worse. In particular, the DM significantly improves on the SVM for the annotation precision, F-score, P5, P10 and AROC metrics (t-test, 10% significance level); all the other metrics generally improve and never perform significantly worse.

Table 2 breaks up the evaluation over the different tag categories. As can be seen, all categories but "Genre" show clear benefit from contextual modeling. Note that improvements are related to the tag co-occurrences depicted in Figure 1. In fact, all the categories showing a high degree of co-occurrences ("Emotion", "Instrument" and "Acoustic") improved with respect to the GMM. Though not exhibiting as much co-occurrence, the "Usage" and "Vocals" categories, which perform poorly using semantics alone, benefit from the de-noising effect of learning contextual relations. In these cases, the extra information from even only few co-occurrences can lead to improvements in the quality of auto-tagging. Conversely, since the "Genre" category does not exhibit much co-occurrence (i.e., genres are more exclusive), we do not gain benefit from additional contextual modeling. It has to be noted that SVM performs better for the "Genre" category, especially in the top of the ranking list; we believe that in this case SVM benefits from some de-noising effects that DM is not able to capture.

4.5 Predictive co-occurrences

Finally, we include some examples of learned contextual models for 6 tags, representing each semantic category in CAL500. Table 3 shows the top three semantic multinomial dimensions that have most influence on the contextual models for each tag. These examples illustrate how the DM uses context to improve automatic tagging by learning to put most weight on semantic dimensions that are predictive of the tag being modeled e.g., "calming, low energy, mellow" music is good for "going to sleep". This demonstration of the dependence between tags indicates the importance of including context when modeling the relationship between semantics and music.

5. CONCLUSIONS

In this paper we have presented the Dirichlet mixture model, a novel approach for improving automatic music tagging by effectively modeling contextual relationships among tags. Starting from the SMN of each song, the DM adds an additional layer to model tag co-occurrences, giving context to the semantic representations derived from acoustic content. A tag's affinity with a song is computed as the posterior probability under the tag's DM model. The

Category	# Tags	Model	P5	P10	MAP
Emotion	35	GMM	0.513	0.506	0.477
		SVM	0.539	0.514	0.481
		DM	**0.561**	**0.535**	**0.489**
Genre	11	GMM	0.367	0.325	**0.355**
		SVM	**0.396**	**0.336**	0.350
		DM	0.360	0.331	0.341
Instrument	14	GMM	0.460	0.431	0.441
		SVM	0.495	0.452	0.455
		DM	**0.506**	**0.458**	**0.463**
Acoustic	25	GMM	0.508	0.501	0.472
		SVM	0.524	0.516	0.471
		DM	**0.564**	**0.546**	**0.496**
Usage	6	GMM	0.253	0.233	0.258
		SVM	0.266	0.226	0.237
		DM	**0.308**	**0.273**	**0.281**
Vocals	6	GMM	0.253	0.240	0.261
		SVM	0.260	0.210	0.235
		DM	**0.287**	**0.267**	**0.278**

Table 2. Retrieval results considering the different word categories for the semantic GMM, and the contextual SVM and DM models.

Context Tag	Semantic Influence		
calming	low energy	tender	slow tempo
hard rock	hard rock	rock	strong
acoustic guitar	slow tempo	tender	acoustic guitar
acoustic texture	low energy	soft rock	light beat
going to sleep	calming	low energy	mellow
emotional	tender	sad	soft rock

Table 3. Examples of the top three semantic influences on contextual tag models.

set of all posterior tag probabilities provides a contextual description of the song.

Experiments reported that modeling context outperforms approaches based on a semantic representation alone, especially considering the top of the ranked retrieval lists. We demonstrate that the DM is an appropriate choice for modeling semantic context by comparison to learning context with an SVM. More specifically, examining the performance across semantic categories, we showed that the DM improves performance for tags that exhibit a high degree of correlation, as well as for noisy tags that are poorly represented by acoustic patterns.

6. REFERENCES

[1] J.J. Aucouturier, F. Pachet, P. Roy, and A. Beuriv. Signal + context = better classification. In *Proceedings of ISMIR*, 2007.

[2] L. Barrington, M. Yazdani, D. Turnbull, and G. Lanckriet. Combining feature kernels for semantic music retrieval. In *Proceedings of ISMIR*, 2008.

[3] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–38, 1977.

[4] M. Hoffman, D. Blei, and P. Cook. Easy as CBA: A simple probabilistic model for tagging music. In *Proceedings of ISMIR*, 2009.

[5] P. Knees, T. Pohle, M. Schedl, and G. Widmer. A music search engine built upon audio-based and web-based similarity measures. In *Proceedings of ACM SIGIR*, 2007.

[6] P. Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2), 2008.

[7] T.B. Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger: a model for predicting social tags from acoustic features on large music database. *Journal of New Music Research*, 37(2), 2008.

[8] C.D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[9] T. Minka. Estimating a Dirichlet distribution. 2009. http://research.microsoft.com/en-us/um/people/minka/papers/dirichlet/.

[10] S.R. Ness, A. Theocharis, G. Tzanetakis, and L.G. Martins. Improving automatic music tag annotation using stacked generalization of probabilistic SVM outputs. In *Proceedings of ACM MULTIMEDIA*, 2009.

[11] N. Rasiwasia and N. Vasconcelos. Holistic context modeling using semantic co-occurences. In *Proceedings of IEEE CVPR*, 2009.

[12] M. Sordo, C. Laurier, and O. Celma. Annotating music collections how content-based similarity helps to propagate labels. In *Proceedings of ISMIR*, 2007.

[13] C.F. Tsai and C. Hung. Automatically annotating images with keywords: a review of image annotation systems. *Recent Patents on Computer Science*, 1, 2008.

[14] D. Turnbull, L. Barrington, and G. Lanckriet. Five approaches to collecting tags for music. In *Proceedings of ISMIR*, 2008.

[15] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE TASLP*, 16(2):467–476, February 2008.

[16] Y.H. Yang, Y.C. Lin, A. Lee, and H. Chen:. Improving musical concept detection by ordinal regression and context fusion. In *Proceedings of ISMIR*, 2009.

IMPROVING MARKOV MODEL-BASED MUSIC PIECE STRUCTURE LABELLING WITH ACOUSTIC INFORMATION

Jouni Paulus
Fraunhofer Institute for Integrated Circuits IIS
Erlangen, Germany
jouni.paulus@iis.fraunhofer.de

ABSTRACT

This paper proposes using acoustic information in the labelling of music piece structure descriptions. Here, music piece structure means the sectional form of the piece: temporal segmentation and grouping to parts such as chorus or verse. The structure analysis methods rarely provide the parts with musically meaningful names. The proposed method labels the parts in a description. The baseline method models the sequential dependencies between musical parts with N-grams and uses them for the labelling. The acoustic model proposed in this paper is based on the assumption that the parts with the same label even in different pieces share some acoustic properties compared to other parts in the same pieces. The proposed method uses mean and standard deviation of relative loudness in a part as the feature which is then modelled with a single multivariate Gaussian distribution. The method is evaluated on three data sets of popular music pieces, and in all of them the inclusion of the acoustic model improves the labelling accuracy over the baseline method.

1. INTRODUCTION

This paper proposes a method for providing musically meaningful labelling to sectional parts in Western popular music using two complementary statistical models. The first one relies on the sequential dependencies between the occurrences of different parts, while the second models some acoustic properties of the them. A labelling method using the sequence model was proposed earlier by Paulus and Klapuri [9] and this paper proposes an extension that method by including also acoustic information.

In sectional form a music piece is constructed from shorter, possibly repeated *parts*. Especially many Western pop/rock pieces follow this form. The parts can be named according to the musical role they have in the piece, for example, "intro" is in the beginning of the piece and provides an introduction to the song and "verse" tells the main story of the song. Music piece structure analysis aims to provide a description of the sectional form of the piece based on the acoustic signal. Usually the description consists of a temporal segmentation of the piece to occurrences of parts, and of grouping of segments being occurrences of the same part. For a review of methods proposed for the task, see the book chapter by Dannenberg and Goto [2] or the dissertation by Paulus [8]. With the exception of few methods [6, 14], most structure analysis methods do not provide the segment groups with *musically meaningful label*, instead they only provide a *tag* for distinguishing the different groups. However, if the analysis result is presented for a user, providing also meaningful labels for the segments would be valued, as noted by Boutard et al. [1].

A method for musical part labelling given the description with arbitrary tags was proposed by Paulus and Klapuri [9]. It relies on the assumption that musical parts have sequential dependencies which are then modelled with N-grams. The method searches for the labelling that maximises the overall N-gram probability over the resulting label sequence. The obtained results indicate that such a model manages to capture useful information of the music piece structures. This paper proposes to extend that work by including acoustic information in the process. This is motivated by the frequently encountered assumption that the chorus is louder than the other parts. It should be noted that this paper does not discuss the underlying problems in defining the structural description that have been discussed by Peeters and Deruty [11], but instead studies the performance of the proposed models in replicating the labelling in the manual annotations.

The rest of this paper is organised as follows: Sec. 2 describes the labelling problem more formally, revisits the sequential modelling baseline method, and details the proposed acoustic modelling method. Sec. 3 describes the experiments for evaluating the proposed method and presents the obtained results. Finally Sec. 4 provides the conclusions of this paper.

This work was performed when the author was at the Department of Signal Processing, Tampere University of Technology, Tampere, Finland. This work was supported by the Academy of Finland, (application number 129657, Finnish Programme for Centres of Excellence in Research 2006–2011).

2. PROPOSED METHOD

This section provides a more formal definition of the labelling problem, provides a short description of the baseline method relying only on sequence modelling, and details the proposed acoustic modelling extension.

2.1 Labelling Problem

The input to the method consists of a music piece description and the acoustic signal. The description itself is a temporal segmentation of the piece and a grouping of the segments. Each of the groups is assigned with a unique tag r. When the K tags in the description are organised into a sequence based on the temporal locations of the segments, a tag sequence $r_{1:K} \equiv r_1, r_2, \ldots, r_K$ is obtained. The problem of label assignment is to find an injective [1] mapping

$$f : R \rightarrow L \tag{1}$$

from the set R of tags present in the description to the set L of musically meaningful labels. Application of the mapping is denoted with

$$f(r) = l, \tag{2}$$

and it can be done also on sequences:

$$f(r_{1:K}) = l_{1:K}. \tag{3}$$

Since any injective mapping is a valid mapping from tags to labels, the problem is to select the "best" mapping from all the possible choises. The earlier publication [9] proposed a statistical sequence model for the the labels l for selecting the mapping producing the highest model probability. This paper proposes to include acoustic information to the process of selecting the mapping function.

2.2 Markov Model Baseline Method

Some sectional forms are more common in music than the others. An example of this was presented in [9] where it was noted that almost 10% of the songs by The Beatles have the form "intro", "verse", "verse", "bridge", "verse", "bridge", "verse", "outro". Though this cannot be directly generalised to all pieces, some sequences of parts occur more frequently than others and this can be utilised in the labelling.

In sequence modelling the prediction problem is to provide probabilities for the possible continuations of a given sequence. $p\left(s_i|s_{1:(i-1)}\right)$ denotes the conditional probability of s_i to follow the sequence $s_{1:(i-1)}$. Markov models make the assumption that the process has a limited memory and the probabilities depend only on a limited length history. The length of the history is parametrised with N which provides a motivation for the alternative name of N-grams. An N-gram of length N utilises $N-1^{th}$ order Markov assumption

$$p\left(s_i|s_{1:(i-1)}\right) = p\left(s_i|s_{(i-N+1):(i-1)}\right). \tag{4}$$

[1] All tags in input sequence are mapped to a label, but each tag can be mapped only to one label and no two tags may be mapped to same label.

Given a sequence $s_{1:K}$ and the conditional N-gram probabilities the total probability of a sequence can be calculated with

$$p(s_{1:K}) = \prod_{i=1}^{K} p\left(s_i|s_{(i-N+1):(i-1)}\right). \tag{5}$$

For more information on N-grams and language modelling, see [5].

The baseline method proposed by Paulus and Klapuri [9] calculates N-grams using the musical part labels as the alphabet L, and then locates the mapping f_{OPT} maximising the overall sequential probability of (5) while conforming to the injectivity constraint:

$$f_{\mathrm{OPT}} = \underset{\mathrm{f}}{\operatorname{argmax}} \left\{ p_{\mathrm{L}}\left(f|r_{1:K}\right) \right\}, f : R \rightarrow L \text{ injective.} \tag{6}$$

In (6) $p_{\mathrm{L}}\left(f|r_{1:K}\right)$ denotes the Markov probability of the sequence resulting from applying the mapping f

$$p_{\mathrm{L}}\left(f|r_{1:K}\right) = p(f(r_{1:K})). \tag{7}$$

The combinatorial optimisation problem of (6) can be solved, e.g., in a greedy manner by applying a variant of N-best token passing algorithm proposed in [9], or by applying the Bubble token passing algorithm proposed in [10]. Both operate on the same basic principle of creating a directed acyclic graph from the parts and possible labellings, and searching a path through it. Each part in the sequence is associated with each possible label and these combinations form the nodes of the graph. Edges are created between parts that are directly consecutive in the input sequence. Paths through the graph represent label mappings, and the path with the highest probability is returned as the result. Even though the search does not guarantee finding the optimal solution, in small experiments it found the same solution as an exhaustive search with a fractional computational cost. Viterbi or similar more efficient search algorithm cannot be employed here as the mapping has to respect the injectivity and the whole sequence history affects the probabilities instead of only the limited memory of N-grams.

2.3 Sequence Modelling Issues

The number of conditional probabilities $p\left(s_i|s_{1:(i-1)}\right)$ that need to be estimated for N-gram modelling increases rapidly as a function of the model order N and the alphabet size V: there are V^N probabilities that need to be estimated. Usually, the probabilities are estimated from a limited amount of training data, and not all probabilities can be estimated reliably. This problem can be partly alleviated by applying smoothing to the probabilities (assigning some of the probability mass of the more frequently occurring combinations to the less frequent ones), or by discounting methods (estimating high-order models as combinations of lower-order models). Variable-order Markov models (VMMs) [13] attempt solving the model order problem based on the training data by setting the order independently to different subsequences. In other words, if increasing the model order does not bring more accurate information, it is not done.

2.4 Acoustic Modelling Method

The baseline method operates only on the sequential information of the musical parts and has no information of the actual content of them. However, if the acoustic signal is available, it can be utilised in the labelling. Naturally, the parts of a song differ from each other in view of the acoustic properties. This is closely related to the definition of sectional form. However, the assumption made here is that there exists acoustic properties that exhibit similar behaviour in large body of the pieces, e.g., it is often stated that "chorus" is the most energetic, or the loudest, part in a song. In addition to "chorus" being most energetic, very few other parts can be said to have any typical acoustic property. Still, e.g., "break" or "breakdown" often has considerably reduced instrumentation, thus it is expected that it exhibits a lower average loudness than the other parts. Despite this, the acoustic modelling is applied to all parts even though is might not produce meaningful information for all labels.

The proposed acoustic modelling represents the acoustic information by associating a single observation vector $\mathbf{x}_i$ to each of the musical parts, thus utilising a highly condensed representation. The input to the labelling now consists of the tag sequence $r_{1:K}$ and acoustic observations $\mathbf{x}_{1:K}$, one vector $\mathbf{x}_i$ for each part. The acoustic model considers now the likelihoods $p_{\mathrm{A}}(\mathbf{x}_i|l)$ of observing $\mathbf{x}_i$ if the musical part label is l. The overall likelihood of the mapping definition f in view of the acoustic observations $\mathbf{x}_{1:K}$ is now calculated with

$$p_{\mathrm{A}}(f|r_{1:K},\mathbf{x}_{1:K}) = \prod_{i=1}^{K} p_{\mathrm{A}}(\mathbf{x}_i|f(r_i)). \tag{8}$$

2.5 Combined Method

Assuming statistical independence, combining the two models (7) and (8) in the same function produces a new likelihood function for the mapping f

$$p(f|r_{1:K},\mathbf{x}_{1:K}) = p(\mathbf{x}_{1:K}|f(r_{1:K}))\,p(f(r_{1:K})) \tag{9}$$

$$= \prod_{i=1}^{K} p(\mathbf{x}_i|f(r_i)) \prod_{i=1}^{K} p\left(f(r_i)|f(r_{1:(i-1)})\right), \tag{10}$$

where the first term is from the acoustic observations and the latter from the N-gram models. The labelling problem can be expressed as the optimisation task

$$f_{\mathrm{OPT}} = \underset{\mathrm{f}}{\operatorname{argmax}} \{p(f|r_{1:K},\mathbf{x}_{1:K})\}, f : R \to L \text{ injective}. \tag{11}$$

The optimisation of (11) can be done with the same algorithm as the optimisation of the sequential model alone. The only required modification is to include the acoustic observation likelihoods. It should be noted that even though the problem resembles hidden Markov model decoding, the injectivity requirement violates the Markov assumption thus prohibiting the use of Viterbi decoding.

2.6 Acoustic Features

As the assumption about the globally informative acoustic property was related to the energy level or loudness, they were tested for the acoustic modelling. The energy is measured by calculating the root-mean-squared value of the signal within the part. However, in preliminary experiments it was noted that using perceived loudness instead produced better results. This is presumably because the loudness calculation addresses also the non-linear properties of human auditory system in amplitude, frequency, and temporal dimensions, the main difference being in the dynamic amplitude scale compression from representing the data in logarithmic decibel scale. [2] The calculation is done using the function `ma_sone` from the MA Toolbox by Pampalk [7]. The loudness is calculated in 11.6 ms frames with 50% overlap and the part loudness is approximated by the mean loudness of the frames within the part in question. In addition to the mean loudness also standard deviation of the framewise loudness values over the part is used to describe the dynamics of the signal. The features are normalised by dividing them by the mean over the piece making the mean over the piece to be 1. An illustration of the feature distributions is provided in Fig. 1.

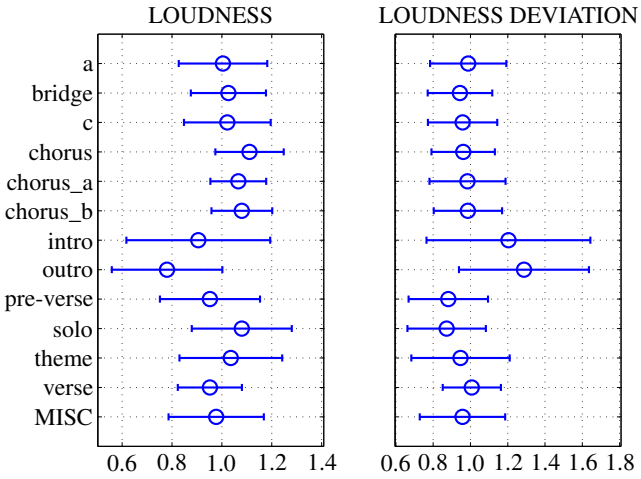


Figure 1. Statistics of the features used in data set *TUT-structure07*. The mean of all occurrences of the part is indicated with circle and the surrounding error bars illustrate the standard deviation over the occurrences. Note that the mean loudness of "chorus" and it's variations support the original assumption.

The acoustic observation likelihoods $p_{\mathrm{A}}(\mathbf{x}|l)$ are modelled as a single multivariate Gaussian distribution

$$p_{\mathrm{A}}(\mathbf{x}|l) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu)\right), \tag{12}$$

where D is the feature vector dimensionality, Σ and μ are the covariance matrix and mean vector of the estimated distribution of the part label l.

[2] The preliminary experiments included also acoustic features corresponding to the brightness (spectral centroid) and bandwidth of the signal. The various combinations of different features were tested and based on the results of the small-scale experiments, the set used was limit to loudness and it's deviation.

3. EVALUATIONS

The proposed extension is evaluated with three data sets of popular music pieces. The first set *TUTstructure07* consists of 557 pieces from various genres, mainly from pop and rock, but including also pieces from metal, hip hop, schlager, jazz, blues, country, electronic, and rnb. The pieces have been manually annotated at Tampere University of Technology (TUT). [3] The second data set *UPF Beatles* consists of 174 pieces by The Beatles. The piece forms were analysed by Alan W. Pollack [12], and the part time stamps were later added at Universitat Pompeu Fabra (UPF) and TUT. [4] The third data set *RWC pop* contains 100 pieces from the Real World Computing Popular Music collection [3, 4] aiming to represent typical 1980's and 1990's chart music from Japan and USA.

3.1 Evaluation Setup

Since the ground truth annotations in the data sets originate from different sources, the used labels also differ. For this reason the evaluations are run separately for each data set. The data sets contain relatively large number of unique part labels (e.g., *TUTstructure07* has 82 unique labels) some of which occur very rarely making the modelling more difficult. To alleviate this problem only the most frequent labels contributing to 90% of all part occurrences are retained, and the rest are replaced with an artificial label "MISC". This reduces the number of labels considerably (e.g., to 13 in *TUTstructure07*). The evaluations are run in leave-one-out cross-validation scheme and the presented results are calculated over all folds.

The performance is evaluated with per-label accuracy, which is the ratio of the sum of durations of correctly identified label occurrences to the sum of durations of all occurrence of the label, calculated over the entire data set. Similarly, the total accuracy describes how much of the entire data set duration is labelled correctly, effectively applying weighting to the more frequently occurring labels, such as "chorus".

It should be noted that the segmentation to the input tag sequence $r_{1:K}$ is obtained from the ground truth annotations instead of an automatic signal-based analysis method. This is done to be enable evaluating the accuracy of the labelling method independent of the segmentation performance.

The complementary aspects of the proposed method are evaluated: sequence modelling alone (effectively reproducing the results from [9]), acoustic modelling alone, and the two combined. The sequence modelling is attempted with N-gram length of 1 to 5 (from only prior probabilities to utilising history of length 4), and with a variable-order Markov model. The VMM method employed was decomposed context tree weighting after the earlier results, and the implementation was from [13]. These results operate as the baseline on top of which the acoustic modelling is added. The sequence modelling choises were done to follow the experiments in the earlier paper, thus providing a clear baseline for comparing the effect of the added acoustic model.

3.2 Results

The evaluation results are presented in Tables 1–3, each table containing the results for a different data set. The column denoted with "N=0" provides the result for using only the proposed acoustic model, while the other columns contain the results of the combined modelling with different N-gram lengths. The results of using only the sequence model are provided in parentheses.

The results indicate that including the acoustic information into the labelling model improves the result in some cases. In all data sets the best overall result is obtained by including the acoustic information, though the improvement in *UPF Beatles* is so small that it may not be statistically significant. [5] The same relatively small obtained improvement is observed in the results for individual labels in *UPF Beatles*. This may be because the pieces are from a single band mainly from the 1960's and thus may not exhibit all the stereotypical properties found in more modern pop music, as noted also by Peeters [11]. The improvement in *TUTstructure07* is slightly larger. It is assumed that the lower impact of the acoustic model is partly caused by the large variety of musical styles present in the data, thus the modelling assumption may not hold in all cases. The improvement due to the inclusion of the acoustic model is most prominent with the *RWC pop* data which represents more typical chart music.

4. CONCLUSIONS

This paper has presented a method for assigning musically meaningful labels music piece structure descriptions. The baseline method utilises the sequential dependencies between musical parts. This paper proposes a simple acoustic model for the labelling and combines it with the sequential modelling method. The proposed method is evaluated on three data sets of real popular music. The obtained results support the original assumption that musical parts differ in their loudness, and the acoustic information alone can be used to some extent to label the parts. The acoustic information alone has the labelling performance in par with using only part occurrence priors. Combining the acoustic model with the baseline sequential model provides in most cases a improvement in the accuracy. However, the improvement cannot be obtained with all data, because typical loudness relations between different parts seem to depend on the musical genre. Finally, the same search algorithm as with the baseline method can be used for the combined model with very small modifications.

[3] A full list of pieces is available at `http://www.cs.tut.fi/sgn/arg/paulus/TUTstructure07_files.html`.

[4] The annotations are available at `http://www.iua.upf.edu/%7Eperfe/annotations/sections/license.html`, and including some corrections at `http://www.cs.tut.fi/sgn/arg/paulus/structure.html#beatles_data`.

[5] As the entire data set forms one instance in the evaluation measure calculation, no statistical measure could be calculated for proper comparison.

	N=0	N=1	N=2	N=3	N=4	N=5	VMM
a	0.0	0.0 (0.0)	0.8 (0.0)	22.2 (**34.9**)	23.8 (31.7)	27.8 (27.0)	24.6 (29.4)
bridge	18.6	25.8 (17.9)	47.4 (38.6)	51.1 (45.4)	**50.2** (47.4)	49.1 (43.9)	47.7 (41.4)
c	3.3	13.5 (3.6)	41.6 (38.3)	44.6 (42.1)	47.4 (47.7)	**56.2** (54.8)	49.6 (48.5)
chorus	29.5	75.5 (67.9)	83.4 (76.3)	**85.0** (80.6)	82.7 (76.6)	79.8 (75.3)	82.4 (77.9)
chorus_a	11.9	0.0 (0.0)	0.0 (0.0)	8.2 (7.5)	**15.7** (**15.7**)	**15.7** (11.2)	0.0 (3.0)
chorus_b	4.4	0.0 (0.0)	0.9 (0.9)	8.8 (5.3)	**12.4** (**12.4**)	**12.4** (7.1)	0.9 (2.7)
intro	32.7	43.4 (22.7)	97.2 (97.6)	97.6 (98.2)	97.0 (97.8)	**98.2** (97.8)	97.0 (96.8)
outro	52.4	47.4 (9.9)	98.3 (98.3)	**98.6** (**98.6**)	98.3 (97.6)	95.9 (90.9)	98.1 (98.3)
pre-verse	30.1	10.0 (3.7)	51.4 (40.5)	**55.6** (45.6)	50.7 (43.3)	46.8 (41.7)	52.5 (42.6)
solo	23.8	2.2 (0.0)	6.6 (4.4)	6.6 (7.2)	13.8 (15.5)	**23.2** (18.8)	21.0 (16.0)
theme	5.5	0.0 (0.0)	2.7 (0.0)	2.7 (2.7)	2.7 (4.4)	**6.6** (3.3)	3.3 (0.5)
verse	46.5	59.0 (38.4)	72.5 (62.6)	71.0 (64.6)	70.6 (64.5)	72.1 (64.7)	**74.5** (65.4)
MISC	7.8	21.4 (11.7)	35.5 (29.2)	**44.4** (38.6)	42.8 (37.9)	41.7 (40.6)	40.3 (37.3)
total	27.7	42.1 (29.6)	61.7 (55.6)	**64.3** (60.2)	63.6 (59.9)	63.6 (59.3)	63.7 (59.2)

Table 1. Per-label accuracy (%) on *TUTstructure07* obtained using only acoustic modelling (N=0 column), only sequence modelling (values in parentheses), and combining sequence and acoustic modelling (other values).

	N=0	N=1	N=2	N=3	N=4	N=5	VMM
bridge	6.2	22.0 (24.3)	48.0 (45.8)	**77.4** (76.8)	75.1 (75.1)	70.1 (74.0)	69.5 (69.5)
intro	43.8	50.0 (41.4)	92.6 (92.0)	93.2 (92.6)	**93.8** (**93.8**)	**93.8** (**93.8**)	93.2 (93.2)
outro	73.2	60.6 (0.0)	**99.3** (**99.3**)	**99.3** (**99.3**)	98.6 (97.9)	97.9 (93.7)	**99.3** (**99.3**)
refrain	20.1	30.1 (28.1)	43.8 (45.4)	61.8 (62.2)	69.1 (69.9)	65.1 (67.5)	69.1 (**70.3**)
verse	37.2	73.4 (70.6)	80.9 (81.5)	**88.5** (87.9)	86.5 (85.3)	83.7 (84.5)	87.1 (87.5)
verses	23.2	0.0 (0.0)	8.9 (8.9)	51.8 (**53.6**)	37.5 (37.5)	44.6 (44.6)	42.9 (42.9)
versea	**39.2**	0.0 (0.0)	2.0 (2.0)	7.8 (7.8)	23.5 (19.6)	25.5 (19.6)	11.8 (11.8)
MISC	5.7	3.8 (4.5)	17.8 (17.2)	28.7 (29.3)	**43.9** (37.6)	26.1 (25.5)	30.6 (29.9)
total	31.1	43.7 (36.1)	61.8 (61.8)	73.8 (73.7)	**75.7** (74.6)	71.9 (72.4)	73.6 (73.9)

Table 2. Per-label accuracy (%) on *UPF Beatles* obtained using only acoustic modelling (N=0 column), only sequence modelling (values in parentheses), and combining sequence and acoustic modelling (other values).

	N=0	N=1	N=2	N=3	N=4	N=5	VMM
bridge a	20.1	20.1 (8.2)	72.3 (62.9)	**73.6** (66.7)	64.8 (66.0)	59.7 (49.7)	71.7 (62.9)
chorus a	51.2	70.9 (45.6)	85.3 (76.2)	85.6 (77.6)	80.6 (73.5)	73.5 (71.5)	**86.2** (79.7)
chorus b	28.0	37.5 (6.5)	79.2 (73.2)	**79.8** (71.4)	72.6 (65.5)	72.0 (71.4	76.2 (72.0)
ending	80.6	84.7 (32.7)	**100** (**100**)	99.0 (**100**)	98.0 (94.9)	99.0 (88.8)	**100** (99.0)
intro	50.0	45.1 (10.8)	**100** (**100**)	**100** (**100**)	**100** (**100**)	**100** (**100**)	**100** (**100**)
pre-chorus	7.6	7.6 (3.3)	**64.1** (51.1)	60.9 (52.2)	63.0 (39.1)	48.9 (42.4)	63.0 (45.7)
verse a	35.0	48.1 (20.3)	**85.7** (76.8)	84.4 (78.9)	81.4 (78.1)	77.6 (73.4)	81.0 (76.4)
verse b	19.4	30.3 (17.4)	**85.6** (76.6)	84.1 (80.1)	79.6 (74.6)	73.6 (69.2)	82.1 (76.6)
verse c	41.9	14.0 (0.0)	**60.5** (30.2)	55.8 (39.5)	47.7 (30.2)	32.6 (30.2)	47.7 (33.7)
MISC	29.8	52.4 (8.4)	**84.9** (67.6)	80.4 (73.8)	77.8 (68.4)	69.8 (67.6)	83.1 (74.7)
total	36.0	45.5 (19.1)	**82.8** (72.8)	81.7 (75.3)	77.5 (70.9)	71.8 (68.0)	80.7 (74.1)

Table 3. Per-label accuracy (%) on *RWC pop* obtained using only acoustic modelling (N=0 column), only sequence modelling (values in parentheses), and combining sequence and acoustic modelling (other values).

5. REFERENCES

[1] Guillaume Boutard, Samuel Goldszmidt, and Geoffroy Peeters. Browsing inside a music track, the experimentation case study. In *Proc. of 1st Workshop on Learning the Semantics of Audio Signals*, pages 87–94, Athens, Greece, December 2006.

[2] Roger B. Dannenberg and Masataka Goto. Music structure analysis from acoustic signals. In David Havelock, Sonoko Kuwano, and Michael Vorländer, editors, *Handbook of Signal Processing in Acoustics*, volume 1, pages 305–331. Springer, New York, N.Y., USA, 2008.

[3] Masataka Goto. AIST annotation for the RWC music database. In *Proc. of 7th International Conference on Music Information Retrieval*, pages 359–360, Victoria, B.C., Canada, October 2006.

[4] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, classical, and jazz music databases. In *Proc. of 3rd International Conference on Music Information Retrieval*, pages 287–288, Paris, France, October 2002.

[5] Daniel Jurafsky and James H. Martin. *Speech and language processing*. Prentice-Hall, Upper Saddle River, N.J., USA, 2000.

[6] Namunu C. Maddage. Automatic structure detection for popular music. *IEEE Multimedia*, 13(1):65–77, January 2006.

[7] Elias Pampalk. A Matlab toolbox to compute music similarity from audio. In *Proc. of 5th International Conference on Music Information Retrieval*, Barcelona, Spain, October 2004.

[8] Jouni Paulus. *Signal Processing Methods for Drum Transcription and Music Structure Analysis*. PhD thesis, Tampere University of Technology, Tampere, Finland, December 2009.

[9] Jouni Paulus and Anssi Klapuri. Labelling the structural parts of a music piece with Markov models. In Sølvi Ystad, Richard Kronland-Martinet, and Kristoffer Jensen, editors, *Computer Music Modeling and Retrieval: Genesis of Meaning in Sound and Music - 5th International Symposium, CMMR 2008 Copenhagen, Denmark, May 19-23, 2008, Revised Papers*, volume 5493 of *Lecture Notes in Computer Science*, pages 166–176. Springer Berlin / Heidelberg, 2009.

[10] Jouni Paulus and Anssi Klapuri. Music structure analysis using a probabilistic fitness measure and a greedy search algorithm. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1159–1170, August 2009.

[11] Geoffroy Peeters and Emmanuel Deruty. Is music structure annotation multi-dimensional? A proposal for robust local music annotation. In *Proc. of 3rd Workshop on Learning the Semantics of Audio Signals*, pages 75–90, Graz, Austria, December 2009.

[12] Alan W. Pollack. 'Notes on...' series. The Official rec.music.beatles Home Page (http://www.recmusicbeatles.com), 1989–2001.

[13] Dana Ron, Yoram Singer, and Naftali Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2–3):117–149, 1996.

[14] Yu Shiu, Hong Jeong, and C.-C. Jay Kuo. Musical structure analysis using similarity matrix and dynamic programming. In *Proc. of SPIE Vol. 6015 - Multimedia Systems and Applications VIII*, pages 398–409, 2005.

INFINITE LATENT HARMONIC ALLOCATION: A NONPARAMETRIC BAYESIAN APPROACH TO MULTIPITCH ANALYSIS

Kazuyoshi Yoshii **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{k.yoshii, m.goto}@aist.go.jp

ABSTRACT

This paper presents a statistical method called *Infinite Latent Harmonic Allocation* (iLHA) for detecting multiple fundamental frequencies in polyphonic audio signals. Conventional methods face a crucial problem known as model selection because they assume that the observed spectra are superpositions of a *certain fixed* number of bases (sound sources and/or finer parts). iLHA avoids this problem by assuming that the observed spectra are superpositions of a *stochastically-distributed unbounded* (theoretically infinite) number of bases. Such uncertainty can be treated in a principled way by leveraging the state-of-the-art paradigm of machine-learning called Bayesian nonparametrics. To represent a set of time-sliced spectral strips, we formulated nested infinite Gaussian mixture models (GMMs) based on hierarchical and generalized Dirichlet processes. Each strip is allowed to contain an unbounded number of sound sources (GMMs), each of which is allowed to contain an unbounded number of harmonic partials (Gaussians). To train the nested infinite GMMs efficiently, we used a modern inference technique called collapsed variational Bayes (CVB). Our experiments using audio recordings of real piano and guitar performances showed that fully automated iLHA based on noninformative priors performed as well as optimally tuned conventional methods.

1. INTRODUCTION

Multipitch analysis of polyphonic audio signals [1–11] is one of the most important issues because it is the basis of many applications such as music transcription, chord recognition, and musical instrument recognition. We focus on principled methods based on machine learning, which have recently yielded promising results. Some researchers, for example, have proposed generative probabilistic models that explain how multiple spectral/signal bases (compositional units) are mixed to form polyphonic music [3–6]. The model parameters can be trained by means of statistical inference. Others have used nonnegative matrix factorization (NMF) to decompose polyphonic spectra into individual spectral bases [7–11]. NMF can be interpreted from the viewpoint of statistical inference [10–12].

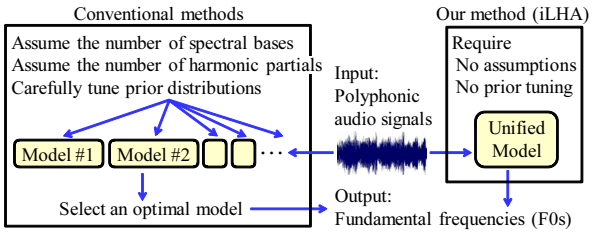


Figure 1. Methodological advantage of our method.

A crucial problem in these methods, known as model selection, is that they perform best only if an appropriate model complexity (the number of bases) is specified *in advance*. One might think that the optimal number of bases must be equal to the number of sound sources, but it is not clear how many bases are most suited to represent a single source if the spectral shape varies through time. Although *uncertainty* is inherent in model selection, conventional methods assume that a *certain* complexity exists uniquely as an oracle. As shown in Figure 1, they require possible models to be examined separately and exhaustively and the optimal model selected *in retrospect*. Such a deterministic framework is not easy-to-use in practice although optimally tuned methods can achieve good performance.

To avoid model selection, we propose a novel statistical method called *Infinite Latent Harmonic Allocation* (iLHA) based on a modern paradigm of machine learning called Bayesian nonparametrics. Note that the term "nonparametric" means that we do not have to fix model complexity uniquely. We assume that an unbounded but finite number of bases stochastically appears in a limited amount of available data although an infinite number of bases theoretically exists in the universe. Uncertainty in model selection can be treated reasonably in a probabilistic framework.

iLHA can be derived by taking the infinite limit of conventional finite models [3, 4]. Conventionally, each spectral basis is often parameterized by means of a Gaussian mixture model (GMM) in which a fixed number of Gaussians corresponds to the spectral peaks of harmonic partials, and a time-sliced polyphonic spectral strip is modeled by mixing a fixed number of GMMs. Here, we consider both the number of bases and the number of partials to approach infinity, where most are regarded as unnecessary and automatically removed through statistical inference.

A fundamental and practically-important advantage of iLHA is that precise prior knowledge is not required. Conventional methods [3–5] heavily rely on prior distributions regarding the relative strengths of harmonic partials, which

have too much impact on performance, and forced us to tune priors and their weighting factors by hand according to the properties of target sound sources. iLHA, in contrast, can be fully automated by layering noninformative hyperpriors on influential priors in a hierarchical Bayesian manner. This is consistent with the fact that humans can adaptively distinguish individual notes of various instruments. One of major contributions of this study is to embody the fundamental Bayesian principle "Let the data speak for itself" in the context of multipitch analysis.

The rest of this paper is organized as follows: Section 2 describes statistical interpretation of polyphonic spectra. Section 3 discusses related work. Sections 4 and 5 explain finite models (LHA) and infinite models (iLHA). Section 6 reports our experiments. Section 7 concludes this paper.

2. STATISTICAL INTERPRETATION

We interpret polyphonic spectra as histograms of observed frequencies that independently occur. This interpretation basically follows conventional studies [3–5].

2.1 Assumptions

Suppose given polyphonic audio signals are generated from K bases, each of which consists of M harmonic partials located on a linear frequency scale at integral multiples of the fundamental frequency (F0). Note that each *basis* can be associated with multiple *sounds* of different temporal positions if these sounds are derived from the same pitch of the same instrument. We transform the audio signals into wavelet spectra. Let D be the number of frames. If a spectral strip at frame d $(1 \leq d \leq D)$ has amplitude a at frequency f, we assume that frequency f was observed a times in frame d. Assuming that amplitudes are additive, we can consider each observed frequency to be generated from one of M partials in one of K bases.

These notations are for the finite case. In Bayesian nonparametrics, we take the limit as K and M go to infinity.

2.2 Observed and Latent Variables

Let the total observed variables over all D frames be represented by $\boldsymbol{X} = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_D\}$, where $\boldsymbol{X}_d$ is a set of observed frequencies $\boldsymbol{X}_d = \{\boldsymbol{x}_{d1}, \cdots, \boldsymbol{x}_{dN_d}\}$ in frame d. N_d is the number of frequency observations. That is, N_d is equal to the sum of spectral amplitudes over all frequency bins in frame d. $\boldsymbol{x}_{dn}$ $(1 \leq n \leq N_d)$ is an one-dimensional vector that represents an observed frequency.

Let the total latent variables corresponding to $\boldsymbol{X}$ be similarly represented by $\boldsymbol{Z} = \{\boldsymbol{Z}_1, \cdots, \boldsymbol{Z}_D\}$, where $\boldsymbol{Z}_d = \{\boldsymbol{z}_{d1}, \cdots, \boldsymbol{z}_{dN_d}\}$. $\boldsymbol{z}_{dn}$ is a KM-dimensional vector in which only one entry, z_{dnkm}, takes a value of 1 and the others take values of 0 when frequency $\boldsymbol{x}_{dn}$ is generated from partial m $(1 \leq m \leq M)$ of basis k $(1 \leq k \leq K)$.

3. COMPARISON WITH RELATED WORK

The properties of iLHA are intermediate between those of two successful approaches–statistical inference and NMF–which are discussed here for comparison and to clarify the positioning of our approach.

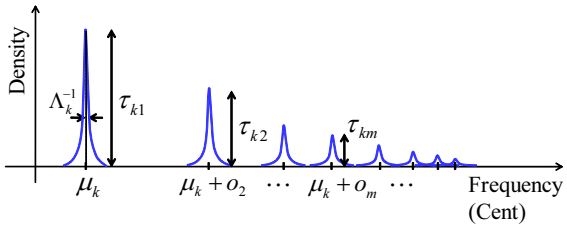


Figure 2. Probabilistic model of a single basis.

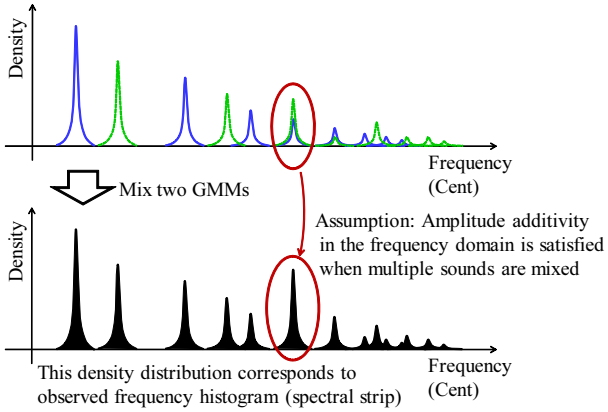


Figure 3. Probabilistic model of mixed multiple bases.

3.1 Statistical Inference

Statistical methods [3–6] assume probabilistic models using a limited number of parameters to represent the generative process of observed spectra (audio signals). F0 estimation directly corresponds to finding model parameters that provide the best explanations of the given data.

Goto [3] first proposed probabilistic models of harmonic sounds by regarding frequency spectra as probabilistic densities (histograms of observed frequencies).

As shown in Figure 2, the spectral distribution of basis k $(1 \leq k \leq K)$ is modeled by a harmonic GMM:

$$\mathcal{M}_k(\boldsymbol{x}) = \sum_{m=1}^{M} \tau_{km} \mathcal{N}\left(\boldsymbol{x} \middle| \boldsymbol{\mu}_k + \boldsymbol{o}_m, \boldsymbol{\Lambda}_k^{-1}\right), \qquad (1)$$

where $\boldsymbol{x}$ is a one-dimensional vector that indicates an observed frequency [cents]. [1] The Gaussian parameters, mean $\boldsymbol{\mu}_k$ and precision $\boldsymbol{\Lambda}_k$, indicate F0 [cents] of basis k and a degree of energy concentration around the F0 in the frequency domain. τ_{km} is a relative strength of the m-th harmonic partial $(1 \leq m \leq M)$ in basis k. We set $\boldsymbol{o}_m$ to $[1200 \log_2 m]$. This means M Gaussians are located to have harmonic relationships on the logarithmic frequency scale.

As shown in Figure 3, the spectral strip of frame d is modeled by mixing K harmonic GMMs as follows:

$$\mathcal{M}_d(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_{dk} \mathcal{M}_k(\boldsymbol{x}) \qquad (2)$$

where π_{dk} is a relative strength of basis k in frame d. Therefore, the polyphonic spectral strip is represented by nested finite Gaussian mixture models.

Several inference methods that have been proposed for parameter estimation are listed in Table 1. Goto [3] pro-

[1] Linear frequency f_h in hertz can be converted to logarithmic frequency f_c in cents as $f_c = 1200 \log_2(f_h/(440^{\frac{3}{12}-5}))$.

	#(bases)	#(partials)	Temporal modeling
PreFEst [3]	Fixed	Fixed	None
HC [4]	Inferred	Fixed	None
HTC [5]	Fixed	Fixed	Continuity treated
NMF [7]	Fixed	Not used	Exchangeable
iLHA	Infinite	Infinite	Exchangeable

Table 1. Comparison of multipitch analysis methods.

posed a method called PreFEst that estimates only relative strengths $\boldsymbol{\tau}$ and $\boldsymbol{\pi}$ while $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$ are fixed by allocating many GMMs to cover the entire frequency range as F0 candidates. Kameoka *et al.* [4] then proposed harmonic clustering (HC), which estimates all the parameters and selects the optimal number of bases by using the Akaike information criterion (AIC). Although these methods yielded the promising results, they analyze the spectral strips of different frames independently. Thus, Kameoka *et al.* [5] proposed harmonic-temporal-structured clustering (HTC) that captures temporal continuity of spectral bases. Note that all these methods are based on maximum-likelihood and maximum-a-posteriori training of the parameters by introducing prior distributions of relative strengths $\boldsymbol{\tau}$, which have a strong impact on the accuracy of F0 estimation.

Our method called iLHA is based on hierarchical nonparametric Bayesian modeling that requires no prior tuning and avoids specifying K and M in advance. More specifically, the limit of the conventional nested finite GMMs is considered as K and M diverge to infinity.

3.2 Nonnegative Matrix Factorization

NMF-based methods [7–12] factorize observed frequency spectra into the product of spectral bases and time-varying envelopes under the nonnegativity constraint. K bases are estimated by sweeping all frames of the given spectra. Although several methods [10, 12] take temporal continuity into account, standard methods are based on temporal exchangeability. In other words, exchange of arbitrary frames does not affect the factorized results. Although such temporal modeling is not sufficient, it is known to work well in practice. Therefore, iLHA adopted the exchangeability.

4. LATENT HARMONIC ALLOCATION

This section explains LHA, the finite version of iLHA, as a preliminary step to deriving iLHA. We formulate the conventional nested *finite* GMMs in a Bayesian manner.

4.1 Model Formulation

Figure 4 illustrates a graphical representation of the LHA model. The full joint distribution is given by

$$\begin{aligned} &p(\boldsymbol{X},\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda}) \\ &= p(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{\mu},\boldsymbol{\Lambda})p(\boldsymbol{Z}|\boldsymbol{\pi},\boldsymbol{\tau})p(\boldsymbol{\pi})p(\boldsymbol{\tau})p(\boldsymbol{\mu},\boldsymbol{\Lambda}) \end{aligned} \tag{3}$$

where the first two terms on the right-hand side are likelihood functions and the other three terms are prior distributions. The likelihood functions are defined as

$$p(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{\mu},\boldsymbol{\Lambda}) = \prod_{dnkm} \mathcal{N}\left(\boldsymbol{x}_{dn}\middle|\boldsymbol{\mu}_k + \boldsymbol{o}_m, \boldsymbol{\Lambda}_k^{-1}\right)^{z_{dnkm}} \tag{4}$$

$$p(\boldsymbol{Z}|\boldsymbol{\pi},\boldsymbol{\tau}) = \prod_{dnkm} (\pi_{dk}\tau_{km})^{z_{dnkm}} \tag{5}$$

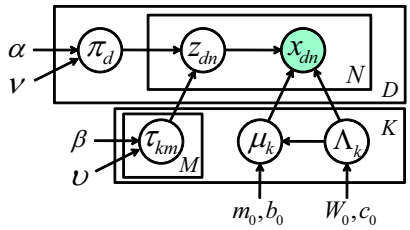


Figure 4. A graphical representation of LHA.

Then, we introduce conjugate priors as follows:

$$p(\boldsymbol{\pi}) = \prod_{d=1}^{D} \mathrm{Dir}(\boldsymbol{\pi}_d|\alpha\boldsymbol{\nu}) \propto \prod_{d=1}^{D}\prod_{k=1}^{K} \pi_{dk}^{\alpha\nu_k - 1} \tag{6}$$

$$p(\boldsymbol{\tau}) = \prod_{k=1}^{K} \mathrm{Dir}(\boldsymbol{\tau}_k|\beta\boldsymbol{\upsilon}) \propto \prod_{k=1}^{K}\prod_{m=1}^{M} \tau_{km}^{\beta\upsilon_m - 1} \tag{7}$$

$$p(\boldsymbol{\mu},\boldsymbol{\Lambda}) = \prod_{k=1}^{K} \mathcal{N}\left(\boldsymbol{\mu}_k\middle|\boldsymbol{m}_0,(b_0\boldsymbol{\Lambda}_k)^{-1}\right)\mathcal{W}\left(\boldsymbol{\Lambda}_k\middle|\boldsymbol{W}_0,c_0\right) \tag{8}$$

where $p(\boldsymbol{\pi})$ and $p(\boldsymbol{\tau})$ are products of Dirichlet distributions and $p(\boldsymbol{\mu},\boldsymbol{\Lambda})$ is a product of Gaussian-Wishart distributions. $\alpha\boldsymbol{\nu}$ and $\beta\boldsymbol{\upsilon}$ are hyperparameters and α and β are called concentration parameters when $\boldsymbol{\nu}$ and $\boldsymbol{\upsilon}$ sum to unity. $\boldsymbol{m}_0$, b_0, $\boldsymbol{W}_0$, and c_0 are also hyperparameters; $\boldsymbol{W}_0$ is a scale matrix and c_0 is a degree of freedom.

4.2 Variational Bayesian Inference

The objective of Bayesian inference is to compute a true posterior distribution of all variables: $p(\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda}|\boldsymbol{X})$. Because analytical calculation of the posterior distribution is intractable, we instead approximate it by using iterative inference techniques such as variational Bayes (VB) and Markov chain Monte Carlo (MCMC). Although MCMC is considered to be more accurate in general, we use VB because it converges much faster.

In the VB framework, we introduce a variational posterior distribution $q(\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda})$ and make it close to the true posterior $p(\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda}|\boldsymbol{X})$ iteratively. Here, we assume that the variational distribution can be factorized as

$$q(\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda}) = q(\boldsymbol{Z})q(\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda}) \tag{9}$$

To optimize $q(\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda})$, we use a variational version of the Expectation-Maximization (EM) algorithm [13]. We iterate VB-E and VB-M steps until a variational lower bound of evidence $p(\boldsymbol{X})$ converges as follows:

$$q^*(\boldsymbol{Z}) \propto \exp\left(\mathbb{E}_{\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda}}\left[\log p(\boldsymbol{X},\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda})\right]\right) \tag{10}$$

$$q^*(\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda}) \propto \exp\left(\mathbb{E}_{\boldsymbol{Z}}\left[\log p(\boldsymbol{X},\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda})\right]\right) \tag{11}$$

4.3 Updating Formula

We derive the formulas for updating variational posterior distributions according to Eqns. (10) and (11).

4.3.1 VB-E Step

An optimal variational posterior distribution of latent variables $\boldsymbol{Z}$ can be computed as follows:

$$\begin{aligned} &\log q^*(\boldsymbol{Z}) = \mathbb{E}_{\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda}}\left[\log p(\boldsymbol{X},\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\tau},\boldsymbol{\mu},\boldsymbol{\Lambda})\right] + \text{const.} \\ &= \mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\Lambda}}\left[\log p(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{\mu},\boldsymbol{\Lambda})\right] + \mathbb{E}_{\boldsymbol{\pi},\boldsymbol{\tau}}\left[\log p(\boldsymbol{Z}|\boldsymbol{\pi},\boldsymbol{\tau})\right] + \text{const.} \\ &= \sum_{dnkm} z_{dnkm}\log\rho_{dnkm} + \text{const.} \end{aligned} \tag{12}$$

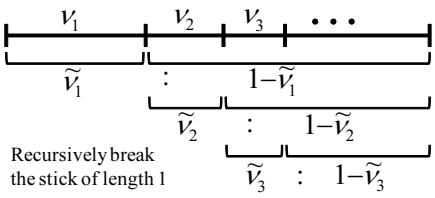


Figure 5. Stick-breaking construction of Dirichlet process.

where ρ_{dnkm} is defined as

$$\log \rho_{dnkm} = \mathbb{E}_{\boldsymbol{\pi}_d}[\log \pi_{dk}] + \mathbb{E}_{\boldsymbol{\tau}_k}[\log \tau_{km}] + \mathbb{E}_{\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k}\left[\log \mathcal{N}\left(\boldsymbol{x}_{dn}\middle|\boldsymbol{\mu}_k + \boldsymbol{o}_m, \boldsymbol{\Lambda}_k^{-1}\right)\right] \quad (13)$$

$q^*(\boldsymbol{Z})$ is obtained as multinomial distributions given by

$$q^*(\boldsymbol{Z}) = \prod_{dnkm} \gamma_{dnkm}^{z_{dnkm}} \quad (14)$$

where γ_{dnkm} is given by $\gamma_{dnkm} = \frac{\rho_{dnkm}}{\sum_{km}\rho_{dnkm}}$ and is called a responsibility that indicates how likely it is that observed frequency $\boldsymbol{x}_{dn}$ is generated from harmonic partial m of basis k. Here, let n_{dkm} be an observation count that indicates how many frequencies were generated from harmonic partial m of basis k in frame d. n_{dkm} and its expected value can be calculated as follows:

$$n_{dkm} = \sum_n z_{dnkm} \quad \mathbb{E}[n_{dkm}] = \sum_n \gamma_{dnkm} \quad (15)$$

For convenience in executing the VB-M step, we compute several sufficient statistics as follows:

$$\mathbb{S}_k[1] \equiv \sum_{dnm} \gamma_{dnkm} \quad \mathbb{S}_k[\boldsymbol{x}] \equiv \sum_{dnm} \gamma_{dnkm}\boldsymbol{x}_{dnm} \quad (16)$$

$$\mathbb{S}_k[\boldsymbol{x}\boldsymbol{x}^T] \equiv \sum_{dnm} \gamma_{dnkm}\boldsymbol{x}_{dnm}\boldsymbol{x}_{dnm}^T \quad (17)$$

where $\boldsymbol{x}_{dnm}$ is defined as $\boldsymbol{x}_{dnm} = \boldsymbol{x}_{dn} - \boldsymbol{o}_m$.

4.3.2 VB-M Step

Consequently, an optimal variational posterior distribution of parameters $\boldsymbol{\pi}, \boldsymbol{\tau}, \boldsymbol{\mu}, \boldsymbol{\Lambda}$ is shown to be given by

$$q^*(\boldsymbol{\pi}, \boldsymbol{\tau}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{d=1}^{D} q^*(\boldsymbol{\pi}_d) \prod_{k=1}^{K} q^*(\boldsymbol{\tau}_k) \prod_{k=1}^{K} q^*(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) \quad (18)$$

Since we use conjugate priors, each posterior has the same form of the corresponding prior as follows:

$$q^*(\boldsymbol{\pi}_d) = \mathrm{Dir}(\boldsymbol{\pi}_d|\boldsymbol{\alpha}_d) \quad (19)$$

$$q^*(\boldsymbol{\tau}_k) = \mathrm{Dir}(\boldsymbol{\tau}_k|\boldsymbol{\beta}_k) \quad (20)$$

$$q^*(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) = \mathcal{N}\left(\boldsymbol{\mu}_k\middle|\boldsymbol{m}_k, (b_k\boldsymbol{\Lambda}_k)^{-1}\right)\mathcal{W}\left(\boldsymbol{\Lambda}_k\middle|\boldsymbol{W}_k, c_k\right) \quad (21)$$

where the variational parameters are given by

$$\alpha_{dk} = \alpha\nu_k + \mathbb{E}[n_{dk\cdot}] \quad \beta_{km} = \beta\upsilon_m + \mathbb{E}[n_{\cdot km}] \quad (22)$$

$$b_k = b_0 + \mathbb{S}_k[1] \quad c_k = c_0 + \mathbb{S}_k[1] \quad (23)$$

$$\boldsymbol{m}_k = \frac{b_0\boldsymbol{m}_0 + \mathbb{S}_k[\boldsymbol{x}]}{b_0 + \mathbb{S}_k[1]} = \frac{b_0\boldsymbol{m}_0 + \mathbb{S}_k[\boldsymbol{x}]}{b_k} \quad (24)$$

$$\boldsymbol{W}_k^{-1} = \boldsymbol{W}_0^{-1} + b_0\boldsymbol{m}_0\boldsymbol{m}_0^T + \mathbb{S}_k[\boldsymbol{x}\boldsymbol{x}^T] - b_k\boldsymbol{m}_k\boldsymbol{m}_k^T \quad (25)$$

Here dot $(\cdot)$ denotes the sum over that index.

5. INFINITE LATENT HARMONIC ALLOCATION

This section derives hierarchical nonparametric Bayesian models, i.e., nested *infinite* GMMs for polyphonic spectra.

5.1 Model Formulation

First we let K approach infinity, where the infinite number of harmonic GMMs is assumed to exist in the universe. More specifically, the dimensionality of the Dirichlet distributions in Eqn. (6) is considered to be infinite. At each frame d, $\boldsymbol{\pi}_d$ is an infinite vector of normalized probabilities (mixing weights) drawn from the infinite-dimensional Dirichlet prior. Such stochastic process is called a Dirichlet process (DP). Every time frequency $\boldsymbol{x}_{dn}$ is generated, one of the infinite number of harmonic GMMs is drawn according to $\boldsymbol{\pi}_d$. Note that most entries of $\boldsymbol{\pi}_d$ take extremely tiny values because all entries sum to unity. If we can observe the infinite number of frequencies ($N_d \to \infty$), the infinite number of harmonic GMMs can be drawn. However, N_d is finite in practice. Therefore, only the finite number of harmonic GMMs, $K_+ \ll \infty$, is drawn at frame d. Here, a problem is that harmonic GMMs that are actually drawn at frame d are completely disjointed from those drawn at another frame d'. This is not a reasonable situation.

To solve this problem, we use the hierarchical Dirichlet Process (HDP) [14]. More specifically, we assume that infinite-dimensional hyperparameter $\boldsymbol{\nu}$ in Eqn. (6), which is shared among all D frames, is a draw from a top-level DP. A generative interpretation is that after an unbounded number of harmonic GMMs is initially drawn from the top-level DP, an unbounded subset is further drawn according to the local DP at each frame. This effectively ties frame d to another frame d'. As shown in Figure 5, $\boldsymbol{\nu}$ is known to follow the stick-breaking construction [14] as follows:

$$\nu_k = \tilde{\nu}_k \prod_{k'=1}^{k-1}(1 - \tilde{\nu}_{k'}) \quad \tilde{\nu}_k \sim \mathrm{Beta}(1, \gamma) \quad (26)$$

where γ is a concentration parameter of the top-level DP. Therefore $\boldsymbol{\nu}$ can be converted into $\tilde{\boldsymbol{\nu}}$.

Now we let M approach infinity, where each harmonic GMM consists of the infinite number of harmonic partials. To put effective priors on $\boldsymbol{\tau}$, we use generalized DPs called Beta two-parameter processes as follows:

$$\tau_{km} = \tilde{\tau}_{km} \prod_{m'=1}^{m-1}(1 - \tilde{\tau}_{km'}) \quad \tilde{\tau}_{km} \sim \mathrm{Beta}(\beta\lambda_1, \beta\lambda_2) \quad (27)$$

where β is a positive scalar and $\lambda_1 + \lambda_2 = 1$.

Because α, β, γ and $\boldsymbol{\lambda}$ are influential hyperparameters, we put Gamma and Beta hyperpriors on them as follows:

$$p(\alpha) = \mathrm{Gam}(\alpha|a_\alpha, b_\alpha) \quad p(\gamma) = \mathrm{Gam}(\gamma|a_\gamma, b_\gamma) \quad (28)$$

$$p(\beta) = \mathrm{Gam}(\beta|a_\beta, b_\beta) \quad p(\boldsymbol{\lambda}) = \mathrm{Beta}(\boldsymbol{\lambda}|u_1, u_2) \quad (29)$$

where $a_{\{\alpha,\beta,\gamma\}}$ and $b_{\{\alpha,\beta,\gamma\}}$ are shape and rate parameters.

Figure 6 shows a graphical representation of the iLHA model. The full joint distribution is given by

$$p(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\pi}, \tilde{\boldsymbol{\tau}}, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \alpha, \beta, \gamma, \boldsymbol{\lambda}, \tilde{\boldsymbol{\nu}}) = p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \\ p(\boldsymbol{Z}|\boldsymbol{\pi}, \tilde{\boldsymbol{\tau}})p(\boldsymbol{\pi}|\alpha, \tilde{\boldsymbol{\nu}})p(\tilde{\boldsymbol{\tau}}|\beta, \boldsymbol{\lambda})p(\alpha)p(\beta)p(\gamma)p(\boldsymbol{\lambda})p(\tilde{\boldsymbol{\nu}}|\gamma) \quad (30)$$

where $p(\boldsymbol{Z}|\boldsymbol{\pi}, \tilde{\boldsymbol{\tau}})$ is obtained by plugging Eqn. (27) into Eqn. (5) and $p(\boldsymbol{\pi}|\alpha, \boldsymbol{\nu})$ is the same as Eqn. (6). $p(\tilde{\boldsymbol{\nu}}|\gamma)$ and $p(\tilde{\boldsymbol{\tau}}|\beta, \boldsymbol{\lambda})$ are defined according to Eqns. (26) and (27) as

$$p(\tilde{\boldsymbol{\nu}}|\gamma) = \prod_k \mathrm{Beta}(\tilde{\tau}_k|1, \gamma) \; p(\tilde{\boldsymbol{\tau}}|\beta, \boldsymbol{\lambda}) = \prod_{km} \mathrm{Beta}(\tilde{\tau}_{km}|\beta\boldsymbol{\lambda}) \quad (31)$$

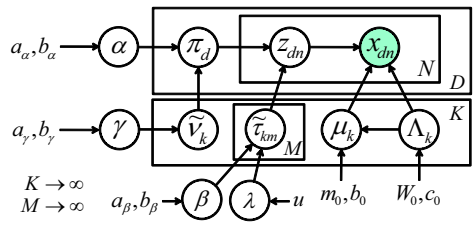


Figure 6. A graphical representation of iLHA.

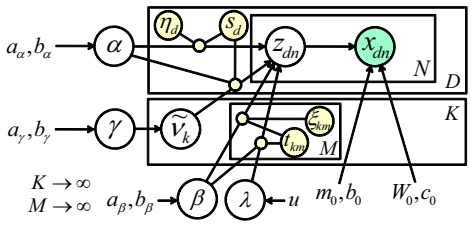


Figure 7. A collapsed model with auxiliary variables.

5.2 Collapsed Variational Bayesian Inference

To train the HDP model we use a sophisticated version of VB called collapsed variational Bayes (CVB) [15]. CVB enables more accurate posterior approximation in the space of latent variables where parameters are integrated out.

Figure 7 shows a collapsed iLHA model. By integrating out $\boldsymbol{\pi}, \tilde{\boldsymbol{\tau}}, \boldsymbol{\mu}, \boldsymbol{\Lambda}$, we obtain the marginal distribution given by

$$p(\boldsymbol{X}, \boldsymbol{Z}, \alpha, \beta, \gamma, \boldsymbol{\lambda}, \tilde{\boldsymbol{\nu}}) = p(\boldsymbol{X}|\boldsymbol{Z})p(\boldsymbol{Z}|\alpha, \beta, \boldsymbol{\lambda}, \tilde{\boldsymbol{\nu}})p(\alpha)p(\beta)p(\gamma)p(\boldsymbol{\lambda})p(\tilde{\boldsymbol{\nu}}|\gamma) \quad (32)$$

where the first two terms are calculated as follows:

$$p(\boldsymbol{X}|\boldsymbol{Z}) = (2\pi)^{-\frac{n_{\cdots}}{2}} \prod_k \left(\frac{b_0}{b_{zk}}\right)^{\frac{1}{2}} \frac{B(\boldsymbol{W}_0, c_0)}{B(\boldsymbol{W}_{zk}, c_{zk})} \quad (33)$$

$$p(\boldsymbol{Z}|\alpha, \beta, \boldsymbol{\lambda}, \tilde{\boldsymbol{\nu}}) = \prod_d \frac{\Gamma(\alpha)}{\Gamma(\alpha + n_{d\cdot\cdot})} \prod_k \frac{\Gamma(\alpha\nu_k + n_{dk\cdot})}{\Gamma(\alpha\nu_k)} \prod_{km} \frac{\Gamma(\beta)\Gamma(\beta\lambda_1 + n_{\cdot km})\Gamma(\beta\lambda_2 + n_{\cdot k>m})}{\Gamma(\beta\lambda_1)\Gamma(\beta\lambda_2)\Gamma(\beta + n_{\cdot k\geq m})} \quad (34)$$

where $b_{zk}, \boldsymbol{W}_{zk}, c_{zk}$ are obtained by substituting z_{dnkm} for γ_{dnkm} in calculating Eqns. (23) and (25).

Because CVB cannot be applied directly to Eqn. (32), we introduce auxiliary variables by using a technique called data augmentation [15]. Let η_d and ξ_{km} be Beta-distributed variables and s_{dk} and $\boldsymbol{t}_{km}$ be positive integers that satisfy $1 \leq s_{dk} \leq n_{dk\cdot}$, $1 \leq t_{km1} \leq n_{\cdot km}$, and $1 \leq t_{km2} \leq n_{\cdot k>m}$. Eqn. (34) can be augmented as

$$p(\boldsymbol{Z}, \boldsymbol{\eta}, \boldsymbol{\xi}, \boldsymbol{s}, \boldsymbol{t}|\alpha, \beta, \boldsymbol{\lambda}, \tilde{\boldsymbol{\nu}}) = \prod_d \frac{\eta_d^{\alpha-1}(1-\eta_d)^{n_{d\cdot\cdot}-1}}{\Gamma(n_{d\cdot\cdot})} \prod_k \begin{bmatrix} n_{dk\cdot} \\ s_{dk} \end{bmatrix} (\alpha\nu_k)^{s_{dk}} \prod_{km} \frac{\xi_{km}^{\beta-1}(1-\xi_{km})^{n_{\cdot k\geq m}-1}}{\Gamma(n_{\cdot k\geq m})} \begin{bmatrix} n_{\cdot km} \\ t_{km1} \end{bmatrix} (\beta\lambda_1)^{t_{km1}} \begin{bmatrix} n_{\cdot k>m} \\ t_{km2} \end{bmatrix} (\beta\lambda_2)^{t_{km2}}$$

where $[\ ]$ denotes a Stirling number of the first kind. The augmented marginal distribution is given by

$$p(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\eta}, \boldsymbol{\xi}, \boldsymbol{s}, \boldsymbol{t}, \alpha, \beta, \gamma, \boldsymbol{\lambda}, \tilde{\boldsymbol{\nu}}) = p(\boldsymbol{Z}, \boldsymbol{\eta}, \boldsymbol{\xi}, \boldsymbol{s}, \boldsymbol{t}|\alpha, \beta, \boldsymbol{\lambda}, \tilde{\boldsymbol{\nu}})p(\alpha)p(\beta)p(\gamma)p(\boldsymbol{\lambda})p(\tilde{\boldsymbol{\nu}}|\gamma) \quad (35)$$

In the CVB framework, we assume that the variational posterior distribution can be factorized as follows:

$$q(\boldsymbol{Z}, \boldsymbol{\eta}, \boldsymbol{\xi}, \boldsymbol{s}, \boldsymbol{t}, \alpha, \beta, \gamma, \boldsymbol{\lambda}, \tilde{\boldsymbol{\nu}}) = q(\alpha, \beta, \gamma, \boldsymbol{\lambda})q(\tilde{\boldsymbol{\nu}})q(\boldsymbol{\eta}, \boldsymbol{\xi}, \boldsymbol{s}, \boldsymbol{t}|\boldsymbol{Z}) \prod_{dn} q(\boldsymbol{z}_{dn}) \quad (36)$$

We also use an approximation technique called variational posterior truncation. More specifically, we assume $q(z_{dnkm}) = 0$ when $k > K$ and $m > M$. In practice, it is enough that K and M are set to sufficiently large integers.

5.3 Updating Formula

We descrive the formulas for updating variational posterior distributions.

5.3.1 CVB-E Step

A variational probability of $z_{dnkm} = 1$ is given by

$$\log q^*(z_{dnkm} = 1) = \mathbb{E}_{\boldsymbol{z}^{\neg dn}}\left[\log\left(\mathbb{G}[\alpha\nu_k] + n_{dk\cdot}^{\neg dn}\right)\right] + \mathbb{E}_{\boldsymbol{z}^{\neg dn}}\left[\log\left(\frac{\mathbb{G}[\beta\lambda_1] + n_{\cdot km}^{\neg dn}}{\mathbb{E}[\beta] + n_{\cdot k\geq m}^{\neg dn}} \prod_{m'=1}^{m-1} \frac{\mathbb{G}[\beta\lambda_2] + n_{\cdot k>m'}^{\neg dn}}{\mathbb{E}[\beta] + n_{\cdot k\geq m'}^{\neg dn}}\right)\right] + \mathbb{E}_{\boldsymbol{z}^{\neg dn}}\left[\log \mathcal{S}(\boldsymbol{x}_{dnm}|\boldsymbol{m}_{zk}^{\neg dn}, \boldsymbol{L}_{zk}^{\neg dn}, c_{zk}^{\neg dn})\right] + \text{const.} \quad (37)$$

where subscript $\neg dn$ denotes a set of indices without d and n, $\mathbb{G}[x]$ denotes the geometric average $\exp(\mathbb{E}[\log x])$, and $\mathcal{S}$ is the Student-t distribution. $\boldsymbol{L}_{zk}^{\neg dn}$ is given by $\boldsymbol{L}_{zk}^{\neg dn} = \frac{b_{zk}^{\neg dn}}{1+b_{zk}^{\neg dn}} c_{zk}^{\neg dn} \boldsymbol{W}_{zk}^{\neg dn}$, where $\boldsymbol{m}_{zk}^{\neg dn}, b_{zk}^{\neg dn}, \boldsymbol{W}_{zk}^{\neg dn}, c_{zk}^{\neg dn}$ are obtained by substituting z_{dnkm} for γ_{dnkm} required by Eqns. (23), (24), and (25) and calculating sum without $\boldsymbol{z}_{dn}$. Each term of Eqn. (37) can be calculated efficiently [15, 16].

5.3.2 CVB-M Step

First, α, β and γ are Gamma distributed as follows:

$$q(\alpha) \propto \alpha^{a_\alpha + \mathbb{E}[s_{\cdot\cdot}]-1} e^{-\alpha\left(b_\alpha - \sum_d \mathbb{E}[\log \eta_d]\right)} \quad (38)$$

$$q(\beta) \propto \beta^{a_\beta + \mathbb{E}[t_{\cdots}]-1} e^{-\beta\left(b_\beta - \sum_{km} \mathbb{E}[\log \xi_{km}]\right)} \quad (39)$$

$$q(\gamma) \propto \gamma^{a_\gamma + K - 1} e^{-\gamma\left(b_\gamma - \sum_k \mathbb{E}[\log(1-\tilde{\nu}_k)]\right)} \quad (40)$$

Then, $\boldsymbol{\lambda}$ and $\tilde{\boldsymbol{\tau}}$ are Beta distributed as follows:

$$q^*(\boldsymbol{\lambda}) \propto \lambda_1^{u_1 + \mathbb{E}[t_{\cdot\cdot 1}]-1} \lambda_2^{u_2 + \mathbb{E}[t_{\cdot\cdot 2}]-1} \quad (41)$$

$$q^*(\tilde{\nu}_k) \propto \tilde{\nu}_k^{1 + \mathbb{E}[s_{\cdot k}]-1} (1-\tilde{\nu}_k)^{\mathbb{E}[\gamma] + \mathbb{E}[s_{\cdot >k}]-1} \quad (42)$$

Finally, the variational posteriors of $\boldsymbol{\eta}, \boldsymbol{\xi}, \boldsymbol{s}, \boldsymbol{t}$ are given by

$$q^*(\eta_d) \propto \eta_d^{\mathbb{E}[\alpha]-1}(1-\eta_d)^{n_{d\cdot\cdot}-1} \quad (43)$$

$$q^*(\xi_{km}|\boldsymbol{Z}) \propto \xi_{km}^{\mathbb{E}[\beta]-1}(1-\xi_{km})^{n_{\cdot k\geq m}-1} \quad (44)$$

$$q^*(s_{dk} = s|\boldsymbol{Z}) \propto \begin{bmatrix} n_{dk\cdot} \\ s \end{bmatrix} \mathbb{G}[\alpha\nu_k]^s \quad (45)$$

$$q^*(t_{km1} = t|\boldsymbol{Z}) \propto \begin{bmatrix} n_{\cdot km} \\ t \end{bmatrix} \mathbb{G}[\beta\lambda_1]^t \quad (46)$$

$$q^*(t_{km2} = t|\boldsymbol{Z}) \propto \begin{bmatrix} n_{\cdot k>m} \\ t \end{bmatrix} \mathbb{G}[\beta\lambda_2]^t \quad (47)$$

To calculate $\mathbb{E}[s_{dk}]$ (average s_{dk} over $\boldsymbol{Z}$), we exactly treat the case $n_{dk\cdot} = 0$ and apply second-order approximation when $n_{dk\cdot} > 0$ (see details in [15]). $\mathbb{E}[\log \xi_{km}]$, $\mathbb{E}[t_{km1}]$, and $\mathbb{E}[t_{km2}]$ can be calculated in the same way.

To estimate F0s, we need explicitly compute the variational posteriors of the integrated-out parameters $\boldsymbol{\mu}, \boldsymbol{\Lambda}$. To do this, we execute the standard VB-M step once by using the responsibilities $q(\boldsymbol{Z})$ obtained in the CVB-E step.

6. EVALUATION

This section reports our comparative experiments evaluating the performance of iLHA.

Piece number	Optimally tuned		**Fully automated**	
RWC-MDB-	PreFEst [3]	HTC [5]	LHA	iLHA
J-2001 No.1	75.8	79.0	70.7	**82.2**
J-2001 No.2	**78.5**	78.0	69.1	77.9
J-2001 No.6	70.4	**78.3**	49.8	71.2
J-2001 No.7	83.0	**86.0**	70.2	85.5
J-2001 No.8	**85.7**	84.4	55.9	84.6
J-2001 No.9	85.9	**89.5**	68.9	84.7
C-2001 No.30	76.0	**83.6**	81.4	81.6
C-2001 No.35	72.8	76.0	58.9	**79.6**
Total	79.4	**82.0**	65.8	81.7

Table 2. Frame-level F-measures of F0 detection.

6.1 Experimental Conditions

We evaluated LHA and iLHA on the same test set used in [5], which consisted of nine pieces of piano and guitar solo performances excerpted from the RWC music database [17]. The first 23 [s] of each piece were used for evaluation. The spectral analysis was conducted by the wavelet transform using Gabor wavelets with a time resolution of 16 [ms]. The values and temporal positions of actual F0s were prepared by hand as ground truth. We evaluated performance in terms of frame-level F-measures. The priors and hyperpriors of LHA and iLHA were set to noninformative uniform distributions. K and M were set to sufficiently large numbers, 60 and 15. iLHA is not sensitive to these values. No other tuning was required. To output F0s at each frame, we extracted bases whose expected weights $\boldsymbol{\pi}$ were over a threshold, which was optimized as in [5].

For comparison, we referred to the experimental results of PreFEst and HTC reported in [5]. Although the ground-truth data was slightly different from ours, it would be sufficient for roughly evaluating performance comparatively. The number of bases, priors, and weighting factors were carefully tuned by using the ground-truth data to optimize the results. Although this is not realistic, the *upper bounds* of potential performance were investigated in [5].

6.2 Experimental Results

The results listed in Table 2 show that the performance of iLHA approached and sometimes surpassed that of HTC. This is consistent with the empirical findings of many studies on Bayesian nonparametrics that nonparametric models were competitive against optimally-tuned parametric models. HTC outperformed PreFEst because HTC can appropriately deal with temporal continuity of spectral bases. This implies that incorporating temporal modeling would improve the performance of iLHA.

The results of LHA were worse than those of iLHA because LHA is not based on hierarchical Bayesian modeling and requires precise priors. In fact, we confirmed that the results of PreFEst and HTC based on MAP estimation were drastically degraded when we used noninformative priors. In contrast, iLHA stably showed the good performance.

7. CONCLUSION

This paper presented a novel statistical method for detecting multiple F0s in polyphonic audio signals. The method allows polyphonic spectra to contain an unbounded number of spectral bases, each of which can consist of an unbounded number of harmonic partials. These numbers can be statistically inferred at the same time that F0s are estimated. Even in experimental evaluation using noninformative priors, our automated method performed well or better than conventional methods manually optimized by trial and error. To our knowledge, this is the first attempt to apply Bayesian nonparametrics to multipitch analysis.

Bayesian nonparametrics is an ultimate methodological framework avoiding the model selection problem faced in various areas of MIR. For example, how many sections should one use for structuring a musical piece? How many groups should one use for clustering listeners according to their tastes or musical pieces according to their contents? We are freed from these problems by assuming that in theory there is an infinite number of classes behind observed data. Unnecessary classes are automatically removed from consideration through statistical inference. We plan to use this powerful framework in a wide range of applications.

Acknowledgement: This study was partially supported by JST CREST and JSPS KAKENHI 20800084.

8. REFERENCES

[1] A. Klapuri. Multipitch analysis of polyphonic music and speech signals using an auditory model. *IEEE Trans. on ASLP*, Vol. 16, No. 2, pp. 255–266, 2008.

[2] M. Marolt. A connectionist approach to transcription of polyphonic piano music. *IEEE Trans. on Multimedia*, Vol. 6, No. 3, pp. 439–449, 2004.

[3] M. Goto. A real-time music scene description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, Vol. 43, No. 4, pp. 311–329, 2004.

[4] H. Kameoka *et al*. Separation of harmonic structures based on tied Gaussian mixture model and information criterion for concurrent sounds. *ICASSP*, Vol. 4, pp. 297–300, 2004.

[5] H. Kameoka *et al*. A multipitch analyzer based on harmonic temporal structured clustering. *IEEE Trans. on ASLP*, Vol. 15, No. 3, pp. 982–994, 2007.

[6] A. Cemgil *et al*. A generative model for music transcription. *IEEE Trans. on ASLP*, Vol. 14, No. 2, pp. 679–694, 2006.

[7] P. Smaragdis and J. Brown. Nonnegative matrix factorization for polyphonic music transcription. *WASPAA*, 2003.

[8] A. Cont. Realtime multiple pitch observation using sparse non-negative constraints. *ISMIR*, pp. 206–211, 2006.

[9] E. Vincent *et al*. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Trans. on ASLP*, Vol. 18, No. 3, pp. 528–537, 2010.

[10] N. Bertin *et al*. Enforcing harmonicity and smoothness in Bayesian non-negative matrix factorization applied to polyphonic music transcription. *IEEE Trans. on ASLP*, Vol. 18, No. 3, pp. 538–549, 2010.

[11] P. Peeling *et al*. Generative spectrogram factorization models for polyphonic piano transcription. *IEEE Trans. on ASLP*, Vol. 18, No. 3, pp. 519–527, 2010.

[12] T. Virtanen *et al*. Bayesian extensions to nonnegative matrix factorisation for audio signal modelling. *ICASSP*, 2008.

[13] H. Attias. A variational Bayesian framework for graphical models. *NIPS*, pp. 209–215, 2000.

[14] Y. W. Teh *et al*. Hierarchical Dirichlet processes. *J. of Am. Stat. Assoc.*, Vol. 101, No. 476, pp. 1566–1581, 2006.

[15] Y. W. Teh *et al*. Collapsed variational inference for HDP. *NIPS*, Vol. 20, 2008.

[16] J. Sung *et al*. Latent-space variational Bayes. *IEEE Trans. on PAMI*, Vol. 30, No. 12, pp. 2236–2242, 2008.

[17] M. Goto *et al*. RWC music database: Popular, classical, and jazz music database. *ISMIR*, pp. 287–288, 2002.

INFORMED SOURCE SEPARATION OF ORCHESTRA AND SOLOIST

Yushen Han
School of Informatics and Computing
Indiana University Bloomington
yushan@indiana.edu

Christopher Raphael
School of Informatics and Computing
Indiana University Bloomington
craphael@indiana.edu

ABSTRACT

A novel technique of unmasking to repair the degradation in sources separated by spectrogram masking is proposed. Our approach is based on explicit knowledge of the musical audio at note level from a score-audio alignment, which we termed Informed Source Separation (ISS). Such knowledge allows the spectrogram energy to be decomposed into note-based models. We assume that a spectrogram mask for the solo is obtained and focus on the problem of repairing audio resulting from applying the mask. We evaluate the spectrogram as well as the harmonic structure of the music. We either search for unmasked (orchestra) partials of the orchestra to be transposed onto a masked (solo) region or reshape a solo partial with phase and amplitude imputed from unmasked regions. We describe a Kalman smoothing technique to decouple the phase and amplitude of a musical partial that enables the modification to the spectrogram. Audio examples from a piano concerto are available for evaluation.

1. INTRODUCTION

We address the "desoloing" problem, in which we attempt to isolate the accompanying instruments in a monaural recording of music for soloist and orchestral accompaniment. The motivation is to produce the audio of the accompaniment part for concertos in the "classical" domain as well as the karaoke in popular music, whereas the ultimate goal is to have the orchestra adapt timing to the live player, a problem we do not discuss there. Nevertheless, the accompanying audio is needed and we offer solutions through our demixing or isolation of the original sources (instruments).

Most past effort in this "source separation" problem treats Blind Source Separation (BSS) problems and assumes little knowledge of the audio content rather than the independence of the sources [1] or relies on general cues of musical sources rather than the content of the sources [2]. In contrast, we assume explicit knowledge in the form of a score match, which establishes a correspondence between the audio data and a symbolic score representation giving the onset times of all musical events. See Figure 1 for an example. Such correspondence, known as "score following" or "alignment", initially introduced and developed by Vercoe and Dannenberg [12] is the foundation of our approach which we termed *Informed Source Separation* (ISS). Other examples in the category of ISS include Dubnov [6].

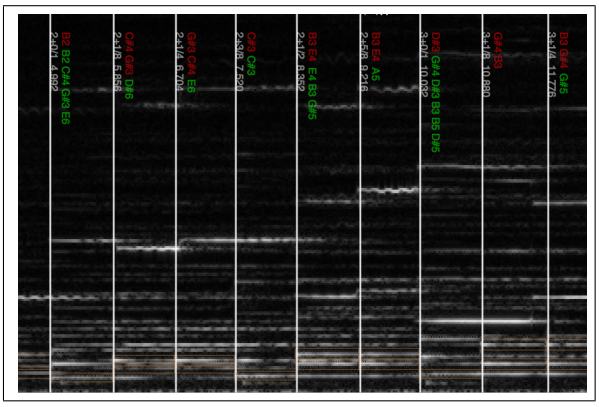


Figure 1. Piano note onsets (vertical lines) of an excerpt from 2nd mvmt. of Ravel's piano concerto in G

In our approach, we begin by masking the short time Fourier transform (STFT) in an attempt to "erase" the soloist's contribution. We also based our exploration toward partial-wise phase/amplitude relationship on previous work [7] in which spectrogram magnitude is decomposed into each partial by fitting note-based models. [11] is another effort of spectrogram decomposition in speech. However, our emphasis here is not on estimating the mask or fitting note models, but on employing a novel set of procedures (see sect. 5) that estimates and transforms note partials, in which the damage caused by our masking procedure is repaired.

Here our assumption is that there is information redundancy in terms of phase and amplitude between the "observable" partials (i.e. not significantly overlapped by the solo or an accompaniment instrument of a different family) and damaged partials. Our hope is to "copy and paste" musical partials from the observable area to the damaged area with some necessary transformations that exploit those redundancy to maintain the consistency between the observable and the damaged. These procedures can be automated by analyzing the texture of the music from the score and testing the soundness of remaining partials on the desoloed spectrogram. We call this process *unmasking* in which the masked-out solo regions will be recovered.

The structure of this paper is as follows: we briefly formulate the masking problem in sect. 2, followed by note-based parameterization in sect. 3 and phase estimation in sect. 4. Such estimating enables our repair-by-unmasking technique in sect. 5 which is applied in the context of a piano concerto in sect. 6.

2. SPECTROGRAM MASKING OF THE SOLO

Given our original audio signal, $x(s)$, we define the short time Fourier transform (STFT) by

$$X(t,k) = \sum_{n=0}^{N-1} x(tH+n)w(n)e^{-2\pi jkn/N}$$

where H is the hop size, N is the window length and w is the window function. We will define our masking operation in this STFT domain. To do so, we estimate two "complementary" masks, $1_s(t,k)$, and $1_a(t,k)$, taking values in $\{0,1\}$ with $1_s(t,k)+1_a(t,k)=1$. These masks are used to isolate the parts of X we attribute to the soloist and accompaniment through

$$X_s(t,k) = 1_s(t,k)X(t,k) \quad (1)$$

$$X_a(t,k) = 1_a(t,k)X(t,k) \quad (2)$$

In other words we label each time-frequency "cell" (t,k) as either solo or accompaniment. Since our focus here is on the *unmasking* problem, we will bias our labeling of each time-frequency cell toward the solo category, since we want to make sure the original soloist is completely removed. Using our score match, it would be relatively easy to simply draw a rectangle around each solo partial while calling the interior of these rectangles our solo mask. Our approach is somewhat more sophisticated, employing special treatment of the wide spectral dispersion associated with note onsets by Ono et al. [13], as well as careful modeling of the steady state partials. However, we will not discuss this mask estimation problem here.

While $X_a(t,k)$ (and $X_s(t,k)$) is, in general, not the STFT of *any* time signal, applying the inverse STFT operation gives perceptually sufficient results with appropriately defined STFT. In particular, if we use a Hann window with $H = N/4$, one can show that applying the STFT inverse to X_a results in the audio signal whose STFT is closest to X_a in the sense of Euclidean distance [5].

The result of this process elimanates more than the soloist, of course, since the accompanying instruments also contributed to the STFT in the region we have masked out. A possible remedy in sect. 4 is the main focus of our paper.

3. NOTE-BASED MUSIC PARAMETERIZATION

In this section we briefly review our parameterization of the music given the score, which is adapted from our technique to decompose the spectrogram magnitude into note models in [7]. This parameterization is also used to facilitate our phase estimation in sect. 4.

From the score, suppose we have a collection of notes $\mathcal{N}$ in the piece of interest, for a note $n \in \mathcal{N}$, we know its instrumentation $i_n \in \mathcal{I}$ where $\mathcal{I}$ is the set of instruments in this piece and can be further partitioned into disjoint subsets $\mathcal{I}_s$ and $\mathcal{I}_a$ for solo and accompaniment instruments separately.

Moreover, we know the time span of note n: $T_n = \{t_n^{on}, \ldots, t_n^{off}\}$ from the score following. Also, as the note pitch p_n indicates its set of valid harmonics under a certain Nyquist frequency: $\mathcal{H}_n = \{1, \ldots, H_n\}$, we confine the frequency bin span of each partial $h \in \mathcal{H}_n$ to $K_{n,h} = \{k_{n,h}^{low}, \ldots, k_{n,h}^{high}\}$. $K_{n,h}$ implements a band-pass filter to specify a frequency bin span where the contribution from the partial of interest (very likely to be mixed with other partials of close frequencies) is significant in terms of spectrogram magnitude while the spectral energy outside of $K_{n,h}$ is ignored.

Such 2-dimensional, rectangular time-bin support $B_{n,h} = \{(t,k)|t \in T_n, k \in K_{n,h}\}$ specifies a band-passed filter bank over T_n to extract time domain partial $p_h(s)$ from $X(t,k)$ We denote $B_n = B_{n,1} \cup \ldots \cup B_{n,H_n}$ to be the support for all harmonic components of note n.

We then assume a Normal mixture model for the spectrogram magnitude of an orchestra note n: each harmonic of the note is one Gaussian component in the mixture with normalized weight $\nu_{n,h}$, coupled frequency bin expectation $\mu_{n,h}(t) = h\mu_{n,1}(t)$, and unknown variance $\sigma_{n,h}^2$. To accommodate the (possibly dramatic) change in amplitude over time of a note, we also introduce a normalized non-negative profile, $\eta_{n,h}(t)$, to outline the frame-wise amplitude of hth partial of nth note.

Strictly, the centroid of each partial may not be precisely coupled by $\mu_{n,h}(t) = h\mu_{n,1}(t)$. But it is approximately true for all the instruments except for piano in our study. To summarize:

- a weight $\nu_{n,h} > 0$ for $\forall(n,h)$ with $\sum_{h\in\mathcal{H}_n} \nu_{n,h} = 1$
- a time support $T_n = \{t_n^{on}, \ldots, t_n^{off}\}$, which is shared among all partials of note n
- an amplitude envelope $\eta_{n,h}(t) > 0$ for $\forall(n,h)$ with $\sum_{h\in\mathcal{H}_n} \eta_{n,h}(t) = 1$
- a frequency bin support $K_{n,h} = \{k_{n,h}^{low}, \ldots, k_{n,h}^{high}\}$
- a frequency bin centroid $\mu_{n,h}(t)$ which reflected the frequency of partial h at t. Among different partials, they are coupled by $\mu_{n,1}(t) = \frac{\mu_{n,h}(t)}{h}$
- a frequency bin variance $\sigma_{n,h}$ that describes magnitude distribution of partial h over frequency bins with expectation $\mu_{n,h}(t)$ under Normal assumption.

Finally we can define a"template" function $q_{n,h}(t,k)$

$$= \begin{cases} 0, \ \forall(t,k) : t \notin T_n \ or \ \ k \notin K_{n,h} \\ \nu_{n,h}\eta_{n,h}(t)f(k;\mu_{n,h},\sigma_{n,h}^2); \ \ otherwise \end{cases} \quad (3)$$

where $f(k;\mu_{n,h},\sigma_{n,h}^2)$ is the normal density function. This parameterization is subjected to normalization to ensure $\sum_{h\in\mathcal{H}_n}\sum_{(t,k)\in B_{n,h}} q_{n,h}(t,k) = 1$ for note n.

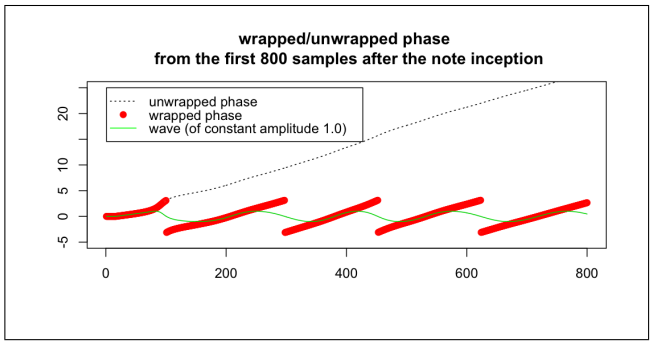


Figure 2. Wrapped and Unwrapped Phase

Our assumption is that the magnitude contribution from each note partial indexed by (n, h) to the spectrogram is raised from a collection of independent Poisson random variables $\{Z_n(t,k)\}$ for $(t,k) \in B_n$ [3]. The expectation of $Z_n(t,k)$ is $\delta_n \sum_h q_{n,h}(t,k)$ where δ_n describes the degree to which $Z_n(t,k)$ contributes to $X(t,k)$. Intuitively, δ_n is our estimate of the total spectrogram magnitude contribution from note n and can be interpreted as the overall "amplitude" of the note n. The estimation of δ_n is, no doubt, a significant factor of source separation quality and our solution by an EM algorithm is documented in [7]. For the rest of the paper, we assume a somewhat reliable δ_n is known so we can focus on the unknown phase of each partial of note n.

4. PARTIAL-WISE PHASE ESTIMATION AND TRANSFORMATIONS

As usually only a subset of partials of a note is damaged by removing the solo partial, we hope to exploited the harmonicity assumption in wind and string instruments supported by Fletcher [9] and Brown [10] to impute the phase of those missing partials in the orchestra. To do so, we first introduce a generic method to decouple the phase and slow-changing amplitude of a band-limited signal in 4.1 which enables our two major tools to "unmask" the damaged spectrogram: harmonic transposition in 4.2 and phase-locked modulation in 4.3.

4.1 Phase Estimation by Kalman Smoothing

In this section we represent our note partial, $p_h(s)$, in terms of a time-varying amplitude and phase:

$$p_h(s) \approx \alpha_h(s) \cos(\theta_h(s))$$

where the time-varying amplitude, $\alpha_h(s)$, is non-negative and varies slowly compared with $p_h(s)$, and the "unwrapped" phase (see Figure 2) function, $\theta_h(s)$, is monotonically non-decreasing. A more precise review of the slow-changing $\alpha_h(s)$ in a sinusoidal model is given by Rodet [14].

In order to estimate $\alpha_h(s)$ and $\theta_h(s)$ we follow the model of Taylan Cemgil [8] and view the harmonic, $p_h(s)$, as the ouput of a Kalman filter model [16] [17]. To this end we define a sequence of two-dimensional state vectors $\{x(s) = (x_1(s), x_2(s))^t\}$ where $x_1(0)$ and $x_2(0)$ are independent 0-mean random variables with variance γ^2, and the remaining variables follow evolution equation $x(s+1) = Ax(s) + w(s)$ where $\{w(s)\}$ is an independent sequence of 0-mean 2-dimensional vectors with independent components of fixed variance (the variance can be tuned empirically). A is the rotation matrix, defined in terms of the expected phase advance per sample, ρ, which is directly computable from the nominal frequency of the partial:

$$A = \begin{pmatrix} \cos\rho & \sin\rho \\ -\sin\rho & \cos\rho \end{pmatrix}$$

Thus, $x(s)$ is a sequence of vectors that circle around the origin and an approximately known frequency with variable distance from the origin. We then model our observed partial as $p_h(s) = x_1(s) + v(s)$ where $\{v(s)\}$ is another sequence of independent 0-mean variables with a certain variance (this variance is tuned empirically too).

It is well known that the Kalman filter allows straightforward computation of the conditional distribution, $p(x(s)|\{p_h(s')\})$, and that this distribution is Normal for each value of s. Thus we estimate $x(s)$ by $\hat{x}(s) = E(x(s)|\{p_h(s')\})$. The representation of the partial in terms of amplitude and non-decreasing phase follows from the polar coordinate representation of $\hat{x}(s)$:

$$\begin{aligned} \alpha_h(s) &= \sqrt{\hat{x}_1^2(s) + \hat{x}_2^2(s)} \\ \theta_h(s) &= 2\pi k(s) + \tan^{-1}(\frac{\hat{x}_2(s)}{\hat{x}_1(s)}) \end{aligned}$$

where each $k(s)$ is chosen to be the non-negative minimal integer value that ensures that $\theta_h(s)$ is non-decreasing.

Note that for phase sequence $\theta_h(s), s \in \{1, \ldots, S\}$, not only the final phase estimate $\hat{\theta}_h(S)$ but also all previous phase estimates are of interest. To get the "best" phase estimation, we need to update the state estimates backward to incorporate the observation that were not "available" at sample s in the forward pass. This motivates Kalman smoothing (see chapter 5 of [17]) which calculates the smoothed phase estimate $\hat{\theta}_h(s)$ recursively backward from the last sample at S.

4.2 Harmonic Transposition

With amplitude $\alpha_h(s)$ and phase $\theta_h(s)$ decoupled from hth harmonic of a note, we are ready to "project" one harmonic into a different harmonic while maintaining the harmonicity between the source and the destination. Supposing we estimated the unwrapped phase of the ith harmonic as $\theta_i(s)$, the "projected" phase sequence at jth harmonic is given by $\tilde{\theta}_j(s) = \frac{j\theta_i(s)}{i}$ and the resulting jth harmonic by

$$\tilde{p}_j(s) = \tilde{\alpha}_j(s) \cos(\frac{j\theta_i(s)}{i}) \quad (4)$$

where $\tilde{\alpha}_j(s)$ is either known or imputed amplitude at jth harmonic. In this work, we usually have an estimate of $\tilde{\alpha}_j(s)$ by scaling δ_n from sect. 3.

Our harmonic transposition exploit such "harmonicity" between partials, which is a well-studied phenomenon. Early

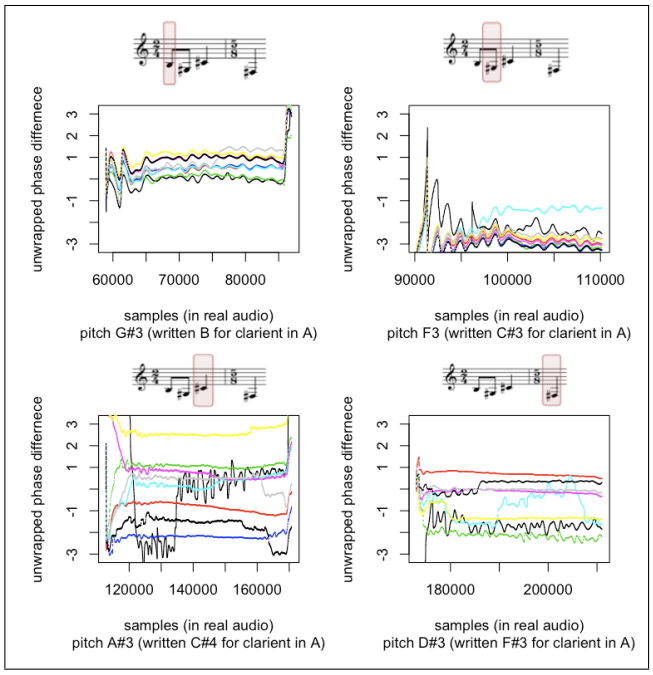


Figure 3. Unwrapped Phase Difference

work mainly by Fletcher showed that frequencies of the partials in"the middle portion of the tone" of string instrument are integral multiples of the fundamental frequency by using sonograph and also derived that partials of string and wind instrument are "rigorously locked into harmonic relationship" [9]. By using single frame approximation on a variety of digital samples, Brown concluded that "continuously driven instruments such as the bowed strings, winds, and voice have phase-locked frequency components with frequencies in the ratio of integers to within the currently achievable measurement accuracy of about 0.2%" [10] from experiments with and without vibrato.

To demonstrate such harmonicity in our framework, we focus on the "projection" of the unwrapped phase $\theta_i(s)$ from partial i to partial j by

$$\theta_{i,j}(s) = \frac{j\theta_{h_1}(s)}{i} \tag{5}$$

By "projecting" the phase of different partials to a common harmonic, we can examine such phase relation on a variety of orchestra instruments. We can visualize *pairwise phase difference* $\theta_{i,1}(s) - \theta_{j,1}(s)$ at the fundamental for any $i \neq j$. Figure 3 shows the *pairwise phase difference* for the first 4 notes from a performance of the first movement of Stravinsky's Three Pieces for Clarinet Solo. The salient message from this plot is: the *pairwise phase difference* is in a very small range (mostly $(-\frac{\pi}{2}, \frac{\pi}{2})$) and never drifts away over the entire note; the error (including measurement error and true difference) is not accumulative. This supports our approximation of phase coherence.

Piano and other impulsively driven instruments such as strings played pizzicato are counter-examples whose partials deviate from integer ratios due to the stiffness of the string [10].

4.3 Phase-locked Modulation

In addition to the partial-wise relationship, we want to exploit timewise similarity in terms of phase and amplitude within one note.

Suppose we have a partition $T_1 = \{s_1, \ldots, s_k - 1\}$, $T_2 = \{s_k, \ldots, s_2\}$ on the sample indices $T = \{s_1, \ldots, s_2\}$ of the sustaining part of a reasonably long orchestra note, we can only observe the unwrapped phase sequence at $\theta_h(T_1)$ but $\theta_h(T_2)$ is missing. We can impute $\theta_h(T_2)$ sequentially by

$$\theta_h(s_k+n) = \theta_h(s_k+n-1)+\theta_h(s_1+1+n)-\theta_h(s_1+n) \tag{6}$$

for any $0 \leq n \leq s_2 - s_k$. We omit the formula to obtain $\theta_h(T_1)$ if we observe $\theta_h(T_2)$.

This operation preserves the phase advance per sample in T_1 and applies such $\Delta\theta_h(T_1)$ cyclically to T_2. This is similar to the phase vocoder except for that we are doing it on the sample level rather than frame level. For a long enough time span T_1, we are capturing the pattern of frequency fluctuation in $\theta_h(T_1)$. To synthesize a segment of a partial, we also need the amplitude envelope over T_2. A simple solution is to reuse the average amplitude α_h over T_1 (with some minor modulation) to "sustain" a note through the end of T_2. If the orchestra note is holding for quite long, which is common in some orchestration, we are effectively synthesizing the sustaining part of the partial.

5. SPECTROGRAM UNMASKING

In an attempt to fix the damage caused by desolo, we examine the spectrogram with a focus on areas where the accompaniment notes (harmonics) are damaged.

In the type of music that we (and many solo musicians) are mainly interested in, for instance, a piano concerto, it is common that a string section may double the solo instrument at the unison, fifth, or octave in either direction. In these cases, masking out the solo part usually results in many damaged partials in the orchestra since consonant intervals mean more partials are likely to share the same frequencies. With this in mind, we use some heuristics to create an algorithm to automatically perform the two partial-wise transformations developed in 4.2 and 4.3. Since the texture of the music can be highly complex, we reconstruct a somewhat "generic" scenario for illustration of this algorithm in Figure 4. The 1-bar score in the figure is a reduction from a piano concerto where the piano part is frequently doubled by the lower string sections.

Supposing we have obtained solo mask $1_s(t,k)$, a damaged region $B^d_{n,h} \subseteq B_{n,h}$, a template $g_{n,h}(t,k)$ and an amplitude estimate δ_n from section 2 and 3 for a damaged partial h of note n, we summarize our *heuristic* algorithm:

First, we need to evaluate the damage. If $\sum_{(t,k)\in B^d_{n,h}} g_{n,h}(t,k) \ll \sum_{(t,k)\in B_{n,h}} g_{n,h}(t,k)$, we leave it as intact; otherwise we need to repair it. Specially, if undamaged part $B_{n,h} \setminus B^d_{n,h}$ is a narrow band-limited "strip" (e.g. a single frequency bin), we need to "expand" the solo mask to remove those initially deemed

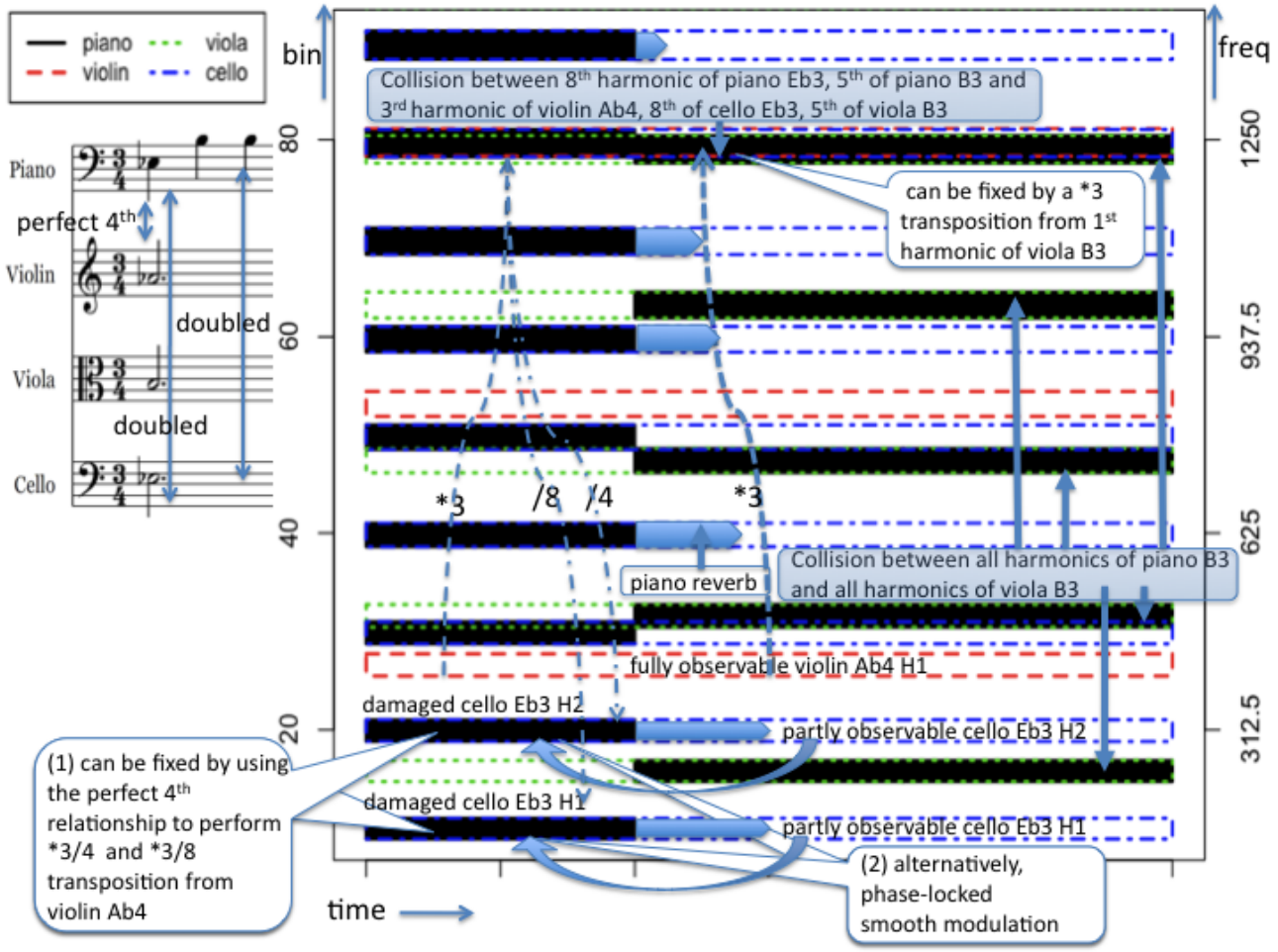


Figure 4. Evaluating Desolo Damage and Possible Fix Using Both Score and Spectrogram

"undamaged" f-t cells as well because such residue tends to create artifact "musical noise" whose suppression deserves treatment, mostly from speech enhancement. After such extra "masking", we use $B^u_{n,h} \subseteq B_{n,h}$ to denote the remaining undamaged region.

Second, since $B_{n_1,h_1} \cap B_{n_2,h_2} \neq \emptyset, n_1 \neq n_2$ for possibly many different note partials contributing energy to the same region, we choose one damaged orchestra partial (n,h) to repair: $\underset{(n,h)}{\operatorname{argmax}} \sum_{(t,k)\in B_{n,h}} \delta_n g_{n,h}(t,k)$ assuming Max-Approximation that only one signal dominates in each time-frequency cell [3].

Third, in the score we look for consonant intervals such as octaves, perfect 5th and perfect 4th in the hope to find an observable partial whose frequency is in a relatively simple ratio to the damaged one waiting to be "transposed" to. We call this partial, if exists, a *candidate*. Usually more than one candidate exist. Large modulus value, simple frequency ratio and identical instrumentation are factors that we favor in choosing the best candidate without creating artifacts. Thus, harmonic transposition can be performed vertically on the spectrogram (e.g. from 3rd to 5th harmonic of viola note B3 in Figure 4) if the duration of the candidate partial covers that of the damaged area.

Fourth, when there is no candidate partial for the partial indexed by (n,h), if there exists a partial (m,i) whose time support of its undamaged portion $T^u_{m,i}$ is adjacent to the damaged duration $T^d_{n,h}$ and whose frequency bin support $K_{m,i}$ satisfies $K^d_{n,h} \subseteq K_{m,i}$ we can perform phase-locked modulation with differenced phase sequence estimated from $B^u_{m,i}$ to $B^d_{n,h}$. The 2 cello partials in Figure 4 are repaired this way.

Occasionally, we are unable to perform either transformation and label the damaged partial as such.

6. EXPERIMENT RESULTS

We experiment with an excerpt of 45 seconds from the 2nd movement of Ravel's piano concerto in G major.

Table 1 lists a breakdown of the number of partials [1] and the number of harmonic transpositions and phase-locked modulation that our algorithm performed. The last column, "unable to fix" gives the number of occurrences that no undamaged orchestra partial is available to estimate phase from. We relax on that the 4 sections of string instruments can be used to repair each other by harmonic transposition but do not allow any harmonic transposition between two different instruments in the woodwind family. This is be-

[1] the number of partials only include partials that have significant spectral energy and are below Nyquist frequency at SR=8000Hz.

	note	partial	tran. from	tran. to	modulation	unable to repair
oboe	20	85	1	1	0	1
clarinet	6	18	3	3	0	0
flute	6	18	0	0	0	0
violin1	5	42	14	9	0	0
violin2	11	107	34	24	24	2
viola	16	160	33	41	64	5
cello	12	120	43	50	22	6

Table 1. Instrument breakdown of partials being repaired

cause the oboe is sharper than the other two in this excerpt. At the end the most of damaged partials are fixed in some way. We also notice that the woodwinds are less damaged because the notes are very high pitched and too loud to yield to the solo piano at their time-frequency region, while the lower string instruments are frequently damaged.

The original, desoloed-but-unrepaired and repaired audio are available at our demo website `http://xavier.informatics.indiana.edu/~yushan/ISMIR2010` to evaluate the solo mask and improvement from unmasking. Plots in color giving a breakdown of the partials on the spectrogram are also available.

7. CONCLUSION, EVALUATION AND FUTURE WORK

Instead of merely extracting one source (instrument) of sound from the mixture, we distinguish our proposed ISS method from other known source separation methods by our explicit *repair* stage that addresses the audio degradation caused by the separation procedure. This stage significantly enhances the perceptual audio quality and boosts performance measurement such as distortion due to interferences proposed by Vincent. That the reconstructed note sounds plausible for some orchestra instruments suggests that the partial-wise phase/amplitude relationship is a potentially fruitful topic to investigate.

At this stage, we admit that the comparison of our method of "unmasking" with other missing data inference techniques such as [15] is not available and hence is our future work. An ideal evaluation of any method of solo/orchestra separation requires a "ground truth" of the two sources recorded separately and an artificial mix of the two. However, such "ground truth" is almost away absent in the real case and the evaluation is mainly subjective. Our exploration begins with a music sample library to artificially construct ground truth according to the score while maintaining the texture of the music of interests.

8. REFERENCES

[1] Bell, A. J., and Sejnowski, T. J.: "An Information-Maximization Approach to Blind Separation and Blind Deconvolutionm *Neural Computation*, vol. 7, no. 6, pp. 1129 - 1159, 1995.

[2] E. Vincent: "Musical Source Separation Using Time-Frequency Source Priors," *IEEE Trans. on Speech and Audio Processing* , Vol. 14, Iss. 1, Jan. 2006 pp. 91 - 98.

[3] D. Ellis: Chap. 4 of *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, Wiley/IEEE Press, pp.115-146, 2006.

[4] A. S. Bregman: *Auditory scene analysis*. MIT Press: Cambridge, MA, 1990.

[5] Francis R. Bach and Michael I. Jordan: "Blind one-microphone speech separation: A spectral learning approach.", *NIPS*, pages 6572, 2005.

[6] S. Dubnov: "Optimal filtering of an instrument sound in a mixed recording using harmonic model and score alignment," *ICMC* 2004, Miami.

[7] Y. Han, C. Raphael: "Desoloing Monaural Audio Using Mixture Models," *ISMIR*, Vienna, 2007

[8] A. T. Cemgil, S. J. Godsill: "Probabilistic Phase Vocoder and its application to Interpolation of Missing Values in Audio Signals." Antalya/Turkey, 2005. EURASIP.

[9] H. Fletcher: "Mode locking in nonlinearly excited inharmonic musical oscillators," Vol. 64, pp. 1566-1569 *J. Acoust. Soc. Am.*, 1978.

[10] Judith C. Brown: "Frequency ratios of spectral components of musical sounds," *J. Acoust. Soc. Am*. vol. 99, no. 2, pp. 1210-1218, February 1996.

[11] B. Raj and P. Smaragdis: "Latent Variable Decomposition of Spectrogram for Single Channel Speaker Separation," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* , pp. 17-20, Oct. 2005.

[12] R. Dannenberg and C.Raphael: "Music Score Alignment and Computer Accompaniment," *Communications of the ACM, 49(8) (August 2006), pp. 38-43.*

[13] N. Ono, K. Miyamoto, J. Le Roux, H. Kameoka, and S. Sagayama: "Separation of a Monaural Audio Signal into Harmonic/Percussive Components by Complementary Diffusion on Spectrogram," *EUSIPCO.*, 2008

[14] Xavier Rodet: "Musical Sound Signal Analysis/Synthesis: Sinusoidal+Residual and Elementary Waveform Models," *IEEE Time-Frequency and Time-Scale Workshop 97*, Coventry, Grande Bretagne, 1997

[15] J Bouvrie and T Ezzat.: "An incremental algorithm for signal reconstruction from short-time fourier transform magnitude." *9th Intl. Conf. on Spoken Language Processing,* 2006.

[16] R. E. Kalman: "A New Approach to Linear Filtering and Prediction Problems," *Transaction of the ASME - Journal of Basic Engineering*, 35-45. March 1960.

[17] R.L. Eubank: *A Kalman Filter Primer*, Chapman & Hall/CRC, 2005.

IS THERE A RELATION BETWEEN THE SYNTAX AND THE FITNESS OF AN AUDIO FEATURE?

Gabriele Barbieri
Dip. di Matematica
Università di Bologna
gbarbieri@dm.unibo.it

François Pachet
Sony CSL
Paris
pachet@csl.sony.fr

Mirko Degli Esposti
Dip. di Matematica
Università di Bologna
desposti@dm.unibo.it

Pierre Roy
Sony CSL
Paris
roy@csl.sony.fr

ABSTRACT

Feature generation has been proposed recently to generate feature sets automatically, as opposed to human-designed feature sets. This technique has shown promising results in many areas of supervised classification, in particular in the audio domain. However, feature generation is usually performed blindly, with genetic algorithms. As a result search performance is poor, thereby limiting its practical use. We propose a method to increase the search performance of feature generation systems. We focus on *analytical features*, i.e. features determined by their syntax. Our method consists in first extracting statistical properties of the feature space called *spin patterns*, by analogy with statistical physics. We show that spin patterns carry information about the topology of the feature space. We exploit these spin patterns to guide a simulated annealing algorithm specifically designed for feature generation. We evaluate our approach on three audio classification problems, and show that it increases performance by an order of magnitude. More generally this work is a first step in using tools from statistical physics for the supervised classification of complex audio signals.

1. INTRODUCTION

The identification of feature sets is a fundamental step in solving supervised classification problems [3]. For problems involving complex signals (e.g. music, images, etc.) the traditional approach is to use "off-the-shelf" features (see, e.g. [9]). However, general-purpose features are not always adapted to solve difficult classification tasks. Another solution is to design manually *ad hoc* features, specific to the problem at hand. Such a task can be conducted only by experts knowledgeable both in the domain (e.g. music) and in signal processing, a difficult, costly and time-consuming task. Moreover, there is no guarantee that humans will find the best possible features.

Feature generation has recently been introduced to address this problem, by generating automatically problem-dependent features, designed to be efficient for any particular supervised classification problem [18]. Feature generation consists in building features by searching in a huge feature space, usually through genetic programming techniques [7]. The *fitness* of a feature is defined as the performance of a classifier using this feature on the problem at hand [8]. These previous works have consistently demonstrated that feature generation outperforms traditional approaches based on feature selection (see e.g. [5, 14]). However, most, if not all, feature generation systems proposed in the literature were shown to necessitate the exploration of a large number of features before finding relevant ones. For instance in [17], in the context of classification of percussive sounds, the feature generation system evaluated about 77500 features to eventually find features which outperformed standard ones.

Although *search performance* is usually not an issue for off-line applications, poor search performance forbids the use of these promising techniques in other contexts. As an example, the multiplication of portable entertainment devices creates a need for application requiring fast classification of *a priori* unknown user data (audio, pictures, gestures, etc.). In these contexts, feature generation cannot be used primarily because of performance issues. We propose a search method that reduces substantially the number of features to actually evaluate.

The main source of inefficiency of feature generation comes from the blind search strategy inherent to genetic programming. Most of the computation time is lost in evaluating irrelevant features, as noted by [5]. This problem worsens in the signal domain, where some features can be costly to evaluate (for instance features involving the computation of complex transforms). In this context, search can take several days.

Search performance can be increased by guiding the search using domain specific heuristics. To find heuristics we have to understand the mathematical structure of the feature space, to estimate a priori what regions are likely to contain relevant features. However, this is impossible to do in general, because we do not have information about the semantics of the features [11]. For this reason we restrict our study to the specific case of *analytical features* [14]. Analytical features (AF) are functional compositions of elementary signal processing operators. An AF is defined by its syntactical form, i.e. a tree of basic operators. A central question of our study, reflected in the title of this paper, is

therefore how to exploit this syntax to extract information about a features fitness before actually computing it. As we will see, this relation is complex.

In this paper, we first show that predicting directly the fitness from the syntax is difficult. We propose to model features from a more fine-grained perspective: borrowing techniques from spin glass theory [10] we introduce the notion of *spin patterns* to model partial statistical information about basic operators. We show that spin patterns contain probabilistic information about the fitness of features that use a given operator. We also show how these patterns can be used to predict feature fitness. We then propose a feature generation algorithm guided by these predictions. Our algorithm can be viewed as a variant of the *simulated annealing* [4]. The comparison between simulated annealing and genetic programming is a well studied topic [2], however it is hard to establish what is the more efficient optimization method in the general case [19]. In our context we use simulated annealing because it is easier to guide by the information obtained from the spin patterns.

Two versions of the algorithm are proposed. The basic version searches for individual features, and the extended version searches for feature sets. Even if it is well known that individual features are not able to solve difficult classification problems [3], we present the basic version because it well describes the theoretical aspects of the algorithm. We evaluate our algorithms on three audio classification problems. We show that our algorithms find features and feature sets which are as good or better than features found by a standard feature generation algorithm, but with a significantly improved search performance (an order of magnitude).

This paper is structured as follows: In Section 2, we introduce analytical features and syntactic neighborhood. In Section 3, we study the prediction of feature fitness from their syntax. In Section 4, we introduce the notion of spin pattern for operators. We illustrate these patterns on a simple audio classification problem. In Section 5, we introduce our search algorithms. In Section 6, we describe the performance of our algorithm on 3 audio classification problems, and compare it to a standard feature generation algorithm.

2. ANALYTICAL FEATURES

Analytical Features are expressed as a functional term, taking as only argument the input signal (represented here as x). This functional term is composed of basic operators. Given a library of basic operators $\mathcal{L}$, an **analytical feature** f is an application $f = O_1 \circ \ldots \circ O_N$ such that $O_i \in \mathcal{L}$. $\mathcal{S}(f) = \{O_1, \ldots, O_N\}$ is the **syntactical form** of f. The **length** $l(f)$ is the number of operators making up the feature. $\mathcal{F}$ is 0the set of all possible analytical features built on $\mathcal{L}$.

For instance, the following feature (A) computes the MFCC (*Mel Frequency Cepstrum Coefficients*) of successive frames of length 100 (samples) with no (0) overlap, and then computes the variance of this value vector:

$$A = Variance(MFCC(Split(x, 100, 0))).$$

The **neighborhood** of f is the set $V_f = \{g \in \mathcal{F} | d_{Syn}(f, g) \leq 1\}$, where $d_{Syn}(f, g)$ is the Levenshtein distance.

Problem	$I(d_{Fit}, d_{Syn})$
PAN	0.032
INS	0.015
MG	0.015

Table 1. Estimation of the mutual information $I(d_{Fit}, d_{Syn})$ between the distances d_{Fit} and d_{Syn} evaluated on three audio classification problem.

2.1 Feature Fitness

Given a classification problem, the fitness $\lambda_D(f)$ of an AF measures the capacity of the feature to distinguish elements of different classes.

There are several ways to assess the fitness of a feature. We follow here a wrapper approach [6], by which features are evaluated using a classifier built on-the-fly during the feature search process [14]. The fitness of the feature is defined as the performance of a classifier built with this unique feature.

We use Support Vector Machines. To evaluate the performance of the classifier (or more precisely its average F-measure) we use 10-fold cross validation on the training database.

3. PREDICTING FEATURE FITNESS

We define the distance $d_{Fit}(f, g)$ based on the fitness of f and g:

$$d_{Fit}(f, g) = |\lambda_D(f) - \lambda_D(g)|.$$

We study here experimentally the relationship between d_{Syn} and d_{Fit} on concrete problems. In this study we consider three audio classification problems. The problem PAN consists in discriminating between six percussive sounds [15], INS consists in discriminating between sounds played by eight different instruments [12] and MG consists in discriminating between six musical genres [13].

We compute a population P of 1000 features randomly generated from $\mathcal{F}$. For each problem and for each couple (f, g) in P, we compute distances $d_{Fit}(f, g)$, $d_{Syn}(f, g)$. Note that d_{Fit} depends on D_i, whereas d_{Syn} is problem-independent.

We then estimate the mutual information $I(d_{Fit}, d_{Syn})$ [1] between the distances. Table 1 shows that in each case the mutual information is smaller than 0.1: if a relation exists between syntax and fitness, it is somehow hidden and difficult to model.

4. SPIN PATTERNS

In this section we introduce a representation of analytical features taking into account the contribution of each sam-

ple. Let D be a labeled data set, composed of N audio samples divided in k classes, $C_1, \ldots, C_k$:

$$D = \{(x_1, l_1), \ldots, (x_N, l_N)\},$$

where x_i is the i-th audio sample and $l_i \in \mathcal{C} = \{C_j\}_{j=1,\ldots,k}$ is its label.
Given a feature $f \in \mathcal{F}$, we define its spin configuration as:

$$f \to \sigma^f = \begin{cases} +1 \text{ if } f \text{ classifies } x_i \text{ correctly} \\ -1 \text{ otherwise} \end{cases}$$

If $C : \mathbb{R} \to \mathcal{C} = \{C_j\}_{j=1,\ldots,k}$ is a classifier trained on D with f as a single feature, we have:

$$\sigma_i^f = \begin{cases} +1 \text{ if } C(f(x_i)) = l_i \\ -1 \text{ otherwise} \end{cases}$$

Given $f \in \mathcal{F}$, $\lambda(f)$ is the fitness of f on D, $0 \leq \lambda(f) \leq 1$. By analogy with the spin glass model [10], we can interpret fitness as the Hamiltonian $H(\sigma^f)$ of a spin configuration σ^f induced by the feature f:

$$H(\sigma^f) = -\lambda(f).$$

Given a basic operator $o \in \mathcal{L}$, where $\mathcal{L}$ is the operator library used to construct $\mathcal{F}$, we define

$$\mathcal{F}_o = \{f \in \mathcal{F} | o \in \mathcal{S}(f), l(f) \geq 10\}.$$

So $\mathcal{F}_o$ is the set of all features that contain o with length smaller than 10. Considering only these features we have $0 < |\mathcal{F}_o| < \infty$.
The **spin pattern** of a basic operator o is a vector $m(o) = \{m_i(o)\}_{i=1,\ldots,N}$ such that:

$$m_i(o) = \left\langle \sigma_i^f \right\rangle_{\mathcal{F}_o} = \frac{1}{|\mathcal{F}_o|} \sum_{f \in \mathcal{F}_o} \sigma_i^f$$

In practice $m_i(o)$ is an indicator of the probability that a feature containing o correctly classifies sample x_i. Indeed, it is easy to show that, given $f \in \mathcal{F}_o$,

$$Pr(\sigma_i^f = +1) = \frac{1 + m_i(o)}{2}$$

$$Pr(\sigma_i^f = -1) = \frac{1 - m_i(o)}{2}$$

Therefore we are interested in operators whose pattern has as many values as possible which are close to 1. Conversely, *zero spin patterns* configuration ($m_i(o) = 0 \ \forall j$) do not provide any information about the set of features considered. In order to measure the amount of information given by a spin pattern, we can compute its *total magnetization*, defined as follows:

$$M(o) = \frac{\sum_{i=1}^{N} m_i(o)}{N}$$

Figure 1 shows a graphical representation of the spin patterns of two operators. The samples are arranged in a picture composed by N squares. $|m_i(o)|$ is mapped to the darkness of the corresponding square. In this representation, high magnetization features correspond to dark images. In the figure we can observe that the spin pattern of a specific operator like $LpFilter$ has more magnetization than the spin pattern of a more general operator like Abs (the Abs pattern is clearly lighter). The main intuition in guiding search toward an optimal path in the feature space is that there is a rich, non uniform distribution of spin patterns for all basic operators. To support this claim we compute the spin patterns of each operator on the three classification problems presented above. In general, operators yield a spin pattern whose magnetization is significantly higher than 0 (the magnetization of the zero spin pattern, taken as a reference): as we see in figure 2, for each problem, the distributions of the magnetizations for all $o \in \mathcal{L}$ are concentrated near 0.5. This study shows that interesting patterns do exist. Of course we do not know the spin patterns a priori for a given classification problem. However, we have seen experimentally that we can estimate these patterns using a relatively small number of computations (≈ 1000).

5. THE ALGORITHM

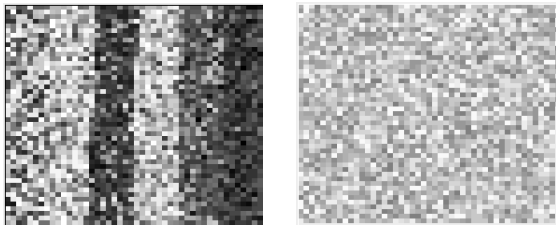
Our algorithm takes as input a labeled database D and a library of basic operators $\mathcal{L}$. The algorithm searches the feature space $\mathcal{F}$ defined by $\mathcal{L}$, guided by spin patterns, as defined in section 4.

5.1 Individual feature search (IFS)

We describe here the IFS algorithm, that searches for individual features. Section 5.2 describes an extension to feature set search. IFS receives as input a labeled database D and a library of basic operators $\mathcal{L}$. The output is a feature with a high fitness on domain D.
The algorithm is a variant of the simulated annealing algorithm. It is based on a Metropolis procedure [4] that guarantees the convergence to a global optimum.
Starting from a random feature f_0, the algorithm iteratively selects neighbors of the current feature, using the spin patterns. At each iteration, the algorithm selects a new feature in the syntactical neighborhood of the current feature. This choice is done according to the estimation of the spin patterns. The algorithm terminates after a specified number of iterations, or as a result of an interactive user request, and returns the best feature (i.e. the feature with highest fitness) found during the search as output.
In the following subsections we detail the components of the algorithm.

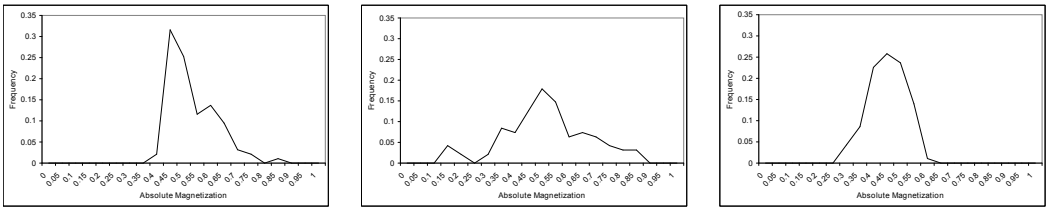
5.1.1 The Spin Pattern Estimator

The *spin pattern estimator* (SPE) is executed only once at the beginning of the algorithm.
The SPE computes a population T of 1000 random features. These features are used to estimate the spin patterns of each operator $o \in \mathcal{L}$.



(a) LpFilter (b) Abs

Figure 1. Graphical representation of the spin patterns of the basic operators $Lpfilter$ (left) and Abs (right) evaluated on the problem PAN (classification of percussive sounds). The representation of a domain specific operator like $LpFilter$ is clearly darker. In the spin pattern of $LpFilter$ note the two dark stripes on the center and on the right. Magnetization are 0.46 and 0.28 (resp.)



(a) INS (b) MG (c) PAN

Figure 2. Distributions of the absolute magnetizations of the spin patterns of all the operators $o \in \mathcal{L}$

5.1.2 The Neighbor Selector

The task of the *neighbor selector* is to decide how to move in the feature space. The selector receives as input the current feature and the spin pattern estimation of each operator $o \in \mathcal{L}$. The output of the selector is either a new feature, or the feature passed in input.

Given the input feature f and the estimations of the spin patterns, the selector assigns a score to each basic operator $o \in \mathcal{L}$. This score is designed to favor operators that tend to correctly classify the samples that are wrongly classified by f. The score $\mu(o)$ of operator o is defined as:

$$\mu(o) = \frac{\sum_{i=1}^{N} b(\sigma_i^f)\bar{m}_i(o)}{N} \tag{1}$$

where $b(x) = -\frac{1}{2}x + \frac{1}{2}$ is a function that converts a ± 1 spin in a $\{0, 1\}$ boolean value ($b(+1) = 0$, $b(-1) = 1$). $\mu(o)$ induces a score function $\mu(g)$ for a feature $g \in \mathcal{F}$ by averaging on all operators of f:

$$\mu(g) = \frac{\sum_{o \in \mathcal{S}(g)} \mu(o)}{l(g)} \tag{2}$$

A crucial point of our method is that in general the score function $\mu(g)$ is easier to compute than the fitness $\lambda(g)$.

The selector then computes the neighborhood V_f of f (see section 3). V_f is then sorted according to the score function defined in (2). The feature $\tilde{f} \in V_f$ with highest score is chosen by the neighbor selector. The spin configuration $\sigma^{\tilde{f}}$ and fitness $\lambda(\tilde{f})$ of $\tilde{f}$ are then computed.

To avoid local maximum effects, $\tilde{f}$ is accepted as the next feature in a stochastic way, using the Metropolis procedure.

More precisely, if $\Delta\lambda = \lambda(\tilde{f}) - \lambda(f) \geq 0$ f is accepted. The case $\Delta\lambda < 0$ is accepted with a probability $Pr(\Delta\lambda) = e^{\frac{\Delta\lambda}{t_k}}$, where

$$t_n = \left(\frac{t_1}{t_0}\right)^n t_0$$

and $k = |T|$.

If the feature $\tilde{f}$ is not accepted, it is removed from V_f, otherwise it will be reselected by the neighbor selector in the next iteration. Using the Metropolis procedure, we can assume that, for a good choice of parameters T_0 and T_1, the algorithm converges to the global maximum [4]. Following [4], we heuristically set $T_1/T_0 = 0.95$ and $T_0 = 10$.

5.2 The Feature Set Version

As described earlier, a single feature is usually not enough to solve a classification problem and a feature set is required. In principle, building feature sets instead of individual features complexifies drastically the procedure, since all combinations of features must be considered at each step. For this reason we propose the follow simple extension of IFS that searches feature sets, in order to maintain an affordable computational cost.

The architecture of the algorithm is essentially the same of the basic version. The *spin pattern estimator* works exactly in the same way. The only module that changes is the *neighbor selector*. In this version it takes as input a feature set of dimension d (d is a fixed parameter) and outputs a feature set instead of a single feature.

Because the spin configuration of a feature f is defined by the classifier built with the values of f, it is possible to define in the same way the spin configuration σ^F and the fitness $\lambda(F)$ of a feature set F, using the classifier produced

by the values of F. Therefore we can define the score of an operator $\mu(o)$ and the score of a feature $\mu(g)$ as in the previous subsection.
In this version of the algorithm, a feature f is randomly chosen $\in F$. A feature $\tilde{f}$ in the neighborhood V_f is selected like in the basic version of the algorithm.
The new feature set $\tilde{F}$ is then built from F by substituting f by $\tilde{f}$. The new feature set is accepted with the same Metropolis procedure described above. Our algorithm is then able to search for feature sets of any dimension.
In the next section we compare the search performance of our two algorithms against the search performance of a standard genetic algorithm.

6. RESULTS

6.1 Individual Feature Search

To assess the search performance of IFS we compare it against the search performance of a standard genetic algorithm on three audio classification problems. These problems are the same presented in the section 3. The genetic algorithm we use is described in [14].
Given a classification problem D, we use our algorithm and the genetic algorithm to search the feature f with highest fitness $\lambda_D(f)$ and we compare the sets of features explored by the two algorithms. We are interested in three quantities: the fitness of the best feature found, how many features are computed before finding the best feature and the distribution of the fitness of the explored features.
To get statistically significant results, we execute both algorithms three times for each classification problem. The results are shown in figure 3 and figure 4. In figure 3 we can observe that IFS finds features at least as good as the features found by the genetic algorithm but converges faster to the solution. In average, it computes less than 3500 features to find the best features. This figure includes the number of features needed to estimate the spin patterns (approximately 1000, as described in section 4). Conversely the genetic algorithm requires more than 48000 features to find the optimal.
As we can see in figure 4 the distribution of the fitness, when using IFS, is concentrated near high values of the fitness. Conversely the genetic algorithm explores blindly the feature space, resulting in a more uniform distribution of the fitness.

6.2 Feature Set Search

The feature set version of our algorithm builds feature sets of dimension N. Again we test this algorithm against the genetic algorithm used above to find feature sets of dimension 3 for the three reference problems.
The genetic algorithm, after a population has been created and each feature has been individually evaluated, selects a subset of features to be retained for the next population. The output of the genetic algorithm is a feature set F_{GA} of dimension 3: F_{GA} is the set $\{f_1, f_2, f_3 | f_1, f_2, f_3 \in F_{last}\}$ with maximum fitness. F_{last} is the last population.
Here we compare the two feature sets obtained by the two algorithms against an other feature set obtained by applying a feature selection algorithm (in our case, InfoGain [16]) to the whole set of features explored by the genetic algorithm. In table 2 we observe that our algorithm performs as well as the genetic one and the feature selection one on the three problems. However, the features sets necessitate a smaller exploration: table 3 shows the number of features explored in order to find the best feature set. It can be seen again that our algorithm improves the search performance by an order of magnitude.

Database	GP	FS	IFS
PAN	0.76	0.78	0.79
INS	0.56	0.56	0.59
MG	0.77	0.79	0.77

Table 2. Comparison between the fitness of the best feature sets obtained by IFS against the best feature sets obtained by the genetic algorithm. In two cases IFS outperforms the genetic one. GP means genetic programming, FS means feature selection.

Database	GP and FS	IFS
PAN	77531	4043
INS	72837	4152
MG	43265	7221

Table 3. Number of features needed to find the best feature sets.

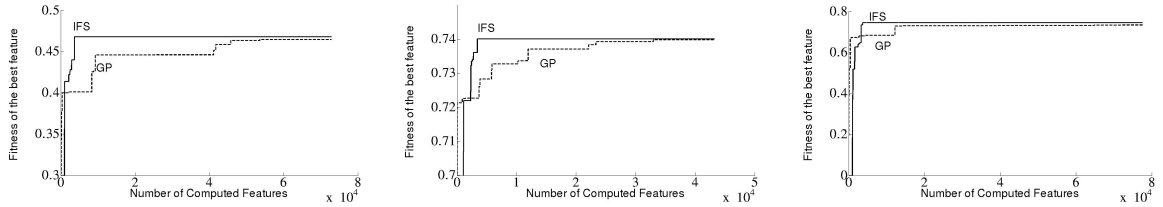
7. CONCLUSION

We have explored the relation between the syntax and the fitness in a supervised classification context. Such a relation seems complex to grasp at a macroscopic level. We have proposed a sample-based approach to model the topology of feature spaces, and exhibited a computable criterion, spin patterns, to guide a feature search algorithm. This algorithm is based on simulated annealing, with a Metropolis procedure, and exploits spin patterns, resulting in a better performance than genetic programming, as measured by the total number of features actually evaluated.
As such, this approach is a promising one to reduce the traditionally high computational cost of feature generation, and increase the applications of this technique. The answer to our title question is therefore: "Yes, there is a complex relationship". Furthermore, we showed how this relationship can be exploited to improve the performance of a feature generation system. More generally, this work represents a first step in applying tools from statistical mechanics of complex systems to supervised classification of complex audio signals.

8. REFERENCES

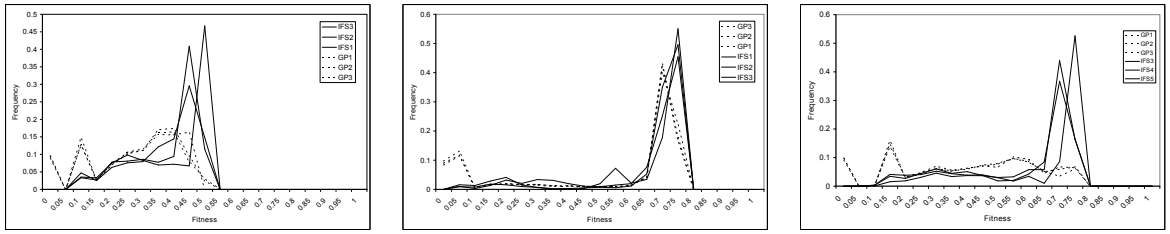
[1] T.M. Cover, J.A. Thomas, *Elements of information theory.* Wiley, New York, 1991.

[2] L. Davis, *Genetic algorithms and Simulated Anneal-*



(a) INS (b) MG (c) PAN

Figure 3. Performance of IFS compared to the performance of a standard genetic algorithm (GP). We report here only the best execution for each algorithm.



(a) INS (b) MG (c) PAN

Figure 4. Distributions of the fitness of the features computed by the two algorithms. In dotted line, the distributions of the features computed by the genetic algorithm, in black the features computed by IFS. For IFS, the distributions of the fitness are concentrated near high values, whereas the features explored by the genetic algorithm have more spread distributions.

ing. Morgan Kaufman Publishers, Inc.,Los Altos, CA. 1987.

[3] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection", *Journal of Machine Learning Research,* 3:11571182, 2003.

[4] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, New Series, Vol. 220, No. 4598, pp. 671-680, May 13, 1983.

[5] Y. Kobayashi, "Automatic Generation of Musical Instrument Detector by Using Evolutionary Learning Method", in *Proc. of the 10th International Conference on Music Information Retrieval (ISMIR '09)*, Kobe, Japan, October 2009.

[6] R. Kohavi, G.H. John, "The Wrapper Approach", in *Feature Selection for Knowledge Discovery and Data Mining,* H. Liu & H. Motoda (eds.), Kluwer Academic Publishers, pp 33-50, 1998.

[7] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press. 1992.

[8] S. Markovitch and D. Rosenstein, "Feature Generation Using General Constructor Functions", *Machine Learning*, Vol. 49, pp. 59-98, 2002.

[9] M.F. McKinney, J. and Breebart, "Features for audio and music classification", in *Proc. of the International Conference on Music Information Retrieval (ISMIR 2003)*, pp. 151-158, 2003

[10] M. Mezard, G. Parisi, M.A. Virasoro, *Spin Glasses Theory and Beyond*, World Scientific, Singapore, 1987.

[11] I. Mierswa, and K. Morik, "Automatic feature extraction for classifying audio data", *Machine Learning Journal*, Vol. 58, pp. 127-149, 2005.

[12] University of Iowa Musical Instrument Samples: http://theremin.music.uiowa.edu/MIS.html

[13] http://ismir2004.ismir.net/genre_contest/index.htm

[14] F. Pachet, P. Roy, "Analytical Features: A Knowledge-Based Approach to Audio Feature Generation", *Eurasip Journal on Audio, Speech, and Music Processing*, 2009(1), February 2009.

[15] http://www.csl.sony.fr/pandeiro/

[16] J. R. Quinlan,*C4.5: Programs for Machine Learning,* Morgan Kaufmann, San Francisco, California, USA, 1993.

[17] P. Roy, F. Pachet, S. Krakowski, "Analytical Features for the classification of percussive sound: the case of Pandeiro", in *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR '07)*, pp. 229-232, Vienna, Austria, September 2007.

[18] E. K. Tang ; P. N. Suganthan; X. Yao, "Nonlinear feature extraction using evolutionary algorithm" in *Lecture notes in computer science,* Vol. 3316, pp. 1014-1019, 2004.

[19] A. Vasan, K.S. Raju, "Comparative analysis of Simulated Annealing, Simulated Quenching and Genetic Algorithms for optimal reservoir operation," in *Applied Soft Computing,* Vol. 9, No. 1, pp. 274-281, January 2009.

ISLANDS OF GAUSSIANS: THE SELF ORGANIZING MAP AND GAUSSIAN MUSIC SIMILARITY FEATURES

Dominik Schnitzer[1,2], **Arthur Flexer**[1], **Gerhard Widmer**[1,2], **Martin Gasser**[1]
[1]Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria
[2]Department of Computational Perception, Johannes Kepler University, Linz, Austria
dominik.schnitzer@ofai.at, arthur.flexer@ofai.at,
gerhard.widmer@jku.at, martin.gasser@ofai.at

ABSTRACT

Multivariate Gaussians are of special interest in the MIR field of automatic music recommendation. They are used as the de facto standard representation of music timbre to compute music similarity. However, standard algorithms for clustering and visualization are usually not designed to handle Gaussian distributions and their attached metrics (e.g. the Kullback-Leibler divergence). Hence to use these features the algorithms generally handle them indirectly by first mapping them to a vector space, for example by deriving a feature vector representation from a similarity matrix.

This paper uses the symmetrized Kullback-Leibler centroid of Gaussians to show how to avoid the vectorization detour for the Self Organizing Maps (SOM) data visualization algorithm. We propose an approach so that the algorithm can directly and naturally work on Gaussian music similarity features to compute maps of music collections. We show that by using our approach we can create SOMs which (1) better preserve the original similarity topology and (2) are far less complex to compute, as the often costly vectorization step is eliminated.

1. INTRODUCTION

Good content-based music recommendation systems are currently on the wish list of many music services since they help handling the massive audio databases which are currently emerging: be it simple music recommendations in the form of automatically generated playlists, advanced visualizations of the collections, or other new ideas for music discovery and listening.

One of the basic foundations of automatic content-based music recommendation systems is the ability to compute music similarity. In research it is not yet settled how to extract and represent good music similarity features that correspond well with the human perception of general music similarity. However, the currently best performing methods have one thing in common: A central component in most of the currently best working methods is a representation of timbre in terms of a multivariate Gaussian. Take for example, the top three algorithms in the MIREX [1] 2009 (Music Information Retrieval Exchange [4]) Automatic Music Recommendation evaluations. All used multivariate Gaussians and Kullback-Leibler-related divergences to describe and compare their similarity features. The basic idea of using a single multivariate Gaussian to model timbre similarity was first used by Mandel and Ellis [10]. In their case the Gaussian is computed over the Mel Frequency Cepstrum Coefficient representation [8] of the song.

We see the Gaussian representation of the music features as a very powerful way to describe the variability of the features in a song. With the Kullback-Leibler divergence and related divergence measures (Symmetrized Kullback-Leibler, Jensen-Shannon divergence) there also exist well founded ways to compute a distance/similarity value between the features (even though there are some problems related to the non-metricity of the divergences).

Things get interesting when we leave the path of simple feature representation and similarity computation. Standard algorithms for indexing, clustering or visualization are usually just not designed to work with Gaussian distributions and non-standard metrics like the one the Kullback-Leibler divergence induces. For instance, how would computing a simple average be performed with Gaussian features?

To work around that limitation the feature data is often artificially vectorized. In the domain of content based music recommendation techniques we have seen approaches computing the full distance matrix and using each row of the matrix as a feature vector [5, 14], or more venturesome ones reshaping the Gaussian covariance matrix and mean vector into a single long vector [11]. The first solution is expensive to compute the larger the music collection is, and the latter one, although fast, takes away the sense of using Gaussians.

In 2005 Banerjee et al. published an important paper in the machine learning literature where they show how to generalize the k-means clustering algorithm to the broad class of Bregman divergences [1]. This generalization practically opened all centroid-based algorithms to the

[1] http://www.music-ir.org/mirex/2009

wide range of Bregman divergences, which the Kullback-Leibler divergence is part of.

This paper builds on these findings and defines the weighted symmetrized Kullback-Leibler centroid to show how the Self Organizing Map (SOM) algorithm can work directly and naturally with Gaussians. The approach is able to create higher-quality two dimensional visualizations of music archives while retaining the nice scalability characteristics of the general SOM algorithm.

2. RELATED WORK

There already exists a wide range of publications dealing with visualizing acoustic music similarity features on SOMs. One of the first to do so were Rauber and Frühwirth [18] who use a very basic music similarity feature and a simple tabular grid which displays the clustered song titles on the map. This idea was extended by Pampalk et al. who use rhythmic similarity features and the Smoothed Data Histogram [16] (SDH) visualization to draw the SOM. Their visualization is inspired by geographical maps: blue regions (oceans) indicate areas onto which very few pieces of music are mapped, whereas clusters containing a larger quantity of pieces are colored in brown and white (mountains and snow). It was published under the name "Islands of Music" [15], which inspired the title of the presented paper.

'Neptune' [5] developed by Knees et al. improved Pampalk's visualization by taking the two dimensional map into the third dimension. They add crawled meta-information and pictures from the web and allow a 3D walk through a music collection. They use a mix of rhythmic and timbre based similarity measures.

The 'Globe of Music' [7] by Leitich and Topf uses a GeoSOM [20] to map the music collection onto a globe for exploration. Lübbers et al. developed the 'SoniXplorer' [9] to navigate through music archives. They use a multimodal navigation model where the auralization of music supports the user on the SOM visualization of the music collection.

Mörchen et al. use the Emergent SOM algorithm to visualize and cluster music collections in their 'Music Miner' [12] system. For music similarity they use a large set of low-level features. In their paper they also point out that they cannot use Gaussian music similarity features as "they can not be used with datamining algorithms requiring the calculation of a centroid". Solving that is the focus of the next sections of this paper.

3. PRELIMINARIES

To demonstrate how to use the SOM algorithm with Gaussian features we use the standard music similarity algorithm proposed by Mandel and Ellis [10]. This approach computes a single Gaussian music-timbre similarity feature. Similarity is computed with the symmetrized Kullback-Leibler divergence.

Since its publication this approach has been modified and improved in various ways, yet most stayed with the Gaussian feature representation. So everything presented here will of course work with the derived approaches too.

3.1 Music Features and Similarity Computation

To extract the timbre music similarity features we extract 25 Mel Cepstrum Frequency Coefficients [8] (MFCCs) for every $46ms$ of audio. This corresponds to a window size of 1024 audio samples at $22.05kHz$. In this way a Gaussian timbre model x finally consists of a 25-dimensional mean vector μ, and a 25×25-dimensional covariance matrix Σ.

To compute the similarity between two Gaussians the Kullback-Leibler divergence (kld) can be used. There exists a closed form of this divergence [17] which allows the divergence to be computed between two m-dimensional Gaussians $x_{1,2} \sim \mathcal{N}(\mu, \Sigma)$.

$$\begin{aligned} 2\, kld(x_1||x_2) = \\ \log_e\left(\frac{\det \Sigma_2}{\det \Sigma_1}\right) + \operatorname{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + \\ (\mu_2 - \mu_1)^\top \Sigma_1^{-1}(\mu_2 - \mu_1) - m \end{aligned} \quad (1)$$

Since the kld is asymmetric usually a symmetrized variant ($skld$) of the divergence is used for music similarity estimation:

$$skld(x_1||x_2) = \frac{kld(x_1||x_2) + kld(x_2||x_1)}{2} \quad (2)$$

3.2 Self Organizing Map (SOM) Algorithm

The SOM [6] is an unsupervised neural network that organizes multivariate data on a two dimensional map and is suited well for visualizations. It maps items which are similar in the original high-dimensional space onto locations close to each other on the map.

Basically the SOM consists of an ordered set of so-called *map units* r_i, each of which is assigned a *reference vector* (or *model vector*) m_i in the feature space. The set of all reference vectors of a SOM is called its *codebook*. In the simplest case the codebook is initialized by a random strategy.

To compute a SOM, first the map dimensions and the number of training iterations (t) are fixed. Training is done in four basic repeating steps:

1. At iteration t select a random vector $v(t)$ from the set of features.

2. Search for the best matching map unit c on the SOM by computing the (Euclidean) distance of $v(t)$ to all m_i.

3. The codebook is updated by calculating a weighted centroid between $v(t)$ and the best matching unit r_c. Based on a neighborhood weighting function $h_{ci}(t)$ all map units participate in the adaptations depending on their distance on the two-dimensional outout

map. Equation 3 shows a standard Gaussian neighborhood function.

$$h_{ci}(t) = \alpha(t) \exp\left(-\frac{||r_c - r_i||}{2\alpha^2(t)}\right) \quad (3)$$

$$m_i(t+1) = m_i(t) + h_{ci}(t)\left[v(t) - m_i(t)\right] \quad (4)$$

4. The adaptation strength $\alpha(t)$ is decreased gradually with the iteration cycle t. This supports the formation of large clusters in the beginning and a fine-tuning toward the end of the training.

Usually, the iterative training is continued until a convergence criterion is fulfilled or a preselected number of training iterations is finished. In the final step all items are assigned to the map unit they are most similar to.

A popular way of visualizing SOMs trained with music similarity features is the Smoothed Data Histogram (SDH) [16].

4. FROM VECTORS TO GAUSSIAN DISTRIBUTIONS

Although originally SOMs were defined for Euclidean feature vectors only, the algorithm per se is not limited to the vector space. Kohonen himself mentions this in the most recent edition of his standard work on Self-Organizing Maps [6]. This observation will be the basis for extending the SOM algorithm to the 'distribution space'.

A closer look at the SOM algorithm sketched in the last section, shows that there is only a single step where the algorithm in fact depends on vectors. It is the computation of the weighted centroid in Equation 4 [2]. We now rewrite Equation 4 so that it is more obvious that a centroid (a weighted mean of several vectors) is computed:

$$m_i(t+1) = (1 - h_{ci}(t))\, m_i(t) + h_{ci}(t) v(t) \quad (5)$$

The essence of this is if a weighted centroid can be computed for Gaussians and the symmetrized Kullback-Leibler divergence, the SOM algorithm would, without modifications, work for our data.

4.1 Weighted Symmetrized Kullback-Leibler Centroid

The Kullback-Leibler divergence is part of the broad family of Bregman divergences [3]. In 2005 Banerjee et al. showed that a unique centroid exists for any Bregman divergence and proved that the standard k-means (and with that basically any centroid-based) works in this rich family of divergences [1].

As Bregman divergences are asymmetric divergences, there exist three uniquely defined centroids for a divergence D and a set of points x_i: the left-sided centroid c_L, right-sided centroid c_R and symmetrized centroid c_S. The centroids are the optimizers of the minimum average distance:

$$\boldsymbol{c_L} = \underset{c}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} D(\boldsymbol{c}||x_i) \quad (6)$$

$$\boldsymbol{c_R} = \underset{c}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} D(x_i||\boldsymbol{c}) \quad (7)$$

$$\boldsymbol{c_S} = \underset{c}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \frac{D(x_i||\boldsymbol{c}) + D(\boldsymbol{c}||x_i)}{2} \quad (8)$$

As Nielsen and Nock show [13], no closed analytical form to compute the symmetrized centroid exists. In their paper they present an efficient geodesic walk algorithm to find the symmetrized Bregman centroid c using the left c_L and right c_R centroids. In the last section they also define the three centroids for the Kullback-Leibler divergence and $x_i \sim \mathcal{N}(\mu_{x_i}, \Sigma_{x_i})$ multivariate Gaussians.

To use these centroids in the SOM algorithm we need to modify them and add a weighing term λ_i (with $\sum_{i=1}^{n} \lambda_i = 1$) for each Gaussian. The individual weighted centroid definitions are given in the next paragraphs [3].

4.1.1 Weighted Right-Sided Gaussian kld-Centroid

The right-type kld-centroid Gaussian $c_R \sim \mathcal{N}(\mu_{c_R}, \Sigma_{c_R})$ coincides with the center of mass. For Gaussians x_i and the kld it is defined as:

$$\mu_{c_R} = \sum_{i=1}^{n} \lambda_i \mu_i \quad (9)$$

$$\Sigma_{c_R} = \sum_{i=1}^{n} \lambda_i \left(\mu_i \times \mu_i^T + \Sigma_i\right) - \mu_{c_R} \times \mu_{c_R}^T \quad (10)$$

with λ_i as defined above.

4.1.2 Weighted Left-Sided Gaussian kld-Centroid

The left-type kld-centroid Gaussian $c_L \sim \mathcal{N}(\mu_{c_L}, \Sigma_{c_L})$ is obtained equivalently by minimizing the dual right-type centroid problem. For Gaussians x_i and the kld it is defined as:

$$\mu_{c_L} = \Sigma_{c_L} \times \sum_{i=1}^{n} \lambda_i (\Sigma_i^{-1} \times \mu_i) \quad (11)$$

$$\Sigma_{c_L} = \left(\sum_{i=1}^{n} \lambda_i \Sigma_i^{-1}\right)^{-1} \quad (12)$$

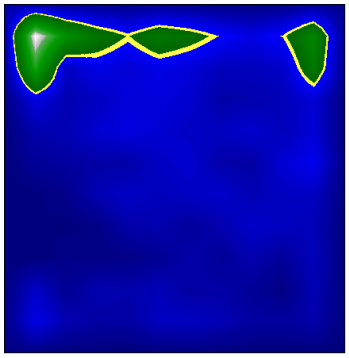
with λ_i as defined above.

4.1.3 Weighted (mid-point) Gaussian $skld$-Centroid

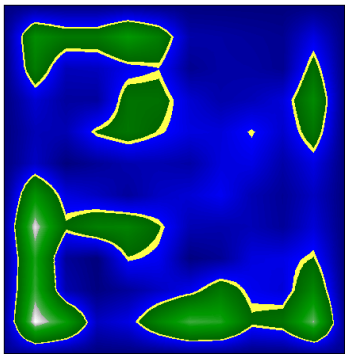
To compute the weighted symmetrized Kullback-Leibler centroid, the weighted left and right centroids need to be computed. We then use the mid-point centroid approximation instead of computing the exact centroid. The mid-point empirically proved to be a good approximation of the true centroid of the $skld$ [19]. The approximation

[2] The distance $||r_c - r_i||$ in Equation 3 is computed in 'map space' and does not need to be modified.

[3] A detailed listing and explaination of the derivation of each centroid can not be given due to the length constraints of this paper.



(a) Euclidean Vectors (from the similarity matrix)



(b) Gaussians

Figure 1. A 10×10 SOM, computed with the two different approaches. The SOMs are visualized using the Matlab Smoothed Data Histogram Toolbox [16] clustering a collection of 1 000 music pieces. They were created using identical parameters for initialization and learning.

$c_S \sim \mathcal{N}(\mu_{c_S}, \Sigma_{c_S})$ merges the weighted left and right centroids in one step:

$$\mu_{c_S} = \frac{1}{2}\left(\mu_{c_L} + \mu_{c_R}\right) \tag{13}$$

$$\Sigma_{c_S} = \frac{1}{2}\sum_{i=\{L,R\}}\left(\mu_{c_i} \times \mu_{c_i}^T + \Sigma_{c_i}\right) - \mu_{c_S} \times \mu_{c_S}^T \tag{14}$$

4.2 The Generalized SOM

With the definition of the weighted $skld$-centroid everything is in place to use the SOM algorithm with Gaussian music-timbre models and the $skld$:

1. Initialization of the SOM and its m_i is done by selecting random Gaussians from the music-timbre models.
2. Most importantly the iterative computation of the weighted centroid during the training of the codebook can now be replaced with the weighted symmetrized kld centroid.
3. The learning rate adaptation and neighborhood functions do not need to be changed. They are not dependent on the features.
4. In the final step the Gaussians are assigned to the nearest map units according to $skld$.

The approach of using the SOM directly with the Gaussian features is very close to the data and the original intention of the algorithm. We evaluate the generalized SOM in the next section.

5. EVALUATION

To compare SOMs generated with different approaches, we quantify how well the original neighborhood topology is preserved in a SOM mapping. As we need to compare SOMs using different metrics it is not possible to use standard SOM quality measures like the quantization error. Therefore we are using a rank distance. We search for the n nearest neighbors of every item x_i in the original space and check their location on the SOM. Ideally the nearest neighbors should also be mapped close to each other on the SOM. For a given number n of nearest neighbors and a Gaussian x_i this will be measured as the n *nearest neighbor rank distance*:

1. Assign all Gaussians to their corresponding map unit on the SOM.
2. For Gaussian x_i compute the Euclidean distance of its assigned map unit on the SOM to the map units assigned to all other Gaussians.
3. Sort the list of Euclidean distances in ascending order and transform it into a list of ranks.
4. Find the n nearest neighbors of x_i in the original space and average across their corresponding ranks.

The *average* n *nearest neighbor rank distance* of all x_i is a value describing the whole SOM. The lower its value, the better the preservation of the neighborhood on the SOM.

5.1 Setup

To test how the SOM algorithm performs operating directly on the Gaussians we used a test collection of 16 754 songs. The songs are typical full three to five minutes songs of a mixed genre music collection. We compute the Gaussian timbre music similarity features for these songs (see Section 3.1) so that every song is characterized by a 25-dimensional Gaussian.

To compare the quality of the SOMs generated with our approach, we also generate SOMs with vectorized features. For each Gaussian feature we build a vector by computing the distance to all other features and normalizing this distance vector to zero mean and unit variance. This is equivalent to computing the full similarity matrix and using each row as a feature vector (done e.g. in [5, 14]).

As a baseline for our experiments we also use a randomly initialized SOM without any training. In our experiments we vary various SOM parameters to test different

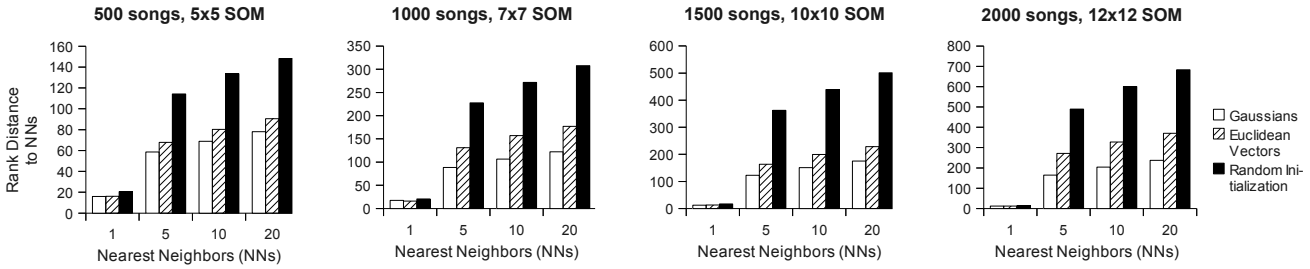


Figure 2. Four plots of the average n-nearest neighbor rank distance using different SOM configurations. Each plot compares the two strategies (Gaussian/Vectorized) to compute a SOM. As a baseline a randomly initialized SOM (black bar) is added. Lower rank distance values indicate that the original nearest neighbors (NNs) are mapped closer to each other on the SOM. The plots clearly shows that using Gaussians directly produces SOMs better preserving the neighborhood.

configurations: (1) we used SOM grid sizes of 5×5, 7×7, 10×10, 12×12, (2) and mapped 500, 1 000, 1 500 or 2 000 songs (randomly drawn from the base collection).

To ensure a fair evaluation we took the following precautions:

- In each run the same random seed is used for the random, vectorized and Gaussian SOM. This ensures identical random initialization and use of the same randomly chosen features during the training phase.
- The previously defined average n-nearest neighbor rank distance is computed for each map ($n = 1, 5, 10, 20$).
- Each unique experiment configuration is repeated ten times. Results are averaged.

5.2 Results

Figure 2 shows the average n-nearest neighbor rank distance for four selected SOM configurations which have been evaluated. In these experiments it can be seen that directly using Gaussians to train a SOM results in maps which are able to better preserve the neighborhood.

The results of all experiments conducted are summarized in Table 1. The table expresses the improvement of SOMs created with the Gaussian and vectorized approach relative to the randomly initialized SOMs. It confirms the results of the previous figure that throughout the configurations SOMs computed directly with the Gaussians produce higher-quality mappings and that this method should be preferred.

An illustration comparing two SOMs created with the two approaches is plotted in Figure 1. Albeit we can not make any judgements concerning the quality of the SOMs from the plots, we can see that a more structured SOM emerged from directly using the Gaussian features.

Besides producing higher-quality SOMs, we emphasize that this approach is also far less complex to compute than a variant working with vectorized features: (1) it is almost impossible to compute the full similarity matrix on a large collection of songs (i.e. over 100 000 songs) and (2) a SOM with 100 000-dimensional (or larger) vectors would be very expensive to compute. By using random projections [2] one can overcome that, but that would probably come with a loss of mapping quality. A SOM computed directly with the Gaussians, on the other hand, requires only a fraction of the computational effort, as the full similarity matrix does not need to be computed and the original features are used as intended.

		Nearest Neighbors			
Features	*Type*	1	5	10	20
500	Gaussians	**0.90**	**0.41**	**0.41**	**0.43**
	Vectors	0.97	.55	0.56	0.54
1000	Gaussians	**0.80**	**0.40**	**0.40**	**0.41**
	Vectors	1.07	0.64	0.63	0.62
1500	Gaussians	**0.75**	**0.38**	**0.38**	**0.40**
	Vectors	0.75	0.60	0.60	0.60
2000	Gaussians	**0.76**	**0.41**	**0.42**	**0.43**
	Vectors	0.77	0.70	0.70	0.69

Table 1. The table shows the average n-nearest neighbor rank distance of a SOM in relation to a randomly initialized one. Lower ratios indicate a better neighborhood topology preservation. It can be seen that in each configuration the Gaussian approach produces better mappings.

6. DISCUSSION & FUTURE WORK

We have shown how to compute a weighted symmetrized Kullback-Leibler centroid on multivariate Gaussians and on top of that how to directly and naturally compute a SOM with Gaussian music similarity features. The SOMs computed with that approach are shown to produce better mappings and omit the so far necessary step of vectorizing the data to compute a SOM.

The approach easily fits into the large number of existing SOM visualizations using Gaussian music similarity features together with a Kullback-Leibler related divergence and is of course not limited to music similarity. By using it the quality of the produced SOM should increase and the application can scale to larger collections of features.

Besides computing SOMs we also gave a clear definition of how to compute the weighted symmetrized Kullback-Leibler centroid so that it can be re-used to solve different problems where the features are also parametric Gaussians: for example to do a k-means cluster analysis in music collections directly with the Gaussian features. Maybe the centroid could also be of use to build an indexing algorithm for faster nearest neighbor retrieval.

ACKNOWLEDGMENTS

This research is supported by the Austrian Research Fund (FWF) under grants L511-N15, P21247 and the Vienna Science and Technology Fund (WWTF). The Austrian Research Institute for Artificial Intelligence (OFAI) acknowledges financial support form the Austrian Federal Ministries BMVIT and BMWF.

7. REFERENCES

[1] A. Banerjee, S. Merugu, I.S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

[2] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM New York, NY, USA, 2001.

[3] L.M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

[4] J. Stephen Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[5] P. Knees, M. Schedl, T. Pohle, and G. Widmer. Exploring music collections in virtual landscapes. *IEEE multimedia*, 14(3):46–54, 2007.

[6] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[7] S. Leitich and M. Topf. Globe of music: Music library visualization using geosom. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR07), Vienna, Austria*, 2007.

[8] B. Logan. Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval*, 2000.

[9] D. Lübbers. SoniXplorer: Combining visualization and auralization for content-based exploration of music collections. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR05)*, 2005.

[10] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR), London, UK*, 2005.

[11] M. Mandel and D. Ellis. Labrosas audio music similarity and classification submissions. *Music Information Retrieval Information Exchange (MIREX)*, 2007.

[12] F. Mörchen, A. Ultsch, M. Nöcker, and C. Stamm. Databionic visualization of music collections according to perceptual distance. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR), London, UK*, 2005.

[13] F. Nielsen and R. Nock. Sided and symmetrized Bregman centroids. *IEEE Transactions on Information Theory*, 55(6):2048–2059, 2009.

[14] E. Pampalk. Computational models of music similarity and their application in music information retrieval. *Docteral dissertation, Vienna University of Technology, Austria*, 2006.

[15] E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *Proceedings of the 10th ACM international conference on Multimedia*, page 579. ACM, 2002.

[16] E. Pampalk, A. Rauber, and D. Merkl. Using smoothed data histograms for cluster visualization in self-organizing maps. *Lecture notes in computer science*, pages 871–876, 2002.

[17] W.D. Penny. Kullback-Leibler divergences of normal, gamma, Dirichlet and Wishart densities. *Wellcome Department of Cognitive Neurology*, 2001.

[18] A. Rauber and M. Frühwirth. Automatically analyzing and organizing music archives. *Lecture Notes in Computer Science*, pages 402–414, 2001.

[19] R. Veldhuis. The centroid of the symmetrical Kullback-Leibler distance. *IEEE signal processing letters*, 9(3):96–99, 2002.

[20] Y. Wu and M. Takatsuka. Spherical self-organizing map using efficient indexed geodesic data structure. *Neural Networks*, 19(6-7):900–910, 2006.

IT'S TIME FOR A SONG – TRANSCRIBING RECORDINGS OF BELL-PLAYING CLOCKS

Matija Marolt
University of Ljubljana
Faculty of Computer and Information Science
matija.marolt@fri.uni-lj.si

Marieke Lefeber
Meertens Instituut and Museum Speelklok
marieke.lefeber@meertens.knaw.nl

ABSTRACT

The paper presents an algorithm for automatic transcription of recordings of bell-playing clocks. Bell-playing clocks are clocks containing a hidden bell-playing mechanism that is periodically activated to play a melody. Clocks from the eighteenth century give us unique insight into the musical taste of their owners, so we are interested in studying their repertoire and performances - thus the need for automatic transcription. In the paper, we first present an analysis of acoustical properties of bells found in bell-playing clocks. We propose a model that describes positions of bell partials and an algorithm that discovers the number of bells and positions of their partials in a given recording. To transcribe a recording, we developed a probabilistic method that maximizes the joint probability of a note sequence given the recording and positions of bell partials. Finally, we evaluate our algorithms on a set of recordings of bell-playing clocks.

1. INTRODUCTION

Bell-playing clocks are clocks containing a hidden bell-playing mechanism, which is activated every hour, every half an hour or even every quarter of an hour to play a melody (see Figure 1). To make this happen, the going train activates the musical train, which starts the rotation of the musical cylinder. The cylinder contains a pattern of pins which 'play' a series of keys as the cylinder rotates. Through threads these keys are connected to hammers, which strike the bells. Such bell-playing mechanisms are usually part of longcase- or bracket clocks.

Bell-playing clocks probably originate from carillons which played their melodies already in the thirteenth century in the towns of the Low Countries. From the end of the fifteenth century these instruments were also made for domestic use, but they were unique pieces, only affordable for the very rich. From the end of the seventeenth century, bell-playing clocks became more and more popular, although they still remained a status symbol, only affordable for the rich elite. Many eighteenth-century bell-playing clocks have been preserved. Clock restorer Melgert Spaander from Zutphen (Netherlands) restored and recorded over 150 of these clocks and made these recordings available for our researches. The collection consists of approximately 1500 melodies, which offer us, in a way, recordings from the eighteenth century. We are studying the repertoire of these clocks and also the performances of melodies with the aim of gaining more insight into the musical taste of the eighteenth-century elite. In order to study the repertoire of clocks, we need to transcribe all of the recorded melodies, so that they can be analyzed. Transcribing these melodies by hand requires a lot of practice and is made even more difficult by the inharmonicity and long decay times of bell sounds.

Figure 1. Melodies of a bell-playing clock.

In this paper, we present an algorithm for automatic transcription of recordings of bell-playing clocks. Automatic music transcription is a difficult problem to solve, although methods are improving constantly; Klapuri and Davy provide an extensive overview of the current state of the art [1]. Because we know little of the acoustical properties of clock bells, we could use unsupervised learning techniques for transcription. Such techniques have been used previously by several authors: Abdallah and Plumbley [2] used sparse coding for transcription of synthesized harpsichord music, while Virtanen used it to transcribe drums [3]. A number of authors use variants of non-negative matrix factorization to transcribe polyphonic music [4-7]. Their methods, however, were devised for music composed of harmonic sounds and are thus difficult to apply to our domain. Recently, Marolt [8] proposed to use non-negative matrix factorization with selective sparsity constraints to transcribe recordings of church bells.

While we initially experimented with unsupervised learning techniques, we obtained better results with the method proposed in this paper. We propose a two step approach to transcription: first, we present an analysis of acoustical properties of bell sounds and derive an algorithm that discovers the number of bells and positions of their partials in a given recording. A probabilistic method that relies on analysis of the recorded signal, the found bell partials, and on some higher-level musical knowledge is used to perform the transcription. We evaluate our approach on a collection of recordings of bell-playing clocks.

2. IDENTIFYING BELLS IN A RECORDING

2.1 Modeling Positions of Bell Partials

The shape or profile of a bell determines the relative frequencies of its vibrations. Bells have distinct but inharmonic partials – a partial being a frequency of vibration present in the sound of a bell. Little is known of the acoustical properties of bells used in clocks, so we first conducted a study to determine whether we can model the positions of bell partials. To estimate the properties of clock bells, we analyzed a set of 10 recordings of different clocks, containing a total of 88 bell sounds and manually annotated positions of their partials. Bells were found to have six strong partials; their positions relative to the perceived pitch (in cents) are listed in Table 1.

Mean and st. dev. of partial - pitch freq. (cents)	Mean magnitude relative to the strongest partial (dB)
0 ± 0	-21.8
1485 ± 81	-11.4
2433 ± 111	-4.8
3145 ± 124	-8.6
3719 ± 123	-13.0
4236 ± 91	-17.9

Table 1. Means and std. deviations of relative partial positions (in cents) for the analyzed bells. Mean magnitude of each partial relative to the strongest partial (in dB) is also shown

We can observe that partials are centered at approximately 2.4, 4, 6, 8.5 and 11.5 times the fundamental frequency (in Hz), the 3th and 4th partial being the loudest. The fundamental frequency corresponds to the perceived pitch. When studying relationships between these partials, we discovered a regularity, not unlike what Hibbert [9] discovered for church bells, namely that relationships between relative positions of partials (i.e. logarithmic frequency ratios) are linear. Figure 2 shows scatter plots of relative positions of partials 2, 4, 5 and 6 versus the relative position of the third partial - the linearity is obvious. This enables us to fit a linear regression model (also shown in Figure 2) that can be used to predict the positions of all six bell partials, if we know the positions of two of them, with an average error below 20 cents.

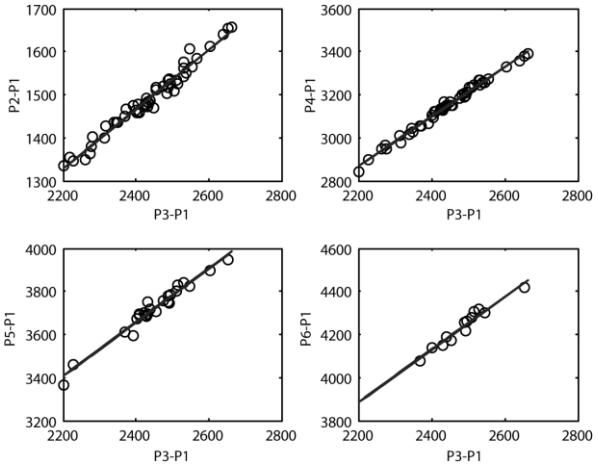


Figure 2. Linearity of relative partial positions.

Formally, a model that defines positions of bell partials given positions of two partials is expressed as a sum of Gaussians:

$$M_{f1,f2}(x) = \sum_{k=1}^{6} \exp\left(-\frac{(x - r^{k}_{f1,f2})^2}{2\sigma^2} \right), \qquad (1)$$

where f_1 and f_2 are positions of any two bell partials, $r^{k}_{f1,f2}$ the k-th partial position as calculated by the regression fit and σ the allowed deviation from the regression fit.

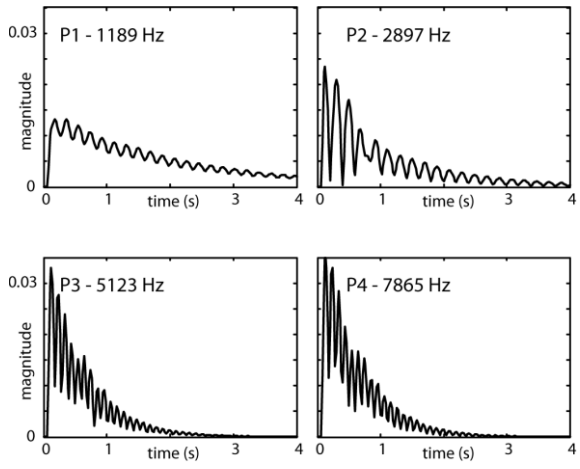


Figure 3. Time evolution of the first four partials of a clock bell.

Time evolution of partials follows an exponential decay curve, and is faster for higher partials, which on the other hand, are initially louder. Individual partials frequently exhibit beating, also evident in Figure 3. Beating is caused by the so-called doublets, which arise when bells are not symmetrical about a vertical axis through

their centers. This asymmetry causes most of the vibrational modes in bells to split into two distinct modes with slightly different frequencies that beat against each other. Beating is problematic when we try to estimate the onsets and time evolution of partials, so we try to remove its effect, as described in section 3.

2.2 Bells? What Bells?

When analyzing a recording of a bell-playing clock, we initially have no information on the number of bells involved, their tuning or positions of their partials. In this section, we introduce an algorithm that uses the bell partial model presented in section 2.1 to estimate the number of bells in a recording and positions of their partials. The algorithm is based on the observation that bells are rarely struck at the same time, which is mainly due to the inharmonic nature of bell sounds and their imperfect tuning. Therefore, we can use the "common fate" auditory grouping principle, especially onset synchrony to find groups of partials that belong to individual bells.

We first calculate a magnitude spectrogram **F** of a recording. To reduce variance in partial magnitudes in different frequency regions, we multiply the spectrogram with a perceptual weighting model, as introduced by Vincent [10]. Weights are calculated on an average spectrum and applied globally to yield a flattened time-frequency representation $\mathbf{F_w}$. Such flattening "amplifies" partials with small magnitudes, which makes it easier for the algorithm to consider those partials in the process of finding partial groups. This is especially important, because magnitude of the fundamental frequency of a bell lies over 20 dB below its loudest partial (see Table 1). As the fundamental corresponds to pitch, we need to accurately estimate it, otherwise the pitch of a bell can only be approximated.

Bells have sharp onsets and long decay times, so the next step of the algorithm accentuates the fast positive changes (sharp onsets) in the magnitude spectrogram. The dynamics of changes within frequency bins of $\mathbf{F_w}$ is estimated by calculating first order delta coefficients **D** of the bins with a sliding window of length N_d. Delta coefficients provide estimates of the gross shape of short time segments of the frequency bins. They emphasize fast and big changes, such as onsets, and deemphasize slower and smaller changes, such as beating. This is illustrated in Figure 4 that displays delta coefficients of a bell partial calculated on a recording of a bell-playing clock.

To discover groups of partials with synchronous onsets, we calculate covariances of their delta coefficients:

$$c_{ij} = \max\left(\frac{1}{n-1} \sum_{k=1}^{n} (d_{ik} - \mu_i)(d_{jk} - \mu_j), 0 \right) \quad , \quad (2)$$

where d_{ij} represents an element of the delta spectrogram **D** and μ_i the average of the i-th row of **D**. Because delta coefficients emphasize onsets, the value c_{ij} represents a measure of onset synchrony of partials with frequencies corresponding to bins i and j. Bells do not share many partials and are rarely struck at the same time, so a bell's partial will have high synchrony with other partials of the same bell, but not with partials of other bells.

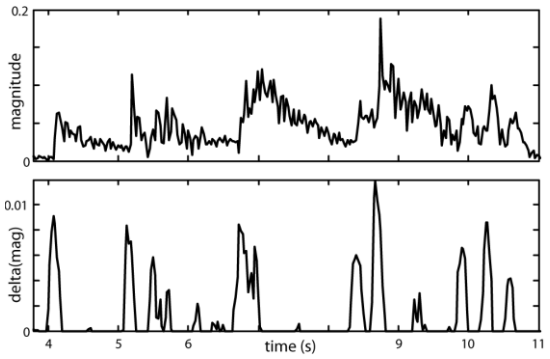


Figure 4. Amplitude envelope and delta coefficients of a bell partial from a bell-playing clock recording

A global covariance matrix **C** could be calculated on the entire delta spectrogram **D**, but we found that this puts too much emphasis on bells that occur frequently in a recording and fails to find partials groups of other less frequent bells. We therefore calculate local covariance matrices on all segments that are obtained by sliding a window of length n over the delta spectrogram **D** with a step size of $n/2$. This results in a set of local covariance matrices $\mathbf{C}^{(t)}$. The overall measure of onset synchrony of a partial i is then calculated by weighting the contributions of local covariance matrices with the overall energy of the partial in each segment, as approximated by $c_{ii}^{(t)}$:

$$s_{ij} = \frac{1}{\sum_{t=1}^{m} c_{ii}^{(t)}} \sum_{t=1}^{m} c_{ii}^{(t)} c_{ij}^{(t)} \quad . \quad (3)$$

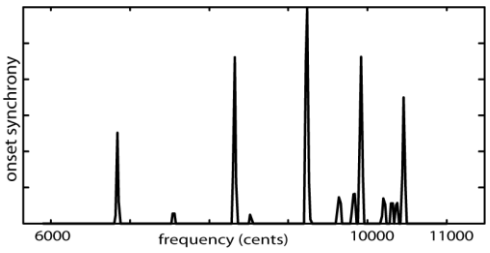


Figure 5. Onset synchrony of a partial at 6840 cents. Four other partials from the same bell (8320, 9240, 9920 and 10460 cents) are clearly visible.

Figure 5 displays one row of the resulting matrix **S**, representing onset synchrony of a partial in a bell clock recording. A group of five partials belonging to the same bell sound clearly stands out.

To discover groups of synchronous partials, we analyze each row $\mathbf{s}_i$ of the matrix **S**, and search for parameters of the bell model presented in section 2.1 that best

describe $\mathbf{s}_i$. Specifically, for each row $\mathbf{s}_i$ we find model parameters f_1 and f_2 that maximize:

$$\arg\max_{f1,f2} M_{f1,f2} \cdot \mathbf{s}_i \quad , \tag{4}$$

where · denotes the dot product operator. Due to the sparseness of **S** (see Figure 5), an exhaustive search for optimal parameters can be performed efficiently. If the dot product in eq. (4) exceeds a preset threshold *T*, the model $M_{f1,f2}$ is considered to represent one of the bells in the analyzed recording. The actual positions of bell's partials may deviate from the model, so we estimate them from $\mathbf{s}_i$ by simple component-wise multiplication:

$$\mathbf{b} = (M_{f1,f2} \bullet \mathbf{s}_i)^{\frac{1}{K}} \ , \tag{5}$$

where *K* is used to compress partial magnitudes.

The final outcome of the algorithm is a set of vectors $\mathbf{b} \in \mathfrak{B}$ describing the sounds of bells in a recording. We present an evaluation of the algorithm in section 4.

3. TRANSCRIPTION

To transcribe a recording of a bell-playing clock, we need to find which notes (bells) were played and when they were played - their onset times. Although this may seem to be an easy task given positions of partials of bell sounds, the task is complicated due to several factors. First, partials interact; they get amplified, cancelled or beat against each other, which makes it difficult to follow their amplitude envelopes and find their onsets. Decay times of bell sounds are long and although bells are usually not played at the same time, the number of concurrently sounding bells (polyphony) is always high. Partials decay at different rates, so the spectrum of bells changes with time. Recordings contain fast passages with inter-onset times of less than 100ms, as well as embellishments such as grace notes and arpeggios that further complicate transcription. And last, these are not synthetic recordings, nor are they very professionally made; they contain many noisy artefacts, such as background noise, noises coming from the clock mechanism or similar.

We chose to take a probabilistic approach to transcription and search for the most probable sequence of notes in a recording. The transcription process starts by calculating the onset times and onset probabilities of bells. We use the complex domain onset detection function and peak picking algorithm [11], which performs well with bell sounds, because of their sharp percussive onsets. Onset probabilities are calculated from the value of the onset detection function at each onset.

Given *N* onset times and the fact that bells are seldom struck at the same time, transcription can be viewed as a problem of finding a sequence of notes and rests $s_1,s_2,s_3...s_N$ that best describes the analyzed signal; s_1 starts at the first found onset, s_2 at the second and so on. *s* may represent a note (all notes n_i, i=1..*M* are described by their corresponding bell models from the set $\mathfrak{B}$); or may be a rest (*r*). Specifically, we wish to find a sequence of notes and rests that maximizes the joint probability:

$$P(s_1)P(s_2 \mid s_1)P(s_3 \mid s_2)\cdot\ldots\cdot P(s_N \mid s_{N-1}) \, . \tag{6}$$

To estimate probability of a note $P(s_i=n_j)$, we take two factors into consideration: the probability that note n_j described by the corresponding bell model $\mathbf{b}_j$ actually occurred in the signal at onset *i*, and the probability of that onset. Note probability is calculated by multiplying the bell model with the time-frequency representation **D** (as defined in section 2.1). Onset probability is proportional to the value of the onset detection function at the onset. We can thus write the probability of a note n_j occurring at onset *i* as:

$$P(s_i = n_j) = P(o_i) \tfrac{1}{C_i} \mathbf{b}_j \cdot \mathbf{d}_i \, . \tag{7}$$

where · denotes dot product, $\mathbf{d}_i$ represents the time-frequency representation **D** at time *i* and $P(o_i)$ the probability of an onset at that time. C_i is a scaling factor used to normalize the dot product to a [0-1] range.

Probability of a rest is defined as:

$$P(s_i = r) = (1 - P(o_i)) \prod_{k=1}^{M} \left(1 - P(s_i = n_k)\right), \tag{8}$$

thus if no notes are likely to occur and the onset is also not likely, a rest will be likely.

To define conditional probabilities $P(s_i \mid s_{i-1})$, we introduced two changes to the above expressions. First, if note n_k occurred at time *i-1*, we subtract the note from the time-frequency representation **D**, thus eliminating its effect at time *i*:

$$\mathbf{d}_i(s_{i-1}=n_k) = \max(\mathbf{d}_i - N(i-1,\sigma)\, \mathbf{b}_k \bullet \mathbf{d}_{i-1},\ 0) \, . \tag{9}$$

Operator • denotes component-wise multiplication and *N* the unscaled normal distribution, which models the time evolution of delta coefficients. As we can observe in Figure 4, delta coefficients are roughly bell-shaped at onsets, so we approximate them with a normal distribution. If s_{i-1} is a rest, nothing is subtracted, so $\mathbf{d}_i(s_{i-1}=r)=\mathbf{d}_i$.

As intervals between adjacent notes in a melody are usually small (a phenomenon also known as pitch proximity), we include an additional factor into $P(s_i \mid s_{i-1})$. Pitch proximity is modeled by a proximity profile $\mathbf{R}(n)$, which as in [12], is represented by a normal distribution centered around a given pitch *n*, indicating pitch probabilities of the following note. The obtained conditional probability of consecutive notes can thus be written as:

$$P(s_i = n_j \mid s_{i-1} = n_k) = P(o_i) R_i(n_k) \tfrac{1}{C_i} \mathbf{b}_j \cdot \mathbf{d}_i(s_{i-1}=n_k) \, . \tag{10}$$

If s_i is a rest, we can calculate the conditional probability with the expression given in eq. (8), whereby we replace note probabilities with conditional probabilities and multiply the expression with a constant representing the proximity profile.

The most likely sequence of notes and rests can be efficiently estimated with dynamic programming; the resulting set of onset times and notes represents the transcription of a recording.

4. EVALUATION

In order to evaluate our algorithm, we manually transcribed and annotated positions of partials in a set of 10 recordings of different bell-playing clocks. Results and discussion are given in the following sections.

4.1 The Bell-finding Algorithm

We used the following parameters to test the bell-finding algoritm: the magnitude spectrogram was calculated with the Constant-Q transform [13], using a maximum window size of 100ms, a step size of 25 ms and 20 cent spacing between adjacent frequency bins. The deltas were calculated with a sliding window of N_d=9 frames, the covariance matrices on n=100 frames long segments. Finally, the threshold T that determines whether a bell model should be included in the final results was set to $1/20^{th}$ of the maximum value of all models and the compression coefficient K to 10.

For comparison, we also developed an alternative approach for estimating partials in bell sounds. We used non-negative matrix factorization (NMF) to factorize the delta magnitude spectrogram **D** into matrices **W** and **H**, where the basis vectors in **W** would ideally correspond to bell spectra and **H** would explain how bell magnitudes change over time. Several efficient implementations of NMF exist in the literature; in our experiments we used the SNMF algorithm introduced by Kim and Park [14]. The algorithm is based on the alternating non-negativity constrained least squares and active set method and allows to impose sparsity constraints on **H** or **W**.

With NFM learning, the number of basis vectors is fixed and we need to set it in advance, prior to the actual learning. Because in our case basis vectors correspond to spectra of individual bells, we need to know the number of bells in a recording prior to learning. This is not usually the case, but to perform the comparison of both approaches, we give the NMF algorithm a small "advantage" by setting the number of basis vectors to the actual number of bells as was manually annotated for each recording.

	precision	recall
proposed algorithm	0.94	0.98
SNMF	0.87	0.87

Table 2. Comparison of two bell finding algorithms

Table 2 shows average precision and recall scores of the two algorithms on all recordings. Although both perform well, the proposed approach outperforms non-negative matrix factorization. We contribute the difference to two main reasons. To find partials of bell sounds, the proposed algorithm uses a local approach; namely covariance matrices are calculated on short segments of the entire recording and then combined based on magnitudes of the analyzed partials in these segments. On the other hand, NMF works globally by iteratively minimizing the factorization error. The difference is important when searching for bells that are not frequently played. NMF will tend to ignore them and rather focus on minimizing the overall error which may lie in varying decay times of bell partials or noise. The local nature of our approach will not fail for such cases, as the bells will stand out in individual local segments and will consequently also show in the global matrix **S**. Another advantage of the proposed approach is its use of the knowledge provided by the model of bell partial positions, as presented in section 2.1. Namely, the search for bell partials is limited by the model, so only regions of the signal that correspond to predicted partial frequencies are considered. Therefore, noise, either background or made by the clock mechanism or other external factors, can largely be ignored. NMF uses no such high-level knowledge, so it is affected by noise on all levels, as it tries to accommodate it and include it into the basis vectors.

All of the false negative errors (missed bells) made by our bell finding algorithm were bells a semitone apart from another more dominant bell, with most of their partials overlapping. Such bells are mainly used as embellishments and were ignored because their onsets were masked by the more dominant bell. However, since these bells are not very frequently played, these errors do not have a large influence on overall transcription accuracy, as we show in the following section.

4.2 The Transcription Algorithm

To evaluate how various choices made when designing our transcription algorithm influence its performance, we tested several variants of the algorithm: *A* – the described algorithm, *B* – excluding the pitch proximity profile, thus making all note transitions equally probable, *C* – excluding note subtraction, thus avoiding conditional note probabilities and *D* – using annotated onsets instead of the calculated ones and *E* – using annotated bell partials instead of the calculated ones. Average precision and recall scores of transcriptions of all recordings are shown in Table 3.

As we can observe, differences between these variants are not very big. This is due to the fact that the differences mostly affect "problematic" parts of recordings that include fast passages and embellishments, while elsewhere the combination of the delta magnitude spectrogram, accurately estimated positions of bell partials, and correctly found onsets makes them irrelevant.

For the problematic parts, the pitch proximity profile that favors smaller intervals (*B*) and especially note subtraction (*C*), which mostly prevents repetitions of predo-

minant notes, do have a positive effect on performance. As (*D*) shows, approx. half of the missed notes are caused by missed onsets and recall is raised by approx. 0.05 if perfect onset detection is used. On the other hand, nothing is gained by using the manually annotated bell partials, so the bell finding algorithm seems to be working very well and the errors it made seem to be almost irrelevant.

	transcription	
	precision	recall
A: proposed algorithm	**0.95**	**0.89**
B: no proximity	0.94	0.87
C: no conditional prob.	0.91	0.88
D: perfect onsets	0.94	0.94
E: perfect bell models	0.95	0.89

Table 3. Comparison of variants of the transcription algorithm

Most of the errors, either missed notes (false negatives) or extraneous notes (false positives) are made in fast passages, where note repetitions are missed, notes are transcribed in an incorrect order or weak onsets ignored. Overall, the performance is good enough, so that transcriptions will be used for further analysis and included in a searchable database of melodies; in fact when analyzing the errors, we discovered that several errors were in the ground truth and not in the calculated transcriptions.

5. CONCLUSION

The proposed approach to transcription of bell-playing clock recordings is a good first step towards analysis of these recordings. The bell-finding and transcription algorithms perform well and will be used to transcribe the entire collection of recordings of bell-playing clocks. We will add the resulting transcriptions to a searchable database of melodies, thus making them available to interested researchers for further analysis. There is room for improvements of the algorithm; we plan to consider ways of allowing for correct treatment of simultaneous notes, as well as to test the algorithm on other recordings containing bell sounds.

Acknowledgments. This work was supported in part by the Slovenian Government-Founded R&D project EthnoCatalogue: creating semantic descriptions of Slovene folk song and music.

6. REFERENCES

[1] A. Klapuri and M. Davy, Eds., *Signal Processing Methods for Music Transcription*. New York: Springer, 2006, p.^pp. Pages.

[2] S. A. Abdallah and M. D. Plumbley, "Unsupervised analysis of polyphonic music by sparse coding," *Neural Networks, IEEE Transactions on*, vol. 17, pp. 179-196, 2006.

[3] T. Virtanen, "Monaural Sound Source Separation by Nonnegative Matrix Factorization With Temporal Continuity and Sparseness Criteria," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, pp. 1066-1074, 2007.

[4] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, 2003, pp. 177-180.

[5] E. Vincent, *et al.*, "Harmonic and inharmonic Nonnegative Matrix Factorization for Polyphonic Pitch transcription," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, 2008, pp. 109-112.

[6] S. A. Raczynski, *et al.*, "Multipitch Analysis with Harmonic Nonnegative Matrix Approximation," in *ISMIR 2007, 8th International Conference on Music Information Retrieval*, Vienna, Austria, 2007, pp. 381-386.

[7] B. Niedermayer, "Non-negative Matrix Division for the Automatic Transcription of Polyphonic Music," in *ISMIR 2008, 9th International Conference on Music Information Retrieval*, Philadelphia, USA, 2008, pp. 544-545.

[8] M. Marolt, "Non-negative matrix factorization with selective sparsity constraints for transcription of bell chiming recordings," in *Sound and Music Computing Conference*, Porto, Portugal, 2009, pp. 137-142.

[9] W. A. Hibbert, "The Quantification of Strike Pitch and Pitch Shifts in Church Bells," Ph.D., Faculty of Mathematics, Computing and Technology, The Open University, Milton Keynes, UK, 2008.

[10] E. Vincent and M. D. Plumbley, "Low Bit-Rate Object Coding of Musical Audio Using Bayesian Harmonic Models," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, pp. 1273-1282, 2007.

[11] J. P. Bello, *et al.*, "On the use of phase and energy for musical onset detection in the complex domain," *Signal Processing Letters, IEEE*, vol. 11, pp. 553-556, 2004.

[12] D. Temperley, *Music and Probability*: The MIT Press, 2007.

[13] J. C. Brown, "CALCULATION OF A CONSTANT-Q SPECTRAL TRANSFORM," *Journal of the Acoustical Society of America*, vol. 89, pp. 425-434, Jan 1991.

[14] H. Kim and H. Park, "Nonnegative Matrix Factorization Based on Alternating Nonnegativity Constrained Least Squares and Active Set Method," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, pp. 713-730, 2008.

LEARNING FEATURES FROM MUSIC AUDIO WITH DEEP BELIEF NETWORKS

Philippe Hamel and Douglas Eck
DIRO, Université de Montréal
CIRMMT
{hamelphi,eckdoug}@iro.umontreal.ca

ABSTRACT

Feature extraction is a crucial part of many MIR tasks. In this work, we present a system that can automatically extract relevant features from audio for a given task. The feature extraction system consists of a Deep Belief Network (DBN) on Discrete Fourier Transforms (DFTs) of the audio. We then use the activations of the trained network as inputs for a non-linear Support Vector Machine (SVM) classifier. In particular, we learned the features to solve the task of genre recognition. The learned features perform significantly better than MFCCs. Moreover, we obtain a classification accuracy of 84.3% on the Tzanetakis dataset, which compares favorably against state-of-the-art genre classifiers using frame-based features. We also applied these same features to the task of auto-tagging. The autotaggers trained with our features performed better than those that were trained with timbral and temporal features.

1. INTRODUCTION

Many music information retrieval (MIR) tasks depend on the extraction of low-level acoustic features. These features are usually constructed using task-dependent signal processing techniques. There exist many potentially-useful features for working with music: spectral, timbral, temporal, harmonic, etc (see [21] and [3] for good reviews), and it is not always obvious which features will be relevant for a given MIR task. It would be useful to have a system that can automatically extract relevant features from the audio, without having to depend on *ad-hoc* domain-dependent signal processing strategies.

Among the most widely used frame-level features for audio-related MIR tasks Mel-Frequency Cepstral Coefficients (MFCCs). MFCCs take advantage of source/filter deconvolution from the cepstral transform and perceptually-realistic compression of spectra from the Mel pitch scale. Because the first few MFCC values capture pitch-invariant timbral characteristics of the audio, they are commonly used in tasks where it is useful to generalize across pitch, such as multi-speaker speech recognition and musical timbre recognition.

Practically all audio-based music genre classification models use different types of acoustic features to drive supervised machine learning [4, 13, 14, 23]. These include sparse audio encodings in the time domain [17] and in the frequency (spectral) domain [8]. Other approaches use a Hidden Markov Model (HMM) to build a semantic representation of music [7, 22]. The best reported accuracy on the Tzanetakis dataset [23] for genre classification was achieved by a system that used auditory cortical representations of music recordings and sparse representation-based classifiers [20]. The challenges and motivations of genre classification are discussed in [18]. In these approaches it is difficult to know whether the acoustic features or the machine learning techniques are responsible for success. To address this we apply our model to the Tzanetakis dataset.

A closely related task to genre classification is that of "autotagging" (automatic tag-based annotation of music audio). As for genre classification, timbral and temporal features are often used to solve this task [5]. To test the robustness of our learned features, we applied them to the task of autotagging on the Majorminer dataset [16].

Some work in automatic feature extraction for genre classification have been done. In [19], automatic feature selection was done with genetic algorithms, and used for one-on-one genre classification. In our approach, we use a Deep Belief Network (DBN) [10] to learn a feature representation. DBNs have already been applied in some MIR tasks. In [9], a DBN is compared to other classifiers for the instrument recognition task. In [12], convolutional DBNs are used to learn features for speech recognition and for genre and artist classification.

Can we learn features for a given task directly from musical audio that would better represent the audio than engineered signal-processing features? In this work, we investigate this question.

We propose a method to automatically extract a relevant set of features from musical audio. We will show that these learned features compare favorably against MFCCs and other features extracted by signal-processing.

The paper is divided as follows. In Section 2, we describe the datasets that were used in our experiments. We then explain briefly the DBN model in Section 3. In Section 4 we describe the feature learning process. Then, in Section 5 we give the results of our features used in genre classifica-

tion and autotagging tasks. Finally, we conclude and propose future work in Section 6.

2. DATASETS

We used two different datasets in our experiments. The first one is the Tzanetakis' dataset for genre recognition. We trained our feature extractor over this dataset. To test the robustness of our learned features, we then applied these same features to the task of autotagging on the Majorminer dataset.

2.1 Tzanetakis

This dataset consists of 1000 30-second audio clips assigned to one of 10 musical genres. The dataset is balanced to have 100 clips for each genre. The dataset was introduced in [24], and have since been used as a reference for the genre recognition task.

2.2 Majorminer

This dataset for autotagging was introduced in [16]. The tags were collected by using a web-based "game with a purpose". Over 300 tags have been assigned to more than 2500 10 second audio clips. For our experiment, we used only the 25 most popular tags and compared our results to those obtained in [16].

3. DEEP BELIEF NETWORKS

In the last few years, a large amount of research has been conducted around deep learning [1]. The goal of deep learning is to learn more abstract representations of the input data in a layer-wise fashion using unsupervised learning. These learned representations can be used as input for supervised learning in tasks such as classification and regression. Standard neural networks were intended to learn such deep representations. However, deep neural networks (i.e. networks having many hidden layers) are difficult or impossible to train using gradient descent [2]. The DBN circumvents this problem by performing a greedy layer-wise unsupervised pre-training phase. It has been shown [2, 10] that this unsupervised pre-training builds a representation from which it is possible to do successful supervised learning by "fine-tuning" the resulting weights using gradient descent learning. In other words, the unsupervised stage sets the weights of the network to be closer to a good solution than random initialization, thus avoiding local minima when using supervised gradient descent.

The Deep Belief Network (DBN) is a neural network constructed from many layers of Restricted Boltzmann Machines (RBMs) [2,10]. A schematic representation is shown in Figure 1. A RBM is structured as two layers of neurons: a visible layer and a hidden layer. Each neuron is fully connected to the neurons of the other layer, but there is no connection between neurons of the same layer. The role of a RBM is to model the distribution of its input. We can stack many RBMs on top of each other by linking the hidden layer of one RBM to the visible layer of the next

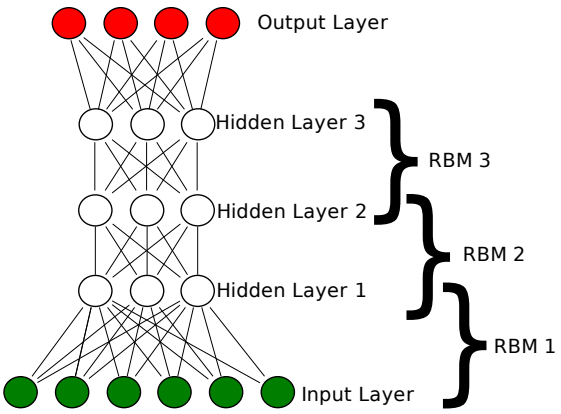


Figure 1. Schematic representation of a DBN. The number of layer and the number of units on each layer in the schema are only examples. We do not require to have the same number of units on each hidden layer.

RBM. In our experiments, we used an algorithm inspired by Gibbs sampling called Contrastive Divergence (CD) to optimize our RBMs. Our focus here is on analyzing the performance of the DBN, not in explaining the technical details of DBNs. The main idea for our purposes is that that DBNs offer an unsupervised way to learn multi-layer probabilistic representations of data that are progressively "deeper" (nonlinear) with each successive layer. For technical and mathematical details see [2, 10]. We used the Theano [1] python library to build and train our DBNs.

4. LEARNING THE FEATURES

Our goal is is to learn a representation of audio that will help us to solve the subsequent tasks of genre classification and autotagging.

4.1 Training the DBN

To learn our representation, we split the Tzanetakis' dataset in the following way: 50% for training, 20% for validation and 30% for testing. We divided the audio into short frames of 46.44ms (1024 samples at 22050 Hz sampling rate). For each of these frames, we calculated the discrete Fourier transform (DFT). We kept only the absolute values of the DFTs, and considering the symmetry in the DFT, we ended up with inputs of dimension 513.

The DBNs were first pre-trained with the training set in a unsupervised manner. We then proceeded to the supervised fine-tuning using the same training set, and using the validation set to do early-stopping. The supervised step used gradient descent to learn a weighted mixture of activations in the deepest layer to predict one of 10 genre. Both soft max and cross-entropy costs were minimized with comparable results.

We tried approximately 200 different hyper-parameters combinations and chose the model with the best validation error on the frame level. The chosen DBN model is described in Table 1.

[1] `http://deeplearning.net/software/theano/`

Number of hidden layers	3
Units per layer	50
Unsupervised learning rate	0.001
Supervised learning rate	0.1
Number of unsupervised epochs	5
Number of supervised epochs	474
Total training time (hours)	104
Classification accuracy	0.737

Table 1. Hyper-parameters and training statistics of the chosen DBN

The classifier trained from the last layer of the DBN yields a prediction of the genre for each frame. We average over all predictions for a song and choose the highest score as the wining prediction. This gave us a prediction accuracy of 73.7%.

Once trained, we can use the activations of the DBN hidden units as a learned representation of the input audio. We analyzed the performance of each layer of the network independently, and also all the layers together. To illustrate what is learned by the DBN, in Figure 2 we have plotted a 2-dimensional projection of some of the representations used. The projection was done by using the t-SNE algorithm described in [25]. Notice how the clustering of the activations of the hidden layers is more definite than for the input or the MFCCs. As we will see in Section 5, this will improve the accuracy of the classifiers.

5. CLASSIFICATION USING OUR LEARNED FEATURES

In this section, we use our learned features as inputs for genre classification and autotagging. In the first task we explore different ways of using our features to get the best classification accuracy. In the second task, we use the method that gave us the best result in the genre recognition in order to do autotagging.

For both experiments, we use a non-linear Support Vector Machine (SVM) with a radial basis function kernel [6] as the classifier. It would also be possible to train our DBN directly to do classification. However our goal is to compare the DBN learned representation with other representations. By using a single classifier we are able to carry out direct comparisons.

5.1 Genre classification

5.1.1 Frame-level features

In our first experiment, we used our frame-level features as direct input to the SVM. Since the SVM doesn't scale well with large datasets, we subsampled the training set by randomly picking $10,000$ frames. We compared these accuracies to the accuracy of the SVM trained with MFCCs over these same frames of audio. As in Section 4.1, we used the frame predictions of a whole song and voted for the best genre in order to compute the test accuracy. The results for this experiments are shown in Table 2. We see

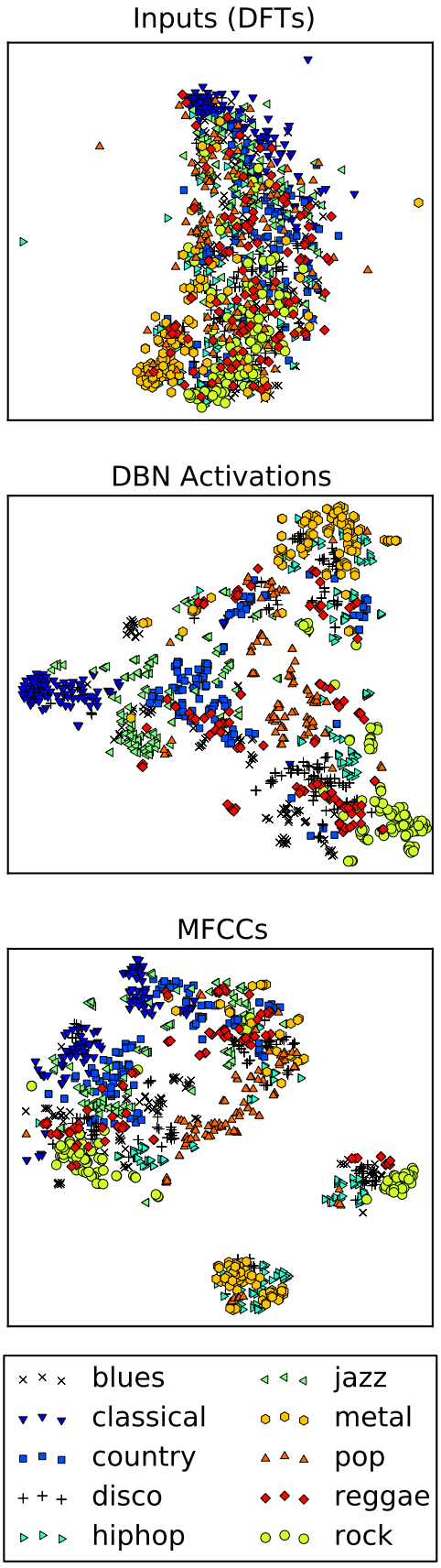


Figure 2. 2-Dimensional projections of different representations of the audio with respect to their genre.

	Accuracy
MFCCs	0.630
Layer 1	0.735
Layer 2	0.770
Layer 3	0.735
All Layers	0.770

Table 2. Classification accuracy for frame-level features

	Accuracy
MFCCs	0.790
Layer 1	0.800
Layer 2	0.837
Layer 3	0.830
All Layers	**0.843**

Table 3. Classification accuracy for features aggregated over 5 seconds

that, at the frame level, our learned features performed significantly better than the MFCCs alone. We also see that the second layer seems to have the best representation out of the three layers. By using all the layers as the input, we don't see any improvement compared to the second layer alone. Since we used the same dataset here that we used for learning the features, we took care to reuse that same training, validation and testing splits as in Section 4, so as not to contaminate our testing set. Because our learned DBN representation was learned on a single test/train split, we were unable to do cross-validation on this dataset with the SVM classifier, since this would have given us a biased result.

5.1.2 Aggregated features

Bergstra et al [4] investigated the impact of feature aggregation on classification performance for genre recognition. It is demonstrated that aggregating frame-level features over a period of time increases classification accuracy. The optimal aggregation time depend depends on the nature of the features and the classifier, with many popular features having optimal aggregation times of between 3 and 5 seconds. With this in mind, we aggregated our features over 5 seconds periods. Thus, for each 5 seconds segment of audio (with 2.5 seconds overlap), we computed the mean and the variance of the feature vectors over time. This method not only raised our classification accuracy, but also reduced the number of training examples, thus accelerating the training of the SVMs. With the aggregation, our classification accuracy by jumped to 84.3%, which is better than the 83% accuracy reported in [4]. However, since this result was reported on a 5-fold cross-validation on the dataset, we cannot directly compare our results. More importantly we observe that our results are in general competitive with the state-of-the-art signal-processing feature extraction for the genre classification task. Also, given a fixed classifier (the nonlinear SVM) our learned representation outperforms MFCCs. As in Section 5.1.1, we see that the second layer gives the best representation of all the layers, but we gain a bit of accuracy by using all of the layers.

5.2 autotagging

To test the robustness of our learned features, we tested their performance on an autotagging task. Following the results in Section 5.1, we used the activations of all the layers of the DBN aggregated on 5 second windows as inputs for the SVMs. We will refer to this set of feature as

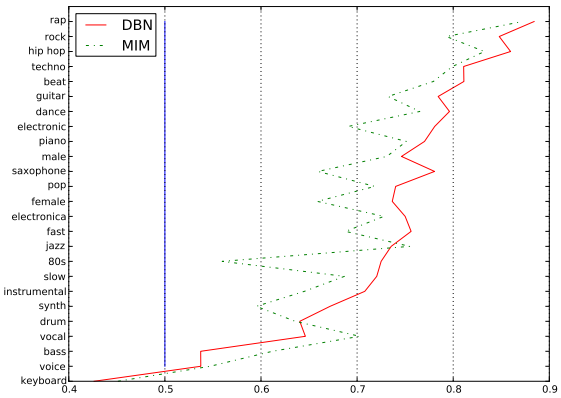


Figure 3. Accuracy of the DBN and the MIM feature sets for the 25 most popular tags. As each tag training set was balanced for positive and negative examples, the vertical line at 0.5 indicates chance accuracy.

the DBN feature set. We compare it to a set of timbral and temporal features presented in [15]. We will refer to this set of feature as the MIM feature set. We used the same method as in [16] to train the SVMs over the dataset. The results for the 25 most popular tags in the dataset are shown in Figure 3 and summarized in Table 4.

	Mean Accuracy	Standard Error
DBN	0.73	0.02
MIM	0.70	0.02

Table 4. Mean and standard error of the autotagging results.

The results show that our features give a better classification performance for almost all the tags. In particular, our features performed significantly better better for tags such as 'rock', 'guitar', 'pop' and '80s'. Except for 'guitar', these particular tags represent genres, which is what our features were optimized to classify.

5.3 Discussion

From the results presented in Section 5.1 and Section 5.2, we see that it is indeed possible to learn features from audio relevant to a particular task. In the case of genre classification, our DBN features performed as well if not better than most signal-processing feature extraction approaches. The features were optimized to discriminate between the

10 genres shown in Figure 2, but we showed that these features were also relevant to describe many other tags, such as 'guitar', that were not related to genre. We believe this is evidence that a DBN can in fact learn to extract important and robust characteristics from audio. Another positive point is that, once the DBN is trained, the feature extraction from audio is very fast and can be done easily in real-time, which could be useful for many applications.

However, there are several areas for improvement. The main one is the long computation time necessary to train the DBN. The model that we used required a few days to train. This is mainly due to the size of the dataset. Since we used uncompressed audio frames overlapping over half a frame, the combination of the training and validation set required around 2 gigabytes of memory. There are many ways to reduce the size of the training set and to speed up the training. We could compress the DFTs with Principal Component Analysis (PCA). We could also aggregate the DFTs over small windows before sending them to the DBN. Randomly choosing a subset of the frames in the dataset could also help. Another solution would be to augment the mini-batch size to optimize the time of training process. However, it is not clear how each of these solutions will affect the quality of the representation. This requires further investigation.

Reducing the training time of a single model would also help to solve the second issue, which is the hyper-parameter search. As mentioned in Section 4.1, there are many hyper-parameters to optimize. It is not clear how the optimal hyper-parameters vary depending on the input and the task. Current research on deep learning is investigating the matter, and some techniques to automatically adjust the hyper-parameters are being developed.

Another flaw of our model is that the features are extracted at the frame level only, so that our model cannot model long-term time dependencies. To better represent musical audio, we would need features that are able to capture the long-term time structure. Convolutional DBNs might provide a suitable model for time hierarchical representations [11].

6. CONCLUSION AND FUTURE WORK

In this paper, we have investigated the ability for DBNs to learn higher level features from audio spectra. We showed that these learned features can outperform MFCCs and carefully-tailored feature sets for autotagging. These results motivate further research with deep learning applied to MIR tasks.

In future work, we will continue investigating ways to reduce the training time of our models. Furthermore, we will learn features over a wider range of datasets and MIR tasks. We are interested, for example, in using the unsupervised DBN training approach to observe a large amount of unlabeled audio data. Finally, we will continue to investigate how we can take advantage of structure found at multiple timescales in music. To this end, a hierarchical convolutional DBN may be appropriate.

7. ACKNOWLEDGEMENTS

Special thanks to Guillaume Desjardins and Michael Mandel for their contribution to the code used in the experiments. Thanks also to Simon Lemieux, Pierre-Antoine Manzagol and James Bergstra for helpful discussions, and to the Theano development team for building such useful tools. This research is supported by grants from the Quebec Fund for Research in Nature and Technology (FQRNT) as and the Natural Sciences and Engineering Research Council of Canada (NSERC).

8. REFERENCES

[1] Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2:1–127, 2009.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.

[3] J. Bergstra. Algorithms for classifying recorded music by genre. Master's thesis, Université de Montréal, 2006.

[4] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl. Aggregate features and AdaBoost for music classification. *Machine Learning*, 65(2-3):473–484, 2006.

[5] T. Bertin-Mahieux, D. Eck, and M. Mandel. Automatic tagging of audio: The state-of-the-art. In *Machine Audition: Principles, Algorithms and Systems*. IGI Publishing, 2010.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[7] K. Chen, S. Gao, Y. Zhu, and Q. Sun. Music genres classification using text categorization method. *IEEE*, 2006.

[8] R. Grosse, R. Raina, H. Kwong, and A. Y. Ng. Shift-invariant sparse coding for audio classification. In *Proceedings of the Conference on Uncertainty in AI*, 2007.

[9] P. Hamel, S. Wood, and D. Eck. Automatic identification of instrument classes in polyphonic and poly-instrument audio. In *in Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR'09) , Kobe, Japan*, 2009.

[10] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[11] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[12] H. Lee, Y. Largman, P. Pham, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in Neural Information Processing Systems (NIPS) 22.*, 2009.

[13] T. Li and G. Tzanetakis. Factors in automatic musical genre classification. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.

[14] M. I. Mandel and D. P. W. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, pages 594–599, 2005.

[15] M. I. Mandel and D. P. W. Ellis. Multiple-instance learning for music information retrieval. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, pages 577–582, 2008.

[16] M. I. Mandel and D. P. W. Ellis. A web-based game for collecting music metadata. *Journal of New Music Research*, 37(2):151–165, 2008.

[17] P.-A. Manzagol, T. Bertin-Mahieux, and D. Eck. On the use of sparse time relative auditory codes for music. In *9th International Conference on Music Information Retrieval (ISMIR 2008)*, 2008.

[18] C. McKay and I. Fujinaga. Musical genre classification: is it worth pursuing and how can it be. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006.

[19] I. Mierswa and K. Morik. Automatic feature extraction for classifying audio data. *Machine Learning Journal*, 58:127–149, 2005.

[20] Y. Panagakis, C. Kotropoulos, and G.R. Arce. Music genre classification using locality preserving non-negative tensor factorization and sparse representations. In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, pages 249–254, 2009.

[21] G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. Technical report, IRCAM, 2004.

[22] J. Reed and C.-H. Lee. A study on attribute-based taxonomy for music information retrieval. In *Proc. of Int. Symposium on Music Information Retrieval*, 2007.

[23] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Trans. on Speech and Audio Processing*, 10(5):293–302, 2002.

[24] G. Tzanetakis, G. Essl, and P. Cook. Automatic musical genre classification of audio signals. In *Proceedings of the 2nd International Conference on Music Information Retrieval (ISMIR 2001), Bloomington, Indiana*, 2001.

[25] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

LEARNING SIMILARITY FROM COLLABORATIVE FILTERS

Brian McFee **Luke Barrington*** **Gert Lanckriet***
Computer Science and Engineering *Electrical and Computer Engineering
University of California, San Diego
bmcfee@cs.ucsd.edu lukeinusa@gmail.com gert@ece.ucsd.edu

ABSTRACT

Collaborative filtering methods (CF) exploit the wisdom of crowds to capture deeply structured similarities in musical objects, such as songs, artists or albums. When CF is available, it frequently outperforms content-based methods in recommendation tasks. However, songs in the so-called "long tail" cannot reap the benefits of collaborative filtering, and practitioners must rely on content-based methods. We propose a method for improving content-based recommendation in the long tail by learning an optimized similarity function from a sample of collaborative filtering data. Our experimental results demonstrate substantial improvements in accuracy by learning optimal similarity functions.

1. INTRODUCTION

"Collaborative filtering" (CF) is a popular method for multimedia recommendation applications in which data (*e.g.*, songs, artists, books or movies) are represented and compared in terms of the people who use them. Systems based on collaborative filtering exploit the "wisdom of crowds" to define similarity between items, which can then be used for recommendation. Indeed, collaborative filtering systems benefit from several attractive properties: CF explicitly represents individual users, and is therefore inherently personalized; data collection can be done passively, rather than requiring users to actively tag items; and CF data directly captures usage habits: exactly the quantity that recommendation engines strive to affect.

It is therefore not surprising that CF methods have become an active research topic in recent years, due in no small part to the recently concluded competition for the Netflix Prize [1]. Within the Music Information Retrieval (MIR) community, recent studies have shown that CF systems consistently outperform content-based methods for playlist generation [6] and tag prediction [15]. However, collaborative filtering suffers from the dreaded "cold start" problem: CF methods fail on items which have not yet been used, and are therefore unsuitable for recommendation in the "long tail". While this problem persists for all media (*e.g.*, movies, books, etc.), it is especially deadly in music, due to the relatively large number of unknown songs and artists in the world today. Netflix boasts 100,000 DVD titles [1], while Apple's iTunes store provides access to over 13 million songs [2].

Motivated by the cold-start problem, MIR researchers have worked steadily to improve *content-based* recommendation engines. Content-based systems operate solely on feature representations of music, eliminating the need for human intervention. While this approach naturally extends to long-tail data, the definition of *similarity* in these systems is frequently ad-hoc and not explicitly optimized for the specific task. As a result, it remains unclear if, or to what extent, content-based systems can capture relevant similarity information expressed by collaborative filtering.

In this paper, we pose the question: can we *learn* content-based similarity from a collaborative filter? Empirically, CF data provides a highly reliable means for determining similarity between musical objects. Our main contribution in this paper is a method for optimizing content-based similarity by learning from a collaborative filter.

The proposed method treats similarity learning as an information retrieval problem, where similarity is evaluated according to the ranked list of results in response to a query example, *e.g.*, a list of artists ordered by similarity to "The Beatles". Optimizing similarity for ranking requires more sophisticated machinery than is used in other methods, *e.g.*, genre classifiers. However, it does offer a few key advantages, which we believe are crucial for realistic music applications. First, there are no assumptions of transitivity or symmetry in the proposed method. As a result, "The Beatles" may be considered a relevant result for "Oasis", and not vice versa; this is not possible with other methods in the literature, *e.g.*, the embedding technique described in [21]. Second, CF data can be collected *passively* from users (*e.g.*, via scrobbles [16]) and directly captures their listening habits. Finally, optimizing similarity for ranking directly attacks the main quantity of interest, *i.e.*, the ordered list of retrieved items, rather than potentially irrelevant or overly coarse abstractions (*e.g.*, genre).

Our proposed method is quite general, and can improve similarities derived from semantic descriptions provided by humans or an auto-tagging engine. As we will demonstrate, even hand-crafted song annotations can be optimized to more accurately reflect and predict the similarity structure encoded by collaborative filtering data.

1.1 Related work

A significant amount of research has been devoted to the topic of musical similarity in the past decade. Ellis, et al. [9] evaluated similarity metrics derived from various data sources against human survey data. Similarly, Kim, et al. [15] evaluate several sources of artist similarity for a tag prediction task, and observe that methods based on collaborative filtering significantly out-perform acoustic or semantic similarity. However, neither of these works attempt to optimize similarity for a specific task.

Slaney, et al. [21] apply several learning algorithms to find similarity metrics over acoustic features which are optimized to cluster songs of the same artist, album, or that appear on the same blog. Our previous work [19] applies similar techniques to predict human survey data and optimally integrate multiple data sources. The method proposed here falls squarely within this line of work, but differs in that the metric is trained from collaborative filtering, and optimized for ranking performance, rather than classification or (comparatively scarce) human survey data.

There is a large body of work which treats collaborative filtering as a *matrix completion* problem (see, *e.g.*, [24]). In the matrix completion view, the goal is to perform user-centric recommendation by filling in missing entries of the users-by-content matrix, *i.e.*, recommending content to a user based on his or her specific preferences. Our application here is slightly different: rather than trying to complete the matrix, we interpret the collaborative filtering matrix as the ground truth, from which, similarity can be derived. Our goal is to train a content-based system to match similarities derived from CF data. We also stress that our proposed method is *not* a hybrid method: once the metric has been trained, collaborative filtering data is not necessary to compute similarities for unseen, long-tail songs.

2. LEARNING SIMILARITY

Our goal is to learn an optimal similarity function for songs, and as such, we must choose a family of similarity functions over which to optimize. Many families of similarity functions have been proposed in the MIR literature, such as distance between generative models of acoustic [3, 17] or semantic [5] descriptors, and playlist-based similarity [18].

Here, we opt for Euclidean distance between song representations. The primary reason for this choice is that Euclidean distance naturally lends itself to optimization by *metric learning* (see, *e.g.*, [19, 21]). In metric learning, each data point is described by a vector in $\mathbb{R}^d$, and the goal is to learn a linear projection matrix L such that distances after projection ($\|Li - Lj\|$) are small for "similar" pairs (i, j) and large for "dissimilar" pairs. Due to computational issues, optimization is performed not on L, but on a positive semi-definite [1] (PSD) matrix $W = L^\mathsf{T}L \succeq 0$. In the metric defined by W, distance between points (i, j) after projection is denoted by the quadratic form

$$d(i,j) = \|i-j\|_W^2 = (i-j)^\mathsf{T}W(i-j)$$
$$= (i-j)^\mathsf{T}L^\mathsf{T}L(i-j) = \|Li - Lj\|^2. \quad (1)$$

For the present application, we apply the *Metric Learning to Rank* (MLR) algorithm [20]. Here, we provide a brief overview of the algorithm.

[1] A positive semi-definite matrix W, denoted $W \succeq 0$ is square, symmetric, and has non-negative eigenvalues.

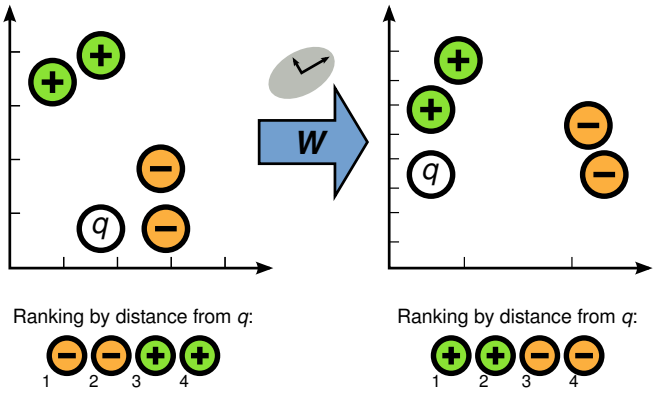


Figure 1. Metric Learning to Rank (MLR) learns a metric (W) so that a query song q is close to relevant results (+) and far from irrelevant results (-). Optimization is performed with respect to the rankings induced by distance from the query.

2.1 Metric learning to rank

Metric Learning to Rank (MLR) [20] is an extension of Structural SVM [13]. Structural SVM has been demonstrated to be an effective method for solving ranking problems in information retrieval systems [8], and the MLR algorithm extends the methodology to the *query-by-example* setting by learning a metric space, rather than a discriminant vector. Specifically, MLR learns a positive semi-definite matrix W such that rankings induced by learned distances are optimized according to a ranking loss measure, *e.g.*, ROC area (AUC) or precision-at-k. In this setting, "relevant" results should lie close in space to the query q, and "irrelevant" results should be pushed far away.

For a query song q, a natural ordering of the database $\mathcal{X}$ is obtained by sorting $x \in \mathcal{X}$ according to increasing distance from q under the metric defined by W (see Figure 1). The metric W is learned by solving a constrained optimization problem such that, for each training query q, a higher score is assigned to the "true" ranking y_q^* than to any other ranking $y \in \mathcal{Y}$ (the set of all rankings):

$$\langle W, \psi(q, y_q^*)\rangle \geq \langle W, \psi(q,y)\rangle + \Delta(y_q^*, y) - \xi_q. \quad (2)$$

Here, the "score" for a query-ranking pair (q, y) is computed by the Frobenius inner product:

$$\langle W, \psi(q,y)\rangle = \mathrm{tr}(W\psi(q,y)). \quad (3)$$

$\psi(q, y)$ is a matrix-valued feature map which encodes the query-ranking pair (q, y), and $\Delta(y_q^*, y)$ computes the loss incurred by predicting y instead of y_q^* for the query q (*e.g.*,

Algorithm 1 Metric Learning to Rank [20]

Input: data $\mathcal{X} = \{q_1, q_2, \ldots, q_n\} \subset \mathbb{R}^d$,
true rankings $y_1^*, y_2^*, \ldots y_n^*$,
slack trade-off $C \geq 0$

Output: $d \times d$ matrix $W \succeq 0$

$$\min_{W \succeq 0, \xi} \quad \mathrm{tr}(W) + C \cdot \frac{1}{n} \sum_{q \in \mathcal{X}} \xi_q$$

$$\text{s.t.} \quad \forall q \in \mathcal{X},\ \forall y \in \mathcal{Y} \setminus \{y_q^*\} :$$
$$\langle W, \psi(q, y_q^*) \rangle \geq \langle W, \psi(q, y) \rangle + \Delta(y_q^*, y) - \xi_q$$
$$\xi_q \geq 0$$

loss in AUC score), essentially playing the role of the "margin" between rankings y_q^* and y. Intuitively, the score for a true ranking y_q^* should exceed the score for any other y by at least the loss $\Delta(y_q^*, y)$. (In the present context, the "true" ranking is any one which places all relevant results before all irrelevant results.) To allow violations of margins during training, a slack variable $\xi_q \geq 0$ is introduced for each query.

MLR encodes query-ranking pairs (q, y) by the *partial order* feature [13]:

$$\psi(q, y) = \sum_{i \in \mathcal{X}_q^+} \sum_{j \in \mathcal{X}_q^-} y_{ij} \left(\frac{\phi(q, i) - \phi(q, j)}{|\mathcal{X}_q^+| \cdot |\mathcal{X}_q^-|} \right), \quad (4)$$

where $\mathcal{X}_q^+$ (resp. $\mathcal{X}_q^-$) is the set of relevant (resp. irrelevant) songs for q, the ranking y is encoded by

$$y_{ij} = \begin{cases} +1 & i \text{ before } j \text{ in } y \\ -1 & i \text{ after } j \end{cases},$$

and

$$\phi(q, i) = -(q - i)(q - i)^\mathsf{T} \quad (5)$$

captures the affinity between the query q and a single item i. Intuitively, ψ is constructed by adding the difference $\phi(q, i) - \phi(q, j)$ whenever y places (relevant) i before (irrelevant) j and subtracted otherwise. This choice of ψ therefore emphasizes directions in the feature space which are correlated with good rankings.

For a test query q', the predicted ranking y is that which achieves the highest score by W, *i.e.*, $\mathrm{argmax}_y \langle W, \psi(q', y) \rangle$. This can be found efficiently by sorting the corpus in descending order of $\langle W, \phi(q', x) \rangle$. Equation 5 defines ϕ so that when taking the inner product with W,

$$\begin{aligned} \langle W, \phi(q', x) \rangle &= -\mathrm{tr}\left(W(q' - x)(q' - x)^\mathsf{T} \right) \quad (6) \\ &= -(q' - x)^\mathsf{T} W (q' - x) = -\|q' - x\|_W^2, \end{aligned}$$

the result is the (negative, squared) distance between q' and x under the metric defined by W. Thus, decreasing $\langle W, \phi(q', x) \rangle$ corresponds to increasing distance from q'.

The MLR optimization problem is listed as Algorithm 1. As in support vector machines, the objective consists of two competing terms: $\mathrm{tr}(W)$ is a convex approximation to the rank of the learned metric, and $1/n \sum \xi_q$ measures the empirical (hinge) loss on the training set, and the two terms are balanced by a trade-off parameter C. Although the full problem includes a super-exponential number of constraints (one for each $y \in \mathcal{Y}$, for each q), [20] describes an efficient approximation algorithm based on cutting planes [14] which works well in practice.

3. DATA

Since our goal is to learn a content-based similarity metric for songs, it would seem logical to derive similarity from CF data relating users to songs. However, in practice, such matrices tend to exhibit high sparsity, which would lead to unstable similarity computations. We instead opt to derive similarity at the *artist* level, and then transfer similarity to the song level. Given a set of artists, and a collaborative filtering matrix over the artists, our experimental procedure is as follows:

1. extract *artist similarity* from the CF data,
2. transfer artist similarity to *song similarity*,
3. construct a feature representation for each song,
4. learn a metric W over song representations to predict *song similarities*, and
5. evaluate W by testing retrieval of similar songs in response to (unseen) test songs.

Steps 1 and 2 are described in Section 3.2, and step 3 is described throughout Section 3.3. Next, we describe the sources of our audio and collaborative filtering data.

3.1 Swat10k

Our experiments use the *Swat10k* dataset of 10,870 songs from 3,748 unique artists [22]. Each song has been weakly-labeled from a vocabulary of 1,053 tags from Pandora's Music Genome Project [2] that include multiple genres and acoustically objective descriptors.

3.2 Collaborative filtering

To define similarity between songs, we use the collaborative filtering (CF) data mined from Last.fm [3] by [7]. The raw collaborative filtering matrix consists of approximately 17.5 million user-song interactions over 359K users and 186K artists with MusicBrainz [4] identifiers (MBIDs).

We first filtered the CF matrix down to include only the Swat10k artists by matching MBIDs, resulting in a reduced CF matrix F:

$$F_{ui} = \begin{cases} 1 & \text{user } u \text{ listened to artist } i \\ 0 & \text{otherwise,} \end{cases} \quad (7)$$

of 356,026 users and 3,748 artists.

From the CF matrix, we define the similarity between artists i and j as the cosine-similarity between the column-vectors F_i and F_j:

$$S_{ij} = \frac{F_i^\mathsf{T} F_j}{\|F_i\| \cdot \|F_j\|}. \quad (8)$$

[2] http://www.pandora.com/mgp.shtml
[3] http://last.fm
[4] http://www.musicbrainz.org/

	Training	Validation	Test	Discard
# Artists	746	700	700	1602
# Songs	1842	1819	1862	5347
# Relevant	39.5	37.7	36.4	

Table 1. Statistics of the Swat10k data. "# Relevant" is the average size of the relevant set for each song.

Intuitively, S_{ij} counts the number of users shared between artists i and j, and normalizes by popularity.

To ensure stable similarity measurements, we discarded all artists from the set which had fewer than 100 users. This leaves 2,146 artists, which we split roughly into thirds for training, validation, and test sets. For each artist, we then define the set of "relevant" artists as the 10 closest training artists according to Equation 8 [5] .

Finally, we convert artist-level relevance to song-level relevance. For each song of an artist a, the relevant set is the union of the sets of songs from each of a's relevant artists. Table 1 summarizes the statistics of the data used in our experiments.

3.3 Features

For each song in our database, we construct three different feature representations: acoustic, auto-tags, and tags provided by human labelers.

3.3.1 Vector quantized MFCCs

Our representation of acoustic features is based upon vector-quantized Mel-Frequency Cepstral Coefficients (MFCCs), and consists of a 3-step process: feature extraction, vector quantization, and kernelization. The method described here is similar to that of [12], and is inspired by similar methods found in the computer vision literature [10].

First, for each song, we extract the first 13 MFCCs from 25ms half-overlapping windows. Each MFCC vector is augmented by appending the first and second instantaneous time derivatives, resulting in a sequence of 39-dimensional delta-MFCC (ΔMFCC) vectors for each song.

Using the songs which were discarded due to insufficient collaborative filtering data, we trained a codebook for use as a vector quantizer. We randomly selected 1000 songs from the discard set, and from each selected song, randomly sampled 1000 ΔMFCC vectors, for a total of 1 million codebook-training vectors. Each training vector v was z-scored, so that the i^{th} coordinate v_i becomes

$$v_i \mapsto \frac{v_i - \mu_i}{\sigma_i}, \tag{9}$$

where (μ_i, σ_i) are the sample mean and standard deviation of the i^{th} coordinate in the codebook-training set. We ran k-means with $k = 5000$ on the z-scored training vectors, using the implementation provided by [11]. The result is a set of 5000 codewords, each of which was subsequently "un"-z-scored by

$$v_i \mapsto \sigma_i v_i + \mu_i. \tag{10}$$

[5] For training artists, we assume self-similarity, so there are technically 11 relevant artists for each training artist.

With the codebook in hand, the ΔMFCC vectors for each song in the training, validation, and test splits were quantized by finding the closest (in Euclidean distance) codeword. Each song was summarized by a histogram over the 5000 codewords, corresponding to the frequency with which a codeword was selected as a quantizer in that song.

Finally, we constructed a χ^2-kernel over songs, so that the similarity between two codeword histograms u and v is calculated as [6]

$$k(u, v) = \exp\left(-\chi^2(u, v)\right) \tag{11}$$

$$\chi^2(u, v) = \sum_{i=1}^{5000} \frac{(u_i - v_i)^2}{u_i + v_i}. \tag{12}$$

(This kernel can itself be viewed as a soft vector quantizer, this time operating at the song-level rather than the feature-level.) Each song is represented by a vector in $\mathbb{R}^{1842}$, where the i^{th} dimension represents similarity to the i^{th} training song. We then compress these vectors by principal components analysis to 35 dimensions, which capture 95% of the variance in the training set.

3.3.2 Auto-tags

We can alternatively represent a song's acoustic information by using descriptive *semantics*. By learning from example songs that humans have labeled with tags, an "auto-tagger" (*e.g.*, [12, 23]) can automatically rate the relevance of these tags to new, unlabeled songs. The resulting "auto-tags" offer a concise description of the song, and semantic similarity between auto-tags has been shown to improve on content-based similarity derived from acoustics alone [5]. We use the auto-tagger described in [23] to label each song with a real-valued vector of 149 auto-tags: the i^{th} dimension of this vector corresponds to the probability that the i^{th} tag applies to the song, given the observed ΔMFCCs.

3.3.3 Human tags

Our third feature describes songs with "human tags" mined from the Music Genome Project by [22] that include descriptors of a song's genre, style, instrumentation, vocals and lyrics. Each song is represented by a 1,053-dimensional binary vector that is "weakly-labeled", meaning that a "1" implies that the tag is relevant but a "0" does not guarantee that the tag does not apply to the song. We consider these "human tags" to be "acoustically objective" as they have been applied by musicological experts and refer only to the acoustic content of the songs. They represent the ideal output that a content-based auto-tagger might achieve.

4. EXPERIMENTS

The MLR algorithm requires that a few parameters be set when training: not only the slack trade-off C, but also the choice of ranking measure to optimize. The implementation described in [20] supports several standard ranking measures: the area under the ROC curve (AUC), mean

[6] For the summation in Equation 12, we adopt the convention $0/0 = 0$.

Data source	AUC	MAP	MRR
MFCC	0.630	0.057	0.249
Optimized MFCC	0.719	0.081	0.275
Auto-tags	0.726	0.090	0.330
Optimized auto-tags	0.776	0.116	0.327
Human tags	0.770	0.187	0.540
Optimized human tags	0.939	0.420	0.636

Table 2. Ranking performance of each data source (MFCC, auto-tags, and human tags), before and after learning with MLR.

reciprocal rank (MRR), mean average precision (MAP), precision-at-k (P@k), and normalized discounted cumulative gain (NDCG); see [8] for a brief summary of these ranking measures. For P@k and NDCG, an additional parameter k must be set, which defines how many songs should be retrieved when evaluating the ranking.

For each data source described in Section 3, we trained metrics with all five variants of MLR. We swept over $C \in \{10^{-2}, 10^{-1}, \ldots, 10^{11}\}$, and for the P@k and NDCG variants, we also swept over $k \in \{2, 4, 8, \ldots, 256\}$. Performance was evaluated on the validation set, and the best-performing metric was then tested on the test set.

4.1 Embedding results

After learning W, we evaluate on the validation and test sets by computing for each query song q, the ranked list of training songs x ordered by increasing $\|q - x\|_W$. The resulting rankings are scored, and scores are averaged over all q to produce a single score for the learned metric. For comparison purposes, we also evaluate rankings derived from *native* metrics (*i.e.*, without learning W). The native metric for auto-tags is taken to be the Kullback-Leibler divergence between auto-tag distributions. For MFCC and human tags, we use standard Euclidean distance.

Table 4 displays some example playlists generated by the native and optimized MFCC spaces. At a high level, the learned metrics successfully de-noise the feature space to generate more cohesive playlists. Table 2 lists ranking performance for each data source, before and after optimization. In all but one case (auto-tag MRR), performance improves across all evaluation criteria. For each data source (MFCC, auto-tags, and human tags), we observe dramatic improvements in accuracy over the corresponding native similarity metric.

Quantitatively, the purely acoustic model improves in AUC score from 0.630 to 0.719. The optimized similarity performs comparably to native auto-tag similarity, but can be constructed entirely from passive data (as opposed to the actively collected data necessary for building auto-tag models). Similarly, optimizing auto-tags improves AUC from 0.726 to 0.776, which is comparable to the native performance of human tags. Finally, optimizing human tags improves AUC substantially, from 0.770 to 0.939. This indicates that even when annotations are hand-crafted by experts, recommendation may still be greatly improved by using an appropriate model of the tag vocabulary.

Top tags		Bottom tags	
1.	LATIN	1044.	TWO-STEP STYLE
2.	A REGGAE FEEL	1045.	UNUSUAL VOCAL SOUNDS
3.	REGGAE	1046.	UPBEAT LYRICS
4.	CHRISTIAN	1047.	CALL-AND-RESPONSE VOCALS
5.	NEW-AGE	1048.	ELECTRIC PIANOS
6.	ROCK ON THE RANGE RADIO	1049.	MODAL HARMONIES
7.	WAKARUSA RADIO	1050.	TONAL HARMONIES
8.	SASQUATCH RADIO	1051.	VOCAL COUNTERPOINT
9.	CMJ MUSIC MARATHON	1052.	VOCAL SAMPLES
10.	REGGAE / CARIBBEAN	1053.	WESTERN SWING

Table 3. The top and bottom 10 tags learned by MLR, ordered by weight. 85 tags receive 0 weight.

4.2 Learning tag weights

Given the substantial improvement observed by optimizing human tags, one may wonder what conclusions can be drawn from the learned metric. In particular, since W can be interpreted as "translation matrix" or vocabulary model, it is natural to ask which tags define the similarity space, and which tags are redundant or non-informative.

Because W contains both positive and negative entries, it is not immediately clear how to interpret a full matrix W in terms of tags. However, placing further restrictions on the form of W can ease interpretability (at the expense of model flexibility). We repeated the "human tags" experiment with a modification of Algorithm 1 that restricts W to be diagonal and non-negative. In the restricted model, the i^{th} element of the diagonal W_{ii} can be interpreted as a weight for the i^{th} tag. The diagonal metric achieves AUC of 0.875 (compared to 0.776 native and 0.939 for a full W).

Table 3 lists the top and bottom 10 tags, ordered by weights W_{ii}. Several interesting observations can be made here: all of the top tags refer either to genre (*e.g.*, LATIN, REGGAE) or streaming radio identifiers (*e.g.*, WAKARUSA, CMJ). This corroborates previous studies which indicate that grouping music by social cues, such as radio playlists or blogs, can assist recommendation [4]. By contrast, the bottom tags are primarily musicological terms (*e.g.*, VOCAL COUNTERPOINT) which apparently convey little useful information for recommendation.

This view of MLR as a supervised learning procedure for vocabulary models suggests comparison to standard, unsupervised techniques, such as TF-IDF with cosine similarity. It turns out that for this data set, using TF-IDF weighting results a *decrease* in AUC from 0.770 to 0.724! From this, we can conclude that it is suboptimal to rely on the natural statistics of tags to define similarity.

5. CONCLUSION

We have proposed a method for improving content-based similarity by learning from a sample of collaborative filtering data. The proposed method learns an optimal transformation of features to reproduce high-quality CF similarity, and can be used to improve the quality of recommendation in the long tail. If songs are described by semantic tags, our method reveals which tags play the most important role in defining an optimal similarity metric. By revealing the most important tags for predicting CF similarity, our method may also be useful for guiding the development of discovery interfaces and automatic tagging algorithms.

	Query song	Native space playlist	Optimized space playlist
MFCC	Chick Corea Elektric Band - Beneath the Mask	Katalyst - Break Up	► Michael Brecker - Two Blocks From The Edge
		Stevie Wonder - Superstition	► Charlie Parker - Wee
		James Brown - Soul Power	Coleman Hawkins - There Is No Greater Love
		Tina Turner - What's Love Got To Do With It	► Miles Davis - Walkin'
		The Whispers - And The Beat Goes On	Clifford Brown - Love Is A Many Splendored Thing
Auto-tags	White Zombie - Electric Head (Pt. 2.) (Remix)	► Sepultura - Apes Of God	► Sepultura - Apes Of God
		Arctic Monkeys - A Certain Romance	► Metallica - Nothing Else Matters (Live)
		Secret Machines - Lightning Blue Eyes	Secret Machines - Lightning Blue Eyes
		Green Day - Longview (Live)	The Warlocks - Gypsy Nightmare
		Perry Farrell - Kinky	Mastodon - Crystal Skull
Human tags	Aaliyah - Miss You	► Ginuwine - In Those Jeans	► Monica - Don't Take It Personal
		► Monica - Don't Take It Personal	► Ginuwine - In Those Jeans
		► Ashanti - Foolish	► Ashanti - Foolish
		Foo Fighters - DOA	► Ne-Yo - Go On Girl
		Say Hi To Your Mom - Northwestern Girls	Jodeci - Freak N You

Table 4. Example playlists in native and optimized MFCC, auto-tag, and human tag spaces. Playlists are generated by finding the five nearest neighbors of the query; relevant results are indicated by ►.

6. REFERENCES

[1] Netflix press release, 2009. http://netflix.mediaroom.com/index.php?s=43&item=307.

[2] Apple itunes, 2010. http://www.apple.com/itunes.

[3] Jean-Julien Aucouturier and François Pachet. Music similarity measures: What's the use? In *Inernational Symposium on Music Information Retrieval (ISMIR2002)*, pages 157–163, 2002.

[4] Claudio Baccigalupo, Justin Donaldson, and Enric Plaza. Uncovering affinity of artists to multiple genres from social behaviour data. In *International Symposium on Music Information Retrieval (ISMIR2008)*, September 2008.

[5] Luke Barrington, Antoni Chan, Douglas Turnbull, and Gert Lanckriet. Audio information retrieval using semantic similarity. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007.

[6] Luke Barrington, Reid Oda, and Gert Lanckriet. Smarter than genius? Human evaluation of music recommender systems. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[7] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[8] Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. Structured learning for non-smooth ranking losses. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 88–96, New York, NY, USA, 2008. ACM.

[9] D. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *Proeedings of the International Symposium on Music Information Retrieval (ISMIR)*, pages 170–177, October 2002.

[10] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 2, 2005.

[11] Peter Gehler. MPIKmeans, 2007. http://mloss.org/software/view/48/.

[12] M. Hoffman, D. Blei, and P. Cook. Easy as CBA: A simple probabilistic model for tagging music. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[13] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384, New York, NY, USA, 2005. ACM.

[14] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27–59, 2009.

[15] Joon Hee Kim, Brian Tomasik, and Douglas Turnbull. Using artist similarity to propagate semantic information. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[16] Last.FM, January 2009. http://www.last.fm/.

[17] B. Logan. Music recommendation from song sets. In *International Symposium on Music Information Retrieval (ISMIR2004)*, 2004.

[18] François Maillet, Douglas Eck, Guillaume Desjardins, and Paul Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[19] Brian McFee and Gert Lanckriet. Heterogeneous embeddding for subjective artist similarity. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[20] Brian McFee and Gert Lanckriet. Metric learning to rank. In *Proceedings of the 27th annual International Conference on Machine Learning (ICML)*, 2010.

[21] M. Slaney, K. Weinberger, and W. White. Learning a metric for music similarity. In *International Symposium on Music Information Retrieval (ISMIR2008)*, pages 313–318, September 2008.

[22] D. Tingle, Y. Kim, and D. Turnbull. Exploring automatic music annotation with "acoustically-objective" tags. In *IEEE International Conference on Multimedia Information Retrieval (MIR)*, 2010.

[23] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE TASLP*, 16(2):467–476, Feb. 2008.

[24] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):435–447, 2008.

Characterization and Melodic Similarity of A Cappella Flamenco *Cantes*

Joaquín Mora
Department of Evolutive and Educational Psychology
University of Seville
mora@us.es

Francisco Gómez
Applied Mathematics Department, School of Computer Science
Polytechnic University of Madrid
fmartin@eui.upm.es

Emilia Gómez
Music Technology Group,
Universitat Pompeu Fabra
emilia.gomez@upf.edu

Francisco Escobar-Borrego
Department of Audiovisual Communication, Publicity and Literature
University of Seville
fescobar@us.es

José Miguel Díaz-Báñez
Department of Applied Mathematics II,
University of Seville
dbanez@us.es

ABSTRACT

This paper intends to research on the link between musical similarity and style and sub-style (variant) classification in the context of flamenco a cappella singing styles. Given the limitation of standard computational models for melodic characterization and similarity computation in this particular context, we have proposed a specific set of melodic features adapted to flamenco singing styles. In order to evaluate them, we have gathered a collection of music recordings from the most representative singers and have manually extracted those proposed features. Based on those features, we have defined a similarity measure between two performances and have validated their usefulness in differentiating several styles and variants. The main conclusion of this work is the need to incorporate specific musical features to the design of similarity measures for flamenco music so that flamenco-adapted MIR systems can be developed.

1. INTRODUCTION

There is a wealth of literature on music research that focuses on the understanding of music similarity from different viewpoints as well as the use of similarity distances to cluster different pieces according to artist, genre or mood. Over the past few years, the Music Information Retrieval (MIR) community has been exploring several ways to measure the similarity between two musical pieces. They are often based on comparing musical excerpts in audio format and computing the distance between a representative set of their musical features (e.g., instrumentation, rhythmic patterns or harmonic progressions). This way, current systems can, for instance, locate different versions of the same song with a high accuracy rate [9].

Alternative approaches are based on comparing contextual information from the considered pieces or artists (e.g., influences, temporal coincidences, geographical location), which is usually extracted from the web. A combination of these two approaches seems to be the most adequate solution [2], but there is still a glass ceiling on current systems. This seems to be owing to the fact that there are still fundamental musical features to be considered or fully incorporated into existing models [10].

On the other hand, research on music similarity has mainly focused on the analysis of music from the so-called "Western tradition", although there is an increasing interest in analyzing music from different traditions, and some recent MIR studies are devoted to this issue [3].

In this paper we study the relationship between music similarity and style definition in the context of a group of flamenco a cappella singing styles. Because of its oral transmission, formal classification and musical analysis of flamenco present considerable difficulties. Classification of styles is usually carried out by flamenco experts and rests upon the analysis of the oral corpus of flamenco music. Flamenco experts have used several criteria in their classification, some of which, unfortunately, are not fully explicit and clear-cut. Moreover, there exists considerable controversy among flamenco experts and at present there is a lack of a unified, unequivocal classification of a cappella *cantes*.

Our previous research has focused on analyzing the relationship between music similarity and style definition from different perspectives, such as computational models for melodic similarity [3] and human ratings of rhythmic patterns [6]. In Cabrera et al. [3], we applied standard melodic similarity measures to a music collection of a cappella flamenco music. Extraction of melodic contours, although arduous, produced faithful descriptions of *cantes*. However, when applied some standard algorithms for comparing melodic contours (i.e. correlation between pitch and interval histograms and edit distance), we obtained rather poor results. Although global style classification seemed to work to some extent, flamenco experts found that for variant classification of some styles those algorithms failed to distinguish subtle, specific nuances of the *cantes*. We applied some clustering techniques and it turned out that c*antes* that by musical reasons should be scattered were unexpectedly clustered together. Thus, we concluded that further research had to be carried out. It soon became apparent that two tasks, at least, had to be accomplished: first, to musically formalize the specific and subtle features of a cappella *cantes*; second, to refine our computational models according to those features.

This paper intends to research on the link between musical similarity and style classification and sub-style

(variant) definition in the context of flamenco a cappella singing styles. The main contributions are (1) Identify relevant and specific features that characterize a given performance in the context of its style. Here, we consider that each style is characterized by a prototypical melodic contour. The features will then account for variations within this contour. (2) Define a similarity measure based on the identified features and provide an automatic method of clustering and classification. (3) Evaluate the results with a music collection of recordings from the most representative performers and contrast them with existing theories for the definition of styles and variants.

This paper is organized as follows. In the next two sections we analyze the characteristics of flamenco singing and a cappella *cantes*. Next we give an overview of the *cantes* to be analyzed and we describe the music collection used in our study. We then present the set of features that describes the two chosen *cante*s, namely, *deblas* and *martinetes*. The following section deals with the similarity distance between *cantes*. A conclusion section summarizes the main findings and presents some proposals for future research.

2. FLAMENCO SINGING

We now describe the main features that characterize flamenco singing and differentiate it from other styles:

- **Instability of pitch**. In general, notes are not clearly attacked. Pitch glides or *portamenti* are very common.
- **Sudden changes in volume** (loudness).
- **Short pitch range *or tessitura.*** It is normally limited to a major sixth interval and characterized by the insistence on a note and those contiguous.
- **Intelligibility of voices.** Since lyrics are important in flamenco, there is a strong preference for intelligibility over range or timbre. Contralto and baritone voices are very common.
- **Timbre**. Timbre characteristics of flamenco singers depend on the period in which it was performed. As relevant timbre aspects, we can mention breathiness in the voice and the absence of high frequency (singer) formant, which is characteristic of classical singing styles.

From a musical point of view, a cappella *cantes* retain the following properties.

- **Conjunct degrees.** Melodic movement mostly occurs by conjunct degrees.
- **Scales.** Certain scales such as the dominant Phrygian mode (with a major tonic) and Ionian mode (E-F-G#-A-B-C-D) are predominant.
- **Ornamentation**. There is also a high degree of complex ornamentation, melismas being one of the most significant devices of expressivity.
- **Microtonality.** The use of intervals smaller than the equal-tempered semitones of Western classical music is frequent.
- **Enharmonic scales.** Microtonal interval differences between enharmonic notes.

Classification of a cappella *cantes* is subject to many difficulties. Two *cantes* belonging to the same style may sound very different to an unaccustomed ear. In general, underlying each *cante* there is a melodic skeleton. Donnier [4] called it the "*cante's* melodic gene". The singer fills this melodic skeleton by using different kind of melismas, ornamentation and other expressive resources. A flamenco aficionado recognizes two versions as to be the same *cante* because certain notes appear in certain order. What happens between two of those notes does not matter regarding style classification, but does matter for assessing a performance or the piece itself. Aficionado's ears recognize the wheat from the chaff when listening and appreciate a particular performance in terms of the quality of the melodic filling, among other features.

In order to put the reader in the position of understanding this point, in Figures 1 and 2 we show a transcription of two versions of the same *cante* to Western musical notation. In this respect, Western notation has been found limited for this kind of music (e.g. Donnier [4] proposed the use of plainchant neumes).

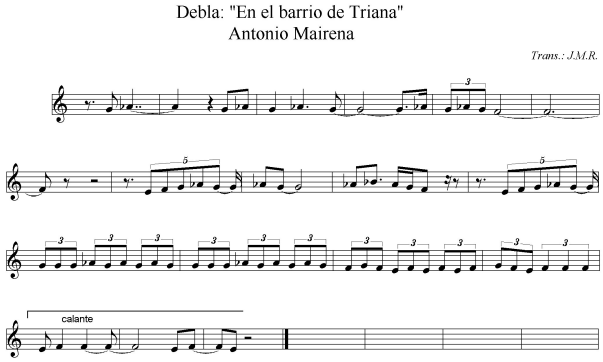


Figure 1: Manual transcription of a *cante* (debla).

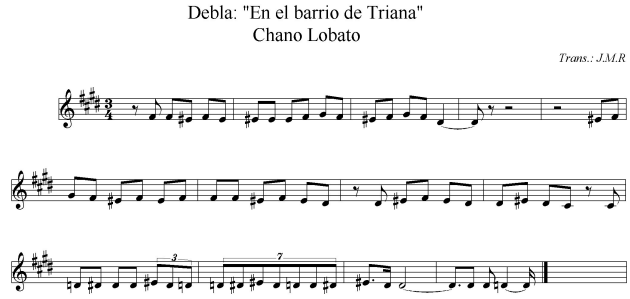


Figure 2: Transcription of another version of the previous *cante*.

3. THE STYLES OF TONÁS

As defined earlier, a cappella *cantes* are songs without instrumentation, or in some cases with some percussion *(*in the flamenco argon also *cantes a palo seco)*. An important group of such *cantes* is composed of *tonás*, style that generically include *martinetes, deblas, saetas, tonás* and *carceleras*. Since in flamenco music the word *toná* refers to both the style and one of its variants, we will refer to "*tonás* style" as the whole style and *toná* as the substyle or *cante.*

These *cantes* are played in free rhythm (sometimes the pattern of *seguiriya* is used as rhythmic accompaniment). Each singer chooses his or her own reference pitch. Modality, depending on the particular style, may be either the major mode or the Andalusian mode (Phrygian mode with a tonic major chord), though frequently both modes alternate within the same song [5]. Lyrics of these songs range widely. Lefranc [8] carried out a classification of the *tonás* style mainly based on lyrics. Blas Vega [1] also studied the *tonás* style from a historical standpoint. A *toná* typically possesses verses formed by words either of three or four syllables, or eight syllables, where the second and the forth verses may have assonant rhyme; often the final verse finishes with an imperfect tercet (a tercet is a group of three lines of a verse; an imperfect tercet is an off-rhyme tercet).

The *toná cante* is normally sung in a four-verse form of eight syllables each. Rhyme is assonant on even verses. The *toná cante* appears in several modes, the Phrygian mode being the most prevailing one. Ionian or Aeolian modes can also be found as well as alternation of two modes. Ornamentation is very complex and profuse; no strict tempo is followed at all.

4. MUSIC COLLECTION

To start with, we gathered a set of 365 pieces belonging to the *tonás* style. Somehow, these *cantes* have scarcely been recorded compared to other styles. Their ritual mood and lyricism might be a plausible reason for that shortage of recordings. In spite of that, we spared no effort to gather as many recordings as possible from all feasible sources (private collections, libraries, historical recordings, several institutions, etc.). We may safely state that our collection is quite representative of this type of *cantes*.

After the analysis phase of the corpus, we decided to focus on two styles, *deblas* and *martinetes*, because: (1) Both styles are central to flamenco music; (2) Contrary to other *cantes*, we had information about singers, geographical locations, singing schools, dates, etc., which allows us to have a complete, in-depth characterization of them from a more general standpoint than just the musical one; (3) In general, recordings had acceptable quality to carry out this research and the future ones, which include, for instance, automatic feature extraction from audio files; (4) There was a high number of recordings, 72 songs in total, where 16 were *deblas* and 56 *martinetes*; (5) Apart from the number, there was enough variability in the sample for a proper evaluation of our methods.

The specific feature set of *deblas* and *martinetes*[1], to be described below, were obtained after a thorough study. First, we opened an analysis phase to identify which musical features were relevant to the characterization of the chosen *cantes*. Preliminary analysis produced too many variables or just variables with little explanatory power. Next, in search of the least complex yet meaningful description of *cantes*, we removed several variables. Most of the features identified were related to melody and form. During the analysis, in our mind it also underlay the idea of capturing some of the features used by flamenco experts and aficionados to recognize the different styles. After deciding on the final feature set, we manually extracted it for the different performances. A working group of flamenco experts agreed on the proposed feature set and annotations. The next two sections describe the musical features in detail.

[1] The music collection studied in this paper can be found at http://mtg.upf.edu/research/projects/cofla Please, contact the authors.

5. MUSICAL FEATURES

5.1. Deblas

The *debla* is a song from the style of *tonás*. In general, it is marked by its great melismatic ornamentation, more abrupt than the other songs from this style, which characterizes its melody. Deblas are characterized by a particular melodic contour.

The musical features that characterize the different variants within the *debla* style can be summarized as follows:

1. **Beginning by the word *¡Ay!***: *¡Ay!* is an interjection expressing pain, which is quite idiosyncratic to flamenco music. Values of the variable: Yes and no.
2. **Linking of *¡Ay!* to the text.** That initial *¡Ay!* may be linked to the text or just be separated from it. Values of the variable: Yes and no.
3. **Initial note.** It refers to the first note of the tercet. Normally, it is the sixth degree of the scale (VI), but the fifth degree (V) can also appear. Values of the variable are 5 and 6.
4. **Direction of melody movement in the first hemistich** (a hemistich is half of a line of a verse.) The direction can be descending (fast appoggiatura in V, then the progression VI-IV), symmetric (III-VI-IV) or ascending. Values here are *D*, *S* and *A*.
5. **Repetition of the first hemistich.** That repetition may be of the whole hemistich or just a part of it. Values of the variable: Yes or no.
6. **Caesura.** The caesura is a pause that breaks up a verse into two hemistiches. Values of the variable: Yes and no.
7. **Direction of melody movement in the second hemistich,** defined as in the first hemistich. Values of the variables are *D*, *S* and *A*.
8. **Highest degree in the second hemistich.** It is the highest degree of the scale reached in the second hemistich. Usually, the seventh degree is reached, but fifth and sixth degrees may also appear. Values of the variable are 5, 6 and 7.
9. **Frequency of the highest degree in the second hemistich.** The commonest melodic line to reach the highest degree of the scale consists of the concatenation of two *torculus*. A *torculus* is a neume signifying three notes, the second higher than the others. The value of this variable indicates how many times this neume is repeated in the second hemistich.

10. **Duration.** Although the duration is measured in ms, our intention was to classify the *cantes* into three categories, fast, regular and slow. To do so, we first computed the average μ and the standard deviation σ of the durations of all the *cantes* in the same style. Then, fast *cantes* are those whose duration is less than μ-σ, regular *cantes* have their duration in the interval [μ-σ, μ+σ], and slow *cantes* have durations greater than μ+σ. Values of this variable are *F, R* and *S*.

5.2. Martinetes

The *martinete* is also considered a variant of the *toná*. It differs from *deblas* in the theme of its lyrics and in its melodic model, which always finishes in the major mode. *Debla*'s mood is usually sad, played without guitar accompaniment, like the other *tonás*. *Martinetes*, however, are usually accompanied by the sound of a hammer struck against an anvil.

There are three clear variants of *martinetes.* The first one, to be called *martinete* 1, has no introduction, whereas the second one, to be called *martinete* 2, starts with a couple of verses from a *toná.* The third one, to be called *martinete* 3, is a concatenation of a *toná* and some of the previous variants of *martinetes*; the *toná* of *martinete* 2 and 3 is called *toná* incipit. A trait of *martinete* 3 is the singing of several poems, all having the same theme as in medieval romances. Because *martinete 3* is a combination of *toná* and *martinetes 1* and *2,* we removed it from our current study, as we just sought characterizing the most fundamental styles.

Next, musical features of *martinete 1* are presented.

1. **Repetition of the first hemistich.** As for *deblas,* repetition may be complete or partial. Values of the variable: Yes and no.
2. **Clivis/flexa** (a neume of two notes, the second lower than the first one) **at the end of the first hemistich.** Normally, fall IV-III or IV-IIIb are found. The commonest ending for a tercet is the fourth degree, whose sound is sustained until reaching the caesura. Some singers like to end on III or IIIb. Values of the variable are: Yes and no.
3. **Highest degree in both hemistiches.** The customary practice is to reach the fourth degree; some singers reach the fifth degree. Values of the variable are 4 and 5.
4. **Frequency of the highest degree in the second hemistich.** The melodic line is formed by a *torculus,* a three-note neume, III-IV-III in this case. This variable stores the number of repetitions of this neume.
5. **Final note of the second hemistich.** The second hemistich of *martinete 1* is ended by falling on the second degree. Sometimes, the second degree is flattened, which produces Phrygian echoes in the cadence. This variable takes two values, *1* when the final note is the tonic and *2* when the final note is II.
6. **Duration.** This variable is defined as in the case of *deblas* (that is, in terms of μ and σ). Values of this variable are *F, R* and *S*.

As for *martinete 2* we have the following features.

1. **Highest degree in both hemistiches.** In this case the customary practice is to reach the sixth degree; in some cases singers just reach the fourth or fifth degrees. Values of the variable are 4, 5 and 6.
2. **Frequency of the highest degree in the second hemistich.** In this case the neume is also a *torculus*. This variable stores the number of repetitions of this neume.
3. **Symmetry of the highest degree in the second hemistich.** The second hemistich of a *martinete 2* is rich in melismas. This feature describes the distribution of the melismas around the highest reached degree, which is usually the sixth degree. Melismas can occur before and after reaching the highest degree (symmetric distribution), only before the highest degree (left asymmetry) or only after the highest degree (right asymmetry).
4. **Duration.** This variable is defined as in the previous cases. Values of this variable are *F, R* and *S*.

6. SIMILARITY COMPUTATION

6.1 Inter-Style Similarity

Carrying out the preceding analysis allowed us to extract a set of musical features to be used in the definition of melodic similarity between *cantes*. As a matter of fact, just using features very peculiar to a given style would distort the analysis, as their discriminating power would be very high. Our intention was to select a set of a few features capable of discriminating between different cantes. We removed those variables that only made sense within one style, gathering the following final set:

1. **Mode of type of scale**: Ionian (major mode), dominant Phrygian (Phrygian mode with a major third) or bimodal (alternation of both modes). This variable makes sense to classify the different styles.
2. **Direction of melody movement in the first hemistich**.
3. **Symmetry of the highest degree in the second hemistich.**
4. **Clivis (or flexa) at the end of the first hemistich.**
5. **Repetition of the first hemistich.**
6. **Initial note.**
7. **Final note of the second hemistich.**
8. **Highest degree in both hemistiches.**
9. **Frequency of the highest degree in the second hemistich with respect to a cante.** This variable has been normalized as follows. Let μ be the average of the frequencies of the all *cantes*, and σ its standard deviation. The frequency takes three values, 0 (seldom), if the frequency is less than μ-σ; 1 (normal) if it is within the interval [μ-σ, μ+σ]; and 2 (often), when it is greater than μ+σ.
10. **Frequency of the highest degree in the second hemistich with respect to the whole corpus.** In this

case the mean and the standard deviation are taken with respect to all pieces in the corpus.

11. **Duration with respect to the piece.** This variable is defined as above, taking three values fast, regular and slow.
12. **Duration with respect to the whole corpus**. Now the mean and the standard deviation are computed for the whole corpus.

Note that some of these variables do not appear as specific features in some cante. We removed those variables that only accounted for one cante and tried to keep the smallest set of explanatory variables. Moreover, we performed a principal component analysis (SPSS 15 were used for this). As a consequence, two variables were discarded: *Clivis* (or *flexa*) at the end of the first hemistich and repetition of the first hemistich. The number of final dimensions turned out to be 3 and the Cronbach's alpha equalled 0.983, a very good value, indeed.

The distance we used to measure the similarity between two cantes was the simplest one could think of: the Euclidean distance between features vectors. Our intention was to test how powerful the musical features would be. The Euclidean is just a geometrical distance and does not reflect perceptual distance whatsoever. However, because of the robustness and power of the musical features, results were good.

We used phylogenetic trees [7] to better visualize the distance matrix. For the actual computing, we used the tool SplitsTree [7]. SplitsTree computes a tree (or more generally a graph) with the property that the distance in the drawing between any two nodes reflects as closely as possible the true distance between the corresponding two pieces in the distance matrix. In general, clustering and other properties are easier to visualize with phylogenetic trees.

6.2 Intra-Style Similarity

On a second step, we analyzed similarities between performances within each particular style. In this case, we considered all the variables presented below for each particular style (*martinete1*, *martinete2* or *debla*). We computed the corresponding distance matrices and generated their phylogenetic trees. As before, we used the Euclidean distance. Again, our intention was to evaluate how well the defined features characterize these *cantes*.

7. RESULTS

7.1 Inter-Style Similarity

Regarding style classification, Figure 2 displays the phylogenetic tree obtained for the whole corpus under study. Label Dx stands for *debla*, label M1 stands for *martinete 1* and label M2 for *martinete 2.*

As it can be noticed, discrimination among *cantes* is very good. There are three clearly separated clusters, each corresponding to a *cante.*

In addition, we have computed for each *cante* its closest neighbour and checked if both belong to the same style. That was the situation for 88.6% of the cases. There was a striking case in one particular *debla. Debla D2-14* whose three nearest neighbours turned out to be of class *martinete 2*. That *debla* is one of the oldest known songs, according to our musicological knowledge. Meaningfully enough, the closest *martinetes* are also old, which suggests a possible common origin or an evolution of *martinete 2* from *debla* (*martinete 2* seems to be a mixture of previous *cantes*). Those *martinetes* are *M2-66, M2-68* and *M2-65*; see Figure 3 at the end of this paper.

We also observed that some distance values were equal to zero for the same style; this will be discussed later in the conclusions section. Finally, precision and recall, two common performance measures, curves are provided in Figure 4.

7.2 Intra-Style Similarity

We did not expect a very fine clustering in the intra-style case, as the musical features account for general properties rather than for specific properties. In general, there was one big cluster, often with several *cantes* having the same distances. Overall, this was in general coherent to the expert's knowledge on the musical relation among performers.

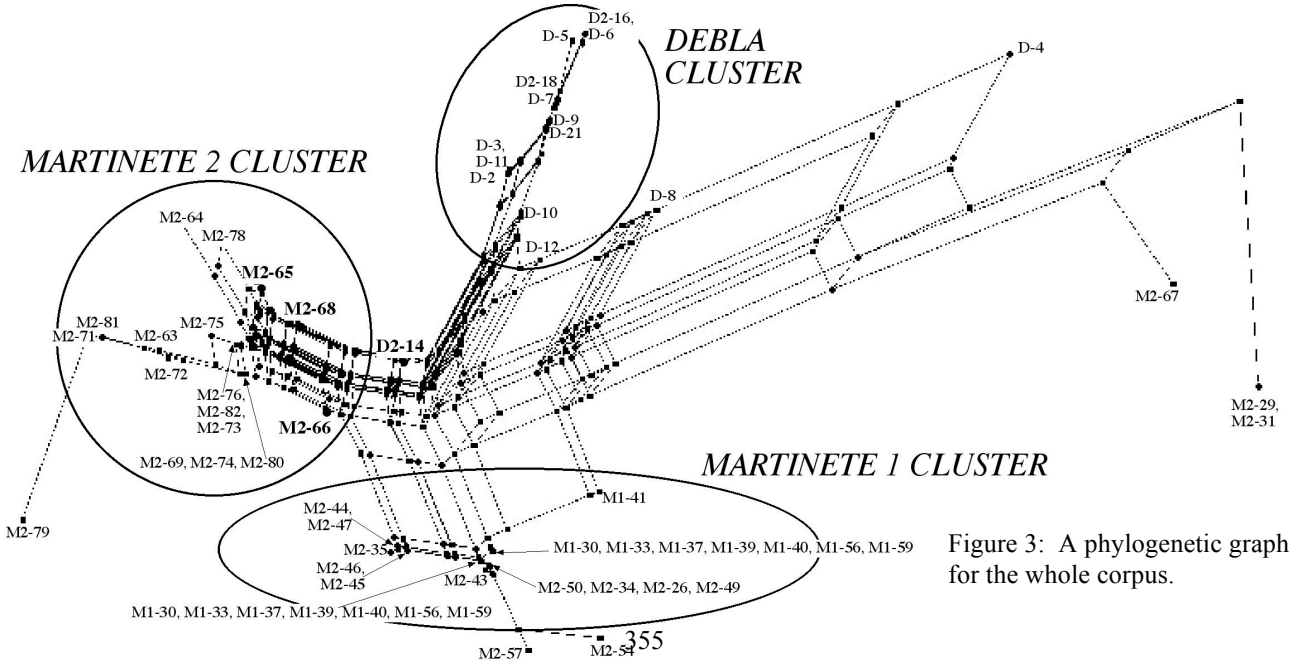


Figure 3: A phylogenetic graph for the whole corpus.

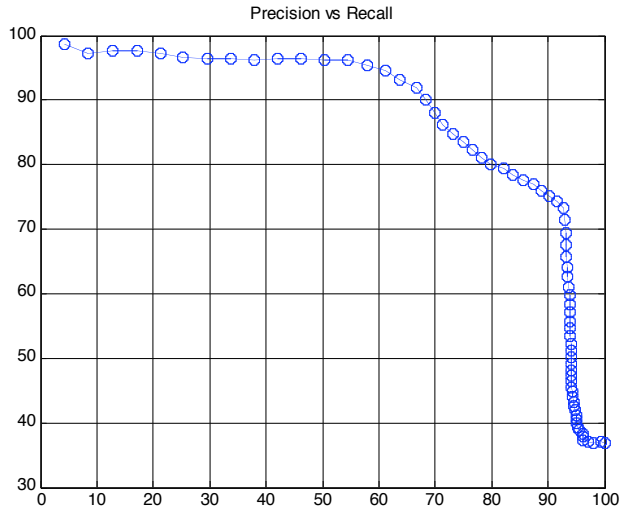


Figure 4: Precision versus recall for style classification.

8. CONCLUSIONS

There are many automatic systems that compute similarity between pieces of music by measuring the distance between some musical features. One common feature is the melodic contour. Therefore, the problem of measuring musical similarity is transformed into the problem of computing distances between melodic contours. For certain type of music that approach may be appropriate and even produce good results. It is not the case for flamenco a cappella flamenco cantes, a kind of music characterized by very special features. In our previous study [3] we followed that approach and understood its drawbacks. Reducing melodic similarity to computing the distance between melodic contours leaves significant variables aside, which results in a limited similarity measure.

In the present study we took another approach. We identified a set of musical features as basis for our measure. Those musical features are related to melodic properties, duration and form and reflect in general high-level characteristics of the *cantes*. We also found some limitations. As the reader may have noticed, many *cantes* give the same feature vector and, therefore, in the phylogenetic trees they all appear on the same node. Why is that so? A moment's thought will reveal that this situation is not as surprising as it might seem at first glance. As we mentioned, the musical features account for high-level characteristics. Two *cantes* may be different at the fine detail level and yet have the same musical features. That is the hardness of dealing with flamenco *cantes*.

It is clear that in order to obtain good similarity measures musical knowledge of the particular type of music must be incorporated into the model. On the other hand, since we know this is not enough, fine detail information such as melodic contours should be incorporated, too. Thus, there should be a tradeoff between both approaches. That is the research we are presently carrying out.

The identification of the musical features of a cappella *cantes* is one of the main contributions of this paper. They have not been described before and our experiments have proved they work quite well for inter-style classification. Our approach can be easily adapted to build a classification system based on machine learning. For the case of intra-style classification, more musical features should be identified. Our goal was to distinguish general musical features rather than specific features. This situation shows that special needs require special treatments and that inter-style and intra-style classification needs a different set of musical features.

At present all the work put forward here has been done in a manual way. First, we wanted to check that the extraction of the musical features were correct and, furthermore, gave a good measure of similarity. We are currently working on building an automatic system based on our measure. This includes the automatic extraction of the musical features from music recordings through an automatic transcription procedure.

9. REFERENCES

[1] Blas-Vega, J *Historia de las tonás*, ed. Guadalhorce, Málaga, 1967.

[2] Celma, O. Music Recommendation and Discovery in the Long Tail. PhD thesis, Universitat Pompeu Fabra, 2008.

[3] Cabrera, J.J., Díaz-Báñez, J.M., Escobar, F.J., Gómez, E., Gómez, F., Mora, J. Comparative Melodic Analysis of A Cappella Flamenco Cantes. In Conference on Interdisciplinary Musicology. Thessaloniki, Greece, 2008.

[4] Donnier, P. Flamenco: elementos para la transcripción del cante y la guitarra, *Proceedings of the IIIrd Congress of the Spanish Ethnomusicology Society*, 1997.

[5] Fernández, L. Flamenco Music Theory. Acordes Concert. 2004.

[6] Guastavino, C., Gómez, F., Toussaint, G., Marandola, F., Gómez, E. Measuring Similarity between Flamenco Rhythmic Patterns. *Journal of New Music Research.* 38(2), 129-13, 2009.

[7] Huson, D. H. SplitsTree: Analyzing and visualizing evolutionary data, *Bioinformatics*, 14:68-73, 1998.

[8] Lefranc, P. *El Cante Jondo. Del territorio a los repertorios: tonás, seguiriyas, soleares.* Publicaciones de la Univ. de Sevilla, Cultura Viva, Vol. 16, 2000.

[9] Serrá, J., Gómez, E., Herrera, P., and Serra, X. Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification. *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 16(6), pp. 1138-1151, 2008.

[10] Widmer, G., Dixon, S., Knees, P., Pampalk, E. Pohle, T. *From Sound to Sense via Feature Extraction and Machine Learning: Deriving High-Level Descriptors for Characterizing Music*, in Sound to Sense, Sense to Sound: A State of the Art in Sound and Music Computing, Chapter 5, 2008.

Acknowledgements: This research has been partially funded by the Junta de Andalucía COFLA project (P09-TIC-4840).

MELODY EXTRACTION FROM POLYPHONIC AUDIO BASED ON PARTICLE FILTER

Seokhwan Jo **Chang D. Yoo**
Department of Electrical Engineering, Korea Advanced Institute of Science Technology,
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Korea
antiland00@kaist.ac.kr cdyoo@ee.kaist.ac.kr

ABSTRACT

This paper considers a particle filter based algorithm to extract melody from a polyphonic audio in the short-time Fourier transforms (STFT) domain. The extraction is focused on overcoming the difficulties due to harmonic / percussive sound interferences, possibility of octave mismatch, and dynamic variation in melody. The main idea of the algorithm is to consider probabilistic relations between melody and polyphonic audio. Melody is assumed to follow a Markov process, and the framed segments of polyphonic audio are assumed to be conditionally independent given the parameters that represent the melody. The melody parameters are estimated using sequential importance sampling (SIS) which is a conventional particle filter method. In this paper, the likelihood and state transition are defined to overcome the aforementioned difficulties. The SIS algorithm relies on sequential importance density, and this density is designed using multiple pitches which are estimated by a simple multi-pitch extraction algorithm. Experimental results show that the considered algorithm outperforms other famous melody extraction algorithms in terms of the raw pitch accuracy (RPA) and the raw chroma accuracy (RCA).

1. INTRODUCTION

Many people believe that people recognize music as a sequence of monophonic notes called melody, and for this reason, melody extraction is playing an important role in music content processing which has recently become an important research area. Although the debate over the definition of melody is on going [1–3], many experts concur that melody should be the dominant pitch sequence of a polyphonic audio. In this paper, melody is defined to be the singing voice pitch sequence in the vocal part and the pitch sequence of the solo instrument in non-vocal part or non-vocal music. When a music contains singing voice, most people recognize music by the vocal melody line in the vocal part. However, in non-vocal part such as intermezzo and non-vocal music such as jazz and orchestra, most people recognize music by the melody line of the solo instrument.

Many melody extraction algorithms have been proposed over the last one decade [1–6], albeit with limited success. Melody extraction from the polyphonic audio is still difficult for the following reasons:

1. Harmonic interference: Harmonics of other instrument signal interfere in the estimation of the melody pitch harmonics.

2. Percussive sound interference: Percussive sound interfere to estimate the melody pitch because the energy of it forms a vertical ridge with strong and wide-band spectral envelopes.

3. Octave mismatch: The estimated pitch can be one octave higher or lower than the ground-truth.

4. Dynamic variation in melody: Accurate pitch estimation in the beginning, end and sudden transient regions of a melody is difficult.

In this paper, melody pitch frequency and harmonic amplitudes that represent the melody are estimated in the short-time Fourier transforms (STFT) domain. The main idea of the algorithm is to consider a probabilistic relations between melody and polyphonic audio. Melody pitch frequency and harmonic amplitudes are assumed to follow Markov processes, and the framed segments of polyphonic audio are assumed to be conditionally independent given melody pitch frequency and harmonic amplitudes. Thus, melody pitch frequency and harmonic amplitudes can be estimated from the polyphonic audio based on the Bayesian sequential model once the likelihood and state transition are defined. The likelihood is defined to be robust to harmonic and percussive sound interferences. The state transition of melody pitch frequency is adjusted by control parameters that discourages octave mismatch and dynamic variation in the melody. The sequential importance sampling (SIS) algorithm, a conventional particle filter algorithm, is used to estimate the melody parameters. The SIS algorithm relies on a so-called sequential importance density, and this density is designed using multiple pitches which are estimated by a simple multi-pitch extraction algorithm.

This paper is organized as follows. Section 2 presents the melody extraction from polyphonic audio based on particle filter. Section 3 provides experimental results. Finally, Section 4 concludes this paper.

2. MELODY EXTRACTION FROM POLYPHONIC AUDIO BASED ON PARTICLE FILTER

2.1 Melody extraction from polyphonic audio

The melody pitch harmonics $x_t[n]$ in the tth frame is defined as follows:

$$x_t[n] = w[n] \sum_{m=1}^{H} A_{m,t} \cos(m\omega_{0,t} n + \phi_{m,t}), \quad (1)$$

where $A_{m,t}$, $\omega_{0,t}$, $\phi_{m,t}$, H and $w[n]$ are the amplitude of the mth harmonic in the tth frame, the melody pitch frequency in the tth frame, the phase of the mth harmonic in the tth frame, number of melody pitch harmonics, and the analysis window function, respectively. The polyphonic audio can be expressed as

$$z_t[n] = x_t[n] + y_t[n], \quad (2)$$

where $z_t[n]$ and $y_t[n]$ are the polyphonic audio signal and signal of other instruments in the tth frame, respectively. In the frequency domain, the following relationship holds:

$$\mathbf{z}_t = \mathbf{x}_t + \mathbf{y}_t, \quad (3)$$

where $\mathbf{z}_t$, $\mathbf{x}_t$, and $\mathbf{y}_t$ are the N-point discrete Fourier transforms (DFT) of $z_t[n]$, $x_t[n]$, and $y_t[n]$, respectively.

The parameters of the melody pitch harmonics – the melody pitch frequency and the harmonic amplitudes – must be estimated for the melody extraction. This paper assumes that the phase of the melody pitch harmonics is the same as the phase of the polyphonic audio, i.e., the phase of the melody pitch is not estimated since human ear is assumed to be unsensitive to phase variations. Thus, the tth frame parameter set is defined as

$$\mathbf{\Theta}_t = (\omega_{0,t}, \mathbf{A}_t), \quad (4)$$

where $\mathbf{A}_t = [A_{1,t}, A_{2,t}, ..., A_{H,t}]$. The objective of melody extraction is to estimate $\mathbf{\Theta}_t$ from given $\mathbf{z}_t$. It is usually observed that successive parameters – $\omega_{0,t}$ and $\mathbf{A}_t$ – are highly correlated. In this paper, it is assumed that $\mathbf{\Theta}_t$ is considered a Markov process and $\mathbf{y}_t$ at each frame is conditionally independent given $\mathbf{\Theta}_t$. Here, $\mathbf{\Theta}_t$ is considered latent while $\mathbf{y}_t$ is observed. From this perspective, the Bayesian sequential model for melody extraction can be constructed as shown in Figure 1. In Figure 1, $p(\mathbf{z}_t|\mathbf{\Theta}_t)$, $p(\mathbf{\Theta}_t|\mathbf{\Theta}_{t-1})$, and ρ_t are likelihood, state transition, and control parameter to decide the state transition of the melody pitch frequency, respectively. From this Bayesian sequential model, the posterior probability $p(\mathbf{\Theta}_{0:t}|\mathbf{z}_{1:t})$ [1] is estimated, and it is used to estimate $\mathbf{\Theta}_t$ for melody extraction. To estimate $p(\mathbf{\Theta}_{0:t}|\mathbf{z}_{1:t})$, likelihood and state evolution equations with state transition needs to be defined.

[1] The notation $a_{0:t}$ means that $a_{0:t} = [a_0, a_1, ..., a_t]^T$

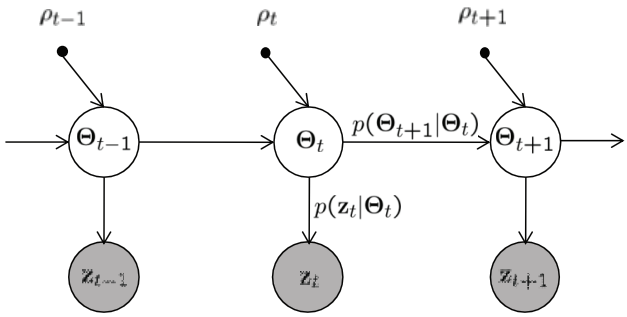


Figure 1. Bayesian sequential model for melody extraction. $\mathbf{z}_t$, $\mathbf{\Theta}_t$, and ρ_t are polyphonic audio, melody parameter ($\omega_{0,t}$ and $\mathbf{A}_t$), and control parameter, respectively.

To obtain the likelihood, it is assumed that the DFT coefficients of $\mathbf{y}_t$ follow a zero mean complex multivariate Gaussian distribution, which is given by

$$\mathbf{y}_t \sim \mathcal{N}(0, \Sigma_t), \quad \Sigma_t = \mathrm{diag}(\sigma_{t,1}^2, \sigma_{t,2}^2, ..., \sigma_{t,N}^2), \quad (5)$$

where Σ_t and $\sigma_{t,k}$ are the covariance matrix in tth frame and the variance of the kth bin in the tth frame, respectively. Eqn. (5) yields the likelihood as follows:

$$\begin{aligned} p(\mathbf{z}_t|\mathbf{\Theta}_t) &= \mathcal{N}(\mathbf{z}_t; \mathbf{x}_t, \Sigma_t) \\ &\propto \exp\left\{-(\mathbf{z}_t - \mathbf{x}_t)^H \Sigma_t^{-1} (\mathbf{z}_t - \mathbf{x}_t)\right\}, \end{aligned} \quad (6)$$

where $(\cdot)^H$ is the Hermitian operator. To define $p(\mathbf{z}_t|\mathbf{\Theta}_t)$, $\sigma_{t,k}$ must be estimated. In this paper, $\sigma_{t,k}$ is estimated using the decision-directed method [7] as follows:

$$\widehat{\sigma}_{t,k} = \alpha \widehat{\sigma}_{t-1,k} + (1-\alpha)|Y_{t,k}|^2, \quad (7)$$

where α and $Y_{t,k}$ are a smoothing factor and the kth bin DFT coefficient of $\mathbf{y}_t$, respectively. However, Eqn. (7) can not be used directly since $Y_{t,k}$ is unknown. It is assumed that $Y_{t,k}$ is highly correlated with $Y_{t-1,k}$. Therefore, the estimation is modified as follows:

$$\widehat{\sigma}_{t,k} = \alpha \widehat{\sigma}_{t-2,k} + (1-\alpha)|\widehat{Y}_{t-1,k}|^2. \quad (8)$$

Accurate estimation of Σ_t will lead to robustness to harmonic and percussive sound interferences. Figure 2 shows an example of $\mathbf{z}_t$ and an estimate of Σ_t, and it is easily shown that the likelihood in Eqn. (6) is maximized at the true $\mathbf{\Theta}_t$.

The state evolution equations, which describe relationships of the parameters at frame t, are set as follows:

$$A_{m,t} = A_{m,t-1} + v_{A,t-1}, \quad (9)$$

$$\omega_{0,t} = \omega_{0,t-1} + v_{\omega_0,t-1}, \quad (10)$$

where $v_{A,t-1}$ and $v_{\omega_0,t-1}$ are the random perturbations corresponding to harmonic amplitudes and melody pitch frequency of the $(t-1)$th frame, respectively. This type of state evolution equations is called *random walk*: the current state is a random perturbation of the previous state. It is important to define $p(v_{A,t-1})$ and $p(v_{\omega_0,t-1})$ accurately, and in this paper, $p(v_{A,t-1})$ is assumed to be a truncated Gaussian as shown in Figure 3 since $A_{m,t} > 0$, and

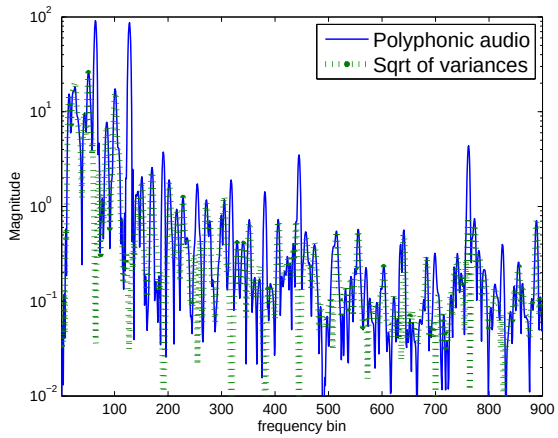


Figure 2. Example of polyphonic audio ($\mathbf{z}_t$) and the estimated variances (Σ_t) of other instrument signal.

$p(v_{\omega_0,t-1})$ is assumed to be a Gaussian whose variance controlled by ρ_t. Melody line is characterized by prolonged periods of smoothness, with infrequent sharp changes in note transition or during vibrato regions.

Furthermore, there are two general rules concerning the melody line: 1) the vibrato exhibits an extent of 60∼200 cents [2] for singing voice and only 20∼30 cents for other [8], and 2) the transitions are typically limited to one octave [1]. Therefore, assumption that $v_{\omega_0,t-1}$ follows a Gaussian distribution with fixed variance is not appropriate. In this paper, the state transition from the from the $(t-1)$th state to the tth state of the melody pitch frequency is controlled by ρ_t which indicates the degree of the melody line being whether in transition or not. Here, transition includes vibrato. And, ρ_t is defined as

$$\rho_t = \widehat{\omega}_{0,t-1} - \widehat{\omega}_{0,t-2}, \tag{11}$$

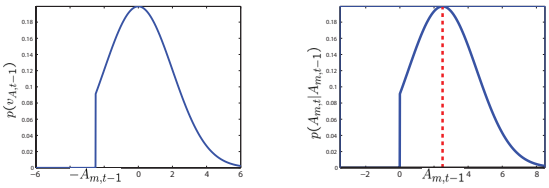
and $p(v_{\omega_0,t-1})$ is given by

$$p(v_{\omega_0,t-1}) = \begin{cases} \mathcal{N}(0, 20 \text{ cent}) & \rho_t < 50 \text{ cent} \\ \mathcal{N}(0, 50 \text{ cent}) & 50 \text{ cent} \le \rho_t < 100 \text{ cent} \\ \mathcal{N}(0, 100 \text{ cent}) & 100 \text{ cent} \le \rho_t \end{cases}. \tag{12}$$

When ρ_t is small, the current melody pitch frequency represents a certain note frequency and has a value similar to the previous melody pitch frequency. When ρ_t is large, the current melody pitch frequency is with high probability in a note transition or vibrato regions and has a value dissimilar to the previous melody pitch frequency. The state transition of melody pitch frequency defined by Eqn. (12) can lead to robustness to octave mismatch and dynamic variation in melody.

[2] The *cent* is a unit of logarithmic frequency range, and it is defined as

$$f_{cent} = 6900 + 1200 \log_2 \frac{f_{Hz}}{440}.$$



(a) $p(v_{A,t-1})$ (b) $p(A_{m,t}|A_{m,t-1})$

Figure 3. State transition in harmonic amplitudes.

2.2 Melody extraction based on particle filter

In this paper, $p(\mathbf{\Theta}_{0:t}|\mathbf{z}_{1:t})$ is approximated using Monte Carlo integration and $\mathbf{\Theta}_t$ is estimated using the particle filter. The SIS algorithm which is a common particle filter method [9, 10] is adopted to estimate the parameters of the melody. If the likelihood and the state transition follow a Gaussian distribution, the problem can be solved by Kalman filter. However, the state transition is not assumed to be a Gaussian. The SIS algorithm is used to obtain $p(\mathbf{\Theta}_{0:t}|\mathbf{z}_{1:t})$ based on the Bayesian sequential model shown as Figure 1.

The posterior density $p(\mathbf{\Theta}_{0:t}|\mathbf{z}_{1:t})$ can be approximated as follows:

$$p(\mathbf{\Theta}_{0:t}|\mathbf{z}_{1:t}) \approx \sum_{i=1}^{N_p} w_t^{(i)} \delta(\mathbf{\Theta}_{0:t} - \mathbf{\Theta}_{0:t}^{(i)}), \tag{13}$$

where $\mathbf{\Theta}_{0:t}^{(i)}$, $w_t(i)$, and N_p are the ith particle of $\mathbf{\Theta}_{0:t}$, associated weight, and the number of particles, respectively. The weights are normalized such that $\sum_{i=1}^{N_p} w_t^{(i)} = 1$. The weights are chosen using the method of importance sampling. If the particle $\mathbf{\Theta}_{0:t}^{(i)}$ were drawn from an importance density $q(\mathbf{\Theta}_{0:t}^{(i)}|\mathbf{z}_{1:t})$, the weights in Eqn. (13) are defined as follows:

$$w_t^{(i)} \propto \frac{p(\mathbf{\Theta}_{0:t}^{(i)}|\mathbf{z}_{1:t})}{q(\mathbf{\Theta}_{0:t}^{(i)}|\mathbf{z}_{1:t})}. \tag{14}$$

If the importance density is chosen to factorize as follows

$$q(\mathbf{\Theta}_{0:t}|\mathbf{z}_{1:t}) = q(\mathbf{\Theta}_t|\mathbf{\Theta}_{0:t-1}, \mathbf{z}_{1:t}) q(\mathbf{\Theta}_{0:t-1}|\mathbf{z}_{1:t-1}), \tag{15}$$

then one can obtain particles $\mathbf{\Theta}_{0:t}^{(i)} \sim q(\mathbf{\Theta}_{0:t}^{(i)}|\mathbf{z}_{1:t})$ by augmenting each of the existing particles $\mathbf{\Theta}_{0:t-1}^{(i)} \sim q(\mathbf{\Theta}_{0:t-1}^{(i)}|\mathbf{z}_{1:t-1})$ with the new state $\mathbf{\Theta}_t^{(i)} \sim q(\mathbf{\Theta}_t|\mathbf{\Theta}_{0:t-1}, \mathbf{z}_{1:t})$. The weight update equation can be derived as follows using Eqn. (14) and Eqn. (15)

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(\mathbf{z}_t|\mathbf{\Theta}_t^{(i)}) p(\mathbf{\Theta}_t^{(i)}|\mathbf{\Theta}_{t-1}^{(i)})}{q(\mathbf{\Theta}_t^{(i)}|\mathbf{\Theta}_{t-1}^{(i)}, \mathbf{z}_t)}. \tag{16}$$

A common problem with the particle filter is the degeneracy phenomenon, where after a few iterations, most particles have negligible weight [9,10]. A suitable measure of degeneracy is the effective particle size, N_{eff}, which is given by

$$\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^{N_p} (w_t^{(i)})^2}. \tag{17}$$

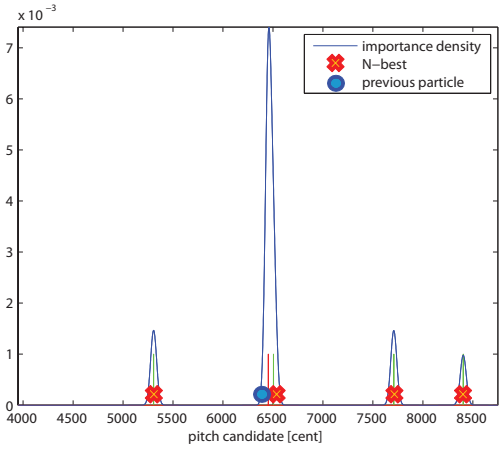


Figure 4. Design of $q(\omega_{0,t}^{(i)}|\omega_{0,t-1}^{(i)}, \mathbf{z}_t)$.

In this paper, to avoid the degeneracy problem, resampling algorithm is used when $N_{eff} \leq \frac{N_p}{2}$.

Finally, estimation of parameters is achieved by posterior mean after obtaining $p(\mathbf{\Theta}_{0:t}|\mathbf{z}_{1:t})$.

$$\widehat{\omega}_{0,0:t} = \sum_{i=1}^{N_p} w_t^{(i)} \omega_{0,0:t}^{(i)}, \tag{18}$$

$$\widehat{\mathbf{A}}_{0:t} = \sum_{i=1}^{N_p} w_t^{(i)} \mathbf{A}_{0:t}^{(i)}. \tag{19}$$

2.2.1 *Design of sequential importance density*

The performance of the SIS algorithm depends on the choice of $q(\mathbf{\Theta}_t^{(i)}|\mathbf{\Theta}_{t-1}^{(i)}, \mathbf{z}_t)$. Setting $q(\mathbf{\Theta}_t^{(i)}|\mathbf{\Theta}_{t-1}^{(i)}, \mathbf{z}_t) = p(\mathbf{\Theta}_t^{(i)}|\mathbf{\Theta}_{t-1}^{(i)})$ leads to not only unnecessary large number of particles but also difficulties in estimating $p(\mathbf{\Theta}_{0:t}|\mathbf{z}_{1:t})$. In this paper, a multiple pitch estimation algorithm is used to define $q(\mathbf{\Theta}_t^{(i)}|\mathbf{\Theta}_{t-1}^{(i)}, \mathbf{z}_t)$ since the melody pitch frequency is assumed to be one of the pitch estimate given by the multiple pitch estimates. A main idea in defining $q(\mathbf{\Theta}_t^{(i)}|\mathbf{\Theta}_{t-1}^{(i)}, \mathbf{z}_t)$ is to generate particles of the melody parameters similar to the estimated multiple pitch parameters. To obtain multiple pitch parameters, the multiple pitch estimation algorithm proposed in [11] is used.

Before drawing particles from the importance density, $q(\mathbf{\Theta}_t^{(i)}|\mathbf{\Theta}_{t-1}^{(i)}, \mathbf{z}_t)$ is factorized as follows:

$$\begin{aligned} & q(\omega_{0,t}^{(i)}, \mathbf{A}_t^{(i)}|\omega_{0,t-1}^{(i)}, \mathbf{A}_{t-1}^{(i)}, \mathbf{z}_t) \\ = \; & q(\mathbf{A}_t^{(i)}|\omega_{0,t}^{(i)}, \mathbf{A}_{t-1}^{(i)}, \mathbf{z}_t) q(\omega_{0,t}^{(i)}|\omega_{0,t-1}^{(i)}, \mathbf{z}_t). \end{aligned} \tag{20}$$

Here, $\omega_{0,t}$ and $\mathbf{A}_t$ are considered conditionally independent given $\omega_{0,t-1}^{(i)}$, $\mathbf{A}_{t-1}^{(i)}$, and $\mathbf{z}_t$. First, melody pitch particles are drawn as given by

$$\omega_{0,t}^{(i)} \sim q(\omega_{0,t}^{(i)}|\omega_{0,t-1}^{(i)}, \mathbf{z}_t), \tag{21}$$

where $q(\omega_{0,t}^{(i)}|\omega_{0,t-1}^{(i)}, \mathbf{z}_t)$ is shown as Figure 4. In defining $q(\omega_{0,t}^{(i)}|\omega_{0,t-1}^{(i)}, \mathbf{z}_t)$, the current melody pitch particles are drawn near the N-best pitch candidates obtained from the multiple-pitch estimation and the melody pitch particles drawn in the previous frame. After drawing melody pitch particles, melody pitch harmonic amplitudes particles are drawn as given by

$$\begin{aligned} & \mathbf{A}_t^{(i)} \sim q(\mathbf{A}_t^{(i)}|\omega_{0,t}^{(i)}, \mathbf{A}_{t-1}^{(i)}, \mathbf{z}_t) \\ = \; & \mathcal{N}\left(\frac{\mathbf{A}_{t-1}^{(i)} + \mathbf{A}_t^{\mathbf{z}^{\omega_{0,t}^{(i)}}}}{2}, \frac{|\mathbf{A}_{t-1}^{(i)} - \mathbf{A}_t^{\mathbf{z}^{\omega_{0,t}^{(i)}}}|}{2} \right) \end{aligned} \tag{22}$$

where $\mathbf{A}_t^{\mathbf{z}^{\omega_{0,t}^{(i)}}}$ is the harmonic amplitudes corresponding pitch candidate near $\omega_{0,t}^{(i)}$ with constraint $\mathbf{A}_t^{(i)} > 0$. In defining $q(\mathbf{A}_t^{(i)}|\omega_{0,t}^{(i)}, \mathbf{A}_{t-1}^{(i)}, \mathbf{z}_t)$, the current harmonic amplitude particles which are similar to the previous harmonic amplitude particles and harmonic amplitudes of the N-best pitch candidates are generated. If $\mathbf{A}_{t-1}^{(i)}$ and $\mathbf{A}_t^{\mathbf{z}^{\omega_{0,t}^{(i)}}}$ are similar, then $\frac{|\mathbf{A}_{t-1}^{(i)} - \mathbf{A}_t^{\mathbf{z}^{\omega_{0,t}^{(i)}}}|}{2} \approx 0$, therefore, $\mathbf{A}_t^{(i)} \approx \frac{\mathbf{A}_{t-1}^{(i)} + \mathbf{A}_t^{\mathbf{z}^{\omega_{0,t}^{(i)}}}}{2}$. If $\mathbf{A}_{t-1}^{(i)}$ and $\mathbf{A}_t^{\mathbf{z}^{\omega_{0,t}^{(i)}}}$ are not similar, then $\frac{|\mathbf{A}_{t-1}^{(i)} - \mathbf{A}_t^{\mathbf{z}^{\omega_{0,t}^{(i)}}}|}{2} >> 0$, therefore, $\mathbf{A}_t^{(i)}$ is generated somewhat randomly.

The outline of the considered algorithm is given below.

Outline of the considered algorithm

Melody extraction based on the SIS

For $i = 1, ..., N_p$

1. Generate the particles
   - Melody pitch particles $\omega_{0,t}^{(i)} \sim q(\omega_{0,t}^{(i)}|\omega_{0,t-1}^{(i)}, \mathbf{z}_t)$
   - Harmonic amplitudes particles $\mathbf{A}_t^{(i)} \sim q(\mathbf{A}_t^{(i)}|\omega_{0,t}^{(i)}, \mathbf{A}_{t-1}^{(i)}, \mathbf{z}_t)$
2. *Update the weights*: Eqn. (16)

Normalize the weights ($\sum_{i=1}^{N_p} w_t^{(i)} = 1$).

Resampling: Resampling algorithm is used when $N_{eff} \leq \frac{N_p}{2}$.

Estimation: Melody pitch frequency in tth frame is estimated by Eqn. (18). Harmonic amplitudes of melody pitch harmonics in tth frame are estimated by Eqn. (19).

---

3. EVALUATION

The considered algorithm was evaluated and compared to other melody extraction algorithms using the ISMIR 2004 Audio Description Contest (ADC04) database. The database contains 20 polyphonic musical audio pieces. All test data are single channel PCM data with 44.1 kHz sample rate and 16-bit quantization. Table 1 shows the data composition of the ADC04 set. Search range of melody pitch frequency was between 80Hz and 1280Hz in frequency do-

Melody Instrument	Sytle
Synthesized voice (4)	POP
Saxophone (4)	Jazz
MIDI instruments (4)	Folk(2), Pop(2)
Human voice (2 male, 2 female)	Classical opera
Male Voice (4)	POP

Table 1. Summary of ADC04 data set. The number in parentheses is the number of corresponding pieces.

	RPA	RCA
Goto [2]	65.8% (2005)	71.8% (2005)
Paiva el al. [3]	62.7% (2005)	66.7% (2005)
Marlot [4]	60.1% (2005)	67.1% (2005)
Ryynanen el al. [5]	68.6% (2005)	74.1% (2005)
Ellis el al. [6]	73.2% (2006)	76.4% (2006)
Considered algorithm	77.3%	83.8%

Table 2. Result comparison. The number in parentheses is the year when their algorithms were submitted to the MIREX.

main (3950 cent and 8750 cent in cent domain). The Hanning window was used with 48ms frame length and 10ms frame hop size. $\alpha = 0.98$ in Eqn. (8) was used. $N_p = 500$ in Eqn. (13) was used.

The estimated melody is correct when the absolute value of the difference between the ground-truth frequency and estimated frequency is less than 50 cent ($\frac{1}{4}$ tone). The performance of the considered algorithm was evaluated in terms of raw pitch accuracy (RPA) and raw chroma accuracy (RCA). The RPA is defined as the proportion of frames in which the estimated melody pitch is within $\pm\frac{1}{4}$ tone of the reference pitch. And the RCA is defined in the same manner as the raw pitch accuracy; however, both the estimated and reference frequencies are mapped into a single octave in order to forgive octave transpositions.

The considered algorithm was compared to the other famous melody extraction algorithms such as algorithms proposed by Goto [2], Paiva et al. [3], Marlot [4], Ryynanen el al. [5], and Ellis et al. [6]. Their performances are based on results of the Music Information Retrieval Evaluation eXchange (MIREX) [12].

Table 2 shows the evaluation results for all algorithms considered. The considered algorithm outperformed the others in terms of the RPA and the RCA. The difference between the RPA and RCA is proportional octave mismatch error. Although the algorithm in this paper is considered to be robust against octave mismatch, the difference between the RPA and the RCA is 6.5 %. The multiple pitch estimation algorithm proposed in [11] was quite simple and vulnerable to octave error, i.e., inaccuracy in sequential importance density led to inaccurate melody pitch candidates.

4. CONCLUSION

The melody extraction algorithm from the polyphonic audio based on particle filter is considered in this paper. Most people recognize music as not all of note sequences but a special monophonic note sequence called melody. However, melody extraction from polyphonic audio is difficult due to the following impediments: harmonic interference, percussive sound interference, octave mismatch, and dynamic variation in melody. The main idea of the algorithm is to consider probabilistic relations between melody and polyphonic audio. Melody is assumed to follow a Markov process, and the framed segments of polyphonic audio are assumed to be conditionally independent given the parameters that represent the melody. The parameters are estimated using the SIS algorithm. This paper shows that likelihood and state transition that are required in the SIS algorithm are defined to be robust against the aforementioned impediments. The performance of the SIS algorithm depends on a sequential importance density, and this density is designed by multiple pitch. Experimental results show that the considered algorithm outperformed the other famous melody extraction algorithms.

5. ACKNOWLEDGEMENTS

This work was supported by the Ministry of Culture, Sports and Tourism (MCST) and Korea Culture Content Agency (KOCCA) in the Culture Technology (CT) Research and Development Program 2009.

6. REFERENCES

[1] G. E. Poliner, D. P. W. Ellis, and A. F. Ehmann: "Melody transcription from music audio: approach and evaluation," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 15, No. 4, pp. 1247–1256, 2007.

[2] M. Goto: "A real-time music-scene-description system: predominant-f0 estimation for detecting melody and bass lines in real-world audio signals," *Speech Communication*, Vol. 43, No. 4, pp. 311–329, 2004.

[3] R. P. Paiva, T. Mendes, and A. Cardoso: "Melody detection in polyphonic musical signals: exploiting perceptual rules, note salience, and melodic smoothness," *Computer Music Journal*, Vol. 30, No. 4, pp. 80–98, 2006.

[4] M. Marolt: "On finding melodic lines in audio recordings," *Proceeding of 7th International Conference on Digital Audio Effects DAFx 04*, pp. 217–221, 2004.

[5] M. P. Ryynanen and A. P. Klapuri: "Note event modeling for audio melody extraction," *MIREX 2005 Audio Melody Extraction Contest*, 2005.

[6] D. P. W. Ellis and G. E. Poliner: "Classification-based melody transcription," *Machine Learning*, Vol. 65, pp. 439–456, 2006.

[7] Yariv Ephraim: "Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 32, No. 6, pp. 1109–1121, 1984.

[8] R. Timmers and P. W. M Desain: "Vibrato: the questions and answers from musicians and science," *Proceedings of International Conference on Music Perception and Cognition*, 2000.

[9] A. Doucet, N. de Freitas, and N. J. Gordon: *Sequential Monte Carlo methods in practice*, Springer-Verlag, New York, 2001.

[10] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp,: "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, Vol. 50, No. 2, pp. 174–188, 2002.

[11] S. Joo, S. Jo, and C. D. Yoo: "Melody extraction from polyphonic audio signal MIREX 2009," *MIREX 2009 Audio Melody Extraction Contest*, 2009.

[12] J. S. Downie, K. West, A. Ehmann, and Vincent E: "The 2005 music information retrieval evaluation exchange (mirex 2005): preliminary overview," *Proceedings of the Sixth International Conference on Music Information Retrieval*, pp. 320–323, 2005.

MULTIPLE PITCH TRANSCRIPTION USING DBN-BASED MUSICOLOGICAL MODELS

Stanisław A. Raczyński* raczynski@ **Emmanuel Vincent+** emmanuel.vincent@ **Frédéric Bimbot+** frederic.bimbot@ **Shigeki Sagayama*** sagayama@

* The University of Tokyo, 7–3–1 Hongo, Bunkyo-ku, Tokyo 133-8656, Japan, email: *hil.t.u-tokyo.ac.jp

+ INRIA Rennes, Britagne Atlantique, 35042 Rennes Cedex, France, email: *inria.fr

ABSTRACT

We propose a novel approach to solve the problem of estimating pitches of notes present in an audio signal. We have developed a probabilistically rigorous model that takes into account temporal dependencies between musical notes and between the underlying chords, as well as the instantaneous dependencies between chords, notes and the observed note saliences. We investigated its modeling ability by measuring the cross-entropy with symbolic (MIDI) data and then proceed to observe the model's performance in multiple pitch estimation of audio data.

1. INTRODUCTION

The problem at hand is *musical note detection*, *i.e.* estimating pitches, onset and offset times, and, if desired, velocities of notes present, often simultaneously, in a recorded audio signal. Typically, this problem is solved by a two-step process [9]. First, pitch candidates are estimated within short time frames and confidence for each is quantified by a salience measure (see, for example, [4,6,11]). Then the salience is tracked over time in order to identify the musical notes.

The salience can be represented by a *note salience matrix* $\boldsymbol{S}$. Its rows contain estimated power envelopes of notes for different pitches, which typically correspond to frequencies of a diatonic scale, *e.g.* twelve-tone equal temperament scale. The activity of the underlying musical notes can be expressed by a *note activity matrix* $\boldsymbol{N}$, *i.e.* a binary matrix of the same dimensions as $\boldsymbol{S}$, elements of which indicate note presence at corresponding times and pitches.

A standard practice is to threshold the estimated saliences to detect notes. This step, although common, is quite problematic: there is no simple way to determine the threshold value and even an optimal value can lead to spurious detections and split notes. Some of the false positives and negatives can be removed by filtering, but it does not solve the problem completely and is not elegant.

Thresholding can in fact be interpreted as a maximum likelihood (ML) estimator of the note activities:

$$\hat{\boldsymbol{N}} = arg\,max_{\boldsymbol{N}}\, P(\boldsymbol{S}|\boldsymbol{N}) \,, \qquad (1)$$

If we assume that the detected saliences $S_{t,k}$ are mutually independent and only depend on whether a corresponding note was active at that moment, we get:

$$\hat{N}_{t,k} = arg\,max_{N_{t,k}} P(S_{t,k}|N_{t,k}) \,, \qquad (2)$$

where k is the piano key number and t is the time frame number. If the probability distributions $P(S_{t,k}|N_{t,k}=1)$ and $P(S_{t,k}|N_{t,k}=0)$ have only one crossing point T, this procedure will be equivalent to thresholding with the threshold value equal to T.

Recently, some researchers have used more advanced musicological models in order to overcome the limitations of thresholding. Ryynänen and Klapuri [9] proposed a melody transcription method that uses a Hidden Markov Model (HMM) together with a simple musical key model. Their approach is limited in the sense that it models only a single voice at a time, and so it is not probabilistically rigorous. It also lacks modeling of instantaneous dependencies between estimated pitches. Raphael and Stoddard [8] proposed to use an HMM as a musicological model for harmonic analysis, *i.e.* estimating the chord progression behind a sequence of notes. Similar HMMs have also been successfully used for harmonic analysis of audio signals (for a recent paper see *e.g.* [10]). These approaches, however, lack note modeling and the temporal dependencies are only present between chords. A very interesting model has been presented by Cemgil *et al.* in [1], but the presented results are still preliminary (the model, like ours, is computationally expensive).

In this paper we have proposed a single, probabilistically rigorous framework based on the Dynamic Bayesian Networks (DBNs). We model both the instantaneous dependencies between notes (harmony) and the temporal dependencies between notes and chords. The notes our found with a maximum a posteriori (MAP) estimator:

$$\hat{\boldsymbol{N}} = arg\,max_{\boldsymbol{N}} P(\boldsymbol{S}|\boldsymbol{N})\, P(\boldsymbol{N}) \,. \qquad (3)$$

The prior over the notes $P(\boldsymbol{N})$ models the temporal dependencies between the hidden variables (similar to those of an Hidden Markov Model) and includes a hidden layer of variables representing chords.

In our work we used a NMF-based front-end proposed in [7] to obtain note salience matrices with 88 rows that correspond to the full range of a piano: from A0 (27.5 Hz) to C8 (4186 Hz).

The model with its theoretical grounds and its practical aspects is described in section 2. Inference of the hidden notes is discussed in section 3. Experiments involving symbolic and audio data are described in section 4. and the conclusion is given in section 5.

2. THE MODEL

2.1 Structure

DBNs provide us with complete freedom as to what set of probabilistic variables and the relations between them can be, and so it is a perfect tool to solve the above formulated problem. We have chosen a network structure that, compared to thresholding, includes dependencies between hidden variables in neighboring time frames and an additional layer of hidden chords (see Fig. 1).

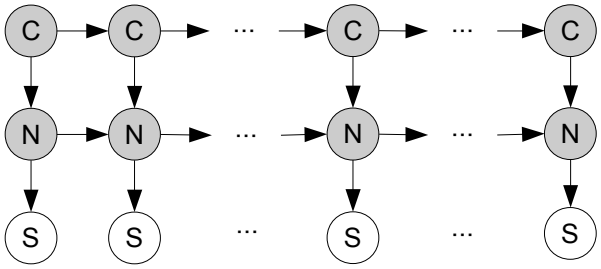


Figure 1: *Structure of the Bayesian network used in experiments*

The network consist of 3 layer of nodes: hidden *chord* layer C_t, hidden *note combination* layer N_t and an observed *note salience* layer S_t.. The prior distribution of notes is therefore given by:

$$P(\boldsymbol{N}) = \sum_{\boldsymbol{C}} P(C_0)\,P(N_0|C_0)\cdot \cdot\prod_{t=2}^{T} P(N_t|N_{t-1},C_t)P(C_t|C_{t-1}) \quad . \tag{4}$$

2.2 Chord level probabilities

Fig. 4 shows the chord transition probabilities that has been trained on the available dataset. A simple smoothing was used: each element was increased by 1 after counting the occurrences and before normalizing. Nevertheless, the data sparsity problem is visible (especially for the minor, rarer, chords). To deal with this problem, chord tying was used: each chord transition probability was assumed to be a function of only the interval between roots of the chords R_t and R_{t-1}, and their types T_t and T_{t-1}:

$$P(C_t|C_{t-1}) = P(R_t - R_{t-1}, T_t, T_{t-1}) \quad . \tag{5}$$

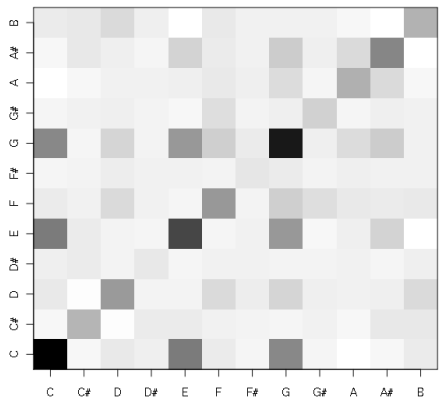


Figure 2: *Covariance matrix for the C-major chord. Note the positive high covariance between the root and the perfect fifth (harmonic interval) and weak covariance between root and minor second (inharmonic interval).*

The motivation behind this approximation is that the probability depends on relative chord positions rather than on the absolute ones. Because the tonal center is not modeled in our approach, it is reasonable to assume the same probability should be given to the transition from C-major chord to F-major (I→IV transition in C-major key) and the from A♭-major to D♭-major (I→IV transition in A♭-major key).

The same motivation led us to use a uniform distribution as the initial chord probability distribution:

$$P(C_0) = \text{const} \quad . \tag{6}$$

2.3 Note level probabilities

Another practical problem concerning the size of the note combination space is the problem of training the model's parameters. The note combination probability $P(N_t|N_{t-1},C_t)$ is a discrete distribution with $|L|^2|C|$ parameters to train, which, even for small values of L is computationally infeasible. To decrease the complexity of the problem, we again tie together some of the parameters: we replace that the note combination probability an approximation, in which it is factorized into the *note transition probability* $P(N_t|N_{t-1})$ and the *note emission probability* $P(N_t|C_t)$:

$$P(N_t|N_{t-1},C_t) \approx \frac{P(N_t|N_{t-1})\,P(N_t|C_t)}{\sum_{N_t} P(N_t|N_{t-1})\,P(N_t|C_t)} \cdot \tag{7}$$

The note probability distribution, as well as the note emission and transition distributions, was normalized over all unique note combinations in the reduced search space. In case of calculating joint likelihood from symbolic data, the sum is performed over all note combinations present in the analyzed data.

2.3.1 Note emission probabilities

There is commonly used multivariate parametric distribution over a discrete set, so to model the note emissions we chose a multivariate Gaussian distribution in the 12-tone chroma space.

$$Cr_{t,l} = \sum_{k \equiv l \,(\mathrm{mod}\,12)} N_{t,k} \tag{8}$$

$$P(N_t|C_t = m) = \frac{\mathcal{N}_{12}(Cr_t;\mu_m,\Sigma_m)}{\sum_{N_t}\mathcal{N}_{12}(Cr_t;\mu_m,\Sigma_m)} \tag{9}$$

The distribution parameters were estimated on the ground truth data (see Fig. 3)and parameters corresponding to the same chord type were tied together as for the chord

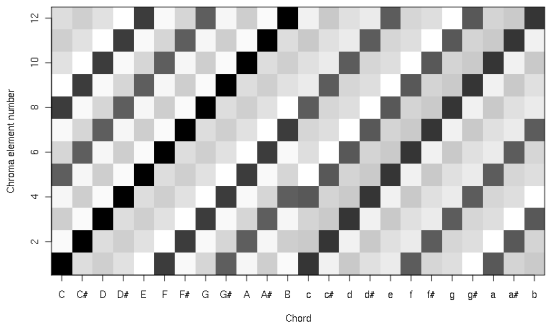


Figure 3: *Mean chroma vectors for different chords.*

level probabilities. To avoid singular covariance matrices due to sparse training data, chroma vectors obtained from reference data were concatenated with a smoothing diagonal matrix $p\boldsymbol{I}$, where p is a control parameter ($p = 2$ was used). The chroma variance is modeled with a full-rank matrix because the pitch classes are not independent (see Fig. 2).

2.3.2 Note transition probabilities

The note transition probability $P(N_t|N_{t-1})$ is responsible for modeling note lengths. There are five basic kinds of changes in the note combination state that can occur in the data (depicted in Fig. 7): no change, insertion of notes, deletion of notes, voice movement (one note changes pitch) and harmony movement or other complex changes (many notes change pitches simultaneously). Because in real life situations note offsets are seldom aligned with other notes' onsets, the last two situations are very rare. In out training data they made up for only 0.2% of note transitions types, while transitions in which all notes stayed the same, if we don't count the insertions and deletions, made up for remaining 99.8% of situations. Motivated by this, in order to simplify the model, we assumed that only the first three kinds are allowed.

The note transition probability is therefore further approximated with the following factorization:

$$P(N_t|N_{t-1}) \approx \frac{P_{len}(L_t|L_{t-1})P_{mov}(N_t|N_{t-1})}{\sum_{N_t} P_1(L_t|L_{t-1})P_2(N_t|N_{t-1})} \qquad (10)$$

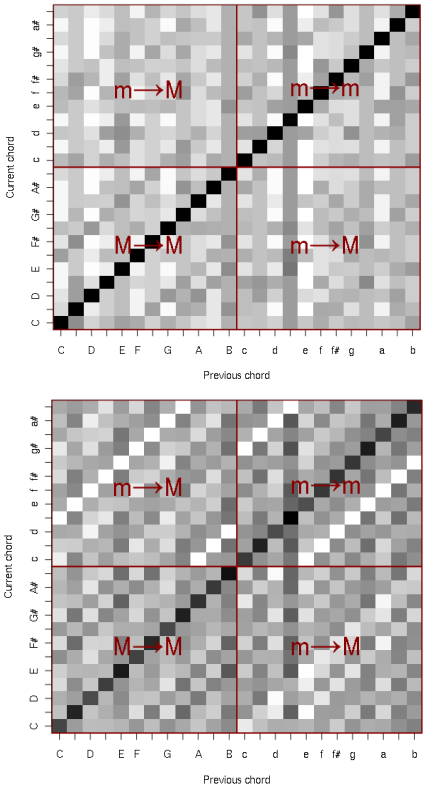


Figure 4: *Chord transition probability matrices: without state tying (top) and with state tying (bottom). Four quarters represent: the major-to-major (M→M), minor-to-major (m→M), major-to-minor (M→m) and minor-to-minor (m→m) transition.*

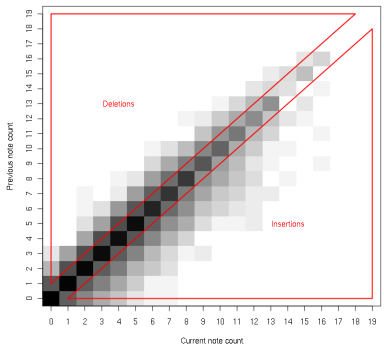


Figure 5: *Distribution of note combination length transitions. The probability matrix is "smeared" more in the area of simultaneous insertions of multiple notes*

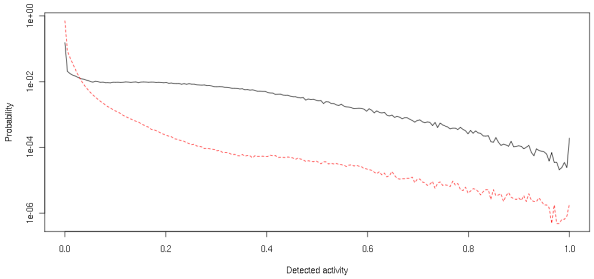


Figure 6: *The estimated output probability . The black solid line depicts the distribution of observed note salience if the note was active ($P(S_{t,k}|N_{t,k}=1)$) and the red dashed line the distribution in case the note was inactive ($P(S_{t,k}|N_{t,k}=0)$). The lines cross at about -70 dB.`*

(e.g. at beginnings of chords) are more probable than simultaneous deletions. The z-axis is logarithmic.

$$P_{mov}(N_t, N_{t-1}) = \begin{cases} 1 & \text{for no pitch movement} \\ 0 & \text{for pitch movement} \end{cases} \qquad (11)$$

where L_t is the size of the current note combination (number of active notes). $P_1(L_t, L_{t-1})$ is presented in Fig. 5.

2.3.3 Output probabilities

The observed note saliences are assumed to be mutually independent:

$$P(S_t|N_t) = \prod_{k \in 1}^{88} P(S_{t,k}|N_{t,k}) \ . \qquad (12)$$

Both obtained by measuring the histograms of the detected salience (see Fig. 6).

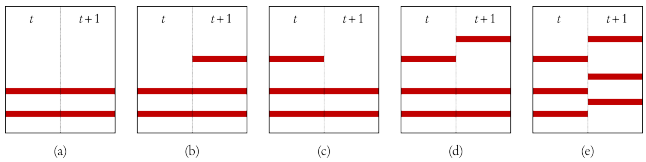


Figure 7: *Five basic note combination transition situations: (a) no change, (b) insertion, (c) deletion, (d) voice movement and (e) harmony movement or other complex changes.*

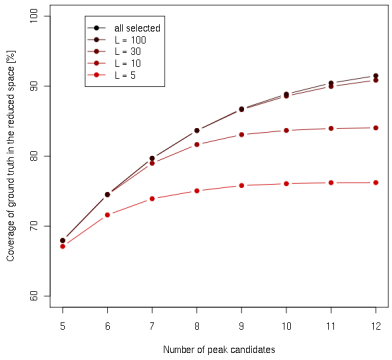


Figure 8: *Note recall for different values of N and L. Data obtained for a = 0.65 and R = 20.*

3. DECODING

3.1 Inference

The problem of multiple frequency estimation becomes a problem of inferring the hidden sequence of note combination states (and, as a side effect, the hidden chord progression). In other words, we need to find the most likely hidden state sequence ($\boldsymbol{C}$, $\boldsymbol{N}$) given the model and the observed note saliences $\boldsymbol{S}$:

$$(\boldsymbol{C}, \boldsymbol{N}) = arg\ max_{(\boldsymbol{C},\boldsymbol{N})} P(\boldsymbol{C}, \boldsymbol{N} | \boldsymbol{S}) . \quad (13$$

This problem is in fact directly related to the Viterbi decoding in Hidden Markov Models (HMMs).

As in with Viterbi decoding, a dynamic-programming-based algorithm can be used to solve this inference problem for DBNs. We refer to this algorithm as modified Frontier Algorithm. The original Frontier Algorithm was proposed by Murphy in [5] to calculate the probability of a given observed sequence (equivalent of the HMM's Forward-Backward algorithm). Murphy noted that it can easily be modified to calculate the most probable sequence of hidden states in any finite-state DBN, *i.e.* solve our inference problem.

3.2 Reduced solution space

N_t is a variable that holds a list of notes active at a certain time (or, equivalently, a vector of binary note presence indicators). The number of all possible values (states) of N_t is enormous: 3.1×10^{26} if we limit the musical range to that of a 88-key piano. Even if we limit the number of simultaneously active notes to K=10 (if no sustain pedal is used this is the physical limit for a single piano player), it is still computationally infeasible: 5.2×10^{12} if K=10 and 4.2×10^{7} if K=5.

To deal with this problem, we reduce the solution prior to inferring the hidden sequence: for each time frame only the most probable note combinations are considered. To identify the most probable note combinations, first, for each time frame, we select K highest elements, or *note candidates*. Then, a list of all 2^K possible note combinations is created and each such combination is evaluated with a *fitness function*. Finally, the L fittest note combinations are selected and used for further analysis. Additionally, a rest (empty note combination) is always selected.

The fitness function was designed to penalize long note combinations (note combinations containing many active notes) while rewarding better explanation of the observed note saliences S_t:

$$F(N_t) = \frac{\sum_{k \in [k:N_{t,k}=1]} S_{t,k}}{\sum_{k=1}^{88} S_{t,k}} |N_t|^{-a} \quad (14)$$

where N_t is the note combination for the current time frame and a is a control parameter. A similar fitness function was used by Klapuri in [3].

Limiting the solution space poses a threat to the note estimation process: if the real note combination is not selected due to fluctuations in the note salience, the language model will not be able to compensate for that. Therefore, to avoid some of the deletions, the observed note saliences are pre-filtered with a causative moving average (MA) filter:

$$\bar{S}_{t,k} = \sum_{\tau=0}^{R} S_{t-\tau,k} \quad (15)$$

Additionally, this filtering removes short spurious peaks, e.g. the ones around onset times resulting from the wide-band onset noise. Unfortunately, it also smooths out the onsets.

We have analyzed how much of the ground truth is contained within the reduced solution space, depending on the chosen K, L and a, and on the chosen MA filter order (length), by measuring the note recall. The results are presented in Fig. 8. Optimal values were determined to be R = 20 (400 ms) and a = 0.65 (similar to Klapuri's [3]).

3.3 Fudging

To gain additional control over the behavior of the algorithm, a set of fudge factors was introduced:

$$P(N_t | N_{t-1}, C_t) \approx \frac{P(N_t | N_{t-1})^{\alpha} P(N_t | C_t)^{\beta}}{\sum_{N_t} P(N_t | N_{t-1})^{\alpha} P(N_t | C_t)^{\beta}} \quad (16)$$

Each factor controls the influence of individual probability distribution on the algorithm. The first factor controls mainly the ratio between the self-transition probability of N_t and the cross-transition probability, so smaller values are better for slower pieces and bigger values for higher tempo.

The values of first two factors were then optimized empirically by maximizing the joint likelihood of the hidden note variables $P(\boldsymbol{N})$ (see Fig. 9) and found to be $\alpha = 1.05$ and $\beta = 0.0015$. The fact that the first factor is close to one does not surprise, because the Gaussian is very sparse due to high dimensionality. A very small value

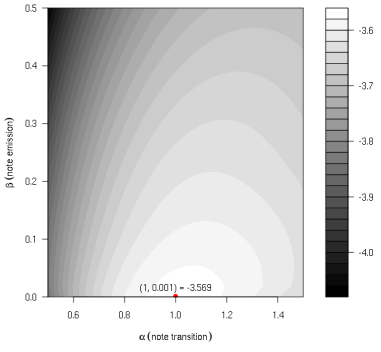


Figure 9: *Optimization of the fudge factors α and β.*

of β is due to a very high sparseness of the note emission distribution, *i.e.* small number of note combinations are assigned significantly higher probability values than the others (which is a result of the curse of dimensionality).

	RWC	Composer	Instrument	Length
1	22	Brahms	2 pianos	2:25
2	23A	Ravel	Piano	1:20
3	23B	〃	Piano	2:45
4	23C	〃	Piano	3:25
5	23E	〃	Piano	4:09
6	24A	Bach	Harpsichord	1:26
7	24B	〃	Harpsichord	1:29
8	24C	〃	Harpsichord	0:52
9	25A	〃	Harpsichord	2:03
10	25B	〃	Harpsichord	2:11
11	25C	〃	Harpsichord	1:31
12	29	Schumann	Piano	2:25
13	30	Chopin	Piano	4:02
14	31	〃	Piano	4:16
15	32	〃	Piano	1:49
16	35A	Satie	Piano	3:49
17	35B	〃	Piano	3:01
18	35C	〃	Piano	2:46
19	40	Massanet	Piano + violin	5:06
			Total:	50:50

Table 1: *RWC pieces used in the experiments.*

4. EXPERIMENTS

4.1 The dataset

The data used in the experiments comes from the widely used RWC database [2]. We have used 19 pieces from the classical portion of the dataset (listed in Table 1).

As a joint effort of the University of Tokyo's Sagayama Laboratory and the Toho Gakuen School of Music (under the supervision of prof. Hitomi Kaneko), the classical pieces of the RWC database were annotated with detailed harmony labels that include: keys and modulations, and chords with their roots, inversions, types and modifications. This data uses abstract musical time (measures and beats), so, additionally, manual labeling of the RWC's audio data was performed.

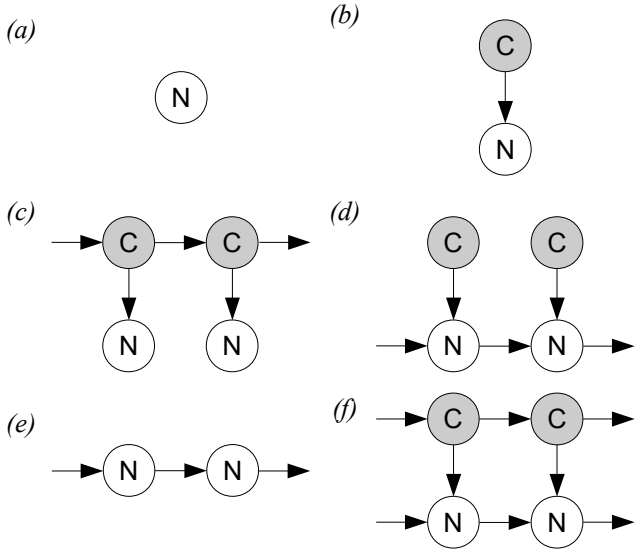


Figure 10: *Six variants of the model used in the evaluation.*

Unfortunately, the RWC database's MIDI and audio files are not synchronized. What is more, it is not only a matter of linear time transformation, but rather a complex one. Further synchronization with the MIDI was needed for the purpose of training model parameters (note emission probabilities). This was done automatically with dynamic time warping (DTW).

4.2 Symbolic data

A simple procedure to evaluate the proposed approach is to measure how well does our Bayesian network model the symbolic data. This can be assessed by calculating the likelihood of the data given the model $P(\boldsymbol{N})$.

Six variants of the proposed model were evaluated:

(a) Reference uniform model

$$P(\boldsymbol{N})=\prod_{t=1}^{T} P(N_t)= A^T \tag{17}$$

(b) Harmony model only

$$P(\boldsymbol{N})=\sum_{\boldsymbol{C}} \prod_{t=1}^{T} P(N_t|C_t) \tag{18}$$

(c) Harmony + chord progression

$$P(\boldsymbol{N}) = \sum_{\boldsymbol{C}} P(C_0)P(N_0|C_0)\cdot \cdot\prod_{t=2}^{T} P(N_t|C_t)\,P(C_t|C_{t-1}) \tag{19}$$

(d) Note duration model only

$$P(\boldsymbol{N})=P(N_0)\prod_{t=2}^{T} P(N_t|N_{t-1}) \tag{20}$$

(e) Duration + harmony

$$P(\boldsymbol{N})=\sum_{\boldsymbol{C}} P(C_0)P(N_0|C_0)\prod_{t=2}^{T} P(N_t|N_{t-1},C_t) \tag{21}$$

(f) Duration + harmony + chord progression

$$P(\boldsymbol{N}) = \sum_{\boldsymbol{C}} P(C_0)P(N_0|C_0)\cdot \cdot\prod_{t=2}^{T} P(N_t|N_{t-1},C_t)\,P(C_t|C_{t-1}) \tag{22}$$

The variants are presented graphically in Fig. 10. The frontier algorithm was used to evaluate the likelihood for model variants with hidden variables. Each model was evaluated by calculating the cross-entropy, *i.e.* the normalized log-likelihood of the data $\boldsymbol{N}$ given the model:

$$E(\boldsymbol{N})=-\frac{1}{T}\log_2 P(\boldsymbol{N}) \tag{23}$$

4.3 Note detection

To evaluate the results of multiple frequency estimation, the F-measure was calculated by comparing the detected notes with the ground truth. A note was considered detected (*true positive*) if its onset was within 100 ms from a true note onset. By measuring the number of true positives, *false positives* (spurious notes) and *false negatives* (undetected notes), the *precision, recall* and F-measure were calculated.

Fig. 11 depicts preliminary note detection results obtained for 7 different models. The first two models were simple thresholding with -40 dB (optimal threshold, deter-

mined empirically) and -70 dB (crossing point between the output probability distributions). In the third model the note were detected based on the trained output probability, but only from the reduced solution space. This means that no prior on the notes was present (no language model) and this model was equivalent to the model (a) from subsection 4.2. The last 4 models correspond to the ones described in subsection 4.2, but with the note variables hidden and the note salience layer on the bottom. The proposed model performed not worse than thresholding and generally yielded better recall, but worse precision. The results for RWC-C24A were significantly improved over the thresholding, which can be attributed to the fact that this piece is played on a harpsichord, which has very strong overtones that were mistaken for pitches. The proposed model was able to remove most of these thanks to the prior distribution on the notes.

5. CONCLUSION

We have proposed a uniform probabilistic framework that estimates note onsets and pitches from a salience matrix obtained by a pitch estimation front-end. The model was evaluated on symbolic (MIDI) data and, preliminarily, on audio signals. The results show significant improvement of the model over a reference model with uniformly and independently distributed notes, with the biggest improvement coming from using temporal dependencies.

Compared to the thresholding, the estimation was more robust and yielded higher precision, though the recall was sometimes lower.

In future we plan to focus on improving the accuracy and, therefore, the impact of the simultaneous pitch model $P(\boldsymbol{N}_t|C_t)$. We would also like to explore the possibilities of unsupervised training that would allow us to use a much larger training set, but also investigate the influence of the chosen chord dictionary size (for example, commonly used chord dictionaries are: 24, 48 [10], 168 [8] and 288, *i.e.* 12 keys × 24 chords, but even larger dictionaries are possible).

6. ACKNOWLEDGMENT

This work is supported by INRIA under the Associate Team Program VERSAMUS (http://versamus.inria.fr/).

7. REFERENCES

[1] A. Cemgil, H. Kappen, D. Barber, S. Netwerken and N. Nimegen: "A generative model for music transcription," *in IEEE trans. ALSP*, vol. 14, nr. 2, pp. 679–694, 2006.

[2] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Music Genre Database and Musical Instrument Sound Database," *in Proc. ISMIR*, pp. 229–230, 2003.

[3] D. Kawakami, H. Kaneko, S. Sagayama: "Developing functional harmony labeled data and its statistical analysis," *in Proc. ASJ Spring Meeting*, p. 2, 2010. *(in japanese)*

[4] A. Klapuri: "Multiple fundamental frequency estimation by summing harmonic amplitudes," *in Proc. ISMIR*, pp. 216–221, 2006.

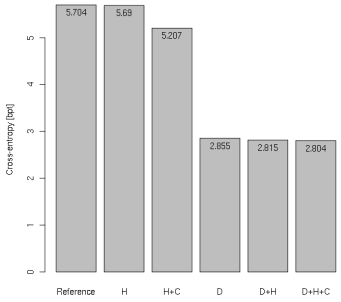


Figure 12: *Average cross-entropy between symbolic data and different variants of the model.*

[5] K. Murphy: "Dynamic bayesian networks: representation, inference and learning," *PhD thesis,* 2002.

[6] A. Pertusa and J.M. Iñesta: "Multiple fundamental frequency estimation using Gaussian smoothness," *in Proc. ICASSP*, pp. 105–108, 2008.

[7] S. Raczyński, N. Ono, S. Sagayama: "Extending Nonnegative Matrix Factorization – a discussion in the context of multiple frequency estimation of musical signals," *in Proc. EUSIPCO*, pp.934–938, 2009.

[8] C. Raphael and J. Stoddard: "Harmonic Analysis with Probabilistic Graphical Models," *in Proc. ISMIR*, pp. 177–181, 2003.

[9] M. Ryynänen and A. Klapuri: "Modelling of Note Events for Singing Transcription," *Proc. ITRW*, 2004

[10] Y. Ueda, Y. Uchiyama, T. Nishimoto, N. Ono, S. Sagayama, "HMM-based Approach for Automatic Chord Detection Using Refined Acoustic Features," *in Proc. ICASSP*, 2010.

[11] E. Vincent, N. Bertin, and R. Badeau: "Adaptive Harmonic Spectral Decomposition for Multiple Pitch Estimation," *in IEEE Trans. ASLP*, vol. 18, nr. 3, pp. 528–537, 2010

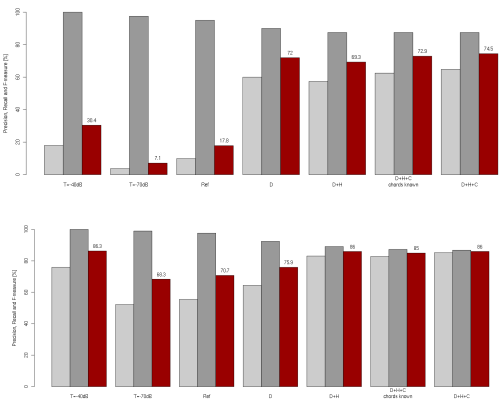


Figure 11: *Note detection results for 7 different models obtained for RWC-C24A (top, α = 2.3, L = 12, N = 70) and RWC-C22 (bottom, α = 1.3, L = 12, N = 75).*

MODIFIED AIS-BASED CLASSIFIER FOR MUSIC GENRE CLASSIFICATION

Noor Azilah Draman, Campbell Wilson and Sea Ling
Caulfield School of IT,
Faculty of Information Technology, Monash University,
Caulfield East, Melbourne, Victoria, 3145
nadra1@student.monash.edu.au
campbell.wilson@infotech.monash.edu.au
chris.ling@infotech.monash.edu.au

ABSTRACT

Automating human capabilities for classifying different genre of songs is a difficult task. This has led to various studies that focused on finding solutions to solve this problem. Analyzing music contents (often referred as content-based analysis) is one of many ways to identify and group similar songs together. Various music contents, for example beat, pitch, timbral and many others were used and analyzed to represent the music. To be able to manipulate these content representations for recognition: feature extraction and classification are two major focuses of investigation in this area. Though various classification techniques proposed so far, we are introducing yet another one. The objective of this paper is to introduce a possible new technique in the Artificial Immune System (AIS) domain called a modified immune classifier (MIC) for music genre classification. MIC is the newest version of Negative Selection Algorithm (NSA) where it stresses the self and non-self cells recognition and a complementary process for generating detectors. The discussion will detail out the MIC procedures applied and the modified part in solving the classification problem. At the end, the results of proposed framework will be presented, discussed and directions for future work are given.

1. INTRODUCTION

Music genre is defined as classes or groups of songs that categorizes a collection of songs that have similar characteristics. It is a label created by music experts so that these songs are easily described and recognized [1]. There have been various studies on music genre classification over the years where generally the focuses would be on the type of features extracted, feature extraction techniques, feature selection mechanisms, and feature classification algorithms. This is because music genre classification is a unique topic, and an investigation that tries to imitate human capability to identify music. It is a process to automate the human skills in recognizing and grouping different type of music into categories by using their hearing senses and logical judgment.

Our research is also about automating the human identification process where we are investigating an algorithm from Artificial Immune System (AIS), called the modified immune classifier (MIC). MIC is a modification of negative selection algorithm, introduced in writer identification study [2]. Negative selection algorithm is one of a few algorithms developed in AIS domain where it stresses the antigen recognition process. Two processes involved: monitoring, a process of recognizing self/non-self cells by performing the affinity binding, and censoring, the process where antibodies (also known as detectors) are randomly generated to match with the antigens. The recognized antigens are called self cells whereas the non-recognized antigens are known as non-self cells. In the human immune system, recognized antigen is referring to cells that prevent human body from disease and non-recognized antigens are referring to cells that bring diseases to human body. MIC eliminates the process to generate detectors randomly, which is the main aspect of the NSA, by introducing a complementary process. This complementary process will define self cells based on how many classes of data they need to identify and then generate the detectors accordingly.

However, to be able to apply the modified immune classifier in this research, which is to be able to identify and recognize different groups of music genre, we need to change some part of the classifier in order to achieve high accuracy of results. We will discuss the changes that we have made later.

We present this paper with the intention of discussing music genre classification that applies modified immune classifier in the classification process. We are discussing in detail the feature extraction and feature selection processes except to explain the features used in the experimental work and the techniques used to select relevant and significant features. We elaborate the AIS approach in the context of music genre classification, their consequences in music recognition performances whether the approach will have a major impact to the classification performances.

We organize the remainder of this paper as follows: Section 2 discusses previous research in music genre recognition. Section 3 discusses the MIC and the changes part, the censoring and monitoring stages, and how these stages relate to the feature extraction, selection, and classification. Section 4 then will be discussing the experimental setup and the classification results. We outline some concluding remarks in the last section.

2. BACKGROUND OF STUDY

In the music genre identification and classification studies, initiated research was to solve problems that occur

during recognition such as, deciding which song belongs to which genre. [3], for example, did an early work of classifying songs into different categories of genre using human auditory skills. Since then, many studies to find solutions to increase the automation performances occurred. Various recorded attempts to solve this problem are in [2] – [9]. Not only the problem of automating the process of classification but the question of how to fill the gap of accuracy behind human skilled classification [3] also need to be answered and solved.

[1] contributed by introducing new music features from pitch, timbre and rhythm contents. Their experiments on genre classification have shown that their attempts can be investigated further as the classification accuracy results were around 61 percent only. The focus of their research was to introduce a new range of music features for music genre classification. As the extracted features are too numerous, many irrelevant and insignificant features were used in their experiments that contributed to the low level of performances.

[10] introduced a new technique to extract music features called Daubechies Wavelet Coefficient Histograms (DWCHs) with a purpose to overcome the classification accuracy problems in the previous study. The authors used the Daubechies wavelet filter, *Db8*, to decompose music signals into layers where at the end of each layer they constructed histograms of coefficient wavelet. During experiments they combined their new feature with [1] features and improved the results but not by much.

There is also another attempt that used pitch, rhythm and timbre contents to classify music into different genres [11]. In this study, the author used the neural network based classifier which was not tested in the previous two studies. Again similar problem that related to the classification performance occurred. The experiments have shown that the accuracy was quite high when the classification processes were to recognize one or two genres only. But, as the classes of genres increased, the performances began to decrease.

[12] proposed a solution to the problem mentioned above. The authors proposed a new feature extraction method called *InMAF*. This new method was quite different from previous approaches where previously, they relied mostly on the spectrum characteristics of music content. *InMAF* on the other hand integrated the acoustic features and the human musical perception into music feature vectors to increase the classification performances. The classification results were so impressive that the achieved accuracies were as high as ninety percent. However, these outcomes were the results of a combination of this new method with pitch, rhythm and pitch contents. There is no classification result from any individual features recorded in the study.

[8] attempted to classify the music genre using MIDI (Musical Instrument Digital Interface) and audio features, such as pitch, rhythm and timbre features by using the data from [13], which contained two different sets of features, the first was MIDI features and the other group was the audio features. However the attempt was not that successful as the result did not show any major improvement in the classification performances.

A new recent study proposed a new approach to classify music genre by emphasizing the features from cepstral contents, such as MFCCs, OSC and MPEG 7 representations [14]. They introduced a novel set of features that were derived from modulation spectral analysis of the spectral representations, and these features were the *Mel-Frequency Cepstral Coefficients* (MFCC), *Octave-based Spectral Contrast* (OSC), *Normalized Audio Spectral Envelope* (NASE) and *Modulation Spectral Analysis* of MFCC, OSC and NASE. Their experiments were conducted on individual features and combinations of features.

The results were very good, where the combination of features tested were able to achieve the accuracy around twenty percent higher than any studies that we have discussed so far. That was an impressive achievement since low classification accuracy is the major problem faced by the domain.

3. AIS-BASED CLASSIFIER

In this part, we discuss Artificial Immune System (AIS) approach specifically on the modified negative selection algorithm (MIC) to classify the music genre. According to [15], the human immunology system inspired this domain to observe the immune functions, models, and principles of immunology. Some references on AIS-based classification task can be found in [16 -17].

AIS are adaptive systems, emulating human body immunology system to solve problems. It is concerned with abstracting the whole concept of immune system to computational systems in solving problems from mathematics, engineering, and information technology point of view. AIS is developed based upon a set of general purposes algorithms that are modelled to generate artificial components of the human immune system. [15] defined AIS as an adaptive system which is enthused by biological immunology and observed functions, principles and models to problem solving.

[18] introduced negative selection algorithm as inspired by negative selection of T-cells in thymus. The algorithm focused on recognizing self or non-self cells where it eliminated the T-cells which thymus does not recognized. Detail explanations of how negative selection algorithm works is in [19]. As has been investigated before, it would be impossible to apply NSA without modification as each problem and solutions are different. However, we will not discuss the NSA further as it is not in the research scope.

In the next section, we will discuss the MIC, the censoring and monitoring stages including features conversion, complementary and identification processes that we have applied to suit with the problem in hand. Then we continue the discussion with detailed explanation of the changes that we have made in the identification accuracy calculation.

3.1 Modified Immune Classifier (MIC)

The inspiration to investigate MIC in this research comes from a writer identification study [2] where the proposed

classifier to identify different writers has provided excellent results as the identification test achieved the accuracy as high as 99 percent. The recognition is evaluated by emphasizing the affinity binding or similarities between those cells.

In this new version of NSA, the author introduced a complementary process, which is a process of generating detectors to detect antigens. Originally, in NSA, the detectors are randomly generated and cost some time. They also do not contain enough information to recognize the whole range of antigens.

This would become a problem because, in order to recognize the antigens, the generated detectors shall not be created randomly as the process will not guarantee there will be enough detectors. By having the complementary process, the detectors will be generated accordingly to compensate the antigens as has been done in the writer identification research where the complementary process generated detectors according to the number of writers that should be recognized. Imitating the immune system's function, MIC works:

(1) self-cells (feature vectors) are transformed into antibodies (detectors) to detect similar cells (antigen),
(2) during detection (identification) antibodies will do the affinity binding with the antigens (finding similarities),
(3) both cells will bind (matched) if there are similarities occurred – antibodies detected antigens as similar to it cells – a pattern is recognized

As has been mentioned earlier, censoring and monitoring modules are two important process of MIC. We will discuss them next.

3.2 Censoring and monitoring modules

Censoring module is responsible to produce detectors, which is the key aspect of identification. This module normally starts after feature extraction and feature selection processes. It involves data feature conversion where the features will be represented by binary bit strings (for example, a feature vector, -3.4523123 is converted into a binary string, 101011001 using –XOR operation). After the conversion, the binary bit strings then will go through the complementary process and become the detectors.

We applied the supervised learning experiments in this research and we used training data to generate the detectors. Once generated, we used them in the classification process by comparing the detectors and generated antigens (we converted testing data into antigens). The comparison occurred in the monitoring module (the training model/detectors created earlier to predict the testing data/antigens) and it was to find matched data between detectors and antigens. If matched, we then calculate the affinity binding.

The comparison produced binary bit '1' or '0' where bit '1' means the data is bind. However, in this scenario, we will use the word 'match' instead of 'bind' to define the similarities. Figure 3.1 illustrates both modules where two important things occurred in censoring module, which are the conversion data from feature vectors into binary bit strings using –XOR and detectors generated processes. In monitoring module, two important things also occurred, which are antigens generated from testing data and identification processes. During binary matching process, we used Hamming distance technique to calculate matched binary bits.

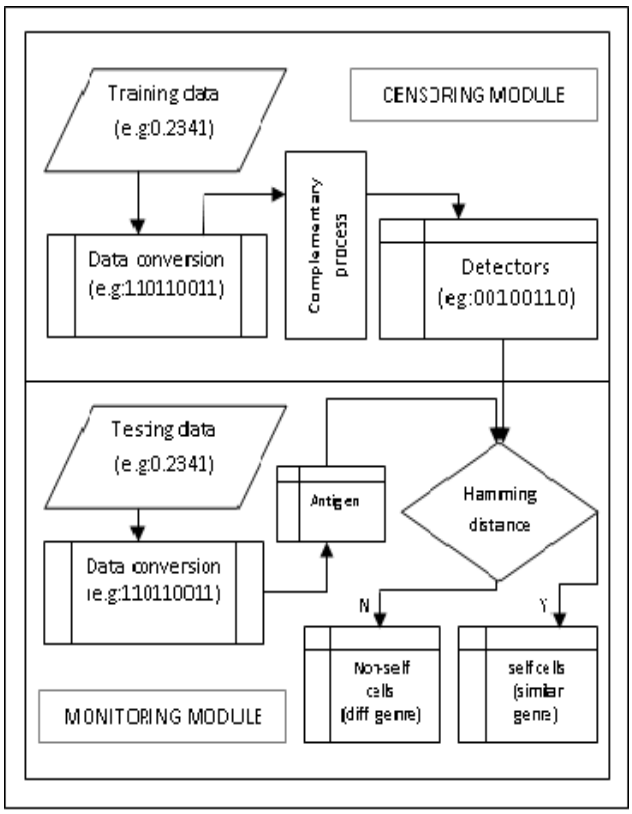


Figure 1. Censoring and monitoring modules

3.3 Accuracy calculation

In the writer identification problem, the calculation emphasized the recognition of each feature where these features will be calculated individually based on a threshold value. The accuracy would be based on how many features were correctly classified. To apply MIC to our problem, we concentrated on the threshold value in the accuracy calculation where the value will be our benchmark to decide whether the songs are classified accurately or not.

During the process, we calculated the result first by combining all features and produced the data accuracy percentage. Then we compared the accumulated value with the threshold value percentage. If the percentage of the combined features is higher than the threshold value, the data then is labeled as accurately classified. The following Table 3.1 and 3.2 will show the difference between the writer identification calculation and ours.

The difference between the original MIC proposed in [2] with ours is that we combined all the feature vectors as one whole data and calculates the matched bits before we compare them with the threshold value, whereas in the author identification study, the matched bit is calculated

separately for each feature and the accuracy is computed based on the total amount of features that exceeded the threshold value.

Category	Calculation formulas
Feature matching stage	Num_of_bit_match $\geq$ threshold
Image accuracy stage	(Num_of_feature_match / num_of_feature) $\times 100$
Data accuracy stage	(Num_of_genre_match / num_of_testing_data) $\times 100$

Table 1. The writer identification calculation

Category	Calculation formulas
Data genre accuracy %	$\sum$ bits_matched / $\sum$ features_bits $\times$ 100
Threshold (***r***) %	($\sum$ ***r*** * num_of_features / $\sum$ bits_per_feature * num_of_features) $\times$ 100
Dataset accuracy	(Num_of_genre_match / num_of_testing_data) $\times 100$

Table 2. The music genre accuracy calculation

4. EXPERIMENTS

In this section, we explain our conducted experiments to evaluate the proposed algorithm**.**

4.1 Datasets

We used Latin music datasets which contains 3160 music pieces in MP3 format classified in 10 different musical genres [20][21]. The songs were grouped into 10 genres: Tango, Bolero, Batchata, Salsa, Merengue, Axé, Forró, Sertaneja, Gaúcha and Pagode. The extracted music features were from timbre contents (containing MFCC, spectral centroid, roll-off, flux, time domain zero crossings), pitch-histograms related features and beat calculated features. The features were extracted using MARSYAS [22] where the combined total of the features produced 30 vectors for each song.

We have prepared training and testing datasets where similar data is used in the experiments except that the data for WEKA experiments was in the attribute related file format (ARFF) and in the data file (DAT) format for MIC demonstrations.

4.2 Feature selection technique

We have used WEKA tool to select relevant and significant features. We used filter approach in this study because it is more practical and suitable for our problem as the approach is independent and work separately from the classifier. The filter approach also works faster than wrapper and embedded approaches.

We have selected significant features using two search approaches, which are the best first search algorithm and the greedy hill search algorithm. The techniques that we used to do the best first search selection and the greedy hill selection are the FilterSubsetEval, the CFSSubsetEval and the ConsistencySubsetEval. The produced selected features from these techniques contained 13, 17, and 18 feature vectors.

We tested the MIC algorithm in the classification processes by defining the threshold value as 12. The reason is that we want to compare the proposed MIC with other classifiers without evaluating various threshold values to select the best one. The chosen threshold value is considered practical and enough to determine the reliability of the proposed technique.

Table 3 describes the feature vectors in detail where they have been numbered (1 to 30) for easy identification.

Features	Description
1 - 6	Beat-related features (peak histograms, amplitude and period)
7 - 25	Timbral features (mean and standard deviation of spectral centroid, rolloff, flux, zero crossings, MFCC, low energy)
26 - 30	Pitch related features (folded and unfolded histograms, period, amplitude pitch interval of unfolded histograms)

Table 3. Features description

4.2 Classification

For comparison purposes, we used classifiers from Waikato Environment for Knowledge Analysis (WEKA) [23] and the MIC algorithm that we have built using C++ language. We have chosen few classifiers from different category in WEKA.

We have setup the experiment cases according to the selected features from selection process. We also have setup experiments to test individual group of features and combinations between the groups. The reason is that we want to test the robustness of our program and the reliability of AIS-based classifier performance in our classification problems. Table 4, 5, and 6 will explain these cases in detail.

Cases	Description
C1	Features 1, 2, 6, 9,10, 13, 17, 18, 22, 25, 26, 28
C2	Features 1, 4, 6, 7, 9, 10, 12, 13, 14, 15, 16, 17, 18, 21, 23, 26, 28
C3	Features 1, 4, 6, 9, 10,11, 12, 13, 14, 15, 16, 17, 18, 21, 22, 23, 25, 26
C4	Contains all 30features

Table 4. List of selected features

Cases	Description
F1	Features 1 – 6 (beat related features only)
F2	Features 7- 25 (timbral related features only)
F3	Features 26 – 30 (pitch related features only)

Table 5. Individual group of features

Cases	Description
FBP	Combination of beat and pitch related features
FBT	Combination of beat and timbral related features
FTP	Combination of timbral and pitch related features

Table 6. Combination of group features

4.3 Results

Table 7, Table 8, and Table 9 list all classification results that we have obtained from the prepared cases classification experiments.

Technique Cases	Case 1	Case 2	Case 3	Case 4
BayesNet	50.00%	53.00%	53.00%	58.33%
SMO	46.00%	48.67%	57.00%	56.33%
IB1	49.00%	51.67%	56.00%	57.00%
Bagging	42.33%	44.00%	47.67%	48.33%
J48	38.00%	38.33%	42.00%	42.00%
MIC	**99.33%**	**95.00%**	**92.67%**	**73.00%**

Table 7. Selected features cases

Technique Cases	F1	F2	F3
BayesNet	29.67%	29.67%	49.33%
SMO	27.33%	31.33%	50.00%
IB1	27.33%	**100 %**	50.33%
Bagging	30.33%	66.33%	53.00%
J48	28.67%	72.00%	45.00%
MIC	**100 %**	**100 %**	**93.33%**

Table 8. Individual group of features cases

Technique Cases	FBP	FBT	FTP
BayesNet	39.3333%	53.0000%	54.3333%
SMO	33.3333%	57.6667%	56.3333%
IB1	38.3333%	55.3333%	56.0000%
Bagging	35.0000%	49.3333%	52.6667%
J48	37.3333%	40.3333%	48.3333%
MIC	**99.00%**	**79.33%**	**91.33%**

Table 9. Combination of group features cases

In Table 7, for feature selection cases, all cases except for the data without feature selection, MIC has obtained the accuracies over 90% compared to other classifiers. The performances of other classifiers did not show any significant improvement compared to MIC.

Table 8, which is referring to the individual group of features experiments. Overall performances for each feature when tested with various classifiers have shown that beat related features produced the lowest accuracy results. Timbral related features came in second however, when tested with MIC classifier pitch and timbral features produced similar percentages. Bagging classifier also produced similar result when tested the timbral related features to classify the songs.

In Table 9, WEKA classifiers produced almost similar results when we experimented with both beat+timbral related and timbral+pitch related features. The lowest accuracy recorded with beat+pitch related features when these features were used for classification. However, the opposite case occurred when the data were classified using MIC classifier because the lowest accuracy recorded when beat+timbral related features were tested.

5. CONCLUSION

The availability of techniques and methods for classification in music analysis field proved that researchers in this area are very concerned with the performance. As the collections of digital songs keep increasing online, their studies have contributed a major breakthrough to the internet users and others.

In this paper, we have experimented and explained the proposed MIC in different category of cases. In each experiment, MIC has outperformed almost every classifier except for Bagging technique where in one of the cases, the result is exactly similar to what MIC has produced. The obtained results have clearly shown that MIC is a new prospective approach for music genre classification. It has been proven the proposed classifier in music recognition research has surpassed other classifiers and the improvement of classification accuracy is phenomenal. The results also showed that among the features, timbral has provided us good classification result in the most cases except for the combined features cases.

We strongly believe that our discussion throughout this paper has given opportunities to other researchers in this area of studies to fill the gaps, to explore further and to provide solutions to the known and un-known problem that has yet to be discovered. Future work will include an investigation on how to manage efficiently the threshold value and probably later on, exhaustive search approach should be applied to evaluate the highest threshold value that can provide high classification accuracies.

6. REFERENCES

[1] G. Tzanetakis and P. Cook, “Musical genre classification of audio signals”, IEEE Transactions on Speech and Audio Processing, vol.10, no.5, pp. 293-302, 2002.

[2] Draman, A. K. “Authorship invarianceness for writer identification using invariant discretiation and modified immune classifier”, PhD thesis. University of Technology Malaysia, 2009.

[3] Lippens, S., Martens, J. P. & Mulder, T. D. “A comparison of human and automatic musical genre classification” *Acoustics, Speech, and Signal Processing,* 2004.

[4] Brecheisen, S., Kriegel, H. P., Kunath, P. & Pryakhin, A. “Hierarchical genre classification for large music collections”. *2006 IEEE International*

Conference on Multimedia and Expo - ICME 2006, Vols 1-5, Proceedings, 1385-1388, 2006.

[5] Ahrendt, P., Larsen, J. & Goutte, C. "Co-occurrence models in music genre classification". *2005 IEEE Workshop on Machine Learning for Signal Processing (MLSP)***,** 247-252, 2005.

[6] Bagci, U. & Erzin, E. "Boosting classifiers for music genre classification". *Computer and Information Sciences - Iscis 2005*, Proceedings, 3733, 575-584, 2005.

[7] Bagci, U. & Erzin, E. "Inter genre similarity modeling for automatic music genre classification". *2006 IEEE 14th Signal Processing and Communications Applications*, Vols 1 and 2**,** 639-642, 2006

[8] Cataltepe, Z., Yaslan, Y. & Sonmez, A. "Music genre classification using MIDI and audio features", *Eurasip Journal on Advances in Signal Processing*, 2007.

[9] Cheng, H. T., Yang, Y. H., Lin, Y. C., Liao, I. B. & Chen, H. H. "Automatic Chord Recognition for Music Classification and Retrieval". *Ieee International Conference on Multimedia and Expo,* Vols 1-4**,** 1505-1508, 2008.

[10] T. Li, M. Ogihara and Q. Li, "A comparative study on content-based music genre classification", *Proceedings of the 26th annual international ACM SIGIR*, Toronto, Canada, pp. 282-289, 2003.

[11] R. Neumayer and A Rauber, "Integration of text and audio features for genre classification in music information retrieval", In *Proceeding of 29th European Conference on Information Retrieval, Rome, Italy,* pp. 724-727, 2007.

[12] Shen, J. L., Shepherd, J. & Ngu, A. H. H. "On efficient music genre classification". Database Systems for Advanced Applications, Proceedings, *3453, 253-264.*

[13] Mckay, C. & Fujinaga, I. "Musical genre classification: Is it worth pursuing and how can it be improved*". ISMIR 2006*. Victoria Canada.

[14] Lee, C. H., Shih, J. L., Yu, K. M. & Lin, H. S. "Automatic Music Genre Classification Based on Modulation Spectral Analysis of Spectral and Cepstral Features". *Ieee Transactions on Multimedia,* 11, 670-682, 2009.

[15] L.N. de Casto and J. Timmis, "*Artificial immune system: A new computational intelligence approach",* Great Britain, Springer, pp. 76-79, 2002.

[16] S. Doraisamy, S. Golzari, N. M. Norowi, M. N. Sulaiman and N. I. Udzir, "A study on feature selection and classification techniques for automatic genre classification of traditional Malay music", in *Proceedings of Ninth International Conference on Music Information Retrieval (ISMIR'08)*, Philadelphia, Pennsylvania USA, pp. 331-336, 2008.

[17] D.N. Sotiropoulus, A.S. Lampropoulus and G.A. Tsihrintzis, "Artificial immune system-based music genre classification", in *New Directions in Intelligent Interactive Multimedia*, pp. 191-200, 2008.

[18] R.-B. Xiao, L. Wang and Y. Liu, "A framework of AIS based pattern classification and matching for engineering creative design", in *Proceedings of the First International Conference on Machine Learning and Cybernetics*, Beijing, China, pp. 1554-1558, 2002.

[19] S. Forrest, A.S. Perelson, L. Allen and R. Cherukuri, "Self-nonself discrimination in a computer", in *Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, USA, pp. 202-212, 1994.

[20] C.N. Silla, A.L. Koerich and C.A.A. Kaestner," The Latin Music Database"*Proceedings of Ninth International Conference on Music Information Retrieval (ISMIR'08)*, Philadelphia, Pennsylvania USA, 2008, pp. 331-336.

[21] C.N Silla, A.L Koerich ans C.A.A Kaesner. "A feature selection approach for music genre classification". *International Journal of Semantic Computing,* Vol 3, No 2, 183 -208, 2009.

[22] G. Tzanetakis, P.Cook, "MARSYAS: A Framework for Audio Analysis". *Organized Sound Journal.* Vol 4 (3), 2000.

[23] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten, "The WEKA Data Mining Software: An Update"; *SIGKDD Explorations*, Volume 11, Issue 1, 2009.

MONOPHONIC INSTRUMENT SOUND SEGREGATION BY CLUSTERING NMF COMPONENTS BASED ON BASIS SIMILARITY AND GAIN DISJOINTNESS

Kazuma Murao Masahiro Nakano Yu Kitano Nobutaka Ono Shigeki Sagayama
Graduate School of Information Science and Technology
The University of Tokyo, Japan
{ k-murao, mnakano, kitano, onono, sagayama } @hil.t.u-tokyo.ac.jp

ABSTRACT

This paper discusses a method for monophonic instrument sound separation based on nonnegative matrix factorization (NMF). In general, it is not easy to classify NMF components into each instrument. By contrast, monophonic instrument sound gives us an important clue to classify them, because no more than one sound would be activated simultaneously. Our approach is to classify NMF components into each instrument based on basis spectrum vector similarity and temporal activity disjointness. Our clustering employs a hierarchical clustering algorithm: group average method (GAM). The efficiency of our approach is evaluated by some experiments.

1. INTRODUCTION

In music signals, there are usually multiple sound sources such as a human singing voice and instruments sound. The task to separate mixed signals into individual sources is called sound source separation for music signals. It has several applications such as music equalizer, music information searching, automatic transcription, and structured coding of music. This paper discusses a method to separate monaural musical audio into individual musical instruments.

Sound source separation for music signal has been widely investigated recently. Some methods are based on supervised learning of individual source models [1–3]. They need solo excerpts beforehand. Other unsupervised approaches have also been studied [4–6]. Because any prior information for instrumental sound sources cannot be used, some unsupervised methods make assumption about common harmonic structure [4, 5] or employ the excitation-filter model of sound production [6]. We propose an efficient unsupervised method focusing on monophonic instrument sound.

Our method have two stages. At the first stage, we factorize the observed spectrogram into some components based on nonnegative matrix factorization (NMF) [9, 10]. In the case of music signals, each component usually represents a musically meaningful element, so that different elements are expected to correspond to different components.

However, considering music instrumental source separation, methods based on NMF generally encounter difficulties in the components clustering step. And most of the algorithm count on manual clustering [7]. Some clustering methods separate percussive instrument sources [8,12], but are rarely used with harmonic instruments sources.

This paper proposes a method for clustering components that employs not only spectral information but also temporal information. The outline of this paper is as follows. Section 2 gives a overview of NMF algorithm and component-clustering problem. The proposed clustering method is explained in Section 3, and experimental evaluation of proposed method are presented in Section 4. Section 5 covers the conclusions and future works.

2. NONNEGATIVE MATRIX FACTORIZATION

Nonnegative matrix factorization and some unsupervised sound source separation algorithms are based on a signal model where the spectrum vector $\mathbf{x}_t$ ($t = 1, ..., T$) in frame is modeled as a linear combination of *basis vectors* $\mathbf{b}_j$ ($j = 1, ..., J$). This can be written as

$$\mathbf{x}_t = \sum_{j=1}^{J} g_{j,t}\mathbf{b}_j, \tag{1}$$

where J is the number of basis vectors, and its time-varying gain (amplitude) $g_{j,t}$, T being the number of frames.

This model can be written using a matrix notation as

$$\mathbf{X} = \mathbf{BG}, \tag{2}$$

where $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_T]$, $\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_J]$, and $[\mathbf{G}]_{j,t} = g_{j,t}$.

Here, $\mathbf{g}_j = [g_{j,1}, ..., g_{j,t}]^{\mathbf{T}}$ is defined as *gain vector* corresponding to the basis vector, then the term *component* refers to one basis vector $\mathbf{b}_j$ and one corresponding gain vector $\mathbf{g}_j$. Each source is modeled as a sum of the components. The separation is done by first factorizing the spectrogram of the input signal into components and second grouping these to sound sources.

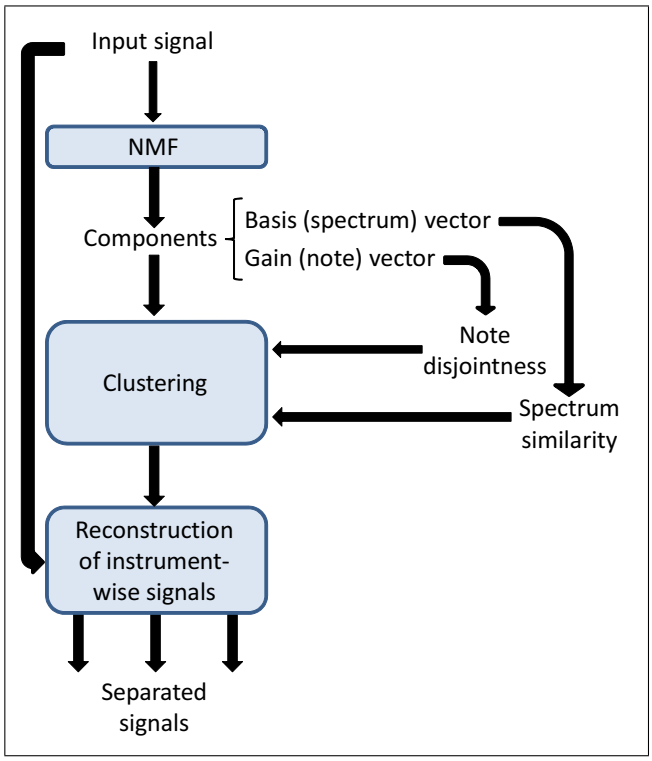


Figure 1. Flow diagram of the method.

The NMF algorithms proposed by Lee and Seung [9] do the decomposition by minimizing the reconstruction error between the observation and the model while constraining the matrices to be entry-wise nonnegative as follows:

$$D(\mathbf{X}\|\mathbf{BG}) = \sum_{f,t} d([\mathbf{X}]_{f,t}\|[\mathbf{BG}]_{f,t}), \qquad (3)$$

here $d(y\|z)$ is a function of two scalar variables. The various measures for reconstruction error are proposed. The Euclidean distance, the generalized Kullback-Leibler divergence [9], or the Itakura-Saito divergence [11] are mostly used. We choose here the generalized Kullback-Leibler divergence, which has produced good results in earlier sound source separation studies [14].

In standard NMF, the only constraints is the elementwise non-negativity of all matrices. Then, several constraints have been proposed in order to achieve expected solutions. The most famous constrains are sparsity [13] and temporal continuity [11, 14]. We use the sparsity and temporal continuity proposed in [14].

We wish to use NMF to decompose the observed signal into the components. However, it is not easy to know which source each component is assigned to. In the next section an automatic clustering method is proposed.

3. CLUSTERING OF NMF COMPONENTS

3.1 Outline

As a result of NMF, basis vectors $\mathbf{b}_j$ and gain vectors $\mathbf{g}_j$ are obtained, each of which could ideally represent spectrum and temporal activity of each note, respectively. The problem here is how to classify obtained components $(\mathbf{b}_j, \mathbf{g}_j)$ into each instrument. The contribution of this paper is to exploit both information of $\mathbf{b}_j$ and $\mathbf{g}_j$) for mixture of monophonic instrumental tracks without any prior about each instrument. Our approach consists of 1) measuring the basis spectrum similarity $C_1(i,j)$ for any pairs of $\mathbf{b}_j$ and $\mathbf{b}_j$, 2) measuring the temporal activity disjointness $\tilde{C}_2(i,j)$ for any pairs of $\mathbf{g}_i$ and $\mathbf{g}_j$, 3) calculating a closeness measure $C(i,j)$ for any pairs of $(\mathbf{b}_i, \mathbf{g}_i)$ and $(\mathbf{b}_j, \mathbf{g}_j)$ by product of $C_1(i,j)$ and $\tilde{C}_2(i,j)$, and 4) applying a kind of hierarchic clustering method.

3.2 Similarity of Basis Spectra

Monophonic source signal is represented by a sinusoidal model [15] as

$$s(t) = \sum_{r=1}^{R} A_r(t)\cos[\theta_r(t)] + e(t) \qquad (4)$$

where $e(t)$ is the noise term, $A_r(t)$ and $\theta_r(t) = \int_0^t 2\pi r f_0(\tau)d\tau$ are the instantaneous amplitude and phase of the rth harmonic, respectively, $f_0(\tau)$ is the fundamental frequency at time τ, and R is number of the harmonic overtone. Harmonic structure is an approximately invariant feature for a harmonic instrument when it is played in a narrow pitch range. [16]

In logarithmic frequency (log-frequency) scale, the harmonic frequencies are located $\log 2$, $\log 3$, ..., away from the log-fundamental frequency, and the relative-location relation remains constant no matter how fundamental frequency fluctuates and is an overall parallel shift depending on the fluctuation degree. Thus among the harmonics between the two spectrums of the same instruments are similar; even in case spectrums fundamental frequencies are different, shapes of the spectrums are same when shifted.

The basis vector $\mathbf{b}_j$, which NMF factorize into, represents average spectrum in logarithmic frequency scale. Therefore the correlation-like criterion between two basis vectors are defined as

$$C_1(i,j) = \max_q \frac{\sum_p b_{p+q,i} b_{p,j}}{|\mathbf{b}_i||\mathbf{b}_j|}, \qquad (5)$$

where $b_{p,j}$ is pth value of the basis vector $\mathbf{b}_j$. Put another way, criterion $C_1(i,j)$ means maximum cross-correlation between normalized $\mathbf{b}_i$ and $\mathbf{b}_j$. In intuitive explanation, two spectra are compared, moving along the frequency axis, and are measured largest overlap. For the spectra by harmonic instrument, two spectrums overlap most when two fundamental pitches nearly go over. As a side-effect, two spectrums by inharmonic instruments mark higher value than value between harmonic and inharmonic instrumental spectrums.

Table 1 shows an example of this correlation-like criterions that is calculated by real instrumental signals: RWC music database [17] RWC-MDB-I-2001 No.31-1 and No.33-1, down-sampled to 16 kHz single-channel files. Each spectrum is taken by Wavelet transform of single tone signal. Two spectra of same instrument almost mark higher

	Clarinet			Flute		
	A4	H4	C5	A4	H4	C5
Clarinet A4	**1.00**	**0.96**	**0.81**	0.58	0.67	**0.81**
Clarinet H4		**1.00**	0.74	0.63	0.66	0.72
Clarinet C5			**1.00**	**0.82**	0.73	**0.92**
Flute A4				**1.00**	**0.95**	**0.80**
Flute H4					**1.00**	**0.80**
Flute C5						**1.00**

Table 1. The simirarity measure of basis spectra calculated by indivisual instrumental signals. The higher values than 0.8 are shown in bold style.

value than two spectrums of other instrument mark. However in some cases two spectra which belonging to other instrument mark high numerical number: for example, Clarinet C5 and Flute C5. This result presents that criterion as basis spectrum similarity (5) indicates measure to some extent, but is not enough for the grouping.

3.3 Disjointness of Temporal Activities

Not only basis spectrum $\mathbf{b}_j$, but also temporal activity $\mathbf{g}_j$ should also include cues for clustering components into instrumental tracks. As a simple case to exploit such information, we suppose that all instrumental tracks are *monophonic*, which means each instrumental track consists of a single note sequence.

Figure 2 shows an example of piano-roll representation of three monophonic instrumental tracks. Obviously, any different note activities are disjoint in the same track. Note that there are also many pairs of disjoint note activities over different tracks. Hence, we can't assert that two different note activities belong to the same track even if they are disjoint. However, if two different note activities are NOT disjoint, they should belong to different instrumental tracks.

The disjointness of two different temporal activities represented by gain vectors $\mathbf{g}_i$ and $\mathbf{g}_j$ can be simply calculated by

$$C_2(i,j) = 1 - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{|\mathbf{g}_i||\mathbf{g}_j|}. \qquad (6)$$

If $\mathbf{g}_i$ and $\mathbf{g}_j$ are disjoint, $C_2(i,j) = 1$. While if they have co-occurrence, $C_2(i,j)$ should take a small value. Therefore, it can be exploited as a closeness measure. Figure 3 shows an expected result, which was calculated by (6) with using temporal activities in piano roll representation shown in Figure 2 as $\mathbf{g}_j$.

The magnitude of $C_2(i,j)$ itself is not significant because it depends on the frequency of the co-occurence. It is only important for clustering whether it is almost zero or not. Furthermore, because of imperfect decomposition by NMF, spectral leakage, reverbration, etc, $C_2(i,j)$ is actualy not equal to zero even if ith component and jth component belong to the same instrumental track. Therefore, we 1) neglect tiny values of $g_{t,j}$ and set them to be zero, 2) calculate $C_2(i,j)$ by (6), and 3) binarize it with a small

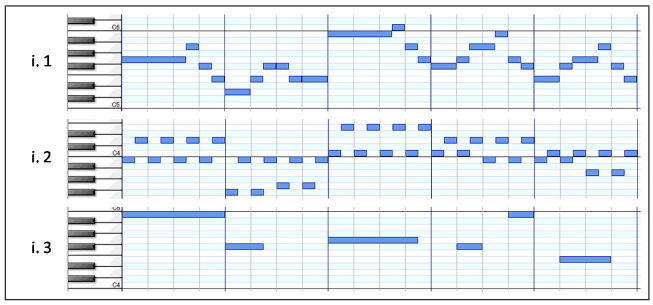


Figure 2. The piano roll representation of three monophonic instrumental track. Any different note activities are disjoint in the same track.

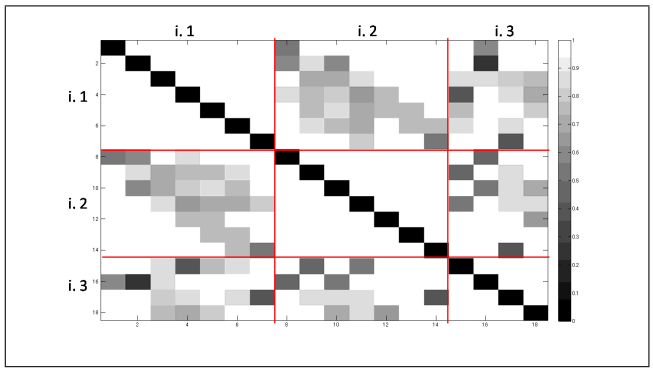


Figure 3. Criterions between two gain vectors according to the equation (6), corresponding to figure 2. The two vertical line and two horizontal line show the borderlines of the instruments. Values on diagram position are ignorable for the clustering.

threshold ϵ such as

$$\tilde{C}_2(i,j) = \begin{cases} 1 & (C_2(i,j) \geq \epsilon) \\ 0 & (C_2(i,j) < \epsilon) \end{cases}. \qquad (7)$$

3.4 Combining Two Different Criterions

Previous criterions are both scales running from zero to one. In both criterions, higher value means two components' sameness. This paper examines the measure of two different components' closeness as

$$C(i,j) = C_1(i,j) \cdot \tilde{C}_2(i,j). \qquad (8)$$

3.5 Clustering by Group Average Method

To find an optimal partitioning of the components into N classes, the following clustering algorithm called *group average method* (GAM) is employed.

1. At the beginning, all components are considered as different clusters.
2. Two components that have the highest criterion value are connected into the same (new) cluster.
3. Criterions between new cluster and other cluster are updated under the update rule:

$$d(K_1, K_2) = \frac{1}{n_1 n_2} \sum_{i \in K_1} \sum_{j \in K_2} d\,(i,j)\,, \qquad (9)$$

input data	sampling rate	16 kHz
	length	10 sec
	number of instruments	3
frequency analysis	frame shift	16 ms
	frequency resolution	12.0 cent
	frequency range	50–7961 Hz
NMF [9]	iteration	200
	number of components	10–40
Clustering	ϵ	0.05
	number of clusters	4

Table 2. Experimental conditions

where $d(A, B)$ is the criterion between cluster A and B, $d(i, j) = C(i, j)$ is the criterion between components i and j, n_1 and n_2 are the number of components that K_1 and K_2 contain.

4. Iteration: repeat step 2 and 3 until total number of clusters reaches L.

Criterion-update avoids chaining effect where wrong components connects into a chain reaction.

3.6 Reconstruction of Instrument-wise Spectrograms

Spectrograms corresponding to a certain instrument K_l ($l = 1, ..., L$) , $\hat{\mathbf{X}}_l$, can be reconstructed by the equation:

$$\hat{\mathbf{X}}_l = \sum_{j \in K_l} \hat{\mathbf{X}}^{(j)} = \sum_{j \in K_l} \mathbf{b}_j \mathbf{g}_j. \tag{10}$$

Spectrogram of instrument *l* is reconstructed as

$$[\hat{\mathbf{Y}}_l]_{f,t} = \frac{[\hat{\mathbf{X}}_l]_{f,t}}{[\hat{\mathbf{X}}]_{f,t}} [\mathbf{X}]_{f,t}. \tag{11}$$

where $\hat{\mathbf{X}} = \sum_{l=1}^{L} \hat{\mathbf{X}}_l$.

4. EXPERIMENTAL EVALUATION

4.1 Source Conditions

To verify the potential performance of the proposed method as sound source separation, the proposed method was tested on a real performance music data from *MIREX 2007 Evaluation Tasks* [18] : transcription of *String Quartet No.5 3rd Movement Var.5* composed by L. V. Beethoven (see table 2 for the list of the experimental data). We used the data composed of three woodwind instruments (flute, oboe and bassoon). Mixed signal was the result of summing the source signals in time domain, and 9 input signals (10 seconds) were clipped from the mixed signal every 5 seconds.

Time series of amplitude spectrum was analyzed using Gabor wavelet transform with a frame shift of $16\mathrm{ms}$ for input digital signals of $16\mathrm{kHz}$ sampling rate. The lower bound of the frequency range and the frequency resolution were $50\mathrm{Hz}$ and $12\mathrm{cent}$, respectively.

4.2 Evaluated Algorithms and Conditions

The following algorithms were tested.

- Proposed method 1: Components clustering employed both basis vector similarity and gain vector disjointness.

  Since there is no reliable method for the estimation of the number of the components, proposed method was tested by factorizing the input signal into 10–40 components and we decided it to earn the best result.

  In the clustering step, the number of the clusters was chosen as 4 because in the real performance music other than pure instrumental sound (e.g. sounds of breath) were contained.

- Proposed method 2: Components clustering employed only basis vector similarity. Compared with Proposed method 1, the contribution of the time activity disjointness can be evaluated.

- Correct clustering: Components clustering to be assigned each component to a source which leads to the highest signal-to-noise (SNR) as

  $$\mathrm{SNR}(m, j) = 10 \log_{10} \frac{\sum_{f,t} [\mathbf{Y}_m]_{f,t}^2}{\sum_{f,t} ([\mathbf{Y}_m]_{f,t} - [\hat{\mathbf{X}}^{(j)}]_{f,t})^2}. \tag{12}$$

  where $\mathbf{Y}_m$ and $\hat{\mathbf{X}}^{(j)}$ are the *m*th reference and *j*th separated component. A component *j* is assigned to a source *m* which leads to the highest SNR.

- NMF2D [4]: Factorization is doned by NMF2D instead of NMF. When analyzing real music signals, the NMF2D was considered to give good results.

4.3 Evaluation Criterion

The quality of the separated sources was measured by calculating the SNR improvement between the original spectrogram $\mathbf{Y}$ and corresponding separated magnitude spectrogram $\hat{\mathbf{Y}}$ according to the equation

$$\mathrm{SNR[dB]} = \frac{1}{M} \sum_{m=1}^{M} 10 \log_{10} \left(\frac{\sum_{f,t} [\mathbf{Y}_m]_{f,t}^2}{\sum_{f,t} ([\mathbf{Y}_m]_{f,t} - [\hat{\mathbf{Y}}_l]_{f,t})^2} - \frac{\sum_{f,t} [\mathbf{Y}_m]_{f,t}^2}{\sum_{f,t} ([\mathbf{Y}_m]_{f,t} - [\mathbf{X}]_{f,t})^2} \right). \tag{13}$$

For each original spectrogram, the SNR improvement that employs baseline using mixed signal are measured. The SNR has been used in several source separation studies to measure the separation quality.

4.4 Results

The SNR improvement for each data and algorithms are shown in table 3. Average values are means among all of data.

Proposed method 1 marks an average improvement 2.75 dB. For all data, proposed method 1 sets positive values.

	SNR [dB]			
	proposed 1	proposed 2	NMF2D	correct clustering
data (1): 0–10 sec	5.62	-9.05	-0.16	5.94
data (2): 5–15 sec	4.87	-0.71	-0.88	4.88
data (3): 10–20 sec	4.41	-8.88	-0.30	4.45
data (4): 15–25 sec	0.25	-6.23	-2.82	2.52
data (5): 20–30 sec	2.08	-2.86	-1.29	3.34
data (6): 25–35 sec	3.00	-7.82	-0.48	3.66
data (7): 30–40 sec	0.72	-3.17	-1.12	1.48
data (8): 35–45 sec	1.11	-5.71	-3.15	1.42
data (9): 40–50 sec	2.70	-13.13	-1.65	3.93
average	2.75	-6.40	-1.32	3.51

Table 3. SNR results of the evaluated algorithm in dB.

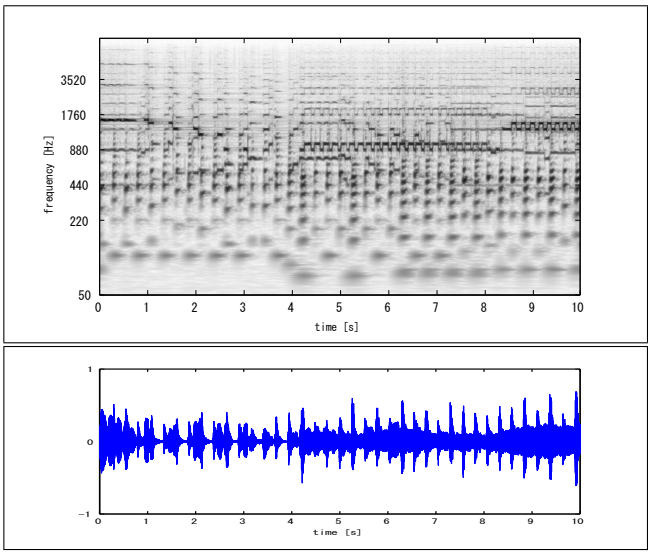


Figure 4. An input signal with three instrumental tracks (flute, oboe, and bassoon). Spectrogram (upper) and corresponding waveform (lower).

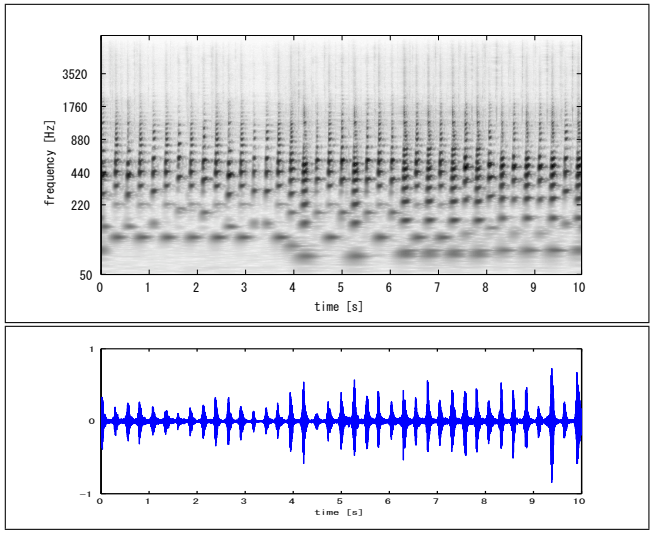


Figure 5. A source signal (bassoon track) of the mixture shown in figure 4.

The average improvement value of correct clustering is 3.54 dB. For two data (data (2) and data (3)) proposed method 1 and correct clustering mark almost same values. It shows that clustering step is maximally effective. In some other data proposed method sets close values to correct clustering.

Comparing SNR values between proposed method 1 and 2, it shows that in clustering step the contribution of the gain vector disjointness is effective.

The SNR values of NMF2D method are lower than that of proposed method 1. The reason is considered to be that, in these real music data, the NMF2D assumption that all notes for an instrument is an identical pitch shifted time-frequency signature does not hold.

Figures 4 , 5 and 6 show an example of experimental results: figure 4 is an input signal in which three instrumental signals (flute, oboe and bassoon) are mixed, figure 5 is a source signal with bassoon sounds, figure 6 is a separated signal which is corresponded to the bassoon's source signal. Even in other two instrumental sounds the results

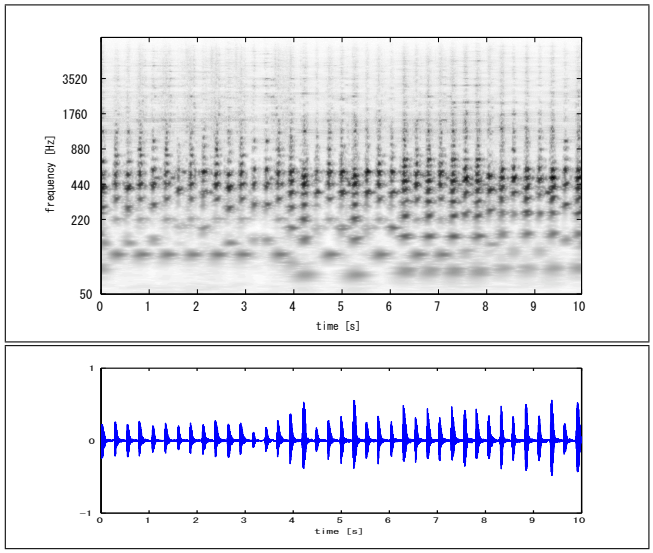


Figure 6. A separated signal (bassoon track) from the mixture shown in figure 4.

equaled to it.

5. CONCLUSION

This paper discussed a method for monophonic instrument sound separation. The method used nonnegative matrix factorization to factorize the spectrogram of the input signal into components. Then we introduced an criterion that measured two distinguish components: basis spectrum similarity and temporal activity disjointness. The grouping was done by clustering components under this measure. The experiment results showed that in some data the proposed method marked values equal to the correct clustering which employed source signals.

Future work includes the improvement of nonnegative matrix factorization by including the proposed criterion, that aims at accuracy enhancement of the decomposition.

6. ACKNOWLEDGEMENT

This research was supported by CrestMuse Project under JST and Grant-in-Aid for Scientific Research (KAKENHI) (A) 20240017.

7. REFERENCES

[1] M. Helén and T. Virtanen: "Separation of Drums from Polyphonic Music using Non-negative Matrix Factorization and Support Vector Machine," in *Proc. ISMIR*, pp. 337–344, 2005.

[2] F. R. Bach and M. I. Jordan: "Blind One-microphone Speech Separation: A Spectral Learning Approach," in *Proc. NIPS*, pp. 65–72, 2005.

[3] P. Leveau, E. Vincent, G. Richard, and L. Daudet: "Instrumentspecific Harmonic Atoms for Mid-level Music Representation," *IEEE Trans. Audio, Speech, and Language Processing*, Vol. 16, No. 1, pp. 116–128, 2008.

[4] M. N. Schmidt and M. Mørup: "Nonnegative matrix Factor 2-D Deconvolution for Blind Single Channel Source Separation," in *Proc. ICA*, pp. 700–707, 2006.

[5] K. Miyamoto, H. Kameoka, T. Nishimoto, N. Ono and S. Sagayama: "Harmonic-Temporal-Timbral Clustering (HTTC) For the Analysis of Multi-instrument Polyphonic Music Signals," in *Proc. ICASSP*, pp. 113–116, 2008.

[6] A. Klapuri, T. Virtanen and T. Heittola: "Sound Source Separation in Monaural Music Signals using Exitation-filter Model and EM Algorithm," in *Proc. ICASSP*, pp. 5510–5513, 2010.

[7] B. Wang and M. D. Plumbley:" Investigating single-channel audio source separation methods based on non-negative matrix factorization, "in *Proc. ICA*, pp. 17–20, 2006.

[8] M. A. Casey and A. Westner: "Separation of Mixed Audio Sources by Independent Subspace Analysis," in *Proc. ICMC*, pp. 154–161, 2000.

[9] D. D. Lee and H. S. Seung: "Algorithms for Non-negative Matrix Factorization," Advances in *Proc. NIPS*, Vol. 13, pp. 556–562, 2000.

[10] P. Smaragdis and J. C. Brown: "Non-negative Matrix Factorization for Polyphonic Music Transcription," in *IEEE Workshop on Applications of Signal Process. Audio Acoust.*, New Paltz, NY, pp. 177–180, 2003.

[11] C. Févotte, N. Bertin, and J.-L. Durrieu: "nonnegative matrix factorization with the Itakura-Saito divergence. With application to music analysis," *Neural Computation*, Vol. 21, No. 3, pp. 793–830, 2009.

[12] C.Uhle, C.Dittmar and T. Sporer: "Extraction of Drum Tracks from Polyphonic Music using Independent Subspace Analysis," in *Proc. ICA*, pp. 843–848, 2003.

[13] P. O. Hoyer: "Non-negative Matrix Factorization with Sparseness Constraints," *J. Mach. Learning Res.*, Vol. 5, pp. 1457–1469, 2004.

[14] T. Virtanen,: "Monaural Sound Source Separation by Non-Negative Matrix Factorization with Temporal Continuity and Sparseness Criteria," *IEEE Transactions on Audio, Speech and Language Process.*, Vol. 15, No. 3, pp. 1066–1074, 2007.

[15] X. Serra: "Musical Sound Modeling with Sinusoids Plus Noise," in *Musical Signal Processing*, C. Roads, S. Popea, A. Picialli and G. D. Poli, Eds. London, U.K.:Swets & Zeitlinger, 1997.

[16] M. Kim and S. Choi: "Monaural Music Source Separation: Nonnegativity, Sparseness, and Shift-invaliance," in *Proc. ICA*, pp. 617–624, 2006.

[17] M. Goto, H. Hashiguchi, T. Nishimura and R. Oka, "RWC music database: music genre database and musical instrument sound database," *Proc. ISMIR*, pp. 229–230, 2003.

[18] MIREX 2007 Evaluation Tasks: "Multiple Fundamental Frequency Estimation & Tracking," http://www.music-ir.org/mirex/2007/index.php/Multiple_Fundamental_Frequency_Estimation_&_Tracking, 2007.

MULTIPLE VIEWPOINTS MODELING OF TABLA SEQUENCES

Parag Chordia
Georgia Tech
Center for
Music Technology
Atlanta, GA, USA
ppc
@gatech.edu

Avinash Sastry
Georgia Tech
Center for
Music Technology
Atlanta, GA, USA
asastry3
@gatech.edu

Trishul Malikarjuna
Georgia Tech
Center for
Music Technology
Atlanta, GA, USA
tmallikarjuna3
@mail.gatech.edu

Aaron Albin
Georgia Tech
Center for
Music Technology
Atlanta, GA, USA
aalbin3
@mail.gatech.edu

ABSTRACT

We describe a system that attempts to predict the continuation of a symbolically encoded tabla composition at each time step using a variable-length n-gram model. Using cross-entropy as a measure of model fit, the best model attained an entropy rate of 0.780 in a cross-validation experiment, showing that symbolic tabla compositions can be effectively encoded using such a model. The choice of smoothing algorithm, which determines how information from different-order models is combined, is found to be an important factor in the models performance. We extend the basic n-gram model by adding viewpoints, other streams of information that can be used to improve predictive performance. First, we show that adding a short-term model, built on the current composition and not the entire corpus, leads to substantial improvements. Additional experiments were conducted with derived types, representations derived from the basic data type (stroke names), and cross-types, which model dependencies between parameters, such as duration and stroke name. For this database, such extensions improved performance only marginally, although this may have been due to the low entropy rate attained by the basic model.

1. INTRODUCTION AND MOTIVATION

When listening to music, humans involuntarily anticipate how it will continue [8]. Such expectations help to process information efficiently, as well as allowing complex, noisy stimuli to be accurately interpreted. For musicians, this anticipation is essential for synchronization and harmonization. In this paper, we explore a computational model of this predictive process based on an ensemble of n-gram models. Specifically, we examine whether such a model can successfully represent the structure of symbolically encoded tabla compositions. Our motivation for building a predictive tabla model is to enable more intuitive modes of interaction between musicians and computers.

In addition to this practical goal, we hope to work towards developing a computational model of musical anticipation. Previous work [11] on Western melodies showed that human judgments of melodic continuation were highly correlated with a variable-length n-gram model. Although we will not address human subject data here, we hope to provide converging evidence from a markedly different musical tradition (tabla), that syntactic structure can be efficiently represented using an n-gram modeling approach.

2. BACKGROUND AND RELATED WORK

Markov and n-gram models have been extensively used to model temporal structure in music [1]. They have been extensively used in algorithmic composition, timbral analysis [2] [7], structure analysis [12], and music cogniton [14].

Markov models are based on a succession of states. In musical contexts, states represent discretely valued attributes, such as pitch, duration, instrument, section, etc. The Markov assumption assumes that, given the current state, the next state is independent of previous states. This can easily be generalized so that the next state depends on a fixed number of past states; a first-order Markov chain depends only on the current state, a second-order on the current and immediately preceding state, and so on. If sequences are directly observable, then most inference problems can be solved by counting transitions. An alternative formulation is the n-gram model in which all possible symbols of length n are constructed from the training sequences, and their frequency tabulated. It is easy to see that the transition probabilities for an nth-order Markov chain can be computed by forming all $n+1$-grams.

A significant problem that arises with fixed-order models is that, as the order n increases, the number of total n-grams increases as v^n, where v is the number of symbols. In music applications, such as melody prediction, where the past ten events could easily influence the next event, and where there might be a dozen or more symbols, we are left attempting to assess the relative frequency of greater than 12^{10} n-grams. Even for large databases, most n-grams will be unseen, leading to the so-called zero frequency problem [10]. This sparsity problem leads to a

fundamental tradeoff between using the predictive power of longer context and the increasing unreliability of higher order n-gram counts. Variable-length n-gram models attempt to overcome this problem in two ways: 1) by building many fixed-order models and integrating information across orders (smoothing), 2) and by reserving a certain amount of probability mass for unseen n-grams (escape probabilities). We describe these techniques in Section 4.2.

Variable length n-gram modeling is an ensemble method in which the predictions of many fixed-order models are integrated. Ensemble methods such as boosting have been shown to be effective for classification tasks [16]. Multiple viewpoint systems, introduced by Conklin [5], and developed by others such as Witten [5] and Pearce [15] can be thought of further generalizing the idea of integrating an ensemble of predictive models. The extension is based on the fact that music can be simultaneously represented in many ways. For example, a melody can be thought of in terms of chromatic pitches, intervals, scale degrees, or contour. A rhythmic pattern can be thought of in terms of onset times, durations or position-in-bar. If, for example, we are trying to predict the next note in a melody, having multiple representations is useful in capturing structure that is obvious given one representation, but less so in another. For example, a scale-degree representation of a melody might make it obvious that the chromatic pitch, say B, is actually the leading tone, making it very likely that the next note is C. However, if the training database contains many melodies in many different keys, this might not be obvious from the chromatic pitch representation. We describe the multiple viewpoints framework in Section 4.3.

Little work to date has been done on statistical modeling of tabla. Gillet [7] and Chordia [4] both used an HMM framework for tabla transcription, while Bel and Kippen [9] created a model of tabla improvisation based on a context-free grammar, one of the earliest computational tabla models.

Tabla is the most widely used percussion instrument in Indian music, both as an accompanying and solo instrument. Its two component drums are played with the fingers and hands and produce a wide variety of timbres, each of which has been named. A sophisticated repertoire of compositions and theme-based improvisations has developed over hundreds of years. Although tabla is primarily learned as part of an oral tradition, it is also notated using a system that indicates strokes and their durations. Unfortunately, the correspondence between strokes and names is not one-to-one. Depending on the context and the stylistic school, the same stroke will be given different names. And, in some cases, different strokes will be given the same name. This is unproblematic in the context of an oral tradition but requires that care be taken when interpreting symbolic notations.

3. TABLA DATABASE

The database used for training the model is a set of traditional tabla compositions compiled by tabla maestro Alok Dutta [6]. The compositions were encoded in a Humdrum-based syntax called **bol that encoded the stroke name and duration [4]. The database which is available online consists of 35 compositions in a variety of forms. Altogether there are 27,189 strokes in the dataset, composed of 42 unique symbols.

4. N-GRAM MODELING

N-gram modeling is a commonly used technique to probabilistically model sequences of elements such as phonemes in speech, letters in a word or musical notes in a phrase. [13] N-grams can be efficiently stored in a tree-shaped data structure, commonly referred to as a trie or prefix tree. Figure 1 is the trie for the sequence ABAB+C. In such a trie, branches represent the succession of certain symbols after others, and a node at a certain level of the trie holds a symbol from the sequence, along with information about the symbol such as the number of times it was seen in the sequence following the symbols above it, and the corresponding probability of occurrence. In Figure 1, the subscript below a symbol represents the symbols probability given the context, defined by the path through the trie to that node, while the superscript above it represents the count value. Thus, in the topmost level, the probabilities represent the priors for the symbols. During construction of the trie, symbols are fed sequentially into the system one-by-one. For the above example, after the sequence ABAB, the trie looks like Trie1 in figure Figure 1. When a new symbol 'C' follows, corresponding nodes are created at all levels of the trie: 5-gram node using 'ABABC', 4-gram node using 'BABC', trigram node using 'ABC', bigram node using 'BC' and a 1-gram/prior entry for 'C' at the topmost level. The corresponding probabilities are also updated resulting in Trie 2 in Figure 1.

After the trie has been built in this manner, it can be used to predict the next symbol given a test sequence. This is done by following the nodes of the trie downwards from its top, in order of the symbols in the test sequence until the last symbol in the sequence (and the corresponding node in the trie) is reached. At that point, the probabilities associated with the children nodes represent the predicitve distribution over the symbol set, given the observed context. To allow for new symbols that may appear in the test sequence and to subsequently allow for a better matching of test sequences with missing or extra symbols compared to training sequences, we incorporate the concept of escape probabilities into our trie structure, as described in [17]. The above example trie would then look like Trie3 in figure Figure 1. We describe the use of escape probabilities in section 4.2. For long training sequences, the depth of the trie can become large and is often restricted to a maximum order to limit memory usage and to speed prediction given a test sequence.

The modeling and evaluation framework was implemented in C++ as an external object in Max/MSP along with supporting patches.

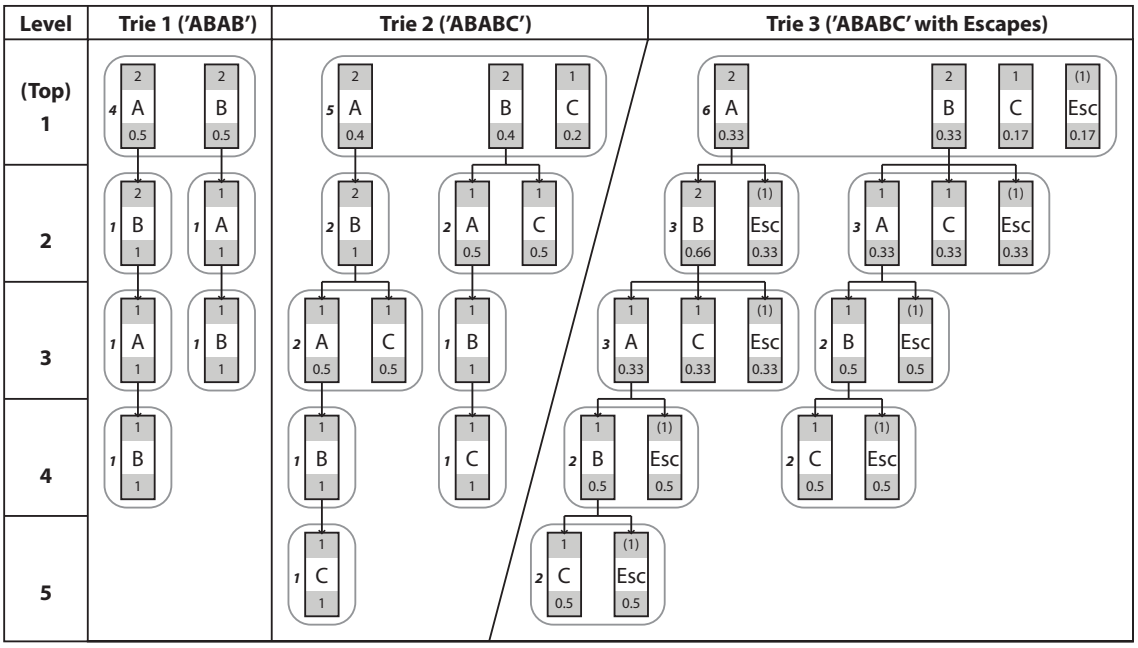


Figure 1. Illustration of tries built for the sequence 'ABAB' followed by the symbol 'C'. Superscripts represent count values, and subscripts represent probability values. Rounded boxes represent siblings, while italicized number at the left of a rounded box represents the total count among the siblings, which is used to calculate the 'probability' values. Trie 3 includes escape probabilities.

4.1 Escape Probabilities

As noted above, the zero frequency problem occurs because, in high-order models, most n-grams will never have been observed [10]. Using a simple counting scheme, the model would assume zero probability for these unseen events, thereby returning infinite entropy should they occur in the test sequence. The solution is to reserve a small amount of probability mass for events that haven't occurred yet. This is done by reserving an escape proabability for each level of the trie. Whenever an event returns zero probability, it returns the escape probability instead. There are many ways to assign the escape probability. Based on the results of Bell and Witten [17], we have implemented the Poisson distribution method. The escape probability for each level is assigned by $e(n) = \frac{T_1(n)}{N(n)}$, where T_1 is the number of tokens that have occurred exactly once and N is the total number of tokens seen by the model so far.

4.2 n-gram Smoothing

Smoothing addresses the tradeoff between the specificity of higher-order models (if a match can be found) and the reliability of the n-gram counts for lower-order models. Since higher order models are much sparser, many n-grams will be assigned zero probability, and counts for n-grams that have been observed will tend to vary greatly based on the particular training database. This variance can be reduced by incorporating information from lower order models. There are two basic types of smoothing algorithms: backoff models and interpolation models. Given a test sequence, a backoff model will search for the entire sequence, and if no match is found in the trie, the process continues recursively after dropping the first element of the sequence. The process stops once a positive match is found and the count for that n-gram count is greater then some threshold. Interpolated smoothing, by contrast, always incoporates lower order information even if the n-gram count in question is non-zero.

For this study, two smoothing methods were primarily used, Kneser-Ney (KN) and an averaging method we term $1/N$. These were also compared to a simple backoff procedure. KN was adopted because earlier work showed it to be a superior smoothing method in the context of natural language processing [3]. The basic idea of KN is to ensure that lower order distributions are only used when there are few, or no, counts in the higher order models. When incorporating lower information, the probability is related not to the true count of the n-grams but rather is proportional to the number of different n-grams that it follows. An example in music might be as follows: given a bigram consisting of two rhythmic durations, where the second duration is transitional and not typically used on its own, we would not assign a high unigram probability since it is only used in association with the first duration. Implementation details can be found in [3].

Given a model, with M as the maximum order, the weights for each model are given by $w(n) = \frac{1}{m(maxOrder-n)}$. In other words, the higher orders receive greater weight than the lower orders. It is worth noting what happens in the case where a higher-order model has not seen a particular n-gram. In that case, even though the weight for that model will be relatively higher than for a lower order model, the probability of the n-gram, which will be determined by the escape probability, will be very small, and

typically much smaller than the weight.

4.3 Multiple Viewpoints

Our focus in the work so far has been to implement a multiple viewpoints system for music analysis, apply these principles to traditional North Indian tabla compositions, and identify the set of parameters that work best on this kind of music. Though we will touch upon the basics of the multiple viewpoint system, a more detailed explanation can be found here [5].

Conventional context-dependent predictive models track only the most basic aspects of music, like pitch, rhythm and onset times. Moreover, these variables are tracked together, so that finding an exact match for every possible context is practically impossible. A multiple viewpoints system, however, tracks each variable independently, maintaining many predictive models simultaneously. The final prediction is obtained by combining all these predictions into a meaningful set of basic parameters (such as pitch and duration). Such a system not only incorporates information from different variables but can also model complex relationships between two or more of these variables and make use of that information to strengthen its prediction. Furthermore, a multiple viewpoint system can make much better predictions in rare event cases, because of its ability to find context matches in at least one of its many models.

A viewpoint is nothing more than a set of events of a particular type. For example, a set of all pitch classes (C, C#, D, D# and so on until B) would consititute a viewpoint for a melody. Similarly, a viewpoint for rhythm would consist of the set of all onset times within a measure. These two viewpoints, pitch and rhythm, can be directly extracted from the music, are independent of each other and are called *basic types*. *Cross types* are formed when two or more basic types are combined and tracked simultaneously (T1 x T2). A cross type formed using Notes and Onset Times would consist of all elements in the Notes viewpoint in combination with all elements in the Onset Times viewpoint. Each element of this viewpoint is represented as a tuple {Note, OnsetTime}, instead of a single value. The number of all possible elements in a cross type is equal to the product of the number of elements in each basic type. A *derived type* depends on information extracted from a basic type. A simple example of this is melodic intervals, which are extracted from pitches. Derived types can use information from more than one viewpoint, and this can lead to the formation of cross types derived from derived types. Selection of appropriate representations is domain dependent and often uses prior knowledge of the music.

Here we use two basic types – strokes and durations. We also look at the following cross types: 1) Strokes x Durations and 2) Strokes x PositionInBar (PIB), where PIB refers to the onset position of a stroke as a fraction of the bar. Finally we introduce three derived types into the model. These were constructed by mapping the stroke names to a reduced set. Reduced Set 1 was made by eliminating different names for the same stroke, reducing the number of symbols from 41 to 32. Reduced Set 2 extended this idea by mapping acoustically similar strokes to the same name, which further reduced the number of symbols to 10. The open/closed mapping was made by classifying each stroke as resonant or non-resonant.

4.4 Merging Model Predictions

An important point here is the actual process of merging the predictions of each of the models. Though there are many different ways to do this, we use a weighted average as described in [15]. Each viewpoint model is assigned a weight depending on its cross-entropy at each time step. The weight for each model is given by $w_{\mathrm{m}} = H(p_{\mathrm{m}})/H_{\max}(p_{\mathrm{m}})$, where $H(p_{\mathrm{m}})$ is the entropy of the probability distribution and $H_{\max}(p_{\mathrm{m}})$ is the maximum entropy for a prediction in the distribution. Higher entropy values result in lower weights. In this way, models that are uncertain (i.e., have higher entropy) make a lesser contribution to the final distribution. The distributions are then combined by taking their weighted average.

4.5 Long Term and Short Term Models

A common limitation of such predictive models built on large databases is that the model is usually unaware of any patterns specific to a particular song. The model becomes too general to be effective, and very often patterns and predictions which seem obvious to humans are missed because they are infrequent in the global training database. To solve this problem, we used two models: a long-term model (LTM) built on the entire training database, and a short-term model that starts out empty and is built up as a particular composition is processed. In this work, the LTM is not updated as the test composition is processed.

When a composition is read, both models return a distribution over the symbol set at each time step. The predictions are merged into a final prediction using a weighted average as described above. Whenever the STM is uncertain, such as the beginning of a composition or new section, the system gives more weight to the LTM. In other sections, such as the end of a song, where the STM is more certain, the weighting scheme assigns more weight to the STM. A comparison of the cross-entropy measure for each model is presented in Table 1.

5. EVALUATION

Cross-validation was performed using a leave-one-out design. For each of the 35 compositions, training of the LTM was performed on the remaining 34. Reported results were averaged over all 35 trials. A common domain-independent approach for evaluating the quality of the models' predictions is cross-entropy [11]. If the true distribution is unknown, the cross entropy can be approximated by $-\frac{1}{n}\sum_{i=1}^{n}\log_2(p_i)$, which is the mean of the entropy values for a given set of predictions. To illustrate, at a given step t, we note the true symbol. We then look at the predictive distribution for symbols at step $t-1$ and calculate the entropy for the true symbol at step t. After running through

	Durations				Strokes				Stroke-Duration			
Order	Priors	Comb.	STM	LTM	Priors	Comb.	STM	LTM	Priors	Comb.	STM	LTM
1	1.891	1.049	0.916	1.890	3.614	3.184	2.958	3.613	4.965	3.883	3.397	4.962
5	1.891	0.563	0.510	0.897	3.614	1.117	0.994	1.780	4.965	1.294	1.162	2.354
10	1.891	0.469	0.429	0.814	3.614	0.868	0.814	1.609	4.965	1.060	0.995	2.220
20	1.891	0.383	0.356	0.736	3.614	0.805	0.780	1.584	4.965	1.007	0.966	2.201

Table 1. Summary of cross-entropy results for LTM, STM, and combined models for order 1-20

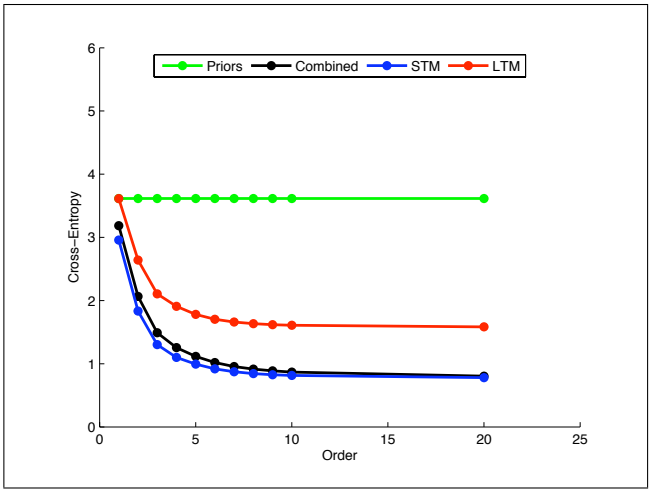


Figure 2. Cross-entropy for stroke prediction using LTM, STM, and combined models for orders 1-20

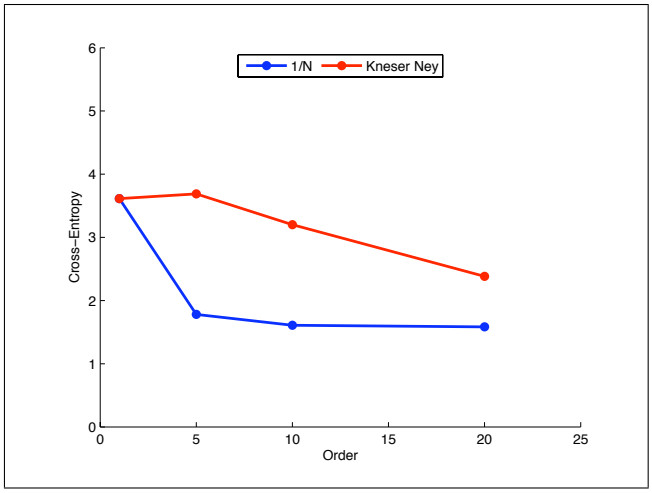


Figure 3. Comparison of Kneser-Ney and 1/N smoothing

all the symbols in the test set, these entropies are averaged, giving a cross-entropy result for that particular test set.

6. RESULTS

Figure 2 shows cross-entropy as a function of model order using $1/N$ smoothing. The cross-entropy for the order 20 stroke LTM is 1.584, which is a surprisingly good result given the number of symbols (42). Compared with using a predictive distribution based on the prior probability of each stroke, cross-entropy was reduced by 2.030 (from 3.614 to 1.584). The STM entropy rate for strokes was a remarkable 0.780. Combining the LTM and STM based on entropy, as described in section 4.3, did not improve performance over the STM alone. The STM also outperformed the LTM when predicting durations (0.365 vs. 0.736) and when jointly predicting the stroke and duration (0.966 vs. 2.201). In both cases, the combined model offered no overall performance increase. Not surprisingly, in some cases the LTM outperformed the STM at the beginning of the composition, before the STM model had seen much data.

As expected, cross-entropy decreases monotonically as a function of model order. The curve decays roughly exponentially, with performance improving dramatically from order 1 to order 5, significantly between 5 and 10, and little between 10 and 20. This suggests that, for these compositions, memorizing more than the past 10 strokes does little to improve predictions. This is true for the prediction of durations, strokes, and joint prediction of the stroke and duration.

Model	Durations	Strokes
Basic	0.814	1.609
Basic + SD	0.756	1.557
Basic + SD entropy	0.744	1.546
SD + PIB entropy	0.744	1.522

Table 2. Cross-types LTM where SD is stroke X duration, and PIB is position-in-bar. Merging of models was done using a weighted average with entropy-based weights, except for Basic + SD, which took the simple mean

Figure 3 shows the effect of different smoothing techniques on performance for the LTM. Both $1/N$ and Kneser-Ney smoothing significantly outperform a simple backoff method. $1/N$ is the clear winner for strokes, durations, and joint prediction. The difference in the quality of prediction decreases as the model size increases but is large throughout. Unusually, Kneser-Ney smoothing decreases slightly in performance as the model order is increased from 1 to 5.

Table 2 shows that cross-types had a small impact on performance with the addition of the stroke X duration type having the most impact.

In Table 3, we show the results for several LTM using derived stroke types, essentially more abstract sound categories based on the stroke name. For the LTM, Reduced

Derived Types	Strokes
Strokes only	1.60858
Strokes + Reduced Set 1	1.83609
Strokes + Reduced Set 2	1.81989
Strokes + Open/Close Set	1.61803

Table 3. Cross-entropy for stroke prediction using derived-types with LTM

Set 1 and 2 decreased performance by approximately 0.2, whereas open/closed marginally improved performance.

7. DISCUSSION

These results suggest that tabla compositions can be effectively encoded using a variable length n-gram model. Given a set of 42 stroke symbols, the best model's cross-entropy was 0.780, essentially meaning that it was on average uncertain between 2 strokes, a dramatic reduction from the 42 strokes in the vocabulary, as well as from the prior distribution which corresponded to approximately 12 strokes. Interestingly, the results suggest that tabla compositions exhibit strong local patterns that can be effectively captured using a STM, providing significantly better performance when compared with the LTM alone. Because many tabla compositions consist of a theme and variations, this result is not surprising. These data also suggest that it is almost always better to only use the STM, except for the very initial portion of the composition. Cross types seem to lead to small improvements, whereas derived types lead to small decreases. More experiments are needed in order to determine whether these changes are significant.

Another important result is that smoothing can have a large impact on predictive performance and seems to be highly domain dependent, with $1/N$ outperforming KN, a technique that had been shown to be amongst the best in another area. It is likely that the correct balancing of model order will depend on the database size and of course the distribution of n-grams. It would be interesting if further work could elucidate a clear theoretical basis for choosing a given smoothing method. In the absence of this, it is likely that performance could be improved by using a validation set and by adjusting how quickly weights fall off for interpolated smoothing techniques as the model order decreases.

8. FUTURE WORK

As always, we plan to continue to encode more tabla compositions to see if these results generalize. Additionally, we hope to test other merging methods such as geometric combinaton, a technique shown to be superior to additive combination in the context of melodies [11], as well as implementing cross and derived types for the STM. We also hope to use our trained models to generate novel tabla compositions and to use human evaluators to judge their quality. Lastly, we hope to use these results in an interactive tabla system that can anticipate and respond to a tabla improvisation.

9. REFERENCES

[1] C Ames. *The Markov Process as a Compositional Model: A Survey and Tutorial*. 1989.

[2] Jean-Julien Aucouturier, Franois Pachet, and M. Sandler. The way it sounds: timbre models for analysis and retrieval of music signals. 2005.

[3] Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *PROCEEDINGS OF THE 34TH ANNUAL MEETING OF THE ACL*, pages 310–318, 1996.

[4] Parag Chordia. *Automatic Transcription of Solo Tabla Music*. PhD thesis, Stanford University, December 2005.

[5] Darrell Conklin and Ian H. Witten. Multiple viewpoint systems for music prediction. 1995.

[6] Alok E Dutta. *Tabla: Lessons and Practice*.

[7] Olivier Gillet and Gael Richard. Supervised and unsupervised sequence modeling for drum transcription. In *Proceedings of International Conference on Music Information Retrieval*, 2007.

[8] David Huron. *Sweet Anticipation: Music and the Psychology of Expectation*. MIT Press, 2006.

[9] Bel B Kippen J. *Bol Processor Grammars In Understanding Music with AI*. AAAI Press, 1992.

[10] W J Teahan John G Cleary. Experiments on the zero frequency problem. 1995.

[11] Marcus Pearce Johnston. *The construction and evaluation of statistical models of melodic structure in music perception and cognition*. PhD thesis, City University, London, 2005.

[12] Kyogu Lee. Automatic chord recognition from audio using an hmm with supervised learning. In *In Proc. ISMIR*, 2006.

[13] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2002.

[14] Pearce, Herrojo Ruiz, Kapasi, Wiggins, and Bhattacharya. Unsupervised statistical learning underpins computational, behavioural and neural manifestations of musical expectation. 2010.

[15] Marcus Pearce, Darrell Conklin, and Geraint Wiggins. Methods for combining statistical models of music. 2004.

[16] Dietterich T.G. Ensemble methods in machine learning. 2000.

[17] Ian H. Witten and Timothy C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. 1991.

MUSIC GENRE CLASSIFICATION VIA COMPRESSIVE SAMPLING

Kaichun K. Chang
Department of Computer Science
King's College London
London, United Kingdom
ken.chang@kcl.ac.uk

Jyh-Shing Roger Jang
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
jang@cs.nthu.edu.tw

Costas S. Iliopoulos
Department of Computer Science
King's College London
London, United Kingdom
csi@dcs.kcl.ac.uk

ABSTRACT

Compressive sampling (CS) is a new research topic in signal processing that has piqued the interest of a wide range of researchers in different fields recently. In this paper, we present a CS-based classifier for music genre classification, with two sets of features, including short-time and long-time features of audio music. The proposed classifier generates a compact signature to achieve a significant reduction in the dimensionality of the audio music signals. The experimental results demonstrate that the computation time of the CS-based classifier is only about 20% of SVM on GTZAN dataset, with an accuracy of 92.7%. Several experiments were conducted in this study to illustrate the feasibility and robustness of the proposed methods as compared to other approaches.

1. INTRODUCTION

1.1 Acoustic Features for Audio Music Analysis

In the literature of music information retrieval (MIR), various content-based features have been proposed [1] for applications such as classification, annotation, and retrieval [15]. These features can be categorized into two types, that is, short-time and long-time features. The short-time features are mainly based on spectrum-derived quantity within a short segment (such as a frame). Typical examples include spectral centroids, Mel-frequency cepstral coefficients (MFCC) [1], and octave based spectral contrast (OSC) [2]. In contrast, the long-time features mainly characterize the variation of spectral shape or beat information over a long segment, such as Daubechies wavelet coefficients histogram (DWCH) [3], octave-based modulation spectral contrast (OMSC), low-energy, beat histogram [1], and so on. According to G. Tzanetakis et al. [1], the short and long segments are often referred to as "analysis window" and "texture window", respectively.

Theoretically, both short-time and long-time features should be used together to realize efficient and effective MIR system since they provide different information for the task under consideration. However, in practice, too many features usually degrade the performance since there might be some noises instead of useful cues in the feature set. Moreover, too many features could also entail excessive computation to downgrade the system's efficiency. As a result, we need an effective method for feature selection, extraction, or distillation. CS turns out to be an effective tool for such a purpose.

1.2 Compressive Sampling

CS is firstly proposed by Candès, Romberg, Tao and Donoho, who have showed that a compressible signal can be precisely reconstructed from only a small set of random linear measurements whose number is below the one demanded by the Shannon theorem Nyquist rate. It implies the potential of a dramatic reduction in sampling rates, power consumption, and computation complexity in digital data acquisitions. CS has proved to be very effective in imaging [6] [7], channel estimation [8], face recognition [9], phonetic classification [18], sensor array [19] and motion estimation [20].

In this paper, we propose a CS-based classifier with long-time and short-time features for music genre classification. The remainder of this paper is organized as follows. In section 2, the multiple feature sets used in the proposed method is briefly discussed. In the section 3, we describe multiple feature sets for audio music, and introduce the corresponding CS-based classifier. In section 4, experimental settings and results are detailed to demonstrate the proposed method's feasibility. Finally, conclusions and future work are addressed in the last section.

2. MULTIPLE FEATURE SETS

In the proposed method, multiple feature sets including long-time and short-time features are adopted for genre classification. These acoustic features include timbral texture features, octave-based spectral contrast (OSC), octave-based modulation spectral contrast (OMSC), modulation spectral flatness measure (MSFM), and modulation spectral crest measure (MSCM).

Timbral texture features are frequently used in various music information retrieval system [11]. Some timbral texture features, described in Table 1, were proposed for audio classification [1]. Among them, MFCC, spectral centroid, spectral rolloff, spectral flux, and zero crossings are

Table 1. Timbral texture features

Feature	Description
MFCC	Representation of the spectral characteristics based on Mel-frequency scaling [12]
Spectral centroid	The centroid of amplitude spectrum
Spectral rolloff	The frequency bin below which 85% of the spectral distribution is concentrated.
Spectral flux	The squared difference of successive amplitude spectrum.
Zero crossings	The number of time domain zero crossings of the music signal.
Low-energy	The percentage of analysis windows that have energy less than the average energy across the texture window.

short-time features, thus their statistics are computed over a texture window. The low-energy feature is a long-time feature.

Besides these features, OSC and OMSC features are also considered. OSC considers the spectral peak, spectral valley, and spectral contrast in each subband [2]. The spectrum is first divided into octave-based subband (as explained next). Then spectral peaks and spectral valleys are estimated by averaging across the small neighborhood around maximum and minimum values of the amplitude spectrum respectively. OMSC [1] is extracted using long-time modulation spectrum analysis [13].

In this paper, the amplitude spectrum of a music signal is divided into octave-based subbands of 0-100Hz, 100Hz-200Hz, 200Hz-400Hz, 400Hz-800Hz, 800Hz-1600Hz, 1600Hz-3200Hz, 3200Hz-8000Hz, 8000Hz-22050Hz. Within each subband, the amplitude spectrum is summed. Then for each subband, the modulation spectrum is obtained by applying the discrete Fourier transform (DFT) on the sequence of the sum of amplitude spectrum.

OMSC is obtained from spectral peaks and spectral contrasts of the modulation spectrum. MSFM and MSCM are obtained from a texture window [4] using the long-time modulation spectrum [13] that can describe the time-varying behavior of the subband energy. These features are also considered as parts of our multiple feature sets.

3. COMPRESSIVE SAMPLING BASED CLASSIFIER

As inspired by CS and the sparse signal representation theory, here we shall propose a CS-based classifier for genre classification. First of all, we shall cover the basics of the CS theory [5].

In Figure 1, consider a signal x (length N) that is K-sparse in sparse basis matrix Ψ, and consider also an $M \times N$ measurement basis matrix Φ, $M << N$(M is far less than N), where the rows of Φ are incoherent with the columns of Ψ. In term of matrix notation, we have $x = \Psi\theta$, in

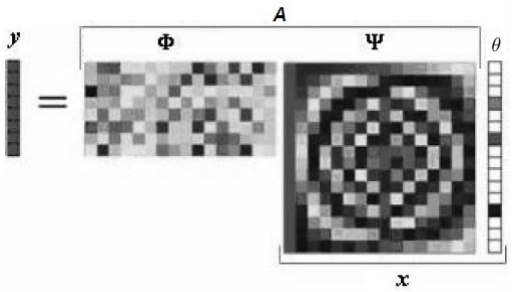


Figure 1. The measurement of Compressive Sampling

which θ can be approximated using only $K << N$ non-zero entries. The CS theory states that such a signal x can be reconstructed by taking only $M = O(K \log N)$ linear, non-adaptive measurement as follows:

$$y = \Phi \cdot x = \Phi \cdot \Psi \cdot \theta = A \cdot \theta, \quad (1)$$

where y represents an $M \times 1$ sampled vector, $A = \Phi\Psi$ is an $M \times N$ matrix. The reconstruction is equivalent to finding the signal's sparse coefficient vectors θ, which can be cast into a ℓ_0 optimization problem.

$$min \left\|\theta\right\|_0 \;\; s.t.\; y = \Phi \cdot x = A \cdot \theta \quad (2)$$

Unfortunately (2) is in general NP-hard, and an optimization ℓ_1 is used to replace the above ℓ_0 optimization [10].

$$min \left\|\theta\right\|_1 \;\; s.t.\; y = \Phi \cdot x = A \cdot \theta \quad (3)$$

Let the dimension of the extracted feature be denoted as m, and the extracted feature vector of the j-th music in the i-th class as $\nu_{i,j} \in R^m$. Moreover, let us assume there are sufficient training samples for the i-th class $A_i = [\nu_{i,1}, ...\nu_{i,n_i}] \in R^{m \times n_i}$. Then any new (test) sample $y \in R^m$ (i.e, the extracted feature of the test music) from the same class will approximately lie in the linear span of the training samples associated with object i:

$$y = \sum_{i=1}^{n_i} \alpha_{i,n_i} \, \nu_{i,n_i}, \quad (4)$$

for some scalars $\alpha_{i,j}(j = 1, .., n_i)$. Since the membership i (or the label) of the test sample is initially unknown, we define a new matrix $\mathbf{A}$ for the entire training set as the concatenation of the n training samples of all k classes: $\mathbf{A} = [A_1, ...A_k]$ Then the linear representation of y can be rewritten in terms of all training samples as:

$$y = \mathbf{A}x_0 \in R, \quad (5)$$

where $x_0 = [0, .., 0, \alpha_{i,1}, ..., \alpha_{i,n}, ..., 0, .., 0]^T \in R^n$ is a coefficient vector whose entries are zero except those associated with the i-th class. As the entries of the vector x_0 encode the identity of the test sample y, it is tempting to obtain it by solving the equation (4). This is called a sparse representation based classifier (SRC) [9].

In SRC, for a new test sample y from one of the classes in the training set, we first compute its sparse representation $\hat{x}$ via (2). Ideally, the nonzero entries in the estimate $\hat{x}$ will all be associated with the columns of $\mathbf{A}$ from a single object class i, and we can easily assign the test sample y to that class. To better harness such linear structure, we instead classify y based on how well the coefficients associated with all training samples of each object reproduce y. For each class i, let $\delta_i : R^n \rightarrow R^n$ be the characteristic function which selects the coefficients associated with the i-th class[12]. For $x \in R^n$, $\delta_i(x) \in R^n$ is a new vector whose only nonzero entries are the entries in x that are associated with class i. Using only the coefficients associated with the i-th class, one can approximate the given test sample y as

$$\hat{y}_i = A\delta_i(\hat{x}) \tag{6}$$

We then classified y based on these approximations by assigning it to the object class that minimizes the residual between y and $\hat{y}_i$:

$$\min r_i(y) = \|y_i - A\delta(\hat{x})\|_2 \tag{7}$$

The proposed CS-based classifier is based on the principle of SRC, with an additional random measurement on the extracted features to reduce the dimension of the input. According to the CS theory, this reduction can capture the structure of the features and automatically remove possible redundancy. The realization of the algorithm is summarized in Table 2.

It should be noted that SRC is a sparse representation based classifier, without the dimension reduction over the input signals. Here the random measurement of compressive sampling is used to perform a dimension reduction and feature extraction. So the classification complexity of CS based method is remarkably lower than that of SRC. Moreover, the multiple features will also improve the classification accuracy. The sparse representation is one part of compressive sampling. Taking the training samples matrix as the transform matrix will be helpful to the classification. We will find that the procedure of CS based classifier are very different from the classical methods, because steps 3 to 6 are all based on compressive sampling. Currently many non-linear dimensionality reduction methods have been proposed, such as Local Coordinates Alignment(LCA) and Non-Negative Matrix Factorization(NMF). Compressive sampling theory provides a random measurement of signals, and proves to be able to keep the information of the signals under the condition of enough number of measurement and incoherence between the measurement matrix and the transform matrix.

Consequently, it is a natural compressive process of signals, which can also be regarded as the process of dimension reduction. CS is different from LCA and NMF due to that fact that it is a near method, which lend itself to efficient implementation.

Table 2. CS-based Classification

Algorithm: CS-based Classification

Step 1:
Perform a feature extraction on the music samples for k classes.
Step 2:
Perform a feature extraction (described in section 2) on the training songs to obtain a matrix of training samples $\mathbf{A} = [A_1, ...A_k]$ and calculate the feature y of the test sample.
Step 3:
Perform a random measurement (the measurement matrix is a Gaussian random matrix) on the features of the training samples and the test sample feature to obtain $\mathbf{A}' = H \cdot \mathbf{A}$ and $y' = H \cdot y$ respectively.
Step 4:
Normalize the columns of $\mathbf{A}'$ to have unit ℓ_2 norm and solve the ℓ_1-minimization problem:
$\min \|x\|_1 \ \ s.t.\ y' = \mathbf{A}' \cdot x$
Step 5:
Compute the residuals
$\min r_i(y') = \left\|y' - \mathbf{A}'\delta_i(\hat{x})\right\|_2$
Step 6:
Output: $identity(y) = \arg\min r_i(y')$

Table 3. Classification accuracies achieved by various methods on GTZAN dataset.

Method	**Dataset**	**Accuracy**	**Feature dimensions**
MF + CSC (Ours)	GTZAN	92.7	64
TPNTF + SRC	GTZAN	93.7	135
NTF + SRC	GTZAN	92.0	135
MPCA + SRC	GTZAN	89.7	216
GTDA + SRC	GTZAN	92.1	216

4. EXPERIMENTAL RESULTS

The experiments are divided into three parts. Section 4.1 details our experiment with music genre classification. Section 4.2 explores multiple features and dimension reduction. Section 4.3 investigates the feature extractor in an noisy environment.

4.1 Music Genre Classification

Our experiments of music genre classification are performed on GTZAN dataset, which are widely used in the literature [16]. GTZAN consists of the following ten genre classes: Classical, Country, Disco, Hip-Hop, Jazz, Rock, Blues, Reggae, Pop, and Metal. Each genre class contains 100 audio recordings of 30 seconds, with sampling rate of 44.1kHz and resolution of 16 bits.

To evaluate the proposed method for genre classification, we set up all the experimental parameters to be as close as possible to those used in [18]. In particular, the recognition rate is obtained from 10-fold cross validation. Table 3 is a comparison table which lists several other ex-

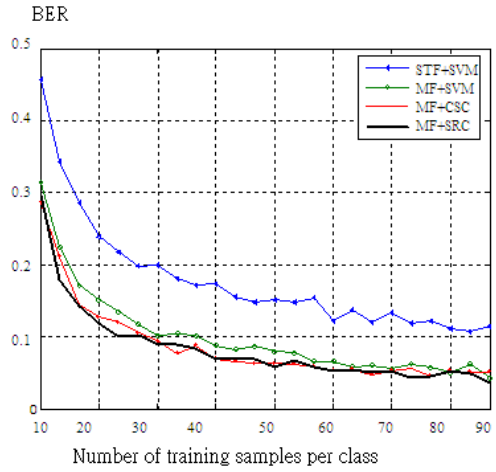


Figure 2. Genre classification result.(CSC is ours)

isting methods together with their recognition rates, such as Topology Preserving Non-Negative Matrix Factorization (TPNMF), Non-Negative Tensor Factorization (NTF), Multilinear Principal Component Analysis (MPCA), and General Tensor Discriminant Analysis (GTDA) [17]. As can be seen from the table, the proposed method (MF + CSC) outperforms all the state-of-the-art SRC-based approaches except one. Moreover, the feature dimension of the proposed approach is considerably lower than those of the SRC-based approaches, demonstrating the effective of CS in extracting features with discriminating power.

This experiment addresses the problem of genre classification using Compressive Sampling. A CS recovery is applied on short-term and long-term features that are relevant for genre classification. The measurement vectors are trained on labeled sets, then the classification is performed by computing the approximation of unknown samples with each class-specific features.

Figure 2 plots the recognition rates of the four methods with respect to no. of training samples per class. (The no. of training samples were randomly selected from each class, while the test samples stayed the same.) The figure demonstrates that multiple features indeed improve the classification accuracy. Moreover, CSC and SRC do have consistent higher accuracy than SVM classifier. More importantly, these two methods do not require the long training process of SVM. In Figure 3 , the computation time of MF+SRC and MF+CSC is only 30% and 20%, respectively, of SVM, due to dimension reduction in compressive sampling.

Table 4 shows the confusion matrix of the CS-based classifier [1]. The columns stands for the actual genre and the rows for the predicted genre. It can be seen that the recognition rate of each class is almost evenly distributed.

4.2 Multiple Features Dimension

In this experiment, we combine feature sets (long-time features and short-time features, and short-time features only) and different classifiers (SVM [14], SRC and the proposed classifier) to investigated their joint effects. The descrip-

Table 4. Confusion matrix of the proposed method

	cl	co	di	hi	ja	ro	bl	re	po	me
cl	**96**	0	0	3	1	0	0	0	0	0
co	0	**92**	4	0	2	0	0	1	0	1
di	0	4	**93**	0	0	1	0	1	0	1
hi	3	0	0	**94**	0	1	1	1	0	0
ja	1	2	0	0	**93**	0	3	1	0	0
ro	0	0	1	1	0	**89**	2	3	3	1
bl	0	0	0	1	3	2	**90**	1	3	0
re	0	1	1	1	1	3	1	**92**	0	0
po	0	0	0	0	0	3	3	0	**94**	0
me	0	1	1	0	0	1	0	0	0	**97**

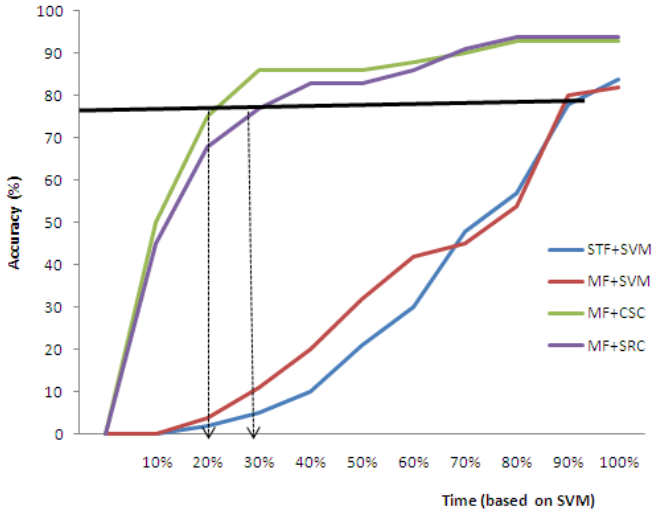


Figure 3. Genre classification time analysis.(CSC is ours)

tions of these methods and their parameter settings are shown in Table 5.

All the samples are digitized 44.1 kHz, 16-bit, and mono signal in preprocessing. The 30-seconds of audio clips after initial 12 seconds are used. The length of the analysis window was set to 93ms, and 50% overlap was used for feature extraction. The length of texture window was set to 3 second, thus a texture window contains 63 analysis windows. The 13-dimensional MFCCs are computed in an analysis window, mean and variance of each dimension are computed in a texture window.

Table 6 shows the multiple features set and dimension. As mentioned in section 2, eight octave subbands were used to compute the OSC, OMSC, MSFM, and MSCM. They are computed based on octave subband. Thus, the dimensions of the features are dependent on the number of octave subband (eight subbands were used in this experiment). The dimensions of the OSC, the OMSC, the MSFM, and the MSCM are respectively 32, 32, 8 and 8.

4.3 Under Noise Environment

In Figure 2, sparse representation based classifier and CS-based classifier have similar performance in music genre classification. The robustness of the system is tested under

Table 5. Methods used in the experiment

Method	Description	Parameters
STF+SVM	Short-time feature only and followed by a SVM classifier	SVM is used and α takes between 0 and 1. The optimal value is chosen experientially.
MF+SVM	Multiple feature and followed by a SVM classifier	As above
MF+CSC	Multiple feature and followed by a compressive sampling based classifier	The sampling rate takes 67% and the optimization algorithm is basis pursuit algorithm.
MF+SRC	Multiple feature and followed by a sparse representation based classifier	The optimization algorithm is basis pursuit algorithm.

Table 6. Multiple features set and dimension

Feature	Set	Dimension
OMSC	Long-time feature	32
Low-energy	Long-time feature	1
OSC	Short-time feature	32
MFCCs	Short-time feature	26
MSFM	Short-time feature	8
Spectral centroid	Short-time feature	2
Spectral rolloff	Short-time feature	2
Spectral flux	Short-time feature	2
Zero crossings	Short-time feature	2

the following conditions.

- Additive white uniform noise (AWUN)
- Additive white Gaussian noise (AWGN)
- Linear speed change (LSC)
- Band-pass filter (BPF)

The robustness of these two methods was compared, as shown in Table 7. We can find the average BER of the CSC system is lower than SRC. CSC has better performance under the conditions of linear speed change, band-pass filter, and additive white uniform noise.

Figure 4 shows the classification results of different methods when the Gaussian noise with different variance are added to the music. From the figure, we can see that the proposed method is quite immune to noise.

5. CONCLUSIONS

In this study, we have proposed a CS-based classifier and verified its performance by a common dataset for music genre classification. Moreover, we have also explored the possibility of using multiple feature sets for improving the performance of genre classification. The experiments demonstrates that the proposed CS-based classification together with the use of multiple feature sets outperform quite a few state-of-the-art approaches for music genre classification. The success of the proposed CS-based classifier is attributed to CS's superb capability in feature extraction for generating parsimonious representation of the original signals.

For immediate future work, we will focus on the possibility of porting the proposed CS-based classifier for other MIR tasks, such as onset detection, beat tracking, and tempo estimation.

Table 7. The comparison result about robustness

	CSC		SRC	
	Rate(%)	BER	Rate (%)	BER
AWUN	73.8	0.262	73.5	0.265
AWGN	76.6	0.234	78.8	0.212
LSC	81.7	0.183	64.8	0.352
BPF	71.2	0.288	65.8	0.342

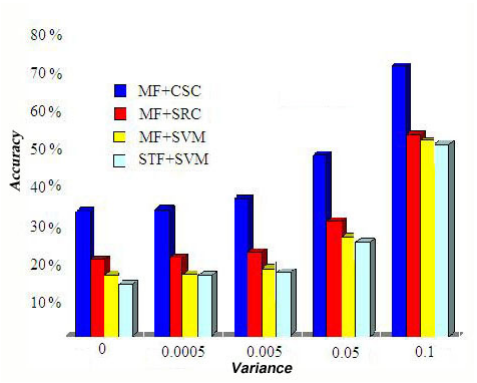


Figure 4. Genre classification result under noise.

6. REFERENCES

[1] G. Tzanetakis and P. Cook: "Musical genre classification of audio signals,"*IEEE Trans. Speech Audio Process.*, vol. 10, no. 5, pp. 293-302, 2002.

[2] D. N. Jiang, L. Lu, H. J. Zhang, J. II. Tao, and L. II. Cai: "Music type classification by spectral contrast feature,"*Proc. ICME 02*, vol. I, pp. 113-116, 2002.

[3] T. Li , M. Ogihara, and Q. Li: "A comparative study on content based music genre classification,"*Proc. ACM Con! on Research and Development in Information Retrieval*, pp. 282-289, 2003.

[4] D. Jang and C. Yoo: "Music information retrieval using novel features and a weighted voting method,"*IEEE International Symposium on Industrial Electronics (ISlE 2009) Seoul Olympic Parktel*, Seoul, Korea July 5-8, 2009.

[5] D. Donoho: "Compressed sensing,"*IEEE Transactions on Information. Theory*, 52(4), pp. 1289-1306, Apr. 2006.

[6] J. Romberg: "Imaging via compressive sampling,"*IEEE Signal Processing Magazine*, 25(2), pp. 14 - 20, March 2008.

[7] Z. Chen, J. Wen, Y. Han, J. Villasenor, S. Yang: "A compressive sensing image compression algorithm using quantized DCT and noiselet information,"*ICASSP*, 2010.

[8] W. Bajwa, J. Haupt, A. Sayeed and R. Nowak: "Joint source-channel communication for distributed estimation in sensor networks,"*IEEE Transactions on Signal Processing*, 53(10), pp. 3629-3653, October 2007.

[9] J. Wright, A. Yang, A. Ganesh, S. Shastry, and Y. Ma: "Robust face recognition via sparse representation,"*IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2), pp. 210-227, February 2009.

[10] E. Candes, M. Wakin, and S. Boyd: "Enhancing sparsity by reweighed ℓ_1 Minimization,"*Journal of Fourier Analysis and Applications*, 14(5), pp. 877-905, October 2008.

[11] L. Lu, D. Liu, and H-J. Zhang: "Automatic mood detection and tracking of music audio signals,"*IEEE Trans. on Audio, Speech, and Language Processing*, vol. 14, no. 1, Jan. 2006.

[12] X. Huang, A. Acero and H.-W. Hon: "Spoken Language Processing,"*Prentice Hall PTR*, 2001.

[13] S. Sukittanon, L. E. Atlas, and 1. W. Pitton: "Modulation-scale analysis for content identification,"*IEEE Trans. on Signal Processing*, vol. 52, no. 10, pp. 3023-3035, Oct., 2004.

[14] C.-C. Chang and C.-J. Lin: "LIBSVM - A Library for Support Vector Machines,"*[Online] Available: www.csie.ntu.edu.tw/.*

[15] I. Karydis , A. Nanopoulos , A. Papadopoulos , Y. Manolopoulos: "Audio Indexing for Efficient Music Information Retrieval,"*Proceedings of the 11th International Multimedia Modeling Conference*, p.22-29, January 12-14, 2005.

[16] D. Jang; M. Jin; C. Yoo: "Music genre classification using novel features and a weighted voting method,"*In Proceeding of the 2008 IEEE International Conference on Multimedia and Expo*, pp. 1377–1380, April 2008.

[17] Y. Panagakis, C. Kotropoulos: "Music genre classification via topology preserving non-negative tensor factorization and sparse representations,"*ICASSP*, 2010.

[18] T. Sainath, A. Carmi, D. Kanevsky, B. Ramabhadran: "Bayesian compressive sensing for phonetic classification,"*ICASSP*, 2010.

[19] Y. Yoon, M. Amin: "Through-the-wall radar imaging using compressive sensing along temporal frequency domain,"*ICASSP*, 2010.

[20] N. Jacobs, S. Schuh, R. Pless: "Compressive sensing and differential image motion estimation,"*ICASSP*, 2010.

SPARSE MULTI-LABEL LINEAR EMBEDDING WITHIN NONNEGATIVE TENSOR FACTORIZATION APPLIED TO MUSIC TAGGING

Yannis Panagakis* **Constantine Kotropoulos***
*Dept. of Informatics
Aristotle University of Thessaloniki
Box 451 Thessaloniki, GR-54124, Greece
{panagakis,costas}@aiia.csd.auth.gr

Gonzalo R. Arce†
†Dept. of Electrical & Computer Engineering
University of Delaware
Newark, DE 19716-3130, U.S.A.
arce@ece.udel.edu

ABSTRACT

A novel framework for music tagging is proposed. First, each music recording is represented by bio-inspired auditory temporal modulations. Then, a multilinear subspace learning algorithm based on sparse label coding is developed to effectively harness the multi-label information for dimensionality reduction. The proposed algorithm is referred to as *Sparse Multi-label Linear Embedding Nonnegative Tensor Factorization*, whose convergence to a stationary point is guaranteed. Finally, a recently proposed method is employed to propagate the multiple labels of training auditory temporal modulations to auditory temporal modulations extracted from a test music recording by means of the sparse ℓ_1 reconstruction coefficients. The overall framework, that is described here, outperforms both humans and state-of-the-art computer audition systems in the music tagging task, when applied to the CAL500 dataset.

1. INTRODUCTION

The emergence of Web 2.0 and the success of music oriented social network websites, such as *last.fm*, has revealed the concept of music tagging. Tags are text-based labels that encode semantic information related to music (i.e., instrumentation, genres, emotions, etc.). They result into a semantic representation of music, which can be used as input to collaborative filtering systems assisting users to search for music content. However, a drawback of such approach is that a newly added music recording must be tagged manually first, before it can be retrieved [18, 19], which is a time consuming and expensive process. Therefore, an emerging problem in Music Information Retrieval (MIR) aims to automate the process of music tagging. This problem is referred to as *automatic music tagging* or *automatic multi-label music annotation*.

MIR has mainly focused on content-based classification of music by genre [11–13] and emotion [14]. These classification systems effectively annotate music with class labels, such as "rock", "happy", etc., by assuming a predefined taxonomy and an explicit mapping of a music recording onto mutually exclusive classes. However, such assumptions are unrealistic and result into a number of problems, since music perception is inherently subjective [19]. The latter problems can be overcome by the less restrictive approach of annotating the audio content with more than one labels in order to reflect more aspects of music. Relatively little work has been made on multi-label automatic music annotation compared to the work made on multi-label automatic image annotation (cf. [3, 20] and the references therein). However, various automatic music tagging algorithms have been proposed [2, 6, 8, 17, 19]. For instance, audio tag prediction is treated as a set of binary classification problems where standard classifiers, such as the Support Vector Machines [17] or Ada-Boost [2] can be applied. Furthermore, methods that resort to probabilistic modeling have been proposed [6, 19]. These methods attempt to infer the correlations or joint probabilities between the tags and the low-level acoustic features extracted from audio.

In this paper, the problem of automatic music tagging is addressed as a multi-label multi-class classification problem by employing a novel multilinear subspace learning algorithm and sparse representations. Motivated by the robustness of the auditory representations in music genre classification [11–13], each audio recording is represented in terms of its slow temporal modulations by a two dimensional (2D) auditory representation as in [13]. Consequently, an ensemble of audio recordings is represented by a third-order tensor. The auditory temporal modulations do not explicitly utilize the label set (i.e., the tags) of music recordings. Due to the semantic gap, it is unclear how to exploit the semantic similarity between the label sets associated to two music recordings for efficient feature extraction within multi-label music tagging. Motivated by the automatic multi-label image annotation framework proposed in [20], the semantic similarities between two music recordings with overlapped labels are measured in a sparse representation based way rather than in one-to-one way as in [2, 6, 17, 19]. There is substantial evidence in the literature that the multilinear subspace learning algorithms are more appropriate for reducing the dimensionality of tensor objects [13, 16]. To this end, a novel multilinear subspace learning algorithm is developed here to efficiently harness

the multi-label information for feature extraction. In particular, the proposed method incorporates the Multi-label Linear Embedding (MLE) [20] into the Nonnegative Tensor Factorization (NTF) [11] by formulating an optimization problem, which is then solved by the Projected Gradient method [1, 9]. The proposed method is referred to as *Sparse Multi-label Linear Embedding Nonnegative Tensor Factorization* (SMLENTF). The SMLENTF reduces the high-dimensional feature space, where the high-order data (i.e. the auditory temporal modulations) lie, into a lower-dimensional semantic space dominated by the label information. Features extracted by the SMLENTF form an overcomplete dictionary for the semantic space of music. If sufficient training music recordings are available, it is possible to express any test representation of auditory temporal modulations as a compact linear combination of the dictionary atoms, which are semantically close. This representation is designed to be sparse, because it involves only a small fraction of the dictionary atoms and can be computed efficiently via ℓ_1 optimization. Finally, tags are propagated from the training atoms to a test music recording with the coefficients of sparse ℓ_1 representation.

The performance of the proposed automatic music tagging framework is assessed by conducting experiments on the CAL500 dataset [18,19]. For comparison purposes, the MLE [20] is also tested in this task. The reported experimental results demonstrate the superiority of the proposed SMLENTF over the MLE, the human performance as well as that of state-of-the-art computer audition systems in music tagging on the CAL500 dataset.

The paper is organized as follows. In Section 2, basic multilinear algebra concepts and notations are defined. In Section 3, the bio-inspired auditory representation derived by a computational auditory model is briefly described. The SMLENTF is introduced in Section 4. The multi-label annotation framework, that is based on the sparse representations, is detailed in Section 5. Experimental results are demonstrated in Section 6 and conclusions are drawn in Section 7.

2. NOTATION AND MULTILINEAR ALGEBRA BASICS

Tensors are considered as the multidimensional equivalent of matrices (i.e., second-order tensors) and vectors (i.e., first-order tensors) [7]. Throughout the paper, tensors are denoted by boldface Euler script calligraphic letters (e.g. $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{A}}$), matrices are denoted by uppercase boldface letters (e.g. $\mathbf{U}$), vectors are denoted by lowercase boldface letters (e.g. $\mathbf{u}$), and scalars are denoted by lowercase letters (e.g. u). The ith row of $\mathbf{U}$ is denoted as $\mathbf{u}_{i:}$ while its jth column is denoted as $\mathbf{u}_{:j}$.

Let $\mathbb{Z}$ and $\mathbb{R}$ denote the set of integer and real numbers, respectively. A high-order real valued tensor $\boldsymbol{\mathcal{X}}$ of order N is defined over the tensor space $\mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, where $I_n \in \mathbb{Z}$ and $n = 1, 2, \ldots, N$. Each element of $\boldsymbol{\mathcal{X}}$ is addressed by N indices, i.e., $x_{i_1 i_2 i_3 \ldots i_N}$. Mode-$n$ unfolding of tensor $\boldsymbol{\mathcal{X}}$ yields the matrix $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \ldots I_{n-1} I_{n+1} \ldots I_N)}$. In the following, the operations on tensors are expressed in matricized form [7].

An N-order tensor $\boldsymbol{\mathcal{X}}$ has rank-1, when it is decomposed as the outer product of N vectors $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots, \mathbf{u}^{(N)}$, i.e. $\boldsymbol{\mathcal{X}} = \mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ \ldots \circ \mathbf{u}^{(N)}$. That is, each element of the tensor is the product of the corresponding vector elements, $x_{i_1 i_2 \ldots i_N} = u^{(1)}_{i_1} u^{(2)}_{i_2} \ldots u^{(N)}_{i_N}$ for $i_n = 1, 2, \ldots, I_n$. The rank of an arbitrary N-order tensor $\boldsymbol{\mathcal{X}}$ is the minimal number of rank-1 tensors that yield $\boldsymbol{\mathcal{X}}$ when linearly combined. Next, several products between matrices will be used, such as the Khatri-Rao product denoted by $\odot$ and the Hadamard product (i.e. element-wise product) denoted by $*$, whose definitions can be found in [7] for example.

3. AUDITORY REPRESENTATION OF TEMPORAL MODULATIONS

A key step for representing music signals in a psychophysiologically consistent manner is to resort on how the audio is encoded in the human *primary auditory cortex*. The primary auditory cortex is the first stage of the central auditory system, where higher level mental processes take place, such as perception and cognition [10]. To this end the *auditory representation of temporal modulations* is employed [13]. The auditory representation is a joint acoustic and modulation frequency representation that discards much of the spectro-temporal details and focuses on the underlying slow temporal modulations of the music signal [15]. Such a representation has been proven very robust in representing music signals for music genre classification [12, 13].

The 2D representation of auditory temporal modulations can be obtained by modeling the path of auditory processing as detailed in [13]. The computational model of human auditory system consists of two basic processing stages. The first stage models the early auditory system. It converts the acoustic signal into an auditory representation, the so-called *auditory spectrogram*, i.e. a time-frequency distribution along a tonotopic (logarithmic frequency) axis. At the second stage, the temporal modulation content of the auditory spectrogram is estimated by wavelets applied to each channel of the auditory spectrogram. Psychophysiological evidence justifies the discrete rate $r \in \{2, 4, 8, 16, 32, 64, 128, 256\}$ (Hz) in order to represent the temporal modulation content of sound [13]. The cochlear model, employed in the first stage, has 96 filters covering 4 octaves along the tonotopic axis (i.e. 24 filters per octave). Accordingly, the auditory temporal modulations of a music recording are represented by a real-valued nonnegative second-order tensor (i.e. a matrix) $\mathbf{X} \in \mathbb{R}_+^{I_1 \times I_2}$, where $I_1 = I_f = 96$ and $I_2 = I_r = 8$. Hereafter, let $\mathbf{x} = \mathrm{vec}(\mathbf{X}) \in \mathbb{R}_+^{I_1 \cdot I_2} = \mathbb{R}_+^{768}$ denote the lexicographically ordered vectorial representation of the auditory temporal modulations.

4. SPARSE MULTI-LABEL LINEAR EMBEDDING NONNEGATIVE TENSOR FACTORIZATION

Multilinear subspace learning algorithms are required in order to map the high-dimensional original tensor space

onto a lower-dimensional semantic space defined by the labels. In conventional supervised multilinear subspace learning algorithms, such as the General Tensor Discrimininant Analysis [16], it is assumed that data points annotated by the same label should be close to each other in the feature space, while data bearing different labels should be far away. However, this assumption is not valid in a multi-label task, as discussed in [20]. Accordingly, such subspace learning algorithms will fail to derive a lower-dimensional semantic space based on multiple labels.

Let $\{\mathcal{X}_i|_{i=1}^I\}$ be a set of I training nonnegative tensors $\mathcal{X}_i \in \mathbb{R}_+^{I_1 \times I_2 \times \ldots \times I_N}$ of order N. We can represent such a set by a $(N+1)$-order tensor $\mathcal{Y} \in \mathbb{R}_+^{I_1 \times I_2 \times \ldots \times I_N \times I_{N+1}}$ with $I_{N+1} = I$. Furthermore, let us assume that the multi-labels of the training tensor $\mathcal{Y}$ are represented by the matrix $\mathbf{C} \in \mathbb{R}_+^{V \times I}$, where V indicates the cardinality of the tag vocabulary. Obviously, $c_{ki} = 1$ if the ith tensor is labeled with the kth tag in the vocabulary and 0 otherwise. Since, every tensor object (music recording here) can be labeled by multiple labels, there may exist more than one non-zero elements in a label vector (i.e. $\mathbf{c}_{:i}$).

To overcome the limitation of conventional multilinear subspace learning algorithms, the MLE [20] is incorporated into the NTF. To this end, two methods exploit the multi-label information in order to drive semantically oriented feature extraction from tensor objects. First, the tensor objects with the same label set, that is $\mathbf{c}_{:i} = \mathbf{c}_{:j}$, are considered to be fully semantically related and thus the similarity graph $\mathbf{W}^1$ has elements $w_{ij}^1 = w_{ji}^1 = 1$ and 0 otherwise. However, in real-world datasets, data samples with exactly the same label set are rare. In such a case, the semantic relationship between the data samples can be inferred via the ℓ_1 semantic graph as proposed in [20]. Let us denote by $\mathbf{W}^2$ the ℓ_1 semantic graph. $\mathbf{W}^2$ contains the coefficients that represent each label vector $\mathbf{c}_{:i}$ as a compact linear combination of the remaining semantically related label vectors. Formally, let us define $\hat{\mathbf{C}}_i = [\mathbf{c}_{:1}|\mathbf{c}_{:2}|\ldots|\mathbf{c}_{:i-1}|\mathbf{c}_{:i+1}|\ldots|\mathbf{c}_{:I}]$. If $V \ll I$ the linear combination coefficients $\mathbf{a}$ can be obtained by seeking the sparsest solution to the undetermined system of equations $\mathbf{c}_{:i} = \hat{\mathbf{C}}_i\mathbf{a}$. That is, solving the following optimization problem:

$$\arg\min_{\mathbf{a}} \|\mathbf{a}\|_0 \quad \text{subject to } \hat{\mathbf{C}}_i\mathbf{a} = \mathbf{c}_{:i}, \tag{1}$$

where $\|.\|_0$ is the ℓ_0 quasi-norm returning the number of the non-zero entries of a vector. Finding the solution to the optimization problem (1) is NP-hard due to the nature of the underlying combinational optimization. In [5], it has been proved that if the solution is sparse enough, then the solution of (1) is equivalent to the solution of the following optimization problem:

$$\arg\min_{\mathbf{a}} \|\mathbf{a}\|_1 \quad \text{subject to } \hat{\mathbf{C}}_i\,\mathbf{a} = \mathbf{c}_{:i}, \tag{2}$$

where $\|.\|_1$ denotes the ℓ_1 norm of a vector. (2) can be solved in polynomial time by standard linear programming methods [4].

The ℓ_1 semantic graph $\mathbf{W}^2$ is derived as follows. For each label vector, $\hat{\mathbf{C}}_i$ is constructed and then it is normalized so as its column vectors have unit norm. Then, (2) is solved by replacing $\hat{\mathbf{C}}_i$ with its normalized variant and the sparse representation vector $\mathbf{a}$ is obtained. Next, $w_{ij}^2 = a_j$ for $1 \leq j \leq i-1$; $w_{ij}^2 = a_{j-1}$ for $i+1 \leq j \leq I$. Clearly, the diagonal elements of $\mathbf{W}^2$ are equal to zero.

Let $d_{ii}^1 = \sum_{i \neq j} w_{ij}^1$ be the diagonal elements of the diagonal matrix $\mathbf{D}^1$. Given $\{\mathcal{X}_i|_{i=1}^I\}$, one can model the semantic relationships between the tensor objects by constructing the multi-label linear embedding matrix, which exploits $\mathbf{W}^1$ and $\mathbf{W}^2$ as in [20]: $\mathbf{M} = \mathbf{D}^1 - \mathbf{W}^1 + \frac{\beta}{2}(\mathbf{I} - \mathbf{W}^2)^T(\mathbf{I} - \mathbf{W}^2)$, where $\beta > 0$ is a parameter, which adjusts the contribution of the ℓ_1 graph in the multi-label linear embedding [20]. Let $\{\mathbf{U}^{(n)}|_{n=1}^{N+1}\}$ be the mode-n factor matrices derived by the NTF applied to $\mathcal{Y}$ [11]. We define $\mathbf{Z}^{(n)} \triangleq \mathbf{U}^{(N+1)} \odot \ldots \odot \mathbf{U}^{(n+1)} \odot \mathbf{U}^{(n-1)} \odot \ldots \odot \mathbf{U}^{(1)}$. One can incorporate the semantic information of tensor objects into the NTF by minimizing the following objective function for the SMLENTF in matricized form:

$$f\big(\mathbf{U}^{(n)}|_{n=1}^{N+1}\big) = \frac{1}{2}\|\mathbf{Y}_{(n)} - \mathbf{U}^{(n)}\big[\mathbf{Z}^{(n)}\big]^T\|_F^2 + \lambda\,\mathrm{tr}\Big\{\big[\mathbf{U}^{(N+1)}\big]^T\,\mathbf{M}\,\mathbf{U}^{(N+1)}\Big\}, \tag{3}$$

where $\lambda > 0$ is a parameter, which controls the trade off between the goodness of fit to the training data tensor $\mathcal{Y}$ and the multi-label linear embedding and $\|.\|_F$ denotes the Frobenious norm. Consequently, we propose to minimize (3) subject to the nonnegative factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}_+^{I_n \times k}$, $n = 1, 2, \ldots, N+1$, where k is the desirable number of rank-1 tensors approximating $\mathcal{Y}$ when linearly combined.

Let $\nabla_{\mathbf{U}^{(n)}} f = \frac{\partial f}{\partial \mathbf{U}^{(n)}}$ be the partial derivative of the objective function $f(\mathbf{U}^{(n)}|_{n=1}^{N+1})$ with respect to $\mathbf{U}^{(n)}$. It can be shown that for $n = 1, 2, \ldots, N$ we have

$$\nabla_{\mathbf{U}^{(n)}} f = \mathbf{U}^{(n)}\big[\mathbf{Z}^{(n)}\big]^T\mathbf{Z}^{(n)} - \mathbf{Y}_{(n)}\mathbf{Z}^{(n)}, \tag{4}$$

while for $n = N+1$ we obtain

$$\nabla_{\mathbf{U}^{(N+1)}} f = \mathbf{U}^{(N+1)}\big[\mathbf{Z}^{(N+1)}\big]^T\mathbf{Z}^{(N+1)} + \lambda\, M\mathbf{U}^{(N+1)} - \mathbf{Y}_{(N+1)}\mathbf{Z}^{(N+1)}. \tag{5}$$

Following the strategy employed in the derivation of the Projected Gradient Nonnegative Matrix Factorization [9], we obtain an iterative alternating algorithm for the SMLENTF as follows. Given $N+1$ randomly initialized nonnegative matrices $\mathbf{U}^{(n)}|_{n=1}^{N+1} \in \mathbb{R}_+^{I_n \times k}$, a stationary point of (3) can be found by the update rule:

$$\mathbf{U}_{[t+1]}^{(n)} = [\,\mathbf{U}_{[t]}^{(n)} - n_{[t]}\nabla_{\mathbf{U}_{[t]}^{(n)}} f\,]^+, \tag{6}$$

where t denotes the iteration index and $[.]^+$ is the projection operator, which is defined element-wise as $[.]^+ \triangleq \max(., 0)$. The projection operator ensures that $\mathbf{U}_{[t+1]}^{(n)}$ contains only nonnegative elements after each iteration. The learning rate $n_{[t]}$ can be determined by the Armijo rule along the projection arc [1] or more effectively by the Algorithm 4 in [9] in order to ensure the convergence of the algorithm to a stationary point. The update rule (6) is executed iteratively in an alternating fashion for $n = 1, 2,$

$\ldots, N+1$ until the global convergence criterion is met:

$$\sum_{n=1}^{N+1} \|\nabla^P_{\mathbf{U}^{(n)}_{[t]}} f\|_F \leq \epsilon \sum_{n=1}^{N+1} \|\nabla_{\mathbf{U}^{(n)}_{[t]}} f\|_F, \qquad (7)$$

where $[\nabla^P_{\mathbf{U}^{(n)}_{[t]}} f]_{ij} = \min\left(0, [\nabla_{\mathbf{U}^{(n)}_{[t]}} f]_{ij}\right)$ if $[\mathbf{U}^{(n)}_{[t]}]_{ij} = 0$; and $[\nabla^P_{\mathbf{U}^{(n)}_{[t]}} f]_{ij} = [\nabla_{\mathbf{U}^{(n)}_{[t]}} f]_{ij}$ if $[\mathbf{U}^{(n)}_{[t]}]_{ij} \geq 0$. The parameter ϵ is a predefined small positive number, typically 10^{-5} [9]. The convergence criterion (7) is employed in order to check the stationarity of the solution set $\{\mathbf{U}^{(n)}_{[t]}|^{N+1}_{n=1}\}$ since it is equivalent to the Karush-Kuhn-Tucker optimality condition [1, 9].

5. MULTI-LABEL ANNOTATION VIA SPARSE REPRESENTATIONS

In this section, the task of automatic music tagging is addressed by sparse representations of auditory temporal modulations projected onto a reduced dimension feature space, where the semantic relations between them are retained.

For each music recording a 2D auditory representation of temporal modulations is extracted as is briefly described in Section 3 and detailed in [13]. Thus, each ensemble of recordings is represented by a third-order data tensor, which is created by stacking the second-order feature tensors associated to the recordings. Consequently, the data tensor $\mathcal{Y} \in \mathbb{R}_+^{I_1 \times I_2 \times I_3}$, where $I_1 = I_f = 96$, $I_2 = I_r = 8$, and $I_3 = I_{samples}$ is obtained. Let $\mathcal{Y}_{train} \in \mathbb{R}_+^{I_1 \times I_2 \times I}$, $I < I_{samples}$, be the tensor where the training auditory temporal modulations representations are stored. By applying the SMLENTF onto the $\mathcal{Y}_{train}$ three factor matrices are derived, namely $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, $\mathbf{U}^{(3)}$, associated to the frequency, rate, and samples modes of the training tensor $\mathcal{Y}_{train}$, respectively. Next, the projection matrix $\mathbf{P} = \mathbf{U}^{(2)} \odot \mathbf{U}^{(1)} \in \mathbb{R}_+^{768 \times k}$, with $k \ll \min(768, I)$, is obtained. The columns of $\mathbf{P}$ span a reduced dimension feature space, where the semantic relations between the vectorized auditory temporal modulations are retained. Consequently, by projecting all the training auditory temporal modulations onto this reduced dimension space an overcomplete dictionary $\mathbf{D} = \mathbf{P}^T \mathbf{Y}^T_{train(3)} \in \mathbb{R}_+^{k \times I}$ is obtained. Alternatively, the dictionary can be obtained by $\mathbf{D} = \mathbf{P}^\dagger \mathbf{Y}^T_{train(3)}$, where $(.)^\dagger$ denotes the Moore-Penrose pseudoinverse.

Given a vectorized representation of auditory temporal modulations $\mathbf{x} \in \mathbb{R}_+^{768}$ associated to a test music recording, first is projected onto the reduced dimension space and a new feature vector is obtained i.e. $\bar{\mathbf{x}} = \mathbf{P}^T \mathbf{x} \in \mathbb{R}_+^k$ or $\bar{\mathbf{x}} = \mathbf{P}^\dagger \mathbf{x} \in \mathbb{R}^k$. Now, $\bar{\mathbf{x}}$ can be represented as a compact linear combination of the semantically related atoms of $\mathbf{D}$. That is, the test auditory representation of temporal modulations is considered semantically related to the few training auditory representations of temporal modulations with non-zero approximation coefficients. This implies that the corresponding music recordings are semantically related, as well. Again, since $\mathbf{D}$ is overcomplete, the sparse coefficient vector $\mathbf{b}$ can be obtained by solving the following optimization problem:

$$\arg\min_{\mathbf{b}} \|\mathbf{b}\|_1 \quad \text{subject to } \mathbf{D}\,\mathbf{b} = \bar{\mathbf{x}}. \qquad (8)$$

By applying the SMLENTF, the semantic relations between the label vectors are propagated to the feature space. In music tagging, the semantic relations are expected to propagate from the feature space to the label vector space. Let us denote by $\bar{\mathbf{a}}$ the label vector of the test music recording. Then, $\bar{\mathbf{a}}$ is obtained by

$$\bar{\mathbf{a}} = \mathbf{C}\,\mathbf{b}. \qquad (9)$$

The labels with the largest values in $\bar{\mathbf{a}}$ yield the final tag vector of the test music recording.

6. EXPERIMENTAL EVALUATION

In order to assess the performance of the proposed framework in automatic music tagging, experiments were conducted on the CAL500 dataset [18, 19]. The CAL500 is a corpus of 500 tracks of Western popular music, each of which has been manually annotated by three human annotators at least, who employ a vocabulary of 174 tags. The tags used in CAL500 dataset annotation span six semantic categories, namely instrumentation, vocal characteristics, genres, emotions, acoustic quality of the song, and usage terms (e.g. "I would like to listen this song while *driving*, *sleeping* etc.") [19]. All the recordings were converted to monaural wave format at a sampling frequency of 16 kHz and quantized with 16 bits. Moreover, the music signals have been normalized, so that they have zero mean amplitude with unit variance in order to remove any factors related to the recording conditions.

Following the experimental set-up used in [2, 6, 19], 10-fold cross-validation was employed during the experimental evaluation process. Thus each training set consists of 450 audio files. Accordingly, the training tensor $\mathcal{Y}_{train} \in \mathbb{R}_+^{96 \times 8 \times 450}$ was constructed by stacking the auditory temporal modulations representations. The projection matrix $\mathbf{P}$ was derived from the training tensor $\mathcal{Y}_{train}$ by employing either the SMLENTF or the MLE [20]. The length of the tag vector returned by our system was 10. That is, each test music recording was annotated with 10 tags. Throughout the experiments, the value of λ in SMLENTF was empirically set to 0.5, while the value of β used in forming the matrix $\mathbf{M}$ was set to 0.5 for both the SMLENTF and the MLE.

Three metrics, the mean per-word precision and the mean per-word recall and the F_1 score are employed in order to assess the annotation performance of the proposed automatic music tagging system. Per-word recall is defined as the fraction of songs actually labeled with word w that the system annotates with label w. Per-word precision is defined as the fraction of songs annotated by the system with label w that are actually labeled with word w. As in [6], if no test music recordings are labeled with the word w, then the per-word precision is undefined, accordingly these words are omitted during the evaluation procedure. The F_1

score is the harmonic mean of precision and recall, that is $F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$.

In Table 1, quantitative results on automatic music tagging are presented. In particular, CBA refers to the probabilistic model proposed in [6]. MixHier is Turnbull *et al.* system based on a Gaussian mixture model [19], while Autotag refers to Bertin-Mahieux *et al.* system proposed in [2]. Random refers to a baseline system that annotates songs randomly based on tags' empirical frequencies. Even though the range of precision and recall is $[0, 1]$, the aforementioned metrics may be upper-bounded by a value less than 1 if the number of tags appearing in the ground truth annotation is either greater or less than the number of tags that are returned by the automatic music annotation system. Consequently, UpperBnd indicates the best possible performance under each metric. Random and UpperBnd were computed by Turnbull *et al.* [19], and give a sense of the actual range for each metric. Finally, Human indicates the performance of humans in assigning tags to the recordings of the CAL500 dataset. All the reported performance metrics are means and standard errors (i.e. the sample standard deviation divided by the sample size) inside parentheses computed from 10-fold cross-validation on the CAL500 dataset.

System	Precision	Recall
CBA [6]	0.286 (0.005)	0.162 (0.004)
MixHier [19]	0.265 (0.007)	0.158 (0.006)
Autotag [2]	0.281	0.131
UpperBnd [19]	0.712 (0.007)	0.375 (0.006)
Random [19]	0.144 (0.004)	0.064 (0.002)
Human [19]	0.296 (0.008)	0.145 (0.003)

Table 1. Mean annotation results on the CAL500 Dataset.

In Figure 1, the mean precision, the mean recall, and the F_1 score is plotted as a function of the feature space dimensionality derived by the MLE and the SMLENTF. Clearly, the SMLENTF outperforms the MLE for all the dimensions of the feature space. The best music annotation performance with respect to the mean per-word precision and the mean per-word recall is summarized in Table 2. The numbers inside parentheses are the standards errors estimated thanks to the 10-fold cross-validation.

System	Dimension (k)	Precision	Recall	F_1 Score
MLE [20]	150	0.346 (0.004)	0.154 (0.002)	0.2128
SMLENTF	150	0.371 (0.003)	0.165 (0.002)	0.2291

Table 2. Best mean annotation results obtained by MLE and SMLENTF on the CAL500 Dataset.

By inspecting Table 1, Table 2, and Figure 1 SMLENTF clearly exhibits the best performance with respect to the per-word precision and per-word recall among the state-of-the-art computer audition systems that is compared to, no matter what the feature space dimensionality is. Furthermore, MLE outperforms the CBA, the MixHier, and the Autotag system with respect to the per-word precision, while in terms of the per-word recall its performance is comparable to that achieved by the MixHier. In addition both the SMLENTF and the MLE perform better than humans with respect to the per-word precision and the per-word recall in the task under study. These results make our framework the top performing system in music tagging motivating further research. The success of the proposed system can be attributed to the fact that the semantic similarities between two music signals with overlapped labels that are measured in a sparse representation-based way rather than in an one-to-one way as in [2, 6, 17, 19] by applying the multi-label linear embedding and the sparse representations both in the features extraction and the classification process.

7. CONCLUSIONS

In this paper, an appealing automatic music tagging framework has been proposed. This framework resorts to auditory temporal modulations for music representation, while multi-label linear embedding as well as sparse representations have been employed for multi-label music annotation. A multilinear subspace learning technique, the SMLENTF, has been developed, which incorporates the semantic information of the auditory temporal modulations with respect to the music tags into the NTF. The results reported in the paper outperform humans' performance as well as any other result obtained by the state-of-the-art computer audition systems in music tagging applied to the CAL500 dataset.

In many real commercial applications, the number of available tags is large. Usually most of the tags are associated to a small number of audio recordings. Thus, it is desirable the automatic music tagging systems to perform well in such small sets. Future research will address the performance of the proposed framework under such conditions.

8. REFERENCES

[1] D. P. Bertsekas: *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1999.

[2] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere: "Autotagger: A Model for Predicting Social Tags from Acoustic Features on Large Music Databases," *J. New Music Research*, Vol. 37, No. 2, pp. 115-135, 2008.

[3] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos: "Supervised Learning of Semantic Classes for Image Annotation and Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 29, No. 3, pp. 394–410, 2007.

[4] S. S. Chen, D. L. Donoho, and M. A. Saunders: "Atomic Decomposition by Basis Pursuit," *SIAM J. Sci. Comput.*, Vol. 20, No. 1, pp. 33–61, 1998.

[5] D. L. Donoho, and X. Huo: "Uncertainty Principles and Ideal Atomic Decomposition," *IEEE Trans. Information Theory*, Vol. 47, No. 7. pp. 2845–2862, 2001.

[6] M. Hoffman, D. Blei, and P. Cook: "Easy as CBA: A Simple Probabilistic Model for Tagging Music," *Proceedings of the 10th Int. Symp. Music Information Retrieval*, Kobe, Japan, 2009.

[7] T. Kolda and B. W. Bader: "Tensor Decompositions and Applications," *SIAM Review*, Vol. 51, No. 3, pp. 455–500, 2009.

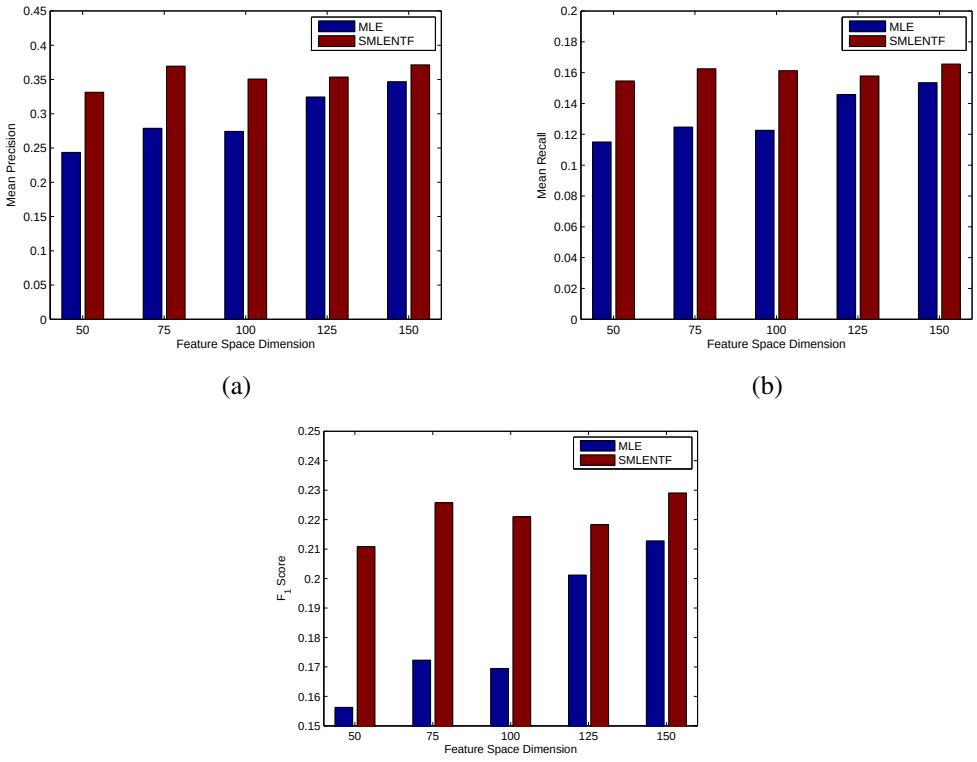


Figure 1. Mean annotation results for the MLE and the SMLENTF with respect to (a) the mean precision, (b) the mean recall, and (c) the F_1 score on the CAL500 dataset.

[8] E. Law, K. West, M. Mandel, M. Bay, and J. S. Downie: "Evaluation of Algorithms Using Games: The Case of Music Tagging," *Proceedings of 10th Int. Symp. Music Information Retrieval*, Kobe, Japan, pp. 387–392, 2009.

[9] C. J. Lin: "Projected Gradient Methods for Nonnegative Matrix Factorization," *Neural Computation*, Vol. 19, No. 10, pp. 2756–2779, 2007.

[10] R. Munkong and J. Biing-Hwang: "Auditory Perception and Cognition," *IEEE Signal Processing Magazine*, Vol. 25, No. 3, pp. 98–117, 2008.

[11] Y. Panagakis, C. Kotropoulos, and G. R. Arce: "Music Genre Classification Using Locality Preserving Non-Negative Tensor Factorization and Sparse Representations," *Proceedings of 10th Int. Symp. Music Information Retrieval*, Kobe, Japan, pp. 249–254, 2009.

[12] Y. Panagakis, C. Kotropoulos, and G. R. Arce: "Music Genre Classification via Sparse Representation of Auditory Temporal Modulations," *Proceedings of EUSIPCO 2009*, Glasgow, Scotland, 2009.

[13] Y. Panagakis, C. Kotropoulos, and G. R. Arce: "Non-Negative Multilinear Principal Component Analysis of Auditory Temporal Modulations for Music Genre Classification," *IEEE Trans. Audio Speech and Language Technology*, Vol. 18, No. 3, pp. 576–588, 2010.

[14] S. Rho, B. Han, and E. Hwang: "SVR-based Music Mood Classification and Context-based Music Recommendation," *Proceedings of 17th ACM Int. Conf. Multimedia*, pp. 713–716, Beijing, China, 2009.

[15] S. Sukittanon, L. E. Atlas, and J. W Pitton: "Modulation-scale Analysis for Content Identification," *IEEE Trans. Signal Processing*, Vol. 52, No. 10, pp. 3023–3035, 2004.

[16] D. Tao, X. Li, X. Wu, and S. J. Maybank: "General Tensor Discriminant Analysis and Gabor Features for Gait Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 29, No. 10, pp. 1700–1715, 2007.

[17] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas: "Multilabel Classification of Music into Emotions," *Proceedings of 9th Int. Symp. Music Information Retrieval*, Philadelphia, USA, pp. 325–330, 2008.

[18] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet: "Towards Musical Query-By-Semantic-Description Using the CAL500 Data Set," *Proceedings of 30th ACM Int. Conf. Research and Development in Information Retrieval*, Amsterdam, The Netherlands, pp. 439-446, 2007.

[19] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet: "Semantic Annotation and Retrieval of Music and Sound Effects," *IEEE Trans. Audio Speech and Language Processing*, Vol. 16, No. 2, pp. 467–476, 2008.

[20] C. Wang, S. Yan, L. Zhang, and H.-J. Zhang: "Multi-label Sparse Coding for Automatic Image Annotation," *Proceedings of IEEE Int. Conf. Computer Vision and Pattern Recognition*, Florida, USA, pp. 1643-1650, 2009.

LEARNING TAGS THAT VARY WITHIN A SONG

Michael I Mandel, Douglas Eck, Yoshua Bengio
LISA Lab, Université de Montréal
{mandelm,eckdoug}@iro.umontreal.ca, yoshua.bengio@umontreal.ca

ABSTRACT

This paper examines the relationship between human generated tags describing different parts of the same song. These tags were collected using Amazon's Mechanical Turk service. We find that the agreement between different people's tags decreases as the distance between the parts of a song that they heard increases. To model these tags and these relationships, we describe a conditional restricted Boltzmann machine. Using this model to fill in tags that should probably be present given a context of other tags, we train automatic tag classifiers (autotaggers) that outperform those trained on the original data.

1. INTRODUCTION

Social tags are short free-form descriptions of music that users apply to songs, albums, and artists. They have proven to be a popular way for users to organize and discover music in large collections [5]. There remain, however, millions of tracks that have never been tagged by a user that cannot be included in these systems. Automatic tagging, based on an analysis of the audio of these tracks and user tagging behavior, could enable them to be included in these systems immediately. To this end, this paper explores the relationship between audio and the tags that humans apply to it, especially at different time scales and at different points within the same track.

We perform this examination in the context of a "Human Intelligence Task" (HIT) on the Mechanical Turk website [1], where users are paid small amounts of money to perform tasks for which human intelligence is required. Mechanical Turk has been used extensively in natural language processing [10] and vision [11, 13], but to our knowledge has not been used in music information retrieval before. Mechanical Turk is one means to the end of *human computation*, the field of cleverly harnessing human intelligence to solve computational problems. This field has been growing in popularity recently, especially in the context of games for collecting descriptions of music [6, 7, 12]. While these games have proven popular among researchers for collecting these data, they require significant investment of development time and effort in order to attract and retain players. By using Mechanical Turk, a researcher can trade a little extra money for significant savings in development time.

This paper makes three contributions. First, in Section 2 we discuss data collection and analysis from a new source, Mechanical Turk, and Section 2.1 shows that clips from different parts of the same song tend to be described differently from one another. Second, Section 3.1 presents a probabilistic model of tags and their relationships with each other to combat the sparsity of music tagging data. Section 3.3 shows that explicitly including information linking tags from the same user, track, and clip improves the likelihood of held out data under the model. Finally, we use this model to "smooth" tag data, i.e. to infer tags that were not provided, but perhaps should have been, given the tags that were. Section 4 shows that these smoothed tags are more "learnable" from the audio signal than the raw tags provided directly by the users, especially when fewer users have seen a given clip.

2. DATA COLLECTION

Users of the Mechanical Turk website, known as "turkers", were asked to listen to a clip from a song and describe its unique characteristics using between 5 and 15 words. The task was free response, but to provide some guidance, we requested tags in 5 categories: Styles/Genres, Vocals/Instruments, Overall sound/feel (global qualities like production and rhythm), Moods/Emotions, Other (sounds alike artists, era, locale, song section, audience, activities, etc.). In order to avoid biasing the turkers' responses, no examples of tags in each category were provided. Turkers were paid between $0.03 and $0.05 per clip, on which they generally spent about one minute.

The music used in the experiment was collected from music blogs that are indexed by the Hype Machine [2]. We downloaded the front page of each of the approximately 2000 blogs and recorded the URLs of any mp3 files linked from them, a total of approximately 17,000 mp3s. We downloaded 1500 of these mp3s at random, of which approximately 700 were available, error free, and at least 128 kbps while still being below 10 megabytes (to avoid DJ sets, podcasts, etc). Of these, we selected 185 at random.

From each of these 185 tracks, we extracted five 10-second clips evenly spaced throughout the track. We pre-

[1] http://mturk.com

[2] http://hypem.com/list

User	Track	Clip	Tags	Num pairs
+	+	+	6.0370 ± 0.0290	2,566
+	+	−	2.3797 ± 0.0511	690
+	−	−	1.2006 ± 0.0026	227,006
−	+	+	1.1137 ± 0.0142	4,838
−	+	−	1.0022 ± 0.0083	13,560
−	−	−	0.5240 ± 0.0004	3,702,481

Table 1. Average number of tags ($\pm$ 1 standard error) shared by HITs with various characteristics in common and number of such pairs of HITs. A + indicates that the clips shared that characteristic, a − that they differed in it.

sented these clips to turkers in a random order, and generally multiple clips from the same track were not available simultaneously. Each clip was seen by 3 different turkers.

Mechanical Turk gives the "requester" the opportunity to accept or reject completed HITs either manually or automatically. In order to avoid spammers, we designed a number of rules for automatically rejecting HITs based on analyses of each and all of a user's HITs. Individual HITs were rejected if: (1) they had fewer than 5 tags, (2) a tag had more than 25 characters, or (3) less than half of the tags were found in a dictionary of Last.fm tags. All of a users' HITs were rejected if: (1) that user had a very small vocabulary compared to the number of HITs they performed (fewer than 1 unique tag per HIT), (2) they used any tag too frequently (4 tags were used in more than half of their HITs), (3) they used more than 15% "stop words" like **nice**, **music**, **genre**, etc., or (4) at least half of their HITs were rejected for other reasons. The list of stop words was assembled by hand from HITs that were deemed to be spam.

We pre-processed the data by transforming tags into a canonical form. We normalized the spelling of decades and the word "and", removed words like "sounds like" from the beginning of tags, removed words like "music", "sound", and "feel" from the ends of tags, and removed punctuation. We also *stemmed* each word in the tag so that different forms of the same word would match each other, e.g. **drums**, **drum**, and **drumming**.

We posted a total of 925 clips, each of which was to be seen by 3 turkers for a total of 2775 HITs. We accepted 2566 completed HITs and rejected 305 HITs. Some of the rejected HITs were re-posted and others were never completed. The completed HITs included 15,500 (user, tag, clip) triples from 209 unique turkers who provided 2100 unique tags. Of these tags, 113 were used by at least 10 turkers, making up 13,000 of the (user, tag, clip) triples. We paid approximately $100 for these data, although this number doesn't include additional rounds of data collection and questionnaire tuning.

2.1 Co-occurrence analysis

The first analysis that can be applied to these data is a simple counting of the number of tags shared by pairs of HITs. By categorizing the relationships between two HITs in terms of the users, tracks, and clips involved, an interesting picture emerges. Table 1 shows the first analysis of the number of shared tags for all possible pairs of HITs grouped by the relationships of these characteristics.

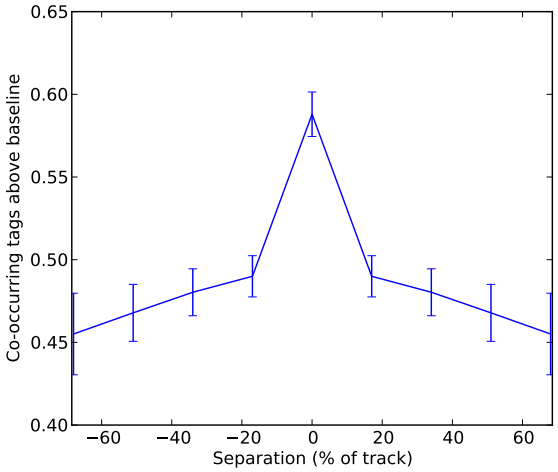


Figure 1. Average number of tags above the baseline shared by HITs from the same track as a function of the separation between the clips measured as % of a track.

The bottom row of the table shows that HITs with nothing in common still share 0.5240 tags on average because of the distribution of tags and music in this dataset. The second line from the bottom shows that HITs involving different users and different clips within the same track share 1.002 tags on average. And the third to last row shows that HITs with different users, but the same clip share 1.11 tags on average, significantly more than HITs that only share the same track. This same pattern also holds for HITs from the same user, but with higher co-occurrences. The large difference between HITs from the same user and HITs from different users can probably be attributed to the lack of feedback to the users in the task, allowing somewhat idiosyncratic vocabularies to perpetuate. Note that the top row of the table shows the average number of tags per HIT.

A related analysis can be performed measuring the dependence of tag co-occurrence on the distance between clips in the same track. Figure 1 shows the average tag co-occurrence of two clips in the same track above the baseline level of co-occurrences for two clips from different tracks. It reveals that the number of tags shared by clips decreases as the clips get farther apart. The error bars show that this result is not quite statistically significant, but it is still a notable trend. Results are similar for HITs from the same user and for cosine similarity instead of plain co-occurrence.

3. DATA MODELING

While stemming can make connections between certain tags in the dataset, it is only able to do this for tags which are syntactically related to one another. Another kind of model is required to capture relationships between tags like **indie** and **rock**. We choose to capture these relationships using a restricted Boltzmann machine (RBM), a generative

probabilistic model. The RBM observes binary vectors representing the tags that a single user gave to a single clip. Once trained, the model can compare the relative likelihood of two such observations and can draw samples from the observation distribution.

3.1 Conditional restricted Boltzmann machine

More formally, an RBM [9] is a probabilistic model of the relationship between binary visible units, denoted, v_i and binary hidden units, denoted h_j. Conditioned on the visible units, the hidden units are independent of one another, and vice-versa. The joint probability density function is

$$p(v,h) = \frac{1}{Z}\exp\left(v^T W h + b^T v + c^T h\right) \qquad (1)$$

where the partition function $Z \equiv \sum_{v,h} p(v,h)$ is computationally intractable (exponential either in the number of visibles or of hiddens). The likelihood of the observation v is obtained by marginalizing over h: $p(v) = \sum_h p(v,h)$, and can be computed easily up to Z. In this paper, we condition the model on "auxiliary" hidden units, a,

$$p(v,h \mid a) = \frac{1}{Z}\exp\left(v^T W h + v^T W_a a + b^T v + c^T h\right) \qquad (2)$$

where the partition function is now conditioned on a as well, $Z = \sum_{v,h} p(v,h \mid a)$. Conditional RBMs have been used for collaborative filtering [8], although in that case the conditioning variables influenced the hidden states, whereas in our model they directly influence the visible units. The matrices W and W_a and the bias vectors c and b are learned using the contrastive divergence algorithm [4]. In addition to the normal contrastive divergence updates, we place an L_1 penalty on W_a to promote sparseness of its entries.

In practice, the vector a is set *a priori* to represent the user, the artist, the track, and/or the clip using a so-called *one hot* representation. For example, each user has their own column of the W_a matrix, providing a different bias to the tag probabilities. We sometimes refer to the quantity $W_a a$ as the auxiliary biases for this reason. Each user in effect has a different baseline probability for the visible units, meaning that they tend to use the tags in different proportions. Because the entries of the W_a matrix are L_1-penalized, the user columns tend to represent discrepancies between a user's tags and the global average, which is captured in the bias vector b. Thus the W_a matrix is like a term frequency-inverse document frequency (TF-IDF) representation (see e.g. [14]) of the variables that it is modeling, but learned in a more probabilistically grounded way.

3.2 Purely textual datasets

We apply this model to three different tag datasets with the goal of discovering relationships between tags, and the tags that are used unexpectedly frequently or infrequently on particular items. The first dataset is purely textual, from Last.fm [1]. It includes (artist, tag) pairs, along with the number of times that that pair appears. The second dataset, from MajorMiner [7], includes (clip, user, tag) triples and also includes the audio associated with each clip. The third dataset, from the Mechanical Turk experiments described in Section 2, similarly includes (clip, user, tag) triples and audio. While it is smaller than the MajorMiner data, it includes many more clips per track, and so can provide perhaps more insight into clip-level and track-level modeling.

The dataset from [1] was collected from Last.fm in the spring of 2007. It includes the tags that users applied to approximately 21,000 unique artists and the number of users who applied each tag to each artist. There are approximately 100,000 unique tags, and 7.2 million (artist, tag) pairs, including duplicates. To reduce the size of the required model, we discarded tags that had been applied to fewer than 8000 artists (98 tags), and only kept the 200 most frequently tagged artists.

In order to transform this dataset into a form that can be used by the RBM model, we simulated taggings from individual users. We characterized each artist with independent Bernoulli probabilities over each tag and drew multi-tag samples from this distribution. The probability of each tag was proportional to the number of times each tag was applied to an artist, so the counts were first normalized to sum to 1. These normalized counts were multiplied by 5 (and truncated to prevent probabilities greater than 1) so that the expected total number of tags was 5, a number that a typical user might provide. To create the dataset, we repeatedly drew an artist at random and simulated a user's tagging of that artist. The artists' tag probabilities provided a baseline against which to measure the estimation of the relevant W_a columns, which only modeled artist auxiliary information.

The dataset from [7] was collected from the MajorMiner music labeling game over the course of the last three years. It includes approximately 80,000 (clip, user, tag) triples with 2600 unique clips, 650 unique users, and 1000 unique tags. Each observation was encoded as a binary vector indicating the tags that a single user applied to a single clip. The a vector in this case indicated both the clip, the track that it came from, and the user. On average, each track was represented by fewer than two clips.

Finally, this new Mechanical Turk dataset provides (clip, user, tag) triples along with relationships between clips and tracks. While it contains the fewest triples, it contains the most structure of the datasets because by design there are five clips per track. To model it, the a vector represents the user, the track, and the clip, so there is a separate auxiliary term learned for each of them.

3.3 Textual experiments

Qualitative experiments on the Last.fm dataset showed that our model successfully learned the auxiliary inputs, i.e. the W_a matrix acted as a sort of TF-IDF model for tags. Specifically, the W matrix modeled relationships between pairs of tags, the b vector modeled overall popularity of individual tags, and the columns of W_a modeled any tags that were unusually prevalent or absent for an artist given its other tags. For example, Nirvana's W_a column included a large value for **grunge**, and the Red Hot Chili Peppers' included a large value for **funk**, both of which might not

have been expected from their other tags like **rock** and **alternative**. Similarly, the Beatles have a negative bias for **seen live** because presumably fewer Last.fm listeners have seen the Beatles live than other artists tagged **rock** and **pop**. These issues are addressed more quantitatively below.

All three of the datasets described in Section 3.2 can be used in a leave-one-out *tag* prediction task. In this task, the relative probability of a novel observation is compared to that of the same observation with one bit flipped (one tag added or deleted). If the model has captured important structure in the data, then it will judge the true observation to be more likely than the bit-flipped version of it. This ratio is directly connected to the so-called pseudo-likelihood of the test set [2]. Because it is a ratio of probabilities, it does not require the computation of the partition function, Z, which is very computationally intensive. Mathematically, the pseudo-likelihood is defined as

$$\mathrm{PL}(v \mid a) \equiv \prod_i p(v_i \mid v_{\setminus i}, a) = \prod_i \frac{p(v \mid a)}{p(v \mid a) + p(\tilde{v}_i \mid a)} \tag{3}$$

where v_i is the ith visible unit, $v_{\setminus i}$ is all of the visible units except for the ith unit, and $\tilde{v}_i$ is the observation v with the ith bit flipped. Even though our observation vectors are generally very sparse ($\sim$4% of the bits were 1s), the 1s are more important than the 0s, so we compute the average log pseudo-likelihood over the 1s and 0s separately and then average those two numbers together. This provides a better indication of whether the model can properly account for the tags that are present, than the tags that aren't present.

This leave-one-out tag prediction can be done with any model that computes the likelihood of tags. Thus we can train models with different combinations of auxiliary variables, or different models entirely, as long as they can predict the likelihood of novel data. A baseline comparison to all of our RBMs is a factored model that estimates the probability of each tag independently from training data and then measures the likelihood of each tag independently on test data. Because of the independence of the variables, in this case the pseudo-likelihood is identical to the true likelihood.

We performed this experiment with the textual component of these three datasets, dividing the data 60-20-20 into training, validation, and test sets. The observations were shuffled, but then rearranged slightly to ensure that all of the auxiliary classes appeared at least once in the training set to avoid "out-of-vocabulary" problems. We ran a grid search over the number of hidden units, the learning rate, and the regularization coefficients using only the track-based auxiliary variables, those with the most even coverage. This grid search involved training approximately 500 different models, each taking 10 minutes on average. We selected the system with the best hyperparameters based on the pseudo-likelihood of the validation dataset. Once we had selected reasonable hyperparameters, we ran experiments using all combinations of the auxiliary variables with the other hyperparameters held constant. Five different random divisions of the data allowed the computation of standard errors.

The log pseudo-likelihoods of the test datasets under these systems are shown in Table 2. The results are not strictly comparable across datasets because they involved slightly different numbers of visible units. The results are shown on a per-bit basis, however, to facilitate comparison. These results show first that non-conditional restricted Boltzmann machines (rows with three $-$s) are much more effective than the factored models at modeling test data. This is because in addition to modeling the relative frequencies of tags, the RBM also models the relationships between tags through its hidden units. Conditioning the RBM on auxiliary information (rows with at least one $+$) further improves the pseudo-likelihoods. Specifically, it seems that the most useful auxiliary variable is the identity of the user, but the identity of the track helps as well. Including clip information is slightly detrimental, although not statistically significantly so, possibly because it introduces a large number of extra parameters to estimate in the W_a matrix from few observations.

	Auxiliary info			
Dataset	User	Track	Item	$\log(\mathrm{PL}) \pm$ stderr
MajorMiner	$+$	$+$	$+$	-0.9179 ± 0.0088
MajorMiner	$+$	$+$	$-$	-0.9189 ± 0.0070
MajorMiner	$+$	$-$	$-$	-0.9416 ± 0.0074
MajorMiner	$-$	$-$	$-$	-1.0431 ± 0.0095
MajorMiner		baseline		-1.4029 ± 0.0024
Mech. Turk	$+$	$+$	$-$	-0.893 ± 0.015
Mech. Turk	$+$	$-$	$-$	-0.904 ± 0.013
Mech. Turk	$+$	$+$	$+$	-0.914 ± 0.012
Mech. Turk	$-$	$-$	$-$	-1.039 ± 0.013
Mech. Turk		baseline		-1.300 ± 0.007
Last.fm	$-$	$-$	$+$	-0.5623 ± 0.0042
Last.fm	$-$	$-$	$-$	-0.7082 ± 0.0029
Last.fm		baseline		-1.1825 ± 0.0018

Table 2. Average per-bit log pseudo-likelihood (less negative is better) for restricted Boltzmann machines conditioned on different types of auxiliary information. A $+$ indicates that the auxiliary information was present, a $-$ indicates that it was absent. The baseline system is a factored model evaluated in the same way.

4. AUTOTAGGING EXPERIMENTS

The final set of experiments involves not just the textual tags, but also the audio for both the MajorMiner dataset and this new data collected from Mechanical Turk. In this experiment, we measure the usefulness of the RBM model from Section 3.1 for "smoothing" the tag data. Specifically, we create two datasets: the first, labeled "raw", consists of just the original (clip, user, tag) triples in the dataset. The second, labeled "smoothed", consists of labels imputed by the RBM trained with all of the available auxiliary information. For each clip, we drew 1000 samples from the RBM conditioned on that sample's auxiliary information, but with no user indicated. We factored out the user so the taggers were trained from a general point of view, not that of any

Mechanical Turk		
	Tested	
Trained	Raw	Smoothed
Raw	56.87 ± 0.52	56.56 ± 0.36
Smoothed	61.43 ± 0.51	63.40 ± 0.35

MajorMiner		
	Tested	
Trained	Raw	Smoothed
Raw	65.97 ± 0.49	60.58 ± 0.35
Smoothed	66.67 ± 0.49	63.09 ± 0.35

Table 3. Average classification accuracy and standard errors of autotaggers trained and tested on different tag labelings for Mechanical Turk and MajorMiner data. The tags were either raw or smoothed from RBM samples.

particular user. Because the model assumes the effects of user, track, and clip are additive on the tag probabilities, the effect of one can be factored out by not adding it. This is further ensured by the regularization of the W_a matrix, which forces many of the elements of the matrix to 0 and the rest to be small.

To compare these datasets, we hold the acoustic features constant, but change the labels used to train and test classifiers. We first split the data into 5 cross-validation folds. Then the positive and negative test examples for a particular tag are the top- and bottom-ranked clips from one cross-validation fold. The training examples are the top- and bottom-ranked clips excluding that fold. Because the cross-validation breakdowns are preserved across tag sets, it is possible to train on one tag set and test on another. For the smoothed dataset, we select the top and bottom 100 examples for each tag. For the raw counts, we choose for each tag the smaller of the top 100 examples or all of the examples verified by at least 2 people.

The autotaggers are inspired by those from [7], which use timbral and rhythmic features and a support vector machine (SVM) classifier. For this experiment we use LibSVM's ν-SVM as our SVM implementation, with probability estimates and a linear kernel [3]. Performance with the Gaussian kernel was similar. One binary SVM is trained per tag using a balanced number of positive and negative examples selected in order of tag affinity in the training set. Performance is measured in terms of average accuracy on a test dataset that is balanced in terms of positive and negative examples to set a constant baseline of 50% for a randomly guessing classifier. This metric is more appropriate than overall classification accuracy for tasks like autotagging where it is important to recognize positive examples in the presence of a large number of negative examples. To avoid the "album effect", the cross-validation folds were assigned so that clips from the same track were in the same fold in the Mechanical Turk data and that clips from the same album were in the same fold in the MajorMiner data.

The results of these experiments are shown in Table 3 and Figure 2. Each row of the tables represents a training tag labeling and each column represents a test tag labeling. The tables show these accuracies averaged over the 95 tags used by the most people on each dataset. The first column of each table shows the result of training on different tag labelings and testing on the raw tags. For both the MajorMiner and Mechanical Turk datasets, smoothing with the RBM improves test performance on the raw, user-supplied tags, although for the MajorMiner dataset, this difference is not statistically significant. The second column of each table indicates the performance of both models in predicting the smoothed data. In this case as well, the smoothed data trains more accurate models.

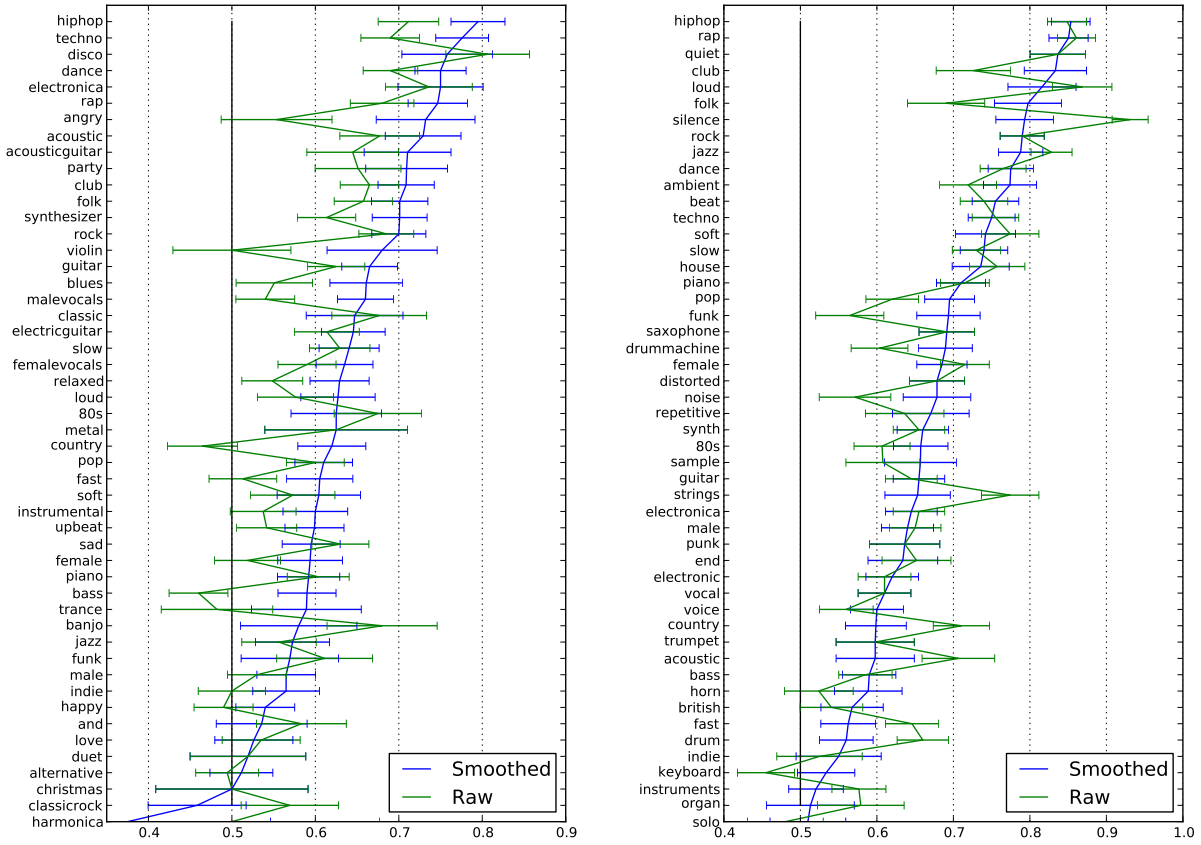
The diagonals of these tables show the "learnability" of the tag labelings. For the Mechanical Turk dataset, the smoothed tag set is more learnable than the raw tags. For the MajorMiner dataset, however, the raw tags are more learnable than the smoothed tags. These accuracies may not be directly comparable, however, because the measurements differ in both the models used and the test data. The difference in accuracy might indicate that the smoothing is less necessary in the MajorMiner dataset due to its larger size and larger number of repeated (clip, tag) pairs.

Figure 2 shows the autotag classification accuracy on the raw tags when trained with the raw and smoothed tags. The tags shown are the 50 used by the most people, and are sorted in the plots by the performance of the best system, that trained on the smoothed tags. For the Mechanical Turk data, shown in Figure 2(a), these smoothed tags train better classifiers almost across the board. Certain tags perform slightly better when trained on the raw data, but not significantly so. Smoothing is particularly useful for training **angry**, **violin**, and **country**, where autotaggers trained from the raw tags perform at chance levels.

For the MajorMiner data, shown in Figure 2(b), the smoothed tags and the raw tags perform similarly to one another. The smoothed tags train better autotaggers for **club**, **folk**, **pop**, and **funk**, while the raw tags train better autotaggers for **silence**, **strings**, **country**, and **acoustic**. The occurrence of the **silence** tag was due to the inclusion of a few broken clips in the game, which makes it a very specific, context-dependent tag that the RBM might not be able to generalize. It is not clear why performance on **country** is so different between the two datasets. It could be because in the Mechanical Turk dataset the top co-occurring tags with **country** are **guitar** 61% of the time and **folk** 27%, while in MajorMiner, they are **guitar** 44% of the time, **female** 27%, and **male** 26%. Thus in Mechanical Turk smoothing gives better results for **country** because it occurs more frequently with **guitar** and occurs with the more informative tag **folk**.

5. CONCLUSION

This paper has discussed the relationships between tags and music at a sub-track scale. We found that Mechanical Turk was a viable means of collecting ground truth tag data from humans, although the lack of the immediate feedback of a game might have contributed to lower inter-user agreement.



(a) Mechanical Turk

(b) MajorMiner

Figure 2. Accuracy of autotaggers for the top 50 tags in the Mechanical Turk and MajorMiner datasets. The autotaggers were trained on raw and smoothed tags and tested on the raw, human generated tags. Error bars show 1 standard error.

We also found that different parts of the same song tend to be described differently, especially as they get farther from one another. By modeling these differences with a conditional restricted Boltzmann machine, we were able to recover false negative tags in the user-generated data and use these data to more effectively train autotaggers, especially in smaller datasets. In the future we will explore additional models of tag-tag similarity, joint tag-audio models, and models of tagging that take into account the relationships between clips' different distances from one another.

Acknowledgements The authors acknowledge the support of an NSERC Discovery grant and would like to thank Razvan Pascanu and Johanna Devaney for their assistance.

6. REFERENCES

[1] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger: A model for predicting social tags from acoustic features on large music databases. *J. New Music Res.*, 37(2):115–135, 2008.

[2] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975.

[3] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

[5] P. Lamere. Social tagging and music information retrieval. *J. New Music Res.*, 37(2):101–114, 2008.

[6] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie. Evaluation of algorithms using games: the case of music annotation. In *Proc. ISMIR*, pages 387–392, 2009.

[7] M. I. Mandel and D. P. W. Ellis. A web-based game for collecting music metadata. *J. New Music Res.*, 37(2):151–165, 2008.

[8] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proc. ICML*, pages 791–798, 2007.

[9] P. Smolensky. *Information processing in dynamical systems: foundations of harmony theory*. MIT Press, 1986.

[10] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proc. Empirical Methods in NLP*, pages 254–263, 2008.

[11] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *CVPR Workshops*, pages 1–8, 2008.

[12] D. Turnbull, L. Barrington, and G. Lanckriet. Five approaches to collecting tags for music. In *Proc. ISMIR*, pages 225–230, 2008.

[13] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS 22*, pages 2035–2043, 2009.

[14] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1):18–34, 1998.

Predicting High-level Music Semantics using Social Tags via Ontology-based Reasoning

Jun Wang, Xiaoou Chen, Yajie Hu, Tao Feng
Institute of Computer Science and Technology, Peking University
{wangjun, chenxiaoou, huyajie, fengtao}@icst.pku.edu.cn

ABSTRACT

High-level semantics such as "mood" and "usage" are very useful in music retrieval and recommendation but they are normally hard to acquire. Can we predict them from a cloud of social tags? We propose a semantic identification and reasoning method: Given a music taxonomy system, we map it to an ontology's terminology, map its finite set of terms to the ontology's assertional axioms, and then map tags to the closest conceptual level of the referenced terms in WordNet to enrich the knowledge base, then we predict richer high-level semantic information with a set of reasoning rules. We find this method predicts mood annotations for music with higher accuracy, as well as giving richer semantic association information, than alternative SVM-based methods do.

1. INTRODUCTION

Semantic information extraction of music is given more and more emphasis based on the explosive growth of music resources. However, despite its high importance in a wide range of applications, there are various challenges in extracting semantic information from different existing resources. We sum up these existing information resources as three main classes:

Professional databases, web services, ontologies: These resources are created by professional data entry staff, editors, and writers. They commonly consist of basic editorial metadata such as names, titles, product numbers, biographies, nationalities, reviews etc., relational content such as similar artists and albums, influences, etc., and some culturally descriptive content such as styles, tones, moods, themes, etc. There are standard taxonomies forcing objects into predefined categories and the information is normally very precise, trustful and useful. However, information like descriptive content is expensive to generate, besides, the explosive growth of music has brought more and more challenge for manipulating such large scale content. Professional editors of those systems such as Allmusic and Pandora are hardly keeping pace with the ever-growing content.

Audio content: Currently content-based methods are the dominant players for automatic music information extraction. Some of the representative works can be referred to the Music Information Retrieval Evaluation eXchange (MIREX) [1]. However, the acoustic aspect is just one facet of music, besides there are unneglectable influences from subjectivity, social and cultural aspects, so high-level semantic information extraction purely from audio is quite an arduous challenge. For example, in the Audio Mood Classification evaluation (Hu et al. 2008), the resulting accuracies for 5-cluster mood classification was up to 61.5% in 2007, to 63.7% in 2008, and to 65.67% in 2009. Some mood perceptions are just too subtle and subjective, such as autumnal, brash, passionate, to be captured well enough by audio features only.

Social tags: Fortunately, nowadays the Web has become a primary host of a sizeable amount of text-based and semantic information. Web 2.0 technologies— e.g., Last.fm, MusicBrainz, and the so-called Shared Station in Pandora— have drastically augmented social media with rich context, such as user-provided tags, comments, reviews, folksonomies etc. By contrast to the above professional systems, these resources have some nontrivial advantages: flexibility to rapid content changes, intrinsically containing rich high level semantic information, etc. However, due to the noisy and unstructured data, existing systems are mainly based on simple keyword matching approaches, so knowledge from these resources is barely being well discovered.

The motivation is that the prediction of high level semantic metadata could benefit from a comprehensive consideration of information from multiple resources. We were inspired by a WordNet-based method proposed in [2] acquiring open-domain class attributes. In this work we propose a way to automatically identify the social tags' concepts. By mapping a music ontology to a semantic lexicon such as WordNet, we acquire more lexicalization of the concepts and better semantically classify/cluster the social tags (i.e. with more coverage), and we are also able to acquire in the ontology-based system the meaning and association between tags, to conduct reasoning on the resultant knowledge base giving a declarative representation with well-defined semantics, and to produce higher prediction accuracy for high level semantic data. By contrast to [2], our work is domain-specific, so it does not require applying extraction patterns to text and mining query logs to capture attributes. Instead, existing predefined professional taxonomies from reference systems are firstly mapped to an ontology's terminology, i.e. an ontology's terminology (TBox) consists of classes and roles, and secondly, we consider their finite set of terms as seed

axioms and propose a WordNet-based method to use these seed axioms to identify the most appropriate classes and roles for social tags, so that social tags can be mapped to the ontology's assertional axioms (ABox) associated with the constructed TBox. Lastly, we consider one of the most challenging tasks in MIR, i.e. mood cluster prediction, and perform a set of DL-safe reasoning rules on the resultant knowledge base (KB) to further augment the ABox with enriched mood annotation.

2. RELATED WORK

Recently researchers have brought up novel web-based methods for MIR tasks. In particular, some researchers have proposed approaches about automatically extracting music semantic information from the social tags. Luke et al. [3] consider social tags and web-mined documents as feature vectors and input them to Support Vector Machine (SVM), for classification to determine whether a song represents a certain tag. Bischoff et al. [4] apply SVM classifier to audio features and apply Naïve Bayes Multinomial to tag features, and then combine them in a programming way. Although significant improvements by combining web information are reported, these approaches dismiss the semantics of social tags or web-mined documents and we argue that some valuable information goes lost. We will look into a detailed comparison in our evaluation section. Algorithms originally developed in text information retrieval domain, such as Latent Semantic Analysis (LSA), probabilistic Latent Semantic Analysis (pLSA) and Latent Dirichlet Allocation (LDA) [5] can also be successfully adopted in MIR here, e.g., Levy et al. [6] and Laurier et al. [7] apply LSA method to gain an effective feature space with low dimensionality for capturing similarity. However, a LSA method has intrinsic limitations that the resultant dimensions might not have interpretable meaning, i.e., the derived semantic spaces still do not have explicitly defined semantics.

On the other hand, the extension of semantic information extraction to the field of knowledge representation formalisms has been widely deployed in the non-music-specific multimedia community. Great emphasis has been given to the extensional aspects of multimedia ontologies. There are many works in the literature proposed for managing multimedia data using ontologies, including image annotation, video annotation and recommendation [8, 9]. Exclusively for the domain of image and video annotation, novel works have been proposed for obtaining high level semantics. For example, Peraldi et al. [8] give a concrete example considering the interpretation of images of a sports event, and show how retrieval and interpretation of image data can be obtained by abductive reasoning; Penta et al. [9] proposed a novel ontology model for organizing low level multimedia data and semantic description. It exploits abductive reasoning to provide the most probable explanation of observed facts. All these works are using the benefits of ontology systems, which have scalability and extendibility capabilities to achieve effective image retrieval. However, to the best of our knowledge, ontology-based system for combining high level semantic information derived from social tags and professional taxonomies with information from audio features has rarely been studied in the music annotation domain.

3. SOCIAL TAG SEMANTIC IDENTIFICATION

For open-domain tasks as in [2], they heuristically choose the first sense uniformly in WordNet. Meanwhile the authors have pointed out, this heuristic is bound to make errors yet proved to be efficient enough in open-domain experimental results. However, this solution does not suit our work. As in a domain-specific system, the correct sense should be exclusive, e.g., Blues should be a kind of music genre rather than a color. Our approach will consider the fact that in professional music databases or web services, there are standard taxonomies forcing objects into predefined categories. While manually-constructed language resource WordNet has open-domain, wide-coverage conceptual hierarchies, by grouping terms and phrases with the same meaning into sets of synomyms, associated with the same definition. By mapping those predefined categories to WordNet, we acquire more lexicalization of the concepts and better semantically classify the social tags with more coverage.

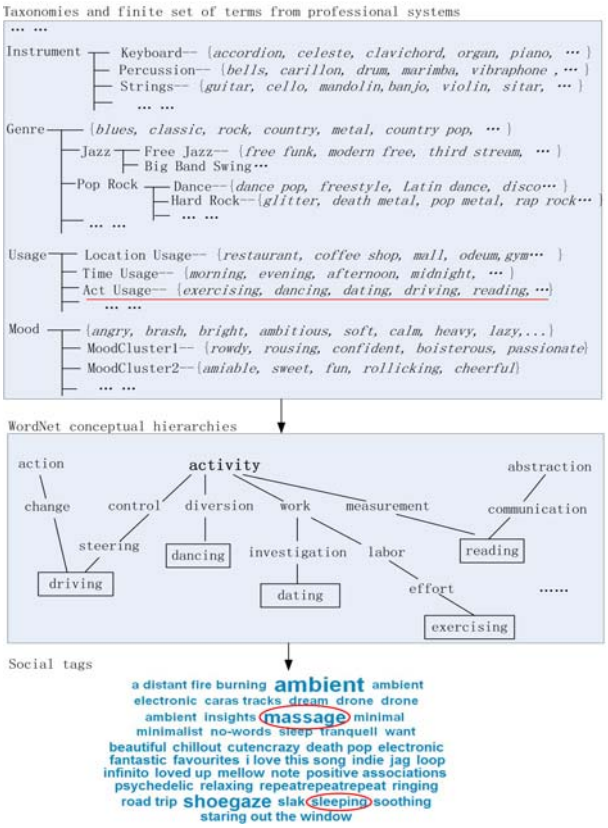


Figure 1. Social tag semantic identification framework.

3.1 Mapping to WordNet Concept Hierarchies

As shown in Fig. 1, the first task is to identify the most appropriate concept level in WordNet which best represents each category in the professional taxonomy. For each category $\mathcal{C}$ in the professional taxonomy, we consider its instances as seed words and retrieve them in WordNet. For each pair of seed nodes in WordNet, we find the closest common node in the upper level (ancestor node) which connects the two seed nodes via shortest path, then we get a set $\mathcal{T}$ of candidate ancestor nodes. Here we define a scoring function to select the best ancestor node in $\mathcal{T}$ as below:

$$Score(S) = \frac{\#descentSeeds_S}{\#Seeds_S} \cdot level_S \cdot \log(level_S) \qquad (1)$$

Where, $\#descentSeeds_S$ means the number of seed words that node S covers in its descent nodes, $\#Seeds_S$ means the number of seed words in the corresponding category $\mathcal{C}$, and $level_S$ means the depth from S to the top concept. Finally S with the highest score in $\mathcal{T}$ will be selected as the most appropriate concept in WordNet for the corresponding category $\mathcal{C}$. As an example in Fig.1, given a set of seed nodes <driving, dancing, dating, exercising, reading>, the approach detect "activity" as the most appropriate concept for this set rather than "action" or "abstraction".

Two facets have been considered in equation (1) defining the scoring function: concept specificity and concept coverage. On the one hand, the score is constrained by $level_S$ because if the level is too close to the top concept, then the node S would be too general and would harm the identification precision; on the other hand, the score is also constrained by $\#descentSeeds_S$, because if the level is too low and too specific, it would cause an insufficient coverage and harm the recall since many potential words which belong to the category would not be identified. Comparing to a simple linear function of $level_S$, the function defined in (1) experimentally gives an optimal tradeoff between coverage and identification precision.

3.2 Expanding Word List of Pre-defined Taxonomies

In this work, we adopt the taxonomies from Allmusic — a large-scale music database that provides professional reviews and metadata for albums, tracks and artists, and are frequently used for MIR research purposes [1]. In particular for mood annotation, for the convenience of evaluation and comparison to state of the art, we adopt the five cluster mood taxonomies from MIREX, which have been commonly adopted by the community.

The taxonomies are mapped to an ontology and it results in a TBox consisting of classes, related subclasses, roles of objects and datatype properties. Details about constructing the music ontology are dismissed here. Related similar works can be referred to [10]. XMLMapper tools can automatically transform available Web XML based resources (e.g. Allmusic.com) to an OWL ontology.

Once the concept has been identified via the approach as described in section 3.1, we construct a word list with more coverage for each pre-defined classes by retrieving the hyponyms, their synonyms and siblings, each with different weights (hyponyms> synonyms> siblings). In all, based on the pre-defined taxonomies it generates a word list with 71,022 words. While matching a tag with the word list, if the tag exactly matches a word in the list, it is then identified as the corresponding class directly; if the tag has words matching with different concepts and each with different weight, we only consider the word with the highest weight and match the tag with this word's related class; or else if the weights of different concepts equal, we simply consider the front words, as users usually put highly descriptive word in front of a less informing word, for example, "road trip", "brash band", although it is not always the case.

4. ONTOLOGY-BASED REASONING

So far, knowledge bases have been constructed using information from several different sources, including:

- Social tags identified with well-defined semantics
- Editorial and relational metadata from professional taxonomy systems
- Probabilistic classification output extracted from audio content

4.1 TBox and ABox Construction

As previously described, we map the taxonomies from Allmusic to the TBox of our ontology, and result in 155 classes and 62 roles in all. These roles consist of object properties indicating relationships between classes, such as <artist, song> <hasStyles> <genre>, <artist> <similarTo, follows, followedBy, inflences> <artist>, <artist> <performedVia> <instrument>, <song> <playedBy> <artist>, etc., and several datatype properties indicating data attributes of classes, such as <artist, song> <hasMoodProbability1> <"float">, <artist, song, genre, instrument, …> <hasConfidenceFactor> <"float">, etc.

In the following we illustrate steps and rules for ontology-based reasoning on music mood:

Initialization. Firstly, we define datatype properties <hasMoodProbability1, hasMoodProbability2,… , hasMoodProbability5>, of which each denotes prediction probability that the individual be classified into mood cluster1, cluster2,… cluster5. As shown in Fig.2, initial assertions about these mood probability properties of songs and tags are added in ABox. Given a tag having been identified into one of the mood clusters in the concept identification step, we assert an initial mood probability property, e.g., <0.0, 0.0, 1.0, 0.0, 0.0> for a tag identified as in mood cluster 3. For songs, we extract

112-dimension audio feature vectors via the library in jAudio toolkit, including intensity, timbre and rhythm features as well as the overall standard deviation and average values of Mel-frequency cepstral coefficients (MFCCs) and spectral shape features etc. We apply the feature selection library CfsSubsetEval in WEKA [11] and reduce the feature space from 112 to 23 dimensions, then we apply the SMV classification library in WEKA, and output the predication probabilities for each of the five mood clusters. For more details about the above content-based system, audience could refer to our previous work as FCY1 system in MIREX 2009 audio mood classification tasks. The output probabilities are asserted in ABox as the songs' initial value of datatype properties < hasProbabilityMood1, hasProbabilityMood2,…, hasProbabilityMood5>. These audio individuals initialized with mood probability properties, e.g., <0.25, 0.12, 0.33, 0.14, 0.16> in fig. 2, are to be considered as seed atoms as well.

Reasoning: Secondly, a set of DL-safe rules are applied on the ABox to infer mood probability for target atoms from seed atoms, as shown in fig. 2. Heuristically, different classes and roles should have distinct importance. For example, a song's mood could be inferred with higher confidence from its social tags describing mood than from its audio content. For another example, a song's mood could be inferred with higher confidence from its artist's mood style than from its genres'. Thanks to the well-defined semantics in the ontology, these factors can be efficiently considered in a semantic reasoning engine, e.g. Racerpro. We use nRQL in Racerpro [12], an expressive ABox query language for the very expressive DL *ALCQHI*_R+ (D-), to apply rules and generate new assertions to ABox. Besides many other advantages, the main reason we chose nRQL is that it allows for the formulation of complex retrieval conditions on concrete domain attribute fillers of ABox individuals by means of complex concrete domain predicates. Atoms of different classes are attached with several datatype properties which indicate their corresponding confidence/importance degree during the inferring process:

- Role Factor (RF): constant value related to the seed atom's object property, e.g, an artist <plays> a song, a song <hasStyle> a genre.
- Confidence Factor (CF): dynamic value indicating the overall confidence estimation about the precision of its mood prediction. Initial CF values for song and tag atoms are typically set as 0.3 and 1.0.
- Weighting Factor (WF): weighting value that a seed atom has while propagating its mood prediction to a target atom, so that mood prediction value that the target atom acquires could be weighted. We simply consider WF= CF*RF

We then apply rules on nRQL and generate new assertions in ABox. Given a set of triggered seed atoms of

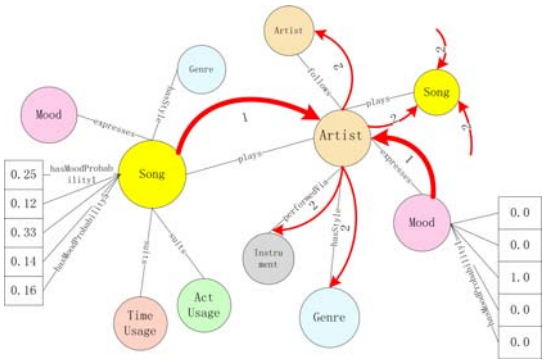


Figure 2. Applying reasoning rules between atoms.

mood tags $< t_1, t_2, \ldots, t_m >$ and songs $< s_1, s_2, \ldots, s_n >$, for example, Rule I is an illustrative rule as below:

Rule I:

x hasMoodProbability$I$ ← x is an artist;
$\{w_4 \cdot m_4, w_7 \cdot m_7, w_8 \cdot m_8,$ x plays $< s_4, s_7, s_8 >$;
$w_1 \cdot m_1, w_3 \cdot m_3\}$; x is tagged with $< t_1, t_3 >$;
x have Confidence Factor $< s_4, s_7, s_8 >$ hasMoodProbability$I$
$c_x = \text{maximum}\{w_4, w_7, w_8, w_1, w_3\}$ $< m_4, m_7, m_8 >$;
$< s_4, s_7, s_8 >$ have Weighting Factor
$< w_4, w_7, w_8 >$;
$< t_1, t_3 >$ hasMoodProbability$I$ $< m_1, m_3 >$;
$< t_1, t_3 >$ have Weighting Factor $< w_1, w_3 >$

where I denotes the index of mood clusters. The accumulated mood probability values are summed up and normalized to ensure the sum probability of all clusters equals 1. In the above example, the artist atom x is triggered and continues to be use as seed atoms for further reasoning iterations. The rules are applied iteratively until no more atoms are triggered. Rule II and III are another two illustrative rules as below:

Rule II: y hasMoodPro bability$I$ ← x is an artist; y is an artist;
$RF_{xy} \cdot c_x \cdot m_x$ y is similar to x;
x hasMoodPro bability$I$ m_x;
x has Confidence Factor c_x

Rule III: z hasMoodPro bability$I$ ← x is an artist; z is a genre;
$RF_{xz} \cdot c_x \cdot m_x$ x has styles z;
x hasMoodPro bability$I$ m_x;
x has Confidence Factor c_x

5. EXPERIMENTAL SETTING

Our album consists of 1804 songs, covers about 21 major genres, 1022 different artists, and evenly covers mood labels created by professional Allmusic editors with one or more terms in one of the five mood clusters.

On one hand, each song is processed into 30s, mono, 22.05 kHz, .wav audio clips. We then apply the content-based system described in section 4.1. This system gives an accuracy of >60% for the data set of MIREX mood classification task, yet the same system gives a much lower accuracy of ~40% for our data set, which mainly

due to the album's larger scale, larger diversity and lacking of manually pre-filtering of ambiguous songs.

On the other hand, we crawled tags from Last.fm and collected 65,272 tags in all for 1364 songs in our album, while the remaining 353 songs do not have any tags. After removing the duplicate ones, we have got 15,400 distinct tags in all. Despite many spelling errors and noisiness in these 15,400 tags, we manage to identify most of them with semantic-rich classes: 47% tags into subclasses of genre, 13% into subclasses of mood, 3% into subclasses of usage, 2% into subclasses of instrument and finally augment the ABox with 10,015 concept assertions.

6. EVALUATION

6.1 Social Tag Semantic Identification

We test the precision of identifying subclasses of genre, mood, usage and instrument. There are in all 141 atomic concepts which are related to the above 4 main classes and the deepest level is 4.

Ground Truth: To create the ground truth for the evaluation, we randomly sampled 3300 tags from the 15,400 tags. These tags are then manually labeled with classes by three people. The labels are cross-checked and the ones with inconsistent labels are considered as obscure tags. The level of the classes is required to be as deep as one can identify. Among the 3300 tags, there are 1085 tags labeled with subclasses belonging to the 4 main classes, and the remaining 1915 tags consists of: the obscure ones, the ones belonging to other classes, the ones not belonging to any classes, concrete specific terms such as artist names and manually unrecognizable spelling errors.

Identification Precision: As shown in Fig.3, among the tested 1085 tags, there are 89 tags which can be recognized by human yet not in WordNet because of the aberrant spelling, which is very common in user tags; there are 137 tags yet not covered in our resultant word list; there are 203 tags mis-identified. Counting in the 150 tags correctly identified in the same main class of the label, the 150 ones exactly identified in the same subclass of the label, and the 8 ones identified in the sibling class of the label, the result gives a precision of 60.5%. In the future work, we will add some nature language processing (NLP) and information retrieve (IR) techniques including stemming, lemmatization and token normalization, to improve the social tag semantic identification precision.

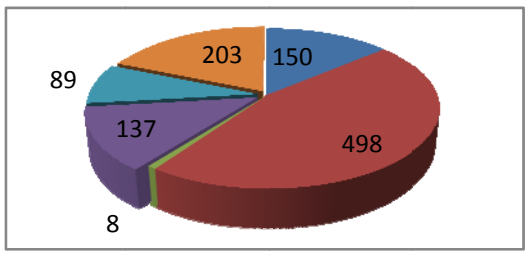


Figure 3. Distribution of Tag Identification Result.

6.2 Ontology-based Reasoning versus SVM Method

SVM is a widely applied machine learning method and has been found superior to other classifiers in many cases of MIR. It takes attributes as features vector input, dismissing their meanings and semantic associations, and learns the separating hyperplane that maximizes the margin between two classes. In contrast, the ABox reasoning method considers the semantic associations between classes and applies inference services with a set of DL-safe rules, i.e., given a precondition, it outcomes a set of ABox assertions as consequences. In our case, these consequences are probability prediction about a song's mood.

For SVM-based system, we transform the social tag information into feature vectors as input. As the very large 15,400-dimension and very sparse feature space of social tag disrupt the SVM-based system, we reduce the dimension to 2919 by filtering out the tags which have occurrence frequency less than 3 times. Moreover, we apply Principal Component Analysis (PCA) method to further reduce the redundancy and map the 2919-dimension social tag feature space to a 982-dimension feature space. The system using 2919-dimension feature vectors reaches average classification accuracy **52.0528%** in 3-fold cross validation, while the system using 982-dimension feature vectors reaches **54.6921%** in 3-fold cross validation. Hence we adopt the 982-dimension feature space for the SVM-based system.

To ensure that the reasoning-based system and the SVM-based have input information as fair as possible, we consider both the social tags and content-based attributes as input information for both systems: for reasoning-based system, we add assertions about the identified social tags into ABox and initiate each

Table 1. Confusion matrixes of SVM-based and Reasoning-based systems

Actual / Prediction	SVM-based				
	C1	C2	C3	C4	C5
C1	**0.29**	0.11	0.01	0.08	0.12
C2	0.24	**0.54**	0.08	0.11	0.03
C3	0.08	0.16	**0.71**	0.16	0.06
C4	0.15	0.18	0.15	**0.52**	0.18
C5	0.24	0.01	0.05	0.13	**0.61**
	Reasoning-based				
C1	**0.59**	0.18	0.06	0.10	0.16
C2	0.13	**0.54**	0.08	0.07	0.01
C3	0.09	0.15	**0.68**	0.16	0.07
C4	0.08	0.12	0.09	**0.59**	0.07
C5	0.11	0.01	0.09	0.06	**0.69**

song's datatype properties— hasProbabilityMood1 etc.— with mood prediction probabilities learned from the 112-dimension audio features, as described in the experiment setting section; for SVM-based system, we construct its feature space by combining the 112-dimension audio features with the 982-dimension feature space mapped from social tags.

Tab. 1 gives the confusion matrix of each system, where C1 to C5 indicate the five mood clusters. The SVM-based system achieves an average classification accuracy of **55.7185%** in 3-fold cross validation. The reasoning-based system achieves prediction accuracy of **62.07%**, which outperforms the SVM-based system, as well as having a more even precision distribution among clusters. The SVM-based system gives better precision only in predicting mood cluster3, indicating that SVM-based method can well discriminate cluster3 (brooding, poignant, sad, somber, etc.). This has also been reflected in MIREX [1] reports.

6.3 Knowledge Base Enrichment

Relational content such as similar artists and albums, influences, follows, etc., are much less expensive to acquire from professional systems than for high level semantic information like music mood and usage. In all, there are 29,253 assertions acquired from Allmusic about the relational content such as <artist> <influences, similar to, follows> <other artists>.

To evaluate the prediction performance, we conduct a prediction process on artist atoms in the Knowledge Base. To simplify the process, we consider an artist's tags and mood cluster the same as his song. We partition the artist axioms who are players of the album— so that we have the ground truth as their song's mood label— into two complementary subsets: a "known" subset A_516 (with 516 artist atoms) having ABox assertions generated from editorial metadata and social tag information, and the other is "unknown" subset A_512 (with 512 artist atoms) to be predicted and validated. To reduce variability, we perform another round by changing the A_512 to "known" subset. After the reasoning process, we have got 461 artists in A_512, and 469 artists in A_516, who gained mood prediction via the inferring rules. The prediction precision is 50.76% for A_512 and 50.32% for A_516 and the average precision is 50.54%. This prediction method could be effective, given random five-mood-cluster classification's precision is as low as 20%.

Some interesting knowledge can also be discovered. For example, genre atoms gain a set of mood prediction datatype value during the semantic reasoning, and after accumulation and normalization, some of them reflect very strong associations with mood. Tab. 2 lists the result of genre atoms ranked by their bias degree among mood clusters, which is in good accordance with people's judgement and discovers the implied semantic associations.

Table 2. Ranking genre atoms according to mood bias

Genre	Mood probability prediction				
	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5
Solo istru.	0	0.14	**0.83**	0.03	0
Halloween	0.01	0.23	0	**0.76**	0
Noise	0.13	0.07	0	0.07	**0.73**
Comedy	0.1	0.06	0.06	**0.71**	0.07
Sad core	0.01	0.03	**0.71**	0.09	0.16
Punk metal	0.32	0	0.04	0	**0.64**
Children's	0	**0.61**	0	0.39	0
Sweet band	0.20	**0.58**	0.14	0.08	0
Hair metal	**0.54**	0.13	0.05	0.09	0.18
Skiffle	**0.53**	0.31	0	0.04	0.12

7. CONCLUSION

We found that by unleashing music related information from various resources via an ontology-based system and by considering the internal semantic links for reasoning, we achieve a significant precision improvement for predicting mood. To augment the knowledge base efficiently and to make it free of manual annotation, we propose a WordNet-based method to map social tags to a predefined taxomony. Although in this work we mainly discuss mood, since it is one of the most representative high-level music semantic information, we argue that the proposed method could also be applied for predicting other high-level semantics, for example, if music usage or genre style are of interest for an application, we could adjust the initiation processes and modify corresponding reasoning rules accordingly, so this work has potential applications for other tasks of music recommendation, indexing, documentation and retrieval.

8. REFERENCES

[1] Hu, X.; Downie, J. S.; Laurier, C.; Bay, M.; Ehmann, A. F.: "The 2007 MIREX Audio Mood Classification Task: Lessons Learned," ISMIR'08.

[2] Pasca, M.: "Turing Web Text and Search Queries into Factual Knowledge: Hierarchical Class Attribute Extraction," In Proceedings of the 23rd AAAI Conference, 1225-1230.

[3] Barrington, L.; Yazdani, M.; Turnbull, D.; Lanckriet, G.: "Combining Feature Kernels for Semantic Music Retrieval," ISMIR'08.

[4] Bischoff, K.; Firan, C.S.; Paiu R.; Nejdl, W.; Laurier, C.; Sordo, M.: "Music Mood and Theme Classification- a Hybrid Approach," Proc. of ISMIR 2009.

[5] Blei D.; Ng A.; Jordan M.: "Latent Dirichlet Allocation," Journal of Machine Learning Research, Vol. 3, pp.993–1022, Jan. 2003, MIT Press.

[6] Levy, M.; Sandler, M.: "A Semantic Space for Music Derived from Social Tags," In Proceedings of ISMIR 2007.

[7] C. Laurier, M. Sordo, J. Serrà, P. Herrera: "Music Mood Representations from Social Tags," Proc. of ISMIR 2009.

[8] Peraldi, S. E.; Kaya, A.; Melzer, S.; Moller, R.; Wessel, M.: "Multimedia Interpretation as Abduction". Proc. DL 2007.

[9] Penta, A.; Picariello, A.; Tanca, L.: "Multimedia Knowledge Management Using Ontologies," IEEE Intelligent Systems, 2003.

[10] F. Giasson and Y. Raimond: "Music ontology specification. Online ontology," 2008.

[11] Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B. Reutemann, P.; Witten I.H.: "The Weka Data Mining Software: An Update," SIGKDD Explorations, Vol. 11, Issue1.

[12] Haarslev,V.; Moller,R.; Wessel,M.: "RacerPro User's Guide and Reference Manual Version 1.9.2".

UNDERSTANDING FEATURES AND DISTANCE FUNCTIONS FOR MUSIC SEQUENCE ALIGNMENT

Özgür İzmirli
Center for Arts and Technology
Computer Science Department
Connecticut College
oizm@conncoll.edu

Roger B. Dannenberg
School of Computer Science
Carnegie Mellon University
rbd@cs.cmu.edu

ABSTRACT

We investigate the problem of matching symbolic representations directly to audio based representations for applications that use data from both domains. One such application is score alignment, which aligns a sequence of frames based on features such as chroma vectors and distance functions such as Euclidean distance. Good representations are critical, yet current systems use ad hoc constructions such as the chromagram that have been shown to work quite well. We investigate ways to learn chromagram-like representations that optimize the classification of "matching" vs. "non-matching" frame pairs of audio and MIDI. New representations learned automatically from examples not only perform better than the chromagram representation but they also reveal interesting projection structures that differ distinctly from the traditional chromagram.

1. INTRODUCTION

Score alignment [4], score following [3], chord and key recognition [6, 7] chorus spotting [1, 8], audio-to-audio alignment [9, 13] and music structure analysis [2, 11] are all tasks where it is useful to compare two segments of music. A common representation for this is the chromagram [1], a sequence of chroma vectors, where each vector typically has 12 elements and each element represents the energy corresponding to one pitch class in the spectrum but not necessarily one pitch class in the score. Most algorithms use a distance function in conjunction with the chromagram representation to measure the similarity between frames. While it may be obvious, especially in hindsight, why the chromagram works well in many applications, it should be noted that the chromagram is a contrived representation, and there is no reason to believe it should be optimal. Very little research has been conducted on alternative ways to compare audio to audio let alone audio to symbolic representations. The existing approaches are generally domain specific. For example, in [12] the chromagram is made less timbre dependent by discarding the lower mel-frequency cepstral coefficients and then projecting the remaining coefficients onto the twelve chroma bins. Another example can be found in [15] in which a binary chroma similarity measure is used for alignment in the context of cover song detection. In this work, we explore various ways to derive good features and functions from real data. We specifically look at the problem of score alignment directly from a MIDI representation to audio without going through a synthesized version of the MIDI data. In this paper we give a formulation based on the score alignment task; however, results should be applicable to all other problems that require frame-based comparison. The goal of this work is to gain insight into why the chromagram works in practice and to learn what modifications might make it work even better. Our results suggest that there is room for at least some improvement.

2. THE SCORE ALIGNMENT TASK

Our work is aimed at optimizing score alignment: finding a mapping from a symbolic score or standard MIDI file to an audio recording. The basic algorithm transforms both the MIDI file and the audio file into chromagrams **A** and **B**, which are sequences of chroma vectors. We will denote the chroma vector corresponding to the i^{th} time frame (column) of **A** as $\mathbf{A}_i$. Then, construct a distance matrix $\mathbf{D}_{i,j} = f(\mathbf{A}_i, \mathbf{B}_j)$, where f is a distance function. The idea is that f is small when $\mathbf{A}_i$ is "similar" to $\mathbf{B}_j$ and large otherwise. Often, f is based on the cosine distance, correlation distance, or Euclidean distance from $\mathbf{A}_i$ to $\mathbf{B}_j$. The next step uses dynamic programming to find the lowest-cost path from $\mathbf{D}_{0,0}$ to $\mathbf{D}_{m-1,n-1}$, where m and n are the number of frames in **A** and **B** respectively.

Path smoothing or constraints may be useful to obtain even more accurate alignment. Experience has shown that the chromagram representation for audio, and a chromagram-like representation for MIDI data [9] results in a very robust score alignment algorithm. However, the chromagram is an arbitrary choice. There are many other possible features, including the spectrum and mel cepstrum, and even the chromagram has parameters including the range of spectral bins considered. How can we search for better representations and distance functions?

3. THE LOG-FREQUENCY SPECTRUM OR "SEMIGRAM" REPRESENTATION

Although we are interested in learning better representations and distance functions, it would be difficult to learn a relationship between audio and symbolic representations starting from raw signal frames and raw MIDI data. To simplify the representation, we use a magnitude spectrum with bins logarithmically spaced by semitones (12 bins per octave). The input audio is downsampled to a sampling rate of 11025 Hz. A frame duration of 93 ms with 50% overlap is used. This representation is able to resolve semitone differences in frequency across the spectrum with far less data than the standard magnitude spectrum where each bin has a constant bandwidth. By analogy to the spectrogram, we call this representation the *semigram* **S**: a matrix where each column is a *semi vector* and each semi vector element represents the magnitude associated with the frequency range of one semitone. We note that the traditional 12-element chromagram can be understood as an octave-folded version of the semigram.

For MIDI data we construct a similar representation, a matrix **R** (also called a semigram), where each column represents a time window and each row represents a pitch (key number). If only one note is sounding in the time window at a given pitch, the matrix element is the note's MIDI velocity. If the note is not on during the entire time window, the velocity is weighted by the fraction of time the note is on. If there is more than one note on at the given pitch, the maximum of the weighted velocities is used.

4. TRAINING DATA FOR LEARNING PROJECTIONS

One way to search for good distance functions is simply to attempt alignment with various parameter settings, but this kind of evaluation is difficult. How do we score alignments? And if the chromagram is already robust, then it might take a huge number of examples to find enough failure cases for another method to show improvement.

Another possibility is to change the task. In our study, we use a classification task that labels frame pairs as "matching" or "non-matching." We assume that optimizing performance on this task will also be very good for the alignment task. We derive labeled training data (for supervised machine learning) from aligned scores, using 7 orchestra and wind ensemble recordings from one collection and 2 sets of 20 pieces from the RWC classical collection, as listed in Table 1. CLA1 consists of mostly symphonic pieces whereas CLA2 is a random selection of pieces with different combinations of instruments.

The alignment for the orchestra and wind ensemble recordings was done using chromagrams, but post processed with some spline fitting and smoothing techniques that generally improved the perceptual alignment. The alignment for the RWC pieces are taken from alignment data provided by Ewert, Müller, and Grosche [5]. The alignments of the corresponding scores were verified to be acceptable by listening to the MIDI synthesized versions simultaneously with the original audio.

From the aligned data, it is simple to extract all matching frames. To increase the number of matching frames and reduce overfitting to specific keys, we transpose the matching frames up to +6 and –5 semitone steps, thus covering all 12 chromatic degrees. To obtain non-matching frames, we select a random frame from audio for each frame from the MIDI data. These randomly selected pairs will run the gamut from very similar to very different, but for training purposes, we consider them all to be examples of "non-matching." For the training, the number of "non-matching" frames is equal to the number of "matching" frames including transpositions. All audio listed in the table was used for training, resulting in about 10^6 matching and the same number of non-matching frame pairs after transposition.

Table 1. The training data.

Recording	ID	Duration(secs.)
Tarantella from Incidental Suite, C. T. Smith	TAR	127
Nocturne from Incidental Suite, C. T. Smith	NOC	351
The Music of Disneyland, arr. by J. Brubaker	DIS	499
Medieval Legend, M. Story	LEG	248
The Travelin' Hat Rag, D. Bobrowitz	HAT	162
The Thunderer, J. Sousa	THU	148
Rondo from Incidental Suite, C. T. Smith	RON	168
RWC Classical Music Collection (20 pieces)	CLA1	1182
RWC Classical Music Collection (20 pieces)	CLA2	1192

5. LEARNING A FEATURE VECTOR

As a preliminary study, to find a good distance function for alignment, we trained a multi-layer perceptron neural network to classify semi vector pairs as "matching" or "non-matching." The inputs were midi and spectral vectors and the output was trained to be 0 or 1 based on whether the vectors were matching or not. 20 hidden nodes were used. We trained this on a particular set of three pieces: HAT, LEG and RON. These yielded 92.3% accuracy on the training data. Testing individually we obtained HAT: 89.5%, LEG: 93.0%, RON: 90.5%, TAR: 83.8%, NOC: 85.3%, DIS: 87.7 % and THU: 86.9%. We also trained the neural network separately on the 20-piece RWC sets and tested on the remaining 36 pieces in that set. We obtained 94.0% and 86.4% accuracy for CLA1, and 90.2% and 89.0% accuracy for CLA2 on the training and test data respectively. These results showed us that a model of this nature could generalize a matched-unmatched classification quite well with the given input representations. To reiterate our aim in this work, we are interested in understanding why chroma vectors work so well, whether they work better than a trained neural net, and whether variations can work even better. After all, the chroma vector is basically one particular projection from the semi vector. We can use machine learning to explore the space of projections and visualize the results to gain better understanding of the nature of the projections that work better.

Let us first write the chroma vector computation as a projection. For MIDI data, we have the $p \times m$ semigram

R. We define an $r \times p$ matrix **L** (r = 12) that projects each semi vector (column) of **R** to a chroma vector. Similarly, we define an $r \times q$ matrix **M** to project the audio q x m semigram **S** to chroma vectors:

$$\mathbf{A} = \mathbf{LR} \quad (2)$$

$$\mathbf{B} = \mathbf{MS} \quad (3)$$

Matrices A and B consist of pairs of feature vectors resulting from the respective projections in L and M. We first use this framework with fixed projections and then generalize the approach by training these projections to better understand their nature and compare them with the commonly used ones. Figure 1 illustrates the standard form of M (and similarly for L), which collapses octaves in the semigram to form a 12-element chroma vector. Note that the frequency ranges corresponding to the audio semigram (the horizontal axis) are labeled with midi numbers.

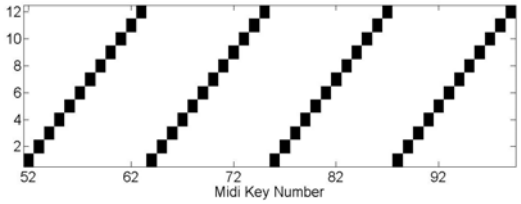


Figure 1. The conventional projection from semigram (log-frequency discrete magnitude spectrogram) to chromagram.

Next, a standard distance is taken between two corresponding feature vectors to obtain a measure of similarity. Hence, the required distance for the score alignment algorithm in terms of two input semi vectors $\mathbf{R}_i$ and $\mathbf{S}_j$ is given by $f(\mathbf{A}_i, \mathbf{B}_j)=\boldsymbol{C}(\mathbf{LR}_i, \mathbf{MS}_j)$, where f represents the desired distance and $\boldsymbol{C}$ is the centered cosine distance (found by first removing the means of the vectors and then calculating the cosine distance). To obtain a binary output ("matched" or "non-matched"), the distance is compared to a fixed threshold. This result is used for evaluation, but for training, we use the continuous real value as the output and try to train the system to output a zero (0) or one (1) value.

Now, suppose we generalize the chromagram to allow *any* projection. Although this is not a neural network, the back-propagation algorithm can be used to learn weights for the matrices **L** and **M**. The basic idea is to evaluate the partial derivative of the output with respect to each element of each matrix. Then, for each training example, the partial derivative for each coefficient is scaled by the output error, multiplied by a small rate parameter, and subtracted from the coefficient, thus adjusting each coefficient in a direction that would move the output closer to the correct value. This update is applied to all elements in the **M** and **L** matrices for all training frame pairs, and this process is iterated many times until the output converges. Given a large enough dimension r, this gradient descent algorithm will normally converge to a local optimum.

We can write $\boldsymbol{C}(\mathbf{LR}_i, \mathbf{MS}_j)$ as $D(x_k, \mathbf{R}, \mathbf{S})$ where x_k is some element of **M** or **L**, letting the remainder of **M** and **L** be constants for the moment. We can then evaluate $D(x_k + eps, \mathbf{R}, \mathbf{S}) - D(x_k, \mathbf{R}, \mathbf{S})$ to estimate the partial derivative of D with respect to x_k. The learning algorithm is as follows:

```
while convergence criterion not met
   for all pairs R and S
      for each parameter indexed by k
         delta_k = D(x_k + eps, R, S) - D(x_k, R, S)
         error_k = D(x_k, R, S) - GT
         new x_k = x_k - alpha * error_k * delta_k
```

In this algorithm, eps is a small number used to calculate the derivative, GT is the ground truth and it has a value of 0 when S matches R and 1 otherwise. The constant alpha is the learning rate.

The training can be performed in different ways. Normally, both matrices co-learn but it is also possible to fix the weights of one matrix and learn the other. The initial values for both matrices can be assigned to chroma mappings or assigned random values. In addition to this, different learning rates for the matrices can be set.

One advantage of using a linear projection (multiplication by **L** and **M**) to obtain paired feature vectors is that the matrix can be visualized to give some insight as to what features are being used by the system. For example, if the chromagram representation were optimal, we would expect **L** and **M** to maintain their projections shown in Figure 1 during training. In contrast, Figure 2 shows the actual result of learning matrix **M** starting from a chroma mapping. In this particular case, the learned weights are systematically different.

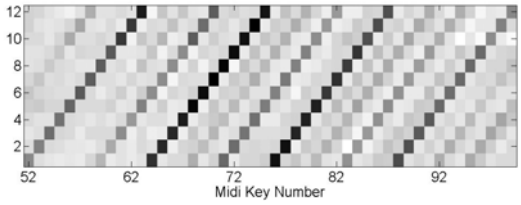


Figure 2. The trained matrix M with initial chroma pattern and fixed L with chroma projection (as in Figure 1).

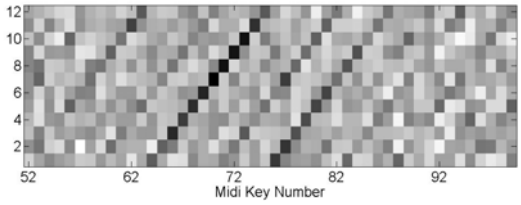


Figure 3. The trained M matrix with random initial values and initial chroma pattern for L.

In comparison to the preceding two figures, Figure 3 shows a trained **M** where initial weights were random. The **L** matrix had initial values for the chroma projection and was allowed to co-learn with **M**. The matrix in Figure 3 is similar to the chromagram in that each row

corresponds to the detection of a different pitch class or chroma. In at least some of the rows, there is a clear pattern of high weights separated by octaves.

The matrix in Figure 2 differs from the chroma mapping in several ways. First, the matrix is not symmetric, but this would be expected from the asymmetry of the training data and the nature of the training algorithm. Second, the rows are not just selecting octaves and pitch classes. It has been noted that the chromagram does not really compute the strength of 12 pitch classes because harmonics of the fundamental will generally include energy in bins that are mapped to other pitch classes. Here, we see that learned rows are selecting not only octave-related frequencies, but some fifths, thirds and other relationships. In fact, the rows are quite similar, at least for the octaves, fifths and major thirds, to pitch histograms for diatonic scales and the Krumhansl template [10] for key finding. This relationship has been studied in [16]. In this study, the empirical profiles have been found to present statistically significant correlations with tonal profiles obtained from human judgments. They demonstrate this by extracting tonal profiles based on covariance analysis of chroma features computed from western tonal musical recordings. Similarly, in our case, we find that the rows contain effects of both pitch distributions and overtone strength distributions. It seems likely that all of these factors play a role in determining the optimal patterns. A third property we can observe in the learned matrix is that low and high frequencies seem to have less significance. There is more variation between rows in the middle frequencies. In this work, the note range for the midi semigram was chosen to be from E_1 to $D\#_7$ and the range for the audio semigram was chosen to be E_3 to $D\#_7$. The audio semigram has a shorter note span than the midi semigram because of the time-frequency trade-off for the given time window length, which is kept short in the interest of higher time resolution.

We have learned the matrices many times using different training data and different initial conditions. Ideally, the matrices would converge to a configuration where the 12 rows represent 12 unique transpositions of some underlying pattern. To test this, we can rotate each row left and right until the correlation with a commonly used pitch distribution, such as the Krumhansl template, is maximized. For this, the pitch-class Krumhansl template is unwrapped to span multiple octaves and is weighted by a Hann window. The choice of the type of pitch distribution is not critical because the purpose of the window is only to shift the elements in a row to line up with the other rows. Figure 4 shows the aligned matrix **M**, the averages over rows of **M** and the weighted Krumhansl template used in the alignment. We observe that there is usually a unique shift (modulo 12) for each row of a pattern that is somewhat similar between rows of both matrices. However, there are also some irregularities possibly due to registral pitch effects and co-learning dynamics of the matrices.

It is reasonable to assume that there is some underlying "ideal" pattern that is learned in 12 different transpositions. Next, we test this assumption by forcing all 12 rows to contain the same basic pattern, shifted by 12 different offsets. First, we average the aligned matrix over all rows to find the estimated "ideal" pattern as shown in the middle plot of Figure 4. We then form a new matrix by copying the "ideal" pattern into every row and then un-rotating the rows according to the rotations performed to obtain the aligned matrix. The effect is to force the matrix to a more symmetric configuration and perhaps eliminate any overfitting due to the many degrees of freedom offered by an unconstrained matrix. Figure 5 shows the resulting un-rotated matrix. We can then re-evaluate the test data with the new matrices and compare the performance to the trained versions. This shows us how well the single pattern captures the essential information. The evaluations have been carried out with this process applied to **L** and **M** separately.

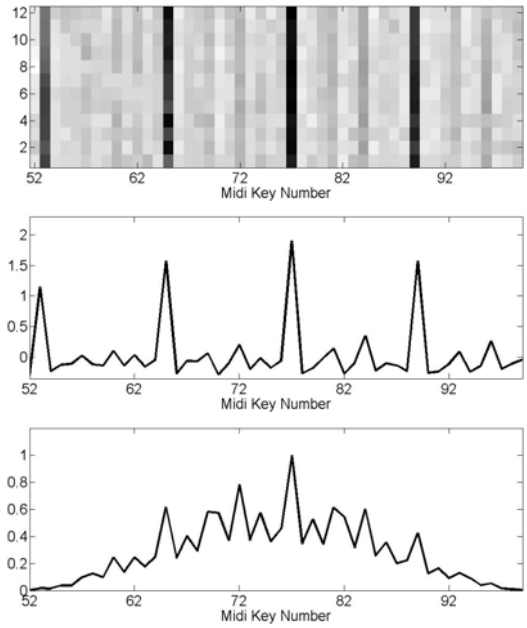


Figure 4. Matrix M aligned (upper plot). Average over rows of the aligned matrix (middle plot) and the weighted multi-octave Krumhansl template (lower plot). The sub-peaks in the middle plot represent major 3rds and perfect 5ths or perhaps 5th and 3rd harmonics.

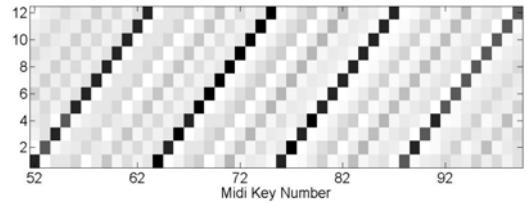


Figure 5. Un-rotated M after averaging over rows of the aligned matrix.

Learning can alternatively be started from random weights in both matrices. In this case, comparable classification accuracy is achieved, however, neither

matrix exhibits an easily visualizable structure similar to those seen in the preceding figures.

6. EVALUATION

Several different evaluations have been carried out on the data set. The first evaluation used a group consisting of 4 pieces (TAR, NOC, DIS, RON) for training and the remaining three pieces (LEG, HAT, THU) for testing. For training, eleven unique transpositions were added to the original aligned MIDI semigram - audio semigram pairs. The aligned pairs were followed by the same length of random pairs. For testing, four transpositions of the test pieces and a fresh set of random pairs were added to the aligned test set to assess its generalization capability. The classification accuracy of this test is given in the top row of Table 2 and is abbreviated TNDR. The table is divided into two groups of 3 columns with the first group showing the accuracy of the model run on the training data itself and the second group showing the accuracy for the test data. Within each group the column labeled 'LE' shows the results for the learned matrices, 'LU' for the aligned, averaged and un-rotated matrices and 'CH' for the matrices in standard chroma form (as shown in Figure 1 for **M**). A similar evaluation was performed by inter-changing the test and training sets in the evaluation mentioned above. The results are given in the second row of Table 2 with the abbreviation LHT. We also tried using Krumhansl templates (rotated to 12 transpositions) as the rows of the projection matrix, but this did not work as well as the standard chromagram. Although we omit those results here, as a summary, they performed about 3% less than the chroma mapping.

Another type of evaluation was carried out by training on each piece in Table 1 and then testing the alignment function using the remaining six pieces (hold out testing). The remaining seven rows of Table 2 show the results of this evaluation.

Table 2. Accuracy for group and hold out tests. LE: learned, LU: learned with averaging and un-rotating, CH: chroma.

	Training Data (%)			Test Data (%)		
Rec.	**LE**	**LU**	**CH**	**LE**	**LU**	**CH**
TNDR	89.6	88.7	85.4	90.5	89.6	87.4
LHT	91.3	90.5	87.3	88.9	88.5	85.3
TAR	90.0	86.8	83.4	88.2	87.8	86.2
NOC	87.6	87.0	83.5	88.9	89.1	86.5
DIS	90.8	90.6	87.4	89.0	88.8	85.3
LEG	91.9	88.0	88.0	88.1	84.3	85.6
HAT	91.4	90.0	83.9	88.9	88.6	86.1
THU	91.6	91.2	88.3	88.5	88.5	85.7
RON	91.5	89.4	85.1	88.8	88.9	86.2
CLA1	93.1	92.7	90.6	90.0	89.6	88.3
CLA2	92.0	91.4	89.2	91.1	90.8	89.1

Overall, for all the tests explained above, learned (LE) and learned averaged (LU) tests performed better than chroma (CH) with one exception in piece LEG where LU was lower than CH. This shows that averaging in this particular case did not work well and degraded the performance. In general, however, results suggest that an asymmetrical multi-octave chroma mapping is better than the commonly used octave independent symmetrical mapping as suggested by [7] and [14] and others.

CLA1 and CLA2 refer to the training data given in Table 1. Each of these were tested with the remaining 36 pieces (about 2000 seconds) from the RWC collection.

The accuracy numbers partially reflect the effects of some foreseeable factors in performing this evaluation: training alignments are not perfectly aligned at the frame level, the time variation of the spectral content in the audio is not reflected in the MIDI representation (timbre effects), audio contains percussion but the MIDI does not.

7. RESULTS

The most interesting result is that learned representations outperform chroma vectors on the task of discriminating aligned vs. unaligned audio frames. Perhaps this should not be too surprising since machine learning from large sets of training data often outperforms hand-tuned algorithms or features. Not only is there nothing "magic" about the chromagram, we see comparable performance from a neural network trained to answer the question "Does this MIDI frame align to that spectral frame?"

We also explored a particular model that maps the spectrum (and MIDI data) into 12-element vectors and computes similarity between these vectors using a standard distance function. Even though this certainly loses information, it allows us to study the representation, which can be viewed as a projection of the spectrum to a new space defined by a set of basis vectors. These vectors are particularly interesting. With chromagrams, the basis vectors are simply chroma (pitch classes), but with our learned projections, the basis vectors also show a remarkable similarity to pitch histograms obtained from music in a fixed key. Thus, even assuming the general form of the chromagram as a projection from the spectrum to a lower-dimensional space, we see room for improvement. This is evident from the fact that a learning system initialized with the chromagram projection will systematically adjust and improve to a new projection.

8. FUTURE WORK

We have limited our study to a 12-element vector representation for comparison to the chromagram. It would be interesting to study larger (and smaller) vectors. In particular, we wonder whether with additional dimensions, the learning algorithms would build different patterns for major, minor, and dominant tonalities, whether some patterns would reflect timbre or overtone characteristics, or whether other structures would be formed. The projection matrix formulation of the problem allows these potential structures to be observed. It should be noted that the nature of the **M** and **L** matrices is slightly different in that **M** incorporates the spectral structure of notes whereas **L** deals with notes alone.

The similarity of our learned patterns to the pitch histogram or Krumhansl template deserves further analysis.

Is this a coincidence? Are the learned patterns a reflection of the pitch distributions as well as average overtone strength distributions in our training data, or have pitch distributions in tonal music evolved to optimize the listener's ability to recognize music structures? Perhaps both forces are at work.

9. CONCLUSIONS

We have described methods for learning features that are useful for score alignment and other comparative and similarity based tasks such as identification of repeating sections, subsequence searching and template based chord recognition. The learned features out-perform the chromagram representation at least in the task of discriminating aligned from non-aligned frames of music. Unlike the chromagram representation, which is a simple projection based on pitch classes, the learned representation uses a projection that appears to be based on pitch distributions as well as the harmonic series common to most pitched musical instruments. In addition, the middle frequencies and pitches receive the most weight in the patterns, indicating that high and low frequencies are less useful for alignment. Another advantage of such an approach in MIR is that an alignment function can be directly learned from and used with almost native representations in both spectral and symbolic domains, thus bridging the gap between audio and symbolic music collections. We believe this work represents a significant advance by suggesting better features for music audio analysis, particularly for alignment and discovering music structure.

10. ACKNOWLEDGEMENTS

This material is based on work partly supported by the National Science Foundation under Grant Nos. 0534370 and 0855958. We would like to thank Meinard Müller for providing alignment data for the RWC dataset.

11. REFERENCES

[1] M. Bartsch and Wakefield, G. H. "Audio Thumbnailing of Popular Music Using Chroma-based Representations," *IEEE Transactions on Multimedia*, vol. 7, pp. 96-104, Feb. 2005.

[2] Dannenberg, R. and Hu, N. "Pattern Discovery Techniques for Music Audio," *Int. Symposium on Music Information Retrieval (ISMIR)*, Paris: IRCAM, pp. 63-70, 2002.

[3] Dannenberg, R. and Raphael, C. "Music Score Alignment and Computer Accompaniment," *Commun. ACM*, 49(8) (August 2006), pp. 38-43.

[4] Dixon, S. and Widmer, G. "Match: A Music Alignment Tool Chest," *Int. Symposium on Music Information Retrieval (ISMIR)*, London: Queen Mary, Univ. of London and Goldsmiths College, Univ. of London, 2005.

[5] Ewert, S., Müller, M., and Grosche, P. "High Resolution Audio Synchronization Using Chroma Onset Features," *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), Taipei, Taiwan, pp. 1869-1872, 2009.

[6] Fujishima, T. "Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music," *Proc. of the 1999 Int. Computer Music Conference (ICMC), pp. 464-467*, 1999.

[7] Gómez, E., "Tonal Description of Music and Audio Signals," Ph.D. dissertation, Barcelona: MTG, Universitat Pompeu Fabra, 2006.

[8] Goto, M. "A Chorus-Section Detection Method for Musical Audio Signals and Its Application to a Music Listening Station," *IEEE Trans. On Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1783-1794, Sep. 2006.

[9] Hu, N., Dannenberg, R., and Tzanetakis, G. "Polyphonic Audio Matching and Alignment for Music Retrieval," *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (WASPAA), New Paltz, USA, pp. 185-188, 2003.

[10] Krumhansl, C. *Cognitive Foundations of Musical Pitch.* New York: Oxford Univ. Press, 1990.

[11] Lu, L., Wang, M., and Zhang, H.-J. "Repeating Pattern Discovery and Structure Analysis from Acoustic Music Data." *Proc. of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval.* New York: Assoc. for Computing Machinery, pp. 275-282, 2004.

[12] Müller, M., Ewert, S., and Kreuzer, S. "Making Chroma Features more Robust to Timbre Changes," *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), Taipei, Taiwan, pp. 1869-1872, 2009.

[13] Müller, M., Kurth, F., and Clausen, M. "Audio Matching via Chroma-Based Statistical Features," *Int. Symposium on Music Information Retrieval (ISMIR)*, pp. 144-149, Oct. 2006.

[14] Pauws, S. "Musical Key Extraction from Audio," *Int. Symposium on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 2004.

[15] Serrà, J., Gómez, E., Herrera, P. and Serra, X. "Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification," *IEEE Transactions on Audio, Speech and Language Processing*, 16-6, pp. 1138-1152, August 2008.

[16] Serrà, J., Gómez, E., Herrera, P. and Serra, X. "Statistical Analysis of Chroma Features in Western Music Predicts Human Judgments of Tonality," *Journal of New Music Research*, 37-4, pp. 299-309, December 2008.

A MULTI-PASS ALGORITHM FOR ACCURATE AUDIO-TO-SCORE ALIGNMENT

Bernhard Niedermayer[1]
[1]Department for Computational Perception
Johannes Kepler University Linz, Austria
music@jku.at

Gerhard Widmer[1,2]
[2]Austrian Research Institute for Artificial Intelligence
Vienna, Austria
music@ofai.at

ABSTRACT

Most current audio-to-score alignment algorithms work on the level of score time frames; i.e., they cannot differentiate between several notes occurring at the same discrete time within the score. This level of accuracy is sufficient for a variety of applications. However, for those that deal with, for example, musical expression analysis such micro-timings might also be of interest. Therefore, we propose a method that estimates the onset times of individual notes in a post-processing step. Based on the initial alignment and a feature obtained by matrix factorization, those notes for which the confidence in the alignment is high are chosen as anchor notes. The remaining notes in between are revised, taking into account the additional information about these anchors and the temporal relations given by the score. We show that this method clearly outperforms a reference method that uses the same features but does not differentiate between anchor and non-anchor notes.

1. INTRODUCTION

There are several scenarios in which one wants to know the exact parameters (such as onset time, loudness, and duration) of each individual note within a musical performance. Most of these scenarios can occur in musicology, where data from different performances is used to extract general performance rules or to analyze individual artists' expressive styles. Other applications of such data are pedagogical systems or augmented audio editors and players. Unless the pieces under consideration are played on special computer-monitored instruments, audio recordings are the only sources of data describing expression within actual musical performances.

Our aim here was to extract timing (note onset) parameters from a great variety of classical piano music performances automatically. The most general method for this would be blind audio transcription, but current state of the art methods in this field are not reliable enough to base performance analysis on their results. However, since in classical music the piece and score corresponding to an audio recording can be assumed to be known, we can address the much simpler task of audio-to-score alignment.

Here, most state of the art algorithms start by extracting features (mainly chroma vectors) from each time frame of the audio signal as well as from the score representation. To obtain an optimal alignment between the two feature sequences, a distance measure between the feature vectors is then used as input either for Dynamic Time Warping (DTW) or for graphical models, such as Hidden Markov Models.

However, an inherent shortcoming of these methods is that – since only time frames are matched – they cannot distinguish individual onsets of notes that occur simultaneously in the score. This impedes the analysis of expressive elements, such as arpeggios or the asynchronies between a pianist's hands or within a chord. To resolve this shortcoming, the method proposed here extracts an onset time estimation for each individual note. In order to do so, notes for which the timing can be extracted with a relatively high confidence level are marked as "anchor notes". In a second pass, the system then tries to refine the timings of the remaining notes by combining the expected position between the anchors with spectral information.

Section 2 gives an overview of related work. We explain the extraction of anchor notes in Section 3, and describe the refinement method applied to the notes between such anchors in Section 4. Section 5 presents our experimental results and Section 6 provides the conclusion and an outlook on future work.

2. RELATED WORK

Online versus offline differentiation aside, state-of-the-art audio-to-score alignment algorithms can be clustered into two main approaches. One is based on statistical, graphical models built from the score, such as those in [1, 2, 11]. The other one uses Dynamic Time Warping (DTW) in order to align sequences of features calculated from both the audio and the score representation [5, 8].

The latter method normally uses chroma vectors as feature, resulting in relatively robust global alignments. However, their temporal accuracy cannot compete with other features which are used in onset detection. In [3], so-called DLNCO-features were introduced, which in essence combine chroma vectors and (pitch-wise) spectral flux. More-

over, a very high temporal feature resolution is used. This is not trivial, since DTW is of complexity $O(n^2)$, and computational costs constrain the number of frames that are aligned. A multi-scale approach introduced in [8], however,allows the temporal resolution to be increased iteratively.

Another combination of chroma-based alignment with onset detection was presented in [6]. Here, an initial alignment is used in order to train an onset detector. Results from the onset detection are then used iteratively to refine the alignment and to train a better onset detector on this more accurate data. This allows the use of supervised machine learning techniques without the need for external training data.

In [7] and [9], results of a DTW-based alignment are refined in a second pass. Both approaches place a search window around the tentative onset time of a note. This window is then scanned for a compatible note onset. While the first method relies on an STFT spectrogram, the latter uses a dictionary-based decomposition of the spectrum in order to differentiate between spectral energies induced by individual notes.

Like the method proposed here, these two approaches can distinguish between the onsets of notes that occur at the same discrete time within the score. This is different from most other systems, since it is inherent to both the DTW algorithms and to the graphical models used in [2, 11] that they work on features representing discrete time steps and that they cannot differentiate between two events occurring within the same time step.

3. ANCHOR NOTE ESTIMATION

At first the anchor notes are extracted using a two-pass procedure as proposed in [9]. In the first step, a state-of-the-art audio-to-score alignment based on chroma vectors and Dynamic Time Warping (DTW) is performed. Then, a dictionary of tone models is used in order to extract each note's activation function. Notes for which a significant rise in activation energy can be found near the corresponding estimated onset are selected to serve as anchors.

3.1 Chroma Features and DTW

In [5], chroma vectors were found to perform best amongst several features in the context of audio matching and alignment. They consist of 12 elements per time frame corresponding to the pitch classes (C, C#, D,...). The values are calculated from an audio recording by mapping the frequency bins of a short-time Fourier transform to pitch class indices i using

$$i = \left[\text{round}\left(12\log_2\left(\frac{f_k}{440}\right)\right) + 9\right] \mod 12 \quad (1)$$

where f_k represents the center frequency of the k^{th} bin. The term inside the brackets gives the number of the pitch ($A4 \mathrel{\widehat{=}} 0$) that is nearest to f_k, and applying the module gives the pitch class. The summand 9 shifts the indices such that $i = 0$ corresponds to the pitch class C. The actual values of the chroma vectors are then obtained by summing up the energies of all bins mapped to a certain pitch class.

There are two approaches to calculating chroma features from score representations [5]. One of them is to render the score using a software synthesizer to reduce the problem to the one described above. The other method calculates the chroma vector directly from the score. Here, the mapping becomes trivial, since the pitches are already given. However, one must make assumptions about pitch energies and either use constant energies or a decay model. In our experiments, we compared both methods – the first one using the free software synthesizer *timidity* [1] , and the second one using constant midi note energies. Preliminary experiments showed that the resulting alignments did not differ significantly between the two approaches. Therefore, we decided to use the second one, since it is much cheaper in terms of computational costs.

Given two sequences of feature vectors, a cost function must be defined which accounts for the error made when aligning one specific frame within the first sequence to another specific frame within the other sequence. Our experiments showed that the Euclidean distance yields better results than other possible measures, such as the cosine distance.

Based on the cost function, a similarity matrix SM can be constructed. The rows of this matrix represent the time frames of the audio recording, whereas the columns represent the time frames of the score. Hence, the value of each cell SM_{ij} contains the cost of aligning the i^{th} frame of the audio signal to the j^{th} frame of the score. Any continuous, monotonic path through this matrix that begins and ends at the two end-points of the main diagonal represents a valid alignment between the two sequences. The objective is to minimize the global alignment cost, i.e., the sum of all local costs SM_{ij} along the path through the similarity matrix.

Using DTW, the optimal alignment is calculated in two steps. The forward step starts at $[0, 0]$ and the corresponding cost $SM_{0,0}$. Then, all other optimal partial alignments ending with the i^{th} frame of the audio recording aligned to the j^{th} frame of the score are obtained by recursively building a second matrix $Accu$ according to

$$Accu(i,j) = \min \begin{cases} Accu(i-1, j-1) + SM_{ij} \\ Accu(i-1, j) + SM_{ij} \\ Accu(i, j-1) + SM_{ij} \end{cases} \quad (2)$$

In the forward step, another matrix stores which of these three options has been used in order to advance to the next cell. As soon as the end point $[N-1, M-1]$ has been reached, this information is utilized to reconstruct the path, i.e., the optimal alignment. A more detailed description of the DTW algorithm is given in [10].

[1] http://timidity.sourceforge.net

3.2 NMF and Anchor Selection

The global alignment resulting from the DTW is robust. However, local inaccuracies are inherent to the algorithm. Therefore, an additional feature based on non-negative matrix factorization (NMF) is used to reestimate the onset of each individual note.

NMF is the decomposition of one matrix V of size $m \times n$ into two output matrices W and H of sizes $m \times r$ and $r \times n$, respectively, such that all elements of W and H are non-negative and

$$V \approx WH \tag{3}$$

Applied to audio processing, such a decomposition of a spectrogram results in a dictionary W of r weighted frequency groups and the corresponding activation energies H of these frequency groups over time.

Here we use a modification, as described in [12] and [9], in which W is set to a pretrained set of tone models. These models are computed from audio recordings of single tones played on a piano by, in essence, taking each bin's weighted average energy over the time span where the tone is sustained. The weight of a frame is the inverse of the amplitude envelope to compensate for different loudnesses.

Assuming a fixed W, only H is estimated. Since the pitch described by an individual tone model is known, the i^{th} column of H is a feature representing the activation energy of each pitch in time frame i.

To improve the onset time estimates, a search window of length l is centered around the onset time t_{dtw} obtained by the DTW algorithm. Within this search window a factorization is performed using a dictionary W consisting of tone models of all those pitches that are expected to be played within that time span and an additional white noise component in which the energies are spread uniformly over all frequency bins. A new onset time candidate t_{nmf} is then obtained by choosing the time frame with the largest increase in energy of the pitch under consideration. In contrast to t_{dtw}, t_{nmf} can deviate from other notes with the same score time.

When thinking of repeated notes or of fast passages in which a certain pitch is played several times within the search window, it becomes obvious that this method is too simple to yield meaningful results. However, estimating the onsets of repeated notes is a relatively hard problem in itself. Spectral energy of a sustained note weakens the indicators for the onset of a new note if they have the same pitch. Under these circumstances, algorithms are likely to get mislead by onsets of other notes with overlapping harmonics. This fact makes such notes ineligible to be anchor notes, as a high confidence in the exact estimation of the onset time is essential. Thus, all notes which are played twice or even more often within the time span of the search window, as determined from the score, are discarded from the anchor candidates.

Likewise, all notes are dropped from the list of anchor candidates, for which the initial onset estimate t_{dtw} and the estimate given by the factorization-based feature t_{nmf} differ by more than a certain time span which could have plausibly been caused by an arpeggio or a simple asynchrony. This is justified because such a conflict decreases the confidence in the onset estimation. Moreover, there is no safe way to give either t_{dtw} or t_{nmf} a preference over the other. On the one hand, t_{dtw} is supposed to be more robust, since much more context information is incorporated. On the other hand, t_{nmf} is not bound by the constraints inherent to the DTW algorithm, and therefore able to yield more accurate results [9].

In summary, the two times t_{dtw} and t_{nmf} are calculated by the DTW algorithm and finding the maximum slope within the factorization-based pitch activation. A note is then selected as an anchor if the following two criteria are met:

1. $|t_{dtw} - t_{nmf}| <$ threshold
2. there are no other notes of the same pitch within $t_{dtw} \pm l/2$

In our experiments, we used an STFT with window and hop sizes of 4095 and 1024 frames, respectively, to compute the chroma features from the audio signal. In order to extract chroma vectors from the score, window and hop sizes had been scaled such that the overall number of frames and the overlap ratio remained unchanged relative to the audio representation. Since the DTW misplaces only a negligible fraction of all notes by more than a second, we chose 2.0 seconds for the size l of the search window. Within this search window the hop size was decreased to 256 frames. The maximum difference $|t_{dtw} - t_{nmf}|$ allowed between the two onset estimates was set to 20 frames, i.e., a little more than a tenth of a second.

An evaluation of the extraction of anchor notes is presented in Section 5.

4. NOTE REFINEMENT

After extracting the anchor notes, the remaining notes must be revised. For each of them (with the exception of notes played before the first or after the last anchor notes) the span of time during which it can be played is clearly constrained by the preceding and the successive anchor.

4.1 Beta distribution

In addition to a new search window, bounded by the nearest anchors, rhythmic information in the score can be used to make even more detailed predictions on where to look for an onset. Therefore, the numbers or fractions of beats between the anchor notes and the note under consideration are extracted and their relation is transferred onto the timescale of the audio recording. To account for inexactnesses of the anchor extraction and expressive tempo changes, the "expectation strength" of the onset occurring at time t is modeled by a beta distribution [2] . The beta distribution is defined continuously on the interval $[0, 1]$ and

[2] The beta distribution was chosen for pragmatic reasons (the flexibility of its shape and its restriction to a fixed interval) rather than for precise probability-theoretic reasons.

zero outside this range. Depending on the values of its parameters α and β, the density function can take several forms, for example, that of a uniform distribution, it can be strictly increasing or decreasing, U-shaped, or – as in our case – it is unimodal ($\alpha > 1$ and $\beta > 1$). Its density function is defined as

$$f(x)_{\alpha,\beta} = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1} \quad (4)$$

where B is the beta function

$$B(\alpha,\beta) = 2\int_0^{\pi/2} \cos^{2\alpha-1}\theta \sin^{2\beta-1}\theta \, \mathrm{d}\theta \quad (5)$$

Mode $\hat{x}$ and variance σ^2 of the distribution are therefore given by

$$\hat{x} = \frac{\alpha-1}{\alpha+\beta-2} \quad (6)$$

$$\sigma^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \quad (7)$$

In our application, we set the parameters α and β by fixing a mode $\hat{x}$ and a variance σ^2. The former is assumed to be at the onset time we expect according to score and anchor notes. Since the density function is only defined on $[0, 1]$, we use a linear projection to convert between the domain of the beta distribution and the score time.

The variance is chosen such that it allows for expressive variations and inexactnesses of the anchor extraction, but prevents notes from being placed at rhythmically unreasonable timings. Experiments showed that the value $\min(\hat{x}, 1-\hat{x})/20$ results in plausible expectation strengths.

Two such functions are depicted in Figure 1. The upper plot shows the onset likelihood for the onset time of the third note, assuming that the first and fifth note are anchors. The time span between the anchor comprises three beats. Since the note should be played after the first out of these three beat-to-beat intervals, the function is clearly skewed. This is desirable because a musician's freedom of expressive timing is greater when the score calls for longer inter-onset intervals. The second function is the likelihood of the fourth note's onset time given notes number one and six as anchors. The function is now symmetric, since the onset time given by the score is exactly half the time span (two out of four beat-to-beat intervals).

In order to transfer these expectation strength functions from the score into the audio domain, another linear projection is applied.

4.2 Onset estimation

To extract revised onset estimates for non-anchor notes, we calculate the constant Q spectrogram over the time span in which the onset likelihood as described above is greater than zero. The parameters of the constant Q spectrogram are chosen such that each energy bin corresponds to a specific pitch. The hop size is set to 256 frames, resulting in a very high overlap ratio at the lower bins.

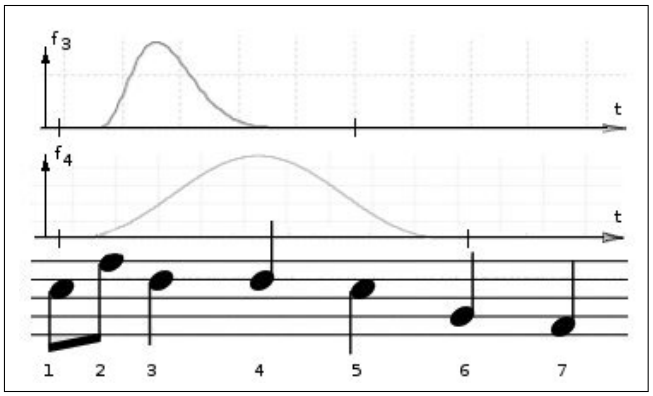


Figure 1. Onset expectation strength for the 3^{rd} and 4^{th} note.

For the purpose of onset detection, energy changes are calculated and half-wave rectified. In order to incorporate the score information, the result is then weighted by the expectation strength. The final onset is set to the time corresponding to the maximum of this detection function.

5. EXPERIMENTAL RESULTS

5.1 Evaluation Method

Since this work was done in the context of musical performance and style analysis, we used classical (polyphonic) piano music to evaluate our system. The test data consisted of the first movements of 11 Mozart sonatas played by a professional pianist. The overall performance time amounted to more than one hour, comprising more than 30.000 notes. The instrument used for the performance was a computer-controlled Bösendorfer SE290 grand piano, which enables exact logging of all events such as keys being hit or released and changes in pedal pressure.

The evaluation was done using mechanical scores represented in MIDI format and the real audio recording from the performances as input data. The data recorded by the Bösendorfer SE290 served as ground truth. The main evaluation criterion was the absolute timing displacement between aligned notes and the ground truth.

On the one hand, robustness and a high overall accuracy are important issues. On the other hand, our work is directed towards providing methods for semi-automatic audio annotation. One objective of such a system must be to minimize human input. In post-processing, the user must correct the onset time as soon as there is a noticeable error. Therefore, we investigated not only the median and percentile errors, but also how many of the notes were detected well enough for a human to accept it.

In [4], listening tests showed that the human hearing system does not detect timing variations of up to 10 ms in sequences of short notes, and even greater displacements in sequences of very long notes. Therefore, our evaluation criteria were the proportions of notes aligned with a displacement of less than 10 ms and 50 ms respectively. The 50 ms tolerance was included because it is a common margin in onset detection.

piece	duration	# notes	# anchors	$50\% < x$[ms]		$75\% < x$[ms]		$95\% < x$[ms]		max [ms]	
				non.a.	anch.	non.a.	anch.	non.a.	anch.	non.a.	anch.
K.279-1	4:55	2803	1136	15.2	5.5	29	11	138	37	879	494
K.280-1	4:48	2491	1257	23.2	5.4	45	11	165	46	687	664
K.281-1	4:29	2648	1235	23.7	6.1	48	12	176	48	993	442
K.282-1	7:35	1907	705	23.8	6.9	60	13	439	72	4805	3008
K.283-1	5:22	3304	1130	16.2	7.9	28	13	75	34	673	467
K.284-1	5:17	3700	1223	15.2	6.1	31	14	120	71	1000	502
K.330-1	6:14	3160	1176	16.3	5.6	30	10	179	35	960	835
K.332-1	6:02	3470	1017	23.2	11.8	42	19	171	82	857	632
K.333-1	6:44	3774	1471	17.8	7.5	31	13	132	38	941	404
K.457-1	6:15	2993	1086	22.0	8.9	42	16	317	62	1773	1787
K.475-1	4:58	1284	483	38.4	16.3	98	24	304	115	4471	2663

Table 1. Comparison between accuracy (median, 75^{th} percentile, 95^{th} percentile, and maximum) of the anchor notes (anch.) and the non-anchor notes (n.a.)

piece	# non-anchors	$50\% < x$[ms]	$75\% < x$[ms]	$95\% < x$[ms]	max [ms]
K.279-1	1667	9.1	28	127	879
K.280-1	1234	9.2	24	147	706
K.281-1	1413	11.2	31	187	1035
K.282-1	1202	15.9	42	432	4822
K.283-1	2174	12.0	21	92	464
K.284-1	2477	9.0	26	125	1004
K.330-1	1983	9.6	21	134	835
K.332-1	2453	18.0	30	175	781
K.333-1	2303	12.1	22	93	1000
K.457-1	1907	16.5	37	246	1790
K.475-1	812	24.1	49	398	4377

Table 2. Accuracy of non-anchor notes after the refinement step (median, 75^{th} percentile, 95^{th} percentile, and maximum)

5.2 Evaluation Results

Table 1 presents the results of the anchor detection step. About a third of the overall notes were chosen as anchors. Although this seems to be a very large fraction, it is justified by the high accuracy of the selected notes. For half of the pieces, the 95th percentile still met the 50 ms criterion used for the evaluation of onset detection algorithms.

However, for each piece a small number of outliers were picked as well. Some of them are due to our trade-off between a small search window at the NMF calculation and computational costs. Notes for which the initial alignment deviates from the real onset by more than a second are post-processed using a time frame that does not even contain the correct onset.

Table 2 shows that, in comparison to Table 1, a majority of non-anchor notes were improved by the refinement step. Both the median deviation and the 75th percentile improved for all the pieces. Only the accuracy of the outliers decreased further in some cases. This might be due to poor anchor notes, which mislead onset detection.

The overall result as given by Table 3 shows the potential of the proposed method. It clearly outperformed the reference algorithm from [9] in which the initial alignment and the factorization-based post-processing were done in a similar way but without using score information to refine critical notes. Especially the proportion of note onsets identified with a deviation of less than 10 ms – i.e., the threshold of human perception, according to [4] – was increased significantly from 40.0% to 49.8%. This is important for the construction of data acquisition tools which are able to extract descriptions of musical expression from audio recordings semi-automatically.

6. CONCLUSION AND FUTURE WORK

We have proposed a multi-pass method for the accurate alignment of musical scores to corresponding audio recordings. The main contribution is the introduction of an expectation strength function modeling the expected onset time of a note between two anchors. Although results are encouraging, there are specific circumstances where the algorithm fails, i.e., temporal displacement of notes is large.

One class of such errors are poor alignments at a piece's ending. There, two disadvantageous factors coincide. On the one hand, there is no additional subsequent note which could serve as hint for the alignment or as anchor in the post-processing. On the other hand, a high degree of polyphony in combination with long and soft notes is to be expected at endings. Such passages are inherently difficult to handle from a signal processing point of view.

An interesting example of such an error can be found in the sonata K.282, in which one note was even wrongly picked as an anchor although it was out of place by more

piece	# notes	$50\% < x$[ms]		$75\% < x$[ms]		$95\% < x$[ms]		$x < 10\,ms$		$x < 50\,ms$	
		ref.	new	ref.	new	ref.	new	ref.	new	ref.	new
K.279-1	2803	12	7.2	27	18	101	103	43.2%	61.7%	88.4%	90.2%
K.280-1	2491	14	7.1	34	16	127	93	42.5%	63.1%	85.0%	90.8%
K.281-1	2648	15	8.5	36	19	112	114	38.5%	56.8%	83.4%	89.9%
K.282-1	1907	15	11.8	44	27	380	378	39.2%	43.5%	76.8%	83.2%
K.283-1	3304	12	10.2	26	18	65	70	44.2%	49.1%	92.2%	92.4%
K.284-1	3700	13	8.0	29	21	98	110	41.7%	58.2%	87.2%	87.7%
K.330-1	3160	11	7.6	24	15	124	103	46.7%	61.0%	89.7%	91.2%
K.332-1	3470	18	16.0	37	27	147	148	32.5%	29.7%	82.7%	87.9%
K.333-1	3774	13	9.9	20	18	80	68	42.2%	50.5%	90.1%	92.8%
K.457-1	2993	15	13.4	35	26	257	183	35.9%	38.2%	83.2%	84.8%
K.475-1	1284	24	20.1	75	37	393	376	23.6%	22.5%	66.8%	78.6%
all	31534	14	10.1	32	21	137	121	40.0%	49.8%	85.6%	88.9%

Table 3. Overall accuracy of the proposed anchor-based method (new) compared to the reference method as described in [9] (ref.)

than three seconds. The explanation is, that the last two chords of this piece differ by only one single note (b♭-a♭-d-f and e♭-a♭-d-f, respectively). The algorithm was not able to distinguish the two chords. As a consequence, the notes of the last chord were aligned to the onset of the preceding chord as well. The resulting temporal displacement of about three seconds is slightly shorter than the duration of the first of these chords.

This clearly leads further work towards the issues of improved mechanisms for anchor detection and the handling of inherently "difficult" passages, such as the endings. An approach that could benefit both fields is the introduction of a more sophisticated local confidence or fitness measure for arbitrary sections of an alignment.

Another aspect which has not been considered yet is the detection of deviations from the score, such as when the pianists adds ornamentations or playing errors occur.

7. ACKNOWLEDGEMENTS

This research is supported by the Austrian Federal Ministry for Transport, Innovation and Technology, and the Austrian Science Fund (FWF) under project numbers TRP 109-N23, P19349-N15, and Z159.

8. REFERENCES

[1] A. Cont: "Realtime Audio to Score Alignment for Polyphonic Music Instruments Using Sparse Non-negative Constraints and Hierarchical HMMs", *Proceedings of the IEEE International Conference in Acoustics and Speech Signal Processing (ICASSP)*, Toulouse, 2006.

[2] A. Cont: "A coupled duration-focused architecture for realtime music to score alignment", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 99(1), 2009.

[3] S. Ewert and M. Müller: "Refinement Strategies for Music Synchronization", *Proceedings of the 5th International Symposium on Computer Music Modeling and Retrieval (CMMR 2008)*, Copenhagen, 2008.

[4] A. Friberg and J. Sundberg: "Perception of just noticeable time displacement of a tone presented in a metrical sequence at different tempos", *Proceedings of the Stockholm Music Acoustics Conference*, pp. 39–43, Stockholm, 1993.

[5] N. Hu, R. B. Dannenberg, and G. Tzanetakis: "Polyphonic Audio Matching and Alignment for Music Retrieval", *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New York, 2003.

[6] N. Hu and R. B. Dannenberg: "Bootstrap Learning for Accurate Onset Detection", *Machine Learning*, Vol. 65, No. 2, pp. 457–471, 2006.

[7] Y. Meron and K. Hirose: "Automatic alignment of a musical score to performed music", *Acoustical Science and Technology*, Vol. 22, No. 3, pp. 189–198, 2001.

[8] M. Müller, H. Mattes, and F. Kurth: "An Efficient Multiscale Approach to Audio Synchronization", *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, 2006.

[9] B. Niedermayer: "Improving Accuracy of PolyphonicMusic-to-Score Alignment", *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, Kobe, 2009.

[10] Rabiner, L. R. and Juang, B.-H. "Fundamentals of speech recognition". Prentice Hall, Englewood Cliffs, NJ, 1993.

[11] C. Raphael: "Aligning Music Audio with Symbolic Scores Using a Hybrid Graphical Model", *Machine Learning*, Vol. 65 (2-3), pp. 389–409, 2006.

[12] F. Sha and L. Saul: "Real-time pitch determination of one or more voices by nonnegative matrix factorization", *Advances in Neural Information Processing Systems 17*, K. Saul, Y. Weiss, and L. Bottou (eds.), MIT Press, Cambridge, MA, 2005.

ACCURATE REAL-TIME WINDOWED TIME WARPING

Robert Macrae
Centre for Digital Music
Queen Mary University of London
robert.macrae@elec.qmul.ac.uk

Simon Dixon
Centre for Digital Music
Queen Mary University of London
simon.dixon@elec.qmul.ac.uk

ABSTRACT

Dynamic Time Warping (DTW) is used to find alignments between two related streams of information and can be used to link data, recognise patterns or find similarities. Typically, DTW requires the complete series of both input streams in advance and has quadratic time and space requirements. As such DTW is unsuitable for real-time applications and is inefficient for aligning long sequences. We present Windowed Time Warping (WTW), a variation on DTW that, by dividing the path into a series of DTW windows and making use of path cost estimation, achieves alignments with an accuracy and efficiency superior to other leading modifications and with the capability of synchronising in real-time. We demonstrate this method in a score following application. Evaluation of the WTW score following system found 97.0% of audio note onsets were correctly aligned within 2000 ms of the known time. Results also show reductions in execution times over state-of-the-art efficient DTW modifications.

1. INTRODUCTION

Dynamic Time Warping (DTW) is used to synchronise two related streams of information by finding the lowest cost path linking feature sequences of the two streams together. It has been used for audio synchronisation [3], cover song identification [13], automatic transcription [14], speech processing [10], gesture recognition [7], face recognition [1], lip-reading [8], data-mining [5], medicine [15], analytical chemistry [2], and genetics [6], as well as other areas. In DTW, dynamic programming is used to find the minimal cost path through an accumulated cost matrix of the elements of two sequences. As each element from one sequence has to be compared with each element from the other, the calculation of the matrix scales inefficiently with longer sequences. This, combined with the requirement of knowing the start and end points of the sequences, makes DTW unsuitable for real-time synchronisation. A real-time variant would make DTW viable at larger scales and capable of driving applications such as score following, automatic accompaniment and live gesture recognition.

Local constraints such as those by Sakoe and Chiba [10] improve the efficiency of DTW to linear time and space complexity by limiting the potential area of the accumulated cost matrix to within a set distance of the diagonal. However, not all alignments necessarily fit within these bounds. Salvador and Chan proposed, in FastDTW [11], a multi-resolution DTW where increasingly higher resolution DTW paths are bounded by a band around the previous lower resolution path, leading to large reductions in the execution time. On-Line Time Warping by Dixon [3] made real-time synchronisation with DTW possible by calculating the accumulated cost in a forward manner and bounding the path by a forward path estimation.

While the efficiency of DTW has been addressed in FastDTW [11] and the real-time aspect has been made possible with On-Line Time Warping [3], WTW contributes to synchronisation by offering steps to further improve the efficiency whilst working in a progressive (real-time applicable) manner and preserving the accuracy of standard DTW. This method consists of breaking down the alignment into a series of separate bounded sub-paths and using a cost estimation to limit the area of the accumulated cost matrix calculated to small regions covering the alignment.

In Section 2 we explain conventional DTW before describing how WTW works in Section 3. In Section 4 we evaluate the accuracy and efficiency of WTW in a score following application. Finally, in Section 5, we draw conclusions from this work and discuss future improvements.

2. DYNAMIC TIME WARPING

DTW requires two sets of features to be extracted from the two input pieces being aligned and a function for calculating the similarity between any two frames of these feature sets. One such measurement of the similarity is the inner product. As the inner product returns a high value for similar frames, we subtract the inner product from one so that the optimal path cost is the path with the minimal cost. Equation 1 shows how to calculate this similarity measurement between frames A_m and B_n from feature sequences $A = (a_1, a_2, ..., a_M)$ and $B = (b_1, b_2, ..., b_N)$ respectively:

$$d_{A,B}(m,n) = 1 - \frac{< a_m, b_n >}{\|a_m\|\|b_n\|} \quad (1)$$

Dynamic programming is used to find the optimum path, $P = (p_1, p_2, ..., p_W)$, through the similarity matrix $\mathcal{C}(m,n)$

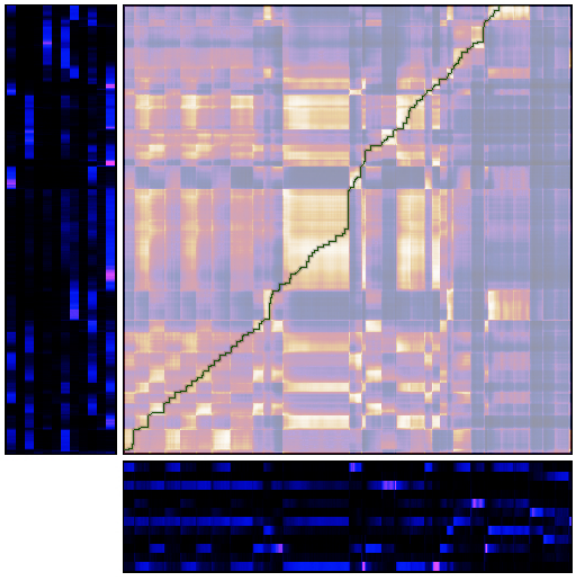


Figure 1. Dynamic Time Warping aligning audio with a musical score. The audio is divided into chroma frames (bottom) which are then compared against the score's chroma frames (left). The similarity matrix (centre) shows a path where the sequences have the lowest cost (highest similarity). Any point on this path indicates where in the score the corresponding audio relates to.

with $m \in [1:M]$ and $n \in [1:N]$ where each $p_k = (m_k, n_k)$ indicates that frames a_{m_k} and b_{n_k} are part of the aligned path at position k. An example of this similarity matrix, including the features used and the lowest cost path, can be seen in Figure 1. The final path is guaranteed to have the minimal overall cost $D(P) = \sum_{k=1}^{W} d_{A,B}(m_k, n_k)$, within the limits of the features used, whilst satisfying the following conditions:

Bounds:	$p_1 = (1,1)$
	$p_W = (M,N)$
Monotonicity:	$m_{k+1} \geq m_k$ for all $k \in [1, W-1]$
	$n_{k+1} \geq n_k$ for all $k \in [1, W-1]$
Continuity:	$m_{k+1} \leq m_k + 1$ for all $k \in [1, W-1]$
	$n_{k+1} \leq n_k + 1$ for all $k \in [1, W-1]$

3. WINDOWED TIME WARPING

WTW consists of calculating small sub-alignments and combining these to form an overall path. Subsequent sub-paths are started from points along the previous sub-paths. Real-time path positions can then be extrapolated from these sub-paths. The end points of these sub-alignments are either undirected, by assuming they lie on the diagonal, or directed, by using a forward path estimate. As such WTW can be seen as a two-pass system similar to FastDTW and OTW. The sub-alignments make use of an optimisation that avoids calculating points with costs that are over the cost estimate (provided by the initial direction path), referred to as the A-Star Cost Matrix. WTW also requires the use of Features, Window Dimensions, and Local Constraints that all affect how the alignments are made. The overall process is outlined in Algorithm 1. In order to demonstrate WTW we implemented a score following application using this method to synchronise audio and musical scores.

```
Input: Feature Sequence A and Feature Sequence B
Output: Alignment Path
Path = new Path.starting(1,1);
while Path.length < min (A.length,B.length) do
    Start = Path.end;
    End = Start;
    while (End - Start).length < Window_Size do
        End =
        argmin(Inner_Product(End.next_points));
    end
    Cost_Estimate = End.cost;
    A-Star_Matrix =
    A_Star_Fill_Rect(Start,End,Cost_Estimate);
    Path.add(A_Star_Matrix.getPath(1,Hop_Size));
end
return Path;
```

Algorithm 1: The Windowed Time Warping algorithm.

3.1 Features

The feature vector describes how the sequence data is represented and segmented. The sequence is divided up into feature frames in order to differentiate the changes in the sequence over time. The frame size and spacing are referred to as the window size and hop size respectively. The implementation of WTW for score following requires a musically based feature vector. In this case, we use chroma features, a 12 dimensional vector corresponding to the unique pitch classes in standard Western music. The intensities of the chroma vectors can be seen as a representation of the harmonic and melodic content of the music. In our implementation we use a window size of 200ms and a hop size of 50ms.

3.2 Window Dimensions

Similar to how the sequence data is segmented, the windows of standard DTW in WTW have a window size and hop size to describe their size and spacing respectively. A larger window size and/or smaller hop size will increase the accuracy of the alignment, as more of the cost matrix is calculated, however will this will be less efficient. Examples of different window and hop sizes can be seen in Figure 2 and a comparison of Window and Hop sizes is made in Section 4.

3.3 Local Constraints

We refer to two types of local constraints in Dynamic Programming. The first, henceforth known as the *cost constraint*, indicates the possible predecessors of a point p_k on a path. The predecessor p_{k-1} with lowest path cost $D(p_{k-1})$ is chosen when calculating the accumulated cost

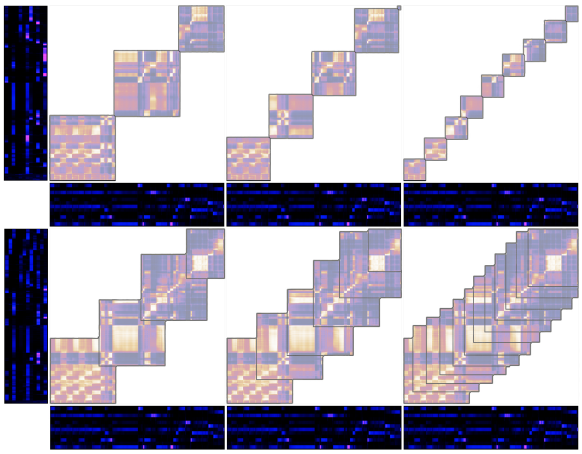


Figure 2. The regions of the similarity matrix computed for various values of the window size (top row) and hop size (bottom row).

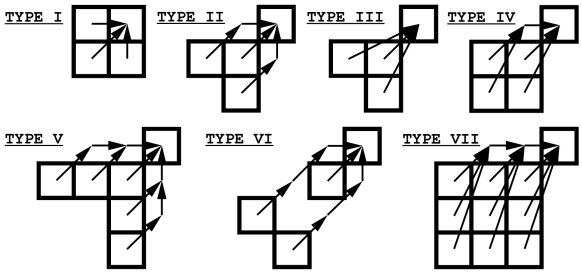


Figure 3. Some example local constraints as defined by Rabiner and Juang [9].

matrix. The second, referred to as the *movement constraint*, indicates the possible successors of a point p_k. Standard DTW doesn't make use of a *movement constraint* as all the frames in the cost matrix are calculated. Examples of local constraints by Rabiner and Juang [9] are show in Figure 3. These constraints define the characteristics of the dynamic programming. For example, Type I allows for horizontal and vertical movement which corresponds to a single frame of one sequence being linked to multiple frames of the other. All the other Types allow high cost frames to be skipped and Type III and II show how the paths can skip these frames directly or add in the single steps, respectively. The two path finding algorithms, described next, make use of the Type I and a modified version of the Type VII (where the steps are taken directly as in Type III) local constraints.

3.4 Window Guidance

The sequential windows that make up the alignment of WTW can be either directed or undirected. Whilst it can help to direct the end point of the windows of DTW (particularly for alignments between disproportional sequences where the expected path angle will be far from 45°), the sub-paths calculated within these windows can make up for an error in the estimation. A low hop size should ensure the point taken from the sub-path as the starting point for the next window is likely to be on the correct path.

For the windows to be directed, a forward estimation is required. The Forward Greedy Path (FGP) is an algorithm which makes steps through the similarity matrix based on whichever subsequent step has the highest similarity (minimal cost) using a *movement constraint* to decide which frames are considered. In this manner the path can work in an efficient forward progressive manner, however, will be more likely to be thrown off the correct path by any periods of dissimilarity within the alignment. The first FGP path $F = (f_1, f_2, ..., f_W)$ where $f_k = (m_k, n_k)$ starts from position $f_1 = (m_1, n_1)$ and from then on each subsequent frame is determined by whichever of the available frames, as determined by the local constraint, has the lowest cost. Therefore the total cost $D(m, n)$ to any point (m, n) on the FGP path F is $D(f_k) = \sum_{l=1}^{k} d(f_l)$ and any point is dependent on the previous point: $f_{k+1} = argmin(d(i, j))$ where the range of possible values for i and j are determined by f_k and the local constraints.

The FGP path only needs to calculate similarities between frames considered within the local constraints and so at this stage a vast majority of the similarity matrix does not need to be calculated. When the FGP reaches f_W, the window size, the final point $f_W = (m_W, n_W)$ is selected as the end point for the accumulated cost-matrix.

Note that some combinations of constraints that skip points (*i.e.* where i or j are greater than 1) will require that jumps in the FGP are filled in order to compute a complete cost estimate, like in the Type V local constraint, so that the cost estimation of the FGP is complete. A comparison of guidance measures is made in Section 4.

3.5 A-Star Cost Matrix

The windowed area selected is calculated as an accumulated cost matrix between the beginning and end points of the FGP *i.e.* $\mathcal{C}(m, n)$ of $m \in [m_{f_1} : m_{f_L}]$ and $n \in [n_{f_1} : n_{f_L}]$. This accumulated cost matrix can be calculated in either a forward or reverse manner, linking the start to the end point or vice versa. This uses the standard Type I *cost constraint* to determine a frame's accumulated cost as shown by Equation 2:

$$D(m, n) = d(m, n) + \min \begin{Bmatrix} D(m-1, n-1) \\ D(m-1, n) \\ D(m, n-1) \end{Bmatrix} \quad (2)$$

The sub-path $S = (s_1, s_2, ..., s_V)$ is given by the accumulated *cost constraints* by following the cost progression from the beginning point in this window until the hop size is reached. When the sub-path reaches s_V, the final point $f_V = (m_v, n_v)$ is then taken as the starting point for the next window and so on until the end of either sequence is reached. The sub-paths are concatenated to construct the global WTW path. This process can also be seen in Figure 4.

Either of the undirected and directed window end point estimations provide an estimate cost $D(F)$ for each sub-path. This estimate can be used to disregard any points within the accumulated cost matrix that are above this cost

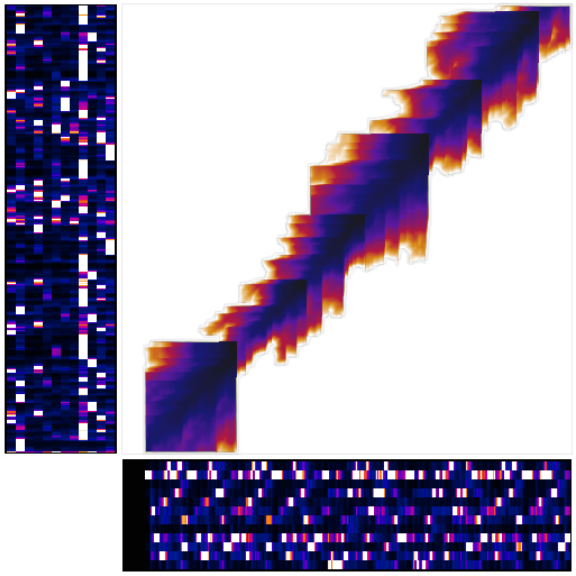


Figure 4. The complete Windowed Time Warping path.

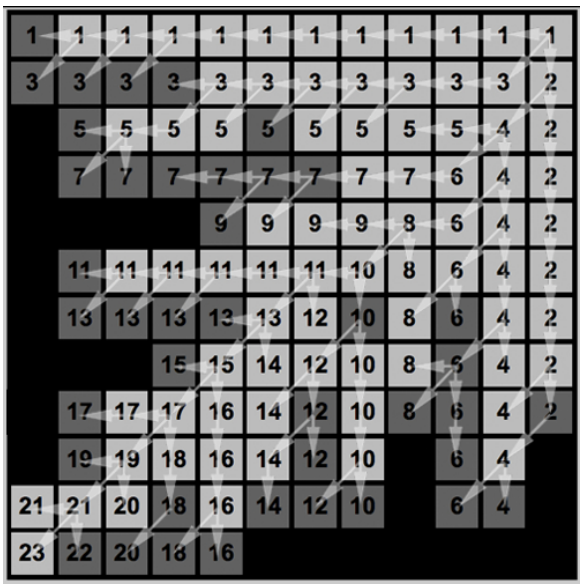


Figure 5. The calculation of the accumulated cost matrix. The numbering shows the order in which rows and columns are calculated and the progression of the path finding algorithm is shown by arrows. Dark squares represent a total cost greater than the estimated path cost whilst black squares indicate points in the accumulated cost matrix that do not need to be calculated.

as it is known there is a potential sub-path that is cheaper. The calculation of the similarity for most of these inefficient points can be avoided by calculating the accumulated cost matrix in rows and columns from the end point f_L to the start f_1. When each possible preceding point for the next step of the current row/column has a total cost above the estimated cost *i.e.* $\min(D(m-1,n-1), D(m-1,n), D(m,n-1)) >= D(F)$ the rest of the row/column is then set as more than the cost estimate, thus avoiding calculating the accumulated cost for a portion of the matrix. This procedure can be seen in Figure 5.

4. EXPERIMENTAL EVALUATION

To evaluate WTW we used the score following system with ground truth MIDI, audio and path reference files and compared the accuracy of the found alignments with the known alignments. MATCH, the implementation of On-Line Time Warping [4], was also used to align the test pieces for comparison purposes. In both cases the MIDI was converted to audio using Timidity.

4.1 Mazurka Test Data

The CHARM Mazurka Project by the Centre for the History and Analysis of Recorded Music led by Nick Cook at Royal Holloway, University of London has published a large number of linked metadata files for Mazurka recordings in the form of reverse conducted data, [1] produced by Craig Sapp [12]. We then used template matching to combine this data with MIDI files, establishing links between MIDI notes and reverse conducted notes at the ms level. This provided a set of ground truth files linking the MIDI score to the audio recordings. These ground truths were compared with an off-line DTW alignment and manually supervised to correct any differences found. Overall, 217 sets of audio recordings, MIDI scores and reference files were produced.

[1] http://mazurka.org.uk/info/revcond/

4.2 Evaluation Metrics

For each path produced by WTW, each estimated audio note time was compared with the reference and the difference was recorded. For differing levels of accuracy requirements (100 ms, 200 ms, 500 ms and 2000 ms), the percentages of notes that were estimated correctly within this requirement for each piece were recorded. These piecewise accuracies are then averaged for an overall rating. The 2000 ms accuracy requirement is used as the MIREX score following accuracy requirement for notes hit.

4.3 Window Dimensions

The effect of the window size and hop size in WTW is examined in Table 1. The accuracy tests (shown in the top half) show a trend that suggests larger window sizes and smaller hop sizes lead to greater accuracy, as is similar to feature frame dimensions. However, larger window sizes and smaller hop sizes also lead to slower execution times as more points on the similarity matrix were calculated.

4.4 Window Guidance

A comparison of guidance methods for WTW is shown in Table 2. This comparison shows that for the test data used, there was not much difference between directed and undirected WTW and directed only offered an improvement when a large local constraint was used.

Alignment Accuracy at 2000 ms				
	Window Size			
Hop Size	100	200	300	400
100	76.0%	83.1%	81.8%	81.9%
200	63.7%	82.2%	82.0%	82.0%
300	57.7%	77.7%	81.2%	82.5%
400	57.4%	66.1%	81.1%	82.0%

Table 1. Accuracy test results comparing different window and hop sizes for WTW. For this test there was a guidance FGP that used a Type VII +6 *movement constraint* (see Table 2) and the accumulated cost matrix used a Type I *cost constraint* and Type I *movement constraint*.

Alignment Accuracy				
Acc. Req.	100 ms	200 ms	500 ms	2000 ms
None	63.8%	75.9%	82.0%	86.9%
Type I	56.2%	68.7%	74.2%	78.1%
Type IV	63.3%	74.8%	80.7%	86.0%
Type VII	64.0%	76.9%	82.6%	86.6%
Type IV +4	58.0%	70.3%	75.8%	79.5%
Type VII +6	59.4%	72.2%	78.0%	81.2%
Type II	63.9%	75.8%	81.8%	87.3%
Type V	64.9%	78.0%	83.9%	88.1%

Table 2. Accuracy test results comparing different methods for guiding the windows in WTW. The name of the guidance method refers to the *movement constraint* used in the Forward Greedy Path. The 'Type 4 +4' and 'Type 7 +6' constraints include additional horizontal and vertical frames to complete the block. For this test the window and hop size were set at 300ms and the accumulated cost matrix used a Type I *cost constraint* and Type I *movement constraint*.

4.5 Accuracy Results

The results of the accuracy test can be seen in Table 3. From this test we can see WTW produces an accuracy rate comparable with that of OTW. What separates the two methods is that the OTW method took on average 7.38 seconds to align a Mazurka audio and score file where as WTW took 0.09 seconds, (approximately one 80th of the time of OTW). The average length of the Mazurka recordings is 141.3 seconds, therefore, in addition to having the ability to calculate the alignment path sequentially, both methods achieve greater than real-time performance by some margin.

4.6 Efficiency Results

The efficiency tests consisted of aligning sequences of different lengths and recording the execution time. The results of this test can be seen in Table 4. These results show that WTW has linear time costs in relation to the length of the input sequence, unlike standard DTW. The optimisations suggested in this work are shown to decrease the time cost in aligning larger sequences over FastDTW.

Alignment Accuracy				
Acc. Req.	100 ms	200 ms	500 ms	2000 ms
WTW	73.6%	88.8%	94.9%	97.0%
OTW	70.9%	86.7%	94.8%	97.3%

Table 3. Accuracy test results comparing WTW and OTW estimated audio note onset times against references for 217 Mazurka recordings at 4 levels of accuracy requirements. For this test the window and hop size were set at 300ms and the accumulated cost matrix used a Type I *cost constraint* and Type VII *movement constraint*.

Execution time (seconds)				
Sequence length	100	1000	10000	100000
DTW	0.02	0.92	57.45	7969.59
FastDTW (r100)	0.02	0.06	8.42	207.19
WTW	0.002	0.06	0.90	9.52

Table 4. Efficiency test results showing the execution time (in seconds) for 4 different lengths of input sequences (in frames). Results for FastDTW and DTW are from [11]. The r value for FastDTW relates to the radius factor.

5. DISCUSSION AND CONCLUSION

This paper has introduced WTW, a linear cost variation on DTW for real-time synchronisations. WTW breaks down the regular task of creating an accumulated cost matrix between the complete series of input sequence vectors, into small, sequential, cost matrices. Additional optimisations include local constraints in the dynamic programming and cut-off limits for the accumulated cost matrices.

Evaluation of WTW has shown it to be more efficient than state of the art DTW based off-line alignment techniques. WTW has also been shown to match the accuracy of OTW whilst improving on the time taken to process files. Whilst this difference has little effect when synchronising live sequences on standard computers, the greater efficiency of WTW could be useful in running real-time synchronisation methods on less powerful processors, such as those in mobile phones, or when data-mining large data-sets for tasks such as cover song identification.

Future work will involve evaluating WTW on a wider variety of test data-sets, including non-audio related tasks and features. Possible improvements may be found in novel local constraints and/or the dynamic programming used to estimate the start and end points of the accumulated cost matrices. Presently, WTW assumes the alignment is continuous from the start to the end. A more flexible approach will be required to handle alignments made of partial sequence matches. Also, the modifications of WTW could potentially be combined with other modifications of DTW, such as those in FastDTW in order to pool efficiencies. Lastly, WTW, like DTW, is applicable to a number of tasks that involve data-mining, recognition systems or similarity measures. It is hoped WTW makes DTW viable for applications on large data sets in a wide range of fields.

6. ACKNOWLEDGMENTS

This work is supported by the EPSRC project OMRAS2 (EP/ E017614/1). Robert Macrae is supported by an EPSRC DTA Grant.

7. REFERENCES

[1] Bir Bhanu and Xiaoli Zhou. Face recognition from face profile using dynamic time warping. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4*, pages 499–502, Washington, DC, USA, 2004. IEEE Computer Society.

[2] David Clifford, Glenn Stone, Ivan Montoliu, Serge Rezzi, François-Pierre Martin, Philippe Guy, Stephen Bruce, and Sunil Kochhar. Alignment using variable penalty dynamic time warping. *Analytical Chemistry*, 81(3):1000–1007, January 2009.

[3] Simon Dixon. Live tracking of musical performances using on-line time warping. In *Proceedings of the 8th International Conference on Digital Audio Effects*, pages 92–97, Madrid, Spain, 2005.

[4] Simon Dixon and Gerhard Widmer. MATCH: A music alignment tool chest. In *Proceedings of the 6th International Conference on Music Information Retrieval*, page 6, 2005.

[5] Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289, New York, NY, USA, 2000. ACM.

[6] Benoít Legrand, C. S. Chang, S. H. Ong, Soek-Ying Neo, and Nallasivam Palanisamy. Chromosome classification using dynamic time warping. *Pattern Recogn. Lett.*, 29(3):215–222, 2008.

[7] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[8] Hiroshi Murase and Rie Sakai. Moving object recognition in eigenspace representation: gait analysis and lip reading. *Pattern Recogn. Lett.*, 17(2):155–162, 1996.

[9] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[10] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1), 1978.

[11] Stan Salvador and Philip Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. In *Workshop on Mining Temporal and Sequential Data*, page 11, 2004.

[12] Craig Sapp. Comparative analysis of multiple musical performances. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR) 2007*, pages 497–500, 2007.

[13] J Serrà, E Gómez, P Herrera, and X Serra. Chroma binary similarity and local alignment applied to cover song identification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(6):1138–1151, 2008.

[14] Robert J. Turetsky and Daniel P.W. Ellis. Ground-truth transcriptions of real music from force-aligned midi syntheses. In *Proceedings of the 4th International Conference on Music Information Retrieval*, 2003.

[15] H. J. L. M. Vullings, M. H. G. Verhaegen, and H. B. Verbruggen. Automated ECG segmentation with dynamic time warping. In *Proceedings of the 20th Annual International Conference of the IEEE In Engineering in Medicine and Biology Society, 1998*, volume 1, pages 163–166, 1998.

MUSIC STRUCTURE DISCOVERY IN POPULAR MUSIC USING NON-NEGATIVE MATRIX FACTORIZATION

Florian Kaiser and Thomas Sikora
Communication Systems Group
Technische Universität Berlin
{kaiser, sikora}@nue.tu-berlin.de

ABSTRACT

We introduce a method for the automatic extraction of musical structures in popular music. The proposed algorithm uses non-negative matrix factorization to segment regions of acoustically similar frames in a self-similarity matrix of the audio data. We show that over the dimensions of the NMF decomposition, structural parts can easily be modeled. Based on that observation, we introduce a clustering algorithm that can explain the structure of the whole music piece. The preliminary evaluation we report in the the paper shows very encouraging results.

1. INTRODUCTION

Music structure discovery (MSD) aims at characterizing the temporal structure of songs. In the case of popular music, this means classifying segments of a music piece into parts such as intro, verse, bridge, chorus or outro. Knowing this musical structure, one can introduce new paradigms in dealing with music collections and develop new applications such as audio thumbnailing and summarization for fast acoustic browsing, active listening (audio based retrieval and organization engines), song remixing or restructuring, learning semantics, etc.

In the past years, MSD has therefore gained an increasing interest in the music information retrieval community. This also led to the constitution of common evaluation data sets and evaluation campaigns (MIREX 09) that strongly stimulate the research in this field.

1.1 Previous work

Structure in music can be defined as the organization of different musical forms or parts through time. How we define musical forms and what builds our perception of these forms is however an open question, and MSD algorithms that have been proposed yet mainly differ in the way they answer those questions. However, Bruderer gives in [2] a general understanding of perception of structural boundaries in popular music, and shows that perception of structure is mainly influenced by a combination of changes in timbre, tonality and rhythm over the music pieces. Therefore, MSD algorithms generally aim at finding similarities and repetitions in timbre, tonality and rhythm based descriptions of the audio signal.

In [4], Foote and Cooper addressed the task of music summarization and proposed to visualize and highlight these repetitions in the audio signal through a self-similarity matrix. The audio signal is therefore parametrized through the extraction of audio features and the similarity between each frame is then measured. Thus using different audio features and similarity measures, most MSD algorithms are a processing of such a self-similarity representation.

In [13], the author distinguishes two categories of structure in the self-similarity matrix: the state representation and the sequence representation. The state representation defines the structure as a succession of states (parts). Each state is a succession of frames that show similar acoustic properties and therefore forms blocks in the self-similarity matrix. This representation is closely related to the notion of structural parts in popular music (intro - verse - chorus - outro), in which the acoustical information does not vary much. Algorithms based on state representation usually start with a segmentation by audio novelty score method [5]. The segments are then merged together with mean of hierarchical clustering, spectral clustering, or HMM.

On the other hand, the sequence representation considers series of times (frames), that are repeated over the music piece. The sequence representation is more related to musical concepts such as melody, progression in chords and harmony. Algorithms based on sequence representation look for repetitions on the off-diagonals of the self-similarity matrix. Matrix filtering of higher-order matrix transformations [14] can also be applied to the self-similarity matrix in order to emphasize off-diagonals. One of the main drawbacks of the sequence representation is that the structure of the music piece can not be fully explained unless all sequences are repeated at least once.

1.2 Approach

Non-negative matrix factorization (NMF) is a low-rank approximation technique that was first introduced in [9]. It is known for extracting parts-based representation of data, that strongly relates to some form of inherent structure in the data. Therefore, it has been successfully used in

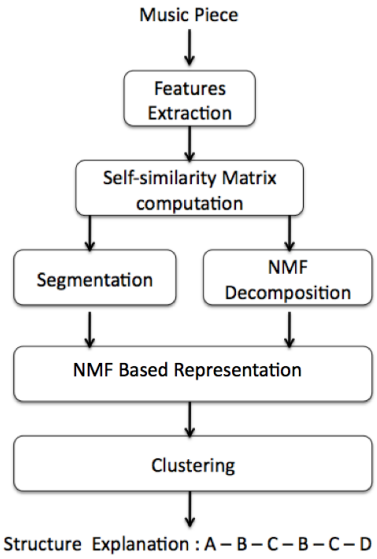


Figure 1. Overview of the proposed music structure discovery system

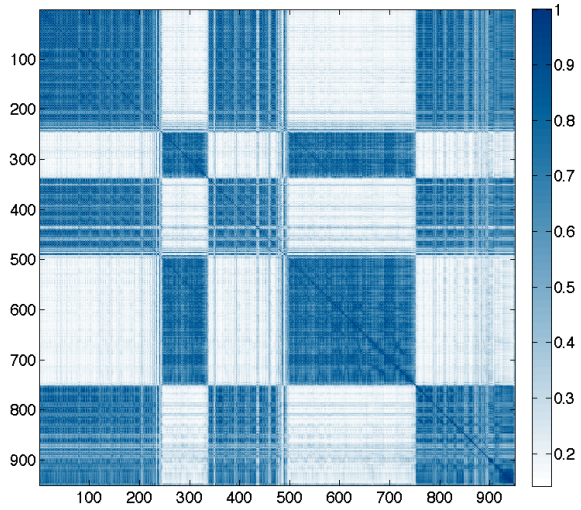


Figure 2. Self-similarity matrix computed on the timbre-related features using the exponential variant of the cosine distance. Audio file : "Creep" by Radiohead

a wide range of multimedia information retrieval applications such as text summarization [9] or sound classification [1]. Moreover, Foote et al. showed in [3] that decomposing the self-similarity matrix of a video stream via NMF could help separating visually similar segments . We propose to extend the approach of Foote to music data.

Defining structural parts as acoustically similar regions like in the state-representation, we apply NMF to the self-similarity matrix. We show that such structural parts can easily be discriminated over the dimensions of the obtained decomposition. With a clustering approach, we are thus able to merge together similar audio segments in the NMF decomposed matrices, and explain the structure of the whole music piece.

In the next section, we provide a detailed description of our system. Evaluation metrics, data set and results are presented in section 3. Section 4. concludes the paper.

2. PROPOSED METHOD

An overview of our system is shown in Figure 1. In this section each individual block of the system is described.

2.1 Feature Extraction

We first extract a set of audio features that are likely to model variations between different musical parts. As mentioned in the introduction, perception of structural boundaries in music is mostly influenced by variations in timbre, tonality and rhythm [2]. However, few rhythmical changes occur between parts in our evaluation data set (see section 3.) and we thus only focus on the description of timbre and tonality. Nevertheless, the reader might refer to [11] for interesting work also using rhythmical clues for structure discovery.

Timbre properties of the audio signal are described by extraction of the following features: the first 13 MFCC coefficients, spectral centroid, spectral slope and spectral spread.

Tonality can be associated to the concepts of melody and harmony. Songs in a popular music context are however very diverse and a melody extractor would hardly be robust over a whole set of popular songs. We thus only focus on the description of harmonic properties through the extraction of the chroma features. Chroma features are 12 dimensional, each element corresponding to a pitch-class profile of a 12 scaled octave.

The frame analysis is performed with mean of a window size of 400 ms and a hop size of 200 ms. Each feature is normalized to mean zero and variance one.

Timbre-related features and chroma features are stored in two different feature matrices and processed separately.

2.2 Self-Similarity Matrix

After parameterization of the audio, we measure the similarity between each signal frame in a self-similarity matrix $\mathbf{S}$. Each element s_{ij} is defined as the distance between the feature vectors $\mathbf{v}_i$ and $\mathbf{v}_j$, extracted over frames i and j. The cosine angle is used as a similarity measure :

$$d(\mathbf{v}_i, \mathbf{v}_j) = \frac{< \mathbf{v}_i, \mathbf{v}_j >}{||\mathbf{v}_i|| ||\mathbf{v}_j||} \quad (1)$$

As proposed in [3], an exponential variant of this distance is used to limit its range to [0,1] :

$$de(\mathbf{v}_i, \mathbf{v}_j) = exp(d(\mathbf{v}_i, \mathbf{v}_j) - 1) \quad (2)$$

As an example, we extracted the timbre-related features over the song "Creep" by Radiohead. The resulting self-similarity matrix is shown in Figure 2. One clearly sees that structural information is conveyed by the self-similarity matrix. Regions of acoustically similar frames form blocks in the matrix and one can also distinguish repetitions of these blocks. This illustrates the state representation of

structure, as explained in the introduction. In this specific example, there are few sequence repetitions to see on the off-diagonals. In fact, the clearness of such sequences in the self-similarity matrix pretty much depends on the nature of the song and the features that describe it (chroma features tend to highlight sequences). In our example, blocks are formed because of the strong presence of saturated guitar, which does not yield much timbre evolution within the structural parts.

2.3 Segmentation

Once the audio has been embedded in the self-similarity matrix $\mathbf{S}$, a segmentation step is needed to estimate potential borders of the structural parts. Therefore the self-similarity matrix is segmented using the audio novelty score introduced in [5]. The main idea is to detect boundaries by correlating a Gaussian checkerboard along with the diagonal of the self-similarity matrix $\mathbf{S}$. The checkerboard basically models the ideal shape of a boundary in $\mathbf{S}$. The correlation values yield a novelty score in which local maxima indicate boundaries. We apply an adaptive threshold as described in [6] to detect these maxima and generate the segmentation.

2.4 Non-negative Matrix Factorization

Matrix factorization techniques such as principal components analysis (PCA), independent component analysis (ICA) or vector quantization (VQ) are common tools for the analysis of multivariable data and are mainly used for dimensionality reduction purposes. In [7], Lee and Seung introduced non-negative matrix factorization (NMF), and proposed to build the decomposition additively by applying a non-negativity constraint on the matrix factors. Unlike PCA and other factorization techniques, cancelation of the decomposed data is thus not allowed, leading to a parts-based representation of the data. An intuitive justification is that not allowing negative coefficients in the decomposition will prevent the loss of the physical meaning of the data.

Given an $n \times m$ non-negative matrix $\mathbf{V}$, NMF aims at estimating the non-negative factors $\mathbf{W}$ $(n \times r)$ and $\mathbf{H}$ $(r \times m)$, that best approximate the original matrix :

$$\mathbf{V} \approx \mathbf{WH} \tag{3}$$

$\mathbf{W}$ contains the basis vectors and $\mathbf{H}$ the encoding coefficients for the best approximation of $\mathbf{V}$. The rank of the decomposition r is usually chosen so that $(n+m)r < nm$, thus providing a compressed version of the original data.

In our approach, we compute NMF on the self-similarity matrix of the audio in order to separate basic structural parts. The algorithm we use for the estimation of the matrix factors $\mathbf{W}$ and $\mathbf{H}$ is detailed in [8]. In the next section, we describe how the factorization via NMF relates to structure and show how we can use that result for music structure discovery.

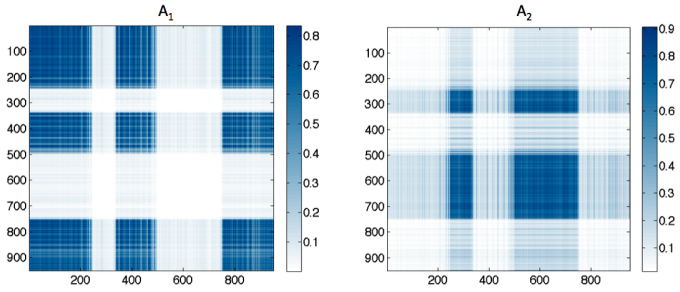


Figure 3. Matrices $\mathbf{A_1}$ and $\mathbf{A_2}$ obtained by NMF decomposition of the timbre self-similarity matrix of the song "Creep" (see Figure 2).

2.5 NMF based feature space

After decomposition via NMF, each element s_{ij} of $\mathbf{S}$ can be written as:

$$s_{ij} \approx \sum_{k=1}^{r} \mathbf{A}_k(i,j) \tag{4}$$

with

$$\mathbf{A}_k = \mathbf{W}(:,k)\mathbf{H}(k,:) \tag{5}$$

To illustrate how NMF can decompose data into basic structural parts, we compute NMF on the self-similarity matrix calculated over the song "Creep" by Radiohead. The rank of decomposition is set to 2 and the decomposed matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are shown in Figure 3.
According to the timbre description in Figure 2, we can say that the music piece is composed of two main structural parts. Figure 3 shows that these two parts are strongly separated over the two dimensions of the NMF decomposition.

This suggests that each dimension of the NMF decomposition somehow relates the contribution of a structural part in the original data. In other words, that means that there is a specific energy distribution over the dimensions of the decomposition for each structural part.

Therefore it seems relevant to study for each segment how the energy is distributed over the matrices $\mathbf{A_k}$. In order to consider temporal dependencies, we choose to consider segments as successions of frames in matrices $\mathbf{A_k}$, and not as blocks. That means that each frame from the music piece is represented by its corresponding values over the diagonals of matrices $\mathbf{A_k}$. We thus define the feature vector $\mathbf{d}_k$, representing the contribution of the k^{th} decomposition over all frames:

$$\mathbf{d}_k = diag(\mathbf{A}_k) \tag{6}$$

Each frame can then be represented in the $(n \times r)$ feature space $\mathbf{D}$:

$$\mathbf{D} = [\mathbf{d}_1 \mathbf{d}_2 \dots \mathbf{d}_r] \tag{7}$$

To illustrate this approach, we show an example with the song "Help" by The Beatles. The self-similarity matrix $\mathbf{S}$ computed on the timbre features of the song and the annotated structure are plotted in Figure 4. We compute the

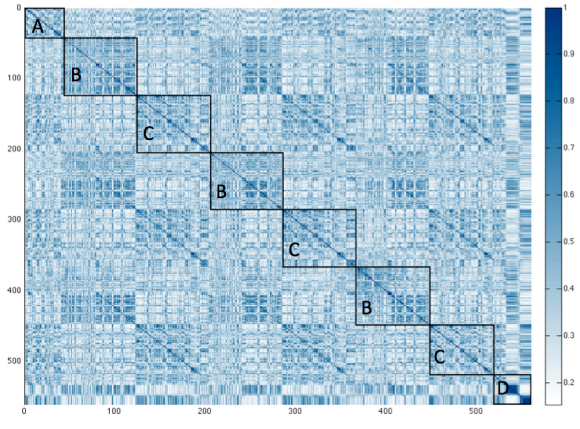


Figure 4. Self-similarity matrix computed on the timbre-related features for the song "Help" by The Beatles. The black boxes indicate the annotated segments, with A being the intro, B the verse, C the chorus and D the outro.

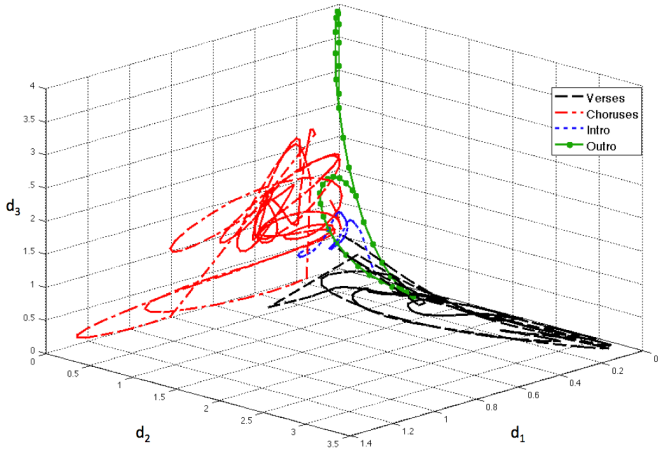


Figure 5. Representation of the structural parts of the song "help" in the feature space $\mathbf{D}$

NMF decomposition of $\mathbf{S}$. For visualization purposes, the rank of decomposition is set to 3. In Figure 5, each of the annotated segments is represented in the feature space $\mathbf{D}$. It is clear that structural parts chorus, verse and outro tend to be well represented over feature vectors $\mathbf{d}_1$, $\mathbf{d}_2$ and $\mathbf{d}_3$ respectively. In this case, we can say that each dimension of the NMF decomposition relates the contribution of a structural part. It is also interesting to note that segments of the same structural part seem to follow similar trajectories, suggesting that temporal dependencies should also be considered.

In classification problems, a feature space should provide good separability between classes. This means that the set of observations for a single class should have a small variance, whereas the set of all observations (for all the classes) should have a large variance. In that sense and according to Figure 5, representing segments in the feature space $\mathbf{D}$ should provide a good basis for structural classification.

2.6 Clustering

Each found segment is now represented in the NMF based feature space $\mathbf{D}$. In order to merge together segments belonging to the same structural part, we propose to use a classical clustering approach. Therefore, the similarity between segments in $\mathbf{D}$ is measured with:

- The Bayesian information criterion (BIC)
- The Mahalanobis distance

The clustering is performed using the two measures separately. A comparison of the performance obtained with both measures is done in section 3. The clustering is done with a classical hierarchical approach.

3. EVALUATION

3.1 Data set

The evaluation data set consists of 174 songs from The Beatles, that were first manually annotated at Universistat Pompeu Fabra (UPF) [1] . Some corrections to the annotation were made at Tampere University of Technology (TUT) [2] . We call the data set *TUT Beatles*.

The structure in each music piece is annotated as a state representation and not as sequences (see section 1.). Each frame is thus affected to a label.

3.2 Metrics for the clustering evaluation

Evaluating the performance of a music structure detection algorithm is not simple. In fact musical structures are mostly hierarchical [10], meaning that the structure can be explained at different levels. For example, a structure A-B-A, could be also be described as abc-def-abc. We choose to evaluate our system using the pairwise precision, recall and F-measure. Therefore, we define F_a the set of identically labelled frames in the reference annotation, and F_e the set of identically labelled frames in the estimated structure. Pairwise precision, recall and F-measure, respectively noted P, R and F are then defined as :

$$P = \frac{|F_e \bigcap F_a|}{|F_e|} \tag{8}$$

$$R = \frac{|F_e \bigcap F_a|}{|F_a|} \tag{9}$$

$$F = \frac{2PR}{P+R} \tag{10}$$

These measures are not perfect for evaluating MSD algorithms because they do not reflect hierarchical aspects in the description of structure. Nevertheless, they give an idea of the global performance of the system.

[1] http://www.iua.upf.edu/%7Eperfe/annotaions/sections/license.html
[2] http://www.cs.tut.fi/sgn/arg/paulus/structure.html

	F-measure	Precision	Recall
Timbre	58.6%	58.1%	61.9%
Chroma	50%	46.5%	52.2%
both	53.6%	49%	55%

Table 1. Segmentation evaluation with the *TUT Beatles* database

3.3 Segmentation Evaluation

We evaluate the segmentation step with classical F-measure, precision and recall. Table 1 reports the performance of the segmentation computed on the timbre-related self-similarity matrix, the chroma-related self-similarity matrix and the sum of the two matrices.

The low precision rate in the segmentation suggests that the algorithm tends to over-segment the audio. In fact, structure is hierarchical and the annotation labels high level parts of the structure. The clustering might cope with that by reassembling segments from the same structural part.

3.4 Rank of decomposition

We ran a small experiment in order to choose a suitable rank for the NMF. Over a subset of ten songs from the database, we compute the similarity matrices. Varying the rank of NMF r from 3 to 12, we measure the separability between structural parts along each dimension d_i of $\mathbf{D}$. To do so, we compute the inertia ratio of the variance of d_i within segments belonging to the same structural part and the variance of d_i over the whole music piece [12]:

$$s(i) = \frac{\sum_{k=1}^{K} \frac{N}{N_k}(m_k - m_i)(m_k - m_i)'}{\frac{1}{N}\sum_{n=1}^{N}(d_i(n) - m_i)(d_i(n) - m_i)'} \quad (11)$$

With K being the number of structural parts, N_k the number of frames in structural part k and N the total number of frames. m_i is the mean of d_i over the all piece and m_k the mean value of d_i over the k^{th} structural part. For a given rank of decomposition r, the separability is then measured as the mean of s:

$$sep(r) = \frac{1}{r}\sum_{i=1}^{r} s(i) \quad (12)$$

We find a maximum of separability with a rank of 9 for NMF (see Figure 6). It is larger than the median number of annotated parts. In fact, as structure can be explained at different hierarchical levels, we don't expect the NMF decomposition to match the parts described in the annotation one-by-one.

3.5 Experimental set up for the clustering

Self-similarity matrices are computed over the timbre and chroma features separately. As shown in Table 1, segmentation using the timbre features provides better performances. Therefore, in the evaluation of the clustering step, we only use the segments positions extracted over the timbre-related self-similarity matrix. We propose four

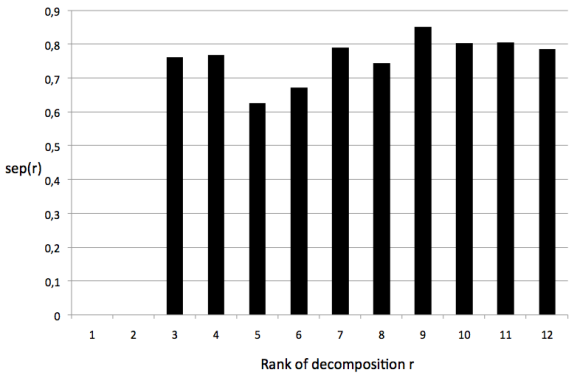


Figure 6. Separability of structural parts given different ranks of decomposition

strategies to evaluate our clustering approach. For the three first strategies, the NMF based feature space is obtained by decomposition of the timbre-related self-similarity matrix (labeled as "Timbre"), the chroma-related self-similarity matrix (labeled as "Chroma") and the sum of the two matrices (labeled as "Fusion 1"). We also study a second fusion strategy where similarity between segments is computed separately in the timbre and chroma related feature spaces and then summed for the clustering algorithm (labeled as "Fusion 2").

We also compare the clustering obtained using the automatic segmentation described in section 2. (labeled "auto") and using the annotated segments (labeled "manual"). Finally, each configuration is run using the BIC (Table 2) and the Mahalanobis distance (Table 3) as similarity measure for the clustering algorithm.

The number of clusters is set to 4, which is the median number of annotated parts within a song in our evaluation data set.

3.6 Clustering Evaluation and Discussion

As a reference we use the system described in [11], that was also evaluated on the *TUT Beatles* database. The system is based on a description of the audio signal through MFCC, chroma and rhythmogram features. Each of these features is then used to estimate the probability for two segments to belong to the same structural part and a fitness measure of the description is introduced. A greedy approach is used to generate the candidate descriptions.

Evaluation of the whole system is reported in Tables 2 and 3, using BIC and Mahalanobis distance respectively. Compared to the reference system, our system shows slightly better F-measure rates. The interesting result is that we show significantly better recall rates. This suggests that our algorithm splits the parts in the annotation as sequences of sub-parts. This also explains why we don't match the precision rates in [11]. There again, the annotation relates a high stage of the structure hierarchy, and over-segmentation causes a lack of precision. Modeling sequences of basic parts in our algorithm might cope with that. This also explains the huge gain of performance when using the annotated segments for the evaluation.

Method	*Segmentation*	F	P	R
[11]		59.9%	72.9%	54.6 %
Timbre	auto	60.2%	64.7%	60%
	manual	76.1%	83.6%	72.6%
Chroma	auto	60.5%	**66%**	59.6%
	manual	80%	87%	76.6%
Fusion 1	auto	60.6%	65%	60%
	manual	78.7%	85%	76.4%
Fusion 2	auto	60.2%	64.7%	60%
	manual	80%	86.5%	77%

Table 2. Evaluation on *TUT Beatles*, BIC

Method	*Segmentation*	F	P	R
[11]		59.9%	72.9%	54.6 %
Timbre	auto	61%	62.4%	63.3%
	manual	78.4%	82.1%	78.3%
Chroma	auto	60.8%	61.5%	**64.6%**
	manual	76.6%	81.2%	75.7%
Fusion 1	auto	**62.1%**	63.6%	64.5%
	manual	77.8%	82.3%	77%
Fusion 2	auto	61%	62.4%	63.3%
	manual	78%	81.7%	78.2%

Table 3. Evaluation on *TUT Beatles*, Mahalanobis

Obviously, fusing both timbral and chroma description as in the "Fusion 1" strategy makes sense and improves the overall performance of the system. Finally, using the Mahalanobis distance yields better performances than the BIC.

4. CONCLUSIONS

We introduced a music structure discovery method that uses the ability of NMF to generate parts-based representation of data. The evaluation conducted on the *TUT Beatles* data set shows that we are able to obtain slightly better performances than the reference system introduced in [11]. The improvements we obtain in the recall rates however suggest that there is still room for improvements. Moreover, the method used for the clustering of segments in the NMF based feature space only considers statistical similarity between the segments over time. We will consider modeling time dependencies between frames and thus model trajectories in the feature space instead of clouds of points. The NMF processing itself could also be enhanced by using sparse constraints on the matrix factors. Further evaluation on more diverse audio material will be done. The first results we obtained are however very encouraging.

5. ACKNOWLEDGMENT

This work was supported by the European Commission under contract FP7-21644 PetaMedia.

6. REFERENCES

[1] Emmanouil Benetos, Margarita Kotti, and Constantine Kotropoulos. Musical instrument classification using non-negative matrix factorization algorithms. In *ISCAS*. IEEE, 2006.

[2] Michael J. Bruderer, Martin F. McKinney, and Armin Kohlrausch. Structural boundary perception in popular music. In *ISMIR*, pages 198–201, 2006.

[3] Matthew L. Cooper and Jonathan Foote. Summarizing video using non-negative similarity matrix factorization. In *IEEE Workshop on Multimedia Signal Processing*, pages 25–28. IEEE Signal Processing Society, 2002.

[4] Jonathan Foote. Visualizing music and audio using self-similarity. In *ACM Multimedia (1)*, pages 77–80, 1999.

[5] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *IEEE International Conference on Multimedia and Expo (I)*, page 452, 2000.

[6] A.L. Jacobson. Auto-threshold peak detection in physiological signals. In *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, volume 3, pages 2194–2195 vol.3, 2001.

[7] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, oct 1999.

[8] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization, July 21 2000.

[9] Ju-Hong Lee, Sun Park, Chan-Min Ahn, and Daeho Kim. Automatic generic document summarization based on non-negative matrix factorization. *Inf. Process. Manage*, 45(1):20–34, 2009.

[10] Namunu C. Maddage. Automatic structure detection for popular music. *IEEE MultiMedia*, 13:65–77, 2006.

[11] Jouni Paulus and Anssi Klapuri. Music structure analysis using a probabilistic fitness measure and a greedy search algorithm. *IEEE Transactions on Audio, Speech & Language Processing*, 17(6):1159–1170, 2009.

[12] Geoffroy Peeters. Automatically selecting signal descriptors for sound classification. In *ICMC*, 2002.

[13] Geoffroy Peeters. Deriving musical structures from signal analysis for music audio summary generation: "sequence" and "state" approach. In Uffe Kock Wiil, editor, *CMMR*, volume 2771 of *Lecture Notes in Computer Science*, pages 143–166. Springer, 2003.

[14] Geoffroy Peeters. Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach. In *ISMIR*, 2007.

MUSICAL INSTRUMENT RECOGNITION USING BIOLOGICALLY INSPIRED FILTERING OF TEMPORAL DICTIONARY ATOMS

Steven K. Tjoa and K. J. Ray Liu
Signals and Information Group, Department of Electrical and Computer Engineering
University of Maryland – College Park, MD 20742 USA
{kiemyang, kjrliu}@umd.edu

ABSTRACT

Most musical instrument recognition systems rely entirely upon spectral information instead of temporal information. In this paper, we test the hypothesis that temporal information can improve upon the accuracy achievable by the state of the art in instrument recognition. Unlike existing temporal classification methods which use traditional features such as temporal moments, we extract novel features from temporal atoms generated by nonnegative matrix factorization by using a *multiresolution gamma filterbank*. Among isolated sounds taken from twenty-four instrument classes, the proposed system can achieve 92.3% accuracy, thus improving upon the state of the art.

1. INTRODUCTION

Advances in sparse coding and dictionary learning have influenced much of the recent progress in musical instrument recognition. Many of these methods depend upon nonnegative matrix factorization (NMF) – a popular, convenient, and effective method for decomposing matrices – to obtain low-rank approximations of audio spectrograms [9]. NMF yields a set of vectors, spectral atoms, which approximately span the frequency space of the spectrogram, and another set of vectors, temporal atoms, which correspond to the temporal activation of each spectral atom. The spectral atoms can then be classified by instrument using features such as mel-frequency cepstral coefficients (MFCCs).

While these methods are effective in exploiting the spectral redundancy in a signal, redundancy remains in the *temporal* domain. Psychoacoustic studies have shown that spectral and temporal information are equally important in the definition of acoustic timbre [10]. Classification methods that only utilize spectral information are discarding the potentially useful temporal information that could be used to improve classification performance.

In this paper, we combine advances in dictionary learning, auditory modeling, and music information retrieval to propose a new timbral representation. This representation is inspired by another widely accepted timbral model, the cortical representation, which estimates the spectral and temporal modulation content of the auditory spectrogram. Our method of extracting temporal information uses a *multiresolution gamma filterbank* which is computed from the temporal atoms extracted from spectrograms using NMF. Extracting and classifying this feature is *simple yet effective* for musical instrument recognition.

After defining the proposed feature extraction and classification method, we test the hypothesis that the proposed feature improves upon the accuracy achievable by the state of the art in musical instrument recognition. For isolated sounds, we show that temporal information can be used to build a classifier capable of 72.9% accuracy when tested among 24 instrument classes. However, when combining temporal and spectral features, the proposed classifier can achieve an accuracy of **92.3%**, thus reflecting state of the art performance.

2. TEMPORAL INFORMATION

Temporal information is incorporated into timbral models in different ways. Many attempts to incorporate temporal information use features such as the temporal centroid, spread, skewness, kurtosis, attack time, decay time, slope, and locations of maxima and minima [5,6]. One timbral representation, the *cortical representation*, incorporates both spectral and temporal information. Essentially, the cortical representation embodies the output of cortical cells as sound is processed by earlier stages in the auditory system. Fig. 1 illustrates the relationship between the early and middle stages of processing in the mammalian auditory system. The early stage models the transformation by the cochlea of an acoustic input signal into a neural representation known as the auditory spectrogram, while the middle stage models the analysis of the auditory spectrogram by the primary auditory cortex.

One property of cortical cells, the spectrotemporal receptive field (STRF), summarizes the way a single cortical cell responds to a stimulus. Mathematically, the STRF is like a two-dimensional impulse response defined across time and frequency. Each STRF has three parameters: scale, rate, and orientation. Scale defines the spectral resolution of an STRF, rate defines its temporal resolution, and orientation determines if the STRF selects upward or down-

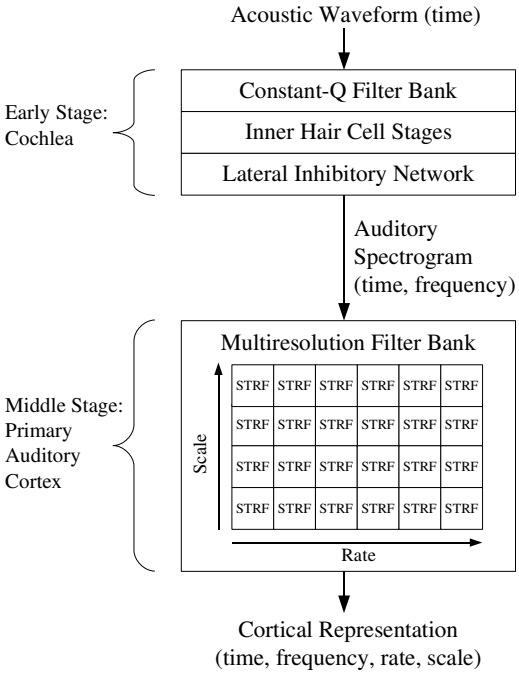


Figure 1. Early and middle stages of the auditory system. The auditory spectrogram is convolved across time and frequency with STRFs of different rates and scales to produce the four-dimensional cortical representation.

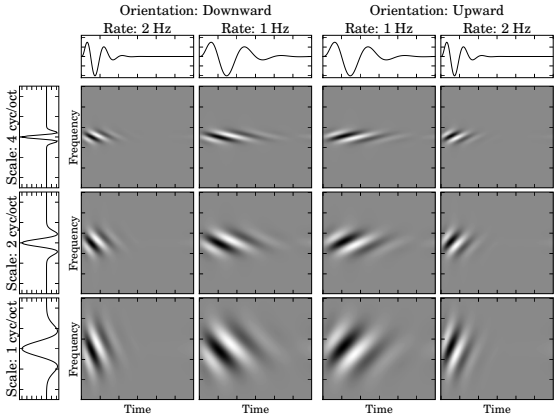


Figure 2. Twelve example STRFs. Together, they constitute a filterbank. The left six STRFs select downward-modulating frequencies, and the right six STRFs select upward-modulating frequencies. Top row: seed functions for rate determination. Left column: seed functions for scale determination.

ward frequency modulations. Fig. 2 illustrates the STRF as a function of these three parameters. Each cortical cell can be interpreted as a filter whose impulse response is an STRF with a particular rate, scale, and orientation. Therefore, a collection of cortical cells constitutes a filterbank. Indeed, it turns out that the cortical representation is mathematically equivalent to a multiresolution wavelet filterbank [2].

Despite the biological relevance between the cortical representation and timbre, this representation has disadvantages for classification purposes. First, because the cortical representation is a complex-valued four-dimensional filterbank output, it is massively redundant. Like many types of redundant data, the cortical representation could benefit from some form of coding, decomposition, or dimensionality reduction. However, proper application of these tools to the cortical representation for engineering purposes such as speech recognition and MIR is not yet well understood. Therefore, these are ongoing areas of research [11]. Second, the STRF is not time-frequency separable [2]. In other words, computation of the cortical representation cannot be decomposed into two procedures that operate on the time and frequency dimensions separately. Because spectral and temporal information require different classification methods, this obstacle impedes classification.

Unlike the cortical representation, the spectrogram computed via short-time Fourier transform (STFT) is easily decomposed, particularly for musical signals. For example, many works have applied decomposition methods to magnitude spectrograms of musical sounds in order to identify a set of spectral and temporal basis vectors from which the magnitude spectrogram can be parameterized [15]. One such decomposition method is NMF [9]. Given an element-wise nonnegative matrix $\mathbf{X}$, NMF attempts to find two nonnegative matrices, $\mathbf{A}$ and $\mathbf{S}$, that minimize some divergence between $\mathbf{X}$ and $\mathbf{AS}$. Among the algorithms that can perform this minimization, one of the most convenient algorithms uses a multiplicative update rule during each iteration in order to maintain nonnegativity of the matrices $\mathbf{A}$ and $\mathbf{S}$ [9].

Many researchers have already demonstrated the usefulness of NMF for separating a musical signal into individual notes [7,15,16]. By first expressing a time-frequency representation of the signal as a matrix, these methods decompose the matrix into a summation of a few individual *atoms*, each corresponding to one musical source or one note. Fig. 3 illustrates the use of NMF upon the spectrogram of a musical signal. We define each column of $\mathbf{A}$ as a *spectral atom* and each row of $\mathbf{S}$ as a *temporal atom*. The temporal atoms usually resemble envelopes of known sounds, particularly in musical signals. For example, observe the difference between the profiles of the temporal atoms in Fig. 3. The three beats generated by the kick drum share the same temporal profiles, and the two beats generated by the snare drum share the same profiles. This general observation motivates the hypothesis that the energy distribution of temporal NMF atoms is a valid timbral representation that can be used to classify instruments.

In the next section, we propose one technique that extracts timbral information from temporal NMF atoms similar to that of the cortical representation. Our technique uses a *multiresolution gamma filterbank* to perform multiresolution analysis upon the factorized spectrogram. However, unlike the cortical representation, this multiresolution analysis is particularly suited to the energy profiles contained in the temporal NMF atoms.

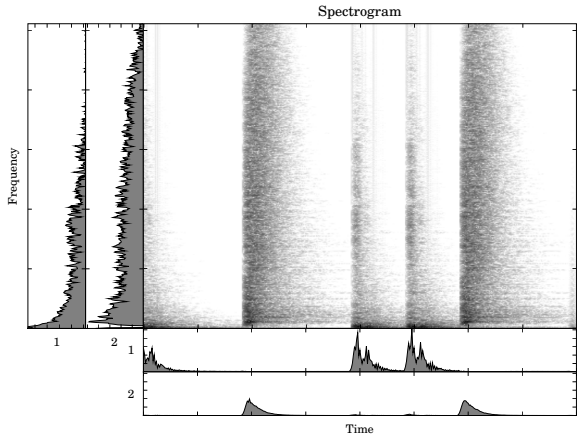


Figure 3. The NMF of a spectrogram drum beats. Component 1: kick drum. Component 2: snare drum. Top right: $\mathbf{X}$. Left: $\mathbf{A}$. Bottom: $\mathbf{S}$.

3. PROPOSED METHOD: MULTIRESOLUTION GAMMA FILTERBANK

The multiresolution gamma filterbank is a collection of gamma filters. For this work, we define the gamma kernel to be

$$g(t; n, b) = \alpha t^{n-1} e^{-bt} u(t) \tag{1}$$

where $b > 0$, $n \geq 1$, $u(t)$ is the unit step function, and

$$\alpha = \sqrt{\frac{(2b)^{2n-1}}{\Gamma(2n-1)}} \tag{2}$$

ensures that $\int |g(t; n, b)|^2 dt = 1$ for any value of n and b, where $\Gamma(n)$ is the Gamma function. Let I be the total number of gamma filters in the filterbank. For each $i \in \{1, ..., I\}$, define the correlation kernel (i.e., time-reversed impulse response) of each gamma filter to be

$$g_i(t) = g(t; n_i, b_i). \tag{3}$$

The set of kernels $\{g_1, g_2, ..., g_I\}$ defines the *multiresolution gamma filterbank*. Fig. 4 illustrates some example kernels of the filterbank.

For each i, let the filter output be the cross-correlation between the input atom, $s(t)$, and the kernel, $g_i(t)$:

$$y_i(\tau) = \int_{-\infty}^{\infty} s(t) g_i(t - \tau) dt \tag{4}$$

The set of outputs $\{y_1, y_2, ..., y_I\}$ from the filterbank is called the *multiresolution gamma filterbank response* (MGFR).

The gamma filter has convenient temporal properties. We define the *attack time* of the kernel $g(t)$ to be the time elapsed until the kernel achieves its maximum. By differentiating $\log g(t)$, we determine the attack time to be

$$t_a = (n - 1)/b \text{ seconds.} \tag{5}$$

Fig. 4 illustrates the relationship between the attack time and the parameter b. Also, as t becomes large, $\log g(t) \approx$

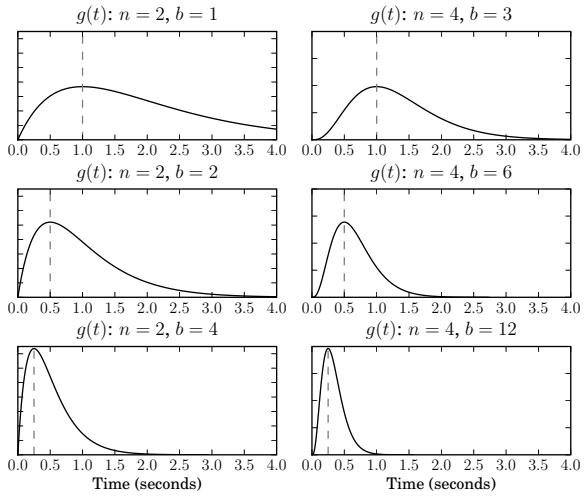


Figure 4. Kernels of gamma filters. The dashed vertical line indicates the location of the maxima. Left column: $n = 2$. Right column: $n = 4$.

$-bt$ plus a constant. Therefore, b is the decay parameter of $g(t)$, where we define the *decay rate* of $g(t)$ to be

$$r_d = 20b \log_{10} e \approx 8.7b \text{ dB per second.} \tag{6}$$

Together, these two temporal properties imply that a gamma kernel with *any* attack time and decay rate can be created from the proper combination of n and b.

Fig. 5 illustrates the operation of the multiresolution gamma filterbank. When a temporal NMF atom is sent through the multiresolution gamma filterbank, the MGFR reveals the strength of the attacks and decays of the atom's envelope for different values for n and b. Observe how the filterbank response is largest for those filters whose attack time matches that of the input atom.

The multiresolution gamma filterbank behaves like a set of STRFs. Both systems perform multiresolution analysis on the input data. Each STRF passes a different spectrotemporal pattern depending upon the rate and scale. In fact, the seed function used to determine the rate of an STRF is a gammatone kernel – a sinusoid whose envelope is a gamma kernel. By altering the parameters of the gammatone kernel, STRFs can select different rates. Similarly, in the multiresolution gamma filterbank, each filter passes different envelope shapes depending upon the parameters n and b which completely characterize the attack and decay of the envelope. Intuitively, the filter with kernel $g_i(t)$ passes envelopes with attack times equal to $(n_i - 1)/b_i$ seconds and envelopes with decay rates equal to $8.7b_i$ dB per second.

4. PROPOSED FEATURE EXTRACTION AND CLASSIFICATION

To extract a shift-invariant feature from the MGFR, we compute the norm for each filter response:

$$z_i = \left(\int_{-\infty}^{\infty} |y_i(t)|^p dt \right)^{1/p} \tag{7}$$

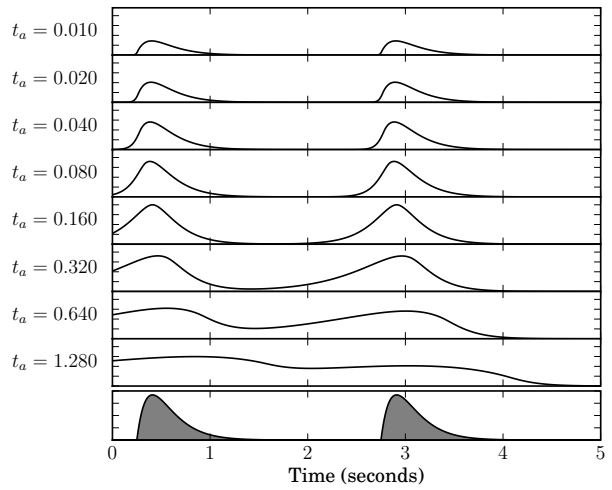


Figure 5. Top: MGFR as a function of time for $n = 2$. Bottom: input atom containing two pulses with attack times of 160 ms.

The vector $\mathbf{z} = [z_1, z_2, ..., z_I]$ is the extracted feature vector. To eliminate scaling ambiguities among the input atoms, every feature vector $\mathbf{z}$ is normalized to have unit Euclidean norm. Different choices of p provide different interpretations of $\mathbf{z}$. For this work, we use $p = \infty$. Our future work will include an investigation into the impact of p on classification performance.

The *proposed feature extraction algorithm* is summarized below.

1. Perform NMF on the magnitude spectrogram, $\mathbf{X}$, to obtain $\mathbf{A}$ and $\mathbf{S}$.
2. Initialize the multiresolution gamma filterbank in (3).
3. For each temporal atom (i.e., row of $\mathbf{S}$), compute the MGFR in (4).
4. Compute the feature vector $\mathbf{z}$ in (7).

Finally, we formulate the instrument recognition problem as a typical supervised classification problem: given a set of training features extracted from signals of known musical instruments, identify all of the instruments present in a test signal. To perform supervised classification, temporal atoms are extracted from training signals of known musical instruments using NMF. The feature vector $\mathbf{z}$ computed from the atom plus its instrument label are used for training. To predict the label of an unknown sample, $\mathbf{z}$ is extracted from the unknown sample and classified using the trained model.

An advantage of the proposed feature extraction and classification procedure is its *simplicity*. The proposed system requires no rule-based preprocessing. Unlike other systems that contain safeguards, thresholds, and hierarchies, the proposed system uses straightforward filtering and a flat classifier. As the next section shows, this simple procedure can achieve state-of-the-art accuracy for instrument recognition.

n	b	t_a	n	b	t_a
1.2	0.200	1.000	1.5	0.500	1.000
1.2	0.250	0.800	1.5	0.625	0.800
1.2	0.333	0.600	1.5	0.833	0.600
1.2	0.500	0.400	1.5	1.25	0.400
1.2	1.00	0.200	1.5	2.50	0.200
1.2	2.00	0.100	1.5	5.00	0.100
1.2	4.00	0.050	1.5	10.0	0.050
1.2	10.0	0.020	1.5	25.0	0.020
2.0	1.00	1.000	3.0	2.00	1.000
2.0	1.25	0.800	3.0	2.50	0.800
2.0	1.67	0.600	3.0	3.33	0.600
2.0	2.50	0.400	3.0	5.00	0.400
2.0	5.00	0.200	3.0	10.0	0.200
2.0	10.0	0.100	3.0	20.0	0.100
2.0	20.0	0.050	3.0	40.0	0.050
2.0	50.0	0.020	3.0	100	0.020

Table 1. Gamma filterbank parameters used in the following experiments.

5. EXPERIMENTS

We perform experiments on an extensive set of isolated sounds. The data set for these experiments combines samples from the University of Iowa database of Musical Instrument Samples [4], McGill University Master Samples [14], the OLPC Samples Collection [13], and the Freesound Project [12]. All of these samples consist of isolated sounds generated by real musical instruments. We have parsed the audio files such that each file consists of a single musical note (for harmonic sounds) or beat (for percussive sounds).

From each input signal, $x(t)$, we obtain the magnitude spectrogram, $\mathbf{X}$, via STFT using frames of length 46.4 ms (i.e., 2048/44100) windowed using a Hamming window and a hop size of 10.0 ms. Then, we perform NMF using the Kullback-Leibler update rules [9] with an inner dimension of $K = 1$ to obtain $\mathbf{A}$ and $\mathbf{S}$. When applicable, we use a multiresolution gamma filterbank of thirty-two filters with the parameters shown in Table 1. These attack times and decay rates cover a wide range of sounds produced by common musical instruments. Each 32-dimensional feature vector, $\mathbf{z}$, is then classified.

For supervised classification, we use the LIBSVM implementation [1] of the support vector machine (SVM) with the radial basis kernel. For multiple classes, LIBSVM uses the one-versus-one classification strategy by default. The remaining programs and simulations were written entirely in Python using the SciPy package [8]. Source code is available upon request.

In total, there are 3907 feature vectors collected among twenty-four instrument classes. Table 2 summarizes this data set. With few exceptions [3], this selection of instruments is more comprehensive than any existing work on isolated instrument recognition. Recognition accuracy for class c is defined to be the percentage of the feature vectors whose true class is c that are correctly classified by the SVM as belonging in class c. Overall recognition accuracy is the average of the accuracy rates for each class.

Instrument	#	S	T	ST
Bassoon	131	**99.2**	75.6	96.9
Clarinet	145	80.7	73.1	**86.2**
Flute	236	84.7	60.6	**89.0**
Oboe	118	72.0	77.1	**91.5**
Saxophone	196	**93.4**	65.8	86.7
Horn	92	80.4	62.0	**85.9**
Trombone	99	**93.9**	53.5	89.9
Trumpet	236	97.5	82.2	**97.9**
Tuba	111	98.2	75.7	**99.1**
Cello	349	94.8	89.7	**97.4**
Viola	309	**94.2**	67.6	90.9
Violin	390	**97.2**	86.2	96.2
Cello Pizz.	321	98.1	87.5	**98.4**
Viola Pizz.	254	**99.6**	81.9	**99.6**
Violin Pizz.	315	97.5	85.4	**99.0**
Glockensp.	10	**100.0**	90.0	**100.0**
Guitar	27	51.9	29.6	**63.0**
Marimba	39	46.2	25.6	**79.5**
Piano	260	95.0	89.2	**98.5**
Xylophone	13	61.5	53.8	**84.6**
Kick	90	98.9	95.6	**100.0**
Snare	86	96.5	88.4	**98.8**
Timpani	47	85.1	61.7	**87.2**
Toms	33	**100.0**	90.9	**100.0**
Total	3907	88.2	72.9	**92.3**

Table 2. Sample sizes and accuracy rates. S: spectral information. T: temporal information. ST: spectral plus temporal information.

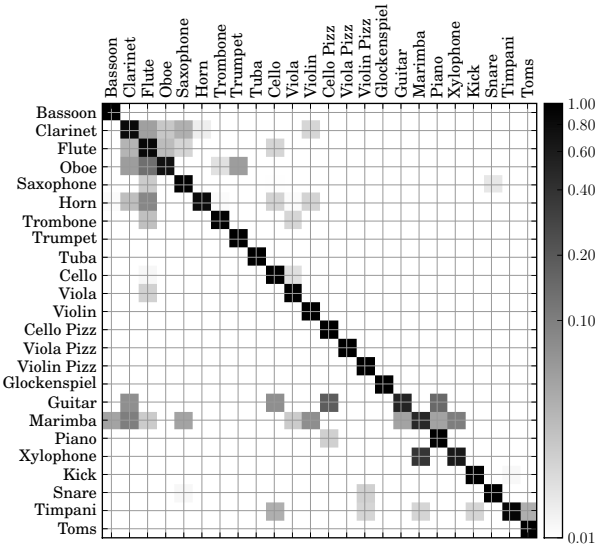


Figure 6. Classification accuracy using spectral information. Row labels: True class. Column labels: Estimated class. Average accuracy: 88.2%.

5.1 Spectral Information

As a control experiment, we evaluate the classification ability of spectral features using MFCCs. From each column of $\mathbf{A}$, we extract 32 MFCCs with center frequencies logarithmically spaced over 5.3 octaves between 110 Hz and 3951 Hz. From the 3907 32-dimensional feature vectors, we evaluate classification performance through ten-fold cross validation.

Fig. 6 illustrates the confusion matrix for this experiment, and Table 2 shows the accuracy rates for each class. The average of the 24 accuracy rates is 88.2%. We notice some understandable misclassifications. For example, 18.5% of guitar samples are misclassified as cello pizzicato and 14.8% are misclassified as piano. 5.5% of clarinet samples and 13.6% of oboe samples are misclassified as flute. 10.3% of marimba samples are misclassified as xylophone. In general, these spectral features can accurately classify the drums, brass, and string instruments. However, accuracy is poor among the woodwinds and pitched percussive instruments. Some of these misclassifications are due to an imbalance in the sample size of each class. Despite its ability to improve the average accuracy rate, the reduction of class imbalance in supervised classification is beyond the scope of this paper.

5.2 Temporal Information

Next, we evaluate the classification ability of temporal features using the proposed feature extraction algorithm with the parameters shown in Table 1. One feature vector $\mathbf{z}$ is computed for each temporal NMF atom as described in Section 4. Like the previous experiment, we evaluate classification performance through ten-fold cross validation among the 3907 32-dimensional feature vectors.

Table 2 shows the accuracy rates for each class. The average accuracy rate is 72.9%. Fig. 7 illustrates the confusion matrix for this experiment. We observe that temporal features alone do not classify instruments as well as spectral features. Nevertheless, for 11 out of the 24 classes, accuracy remains above 80%. In particular, there are very few misclassifications between percussion instruments and non-percussion instruments. Most misclassifications occur within instrument families, e.g., cello and viola, bassoon and clarinet, and guitar and piano.

5.3 Spectral Plus Temporal Information

Finally, we evaluate the classification performance when concatenating spectral and temporal features. The features extracted during the previous two experiments are concatenated to form 3907 64-dimensional feature vectors. Table 2 shows the accuracy rates, and Fig. 8 illustrates the confusion matrix. The total accuracy rate is **92.3%**. Temporal information improves classification accuracy for 16 of the 24 instrument classes along with the overall accuracy. Accuracy improves most for the string pizzicato, percussion, brass, and certain woodwind instruments. The remaining misclassifications occur mostly within families, e.g., clarinet and flute, and guitar and piano. For isolated sounds, this experiment verifies the hypothesis that temporal information can improve instrument recognition accuracy over methods that use only spectral information.

6. CONCLUSION

From the experiments, we conclude that a combination of spectral and temporal information can improve upon those instrument recognition systems that only use spectral information. The proposed method extracts temporal infor-

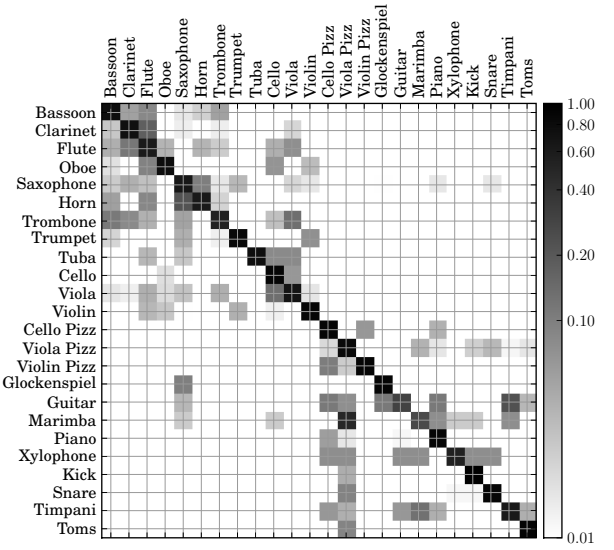


Figure 7. Classification accuracy using temporal information. Row labels: True class. Column labels: Estimated class. Average accuracy: 72.9%.

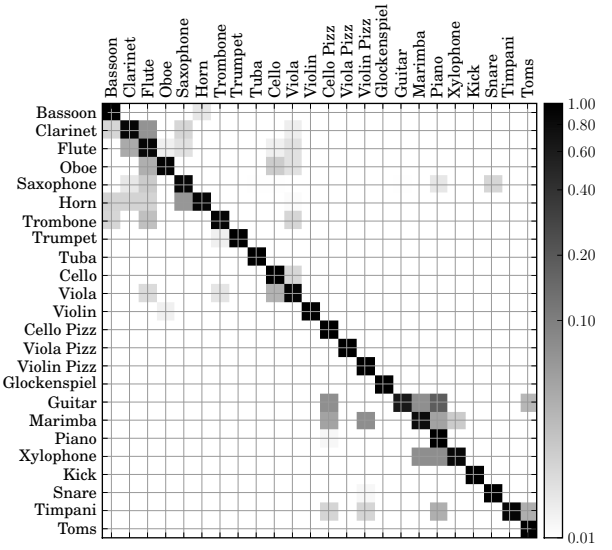


Figure 8. Classification accuracy using spectral plus temporal information. Row labels: True class. Column labels: Estimated class. Average accuracy: **92.3%**.

mation using a multiresolution gamma filterbank which parameterizes each temporal dictionary atom by its most prominent attack times and decay rates. Like the cortical representation, the spectral and temporal dictionary atoms generated by NMF provide a complete timbral representation of musical sounds. However, unlike the cortical representation, each of these dictionary atoms typically represent an individual musical note, thus facilitating music instrument recognition further.

We have already begun an investigation of the proposed method for both solo melodic excerpts and polyphonic mixtures. Also, because the proposed method classifies each individual NMF atom by instrument, we are investigating the use of the proposed method for source separation by grouping, emphasizing, or removing atoms that correspond to chosen instruments.

7. REFERENCES

[1] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," 2001–. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[2] T. Chi, P. Ru, and S. A. Shamma, "Multiresolution spectrotemporal analysis of complex sounds," *J. Acoustical Soc. America*, vol. 118, no. 2, pp. 887–906, Aug. 2005.

[3] A. Eronen, "Automatic musical instrument recognition," Master's thesis, Tampere University of Technology, Oct. 2001.

[4] L. Fritts, "Musical Instrument Samples," Univ. Iowa Electronic Music Studios, 1997–. [Online]. Available: http://theremin.music.uiowa.edu/MIS.html

[5] F. Fuhrmann, M. Haro, and P. Herrera, "Scalability, generability, and temporal aspects in automatic recognition of predominant musical instruments in polyphonic music," in *Proc. Intl. Soc. Music Information Retrieval Conf. (ISMIR)*, 2009, pp. 321–326.

[6] P. Herrera-Boyer, A. Klapuri, and M. Davy, *Signal Processing Methods for Music Transcription*. New York: Springer, 2006, ch. 6, pp. 163–200.

[7] A. Holzapfel and Y. Stylianou, "Musical genre classification using nonnegative matrix factorization-based features," *IEEE Trans. Audio, Speech, Language Processing*, vol. 16, no. 2, pp. 424–434, Feb. 2008.

[8] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org

[9] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Adv. Neural Information Processing Syst.*, vol. 13, Denver, 2001, pp. 556–562.

[10] R. Lyon and S. Shamma, "Auditory representations of timbre and pitch," in *Auditory Computation*, H. L. Hawkins, Ed. Springer, 1996, ch. 6, pp. 221–270.

[11] N. Mesgarani, M. Slaney, and S. A. Shamma, "Discrimination of speech from nonspeech based on multiscale spectro-temporal modulations," *IEEE Trans. Audio, Speech, Language Processing*, vol. 14, no. 3, pp. 920–930, May 2006.

[12] "Freesound Project," Music Technology Group, Univ. Pompeu Fabra. [Online]. Available: http://www.freesound.org

[13] "Free Sound Samples – OLPC," One Laptop per Child. [Online]. Available: http://wiki.laptop.org/go/Sound_samples

[14] F. Opolko and J. Wapnick, "McGill University Master Samples," McGill Univ., 1987.

[15] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Proc. IEEE Workshop on Appl. Signal Processing to Audio and Acoustics*, New Paltz, NY, Oct. 2003, pp. 177–180.

[16] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, Mar. 2007.

YAAFE, AN EASY TO USE AND EFFICIENT AUDIO FEATURE EXTRACTION SOFTWARE

Benoit Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, Gaël Richard
Institut Telecom, Telecom ParisTech, CNRS/LTCI
firstname.lastname@telecom-paristech.fr

ABSTRACT

Music Information Retrieval systems are commonly built on a feature extraction stage. For applications involving automatic classification (e.g. speech/music discrimination, music genre or mood recognition, ...), traditional approaches will consider a large set of audio features to be extracted on a large dataset. In some cases, this will lead to computationally intensive systems and there is, therefore, a strong need for efficient feature extraction.

In this paper, a new audio feature extraction software, YAAFE [1], is presented and compared to widely used libraries. The main advantage of YAAFE is a significantly lower complexity due to the appropriate exploitation of redundancy in the feature calculation. YAAFE remains easy to configure and each feature can be parameterized independently. Finally, the YAAFE framework and most of its core feature library are released in source code under the GNU Lesser General Public License.

1. INTRODUCTION AND RELATED WORK

Most Musical Information Retrieval (MIR) systems include an initial low-level or mid-level audio feature extraction stage. For applications involving automatic classification (e.g. speech/music discrimination, music genre or mood recognition,...), traditional approaches consider a large set of audio features to be extracted on a large dataset, possibly combined with early temporal integration [2]. The importance of the feature extraction stage therefore justifies the increasing effort of the community in this domain and a number of initiatives related to audio features extraction have emerged in the last ten years, with various objectives.

For example, Marsyas is a software framework for audio processing [1], written in C++. It is designed as a dataflow processing framework, with the advantage of efficiency and low memory usage. Various building blocks are available to build real-time applications for audio analysis, synthesis, segmentation, and classification. Marsyas is widely and successfully used for various tasks.

Note however, that the audio feature extraction (*bextract* program) is only a small component of the whole Marsyas's framework. Extracted features are written in ARFF format, and can be directly reused with the WEKA [6] machine learning toolkit. Some classic features are available out-of-the-box. The user can select which features to extract, but parameters like frame size and overlaps are global. The user also has low control upon temporal integration.

VAMP Plugins [3] is the specification of a C++ Application Programming Interface (API) for plugins allowing extraction of low level features on audio signals. The very permissive BSD-style license permits the user to develop his own plugin or application that uses existing plugins. Several plugin libraries have been developed by various research labs. VAMP Plugins comes with the Sonic Visualizer [2] application, a tool for viewing contents of music audio files together with extracted features.

Batch feature extraction using VAMP Plugins can be done with the command line tool Sonic annotator [4]. Users can declare features to extract in RDF files [5] with precise control over each feature parameter. Output can be written to CSV [6] or RDF files. Early temporal integration is limited to predefined segment summaries, and it is not possible to perform temporal integration over overlapping texture windows. VAMP Plugins API allows the development of independent libraries, but prevents the development of new plugins that would depend on already existing plugins.

Another example, the MIR toolbox, is a Matlab toolbox dedicated to musical feature extraction [3]. Algorithms are decomposed into stages, that the user can parameterize. Functions are provided with a simple and adaptive syntax. The MIR toolbox relies on the Matlab environment and therefore benefits from already existing toolboxes and built-in visualization capabilities, but suffers from memory management limitations.

Other projects also exist. jAudio [5] is a java-based audio feature extractor library, whose results are written in

[1] http://yaafe.sourceforge.net

[2] Temporal integration is the process of summarizing features values over a segment or a texture window by computing mean, standard deviation, and/or any relevant statistical function. The term *early* refers to an integration performed before the classification step.

[3] http://vamp-plugins.org/, *Queen Mary, University of London*.

[4] http://www.omras2.org/SonicAnnotator

[5] Resource Description Framework is a semantic web standard.

[6] Comma Separated Values

XML format. Maaate is a C++ toolkit that has been developed to analyze audio in the compressed frequency domain. FEAPI [4] is a plugin API similar to VAMP. MPEG7 also provides Matlab and C codes for feature extraction. Lately, MIR web services have surfaced. For instance, the Echo Nest [7] provides a web service API for audio feature extraction. Input files are submitted through the web, and the user receives a XML description.

Whatever the objectives are, the computational efficiency of the feature extraction process remains of utmost interest. It is also clear that many features share common intermediate representations, such as spectrum magnitude, signal envelope and constant-Q transform. As already observed for the VAMP plugins with the Fast Fourier Transform (FFT), performances can be drastically improved if those representations are computed only once and this especially when large feature sets are extracted. Note also that this philosophy can be extended to the different transformations (such as derivatives) of a given feature.

YAAFE has therefore been created both to get the best of the previous tools and to address their main limitations in situations where a large feature set needs to be extracted from large audio collections with different parameterizations. In particular, YAAFE has been designed with the following requirements in mind:

- Computational efficiency with an appropriate exploitation of feature calculation redundancies.
- Usage simplicity with a particular attention to the feature declaration syntax.
- Capability to process very long audio files.
- Storage efficiency and simplicity.

The paper is organized as follows: the architecture of YAAFE is detailed in section 2. A detailed benchmark is then proposed in section 3. Finally, we suggest some conclusions and future work in section 4.

2. YAAFE

2.1 Overview

YAAFE is a command line program. Figure 1 describes how YAAFE handles feature extraction. The user has to provide the audio files and a feature extraction plan. The feature extraction plan is a text file where the user declares the features to extract, their parameters and transformations (see section 2.2).

To take advantage of feature computation redundancy, YAAFE proceeds in two main stages. In a first stage, a parser analyzes the feature extraction plan in order to find common computational steps (implemented in C++ components), and a reduced dataflow graph is produced. Then in a second stage, feature extraction is applied to the given audio files according to the reduced dataflow graph and results are stored in HDF5 files (see section 2.6).

[7] `http://echonest.com/`

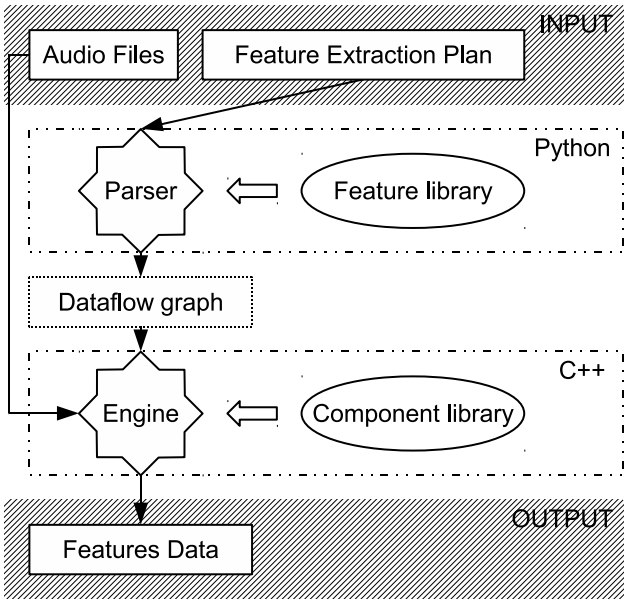


Figure 1. YAAFE internals overview.

Python is preferred to C++ for implementing the feature library and the parser, because the Python object model and reflection allow more concise and readable code to be written. The dataflow engine and the component library have been developed in the C++ language for performance.

YAAFE can be extended. Anyone can create their own extension which consists of a feature library and a component library. Provided extensions are loaded at runtime.

2.2 Feature extraction plan

2.2.1 Features

YAAFE feature extraction plan is a text file that describes the features to extract. Each line defines one feature, with the following syntax:

```
name: Feature param=value param=value
```

An example:

```
m: MFCC blockSize=1024 stepSize=512
z: ZCR blockSize=1024 stepSize=512
l: LPC LPCNbCoeffs=10
ss: SpectralSlope
```

The example above will produce 4 output datasets (see section 2.6) named *m*, *z*, *l* and *ss*, which will hold features MFCC [8], ZCR [9], LPC [10], SpectralSlope with given parameters. Missing parameters are automatically set to a predefined default value.

2.2.2 Transformations and temporal integration

One can also use spatial or temporal feature transforms, such as Derivate [11], StatisticalIntegrator [12], or SlopeInte-

[8] Mel-Frequency Cepstral Coefficients
[9] Zero Crossing Rate
[10] Linear Prediction Coefficients
[11] Derivate computes first and/or second derivatives.
[12] StatisticalIntegrator computes mean and standard deviation over the given frames.

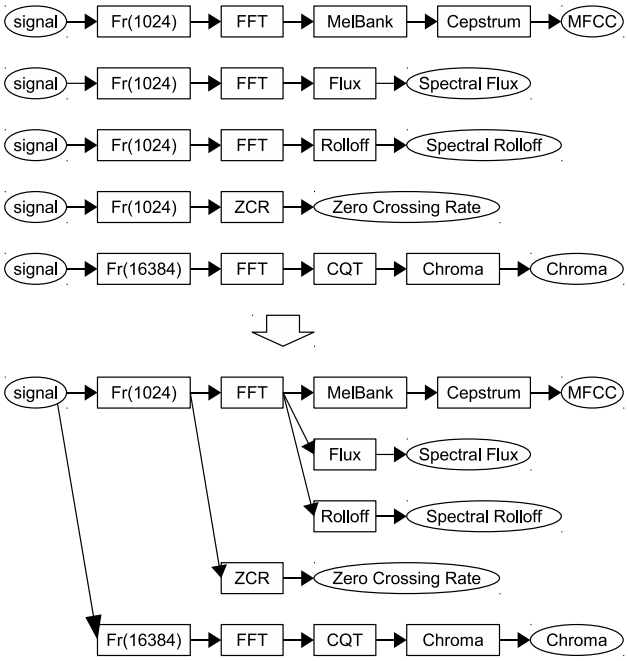


Figure 2. Automatic redundancy removal performed when parsing feature extraction plan. Fr(N) boxes are decompositions into analysis frames of size N. Step size is omitted but assumed equal.

grator [13] to enrich his feature extraction plan. For example, a plan to extract MFCCs along with derivatives and perform early integration over 60 frames will look like this:

```
m: MFCC > StatisticalIntegrator NbFrames=60
m1: MFCC > Derivate DOrder=1 ...
    > StatisticalIntegrator NbFrames=60
m2: MFCC > Derivate DOrder=2 ...
    > StatisticalIntegrator NbFrames=60
```

Obviously, *m, m1, m2* are all based on *MFCC* computation which should be computed only once. This is discussed in the next section.

2.3 Feature plan parser

Within YAAFE, each feature is defined as a sequence of computational steps. For example, MFCC is the succession of steps: Frames, FFT, MelFilterBank, Cepstrum. The same applies to feature transforms and temporal integrators.

As shown in Figure 2, the feature plan parser decomposes each declared feature into steps and groups together identical steps which have the same input into a reduced directed graph of computational steps.

The reduced graph can be dumped into a *dot* file, so an advanced user can discern how the features are really computed.

2.4 Dataflow engine

Each computational step is implemented in a C++ component which performs computation on a data block. Specific components manage audio file reading and output file writing.

The *dataflow engine* loads components, links them according to the given dataflow graph, and manages computations and data blocks. Reading, computations and writing is done block by block, so that arbitrarily long files can be processed with a low memory occupation.

[13] SlopeIntegrator computes the slope over the given frames.

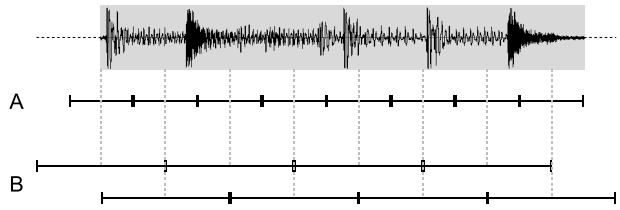


Figure 3. Temporally aligned frame decomposition for different frame sizes A and B, with same step sizes.

2.5 Feature timestamps alignment

In a feature extraction plan, each feature may have its own analysis frame size and step size. Some features require longer analysis frame sizes than others. As we intended to use YAAFE as input for classification systems, we have ensured that extracted features are temporally aligned. This is especially important with operations like the Constant-Q Transform (CQT) that may have very large analysis frames.

YAAFE addresses this issue as follows. We assume that when a feature is computed over an analysis frame, the resulting value corresponds to the time of the analysis frame center. Then, beginning with a frame centered on the signal start (left padded with zeros) ensures that all features with the same step size will be temporally aligned (see Figure 3).

A feature may also have an intrinsic time-delay. For example, when applying a derivative filter, we want the output value to be aligned with the center of the derivative filter. The design of YAAFE ensures that this is handled properly and that output features will be temporally aligned.

YAAFE only deals with equidistantly sampled features. However, some features like onsets have a natural representation which is event-based. In the current version, event-based features are represented as equidistantly sampled features for which the first dimension is a boolean value denoting the presence of an event.

2.6 Output format

YAAFE outputs results in HDF5 files [14] . Other output formats will be added in the future. The choice of the HDF5 format has initially been motivated by storage size and I/O performance. HDF5 allows for on-the-fly compression. Results are stored as double precision floating point numbers hence with no precision loss.

HDF5 is a binary format designed for efficient storage of large amounts of scientific data. HDF5 files can be read

[14] Hierarchical Data Format, `http://www.hdfgroup.org/HDF5/`

in the Matlab environment through built-in functions [15], and in the Python environment with the h5py package [16]. HDF5 files are platform independent, so they can be easily shared.

A HDF5 file can hold several datasets organized into a hierarchical structure. A dataset can be a table with several columns (or fields) of different data types, or simply a 2-D matrix of a specific data type. Attributes can be attached to datasets, an attribute has a name and a value of any data type.

YAAFE creates one HDF5 file for each input audio file. For each feature declared in the feature extraction plan, one dataset is created, with some attributes attached such as the feature definition, the frame size, the step size and the sample rate.

2.7 Availability and License

The YAAFE framework and a core feature library are released together under the GNU Lesser General Public License, so that it can freely be reused as a component of a bigger system. The core feature library contains several spectral features, Mel-Frequencies Cepstrum Coefficients, Loudness, Autocorrelation, Linear Prediction Coefficients, Octave Band Signal Intensities, OBSI ratios, amplitude modulation (tremolo and graininess description), complex domain onset detection [7], Zero Crossing Rate. Derivative and Cepstral transforms as well as statistical, slope and histogram early integrators are also provided. YAAFE is available for Linux platforms, source code can be downloaded [17].

A separate feature library will be available in binary version and for non commercial use only. It will provide Constant-Q Transform, Chromas [8], Chord detection [9], Onset detection [10], Beat histogram summary [11]. An implementation of CQT with normalization and kernels temporal synchronicity improvements [12] from reference implementation [18] is proposed.

3. BENCHMARK

We have run a small benchmark to compare YAAFE with Marsyas' bextract and Sonic Annotator. The objective is to compare the design of the three system, and not the algorithms used to compute feature. We chose few similar and well-defined features, available for the three systems for which we compared CPU time, memory occupation and output size when extracting those features on the same audio collection.

3.1 Protocol

We chose to extract the following features: MFCC (13 coefficients), spectral centroid, spectral rolloff, spectral crest factor, spectral flatness, and zero crossing rate. Features like chroma or beat detection have been avoided because the associated algorithms can be very different. In the case of Sonic Annotator, all features are available in the *Vamp libxtract plugins* [19] [13]. Early temporal integration is not computed.

We ran the feature extraction over about 40 hours of 32 KHz mono wav files (8.7 Gbs). The collection is composed of 80 radio excerpts of about 30mn each. The measures have been done on a Intel Core 2 Duo 3GHz machine, with 4 Gbs of RAM, under the Debian Lenny operating system. We checked that all systems used one core only. The RAM used has been measured with the `ps_mem.py` script [20].

We first ran the benchmark measuring the extraction of all features simultaneously. Then we ran the benchmark a second time measuring the extraction of each feature independently.

	S.A.	Marsyas	YAAFE
CPU time	52m05s	24m21s	6m34s
RAM used	14.0 Mbs	10.6 Mbs	15.5 Mbs
Output format	CSV	ARFF	HDF5
Output size	1.74 Gbs	2.7 Gbs	1.22 Gbs
Feature dim.	16	16 (32)	19

Table 1. Feature extraction with Sonic Annotator with VAMP libxtract plugins, Marsyas's bextract and YAAFE. All features are extracted simultaneously. Audio collection is 40 hours of 32 KHz mono wav files.

Feature	S.A.	Marsyas	YAAFE
MFCC	25m06s	19m28s	2m22s
Centroid	12m04s	15m42s	3m55s
Rolloff	12m11s	15m51s	3m14s
ZCR	3m41s	10m20s	0m57s
Total	53m02s	61m21s	10m28s

Table 2. CPU times for single feature extraction on the same collection as Table 1.

3.2 Results

The results are described in Table 1 and Table 2. It is important to note some differences between the 3 systems that influence the results. Firstly, we could not prevent Marsyas from performing temporal integration, so we reduced integration length to 1. Consequently, the output generated by Marsyas has 32 columns: 16 columns of feature data (mean) and 16 columns of zeros (standard deviation). This explains why Marsyas has a larger output size. Secondly, YAAFE extracts spectral spread, skewness and kurtosis together with the spectral centroid. This explains why output feature dimension is 19 for YAAFE and 16 for other systems.

Due to those differences the measures must be taken with caution. We can say that all systems performed well.

[15] See the *hdf5info*, *hdf5read* and *hdf5write* functions. YAAFE also provide useful scripts to directly load feature data into a matrix.

[16] `http://code.google.com/p/h5py/`

[17] `http://yaafe.sourceforge.net`

[18] B.Blankertz, "The Constant Q Transform", http://wwwmath.uni-muenster.de/logik/Personen/blankertz/constQ/constQ.html

[19] The VAMP libxtract plugins rely on the libxtract library: `http://libxtract.sourceforge.net/`

[20] `http://www.pixelbeat.org/scripts/ps_mem.py`

	YAAFE
CPU time	11m15s
RAM used	30.3 Mbs
Output format	HDF5
Output size	0.64 Gbs
Feature dim.	288

Table 3. Large feature set extraction with YAAFE. Audio collection is 40 hours of 32 KHz mono wav files.

They all succeed at extracting features over audio files of 30 minutes length and with low memory occupation.

The sum of single extraction times in Table 2 compared to the extraction time in Table 1 shows that Sonic Annotator does not exploit computation redundancy. The VAMP plugin API allows for computing feature in the frequency domain, but this is not done by Vamp libxtract plugins. That explains why Sonic Annotator requires more CPU time than others.

Marsyas performance clearly suffers from writing 16 column of zeros. For the evaluated task, the CPU times in Table 1 show that YAAFE tends to be faster than Marsyas.

As Sonic Annotator stored the timestamp in each output files (one per feature), and half of Marsyas' output is additional zeros, we can say that Sonic Annotator and Marsyas outputs are roughly equivalent in space. This is not a surprise as both CSV and ARFF format are text formats. Using HDF5 format, YAAFE stores more feature data, with no precision loss, using less space.

3.3 Extracting many features

YAAFE is designed for extracting a large number of features simultaneously. To check how it performs in such situation we ran YAAFE a second time under the same conditions but with a larger feature extraction plan.

In this run, we extracted MFCCs, various spectral features, loudness, loudness-based sharpness and spread, and zero crossing rate. For each feature except zero crossing rate, we added first and second derivatives. Then we performed early temporal integration by computing mean and standard deviation over sliding windows of 1 second with a step of 0.5 second. The total output dimension is 288.

The results are presented in Table 3. It should be emphasized that temporal integration is done, so the output size is much smaller than in the previous run. As a larger feature set is extracted, the dataflow graph is larger and uses more RAM. The CPU time shows that YAAFE remains very efficient in this situation.

4. CONCLUSIONS AND FUTURE WORK

In this paper, a new audio feature extraction software, YAAFE, is introduced. YAAFE is especially efficient in situations where many features are simultaneously extracted over large audio collections. To achieve this, the feature computation redundancies are appropriately exploited in a two step extraction process. First, the feature extraction plan is analyzed, each feature is decomposed into computational steps and a reduced dataflow graph is produced. Then, a dataflow engine processes computations block by block over the given audio files.

YAAFE remains easy to use. The feature extraction plan is a text file where the user can declare features to extract, transformations and early temporal integration according to a very simple syntax. YAAFE has already been used in Quaero project internal evaluation campaigns for the music/speech discrimination and musical genre recognition tasks.

Future plans include the extension of the toolbox with additional high level features such as fundamental frequency estimator, melody detection and tempo estimator and the extension to alternative output formats.

5. ACKNOWLEDGMENT

This work was done as part of the Quaero Programme, funded by OSEO, French State agency for innovation.

6. REFERENCES

[1] G. Tzanetakis, and P. Cook: "MARSYAS: A framework for audio analysis," *Org. Sound*, Vol. 4, No. 3, pp. 169-175, 1999.

[2] C. Cannam, C. Landone, M. Sandler, and J. Bello: "The Sonic Visualiser: A Visualisation Platform For Semantic Descriptiors From Musical Signals," *Proceedings of International Conference on Musical Information Retrieval*, Victoria, Canada, 2006.

[3] O. Lartillot, and P. Toiviainen: "A MATLAB TOOLBOX FOR MUSICAL FEATURE EXTRACTION FROM AUDIO," *Proceedings of the International Conference on Digital Audio Effects (DAFx'07)*, 2007.

[4] A. Lerch, G. Eisenberg, and K. Tanghe: "FEAPI, A LOW LEVEL FEATURES EXTRACTION PLUGIN API," *Proceedings of the International Conference on Digital Audio Effects*, (DAFx'05), 2005.

[5] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle: "jAudio: A feature extraction library," *Proceedings of the International Conference on Music Information Retrieval*, pp. 600–603, 2005.

[6] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten: "The WEKA Data Mining Software: An Update", *SIGKDD Explorations*, Vol. 11, Issue 1, 2009

[7] C.Duxbury et al., "Complex domain onset detection for musical signals", *Proceedings of the International Conference on Digital Audio Effects*, (DAFx'03), 2003.

[8] J.P. Bello and J. Pickens: "A Robust Mid-level Representation for Harmonic Content in Music Signals.", *Proceedings of the 6th International Conference on Music Information Retrieval*, (ISMIR-05), 2005.

[9] L.Oudre, Y.Grenier, C.Fevotte: "TEMPLATE-BASED CHORD RECOGNITION : INFLUENCE OF THE CHORD TYPES", *Proceedings of the International Conference on Music Information Retrieval*, 2009

[10] M.Alonso, G.Richard. B.David: "EXTRACTING NOTE ONSETS FROM MUSICAL RECORDINGS", *International Conference on Multimedia and Expo* (IEEE-ICME'05), 2005.

[11] G. Tzanetakis, "Musical Genre Classification of Audio Signals", *IEEE Transactions on speech and audio processing*, vol. 10, No. 5, 2002.

[12] J.Prado, "Transformée à Q constant", *technical report 2010D004*, http://service.tsi.telecom-paristech.fr/cgi-bin/valipub_download.cgi?dId=185, Institut TELECOM, TELECOM ParisTech, CNRS LTCI, 2010.

[13] J. Bullock, "Libxtract: A lightweight library for audio feature extraction," in Proceedings of the International Computer Music Conference, 2007.

ON THE USE OF MICROBLOGGING POSTS FOR SIMILARITY ESTIMATION AND ARTIST LABELING

Markus Schedl
Department of Computational Perception
Johannes Kepler University
Linz, Austria
markus.schedl@jku.at

ABSTRACT

Microblogging services, such as *Twitter*, have risen enormously in popularity during the past years. Despite their popularity, such services have never been analyzed for MIR purposes, to the best of our knowledge. We hence present first investigations of the usability of music artist-related microblogging posts to perform *artist labeling* and *similarity estimation* tasks. To this end, we look into different text-based *indexing* models and *term weighting measures*. Two artist collections are used for evaluation, and the different methods are evaluated against data from *last.fm*. We show that microblogging posts are a valuable source for musical meta-data.

1. INTRODUCTION

With the emergence of blogging services, social networks, platforms to share user-generated content and corresponding tags, services for music recommendation and personalized Web radio, such as *last.fm* [12], and in general all services and platforms commonly summarized by the term "Web 2.0", a new era of Web-based user interaction has started. The term "Web 2.0" was coined in 1999 by DiNucci [5], but did not become popular until 2004, when O'Reilly launched the first Web 2.0 conference [19].

Microblogging is one of the more recent phenomena in the context of the "Web 2.0". Microblogging services offer their users a means of communicating to the world in real time what is currently important for them. Such services had their origin in 2005, but gained greater popularity not before the years 2007 and 2008 [28]. Today's most popular microblogging service is *Twitter* [30], where millions of users post what they are currently doing or what is currently important for them. [9]

Despite the enormous rise in usage of microblogging services, to the best of our knowledge, they have not been used for music information extraction and retrieval yet. Hence, in this paper we present first steps towards assessing microblogging posts for the MIR tasks of *music artist labeling* and *similarity measurement*. We will show that even though such data is noisy and rather sparse, results comparable to other text-based approaches can be achieved.

The remainder of the paper presents and discusses related literature (Section 2), elaborates on the methods employed for similarity measurement and artist labeling (Section 3), gives details on the conducted evaluation experiments and discusses their results (Section 4), and finally summarizes the work (Section 5).

2. RELATED WORK

As this work is strongly related to text-based music information extraction and to Web content mining, we are going to review related work on these topics in the context of MIR. The past five years have seen the emergence of various text-based strategies to address MIR tasks, such as automated labeling, categorizing artists according to a given taxonomy, or determining similarities between tracks or artists.

Early work on text-based MIR focused on extracting information from artist-related *Web pages*. Cohen and Fan [4] query search engines to gather music-related Web pages, parse their DOM trees, extract the plain text content, and distill lists of artist names. Similarities based on co-occurrences of artist names are then used for artist recommendation. Web pages as data source for MIR tasks are also used in [7, 32], where the authors rely on a search engine's results to artist-specific queries to determine artist-related Web pages. From these pages, weighted term profiles, based on specific term sets (e.g., adjectives, unigrams, noun phrases), are created and used for classification and recommendation. Baumann and Hummel [3] extend this work by introducing certain filters to prune the set of retrieved Web pages, aiming at suppressing noisy pages. Another extension is presented in [10] for similarity measurement and genre classification. Knees et al. do not use specific term sets, but create a term list directly from the retrieved Web pages and use the χ^2-test for term selection, i.e., to filter out terms that are less important to describe certain genres. Other Web-based MIR approaches use page count estimates returned by search engines. For example, in [8, 26] co-occurrences of artist names and terms specific to the music domain, as returned by search engine's page count estimates, are used to categorize artists.

Another category of Web-based approaches to derive artist similarity exploits *user-generated playlists*. For example, in [2] Baccigalupo et al. analyze co-occurrences of artists in playlists shared by members of a Web commu-

Term Set	Cardinality	Description
`all_terms`	681,334	All terms that occur in the corpus of the retrieved *Twitter* posts.
`artist_names`	224	Names of the artists for which data was retrieved.
`dictionary`	1,398	Manually created dictionary of musically relevant terms.
`last.fm_toptags_overall`	250	Overall top-ranked tags returned by *last.fm*'s `Tags.getTopTags` function.
`last.fm_toptags_collection`	5,932	Aggregated top-ranked tags retrieved from *last.fm* for all artists in the collection.
`last.fm_toptags_topartists`	12,499	Aggregated top-ranked tags retrieved from *last.fm* for *last.fm*'s 2,000 top-played artists.

Table 1. List of the term sets used to index the *Twitter* posts. The cardinalities of term sets `all_terms`, `artist_names`, and `last.fm_toptags_collection` are based on the collection *C224a*.

nity. More than one million playlists made publicly available via *MusicStrands* [18] (no longer in operation) were gathered. The authors not only consider the co-occurrence of two artists in a playlist as an indication for their similarity, but also take into account that two artists that consecutively occur in a playlist are probably more similar than two artists that occur farther away from each other.

A recent approach derives similarity information from the *Gnutella* [22] *P2P file sharing network*. Shavitt and Weinsberg [27] collected metadata of shared music files from more than 1.2 million *Gnutella* users.The authors use this data for artist recommendation and song clustering, giving special emphasis to adjusting for the popularity bias.

Another data source related to the "Web 2.0" is *social tags*. [11] gives a good overview of their use in MIR. In [15] a semantic space is built, based on social tags extracted from *last.fm* and *MusicStrands*. The authors use this data for categorizing tracks into mood categories and present a user interface to browse a music collection according to mood. As an alternative to retrieving social tags from music information systems, tags may also be gathered via games designed to encourage their players to assign meaningful descriptions to a music piece [14, 17, 29]. Due to their design, this method can effectively reduce noise.

3. MINING TWITTER POSTS

To acquire user posts we queried *Twitter*'s Web API [31] in February and March 2010 with the names of the music artists under consideration. We downloaded up to 100 posts per query and extracted the plain text content. Earlier work on text-based music information retrieval [10, 26, 32] suggests to enrich the artist names with additional keywords, such as "music review" or "music genre style", to guide the retrieval process towards sources that contain information on music. However, preliminary classification experiments with various additional music-related keywords revealed that this strategy does not work well for *Twitter* posts. Restricting the search with any keyword in addition to the artist name in fact decreases the number of available user posts so strongly that even for the popular artists in our test collection *C224a* (cf. Subsection 4.1) the resulting feature vectors become very sparse.

After having downloaded the *Twitter* posts for each artist, we built an *inverted word-level index* [34] based on a modified version of the *lucene* [16] indexer. To investigate the influence of the term set used for indexing, we built various indexes using the term sets depicted in Table 1. The table further gives the term sets' cardinality. In cases where this cardinality depends on the size of the corpus, the values are based on collection *C224a* (cf. Subsection 4.1). The list denoted as `dictionary` consists of terms that we manually collected from various sources and somehow relate to music. This list resembles the one used in [21] and [24]. Included terms represent, for example, musical genres and styles, locations, instruments, emotions, and epochs.

Term weighting is performed using variants of the *term frequency* (tf) measure and the *term frequency · inverse document frequency* ($tf \cdot idf$) measure [33]. The term frequency $tf_{t,a}$ is the total number of occurrences of term t in all *Twitter* posts retrieved for artist a. The $tf \cdot idf_{t,a}$ function is defined as follows, where n is the total number of artists and df_t is the number of artists whose retrieved posts contain t at least once:

$$tf \cdot idf_{t,a} = \ln(1 + tf_{t,a}) \cdot \ln\left(1 + \frac{n}{df_t}\right) \qquad (1)$$

The basic idea of the $tf \cdot idf_{t,a}$ measure is to increase the weight of t if t occurs frequently in the posts retrieved for a, and decrease t's weight if t occurs in a large number of posts retrieved for *different* artists and is thus not very discriminative for a.
Since we are not interested in individual *Twitter* posts, but rather in a document describing a certain music artist, we aggregate all posts retrieved for an artist into a virtual document, based on which the term weights are calculated.

3.1 Similarity Estimation

Based on the term weighting vectors, we derive similarity between artists by applying the *cosine similarity measure* [23]. The cosine measure normalizes the data in that it accounts for different document lengths. To this end, only the angle between the weight vectors in the feature space is considered. In our case, the virtual documents for two artists a and b may be of very different length (depending on the number and length of the corresponding posts), which is likely to distort the weighting. [1] Therefore, we apply the cosine similarity measure between the $tf \cdot idf$ vectors of each pair of artists (a, b) according to Formula 2, where $|T|$ is the cardinality of the term set, i.e., the dimensionality of the term weight vectors. θ gives the

[1] The fact that usually much more data is available for popular artists than for lesser known ones, and the resulting likely distortion of results, is commonly referred to as "popularity bias".

angle between a's and b's feature vectors in the Euclidean space.

$$sim(a,b) = \cos\theta = \left(\frac{\sum_{t=1}^{|T|} tf \cdot idf_{t,a} \cdot tf \cdot idf_{t,b}}{\sqrt{\sum_{t=1}^{|T|} tf \cdot idf_{t,a}^2} \cdot \sqrt{\sum_{t=1}^{|T|} tf \cdot idf_{t,b}^2}} \right) \quad (2)$$

3.2 Labeling

A good similarity estimation function is crucial for many application areas of MIR techniques, for example, to build recommender systems, to generate intelligent user interfaces via clustering, or for automated playlist generation. Another related MIR task is automatically assigning labels/descriptors to an artist or a song. This allows to perform categorization of artists or songs into certain classes, for example, mood categories or a genre taxonomy.
We were interested in analyzing the potential of user-generated *Twitter* posts to perform automated categorization or labeling of music artists, also known as "autotagging" [6]. To this end, we compiled a list of *last.fm*'s top tags for the top artists (56,396 unique terms) and subsequently indexed the *Twitter* posts, taking this list as dictionary for our modified *lucene* indexer. Employing either the tf or the $tf \cdot idf$ measure, we used the top-ranked terms of each artist to generate labels.

4. EVALUATION

4.1 Test Collections

To compare the results of the proposed approaches to existing methods, we first ran evaluation experiments on the collection presented in [10]. It comprises 224 well-known artists, uniformly distributed across 14 genres. We will denote this collection as *C224a* in the following.

Since we further aim at evaluating the approaches on a real-world collection, we retrieved the most popular artists as of the end of February 2010 from *last.fm*. To this end, we used *last.fm*'s Web API [13] to gather the most popular artists for each country of the world, which we then aggregated into a single list of 201,135 artist names. Since *last.fm*'s data is prone to misspellings or other mistakes due to their collaborative, user-generated knowledge base, we cleaned the data set by matching each artist name with the database of the expert-based music information system *allmusic.com* [1]. Starting this matching process from the most popular artist found by *last.fm*, and including only artist names that also occur in *allmusic.com*, we eventually are given a list of 3,000 artists. We will denote this collection, which is used for artist labeling, as *3000a*.

4.2 Similarity Estimation

While the authors are well aware of the fact that "genre" is an ill-defined concept and that genre taxonomies tend to be highly inconsistent [20], we unfortunately do not have access to reliable and comprehensive similarity data, against which we could perform comparison. We therefore opted for a genre classification task that serves as a proxy for similarity measurement. We employed a *k-nearest neighbor* classifier (leave-one-out), and we investigated classification accuracy for different values of k, different term sets used for indexing, and different term weighting measures (tf and $tf \cdot idf$). We ran the classification experiments on collection *C224a*, since this artist set is already well-established in the literature, and results are therefore easy to compare.

4.2.1 Results

Figure 1 shows a detailed illustration of the k-NN classification results for different term sets and term weighting measures, using collection *C224a*. In general, $tf \cdot idf$ works better for the task of similarity estimation than the single tf value. The best classification results achievable using $tf \cdot idf$ are 72.52% accuracy with `all_terms` and a 9-NN classifier and 72.38% accuracy with an 8-NN-classifier and `last.fm_toptags_collection`.
Interestingly, the tf-based predictors (which, in general, perform worse than the $tf \cdot idf$-based predictors), perform comparable to the best $tf \cdot idf$-based classifiers when using `artist_names` for indexing. This setting resembles the co-occurrence approach described in [25], where accuracies of 54% and 75% (depending on the query scheme) were achieved for collection *C224a*. Using tf-weighting, our approach achieves a maximum of 65.34% accuracy with a 5-NN classifier. The authors of [10] report accuracy values of up to 77% using a k-NN classifier and up to 85% using a *Support Vector Machine* (SVM).
As for the different term sets used for indexing, using all terms in the corpus of *Twitter* posts (term list `all_terms`) yields the best classification results, but is computationally most complex. Using `artist_names` for indexing does not significantly reduce classification accuracy, while remarkably decrease space and time complexity. The good performance of the `artist_names` set can be explained by many *Twitter* posts containing lists of currently listened or favored artists. Such data therefore reveals information on personal playlists.

To investigate which genres tend to be confused with which others, Figures 2 and 3 show confusion matrices of the two best performing approaches. Using `all_terms` (Figure 2), "Folk" artists are often confused with "Country" artists, "Alternative Rock/Indie" performers are frequently predicted to make "Metal" music, and "Rock 'n' Roll" is often predicted for artists performing "RnB/Soul". Using `last.fm_toptags_collection` (Figure 3), the most frequent confusions are "Electronic" artists predicted as "Rap" artists and "RnB/Soul" artists mistaken for "Rock 'n' Roll" artists.
While some confusions are easy to explain, for example, "Country" and "Folk" music is pretty close and in some taxonomies even considered one genre, others are likely only the result of users' preference relations instead of similarity relations. For example, the co-occurrence of two artists (one from genre "Electronic", the other from "Rap") in a user's post may not necessarily indicate that these artists are similar, but that they are similarly liked or played together by the user.

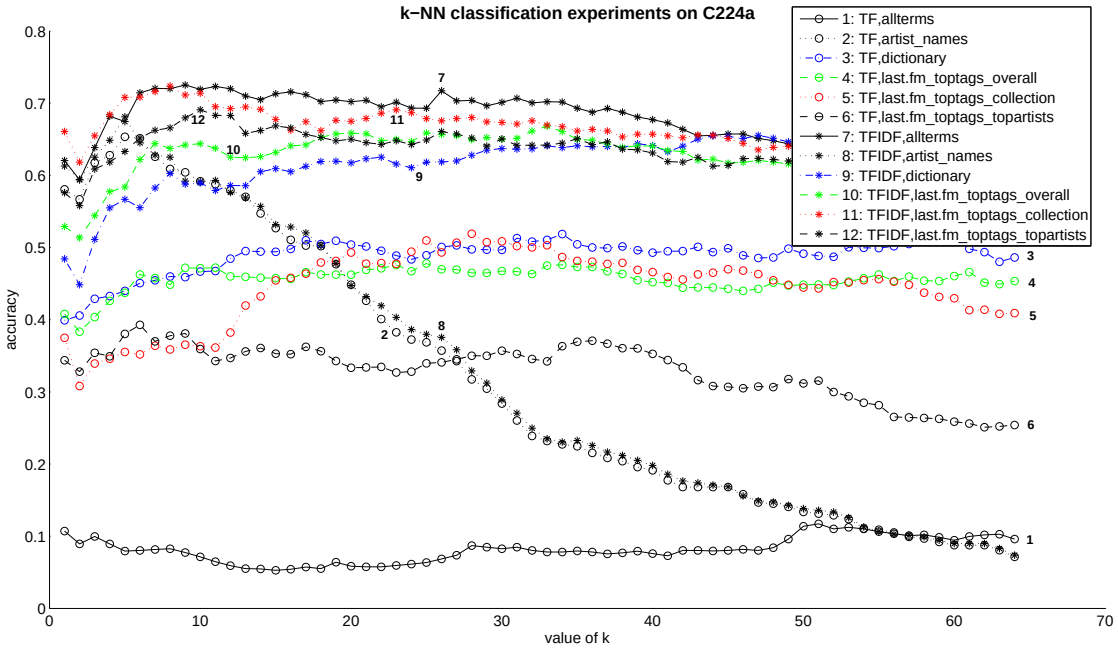


Figure 1. Results of the genre-classification-experiments for different k values of the k-NN classifier, using *C224a*.

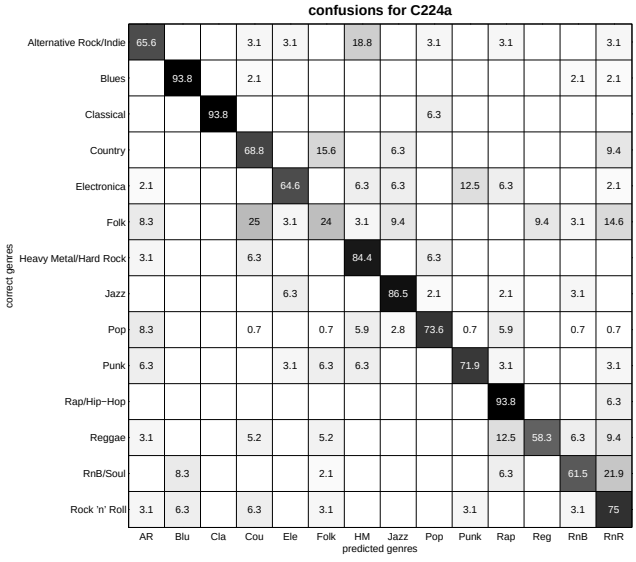


Figure 2. Confusion matrix for the 9-NN classifier on the *C224a* collection using the term list `all_terms`.

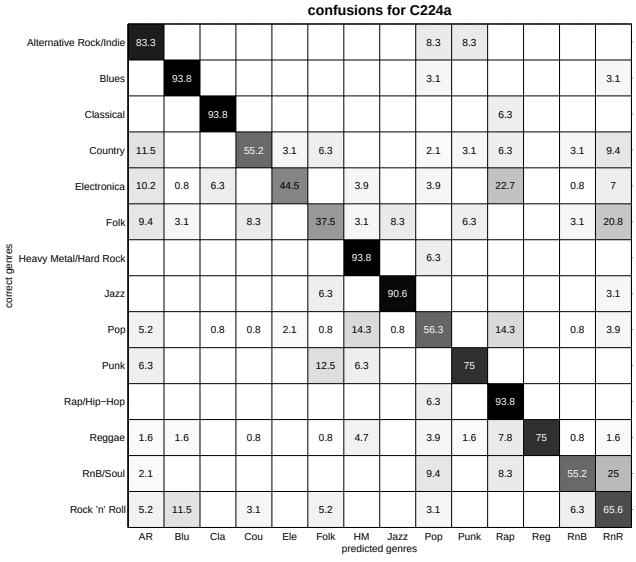


Figure 3. Confusion matrix for the 8-NN classifier on the *C224a* collection using the term list `last.fm_toptags_collection`.

4.3 Labeling

To assess the performance of *Twitter* posts for the task of labeling artists, we use an artist a's top-ranked terms (according to each term weighting measure), to predict labels for a. To this end, we index the posts using term list `last.fm_toptags_overall` and a list of tags extracted from *last.fm* for several thousands top-played artists. In total, 56,396 unique terms were obtained.

For evaluation we compare the top-ranked N labels from *Twitter* (according to the term weighting measure) with the top-ranked N tags from *last.fm*. To this end, we calculate an *overlap score* between the two term sets. Aggregating this score over all artists in the collection reveals the average percentage of overlapping terms, considering different quantities N of top-ranked terms. More formally, the *overlap@top-N* is calculated according to Formula 3, where A denotes the artist set, $\#artists_N$ is the number of artists with at least N terms assigned, and $overlap_{tw,fm,a,N}$ is the number of terms in *Twitter*'s set of top-N terms for

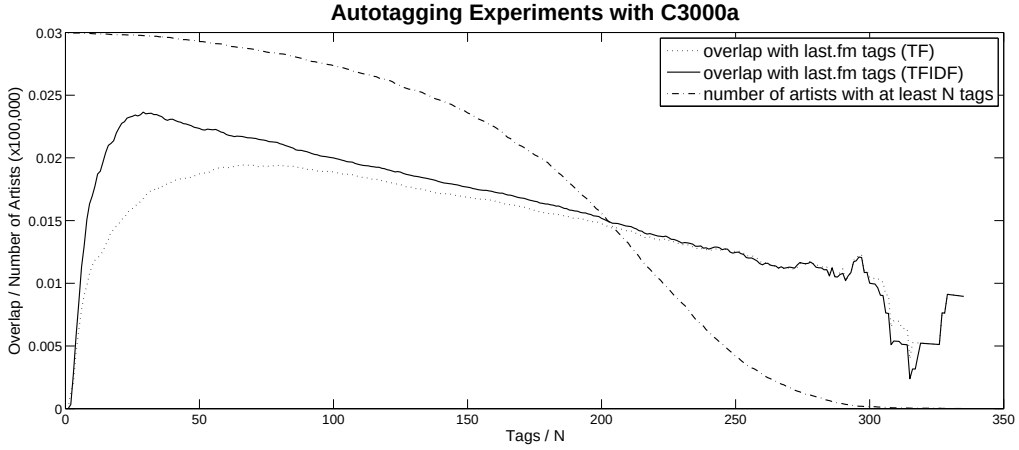


Figure 4. Results of the labeling experiments using *C3000a* and the set of 56,396 tags.

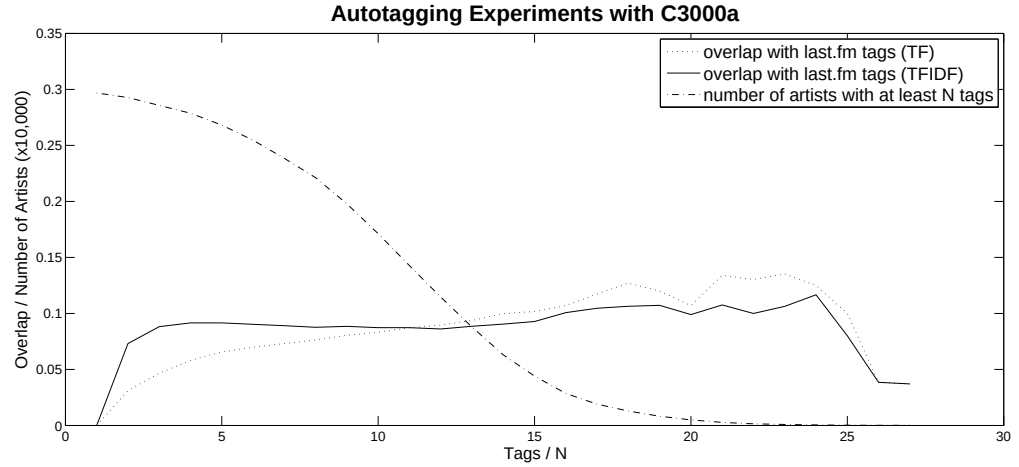


Figure 5. Results of the labeling experiments using *C3000a* and the term set `last.fm_toptags_overall`.

artist a that also occur in *last.fm*'s set of top-ranked tags for a.

$$overlap@top{-}N = \frac{\sum_{a \in A} \frac{overlap_{tw,fm,a,N}}{N}}{\#artists_N} \quad (3)$$

4.3.1 Results

Figures 4 and 5 show the aggregated overlap scores for collection *C3000a* at different levels of top-N terms/tags using the term set of 56,396 tags and the term set `last.fm_toptags_overall`, respectively. The dash-dotted line reveals the number of artists with at least N terms assigned. The solid line gives the overlap score using $tf \cdot idf$ for term weighting, whereas the dotted line gives the score using tf-weighting.

The low maximum overlap of 2.36% for the 56,396-tag-set ($tf \cdot idf$) is likely caused by a large amount of noise in the *last.fm* tags. Using `last.fm_toptags_overall`, the maximum overlap scores are 13.53% (tf) and 11.67% ($tf \cdot idf$). Taking into account that this is a very challenging task (an overlap of 100% for a certain level of N would mean that the top-N terms according to the *Twitter* posts correspond exactly to the top-N tags from *last.fm* for all artists), these results are better than the sole numbers suggest.

The corresponding maximum overlap scores for collection *C224a* using the 56,396-tag-set amount to 6.68% ($tf \cdot idf$) and 5.39% (tf). Term set `last.fm_toptags_overall` yields maximum overlap scores of 16.36% ($tf \cdot idf$) and 15.22% (tf).

5. CONCLUSIONS AND OUTLOOK

We have shown that *Twitter* posts provide a valuable data source for music information research. In particular for the task of similarity measurement on the artist level, classification results resemble the ones achieved with other text-based approaches using community or cultural data sources, e.g., [10, 25], on the same artist set. For the task of automated labeling, in contrast, only weak to medium overlaps between *Twitter* posts and *last.fm* tags could be determined.

As part of future work, we would like to analyze the localization capabilities of the *Twitter* API. Provided sufficient accuracy, additional geographic data could be used, for example, to spot the most popular artists within a region or country. Successively, such information may be used to reveal the spreading of listening trends around the

world. Using geolocation information may also help building country-specific or culture-specific models of music similarity.

6. ACKNOWLEDGMENTS

This research is supported by the *Austrian Fonds zur Förderung der Wissenschaftlichen Forschung* (FWF) under project numbers L511-N15 and Z159.

7. REFERENCES

[1] http://www.allmusic.com (access: January 2010).

[2] Claudio Baccigalupo, Enric Plaza, and Justin Donaldson. Uncovering Affinity of Artists to Multiple Genres from Social Behaviour Data. In *Proc. of ISMIR*, 2008.

[3] Stephan Baumann and Oliver Hummel. Using Cultural Metadata for Artist Recommendation. In *Proc. of WEDELMUSIC*, 2003.

[4] William W. Cohen and Wei Fan. Web-Collaborative Filtering: Recommending Music by Crawling The Web. *WWW9 / Computer Networks*, 33(1–6):685–698, 2000.

[5] Darcy DiNucci. Fragmented Future. *Design & New Media*, 53(4), 1999.

[6] Douglas Eck, Thierry Bertin-Mahieux, and Paul Lamere. Autotagging Music Using Supervised Machine Learning. In *Proc. of ISMIR*, 2007.

[7] Daniel P.W. Ellis, Brian Whitman, Adam Berenzweig, and Steve Lawrence. The Quest For Ground Truth in Musical Artist Similarity. In *Proc. of ISMIR*, 2002.

[8] Gijs Geleijnse and Jan Korst. Web-based Artist Categorization. In *Proc. of ISMIR*, 2006.

[9] Andy Kazeniac. Social Networks: Facebook Takes Over Top Spot, Twitter Climbs. http://blog.compete.com/2009/02/09/facebook-myspace-twitter-social-network (access: March 2010).

[10] Peter Knees, Elias Pampalk, and Gerhard Widmer. Artist Classification with Web-based Data. In *Proc. of ISMIR*, 2004.

[11] Paul Lamere. Social Tagging and Music Information Retrieval. *Journal of New Music Research: From Genres to Tags – Music Information Retrieval in the Age of Social Tagging*, 37(2):101–114, 2008.

[12] http://last.fm (access: March 2010).

[13] http://last.fm/api (access: March 2010).

[14] E. Law, L. von Ahn, R. Dannenberg, and M. Crawford. Tagatune: A Game for Music and Sound Annotation. In *Proc. of ISMIR*, Vienna, Austria, September 2007.

[15] Mark Levy and Mark Sandler. A semantic space for music derived from social tags. In *Proc. of ISMIR*, Vienna, Austria, September 2007.

[16] http://lucene.apache.org (access: February 2010).

[17] Michael I. Mandel and Daniel P.W. Ellis. A Web-based Game for Collecting Music Metadata. In *Proc. of ISMIR*, Vienna, Austria, September 2007.

[18] http://music.strands.com (access: November 2009).

[19] Tim O'Reilly. What Is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software. http://oreilly.com/web2/archive/what-is-web-20.html (access: March 2010).

[20] François Pachet and Daniel Cazaly. A Taxonomy of Musical Genre. In *Proc. of RIAO*, 2000.

[21] Elias Pampalk, Arthur Flexer, and Gerhard Widmer. Hierarchical Organization and Description of Music Collections at the Artist Level. In *Proc. of ECDL*, 2005.

[22] Matei Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network. In *Proc. of IEEE Peer-to-Peer Computing*, 2001.

[23] Gerard Salton. The Use of Citations as an Aid to Automatic Content Analysis. Technical Report ISR-2, Section III, Harvard Computation Laboratory, Cambridge, MA, USA, 1962.

[24] Markus Schedl and Peter Knees. Investigating Different Term Weighting Functions for Browsing Artist-Related Web Pages by Means of Term Co-Occurrences. In *Proc. of LSAS*, 2008.

[25] Markus Schedl, Peter Knees, and Gerhard Widmer. A Web-Based Approach to Assessing Artist Similarity using Co-Occurrences. In *Proc. of CBMI*, 2005.

[26] Markus Schedl, Tim Pohle, Peter Knees, and Gerhard Widmer. Assigning and Visualizing Music Genres by Web-based Co-Occurrence Analysis. In *Proc. of ISMIR*, 2006.

[27] Yuval Shavitt and Udi Weinsberg. Songs Clustering Using Peer-to-Peer Co-occurrences. In *Proc. of AdMIRe (IEEE ISM)*, 2009.

[28] http://www.sysomos.com/insidetwitter (access: March 2010).

[29] D. Turnbull, R. Liu, L. Barrington, and G. Lanckriet. A Game-based Approach for Collecting Semantic Annotations of Music. In *Proc. of ISMIR*, Vienna, Austria, September 2007.

[30] http://twitter.com (access: February 2010).

[31] http://apiwiki.twitter.com/Twitter-API-Documentation (access: February 2010).

[32] Brian Whitman and Steve Lawrence. Inferring Descriptions and Similarity for Music from Community Metadata. In *Proc. of ICMC*, 2002.

[33] Justin Zobel and Alistair Moffat. Exploring the Similarity Space. *ACM SIGIR Forum*, 32(1):18–34, 1998.

[34] Justin Zobel and Alistair Moffat. Inverted Files for Text Search Engines. *ACM Computing Surveys*, 38:1–56, 2006.

PARATAXIS: MORPHOLOGICAL SIMILARITY IN TRADITIONAL MUSIC

Andre Holzapfel
Institute of Computer Science, FORTH and University of Crete
hannover@csd.uoc.gr

Yannis Stylianou
Institute of Computer Science, FORTH and University of Crete
yannis@csd.uoc.gr

ABSTRACT

In this paper an automatic system for the detection of similar phrases in music of the Eastern Mediterranean is proposed. This music follows a specific structure, which is referred to as *parataxis*. The proposed system can be applied to audio signals of complex mixtures that contain the lead melody together with instrumental accompaniment. It is shown that including a lead melody estimation into a state-of-the-art system for cover song detection leads to promising results on a dataset of transcribed traditional dances from the island of Crete in Greece. Furthermore, a general framework that includes also rhythmic aspects is proposed. The proposed method represents a simple framework for the support of ethnomusicological studies on related forms of traditional music.

1. INTRODUCTION

In the field of ethnomusicology, computer based methods are adequate for simplifying musicological studies. Useful methods can be the recognition of intervals played by an instrument, or determining the meter structure of a signal. Using such methods, a search engine can be developed that can detect similarities between different pieces. Such a tool is valuable for research in ethnomusicology, because it enables to get a faster access to pieces that are interesting for a comparison. In this paper, a general framework for the morphological analysis of the Eastern Mediterranean traditional music is proposed and the parts related to melodic characteristics are presented and evaluated.

In general, morphology of music is defined as the methodical description of the structure of the form of musical works [12]. The elements of this organization are themes, phrases and motives, which themselves are made up of sound characteristics like tonal height, duration, intensity and timbre. The analysis aims at the discovery of the sentence structure (Periodenbau) and the transformative structure of these elements. This discovery is the core of morphological analysis.

Recently, the research presented in Sarris *et al.* [11] shed light on the difficulty of understanding traditional music in the eastern Mediterranean area: to a great extent, it is following a different kind of morphology, the logic of *parataxis*. The term *parataxis* stems from the field of linguistics, where it denotes a way of forming phrases using short sentences, without the use of coordinating or subordinating conjunctions [10]. In music following this logic, the tunes are built from small melodic phrases which do not follow a specific morphologic structure. This means, that there is no composed elaboration of a theme like for example in a fuga, neither there is a clear periodic structure, according to which a musical theme is repeated, like the repeating element of a chorus in popular western music. As mentioned in Theodosopoulou [13], it is a major effort to transcribe and analyze a big number of pieces. In this paper, the goal is to derive at least some conclusions about the content and similarity between pieces in an automatic way. Thus, a concept is presented that is aimed to discover recurring elements in a musical signal. These recurring elements are the melodic phrases that are the characteristic themes of the music following the logic of *parataxis*. The recognition of these phrases and their assignment to a specific dance appears to be a complex task even for a human being. In interviews the author conducted with local musicians, repeatedly the recognition of a dance was connected with the recognition of a specific melodic phrase. This process was also described in Tsouchlarakis [15]. Also, in listening test conducted *e.g.* in [6], it was observed that dancing teachers had memorized almost all melodies they have been presented with. With this knowledge they were able to conduct assignments to a class of dance much faster and with higher accuracy than their students. It is apparent that the similarity estimation between the used motifs is the key to a concept for a search engine for this music.

Recently, similarity in folk song and traditional melodies has drawn increasing attention of the Music Information Retrieval research community. Most of the related publications investigate symbolic transcriptions of melodies [7, 14,16]. For audio signals, Moelants *et al.* [9] and Bozkurt [1] derive pitch histograms from monophonic recordings, the former using African music and the latter in the context of Turkish music. Both methods are aimed towards the recognition of underlying tonal concepts (*i.e.* scales or *makams*, respectively), and stress the importance of a finer frequency solution than the one provided by the chroma

features. Cabrera *et al.* [2] investigate the estimation of melodic similarity on a set of mainly monophonic vocal Flamenco recordings.

In this paper, the goal is to estimate similarities between

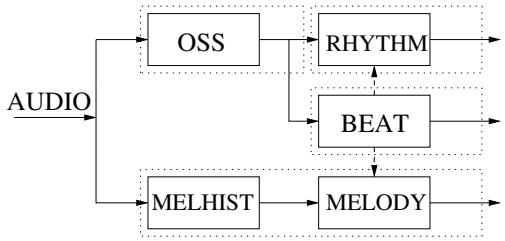


Figure 1. Block diagram of the proposed morphological analysis system

lead melodies in polyphonic mixtures. The focus lies upon the melodic aspects of the morphological analysis system depicted in Figure 1. This is achieved by the beat synchronous computation of melody histograms (MELHIST), as detailed in Section 4. The rhythmic aspects, such as the computation of onset strength signals (OSS), rhythmic similarity and beat tracking on this kind of music has been the subject of investigation in other publications [4, 6]. An integration of rhythmic and melodic similarity as depicted in Figure 1 for such a task is meaningful: It has been repeatedly confirmed by local musicians, that not only the melody is of importance for recognizing a specific dance, but also the way the instrument player puts emphasis on particular notes of the melody. However, the type of available data made it necessary to concentrate on the melodic aspect, as will be detailed in the following Section.

2. DATASET

A small dataset of polyphonic samples has been collected that enables for a preliminary evaluation of a system for the detection of morphological similarity. For this, samples from the Crinno data of the Institute of Mediterranean Studies [1] have been used. In the Crinno collection for some samples of the dance *Sousta* the lead melodies have been transcribed by musicologists and then analyzed for their morphology. All encountered phrases have been indexed, and using the list of these indexes it is feasible to locate the morphologically identical phrases in different pieces. The way to index the phrases follows the method described in Theodosopoulou [13]: the phrases have a length of either one or two bars as shown in Figures 2 and 3. Based on Theodosopoulou [13], during the analysis the first encountered two bar phrase will be titled `1a1b`. If, for example, the next encountered two bar phrase contains the second part of the first phrase in its second measure, while its first measure is an unknown phrase, then it will be titled `2a1b`, denoting the partial relation with the first pattern. In Figures 2 and 3 the titles of the depicted melodic phrases are denoted above the score. It is obvious that an exact partial or complete matching can be localized by using this way of indexing the phrases. However, no conclusions can be drawn about the similarity of phrases with different titles. As the amount of transcribed data is rather small

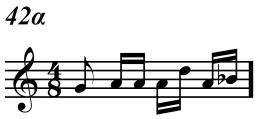


Figure 2. Example of a one measure melodic phrase

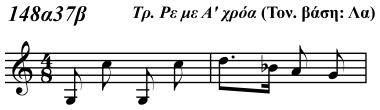


Figure 3. Example of a two measure melodic phrase

(20 pieces), there are not many phrases that appear several times in various pieces. It has been achieved to compile a data set of 40 sound samples, each containing a complex musical mixture signal with the instruments Cretan *laouto* and *lyra* and sometimes singing voice. Each sample contains several repetitions of melodic phrase of two measures length. Each of the 40 pieces has a "partner" within the dataset that contains a similar or equal musical phrase played by the *lyra*, according to the analysis of musicologists. Thus, in this dataset exist 20 pairs of samples that contain similar phrases. Please note that according to the musicological analysis these phrases are exactly the same. However, the audio files differ because they are performed by different artists and vary due to their different playing style.

3. MELODIC PATTERN SIMILARITY

For the computation of similarity, a baseline system as presented in Ellis and Poliner [3] will be used. This system uses beat synchronous chroma features to describe the melodic content. It was proposed for the detection of coversongs in western pop music, and it will serve as a starting point for the studies of detecting morphological similarity in traditional music. The first computational step in this approach is a beat tracking that uses a spectral flux like OSS as an input, and derives the beat time instances using dynamic programming. Then, for each beat time a 12-dimensional chroma feature is computed. These chroma features record the intensity associated with each of the 12 semi-tones of the equal-tempered tonal system. In order to determine, how well two songs match, the cross-correlations between two feature matrices are computed for each possible transposition. In the following, this system will be referred to as BASE-SYS.

As the sound files are complex mixtures, melodic similarity is degraded by the other instruments contained in the mixture, which play to some extent a similar accompaniment in all examples that is characteristic for this type of dance. Thus, a lead melody extraction using a method as the one proposed in Klapuri [8] could be included as

[1] http://gaia.ims.forth.gr/portal/

a pre-processing. Furthermore, instead of using chroma features, in the context of traditional music melodic histograms of a finer resolution have been found useful for the classification of melodic content [1]. In order to determine if such approaches can be adapted to the beat synchronous melody description framework, the lead melody will be estimated using the algorithm presented in Klapuri [8], which was provided by the author of the paper. The parameters given as input to the algorithm are the desired number of fundamental frequency tracks to be estimated from the signal (set to 1), and the fundamental frequency range of the desired F0 tracks. This range was set to $60Hz...480Hz$, after an analysis of the available scores of the recordings. The next step is the computation of beat synchronous melody histograms. Motivated by the work presented in Bozkurt [1], the frequency resolution of these histograms is set higher than necessary for music using scales of the equal-tempered system. This is because in Greek traditional music many modal scales are encountered which make use of tonal steps different from the half tone of the equal-tempered system. For example, some of these scales have their roots in the scales investigated in Bozkurt [1]. Scales like *Hidzaz* and *Kurdi* are examples for this case, and because these scales are also used in Cretan music the finer resolution of the histograms is theoretically justified. Thus, for a song a matrix is obtained with one column for a beat instance which contains the melody histogram for this beat. Again, for matching two samples the method proposed in Ellis and Poliner [3] has been used in the same way as for the chroma features. The system that uses this kind of melody histograms will be referred to as HIST-SYS.

In Ellis and Poliner [3], the features are computed beat synchronous. This means that a beat tracking is necessary as a pre-processing step. For this purpose, OSS derived from amplitude are used to perform the beat tracking [3]. However, results described in Holzapfel and Stylianou [5] indicate that for the investigated type of music a beat tracking based on phase characteristics gives more accurate results. Thus, it should be evaluated as well if the accuracy of the beat tracking has some impact on the results of the matching experiments.

4. EVALUATION METHODS

Two different evaluation methods are suggested. In the first one, only the 40 short samples containing the melodic phrases are used to compute their mutual similarity regarding melodic content. The quality of the obtained similarity measure can be evaluated using the Mean Reciprocal Rank (MRR)

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{Q}\frac{1}{rank_i} \quad (1)$$

where $|Q|$ is the number of queries. For our data set this means that each sample is used as a query once, *i.e.* $|Q| = 40$. If *e.g.* the correct partner is found on place 3 of the most similar samples, the reciprocal rank is $\frac{1}{3}$. This means that the closer the MRR is to the value 1, the better the similarity measurement.

In the second evaluation method, a sample from the dataset is used as a query and similarities are computed for the whole duration of the piece that contains its partner motif at some time instance. If this similarity measure shows a peak at the position of the true partner, the goal of locating it in a continuous piece is achieved.

5. EXPERIMENTS

5.1 Setup 1: Matching pairs

In the first experiment, the BASE-SYS system was applied to the data set of 40 song excerpts. Each song was used as a query and the mean reciprocal rank as defined in (1) was computed, which resulted in a value of $MRR_{BASE-SYS} = 0.38$, as shown in Table 1.

In the following we will show that the performance in terms of the mean reciprocal rank of the BASE-SYS system can be improved by involving an estimation of the main melody from the polyphonic samples and the usage of high resolution histograms in the HIST-SYS system. In Bozkurt [1], a resolution of one Holdrian comma (Hc) was referred to as the smallest interval considered in Turkish music theory, and the authors used a resolution of $\frac{1}{3}$ Hc for their histograms. One Holdrian comma is equal to 22.6415 cents, and the octave interval can be divided into 53 Hc or 1200 cents. Various resolutions have been tried, but no clear result regarding the optimum value could be obtained on the limited sized dataset. For that reason, the resolution was set to 2 Hc, or about 2.25 times higher than the resolution of equal-tempered scales (about 4.5 Hc). As it can be seen from the second row in Table 1, the obtained mean reciprocal rank (0.58) is improved compared to the BASE-SYS system. This improvement is present almost independently of the histogram resolution, which indicates that the sensitivity to microtonal changes is not of importance at least for the present dataset. We acknowledge, however, that bigger and more diverse datasets have to be obtained to achieve more insight into the parameter settings.

Table 1. Mean reciprocal rates (MRR)

BASE-SYS	0.38
HIST-SYS	0.58

5.2 Setup 2: Matching queries in whole songs

As described in Section 4, the second evaluation method is using one of the short samples contained in the dataset as a query. For this experiment 10 phrases of two measures length have been selected as depicted in the first column of Table 2. For example, the query file `13b42b:234` is the phrase 13*b*42*b* taken from the recording number 234 in the collection. The target file is the whole piece which contains the partner of the query at some time instance (*i.e.*

Table 2. Results of matching patterns from MS1 in whole song files

QUERY FILE	$max(R_{neg})$	R_{source}	$max(R_{pos})$	MATCH
(1) 13b42b:234	0.5796	0.9200	0.6403	EXACT
(2) 4a31b:217	0.3602	0.9301	0.6741	EXACT
(3) 3a3b:027	0.5059	0.9297	0.6238	CORRECT
(4) 35a35b:196	0.5482	0.9416	0.6866	CORRECT
(5) 3a21b:051	0.4511	0.8549	0.7040	EXACT
(6) 89a46b:143	0.4881	0.6571	0.5451	EXACT
(7) 31a31b:035	0.4830	0.8989	0.6351	WRONG
(8) 6a72a:167	0.5535	0.8778	0.6578	EXACT
(9) 7a6b:008	0.5073	0.8242	0.5870	EXACT
(10) 62a62b:249	0.4484	0.8333	0.5869	EXACT

a different interpretation of the same phrase). It has been tried to locate the appearance of the phrase in the target file using the HIST-SYS method, which lead to the best pattern matching results as shown in Table 1. The highest correlation measures in these files are depicted in the column titled $max(R_{pos})$ in Table 2. In the column titled MATCH the success of this matching is judged. If the position connected to this highest correlation measure is exactly the position where the partner file has been extracted from, then the label EXACT is assigned. If the position of the correlation maximum is related to another appearance of the same pattern in the file, it is labeled as CORRECT. Finally, when a different pattern from the query pattern is located at the position of the correlation peak, the label WRONG is assigned. This evaluation has been performed entirely by hand, by locating the time instance of the correlation maximum of the melody histogram in the related musical score. It can be seen that only in one case the matching gave a wrong result, while all the other 9 matches were related to an appearance of the same melodic phrase in the target file. Let us stress again that all the target files are different from the file that the query was taken from: The target files used in the column titled $max(R_{pos})$ are different recordings than the recordings which contain the query at some time instance. They have been recorded by different players, but they contain at one or more time instances a melodic phrase that has been judged to be identical with the query by an analysis conducted by musicologists.

The correlation between the F0 histogram of a query sample and the histogram of the whole recording it has been extracted from has been computed as well. This means that at some time instance exactly the same pattern is encountered, without the variation introduced by a different interpretation. This enables to determine how good the matching works in the perfect case, when the pattern we are looking for, is indeed contained in the file exactly as found in the query. The resulting correlations are depicted in the column entitled R_{source} in Table 2. It can be seen that the correlations shown in R_{source} are always larger than the correlation depicted in $max(R_{pos})$, but never equal to 1. This is likely to be caused by slightly differing beat tracking and F0 estimation results on the small query samples and on the whole file.

Furthermore, each query has been applied also to a file, where according to the annotation the phrase is not contained neither as a whole nor half of it. The correlation maxima are depicted in the column titled $max(R_{neg})$, and these values are always smaller than the correlation values computed in the other columns. This supports the assumption that the proposed method is able to separate similar phrases from those that do not share a large similarity with the query phrase.

In Figures 4 and 5, all R_{pos} vectors of the 10 queries shown in Table 2 are plotted. These vectors have been obtained by computing the two dimensional correlations between the query and the target histogram matrices, and the choosing the row (*i.e.* the tonal transposition between the files) in the correlation matrix, that contains the maximum value. In all plots, maxima have been chosen and it has be evaluated if at the related measures in the score indeed the query phrase is found. For these cases, maxima are shown with dashed boxes, while maxima which are not related to the query pattern have been marked with dotted boxes.

A first and important result of this analysis is that in none of the cases an occurrence of the query pattern in the investigated audio file has been missed, which means that in every case the occurrence of the pattern was related to a maximum in R_{pos}. Also the overall number of true positives (dashed boxes) is 21 while the number of false positives (dotted boxes) is only 7. However, these false positives do not imply that there is no similarity between the query and the target at the time instance of the false positive. The false positive only indicates that at this position the phrase played by the lead instrument does not have exactly the same label. Taking a closer look at the false positives reveals that for example all wrong detections for query (3) are phrases which contain the pattern 3a which is also contained in the query sample (3a3b). A closer look has been taken at the only case, where the maximum in R_{pos} is connected to a false positive (query (7)). The query phrase and the phrases found in the dotted boxes in Figure 5.(7) are depicted in Figure 6. It is apparent that at least the first parts of the two phrases share a big amount of similarity. Thus, at least in this case, the false positive is related to a similar melodic phrase.

Another observation from Figures 4 and 5 is that max-

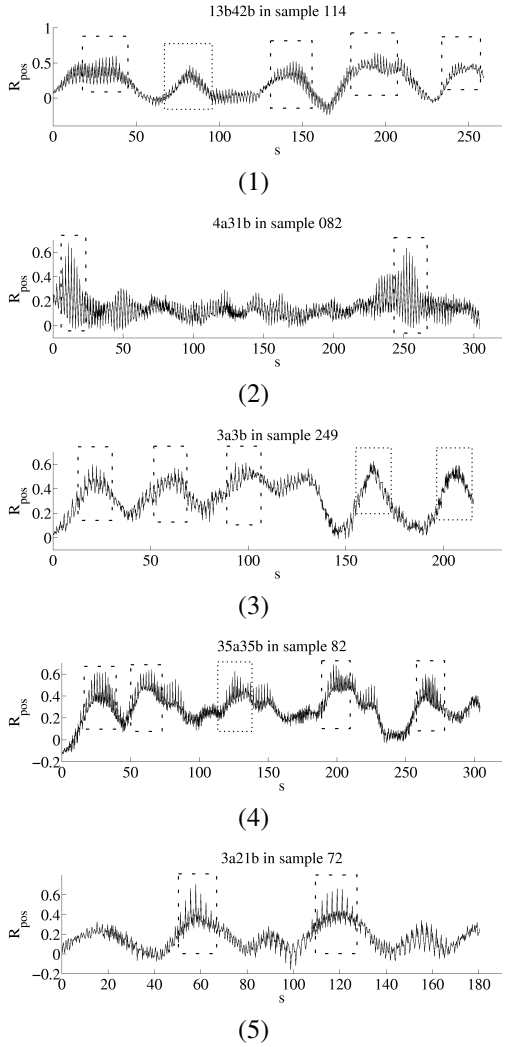


Figure 4. Complete R_{pos} obtained for queries 1-5 in Table 2, positive matches in dashed boxes, negative matches in dotted boxes

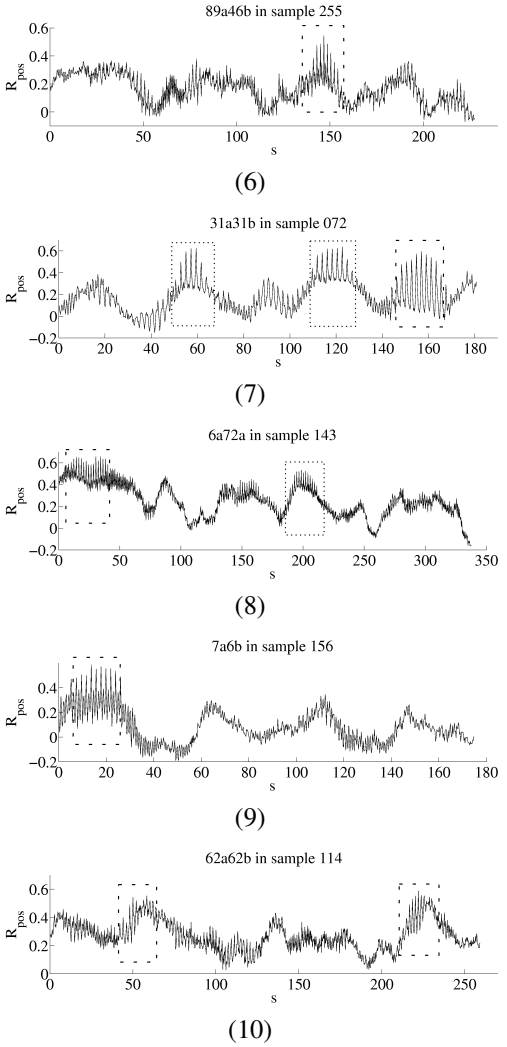


Figure 5. Complete R_{pos} obtained for queries 6-10 in Table 2, positive matches in dashed boxes, negative matches in dotted boxes

ima related to true positives seem to be characterized by a strong oscillation. This oscillation has been observed to have the frequency of exactly two measures. This means that the correlation shows a strong peak whenever the beginnings of the query phrase and the related phrase in the investigated file are aligned. This effect should be further investigated when a larger dataset is available, and it is possible that a detection of such oscillations, besides high correlation envelopes, further improves the result of the pattern retrieval.

Finally, the impact of the beat tracker has been evaluated. In order to determine how large the change in the matching procedures would be if the beat tracking and hence the synchronization is optimized, all samples in the dataset and all complete samples used for the computation of R_{pos} in Table 2 have been beat annotated by the author. However, rerunning all experiments in the experimental setups 1 and 2 using these ground truth beat annotations did not qualitatively change the results. Since the original beat tracker used in Ellis and Poliner [3] lead mainly to local misalignments with the beat annotation, and it has to be concluded that these misalignments have no impact on the systems used in this work, at least when applied to the limited size of data that is currently available.

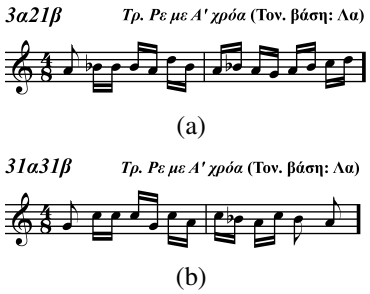


Figure 6. Two phrases found to be similar in query (7)

6. CONCLUSION

In this paper, methods have been evaluated that help to detect melodic similarity in polyphonic recordings following the logic of *parataxis*. It could be shown that a method based on histograms of the F0 estimation of the leading melody enables for an improvement compared to a baseline system that uses chroma features. Furthermore, it could be illustrated that the proposed method is capable of spotting appearances of small melodic patterns in a whole audio file, even when both files are polyphonic mixtures and the query pattern has been derived from a different recording. Such a method can be a valuable tool for research in the field of musicology, where similar phrases in a large collection could be located without the necessity of transcription, thus leading to a large saving of time.
Furthermore, the integration of melodic and rhythmic aspects is straight-forward, and it is likely to improve results for datasets in which different types of rhythms are contained. As features for melody and for rhythm can both be computed in a beat synchronous way, the correlation values obtained for a query from these two aspects could be simply added, or by using some weighting that favors either melody or rhythm derived correlations. However, the rhythmic similarity measure is quite questionable on the available dataset which is rhythmically very homogeneous, and for that reason it had to be postponed. As a future goal, the integration of rhythmic similarity as depicted in Figure 1 has to be evaluated on a more diverse dataset. However, for the compilation of such a dataset the support of experts in musicology is necessary.

7. REFERENCES

[1] Baris Bozkurt. An automatic pitch analysis method for turkish maqam music. *Journal of New Music Research*, 37(1):1–13, 2008.

[2] Juan J. Cabrera, Jose Miguel Díaz-Bánez, Francisco J. Escobar-Borrego, Emilia Gómez, Francisco Gómez, and Joaquín Mora. Comparative melodic analysis of a cappella flamenco cantes. In *Proceedings of the fourth Conference on Interdisciplinary Musicology (CIM08)*, 2008.

[3] Dan Ellis and G. Poliner. Identifying cover songs with chroma features and dynamic programming beat tracking. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP*, pages IV–1429–1432, 2007.

[4] Daniel P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[5] Andre Holzapfel and Yannis Stylianou. Beat tracking using group delay based onset detection. In *Proc. of ISMIR - International Conference on Music Information Retrieval*, pages 653–658, 2008.

[6] Andre Holzapfel and Yannis Stylianou. Scale transform in rhythmic similarity of music. *Accepted for publication in IEEE Transactions on Speech and Audio Processing*, 2010.

[7] Zoltán Juhász. Motive identification in 22 folksong corpora using dynamic time warping and self organizing maps. In *Proc. of ISMIR - International Conference on Music Information Retrieval*, pages 171–176, 2009.

[8] Anssi Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proc. of ISMIR - International Conference on Music Information Retrieval*, pages 216–221, 2006.

[9] Dirk Moelants, Olmo Cornelis, and Marc Leman. Exploring african tone scales. In *Proc. of ISMIR - International Conference on Music Information Retrieval*, pages 489–494, 2009.

[10] Edward P. Morris. *On Principles and Methods in Latin Syntax*. New York, C. Scribner's sons, 1901.

[11] Haris Sarris, Tassos Kolydas, and Panagiotis Tzevelekos. A framework of structure analysis for instrumental folk music. In *Proc. of CIM08, 4th Conference on Interdisciplinary Musicology*, Thessaloniki, Greece, 2008.

[12] Dimitris Themelis. *Morphology and analysis of music, (in Greek language)*. University Studio Press, Thessaloniki, 1994.

[13] Irini B. Theodosopoulou. *Methodology of morphological analysis and analytic data of small rhythmic patterns of cretan folk music, (in Greek Language)*. Athens: Kultura, 2004.

[14] Petri Toiviainen and Tuomas Eerola. Method for comparative analysis of folk music based on musical feature extraction and neural networks. In *In III International Conference on Cognitive Musicology*, pages 41–45, 2001.

[15] Ioannis Tsouchlarakis. *The dances of Crete - Legend, History, Tradition (in Greek language)*. Center of Cretan Culture Studies, Athens, 2000.

[16] Peter van Kranenburg, Anja Volk, Frans Wiering, and Remco C. Veltkamp. Musical models for folk-song melody alignment. In *Proc. of ISMIR - International Conference on Music Information Retrieval*, pages 507–512, 2009.

PITCH CLASS SET CATEGORIES AS ANALYSIS TOOLS FOR DEGREES OF TONALITY

Aline Honingh **Rens Bod**
Institute for Logic, Language and Computation
University of Amsterdam
{A.K.Honingh,Rens.Bod}@uva.nl

ABSTRACT

This is an explorative paper in which we present a new method for music analysis based on pitch class set categories. It has been shown before that pitch class sets can be divided into six different categories. Each category inherits a typical character which can "tell" something about the music in which it appears. In this paper we explore the possibilities of using pitch class set categories for 1) classification in major/minor mode, 2) classification in tonal/atonal music, 3) determination of a degree of tonality, and 4) determination of a composer's period.

1. INTRODUCTION

In Western classical music a distinction can be made between tonal and atonal music. Tonal music is based on a diatonic scale which inherits hierarchical pitch relationships. The pitch relationships are based on a key center or tonic. In contrast, atonal music is music that lacks a tonal center or key, and each note is valued in the same way.

From about 1908 onwards atonality has been used in compositions. Composers such as Scriabin, Debussy, Bartók, Hindemith, Prokofiev, and Stravinsky have written music that has been described, in full or in part, as atonal.

In the same way as there exists music that can be described as partly atonal, one can wonder if, in tonal music, a gradation of tonality can be found. One could argue for example that, within the category of tonal music, music written by Bach is, on average, more tonal than music written by Debussy. In this paper we will show that it is possible to make distinctions in tonality in a computational way.

The method that we will use to investigate these gradations of tonality is based on the notion of pitch class sets (hereafter pc-sets). Pc-sets have been used before as a tool to analyze atonal music [11]. Relations between pc-sets, such as transposition and inversion, have been formalized and even similarity measures have been proposed [15, 16, 20, 21, 25]. With our method, we propose a new approach to analyze music by using pc-sets, that may be valuable by providing statistical information about pieces of music. Furthermore, the approach has possible applications in several research areas, among others in automatically separating tonal from atonal music, automatically distinguishing music in a major key from music in a minor key, finding degrees of tonality, music classification, and possibly more.

Modeling tonality has been done in different ways [3, 26], however, to the best of our knowledge, no attempt has been made to measure degrees of tonality. Style classification of music has been investigated using several different methods [4, 18, 22], ranging from statistical [2, 5, 6, 27] to machine learning [8, 13] approaches. It will be worth investigating the possibilities of formalizing the degrees of tonality as a tool for classification of musical style or period. Furthermore, there are, to the best of our knowledge, no methods based on pc theory for classification of music in major/minor mode and classification in tonal/atonal music.

The rest of this paper is organized as follows. Section 2 explains the notion of pc-set categories and motivates the type of research questions that can possibly be addressed with this tool. In section 3 we will show that the (average) category distribution for tonal music differs from the (average) category distribution for atonal music. In section 4, we will show that the category distribution for music in a major key differs from the category distribution for music in a minor key. Section 5 explores the question of whether a degree of tonality can be found when investigating category distributions of music from different musical periods. Finally, section 6 gives concluding remarks.

2. CATEGORIES OF PITCH CLASS SETS

A pc-set [10] can represent both a melody and a chord since no distinction is made between notes at different onset times. Despite these simplifications, pc-sets have proven to be a useful tool in music analysis [24]. If one exhaustively lists all pc-sets (351 in total), all possible melodies and chords can fit in this list. It has been shown that all pc-sets can be grouped into six different categories [14, 19]. This can be done by applying a cluster analysis [19] to several similarity measures [15, 16, 20, 21, 25] for pc-sets.

Each pc-set category corresponds to a cycle of one of the six interval classes. This can be understood in the fol-

lowing way. A cycle of the interval 1 will read: 0,1,2,3,4, etc. A cycle of the interval 2 will read: 0,2,4,6, etc. A cycle of the interval 3 will read: 0,3,6,9, etc., and so on. Since we only take into account the first six of all twelve pitch classes (the latter six are just the inverses), six different cycles appear (see Table 1). Every category turns out to have its own character resulting from the intervals that appear most frequently, and sets of notes that belong to the same category are 'similar' in this respect.

A prototype can be identified for each category. If a certain pc-set is grouped into a certain category, this pc-set can be said to be similar to the prototype of that category. The set $\{0, 1, 2, 3, 4\}$ is the prototype of the Interval Category 1 (IC1) in the pentachord classification, the set $\{0, 2, 4, 6, 8\}$ the prototype of IC2, and so on. The cycles of IC's that have periodicities that are less than the cardinality of their class (for example, pc 4 has a periodicity of 3: {0,4,8}) are extended in the way described by Hanson [12]: the cycle is shifted to pc 1 and continued from there. For example, the IC-6 cycle proceeds $\{0, 6, 1, 7, 2, 8...\}$ and the IC-4 cycle proceeds $\{0, 4, 8, 1, 5, 9, 2, ...\}$. Thus for every cardinality, a separate prototype characterizes the category. For example, category IC4 has prototype $\{0, 4\}$ for sets of cardinality 2, prototype $\{0, 4, 8\}$ for set of cardinality 3, and so on. Table 1 gives an overview of the prototypes of pc-set categories. Prototypes can been listed for duochords to decachords. Pc-sets with less than 2 notes or more than 10 notes can not be classified. This is because one pc-set of cardinality 1 exists, $\{0\}$, and it belongs equally to every category. The same is true for cardinality 11: only one prime form pc-set exists: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and belongs to every category equally. The pc-set of cardinality 12 contains all possible pitch classes.

2.1 Music analysis using pc-set categories

Each category can be seen as having a particular character resulting from the intervals that appear most frequently. Interval category 1, or category 1 for short, consists of all semitones and is the category of the chromatic scale. Category 2 is the category of the whole-tones or whole-tone scale. Category 3 is the category of the diminished triads or diminished scale. Category 4 is the category of the augmented triads or augmented scale. Category 5 is the category of the diatonic scale. Category 6 is the category of the tritones or D-type all-combinatorial hexachord (see [12]).

Because of the typical character of each of the categories, these categories can 'tell' something about the music in which they appear. If a piece of music is dominated by a particular category, the music is likely to broadcast the character of that category.

Ericksson [9] already argued that music can be divided into categories similar to the ones described above and says that "it is often possible to show that one region [category] dominates an entire section of a piece". Our approach goes further in that we fully formalize and automate these categories. When a piece of music is segmented, the category of each segment can be calculated and the distribution of categories for that piece can be presented. The category distribution of a piece of music can present information about this piece that is possibly new and can lead to new insights on specific music. Furthermore, this information may lead to methods for automatic differentiation of music in a major key from music in a minor key, automatic classification of tonal/atonal music, and style classification [2, 5, 6, 27]. Since a pc-set category is by definition a category that consists of similar pc-sets [14, 19], these categories are also expected to form a useful tool in the research area of music similarity problems [1, 7, 17, 21, 23, 28, 29, 31].

2.2 Derivation of category distributions

The method has been implemented in Java, using parts of the Musitech Framework [30], and operates on MIDI data. The MIDI files are segmented at the bar level, as a first step to investigate the raw regularities that occur on this level [1]. The internal time signature of the music is recognized by methods of the Musitech Framework, meaning that, if there is a time signature change, the segmentation per bar will correctly continue.

The pitches from each segment form a pc-set. From each pc-set, the interval class vector can be calculated after which the pc-set category can be calculated. This is done as follows. Using Rogers' cosθ [21] as similarity measure we calculate the similarity to all prototypes of the required cardinality. The prototype to which the set is most similar, represents the category to which the set belongs [14]. However, if the pc-set that is constructed from a bar contains less than 2 or more than 10 different pitch classes, the set belongs equally to every category, as we explained before. To overcome this problem, the segmentation is changed as follows. If a set (bar) contains more than 10 different pitch classes, the bar is divided into beats and the beats are treated as new segments. If a set contains less than 2 pitch classes, this set is added to the set that is constructed from the next bar, forming a new segment. In this way, the number of occurrences of the categories can be obtained, taking into account all pitches in the MIDI file. The number of occurrences of all categories can be presented as percentages, making comparison to other music possible.

3. TONAL VERSUS ATONAL

Every category represents a particular character; thus it can be expected that different types of music will show a different occurrence rate for each category. Since category 5 is the category of the diatonic scale, we expect the occurrence rate of category 5 to be high for tonal music. Choosing a data set [2] of tonal music, the overall category distribution can be calculated as we explained in the previous section.

[1] Preliminary experiments showed that the results following from segmentation per beat, bar or two bars vary only minimally.

[2] Music in MIDI format has been downloaded from the following websites: `http://www.kunstderfuge.com/`, `http://www.classicalarchives.com/`, `http://www.classicalmidiconnection.com/`, `http://www.musiscope.com/`, and `http://www.classicalmusicmidipage.com/`.

(Interval) Category	prototypes (pc sets)				‘character’ of category
IC1	$\{0,1\}$,	$\{0,1,2\}$,	$\{0,1,2,3\}$,	etc.	semitones
IC2	$\{0,2\}$,	$\{0,2,4\}$,	$\{0,2,4,6\}$,	etc.	whole-tones
IC3	$\{0,3\}$,	$\{0,3,6\}$,	$\{0,3,6,9\}$,	etc.	diminished triads
IC4	$\{0,4\}$,	$\{0,4,8\}$,	$\{0,1,4,8\}$,	etc.	augmented triads
IC5	$\{0,5\}$,	$\{0,2,7\}$,	$\{0,2,5,7\}$,	etc.	diatonic scale
IC6	$\{0,6\}$,	$\{0,1,6\}$,	$\{0,1,6,7\}$,	etc.	tritones

Table 1. Prototypes expressed in pc-sets for the six categories. Prime forms have been used to indicate the prototypes (therefore IC5 may appear differently than one may expect).

composer	**piece**
Bach	Brandenburg concerto no. 3
Mozart	Piano concerto no. 5 part 1
Beethoven	Piano sonata Pathetique
Brahms	Clarinet quintet part 1
Mahler	Symphony no. 4 part 1
Debussy	Nocturnes: Nuages

Table 2. The tonal music that was used to calculate Table 3.

category	number of occurrences	percentage of occurrence	standard deviation
1	54	3.22 %	2.09 %
2	83	4.96 %	5.96 %
3	321	19.16 %	8.48 %
4	247	14.75 %	8.33 %
5	890	53.13 %	19.21 %
6	80	4.78 %	2.50 %

Table 3. Distribution of categories in tonal music listed in Table 2.

Table 2 lists the tonal music that has been used for this experiment and Table 3 gives the percentages of occurrences of the categories that are found in this corpus. We see from Table 3 that the music is dominated by category 5. We indeed expected a high occurrence rate of category 5, as this is the category that represents the diatonic scale. However, since the standard deviation is relatively high, the individual percentages vary quite a bit.

For atonal music, we expect a different behavior. We have run the program on strict atonal music composed by Schoenberg, Webern, Stravinsky and Boulez. The complete list of music is shown in Table 4. On average, the distribution as shown in Table 5 was found, using this corpus of atonal music. We can see that the music is not dominated anymore by category 5 but a much more equal distribution is present in atonal music. From the difference in these category distributions, it seems that especially the occurrence of category 5 could contribute to classification methods to separate atonal from tonal music. Cross validation needs to be performed in order to verify this claim. However, since we could find only few MIDI data of atonal music (all MIDI data we found on atonal music is given in Table 4), it is difficult to perform a cross validation with enough data.

composer	**piece**
Schoenberg	Pierrot Lunaire part 1, 5, 8, 10, 12, 14, 17, 21
Schoenberg	Piece for piano opus 33
Schoenberg	Six little piano pieces opus 19 part 2, 3, 4, 5, 6
Webern	Symphony opus 21 part 1
Webern	String Quartet opus 28
Boulez	Notations part 1
Boulez	Piano sonata no 3, part 2: “Texte”
Boulez	Piano sonata no 3, part 3: “Parenthese”
Stravinsky	in memoriam Dylan Thomas Dirge canons (prelude)

Table 4. The atonal music that was used to calculate Table 5.

category	number of occurrences	percentage of occurrence	standard deviation
1	313	28.25 %	10.56 %
2	117	10.56 %	6.14 %
3	166	14.98 %	7.68 %
4	179	16.16 %	7.97 %
5	138	12.45 %	7.15 %
6	195	17.60 %	6.20 %

Table 5. Distribution of categories from music of Schoenberg, Webern, Stravinsky and Boulez.

4. MAJOR VERSUS MINOR

The tonality turns out not to be the only factor to influence the percentage of occurrence of category 5 in music. If we focus on tonal music, an obvious difference can be measured in the occurrence of category 5 between music in major and in minor mode. To show this behavior, we have chosen Bach’s Well-tempered Clavier book I as test corpus and divided the corpus in two parts: 1) the pieces in a major key, and 2) the pieces in a minor key. From Table 6 we can see the differences in category distribution between the two parts and the overall corpus. The pieces in major mode have an average percentage of occurrence of category 5 of 79.81 %, while for the pieces in minor mode this percentage is considerably lower, namely 53.58 % (see Table 6). As one can see, the standard deviations in Table 6 for the music separated in major and minor are smaller than for all pieces together, which means that the measurements are distributed closer around their mean. We can now understand that the standard deviation in Table 3 was relatively large since the data contained both data in major and in minor mode. Reconsidering the results of the previous section, the average percentage of occurrence of category 5

category	occurrences for pieces in **major**	standard deviation	occurrences for pieces in **minor**	standard deviation	occurrences for **all** pieces	standard deviation
1	(31) 2.30 %	1.03 %	(85) 4.48 %	2.22 %	(116) 3.57 %	2.12 %
2	(25) 1.86 %	1.69 %	(98) 5.16 %	2.29 %	(123) 3.79 %	2.63 %
3	(149) 11.06 %	3.23 %	(453) 23.87 %	3.72 %	(602) 18.55 %	7.23 %
4	(47) 3.49 %	3.22 %	(199) 10.48 %	3.73 %	(246) 7.58 %	4.93 %
5	(1075) 79.81 %	6.52 %	(1017) 53.58 %	3.63 %	(2092) 64.47 %	13.87 %
6	(20) 1.48 %	1.24 %	(46) 2.42 %	1.75 %	(66) 2.03 %	1.63 %

Table 6. Distribution of categories in percentages (the number in given between brackets) from the pieces in major mode, and minor mode, and all pieces together, from Bach's Well-tempered Clavier book I.

composer	percentage of occurrence of category 5 for **major** mode	standard deviation	percentage of occurrence of category 5 for **minor** mode	standard deviation
Palestrina	71.94 % *	4.55 %		
Bach	85.71 %	3.99 %	57.72 %	11.02 %
Mozart	58.17 %	7.75 %	37.94 %	6.04 %
Beethoven	47.98 %	6.62 %	36.09 %	7.66 %
Brahms	40.79 %	1.42 %	40.11 %	4.55 %
Mahler	53.83 %	18.33 %	35.95 %	9.24 %
Debussy	68.01 %	5.24 %	40.57 %	9.23 %
Stravinsky	27.65 % *	3.83 %		

Table 7. The percentage of occurrence of category 5 for several composers, separating music in a major and minor key. * For Palestrina and Stravinsky, the separation between music in major and minor mode has not been made (see text for details).

in the music in minor mode of Table 6 is still considerably higher than the average percentage of occurrence of category 5 in atonal music, where one cannot speak of major or minor mode. Moreover, the tonal data from the previous section contained nearly as much music in major mode as music in minor mode. We have to remark, however, that in general, many pieces of music contain segments in both major and in minor mode, although an overall piece is said to be in either major or minor. In our method, we have classified the pieces of music only in a global way (based on the mode of the overall piece), motivated by the consensus that a piece of music in a specific mode will usually contain a majority of segments that are in that mode.

It is understandable that music in minor mode exhibits a lower percentage of category 5 than music in major mode, for the reason that category 5 is the category of the diatonic (major) scale. Although the natural minor scale is diatonic as well, in music in a minor key, other variants like the melodic and harmonic minor scale are frequently used too. For music in minor mode, apart from a high percentage of category 5, categories 3 and to a lesser extent category 4 represent relatively high percentages as well. In contrast, the atonal music has also relatively high percentages of categories 1, 2 and 6. The raised percentage of category 3 for tonal music in minor mode may be explained from the presence of the minor third, and the raised percentage of category 4 from the presence of the minor sixth.

The example in this section shows that for a particular type of music, measuring the percentage of occurrence of category 5, would enable to make a distinction between music in major and in minor mode.

5. DEGREE OF TONALITY?

In the previous sections, we have seen that of all categories, especially category 5 can give some information about both the tonality and the mode. It may be worth to focus on this category for specific composers and to study the difference between them. In Table 7 the percentage of occurrence of category 5 is shown for several composers. The composers are ordered chronologically. For two composers, Palestrina and Stravinsky, no separation is made between major and minor mode. A lot of work by Stravinsky is difficult to be labeled as completely major or minor, and some of his later works can even be labeled as atonal. For Palestrina, no separation between major and minor has been made, since in Renaissance music, besides the normal major and minor scales, eight church modes are used as well. For each composer and for each mode (major or minor), on average 5 pieces of music have been selected, such as to form a representative sample that contains the different musical forms (symphonies, chamber music, concertos, etc.) present in the repertoire of the composer.

Based on the result from section 3 stating that tonal music contains a higher percentage of category 5, we might expect a decreasing percentage of category 5 when the composers are ordered chronologically. For example, one might label Bach as more tonal than for example Mahler since the latter composer would be closer in time to the contemporary period in which the atonal music flourished. This hypothesis turned out not to be true however. It is indeed the case that Bach embodies a higher percentage of category 5 than Mahler, but if we focus on the composers for whom a distinction was made between major and minor music, we see a decreasing percentage of category 5 from Bach to

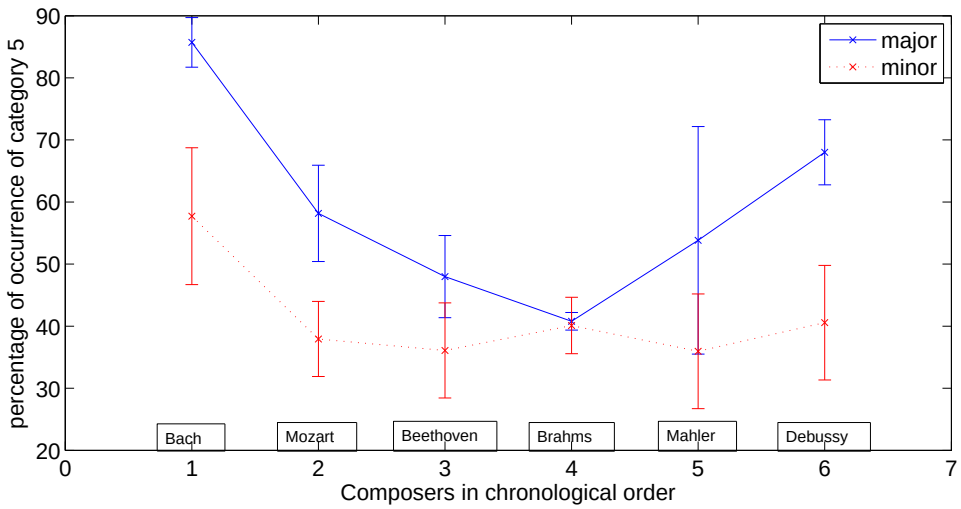


Figure 1. The percentage of occurrence of category 5 for most composers from Table 7, in chronological order. The error bars represent the standard deviation from Table 7.

period	percentage of occurrence of category 5	standard deviation	percentage of occurrence of category 5	standard deviation
Baroque	Bach 85.71 %	3.99 %	Händel 77.40 %	4.64 %
Romantic	Beethoven 47.98 %	6.62 %	Schubert 45.94 %	7.36 %
Impressionist	Debussy 68.01 %	5.24 %	Ravel 40.96 %	14.14 %

Table 8. The percentage of occurrence of category 5 for different composers from the same musical period.

Brahms (focusing on major mode), although from Brahms to Debussy the percentage of category 5 increases again, see Figure 1.

Based on the result from section 4 we expect higher percentages of category 5 for major music than for minor music for each composer. Indeed, this turns out to be the case (see Table 7), although the difference is very small for Brahms.

One could now wonder whether the results of Table 7 show a general behavior that is typical for composers from different musical periods from Renaissance to modern, or that the results are just specific for these composers. We study the differences between composers who lived in the same period, since this might explain the results of Table 7 a bit further. We have zoomed in on music in major mode in three different musical periods, namely the Baroque, Romantic and Impressionist periods (see Table 8) and looked at the difference between two composers within the same period. One can see that for the Baroque and Romantic period, the percentages of category 5 are very much alike for the two composers chosen. However, for the Impressionist period there is a substantial difference between the percentages.

The finding that each composer represents a typical percentage of occurrence of category 5 can possibly be used in applications for style recognition.

6. CONCLUDING REMARKS

In this paper, a new analysis method for music has been proposed, and we explored a number of possible applications. We showed that the six pc-set categories [14, 19] can reveal specific information about music. When music is segmented, and when for each bar is calculated to which category its pc contents belongs, the percentages of the different categories can reveal information about the tonality of the piece (tonal or atonal) and the mode of the piece (major or minor). In particular, category 5, which represents the major diatonic scale, is indicative of this information.

Although on the basis of the percentage of occurrence of category 5, a separation between tonal and atonal music may be made, it does not allow us to order specific music in time. More research needs to be done to be able to explore whether the percentage of occurrence of category 5 can be indicative of a certain style or musical period.

We fully recognize that cross validation experiments need to be carried out in order to verify the suggested possibility of using pc-set categories for the purpose of tonal/atonal classification and major/minor classification, but this is hampered so far by a lack of MIDI data especially for atonal music. Furthermore, in future research we hope to be able to perform an actual classification task. Since the tonal/atonal classification and the major/minor classification both de-

pend on the percentage of occurrence of pc-set category 5, this will not be a trivial task.

To conclude, the distribution of pc-set categories can reveal information about music on different levels, and we suggest that they can serve as a new tool in music analysis.

7. ACKNOWLEDGEMENTS

The authors wish to thank Darrell Conklin and Tillman Weyde and three anonymous reviewers for constructive comments and feedback. This research was supported by grant 277-70-006 of the Netherlands Foundation for Scientific Research (NWO).

8. REFERENCES

[1] Chantal Buteau. Automatic motivic analysis including melodic similarity for different contour cardinalities: application to Schumann's of foreign lands and people. In *Proceedings of the International Computer Music Conference*, Barcelona, 2005.

[2] Wei Chai and Barry Vercoe. Folk music classification using hidden markov models. In *Proceedings of International Conference on Artificial Intelligence*, 2001.

[3] Elaine Chew. *Towards a mathematical model of tonality*. PhD thesis, Operations Research Center, Massachusetts Institute of Technology, Cambridge, 2000.

[4] Elaine Chew, Anja Volk, and Chia-Ying Lee. Dance music classification using inner metric analysis: a computational approach and case study using 101 latin american dances and national anthems. In *Proceedings of the 9th INFORMS Computing Society Conference*, volume 29, pages 355–370. Springer OR/CS Interfaces Series, 2005.

[5] Rudi Cilibrasi, Paul Vitanyi, and Ronald de Wolf. Algorithmic clustering of music based on string compression. *Computer Music Journal*, 28(4):49–67, 2004.

[6] Roger Dannenberg, Belinda Thom, and David Watson. A machine-learning approach to musical style recognition. In *Proceedings of the 1997 International Computer Music Conference*, pages 344–347, San Francisco, 1997.

[7] Irène Deliège. Introduction : Similarity perception, categorization, cue abstraction. *Music Perception*, 18(3):233–243, 2001.

[8] Shlomo Dubnov, Gerard Assayag, Olivier Lartillot, and Gill Bejerano. Using machine-learning methods for musical style modeling. *Computer*, 36(10):73–80, 2003.

[9] Tore Ericksson. The ic max point structure, mm vectors and regions. *Journal of Music Theory*, 30(1):95–111, 1986.

[10] Allen Forte. *The Structure of Atonal Music*. New Haven: Yale University Press, 1973.

[11] Allen Forte. Pitch-class set analysis today. *Music Analysis*, 4(1/2):29–58, 1985.

[12] Howard Hanson. *Harmonic Materials of Modern Music*. New York: Appleton-Century-Crofts, 1960.

[13] Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. Global feature versus event models for folk song classification. In *ISMIR 2009:10th International Society for Music Information Retrieval Conference*, Kobe, Japan, 2009.

[14] Aline K. Honingh, Tillman Weyde, and Darrell Conklin. Sequential association rules in atonal music. In *Proceedings of Mathematics and Computation in Music (MCM2009)*, New Haven, USA, June 19-22, 2009.

[15] Eric J. Isaacson. Similarity of interval-class content between pitch-class sets: the IcVSIM relation. *Journal of Music Theory*, 34:1–28, 1990.

[16] Robert Morris. A similarity index for pitch-class sets. *Perspectives of New Music*, 18:445–460, 1980.

[17] Robert Morris. Equivalence and similarity in pitch and their interaction with pcset theory. *Journal of Music Theory*, 39(2):207–243, 1995.

[18] Carlos Perez-Sancho, David Rizo, and Jose Inesta. Genre classification using chords and stochastic language models. *Connection Science*, 21(2-3):145–159, 2009.

[19] Ian Quinn. Listening to similarity relations. *Perspectives of New Music*, 39:108–158, 2001.

[20] John Rahn. Relating sets. *Perspectives of New Music*, 18:483–498, 1980.

[21] David W. Rogers. A geometric approach to pcset similarity. *Perspectives of New Music*, 37(1):77–90, 1999.

[22] Adi Ruppin and Hezy Yeshurun. Midi music genre classification by invariant features. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 397–399, Canada, 2006.

[23] Art Samplaski. Mapping the geometries of pitch-class set similarity measures via multidimensional scaling. *Music Theory Online*, 11(2), 2005.

[24] Michiel Schuijer. *Atonal music: Pitch-Class Set Theory and Its Contexts*. University of Rochester Press, 2008.

[25] Damon Scott and Eric J. Isaacson. The interval angle: A similarity measure for pitch-class sets. *Perspectives of New Music*, 36(2):107–142, 1998.

[26] David Temperley. The tonal properties of pitch-class sets: Tonal implication, tonal ambiguity and tonalness. *Tonal Theory for the Digital Age, Computing in Musicology*, 15:24–38, 2007.

[27] George Tzanetakis and Perry Cook. Music genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[28] Anja Volk, Peter van Kranenburg, Joerg Garbers, Frans Wiering, Remco C. Veltkamp, and Louis P. Grijp. A manual annotation method for melodic similarity and the study of melody feature sets. In *Proceedings of the Ninth International Conference on Music Information Retrieval (ISMIR)*, pages 101–106, Philadelphia, USA, 2008.

[29] Tillman Weyde. Integrating segmentation and similarity in melodic analysis. In *Proceedings of the International Conference on Music Perception and Cognition 2002*, pages 240–243, Sydney, 2002.

[30] Tillman Weyde. Modelling cognitive and analytic musical structures in the MUSITECH framework. In *UCM 2005 5th Conference "Understanding and Creating Music", Caserta, November 2005*, pages 27–30, 2005.

[31] Geraint Wiggins. Models of musical similarity. *Musicae Scientiae, Discussion Forum 4a*, pages 315–338, 2007.

PREDICTION OF TIME-VARYING MUSICAL MOOD DISTRIBUTIONS FROM AUDIO

Erik M. Schmidt and Youngmoo E. Kim
Electrical and Computer Engineering, Drexel University
{eschmidt,ykim}@drexel.edu

ABSTRACT

The appeal of music lies in its ability to express emotions, and it is natural for us to organize music in terms of emotional associations. But the ambiguities of emotions make the determination of a single, unequivocal response label for the mood of a piece of music unrealistic. We address this lack of specificity by modeling human response labels to music in the arousal-valence (A-V) representation of affect as a *stochastic distribution*. Based upon our collected data, we present and evaluate methods using multiple sets of acoustic features to estimate these mood distributions parametrically using multivariate regression. Furthermore, since the emotional content of music often varies within a song, we explore the estimation of these A-V distributions in a *time-varying* context, demonstrating the ability of our system to track changes on a short-time basis.

1. INTRODUCTION

The problem of automated recognition of emotional content (mood) within music is the subject of increasing attention among music information retrieval (MIR) researchers [1–3]. Human judgements are necessary for deriving emotion labels and associations, but perceptions of the emotional content of a given song or musical excerpt are bound to vary and reflect some degree of disagreement between listeners. In developing computational systems for recognizing musical affect, this lack of specificity presents significant challenges for the traditional approach of using supervised machine learning systems for classification. Instead of viewing musical mood as a singular label or value, the modeling of emotional "ground-truth" as a *probability distribution* potentially provides a more realistic (and accurate) reflection of the perceived emotions conveyed by a song.

A variety of methods are used for collecting mood-specific labels for music corpora, for example, annotations curated by experts (e.g., Allmusic.com) and the analysis of unstructured user-generated tags (e.g., Last.fm). While these approaches efficiently provide data for large collections, they are not well-suited for reflecting variations in the emotional content as the music changes. In prior work we created *MoodSwings* [4], an online collaborative activity designed to collect second-by-second labels of music using the two-dimensional, arousal-valence (A-V) model of human emotion, where valence indicates positive vs. negative emotions and arousal reflects emotional intensity [5]. The game was designed specifically to capture A-V labels dynamically (over time) to reflect emotion changes in synchrony with music and also to collect a distribution of labels across multiple players for a given song or even a moment within a song. This method potentially provides quantitative labels that are well-suited to computational methods for parameter estimation.

In previous work we have investigated short-time regression approaches for emotional modeling, developing a functional mapping from a large number of acoustic features directly to A-V space coordinates [1]. Since the application of a single, unvarying mood label across an entire song belies the time-varying nature of music, we focused on using short-time segments to track emotional changes over time. In our current work we demonstrate that not only does the emotional content change over time, but also that a distribution of (as opposed to singular) ratings is appropriate for even short time slices (down to one second). In observing the collected data, we have found that most examples can be well represented by a single two-dimensional Gaussian distribution.

To perform the mapping from acoustic features to the A-V mood space, we explore parameter prediction using multiple linear regression (MLR), partial least-squares (PLS) regression, and support vector regression (SVR). In modeling the data as a two dimensional Gaussian, our goal is to be able to predict the A-V distribution parameters $\mathcal{N}(\mu, \Sigma)$ from the acoustic content. We first evaluate the effectiveness of this system in predicting emotion distributions for 15 second clips and subsequently shorten the analysis window length to demonstrate its ability to follow changes in A-V label distributions over time.

No dominant acoustic feature has yet emerged for music emotion recognition, and previous work has focused on combining multiple feature sets [1–3, 6]. We evaluate multiple sets of acoustic features for each task, including psychoacoustic (mel-cepstrum and statistical frequency spectrum descriptors) and music-theoretic (estimated pitch chroma) representations of the labeled audio. Although the large number of potential features can present

problems, rather than employing dimensionality reduction methods (e.g., principal components analysis) we explore an alternative method for combining different feature sets, using ensemble methods to determine the relative contribution of single-feature systems for improved overall performance.

2. BACKGROUND

The general approach to implementing automatic mood detection from audio has been to use supervised machine learning to train statistical models based on acoustic features. Recent work has also indicated that regression approaches often outperform classification when using similar features [1, 2].

Yang *et al.* introduced the use of regression for mapping of high-dimensional acoustic features into the two-dimensional space [6]. Support vector regression (SVR), as well as a variety of boosting algorithms including AdaBoost.RT, were applied to solve the regression problem. The ground-truth A-V labels were collected by recruiting 253 college students to annotate the data, and only one label was collected per clip. Compiling a wide corpus of features totaling 114 feature dimensions, they applied principal component analysis (PCA) before regression.

Further confirming the robustness of regression for A-V emotion prediction, Han *et al.* demonstrated that regression approaches can outperform classification when applied to the same problem [2]. Their classification task consisted of a quantized version of the A-V space into 11 blocks. Using a wide variety of audio features, they initially investigated the use of classification, obtaining only $\sim$33%. Still mapping to the same 11 quantized categories, applying regression they obtained up to $\sim$95% accuracy.

Eerola *et al.* introduced the use of a three-dimensional parametric emotion model for labeling music [3]. In their work they investigated multiple regression approaches including Partial Least-Squares (PLS) regression, an approach that considers correlation between label dimensions. They achieve R^2 performance of 0.72, 0.85, and 0.79 for valence, activity, and tension, respectively, using PLS and also report peak R^2 prediction rates for 5 basic emotion classes (angry, scary, happy, sad, and tender) as ranging from 0.58 to 0.74.

3. GROUND TRUTH DATA COLLECTION

Traditional methods for collecting perceived mood labels, such as the soliciting and hiring of human subjects, can be flawed. In MoodSwings, participants use a graphical interface to indicate a dynamic position within the A-V space to annotate five 30-second music clips. Each subject provides a check against the other, reducing the probability of nonsense labels. The song clips used are drawn from the "uspop2002" database, [1] and overall we have collected over 150,000 individual A-V labels spanning more than 1,000 songs.

[1] uspop2002 dataset: http://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html

Since the database consists entirely of popular music, the labels collected thus far display an expected bias towards high-valence and high-arousal values. Although inclusion of this bias could be useful for optimizing classification performance, it is not as helpful for learning a mapping from acoustic features that provides coverage of the entire emotion space. Because of this trend, we developed a reduced dataset consisting of 15-second music clips from 240 songs, selected using the original label set, to approximate an even distribution across the four primary quadrants of the A-V space. These clips were subjected to intense focus within the game in order to form a corpus referred to here as MoodSwings Lite, with significantly more labels per song clip, which is used in this analysis.

4. ACOUSTIC FEATURE COLLECTION

As previously stated, there is no single dominant feature, but rather many that play a role (e.g., loudness, timbre, harmony) in determining the emotional content of music. Since our experiments focus on the tracking of emotion over time, we chose to focus on solely on time-varying features. Our collection (Table 1) contains many features that are popular in Music-IR and speech processing, encompassing both psychoacoustic and music-theoretic representations. Instead of raw chroma we utilize the autocorrelation of each short-time chroma vector, providing a shift-invariant feature. In preliminary experiments we found this feature to perform better than raw chroma, since it promotes similarity in terms of the modes of harmony (e.g. major, minor, augmented, and diminished chords) as opposed to particular chords (e.g., A major vs. D major).

Feature	Description
Mel-frequency cepstral coefficients (MFCCs) [7]	Low-dimensional representation of the spectrum warped according to the mel-scale. 20-dimensions used.
Chroma (i.e., Pitch Class Profile) [8]	Autocorrelation of chroma is used, providing an indication of modality.
Spectral Spectrum Descriptors (SSDs) [9]	Includes spectral centroid, flux, rolloff, and flatness. Often related to timbral texture.
Spectral Contrast [10]	Rough representation of the harmonic content in the frequency domain.

Table 1. Acoustic feature collection for music emotion regression.

5. EXPERIMENTS AND RESULTS

Given the continuous nature of our problem, the prediction of a 2-d Gaussian within the A-V space, we explored several methods for multi-variate parameter regression. In these experiments we employ multiple linear regression (MLR), partial least-squares (PLS), and support vector regression (SVR) to create optimal projections from each of the acoustic feature sets described above. For our initial distribution regression experiments, we averaged feature

Feature/ Topology	Regression Method	Average Mean Distance	Average KL Divergence	Average Randomized KL Divergence	T-test
MFCC	MLR	0.161 ± 0.008	4.098 ± 0.513	8.516 ± 1.566	5.306
Chroma	MLR	0.185 ± 0.010	5.617 ± 0.707	7.765 ± 2.135	5.659
S. Shape	MLR	0.167 ± 0.009	4.183 ± 0.656	7.691 ± 1.573	5.582
S. Contrast	MLR	$\mathbf{0.151 \pm 0.008}$	$\mathbf{3.696 \pm 0.657}$	$\mathbf{8.601 \pm 1.467}$	**5.192**
MFCC	PLS	0.155 ± 0.008	3.863 ± 0.56712	8.306 ± 1.389	5.540
Chroma	PLS	0.183 ± 0.010	5.286 ± 0.96019	7.146 ± 1.665	5.565
S. Shape	PLS	0.151 ± 0.008	3.770 ± 0.84026	8.278 ± 1.527	4.951
S. Contrast	PLS	$\mathbf{0.151 \pm 0.008}$	$\mathbf{3.684 \pm 0.644}$	$\mathbf{8.700 \pm 1.831}$	**5.171**
MFCC	SVR	$\mathbf{0.140 \pm 0.008}$	$\mathbf{3.186 \pm 0.597}$	$\mathbf{7.744 \pm 1.252}$	**5.176**
Chroma	SVR	0.186 ± 0.008	4.831 ± 0.737	6.466 ± 0.935	5.655
S. Shape	SVR	0.176 ± 0.008	4.611 ± 0.841	7.348 ± 1.025	5.251
S. Contrast	SVR	0.150 ± 0.008	3.357 ± 0.500	7.356 ± 1.341	5.301
Stacked Features	MLR	0.152 ± 0.007	3.917 ± 0.496	9.355 ± 1.879	5.737
Fusion Unweighted	MLR	0.149 ± 0.007	3.333 ± 0.433	6.785 ± 0.996	5.879
Fusion Weighted	MLR	0.147 ± 0.007	3.280 ± 0.423	6.803 ± 1.309	5.980
M.L. Seperate	MLR	0.147 ± 0.007	3.399 ± 0.478	8.235 ± 1.598	5.598
M.L. Combined	MLR	$\mathbf{0.145 \pm 0.007}$	$\mathbf{3.198 \pm 0.454}$	$\mathbf{7.637 \pm 1.389}$	**5.551**
Stacked Features	PLS	0.145 ± 0.006	3.403 ± 0.467	8.407 ± 1.635	5.543
Fusion Unweighted	PLS	0.145 ± 0.007	3.332 ± 0.508	7.123 ± 1.461	5.681
Fusion Weighted	PLS	0.145 ± 0.006	3.309 ± 0.501	7.160 ± 1.373	5.619
M.L. Seperate	PLS	0.145 ± 0.008	3.465 ± 0.577	8.426 ± 1.705	5.433
M.L. Combined	PLS	$\mathbf{0.144 \pm 0.007}$	$\mathbf{3.206 \pm 0.515}$	$\mathbf{7.889 \pm 1.656}$	**5.485**

Table 2. Distribution regression results for fifteen second clips.

dimensions across all frames of a given 15-second music clip, thus representing each clip with a single vector of features. Preliminary experiments were performed using second- and higher-order statistics with the 15-second clips, but in all cases the inclusions of such data failed to show any significant performance gains.

In all experiments, to avoid the well-known "album-effect", we ensured that any songs that were recorded on the same album were either placed entirely in the training or testing set. Additionally, each experiment was subject to over 50 cross-validations, varying the distribution of training and testing data sets.

5.1 Single Feature Emotion Distribution Prediction

There are many possible methods for evaluating the performance of our system. Kullback-Liebler (KL) divergence (relative entropy) is commonly used to compare probability distributions. Since the regression problem targets known distributions, our primary performance metric is the non-symmetrized (one-way) KL divergence (from the projected distribution to that of the collected A-V labels). To provide an additional qualitative metric, we also provide results as the Euclidean distance between the projected means as a normalized percentage of the A-V space. However, to provide context to KL values and to benchmark the significance of the regression results, we compared the projections to those of an essentially random baseline. Given a trained regressor and a set of labeled testing examples, we first determined an A-V distribution for each sample. The resulting KL divergence to the corresponding A-V distribution was compared to that of another randomly selected A-V distribution from the test set. Comparing these cases over 50 cross-validations, we computed Student's T-test for paired samples to verify the statistical significance of our results.

From Table 2 it can be seen that the best performing single feature system is SVR with MFCC features at an average KL of 3.186. However, in both the MLR and PLS system the highest performing single feature is spectral contrast with 3.696 and 3.684, respectively. As the main advantage of PLS over MLR is that it observes any correlation between dimensions in the multivariate regression, it is surprising that the performance difference between the two is nearly negligible. Given our degrees of freedom (72 test samples), even our lowest T-test value (5.171) produces confidence of statistical significance greater than 99.999%.

Shown in Figure 1 is the projection of six 15-second clips into the (A-V) space resulting from multiple regression methods and acoustic features. The standard deviation of the ground truth as well as each projection is shown as an ellipse. The performance of the regression can be evaluated in terms of the total amount of overlap between a projection and its ground truth.

5.2 Feature Fusion

While most individual features perform reasonably in mapping to A-V coordinates, a method for combining information from these domains (more informed than simply concatenating the features) could potentially lead to higher performance. In this section we investigate multiple schemes for feature fusion. Given the very small performance gains and high computational overhead of SVR, we chose to narrow our focus to MLR and PLS for these

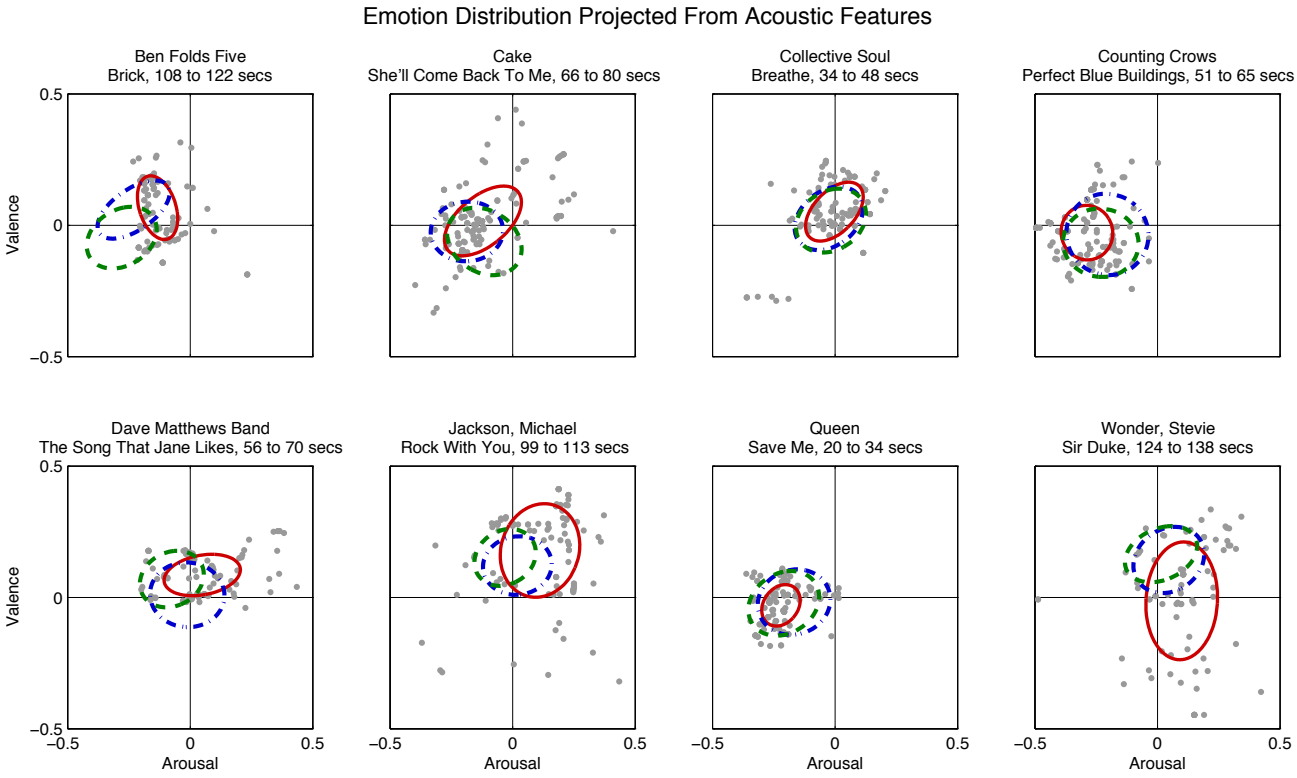


Figure 1. Collected A-V labels and distribution projections resulting from regression analysis. A-V labels: second-by-second labels per song (gray •), Σ of collected labels (solid red ellipse), Σ of MLR projection from spectral contrast features (dash-dot blue ellipse), Σ of MLR Multi-Level combined projection (dashed green ellipse).

experiments. As our ultimate system will require many predictions over time in order to reflect emotional changes, the costs of SVR outweigh the benefits.

In our fusion results the performance for simply stacking features into one large feature vector is provided to give context to the other fusion methods. Our more simple approach consists of a fusion system that is a combination of the outputs from the individual feature regression systems. In the unweighted approach we simply average the parameter outputs from each individual feature regressor, and in the weighted approach we weight each individual feature regressor by its ability to predict a particular parameter, which is determined by leave-one-out cross-validation.

In addition, we develop a two-level regression scheme by feeding the outputs of individual regressors, each trained using distinct features, into a second-stage regressor determining the final prediction. We investigated two topologies (Figure 2): in one case the secondary arousal and valence regressors receive only arousal and valence estimates, respectively; in the second case the secondary arousal and valence regressors receive both arousal and valence estimates from the first-stage. We refer to these two topologies as multi-layer separate and multi-layer combined. In all cases the secondary regressors are trained using a leave-one-out method (on each iteration we train the first-stage regressors leaving one example out and use the estimates of that example from the first stage to train the second stage). The results for both cases are shown in Table 2.

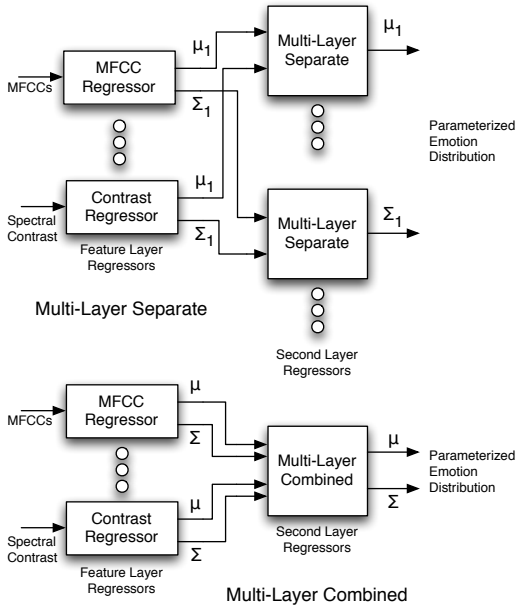


Figure 2. Multi-layer regression topologies.

5.3 Time-varying emotion distribution prediction

In attempting to predict the emotion distribution over time, we next shorten label observation rate to once per-second and attempt to regress multiple feature windows from each song. For the ground truth data collection this means that for each 15-second clip we now have 15 examples, increasing our total corpus to 3600 examples. Of course for any

Feature/ Topology	Average Mean Distance	Average KL Divergence	Average Randomized KL Divergence	T-test
MFCC	0.169 ± 0.007	14.61 ± 3.751	27.00 ± 10.33	10.77
Chroma	0.190 ± 0.007	18.71 ± 6.819	22.53 ± 6.984	9.403
S. Shape	0.173 ± 0.007	15.46 ± 6.402	24.61 ± 9.220	11.06
S. Contrast	$\mathbf{0.160 \pm 0.006}$	$\mathbf{13.61 \pm 5.007}$	$\mathbf{27.29 \pm 9.861}$	**10.23**
M.L. Combined	$\mathbf{0.154 \pm 0.006}$	$\mathbf{13.10 \pm 5.359}$	$\mathbf{28.39 \pm 10.35}$	**10.08**

Table 3. Distribution regression results for short-time (one-second) A-V labels.

experiment, multiple examples from the same song must be either all in the training or testing set. In addition, as it is clear that some past data may be necessary to accurately determine the current emotional content, we include past features and investigate the optimal feature window length.

Given the similar performance of MLR and PLS in fusion methods, for our short time analysis we will restrict ourselves to only the MLR methods. The similarity in performance is likely due to the fact that in the multilayer combined system, both MLR and PLS are able to account for the correlation between label dimensions. In moving forward with time-varying regression, we wish to be able to apply all methods in real-time as a "virtual annotator" for MoodSwings. This directly addresses the bootstrapping problem inherent to the system in cases where multiple annotators are not available at the same time. A preliminary single-user version of MoodSwings called MoodSwings Single Player, [2] which demonstrates our real-time regression system, is available online.

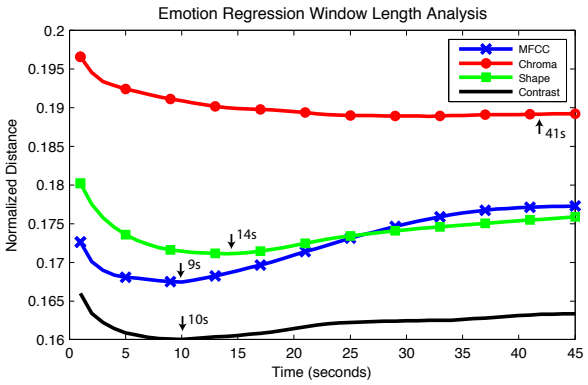


Figure 3. Window length analysis for different acoustic features.

For our time-varying approach, we seek to develop regressors that can predict the emotion for a single second using only current and past audio data. In terms of our data collection this implies that we have 15 distributions for each 15-second music clip (for 240 clips this yields a total of 3600 distributions). But we should also consider the optimal analysis window length for regression from each acoustic feature set. In Figure 3 we perform a regression analysis for each window length from 1 to 45 seconds (in increments of one second) and plot the average KL divergence from the projections to the collected distributions. As in previous experiments, the training/testing data is split 70%/30% and cross-validated 50 times. From the window length analysis in Figure 3, it can be seen that the optimal window length is not the same for all feature domains. For MFCCs we obtain the most accurate prediction using 13 seconds of past feature data, also 13 seconds for SSDs, 15 for spectral contrast, and 41 seconds for chroma. We use these feature window lengths in the regression analysis to follow.

In moving to short-time labels, it can be seen from Table 3 that our overall KL has increased, but our average distance ratings have mostly remained the same. This is most likely attributed to the fact that the underlying label covariance is less consistent due to the smaller quantity of collected A-V labels. Out T-test values have increased as well, which can be attributed to the overall increase in examples (from 240 to 3600). Considering our short-time degrees of freedom (1080 testing examples), our lowest T value (9.403) produces confidence of statistical significance (vs. randomly selected projections) higher than 99.999%. To visualize emotion regression over time, we have chosen three clips which display a clear shift in emotion distribution, plotting both the collected and projected distributions at one second intervals (Figure 4).

6. DISCUSSION AND FUTURE WORK

In working with highly subjective emotional labels, where there is not necessarily a singular rating for even the smallest time slice, it is clear that we can develop a more accurate system (in terms of predicting actual human labels) by representing the ground truth as a distribution. While accounting for potential dependence between the distribution parameters in the A-V emotion space seemed to be of high importance, some of the best performing techniques assumed total independence of parameters. In particular, combining MLR in multiple stages produces results comparable to more computationally complex methods.

One of our targeted applications, a "virtual annotator" to be used in MoodSwings, requires real-time calculation of projections, which also favors the simpler regression implementations. For the activity, the required degree of accuracy is questionable to begin with [11]. In our observations, we have found that it is more important for a virtual annotator to behave "realistically" (appropriate movement when the emotion changes) in order to keep a human participant engaged in the activity. But as we implement the

[2] MoodSwings Single Player: http://music.ece.drexel.edu/mssp

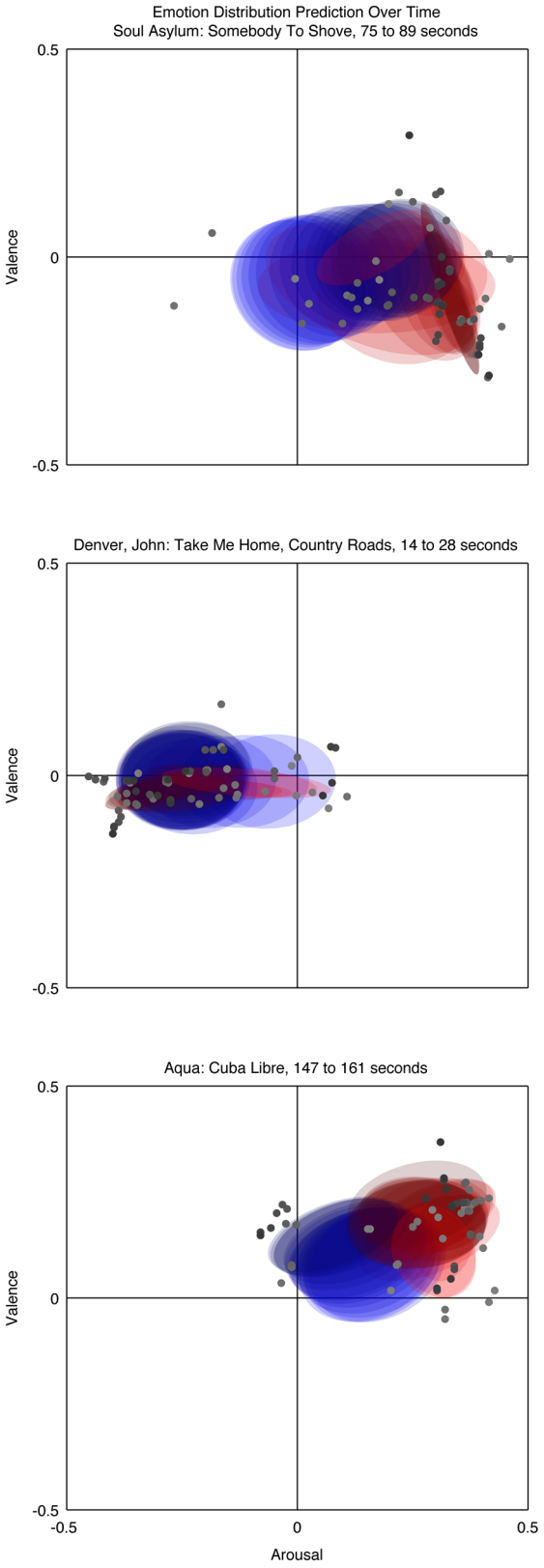


Figure 4. Time-varying emotion distribution regression results for three example 15-second music clips (markers become darker as time advances): second-by-second labels per song (gray •), Σ of the collected labels over 1-second intervals (red ellipse), and Σ of the distribution projected from acoustic features in 1-second intervals (blue ellipse).

virtual annotator to facilitate the collection of more human data, we hope to continue increasing the accuracy of our regression system.

7. ACKNOWLEDGMENTS

This work is supported by National Science Foundation award IIS-0644151.

8. REFERENCES

[1] E. M. Schmidt, D. Turnbull, and Y. E. Kim, "Feature selection for content-based, time-varying musical emotion regression," in *MIR '10: Proc. of the Intl. Conf. on Multimedia Information Retrieval*, Philadelphia, PA, 2010, pp. 267–274.

[2] B. Han, S. Rho, R. B. Dannenberg, and E. Hwang, "Smers: Music emotion recognition using support vector regression," in *Proc. of the 10th Intl. Society for Music Information Conf.*, Kobe, Japan, 2009.

[3] T. Eerola, O. Lartillot, and P. Toiviainen, "Prediction of multidimensional emotional ratings in music from audio using multivariate regression models," in *Proc. of the 10th Intl. Society for Music Information Conf.*, Kobe, Japan, 2009.

[4] Y. E. Kim, E. Schmidt, and L. Emelle, "Moodswings: A collaborative game for music mood label collection," in *Proc. of the 9th Intl. Conf. on Music Information Retrieval*, Philadelphia, PA, September 2008.

[5] R. E. Thayer, *The Biopsychology of Mood and Arousal*. Oxford, U.K.: Oxford Univ. Press, 1989.

[6] Y.-H. Yang, Y.-C. Lin, Y.-F. Su, and H. Chen, "A regression approach to music emotion recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 2, pp. 448–457, 2008.

[7] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, no. 4, pp. 357–366, 1980.

[8] T. Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music." in *Proc. of the Intl. Computer Music Conf.*, 1999.

[9] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *Speech and Audio Processing, IEEE Transactions on*, vol. 10, no. 5, pp. 293–302, 2002.

[10] D. Jiang, L. Lu, H. Zhang, J. Tao, and L. Cai, "Music type classification by spectral contrast feature," in *Proc. Intl. Conf. on Multimedia and Expo*, vol. 1, 2002, pp. 113–116.

[11] B. G. Morton, J. A. Speck, E. M. Schmidt, and Y. E. Kim, "Improving music emotion labeling using human computation," in *HCOMP '10: Proc. of the ACM SIGKDD Workshop on Human Computation*, Washinton, D.C., 2010.

Quantifying the Benefits of Using an Interactive Decision Support Tool for Creating Musical Accompaniment in a Particular Style

Ching-Hua Chuan
University of North Florida
School of Computing
Jacksonville, FL
chchuan@mail.barry.edu

Elaine Chew
University of Southern California
Viterbi School of Engineering
Thornton School of Music
Los Angeles, CA
echew@usc.edu

ABSTRACT

We present a human-centered experiment designed to measure the degree of support for creating musical accompaniment provided by an interactive composition decision-support system. We create an interactive system with visual and audio cues to assist users in the choosing of chords to craft an accompaniment in a desired style. We propose general measures for objectively evaluating the effectiveness and usability of such systems. We use melodies of existing songs by Radiohead as tests. Quantitative measures of musical distance – percentage correct and closely related chords, and average neo-Riemannian distance – compare the user-created accompaniment with the original, with and without decision support. Numbers of backward edits, unique chords explored, and repeated chord choices during composition help quantify composition behavior. We present experimental data from musicians and non-musicians. We observe that decision support reduces the time spent in composition, the number of revisions of earlier choices, redundant behavior such as repeated chord choices, and the gap between musicians' and non-musicians' work, without significantly limiting the range of users' choices.

1. INTRODUCTION

Building computer-assisted composition software tools has long been a common interest among researchers. Besides tools designed for professional use with advanced functions such as audio signal editing, many aim to help amateurs and music lovers write music and to enhance users' musical creativity. These tools often provide alternate ways for representing music, using interfaces such as a drawing pad to avoid score notation. While various systems have been proposed in recent years, it remains unclear how much these systems help the composition process. Evaluations of systems that support art creation often take the form of subjective opinion with varying degrees of success, and cannot be used to systematically study and improve the methodology. Music compositions, like other art forms, are often evaluated using the Consensual Assessment Technique [10], which measures quality according to experts' global and subjective assessment of the outcome. For studies that aim to design tools to support music composition, this technique may not provide sufficient detail for evaluating the systems. For example, it would be difficult to determine to what extent users' creations are improved by a composition assistance system using the Consensual Assessment Technique.

In this paper we focus on experiments that aim to quantifying the added benefit of style-imitating accompaniment composition decision-support systems. Such systems aim to help users create accompaniment to a melody in the style of a particular artist of band. When the goal is to imitate the style of a known song, the original accompaniment serves as ground truth for any user-created accompaniment to the song's melody. If the user is able to produce more chord patterns that are similar to the original accompaniment with automated assistance, it can then be concluded that the system achieves its design goal.

For the experiments described in this paper, we built a composition decision-support software tool by adapting an automated system for generating style-specific accompaniment. We designed an intuitive interactive user interface that assumes no formal music knowledge, using visual cues and sound to make music composition feasible for experts and novices alike. The system provides suggestions to the user at two levels: the system generates a sequence of stylistically appropriate chords as initial suggestions, a range of triads is then listed to guide users in each chord selection.

To measure the benefits of the system, we conducted a human-centered experiment with three composition tasks. In each task, participants were given a melody and were requested to complete the composition by choosing chords to accompany the melody in the style of a particular artist. Before the experiment, the participants listen to several songs by the artist they should emulate. In the first task, participants were given a composing interface without decision support; in the latter two tasks, they worked with a composing interface with decision-support that put forward sequences of computer-suggested chords. The experiment was designed to explore the composition process by examining whether participants could mimic an artist's composition decisions after having heard samples of their songs, and by determining how much their performance improved when they worked with a composition decision-support system.

We separated the participants into two groups – musicians and non-musicians – according to their musical backgrounds, and evaluated the effectiveness of the composition decision-support tool by examining whether non-musicians, with the help of the system, can create accom-

paniments that are as close to the original style as those by musicians. We propose quantitative metrics for evaluating the system's usability by statistically examining the changes in the users' composition behaviors with and without decision support.

2. RELATED WORK

In addition to the systems that are designed to model music composition [1, 4, 13, 15], efforts have been made to build tools to support music composition [2, 7, 8]. Unlike music production software that only provide tools for notating scores and recording sound, software such as Hyperscore [8] and Sonic Sketchpad [7] offer alternative modalities (e.g., drawing) through which to compose music. With this kind of software, users do not need to be familiar with, or knowledgeable about, music notation or instrument-specific characteristics. Evaluations of these systems focus on users' level of satisfaction with the software interface and its functionalities.

In particular, the automatic style-specific accompaniment (ASSA) system described in [5] aims to assist amateurs in songwriting by helping them compose music in the style they desire. The system incorporates both music theoretic knowledge and statistical learning, using small numbers of examples of users' favorite artists to model the accompaniment choices of that professional. It then applies the learned composition style to new, user-created melodies. The ASSA system has been evaluated based on subjective opinion through a Turing test, and on style-related quantitative metrics [6].

3. SYSTEM DESIGN

3.1 Interactive User Interfaces

In this study, the ASSA system [5] forms the basis of our composition decision-support tool. The interactive user interface allows users to create accompaniment to any melody through simple mouse-click actions on graphic icons. Users can explore different chords in each bar and listen to parts of the melody in any order with the accompaniment they create. Users can also adjust the tempo of the song to better fit their listening pace during composition. The system takes care of composition details such as voice arrangement and chord alignment, leaving users free to concentrate on using their aural perception to select the appropriate chords.

To test the improvement resulting from computer assistance in music composition, we created two interface designs. The first interface, shown in Figure 1, provides users with all possible chord choices at all times, i.e. without decision support. The chord collection, all 24 triads (major and minor), appears in the top panel of the interface. Triads are arranged according to the circle-of-fifths, with the major chord cycle on the left and the minor triad cycle relative to the major on the right. The

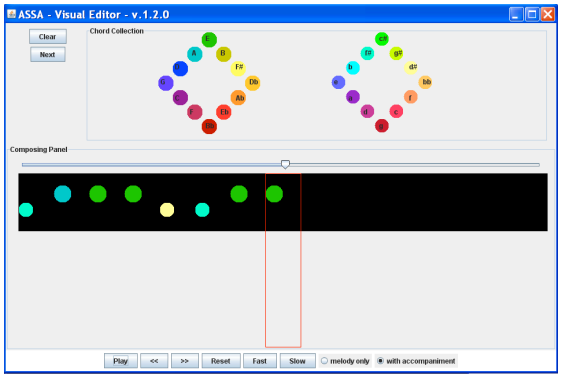


Figure 1. Composing interface without ASSA support.

Figure 2. Composing interface with ASSA support and graph of neo-Riemannian distance.

tonic triad is centered at the top. Each triad is assigned a different color to help users identify chords and the relationships between them. We used the color assignment described in [12], in which related colors are assigned to chords considered close one to another.

During composition, users can choose chords by clicking on one of the colored circles in the chord collection. Once a chord is selected, a corresponding triad with proper pitch arrangements is inserted in the accompaniment MIDI track, and a circle of the same color is placed in the composing panel, the black rectangle in the center of Figure 1. The composing panel allows users to visually examine the chord sequence they create. Major triads appear as larger colored disks, minor triads as smaller ones.

Figure 2 depicts the second interface. The most visible difference between this interface and the previous one is the graph at the bottom of the interface. Each point on the grid represents the neo-Riemannian distance [3] between adjacent chords. In neo-Riemannian chord space, chords are connected by neo-Riemannian operations (NROs), and the number of operations between chords reflects their musical distance. The user can select the time slice to be examined, and edit the neo-Riemannian distance they wish to consider at that time slice. Other chords within that number of NROs from the previous chord are displayed in the top panel. In this manner, the number of chord choices is reduced and constrained by the NRO

distance, which helps users categorize chords into subgroups that reflect their musical distance in the transition.

In addition to the use of neo-Riemannian distance, the user starts with an ASSA-produced chord sequence and its corresponding sequence of neo-Riemannian distances. The providing of this initial suggestion simplifies the process of creating accompaniments from scratch to one of editing the chords until the user is satisfied with the accompaniment.

4. EXPERIMENT DESIGN

Our study aims to determine the effectiveness of a composition decision-support system through quantitative comparisons of users' compositions, and of their decision pathways, with and without decision support. This section presents our experiments to further these goals.

4.1 Experiment Setup

We designed a three-part experiment using the two composition interfaces described in the previous section. In the first part of the experiment, participants were given a melody extracted from a song with which they are familiar, and were asked to compose a sequence of chords without decision support, using the interface shown in Figure 1. In parts two and three, participants were presented with the interface shown in Figure 2, and asked to compose accompaniments for two unfamiliar melodies in a familiar style with decision support.

We used songs by Radiohead, a British rock band, for the experiment. We chose Radiohead not only because of the band's popularity and reputation, but also because their unique style has been analyzed by music theorists in several book-length studies [9, 11, 14]. We selected 13 songs from Radiohead's albums *Pablo Honey* and *The Bends* as samples to familiarize participants with the band's style and as the training set for the ASSA system. As test melodies for the three-part experiment, we extract the melodies of three "hit" songs from the two albums, as indicated by their popularity ratings on the iTunes online music store. The participants' training samples included the test song for the first part, but not for the second and third parts. Each participant was asked to complete all three parts in the experiment. The idea behind these choices was to investigate whether participants, especially non-musicians, could create accompaniments for new melodies (the second and third tasks) in a style similar to the original with decision support.

4.2 Experiment Procedure

The participants for this study consisted of 26 volunteers: 8 musicians and 18 music lovers. The majority of the non-musician participants were college students from various disciplines. Participants who were categorized as musicians either majored in Music at a university or played an instrument in a rock band. Participants were requested to spend at least three days listening to the sample Radiohead songs provided, and to schedule an appointment for the experiment after they are familiar with the band's style.

The experiment was conducted in a computer lab, where each participant was assigned a computer and a headset to complete the task individually. Before beginning, participants first watched a 10-minute instructional video on how to use the two interfaces to create accompaniments. Before they started composing with the software, participants were reminded to choose chords that they think Radiohead would choose, based on their experience listening to music by the band. Participants were then left alone to compose without time constraints.

4.3 Evaluation Metric

This section describes the evaluation criteria designed to fulfill the aforementioned goals.

4.3.1 Effectiveness

The effectiveness of system assistance can be measured in many ways. If musicians as well as non-musicians take less time to complete their compositions with the decision support, we can say that the system helps users by speeding up the composition process.

Decision support effectiveness can also be measured in the improvement of musicians' and non-musicians' compositions produced with (vs. without) decision support, and of non-musicians' compositions with regard to their similarity to musicians' compositions. For instance, if non-musicians create accompaniments more similar to the ones by musicians with decision support than without, it can be claimed that the decision support helps non-musicians make more knowledgeable decisions to better reach the targeted music style.

To measure the quality of the user-created accompaniments, we compared them to that of the original production, as documented in the commercial sheet music. We counted the number of chords in the user-created accompaniment that are identical to the original, and divided it by the total number of chords to report the *same chord* percentage. For the chords in the user-created accompaniment that are different from the original, we further determined the percentage of chords that are closely related to the original. The metric, *chords-in-grid,* reports the percentage of chords that are related to the original by a Dominant (D), Subdominant (S), Parallel (P), Relative (R), or by one of the compound relations: DP, SP, DR, and SR. We also measured the *average NR distance* between the user-created and the original accompaniment to assess how dissimilar the user-chosen chords are from the original in terms of NROs.

4.3.2 Usability

Another goal of the study is to examine how the decision support system affects users' composition processes. We analyzed the changes in the way chords are selected (without decision support) or edited (with decision sup-

port) to investigate whether the decision support guides users in their compositions or limits their choices. Specifically, we focused on three types of behavior patterns: *backward editing*, *exploration*, and *hesitation*.

Backward editing occurs when users change the chord in a bar that is before the current one after they select a chord in the current position. For example, the following action sequence contains two backward edits at times T_2 and T_4, respectively:

Bar:	5	4	5	3
Chord selected:	C	Am	G	Dm
Time:	T_1	T_2	T_3	T_4

Backward editing indicates that the user has become uncertain about a previous choice they made because of a later decision.

We define *exploration* as the average number of unique chords that users have tried assigning in each bar, including the ones they choose and then changed later. For example, in the scenario above, two unique chords were explored in bar 5 at time T_1 and T_3 respectively. In contrast, *hesitation* is defined as the total number of repeated chord choices during the entire composition process. For instance, the following sequence contains a repeated choice at time T_4 in bar 6:

Bar:	5	6	6	6
Chord selected:	C	Am	C	Am
Time:	T_1	T_2	T_3	T_4

For each of the two groups, musicians and non-musicians, we calculated the means and standard deviations of the *backward editing*, *exploration*, and *hesitation* values with and without decision support. Suppose $\mu_{1,m,b}$ represents the true mean of musicians' backward editing without decision support and $\mu_{2,m,b}$ the true mean with decision support. We conducted a hypothesis test on whether decision support has a positive impact on the behavior pattern (i.e. reduces backward editing) by setting the null hypothesis, H_0, and the alternative hypothesis, H_1, as:

$$H_0: \mu_{1,m,b} \leq \mu_{2,m,b}$$
$$H_1: \mu_{1,m,b} > \mu_{2,m,b} \quad (1)$$

We calculated the observed level of significance, the p-value, using the Student's t-distribution with the calculated means and standard deviation. We chose the Student's t-distribution over the normal distribution due to the small sample size. We then compare the p-value with the standard 5% significance level. Similar hypothesis tests were conducted as well for the exploration and hesitation mean values.

5. EXPERIMENT RESULTS AND ANALYSES

In this section, we report the results of the experiments, and describe the analyses of these results. Twenty-six participants volunteered for the experiment, each spending approximately about one hour in completing the tasks. These users, and their composition results, are separated into two groups, namely musicians and non-musicians, according to their musical backgrounds.

5.1 Effectiveness: Inter-Group Comparisons

This section presents analyses of composition decision support effectiveness within the musician and non-musician groups.

5.1.1 Task Completion Time

Table 1 lists the average time musicians and non-musicians spent on each composition task. For song 1, which was included in the participants' training samples, musicians and non-musicians spent an almost equal amount of time choosing the 15 chords for the accompaniment without decision support. For songs 2 and 3, which were new to the participants, non-musicians spent a slightly shorter amount of time generating an accompaniment with an initial computer-produced suggestion followed by edits guided by the neo-Riemannian interface. If we calculate the time spent on each chord in the accompaniment, we can observe that both musicians and non-musicians spent significantly less time on the second and third task than on the first.

Time (min)	Song 1 (15 chords)	Song 2 (36 chords)	Song 3 (30 chords)
Musicians	17.54	20.59	12.76
Non-musicians	17.51	14.13	7.77

Table 1. Time spent on each accompaniment task.

5.1.2 Created Accompaniment versus the Original

Figure 4 shows the mean *same chord*, *chords in grid*, and *average NR distance* percentages for the user-created accompaniments. The results of accompaniments produced with decision support are the average of that for song 2 and song 3.

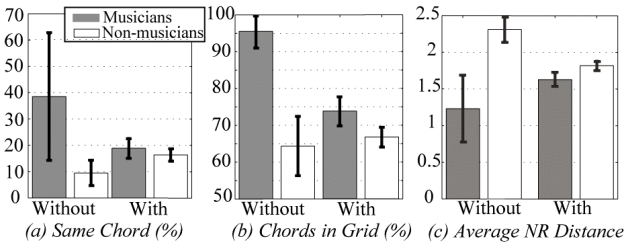


Figure 4. Evaluation of user-created accompaniments.

Figure 4(a) shows that the average percentage of chords in the user-created accompaniment that are identical to those in the original accompaniment is much higher in compositions by musicians than in those by non-musicians. This performance gap between musicians and non-musicians is greatly reduced with the assistance of the ASSA system. Similar patterns can be observed in Figure 4(b) with the *chords-in-grid* metric and in Figure 4(c) with the *average NR distance* metric.

If we focus on the musicians' results, we will find that, even with decision support, musicians performed worse on songs 2 and 3, when they are unfamiliar with the test song, than on song 1, when they had heard the song in its entirety. This finding implies that it is more difficult for musicians to come up with exactly the same chords as the original if they have never heard the song before. In contrast, non-musicians chose more suitable chords for songs 2 and 3, in collaboration with the system, than they did for song 1. With decision support, as in songs 2 and 3, the performance of non-musicians approached that of musicians.

Note that in the bars labeled *without* in Figures 4(a) and 4(b), the standard deviation for the percentage of *same chords* for the musicians' accompaniments is much greater than that for *chords-in-grid* for the same group. This difference is due to the fact that the musician group consists of classical music majors as well as rock band members. It appears that imitating Radiohead's style is a challenge for some classical musicians who were less familiar with the band than rock musicians. However, because of their training in music theory, their choices are closely related to the original as the high *chords-in-grid* values suggest.

5.2 Usability: Intra-Group Comparisons

This section presents analyses of user composition patterns, with and without decision support, comparing the results of musicians against that of non-musicians.

5.2.1 Backward Editing

Figure 5 presents the average number *backward edits* with and without decision support for (a) musicians and (b) non-musicians. Note that for both musicians and non-musicians, decision support reduces the amount of *backward editing*. The results are confirmed statistically by the p-values; both the p-values for musicians (p-value = 0.0066) and non-musicians (p-value = 0.015) are less than 0.05, indicating that the reduction in backward edits is statistically significant.

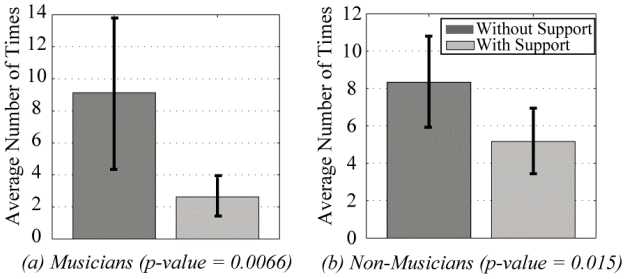


Figure 5. Average backward edit counts with and without the decision support.

5.2.2 Exploration

Figure 6 shows the average number of unique chords explored in each bar for (a) musicians and (b) non-musicians, with and without the decision support.

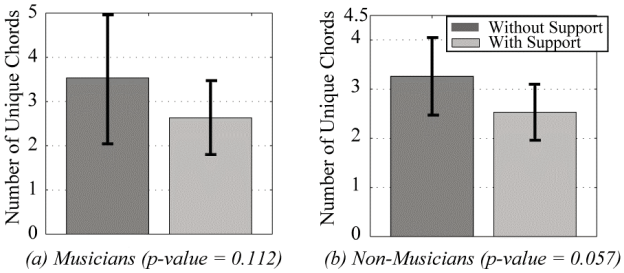


Figure 6. Average number of unique choices explored in each bar with and with the decision support.

While both musicians and non-musicians explore fewer numbers of unique chords on average with system support, the p-values are higher than 0.05. Thus, the observed reduction in unique chords explored is not statistically significant, and we conclude that the system does not significantly limit participants' exploration.

5.2.3 Hesitation

The average number of repeated choices in each bar with and without decision support are shown in Figure 7 for (a) musicians and (b) non-musicians. It can be observed that the number of repeats is reduced when the system provides suggestions during composition, as reflected in the figures as well as the p-values. The result implies that the system helps musicians and non-musicians reach their goals with less confusion.

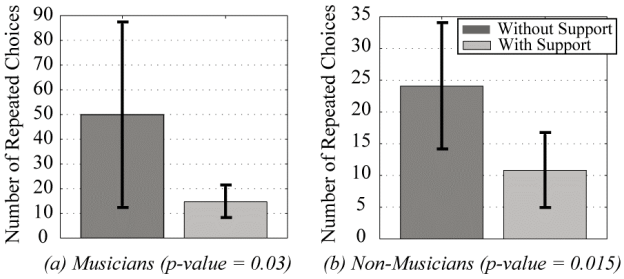


Figure 7. Average number of repeated choices in each bar with and without decision support.

6. CONCLUSION AND FUTURE WORK

In this paper we described an interactive decision support system that aims to assist amateurs in the creating of accompaniments in a desired style. The system uses visual cues to offer chord suggestions and sounds to evaluate chord choices, making composition feasible for people without formal musical training. To investigate the benefits associated with the decision support, we designed three composition tasks. In the first task, participants were provided with the interface without decision support, and asked to create an accompaniment for a melody of a familiar song. In the latter two tasks, participants were given the interface with decision support, and instructed to produce accompaniments for unfamiliar melodies in a familiar style.

With the data obtained in the experiment, we first analyzed how well the user-created accompaniments concurred with the desired style, separating the results by user group, using the quantitative measures: same chord and closely-related chord percentages, and average neo-Riemannian distance. We observed that when decision support was present, the performance gap between musicians and non-musicians was greatly reduced.

To understand the usability of the system, we proposed methods to measure the changes in participants' composition behavior. Statistical analyses revealed that when decision support was available, the number of backward edits and repeat choices were reduced, and the range of unique chords explored was not significantly limited by the system.

Additional analyses that can be performed on the data include examining whether common or distinct decision patterns exist between the two user groups, and separating the support given by the initial ASSA-produced sequence from the assistance provided by the interactive interface and differentiating between them.

In the future, we plan to extend the experiment by considering other musical factors such as timbre and instrument arrangement to investigate the impact of different stimuli on musical creativity. We will refine the interface and conduct studies with larger groups of users, and design questionnaires to obtain their feedback on the system. Last but not least, we will further study the degree to which automation affects music creativity so as to design better composition decision support tools.

7. ACKNOWLEDGEMENTS

We thank the volunteers who participated in the experiment, including students and alumni of the University of Southern California and of Barry University. This work was supported in part by NSF Grant No. 0347988. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors, and do not necessarily reflect those of the NSF.

8. REFERENCES

[1] M. Allan and C. K. I. Williams: "Harmonising Chorales by probabilistic inference." *Proceedings of the Neural Information Processing Systems Conference*, Vancouver, pp. 25–32, 2004.

[2] J. Bamberger: "Developing Musical Intuitions: A Project-Based Introduction to Making and Understanding Music." Oxford University Press, USA.

[3] G. Capuzzo: "Neo-Riemannian theory and the analysis of pop-rock music." *Music Theory Spectrum*, Vol. 26, No. 2, pp. 177–199, 2004.

[4] H. Chan and D. Ventura: "Automatic Composition of Themed Mood Pieces." *Proceedings of the 5th International Joint Workshop on Computational Creativity*, Madrid, pp. 109–115, 2008.

[5] C. H. Chuan and E. Chew: "A Hybrid System for Automatic Generation of Style Specific Accompaniment." *Proceedings of the 4th International Joint Workshop on Computational Creativity*, London, pp. 57–64, 2007.

[6] C. H. Chuan and E. Chew: "Evaluating and Visualizing Effectiveness of Style Emulation in Musical Accompaniment." *Proceedings of the 9th International Conference on Music Information Retrieval*, Philadelphia, pp. 57–62, 2008.

[7] T. Coughlan and P. Johnson: "Interaction in Creative Tasks: Ideation, Representation and Evaluation in Composition." *Proceedings of ACM Conference on Human Factors in Computing Systems*, Montréal, Canada, pp. 531-540, 2006.

[8] M. M. Farbood, H. Kaufman, and K. Jennings: "Composing with Hyperscore: an Intuitive Interface for Visualizing Musical Structure." *Proceedings of International Computer Music Conference*, 2007.

[9] D. Griffiths: Radiohead's OK Computer (Thirty Three and a Third series). Continuum, New York, 2004.

[10] M. Hickey: "More or Less Creative? A Comparison of the Composition Processes and Products of 'Highly-Creative' and 'Less-Creative' Children Composers." www.faculty-web.at.northwestern.edu/music/hickey/Exeter/Exeter.pdf.

[11] N. Hubbs: "The Imagination of Pop-Rock Criticism." *Expression in Pop-Rock Music: Critical and Analytical Essays* (2nd edition). Routledge, New York, pp. 215–237, 2008.

[12] A. Mardirossian and E. Chew: "Visualizing Music: Tonal Progressions and Distributions." *Proceedings of the 8th International Conference on Music Information Retrieval*, Austrian Computer Society, Vienna, pp. 189–194, 2007.

[13] M. T. Pearce and G. A. Wiggins: "Evaluating Cognitive Models of Musical Composition." *Proceedings of the 4th International Joint Workshop on Computational Creativity*, London, pp. 73–80, 2007.

[14] J. Tate: *Music and Art of Radiohead*. Ashgate, Aldershot UK, 2005.

[15] N. Vempala and S. A. Dasgupta: "Computational Model of the Music of Stevie Ray Vaughan." *Proceedings of the 6th Creativity and Cognition Conference*, Washington D.C., pp. 203–212, 2007.

QUERY-BY-CONDUCTING: AN INTERFACE TO RETRIEVE CLASSICAL-MUSIC INTERPRETATIONS BY REAL-TIME TEMPO INPUT

Akira Maezawa,[†] Masataka Goto[‡] and Hiroshi G. Okuno[†]

[†]Dept. of Intelligence Science and Technology
Graduate School of Informatics, Kyoto University
Sakyo-ku, Kyoto 606-8501 Japan
{amaezaw1, okuno}@kuis.kyoto-u.ac.jp

[‡]National Institute of Advanced Industrial
Science and Technology (AIST)
Tsukuba, Ibaraki 305-8568 Japan
m.goto@aist.go.jp

ABSTRACT

This paper presents an interface for finding *interpretations* of a user-specified music, *Query-by-Conducting*. In classical music, there are many interpretations to a particular piece, and finding "the" interpretation that matches the listener's taste allows a listener to further enjoy the piece. The critical issue in finding such an interpretation is the way or interface to allow the listener to listen through different interpretations. Our interface allows a user, by swinging a conducting hardware interface, to conduct the desired global tempo along the playback of a piece, at any time in the piece. The real-time conducting input by the user dynamically switches the interpretation being played back to the one closest to how the user is currently conducting. At the end of the piece, our interface ranks each interpretation according to how close the tempo of each interpretation was to the user input.

At the core of our interface is an automated tempo estimation method based on audio-score alignment. We improve tempo estimation by requiring the audio-score alignment of different interpretations to be consistent with each other. We evaluate the tempo estimation method using a solo, chamber, and orchestral repertoire. The proposed tempo estimation decreases the error by as much as 0.94 times the original error.

1. INTRODUCTION

Classical music is unique in that many audio recordings exist for a given piece of music. For example, as of March 2010, a search on an on-line shopping site for "Mendelssohn Violin Concerto" returns 1200+ hits, or that of "Beethoven Spring Sonata" returns 300+ hits. Each of these recordings is an acoustic rendition of a particular music score, embodied by an unique interpretation of the performer. Finding an interpretation that matches the listener's taste is an important aspect of enjoying classical music. However, searching for such recording is tiresome because it requires the listener to listen through the same piece many times.

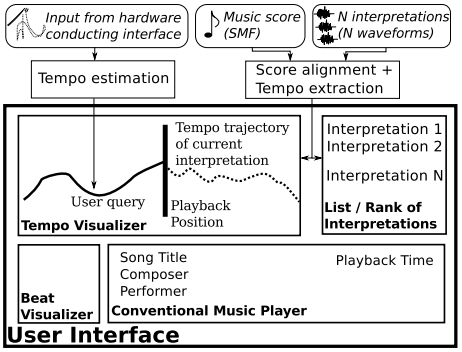


Figure 1: System diagram of *Query-by-Conducting*.

Our goal is to retrieve *interpretations*[1] on the basis of various aspects of music interpretation, similar to content-based music information retrieval (CBMIR), which retrieves *pieces* on the basis of various aspects of music such as rhythm and timbre [2, 3]. What constitutes musical interpretation is a difficult question, though it seems that musicians express interpretation by manipulating the tempo, the timbre, or bringing out interesting melodic lines. This paper focuses on the global tempo – the average tempo of a piece over a few beats. We believe tempo is an aspect of music interpretation that many listeners take note of. Studies in music cognition suggest that similarity of interpretations is strongly reflected in global tempo [4], and many studies are motivated by the significance of tempo on interpretation [1, 5].

We present *Query-by-Conducting*, a new interface for finding *interpretations* of a user-specified music by conducting the global tempo. The interface reads, as the music score, a standard MIDI file of a piece of music, and different interpretations of the music score as audio files. The interface facilitates playback, visualization and query of interpretations by supporting the following features, as shown in Figure 1:

1. Visualization of global tempi of the interpretations
2. Hardware conducting interface (a *Nintendo Wii* remote) for intuitively entering, along the playback of a piece, the user's tempo query in real time
3. Ranking and retrieval of interpretations on the basis of the similarity between each interpretation and the current tempo query entered by the user.

Visualization allows the user to view the range of interpretations available, and thus, the valid range of tempo

[1] We shall use the term "interpretation" to mean a rendition of a particular symbolic representation of music, as per [1].

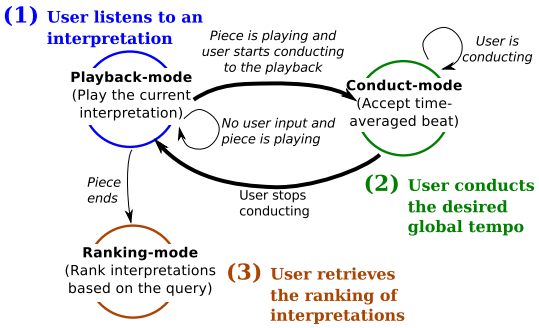


Figure 2: System-level state diagram. The interface alternates between (1) and (2) until the piece ends.

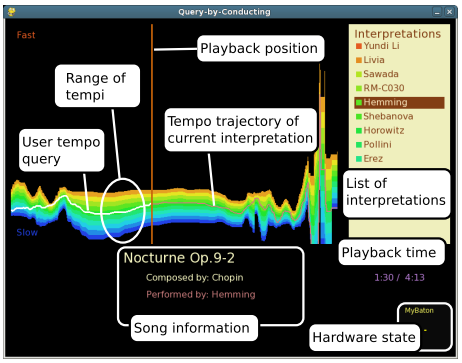


Figure 3: The interface in **playback-mode** visualizes the tempo along playback of an interpretation.

in which the user should conduct to obtain a meaningful query. As a particular interpretation is played back, the user may become dissatisfied with its tempo. Then, the user would "conduct" the desired tempo using the hardware conducting interface. Based on the user's tempo input, the interface retrieves and switches the interpretation being played back to one that is closest to the input. The user may stop conducting if the current interpretation is satisfactory. At the end of playback, the system ranks each interpretation on the basis of how similar the overall global tempo trajectory was to the user's conducting.

Our interface differs from existing conducting interfaces [6–9] in three respects. First, we use conducting device to switch the interpretation being played back, instead of specifying the tempo of the entire piece. The user has the freedom of either listening to a piece, or conducting the tempo of an interpretation that the user wants to listen. Second, our method allows the user to control only the global tempo instead of local tempo or dynamics. We believe that such restriction is an effective way to retrieve a particular interpretation; global tempo is easy for a typical user to specify, but specifying local tempo requires a precise control of the tempo. The notion of restricting the user control to a few dimensions has been proposed in other studies aimed at easily manipulating expressive music [10]. Finally, our conducting interface is meant to retrieve a particular interpretation to play back, whereas most conducting interfaces are aimed at real-time temporal manipulation of a particular audio signal. Unlike query by tapping [11], which uses *rhythm pattern* as the query, as our method uses the *tempo* as the query.

The interface relies on tempo estimation that is determined through audio-score alignment. In existing studies [1, 12–14], audio-score alignment was created using only the information obtained from the audio of interest and the score. There may be errors in the alignment, but given one alignment, there is no way of knowing where an error is. When aligning multiple interpretations, however, it is also possible to create audio-score alignment by aligning the score to some *other* audio, and then aligning that audio to the audio of interest. Thus, given N interpretations to align, N unique audio-score alignments to *one* interpretation can be generated. We use these multiple audio-score alignments generated to estimate the true audio-score alignment that is error-free.

A video demonstration of our interface is available at http://www.youtube.com/QueryByConducting

2. INTERFACE DESIGN

The interface offers functions in a conventional music playback interface such as playback and rewind. Moreover, it features a visualizer of global tempi of various interpretations, a conducting hardware to enter the global tempo query in real time, and retrieval of interpretations on the basis of the user query. As shown in the state diagram in Figure 2, our system alternates between playing back the current interpretation ("playback-mode"), and accepting user conducting and retrieving appropriate interpretation to play back ("conduct-mode"). At the end, our system ranks each interpretation on the basis of the tempo ("ranking-mode").

Figure 3 shows the interface during playback (**playback-mode**). Bottom of the screen displays the title, the performer and the playback time, similar to conventional music playback interface. Top of the screen visualizes the global tempi, and presents each interpretation sorted in descending order of the current global tempo. Bottom right shows the state of the conducting interface.

As the piece is played back, the user may become dissatisfied with the tempo of the piece ("I liked the introduction, but the development section is too slow," a user might think). As shown in Figure 4, the interface allows a user to "conduct" the desired global tempo in **conduct-mode**, in real time. In **conduct-mode**, the interface accepts beat input from the conducting hardware interface, and also visualizes the beat at the bottom left to facilitate proper conducting. The entered tempo is used as a query to retrieve the interpretation whose global tempo is closest to what the user conducts, and to switch the current playback to it. This mode offers the user an active listening experience by constantly retrieving and cross-fading the playback to interpretation that plays like how the user is conducting.

At the end of the piece, the interface enters the **ranking-mode**, and ranks each interpretation based on how similar each interpretation was to the user's overall conducting. The ranking is presented to the user.

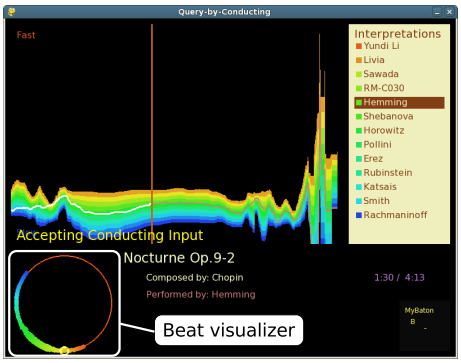


Figure 4: The interface in **conduct-mode** accepts user's conducting using a controller, and switches interpretations being played back. Beat visualizer facilitates user's conducting.

2.1 Interpretation Visualizer

Top half of the interface (in Figure 3) is the interpretation visualizer. It shows the tempi of different interpretations, along with tempo trajectory of the interpretation that is being played back and the user query.

Figure 5 shows the visualizer in further detail. It presents tempo information against time. To allow the user to view the detailed tempo near the current playback position as well as the tempo of the entire piece, we distort the normalized x-coordinate, $x(t)$. $x(t)$ is distorted such that the vicinity of current playback position t_c is zoomed like a lens. Let t, the current beat of playback, be defined for $[0, t_l]$, where t_l is the duration of the piece, and the range of $x \in [0, 1]$. Then, the visualizer applies the following function:

$$x(t) = \frac{t}{2t_l} + \frac{1}{2}\frac{\exp\left(\frac{t-t_c}{T}\right)}{\exp\left(\frac{t-t_c}{T}\right)+1} \quad (1)$$

This shows approximately T-neighborhood of the current playback position in more detail than the rest. T is chosen to be four quarter notes.

A vertical straight line indicates the current playback position. Line segment to the left of the current playback position is the past tempo trajectory of the user's global tempo query. Line segment to the right of the current playback position is the future tempo trajectory of the interpretation that is being played back. This way, the user is able to view the query entered so far, and how the current interpretation will unfold.

To show the range of possible interpretations, the range of global tempi is expressed as a colorful strip, overlayed to to the line segments described above. The strip is colored using a gradation of hue angle, such that fast tempo is associated with small hue (orange), and slow tempo with large hue (blue). At beat t, given the slowest tempo $\tau_{\min}(t)$, fastest tempo $\tau_{\max}(t)$, and some tempo in between, $\tau(t)$, we set the hue to the following angle:

$$\text{hue}(t) = 240^\circ - \frac{\tau(t) - \tau_{\min}(t)}{\tau_{\max}(t) - \tau_{\min}(t)} 230^\circ \quad (2)$$

Right half of the visualizer prints the performer of each interpretation, sorted in descending order of the current global tempo. The interpretation that is being played back is highlighted. Next to each name, a box whose hue value is as described in Equation (2) is painted.

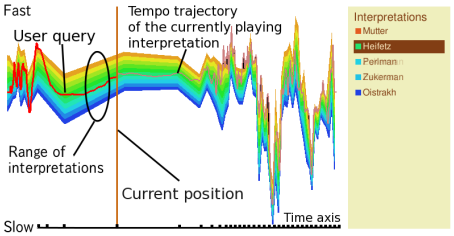


Figure 5: Interpretation visualizer shows the tempo trajectory of the current interpretation, how the user has conducted, and the range of tempi.

2.2 Hardware Conducting Interface

The hardware conducting interface detects beat from an accelerometer embedded in the hardware controller, and converts it into tempo. The user interface shows beat visualizer to facilitate tempo entry.

2.2.1 Beat detection

We accept the user's conducting query using a game controller that features a 3-axis accelerometer (a Nintendo Wii controller). Our system detects beat by checking for peaks in the axis vertical to the controller. Such peak is generated when the controller is flicked up, as a conductor would flick the baton to indicate the beat.

Once a beat is detected, the accelerometer input is ignored for 200msec to prevent false triggering. Therefore, our system accepts tempo of up to 300 beats-per-minute (BPM), which is sufficient for virtually all classical music.

2.2.2 Converting beat input to tempo query

To specify a new tempo, the user must conduct a tempo different from the playback. We observed that people tends to conduct *not* in the desired tempo, but instead ahead or behind of the beat of the playback to indicate faster or slower tempo relative to the current playback. We conjectured that such phenomenon occurs because people are distracted by the downbeat of the playback, and sets the desired beat location *relative* to the last downbeat he/she has heard. Therefore, we convert the offset of the user's conducting with respect to the beat of the playback, to the desired tempo in BPM. Suppose the user conducts Δ_t behind the beat. Then, supposing the current BPM of the playback is BPM_0, we convert Δ_t to user-specified tempo, BPM, as follows:

$$BPM = BPM_0 \frac{1}{\frac{\Delta_t}{60/BPM_0} + 1} \quad (3)$$

The average of user-specified BPM over four beats is used as the query.

2.2.3 Beat visualizer

We observed that, in a preliminary experiment using a few test subjects, people did not always have a clear sense of rhythm, and had trouble finding where the beat is. This was especially true for music whose instrumentation did not include instrument with strong attack and decay, such as the piano or plucked strings.

To facilitate tempo input, we display a beat visualizer, as shown at the bottom left of Figure 3, and in detail in

Figure 6: Visualizer for facilitating tempo input. The color bar rotates in synchrony with the beat so that the user could easily grasp the downbeat.

Figure 6. The visualizer has a colored stripe that rotates around the large circle, in synchrony with the beat of the playback. At each downbeat, the stripe crosses the small circle at the bottom. The arc-length of the rotating stripe corresponds to the range of global tempi at the current beat, and the hue is calculated using Equation (2). Therefore, if the user wants to switch to fast interpretation, for example, from tempo shown as green in the tempo visualizer to orange-colored tempo, the user could flick the controller as the orange-colored segment of the arc crosses the small circle at the bottom.

2.3 Interpretation Retriever

After the user has finished listening through a piece, the interface ranks, in **ranking-mode**, each interpretation on the basis of the similarity between the tempo trajectory of the interpretation and the user query.

We use the tempo trajectories of the interpretations that were played as the query. For example, if the user listened to interpretation x for the first minute and y for the next two minutes, our query would consist of the tempo trajectory of interpretation x for the first minute and y for the next two.

Let us define the dissimilarity score of the user query and each interpretation. Let $\tau_i(t)$ be the global tempo trajectory of the ith interpretation, and $\tau_q(t)$ be the query tempo trajectory. Then, we define tempo dissimilarity for interpretation i, r_i as follows:

$$r_i = \frac{1}{T}\int_0^T \left(\frac{\tau_i(t) - \tau_q(t)}{\tau_q(t)}\right)^2 dt \tag{4}$$

The interpretations are sorted in the ascending order of tempo dissimilarity, as shown in Figure 7. A transparency value is associated to each interpretation being drawn, such that interpretation with lowest dissimilarity is opaque, and the highest transparent. Moreover, dissimilarity measure that is inverted, shifted and scaled between 0 and 1 is shown next to each interpretation.

3. TEMPO EXTRACTION METHOD

Global tempo extraction is based on evaluating the audio-score alignment. Since accurate alignment is essential for accurate tempo estimation, we propose a method to improve the audio-score alignment.

3.1 Initial Audio-Score Alignment

The initial audio-score alignment is based on dynamic time-warping (DTW) using chroma vector as the feature, similar to other works [13–15].

Let $c_k^{(t)}$ be a 12 dimensional vector that contains the chroma vector computed for tth audio frame of the kth

Figure 7: **Ranking-mode** ranks each interpretation based on the tempo similarity, presents similar interpretations as opaque and dissimilar ones transparent.

interpretation. Let $c_S^{(t)}$ be a 12 dimensional chroma vector computed from the music score at tick t. We generate the alignment from the score to the kth interpretation, denoted $M_{k\leftarrow s}$, or from interpretation i to j, denoted $M_{j\leftarrow i}$. To generate $M_{k\leftarrow s}$, a similarity matrix $R_{k\leftarrow s}$ is first computed. Let N_s be the number of ticks in the music score and N_k be the number of audio frames contained in interpretation k. Let $R_{k\leftarrow s}$ be a N_s-by-N_k matrix, whose element i, j contains:

$$R_{k\leftarrow s}(i,j) = 1 - \frac{c_s^{(i)} \cdot c_k^{(j)}}{\|c_s^{(i)}\| \cdot \|c_k^{(j)}\|} \tag{5}$$

Next, we find the alignment path using DTW. Formally, we define a cost matrix N_s-by-N_k matrix, as follows:

$$C_{k\leftarrow s}(i,j) = R_{k\leftarrow s}(i,j) + \min \begin{cases} C_{k\leftarrow s}(i-1,j) \\ C_{k\leftarrow s}(i,j-1) \\ C_{k\leftarrow s}(i-1,j-1) \end{cases} \tag{6}$$

where for all t, $C_{k\leftarrow s}(t,-1) = C_{k\leftarrow s}(-1,t) = 0$. Next, we determine the parametric representation of the audio-score alignment, $M_{k\leftarrow s}^{(t)}$, by backtracking the cost matrix. First, we set $M_{k\leftarrow s}^{(0)}$ to (N_s, N_k), and update in the following manner while incrementing t, until $M_{k\leftarrow s}^{(t)} = (0,0)$:

$$\begin{aligned} M_{k\leftarrow s}^{(t+1)} &:= \operatorname*{argmin}_{(I,J)\in S} C_{k\leftarrow s}(I,J) \\ S &= \{(i-1,j),(i,j-1),(i-1,j-1)\} \end{aligned} \tag{7}$$

where $(i,j) = M_{k\leftarrow s}^{(t)}$. Audio-audio alignment from interpretation i to j, $M_{j\leftarrow i}$, can be achieved in the same way, by computing the similarity matrix between chroma vector sequence of interpretation i and j.

3.2 Improving Audio-Score Alignment

We improve audio-score alignment by requiring the alignments of different interpretations to be consistent with each other. Given one music score and N interpretations, there are N possible paths to generate the alignment from the music score to interpretation i, as shown in Figure 8. Namely, in addition to the direct mapping from the score to interpretation i, it is also possible to generate mapping from the score to interpretation j (audio-score alignment), which is then mapped by using the map from interpretation j to i (audio-audio alignment). Ideally, all N paths from the score to an interpretation should be identical. In reality, however, they are not because they are generated using different similarity matrices.

In order to generate a map from the score to some interpretation i via interpretation j, $M_{i\leftarrow j} \circ M_{j\leftarrow S}$, both $M_{j\leftarrow S}$ and $M_{i\leftarrow j}$ must be one-to-one, but the alignments generated in the previous section are not.

Therefore, we trace, over the alignment determined in the previous section, a new map that is one-to-one. We perform the following procedure for each alignment between some interpretation (or score) s and k:

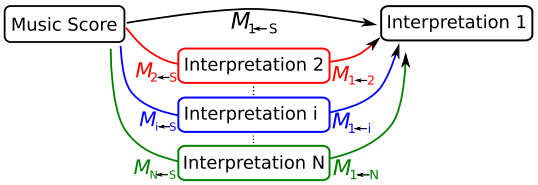


Figure 8: By combining two alignments, there are multiple ways to align the score to an interpretation.

1. Set $t = 0$, and the initial point of the refined alignment $\tilde{M}^{(t)}_{k\leftarrow s}$ to $(0, 0)$.
2. For $\epsilon < \theta < \frac{\pi}{2} - \epsilon$, compute the following cost function:
$$c(\theta) = E_{q\sim \exp(-3q/Q)}[\min_n d^{(t)}_n(q, \theta)] \quad (8)$$
where
$$d^{(t)}_n(q, \theta) = \|\tilde{M}^{(t)}_{k\leftarrow s} + q\,(\cos(\theta), \sin(\theta)) - M^{(n)}_{k\leftarrow s}\| \quad (9)$$
Q is chosen to be 20 frames, and ϵ to be $\pi/20$ radian. θ is evaluated every $\pi/20$ radians.
3. Update $\tilde{M}_{k\leftarrow s}$ as follows, for some $\Delta r \in (0, 1]$:
$$\tilde{M}^{(t+1)}_{k\leftarrow s} := \tilde{M}^{(t)}_{k\leftarrow s} + \Delta r\left(\cos(\hat{\theta}), \sin(\hat{\theta})\right) \quad (10)$$
$$\hat{\theta} = \operatorname*{argmin}_\theta c(\theta)$$
We chose $\Delta r = 1$ frame.
4. Exit if $\tilde{M}^{(t)}_{k\leftarrow s} \cdot (1, 0) \geq N_s$ or $\tilde{M}^{(t)}_{k\leftarrow s} \cdot (0, 1) \geq N_k$.
5. Set $t := t + 1$, and go to 2.

We assume that observed alignments are corrupted by independent and identically distributed noise that follows the Laplace distribution with location parameter $\hat{M}_{i\leftarrow S}(t)$ and scale parameter b, for each beat t:
$$p(t) = \exp\left(-\|M_{i\leftarrow S}(t) - \hat{M}_{i\leftarrow S}(t)\|/b\right)/2b \quad (11)$$
and likewise for $M_{i\leftarrow j} \circ M_{j\leftarrow S}$ for $j \neq i$. We interpret $\hat{M}_{i\leftarrow S}$ as the underlying "correct" alignment that generates $M_{i\leftarrow S}$ and $M_{i\leftarrow j} \circ M_{j\leftarrow S}$. Since an estimator of $\hat{M}_{i\leftarrow S}$ is the sample median, we update $M_{i\leftarrow S}$ as follows:
$$M_{i\leftarrow S}(t) := \text{median}(\{M_{i\leftarrow j} \circ M_{j\leftarrow S}(t)\}) \quad (12)$$
As will be shown in the experiment, iterating this step yields in improved alignment accuracy.

3.3 Tempo Extraction

The tempo is estimated by determining the slope of the audio-score alignment. We compute the tempo at MIDI tick t using alignment information obtained between tick $t - T$ to $t + T$ for $T > 0$. Only information at note onsets are used, as alignment results between two note onsets are not reliable. We choose T dynamically such that at least 20 audio frames that correspond to note onsets are within this range. Let $(s(p), a(p))$ contain a parametric representation of $M_{i\leftarrow S}$ that contain the audio frames chosen. s corresponds to the domain (tick of note onsets) and a the range (audio frame). Then, we compute the BPM at tick t, $\tau(t)$ by first finding the slope $m(t)$ of $(s(p), a(p))$ using linear regression, and multiplying its inverse by a scalar factor:
$$\tau(t) = \frac{1}{m(t)} \frac{\text{audio frame-per-minute}}{\text{ticks-per-beat}} \quad (13)$$

4. EXPERIMENTS

We evaluate the tempo estimation method, and retrieval of interpretation on the basis of global tempo query. We analyzed nine classical pieces of varying instrumentation. Of nine pieces, three are orchestral (denoted *orch-1* to *orch-3*), three are written for small ensemble (denoted *duo-1* to *duo-3*), and three are solo piano (denoted *solo-1* to *solo-3*). For each work, multiple interpretations (between four and thirteen) were obtained and their ground truth tempo data were entered using an in-house tempo entry utility.

Table 1: Average MSE (mean-squared error) improvement in thousandths (10^{-3}) after iterating Equation (12).

Piece (No. Interp.)	None	Iter. 1	Iter. 2	Iter. 10
solo-1 (13)	8.9	8.6	8.4	8.4
solo-2 (6)	17.3	15.0	13.0	12.7
solo-3 (5)	266.7	73.1	85.4	98.8
duo-1 (5)	4.5	3.9	3.7	3.8
duo-2 (4)	34.8	22.1	20.6	20.4
duo-3 (4)	185.4	12.5	10.2	10.2
orch-1 (5)	646.8	54.4	47.2	44.9
orch-2 (5)	231.5	14.7	13.3	13.2
orch-3 (5)	3941.6	1091.4	1038.3	833.2

4.1 Evaluation of Audio-Score Alignment

Let $\tau_g(t)$ be the ground truth tempo trajectory. Given an estimated tempo trajectory $\hat{\tau}(t)$, we evaluate the error using scaled mean squared error (MSE), defined as follows:
$$\text{MSE} = \frac{1}{T}\int_0^T \left(\frac{\tau_g(t) - \hat{\tau}(t)}{\tau_g(t)}\right)^2 dt \quad (14)$$
MSE can be considered as the dissimilarity measure between the ground truth and the estimated tempo.

Table 1 shows the average of MSE over all interpretation for each of the nine pieces, as the number of iterations of the update step (Equation (12)) is changed.

The results suggest that, first, our method is capable of decreasing the error, more so if the initial error is high. For example, *duo-3* has its error decreased by 0.94 times the original error, after ten iterations. Second, in most cases, iterating our method multiple times yields in decreased error. When the error increases with increased number of iterations, we believe that our assumption that alignments are corrupted by independent noise fails. For example, in pieces that involve unnotated *cadenza* (e.g. *solo-3*), incorrect alignment occurs consistently at the cadenza. Then, taking the median of such corrupted data yields not in the underlying "true" alignment, as our method posits, but some meaningless data instead.

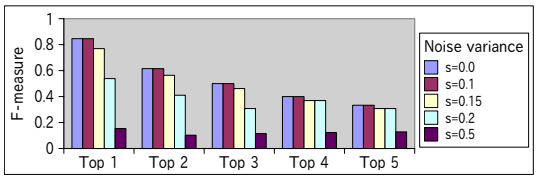
4.2 Evaluation of Music Query

We evaluate the robustness of our system against errors in conducting. When a user conducts like some interpretation i, the system should retrieve i as the most similar interpretation. Other results may be returned for two reasons:

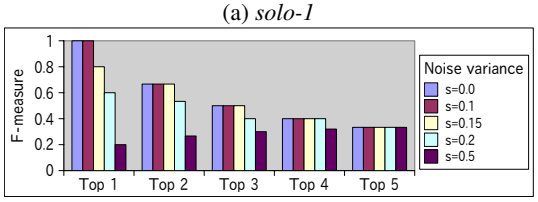
1. The user could not conduct the piece accurately enough to return the desired query.
2. Imprecision in tempo estimation method causes incorrect result to be returned.

In these cases, i may not be the most similar, but one of M most similar interpretations.

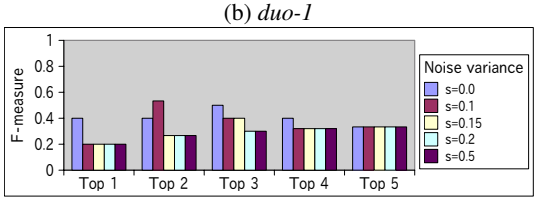
First, we synthesize an artificial query that models human errors in conducting, by adding a smooth noise to the ground truth tempo trajectory of each data. For each interpretation i, we use the following tempo trajectory as the query with some noise variance s :
$$\tau_{\text{query,i}}(t; s) = \tau_{g,i}(t) \cdot 2^{\sqrt{\frac{s}{L}}\sum_{l=0}^{L-1} n(t-l)} | L = 10 \quad (15)$$
$$n(t; s) \sim \mathcal{N}(0, 1)$$



(a) *solo-1*



(b) *duo-1*



(c) *orch-1*

Figure 9: Evaluation results when retrieving up to top 5 results that are similar to artificially generated query which deviates from the ground truth by variance s.

Next, we retrieve M interpretations that are most similar to the artificial query for each interpretation, and evaluate the performance of retrieval the F-measure. Let n be the number of interpretations correctly retrieved by the query. Let N be the total number of interpretations. Then, we let recall $R = n/N$, and precision $P = n/(N \times M)$. The F-measure F is $2PR/(P + R)$. We show the results from *solo-1*, *duo-1*, and *orch-1* in Figure 9 (a), (b), and (c).

Figure 9 (a) and (b) show that the system tolerates small error in conducting, of up to about $s = 0.15$, or 0.7 to 1.3 times the original tempo (three-sigma). Figure 9 (c), however, shows that the F-measure of *orch-1* is considerably lower than the other two.

Retrieving orchestral piece (*orch*'s) is difficult because there is very small variation in the two closest playing, and exacerbated by the particularly unreliable tempo estimation. We compute the smallest dissimilarity measure between ground truth tempo trajectories of any pair of interpretations. The smallest dissimilarity of *orch-1* is about 2×10^{-3}, *solo-1* is 20×10^{-3}, and *duo-1* is 23×10^{-3}. We similarly observed that for orchestral piece, the smallest dissimilarity is much smaller compared to that of chamber (*duo*'s) or solo (*solo*'s). On the other hand, we observe that the average MSE, as seen in Table 1, is substantially greater for orchestral pieces than chamber or solo.

These results suggest that our system retrieves the desired interpretation with robustness against minor errors in conducting, as long as the average MSE is small enough to differentiate the most similar pair of interpretations. The similarity of interpretation is typically influenced by the scale of orchestration, and the average MSE is influenced by the complexity of the ensemble, and the degree to which the interpretation deviates from the music score.

5. CONCLUSION

This paper presented *Query-by-Conducting*, an interface for finding *interpretations* of a given piece of music. It offers the listener an interactive experience of "conducting" the global tempo to dynamically tailor the interpretation played back to the user's choice. It moreover presents the listener with a ranking of interpretation based on how the user conducted through the piece, offering the listener with a list of interpretations whose tempi that the user might like, without the hassle of listening through various interpretations. The accuracy of tempo estimation method improved as a result of considering the consistency of audio-score alignment among different interpretations.

As future work, we would like to deal with aspects of music interpretation other than the global tempo, such as the local tempo deviation and emphasis of a particular melodic line. Integrating these aspects would further enhance the system's capability to retrieve the interpretation of choice. Furthermore, we would like to realize more ways to visualize and interact with various aspects of music interpretation, to allow a listener to further enjoy classical music.

Acknowledgment: This research was partially supported by Graint-in-Aid for Scientific Research (S) of the Ministry of Education, Culture, Sports, Scince and Technology (MEXT), and the CrestMuse Project of the Japan Science and Technology Agency (JST).

6. REFERENCES

[1] M. Müller *et al.* Towards automated extraction of tempo parameters from expressive music recordings. In *ISMIR '09*, pages 69–74, 2009.

[2] C.A.Michael *et al.* Content-based music information retrieval: Current directions and future challenges. volume 96, pages 668–696, April 2008.

[3] S. Dixon, F. Guyon and G. Widmer. Towards characterization of music via rhythmic patterns. In *ISMIR '04*, pages 509–516, 2004.

[4] R. Timmers. Predicting the similarity between expressive performances of music from measurements of tempo and dynamics. *JASA*, 117(1):391–399, 2005.

[5] H. Honing. From time to time: The representation of timing and tempo. *Comp. Music J.*, 25(3):50–61, 2001.

[6] S.Schertenleib *et al.* Conducting a virtual orchestra. *IEEE MultiMedia*, 11(3):40–49, 2004.

[7] J. Segen, S. Kumar and J. Gluckman. Visual interface for conducting virtual orchestra. *ICPR '00*, page 1276, 2000.

[8] H. Katayose and K. Okudaira. Using an expressive performance template in a music conducting interface. In *NIME '04*, pages 124–129, 2004.

[9] F.Bevilacqua *et al.* Wireless sensor interface and gesture-follower for music pedagogy. In *NIME '07*, pages 124–129, 2007.

[10] S. Dixon, W. Goebl and G. Widmer. The "air worm": An interface for real-time manipulation of expressive music performance. In *ICMC '05*, pages 614–617, 2005.

[11] J.R.Jang, H.Lee and C.Yeh. Query by tapping: A new paradigm for content-based music retrieval from acoustic input. In *PCM '01*, pages 590–597. Springer-Verlag, 2001.

[12] F. Kurth *et al.* SyncPlayer - an advanced system for multimodal music access. In *ISMIR '05*, pages 381–388, 2005.

[13] S.Dixon and G.Widmer. MATCH: Music alignment tool chest. In *ISMIR'05*, pages 11–15, 2005.

[14] N.Hu, R. Dannenberg and G. Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *WASPAA '03*, pages 185–188, 2003.

[15] M. Müller, F. Kurth and T. Röder. Towards an efficient algorithm for automatic score-to-audio synchronization. In *ISMIR'04*, pages 365–372, 2004.

QUERYING IMPROVISED MUSIC: DO YOU SOUND LIKE YOURSELF?

Michael O. Jewell, Christophe Rhodes and Mark d'Inverno
Department of Computing
Goldsmiths, University of London
New Cross, London, SE14 6NW
United Kingdom
{m.jewell,c.rhodes,dinverno}@gold.ac.uk

ABSTRACT

Improvisers are often keen to assess how their performance practice stands up to an ideal: whether that ideal is of technical accuracy or instant composition of material meeting complex harmonic constraints at speed. This paper reports on the development of an interface for querying and navigating a collection of recorded material for the purpose of presenting information on musical similarity, and the application of this interface to the investigation of a set of recordings by jazz performers. We investigate the retrieval performance of our tool, and in analysing the 'hits' and particularly the 'misses', provide information suggesting a change in one of the authors' improvisation style.

1. INTRODUCTION

Query-by-Example systems for musical search offer the promise of rich interaction for their users with collections of music. The purpose of a search can be goal-driven or exploratory, while the musical content being searched can be highly focused (as in a curated collection in a sound archive), heterogenous and largely known to the user (a personal collection on a user's personal music player) or heterogenous and largely unknown (an online music vendor's catalogue). The first Query-by-Example systems [8, 16] stored their collections in MIDI format; they admitted audio queries (hence the 'Query-by-Humming' term in the Music Information Retrieval community), and one of the technical hurdles in those systems was a sufficiently accurate transcription of the hummed input – and a search relevance filter that could account for error from imperfect human humming as well as from imperfect transcription algorithms. This mode of interacting with a collection of MIDI-encoded music is available over the web at *Musipedia* [1] .

However, for usable systems, Query-by-Example needs to be augmented by some means of navigating the collection; typically, that navigation mode is specialized to the particular use case enviseaged (and details of the collection being investigated); there exist numerous interfaces and visualisations of collections (such as [9, 12, 17, 18]) and their use for music discovery has been discussed in tutorial sessions [2] .

Achieving intuitive navigation through collections requires some kind of notion of similarity (of which there are many kinds [2]); for systems using primarily content-based information, this means that the audio features or descriptors must encode not only identity but one or more similarity relationships at some level of specificity. Viewed from this perspective, systems based on audio features for classifying and clustering musical tracks [10, 18] or segments [1] are Query-by-Example systems, just as are more modern implementations of the original idea (*e.g.* [7]).

In our work, we are interested in both small-scale and large-scale collections, and in particular at allowing the user to search for and retrieve fragments of tracks (rather than track-to-track or fragment-to-track matches); in principle if given a 5-second audio snippet as a query, we consider all similarly-sized segments in the database – up to some reasonable granularity – as potential matches. This means that collections of even a small number of tracks have a large number of effective database entries to be considered. Achieving fast search through large databases of musical content has been considered in a few applications [14, 15], including the ability to search for specific content within a track in a manner which can still be implemented efficiently [3] and can be generalized [6].

In this paper, we describe a practical use-case for exploratorily searching for fragments of audio by similarity within a small collection. In section 2, we describe in more detail the use case in question; in section 3, we describe how the technology we have developed can meet this need. Our preliminary experiments are reported in section 4, and we draw conclusions and suggest further work in section 5.

2. CASE STUDY

It is often the case that when amateur and semi-professional musicians hear themselves play they cringe at just how

[1] http://www.musipedia.org/

[2] *e.g.* http://musicviz.googlepages.com/home

far away they are from being like the professional heroes that have influenced them. There is a sense of 'I wish I could sound a bit less like me and more like someone *really* good'. We propose to build a tool that provides a general framework for the analysis of performance, allowing performers to both self-analyse and also discover how they relate to their influences.

Performers are often concerned with knowing if their playing has improved in over a time period; whether they can learn about their approach and technique from how professional musicians play particular phrases; or whether they play differently depending on the instrument, event, ensemble, etc.

As such, we are interested in building a tool that enables performing musicians to analyse certain performance characteristics. We propose an iterative development cycle where we increase the scope incrementally in terms of what performance characteristics may be considered, the range of media, the range of extractors, the type of searches (point, track, catalogue) and the options which we make available to a user in the interface. The planned functionality includes, but is not limited to, investigating the following queries:

1. How do performance characteristics of a musician develop over time?

2. How does the performance context (*e.g.* home recording, studio recording) affect performance characteristics?

3. How does the ensemble (*e.g.* solo, duo, trio, big band) affect performance characteristics?

4. How does the type of instrument (*e.g.* in the case of piano, grand, upright, electric) affect performance characteristics?

5. How do certain performance characteristics compare with great musicians?

6. How do performance characteristics develop through a single piece performance?

One of the authors is a reasonable jazz pianist (he has received good reviews in the UK *Guardian* and *Observer* newspapers), so we chose to focus on jazz piano performance, with our ultimate goal as being able to ask the question: 'How much of a performer's improvisation is genuinely improvised, and how much is made from stock patterns?'

Many jazz musicians can come up with phrases or 'licks' that work over chord changes but it is only the greats who can actually approach improvisation as 'instant composition': where what they play is not only appropriate to the sequence but an original passage of notes. The co-author would ideally like to find out where the stock patterns arise in their playing in order to remove them to free up space for more creative improvisation.

3. TECHNOLOGY

3.1 Similarity Measurement

The necessary functionality for our application is the insertion and storage of numerical audio feature information extracted from tracks, and their subsequent searching for similarity. These two aspects are illustrated in figure 1: in the left panel, we schematically show a track which has had d-dimensional audio features extracted for a number of regions of audio. Subsequently, a user wishes to search using a query of region length sl, so successive feature vectors are concatenated (illustrated by the arrows in the left panel) to arrive at shingled [4] feature vectors (right panel). These shingled feature vectors are then compared against the query by summing squared Euclidean distances, and a retrieved list is assembled.

3.2 Interface

iAudioDB is an application developed for Mac OSX in Objective C which provides an intuitive user interface for the creation and exploration of feature databases. As such, it binds directly to the audioDB libraries for creation and querying, and employs Sonic Annotator to extract features from files provided by the user.

Usage of iAudioDB follows a straight-forward process, with the interface providing intuitive abstractions to parameters where possible. The first step is to create the database itself, which is achieved via the interface in Figure 2. The user is prompted for the feature they wish to extract, which corresponds directly to the VAMP plugin [3] which is used with Sonic Annotator, and then a selection of parameters which are database-specific. The first two, 'Max Tracks' and 'Max Length' correspond to the number of audio files the user expects to import into the database and the maximum length in seconds of those tracks. The hop size and window size, equivalent to the step and block size detailed above, are used in conjunction with these values to determine the initial size (in bytes) of the database. Furthermore, the chosen parameters are stored alongside the database to remove the need to enter the settings at the import stage.

Once created, the user imports any audio files, both ground truth and queries. Aside from a standard file dialogue, there is no interface for this, as all parameters required are obtained at creation time. Multiple files may be selected, and progress is indicated as files are imported. At this stage, Sonic Annotator extracts feature information as n3-serialized RDF, which is then imported into the database. Future increments of the software will see it acting as a VAMP host, allowing the use of extractors via a native library. The filenames of the audio files are preserved alongside the unique keys of the tracks in the audioDB instance, thus easing the playback process.

The query process again has an intuitive user interface, shown in Figure 3. The user selects the audio file they wish to use as the query, and from this the length is determined. This length is displayed in the Query Length fields in units

[3] `http://vamp-plugins.org/`

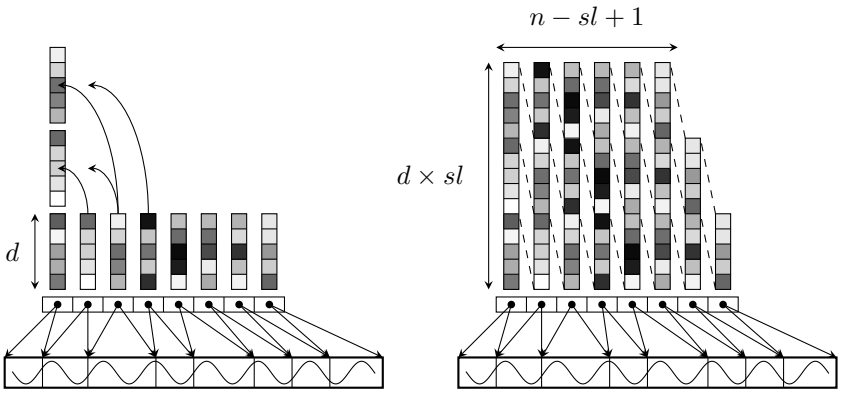


Figure 1. Illustration of the construction of concatenated or shingled feature vectors for our search. Note that while in principle this construction can be done for features over audio regions with temporally-varying extent and step (adjusted to the local tempo), in this paper the step size and block size were kept constant.

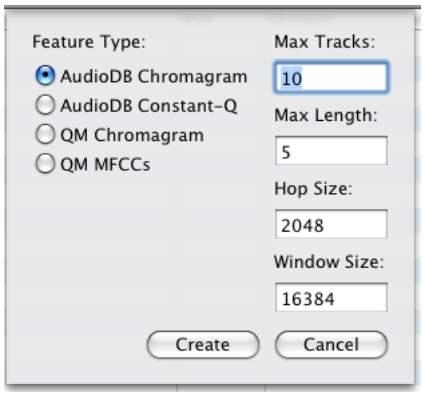


Figure 2. Creating a new database in iAudioDB. The feature extractor is chosen on the left-hand side, while parameters related to the database and the extractor are on the right.

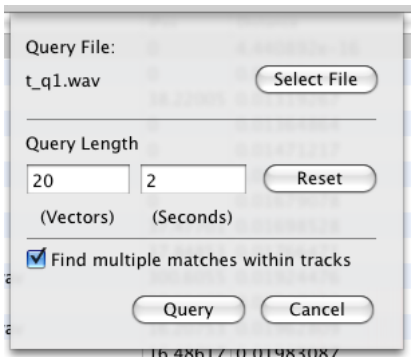


Figure 3. Querying a database in iAudioDB. The query length is generated dynamically from the query audio file, and may then be customized by the user.

of Vectors and Seconds, and both of these fields may be customized by the user to vary the length of the query. The fields are dynamically updated, so a change to the seconds value reflects instantly in the vectors value. If desired, the length may also be reset to the full duration of the query file. Finally, the user may opt to locate multiple matches of a query within the corpus, or to only determine the best match per track.

Once queried, results are displayed in the main application window (see Figure 4). By default, these are sorted by ascending distance values, but this may be customized by clicking the column headers. The other columns are, from left to right, a visual indicator of the closeness of the match (though this varies depending on extractor, so should not be used for comparison), the unique key within the audioDB instance, and the position in seconds at which the query occurs in the track. Results may be played in isolation from the match position, or synchronized with the original query.

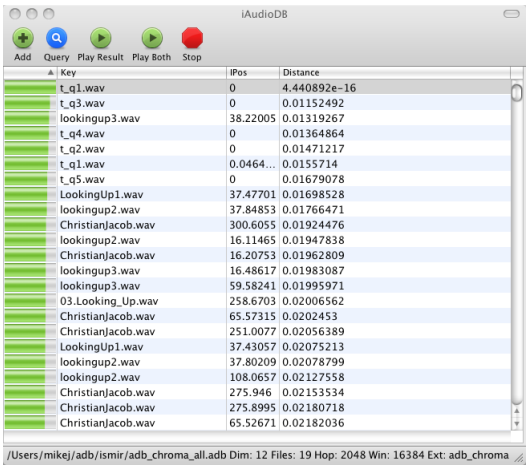


Figure 4. Results generated from an iAudioDB query.

4. FEATURE SPACE INVESTIGATION

The first step in this investigation was to turn our attention to one track and to focus on a single element of the tune. This would at least give us some ground truths whereby we could start to map out a method for getting to our ultimate goal. The track we chose was *Looking Up*, written by the late great jazz pianist Michel Petrucciani. Specifically, we chose the following performance scenarios that would in time enable us to look at all the issues of our case study:

1. The co-author at home using the internal microphone of a laptop recorded three versions of *Looking Up* solo on a Kawai grand piano in an informal setting. These were stored as stereo 44100Hz WAV files.

2. The co-author again, in the same session, but recording three versions of two other tracks – *Ambleside Days* by John Taylor and *My Romance* by Rogers and Hart. (The significance of recording these will become clear later.) As above, these were stored as 44100Hz WAV files.

3. The co-author again, but recorded in a studio context, in trio ensemble and on a Technics electric piano ten years previously.

4. The composer of *Looking Up* and an influence of the co-author, Michel Petrucciani recorded in a concert on a solo grand piano.

5. Michel Petrucciani again but in a band context on a grand piano in a live setting.

6. Another well-regarded pianist and influence of the co-author, Christian Jacob in a trio ensemble, a recording studio with a grand piano.

To begin our iterative development cycle for this application, we focus on one specific phrase in the tune Looking Up (the very first phrase, an 8-note Mixolydian scale in E). This run appears several times in the piece, though the frequency and positions vary on a per-recording basis. The co-author recorded this phrase five times in the same setting as 1 and 2 above to build a library of different queries. These query tracks were played at an even tempo, with no missing or muffled notes.

From this set of tracks three feature databases were built, all with a step size of 2048 samples (0.046s) and a block size of 16384 (0.372s):

1. An MFCC feature database with 20 cepstral coefficients.

2. A constant-Q feature database with 12 bins per octave, a minimum frequency of 65.4064Hz, and a maximum frequency of 1046.5Hz.

3. A chromagram database with the same bins per octave and frequency range as the constant-Q database, and a sum of squares accumulation method.

Track	Position (s)	Missing Notes	Muffled Notes	Rhythm Alterations	Chord Additions	Sustain Pedal
LU1	15		x			
	37					
	59		x			
	86		x			
	162	x				x
LU2	15	x				x
	37					
	59				x	
	107	x		x		
LU3	16	x		x		x
	38					
	59					
	80				x	

Table 1. Locations and comments of fragments corresponding to our queries in the three single-take recordings through a laptop microphone.

The 3 *Looking Up* tracks were examined to locate the positions of the queried tune and thence act as a ground truth. The resultant locations and notes on these instances are shown in Table 1.

Each feature database was then queried with each of the 5 recorded queries, with a maximum length of 20 vectors (1.3s). The recordings of *My Romance* and *Ambleside Days* were used as a boundary, with results examined up to the first match of a track in this set, and duplicated results were discarded. From this set, it was possible to determine those which matched the segments in Table 1 and those which did not. Note that with queries of this length, and with the audio features extracted every 2048 audio samples, there are over 50,000 candidate matching points in our 9-track database; the fact that we are searching for fragments of track rather than whole tracks enlarges the problem.

The mean precision and recall values from these queries can be seen in Table 2, and it is immediately apparent that chromagram features produce the most useful results. While the precision is not as high as that of the constant-Q database the recall is significantly improved, and thus of most benefit to this case study, where the user is looking for a variety of similar matches rather than a small number of exact matches.

Within the results, some notable differences between feature performance were present. Riff instances with muffled notes (15s, 59s, and 86s in *Looking Up* 1) were located in 73% of queries using the chromagram database, 47% using constant-Q, and 20% using MFCCs. Instances with rhythm alterations (107s in *Looking Up* 2 and 16s in *Looking Up* 3) were found in 100% of queries using the chromagram database, 50% using constant-Q (matching the *Look-*

Feature	Precision	Recall	F-Score
MFCC	0.89	0.29	0.44
Constant-Q	1.00	0.57	0.73
Chromagram	0.97	0.83	0.89

Table 2. Average precision, recall, and balanced F-score for our queries against recordings in the same recording environment.

ing Up 2 instance throughout), and none using MFCCs. Finally, the chromagram and constant-Q databases were more resilient to missing notes, matching 75% of the cases in the former and 40% in the latter, with MFCC matching 10%. Interestingly, the riff at 162s in *Looking Up* 1 was entirely unmatched, possibly due to the number of notes missing from the melody.

As a second case, 4 performances of *Looking Up* by professional jazz pianists were added to the databases: a trio studio recording (MDI), a solo piano studio recording (CJ), a live band recording (MP(B)), and a live solo piano recording of the same (MP(S)). The ground truth for this collection is shown in Table 3, and the precision/recall means for the MFCC and chromagram databases in Table 4.

As before, chromagrams provided the most useful results, with a comparatively high mean precision and recall. The CJ recording obtained a mean recall of 1.00 and a mean precision of 0.72, while the MDI recording resulted in a mean recall of 0.43 and a mean precision of 1.00. MP(B) and MP(S) both obtained low recall (0.27 and 0.32 respectively) and good precision (1.00 and 0.78 respectively). Both MP(B) and MP(S) were recorded in a live setting, which may suggest the distance from the query, but notably the queries which didn't match often occurred in locations where the sustain pedal was employed. The CJ recording, while in a studio, was classically precise in terms of note velocity, timing, and consistency, with no sustain pedal employed during the riff instances. The MDI recording only missed matches across all queries when the sustain pedal was used. Further investigation will examine this characteristic more closely.

Track	Position (s)	Missing/Altered Notes	Muffled Notes	Rhythm/Tempo Alterations	Chord Additions	Sustain Pedal
MDI	9					x
	37					x
	64					x
	258					
	285					x
	310					
CJ	15					
	40					
	66					
	250					
	276					
	300					
MP(B)	17	x				
	43			x		
	70				x	
	342	x				x
	368			x		x
	394				x	x
MP(S)	32	x		x		
	65			x		x
	92			x	x	
	202			x		x
	227				x	

Table 3. Locations and comments of fragments corresponding to our queries in the three professional-quality recordings.

5. CONCLUSIONS

Our study, while still at a preliminary stage, is promising: we can achieve good precision and recall for fragments of audio, both for queries recorded under the same conditions as the test database and for queries recorded on consumer hardware against a database of professional studio recordings.

Treated as a pure retrieval task, recall performance is perhaps not as good as might be desired; our observation is that our audio features are not sufficiently robust to the kinds of difference that arise in practice between the query and the matches desired by our userbase. Enhancements in this area would be to incorporate more aspects of desired invariance [11] into our feature, such as for example: constant-Q translations or chroma rotations to model transposition invariance; and beat-based analysis windows

Feature	Precision	Recall	F-Score
MFCC	0.77	0.04	0.08
Chromagram	0.80	0.51	0.62

Table 4. Average precision, recall, and balanced F-score for our queries against the professional, studio recordings.

to incorporate tempo invariance. Because we desire to allow our users to search large databases of audio as well as small ones, we wish to avoid providing invariants using methods scaling worse than linearly with the database size (such as dynamic time warping [14, Chap. 4] for tempo invariance).

However, these invariants are not desired for all applications of our searching technology; in particular, when exploring a corpus for changes in stylistic aspects of performance, it is important for sufficiently different renditions *not* to match a query. The success of our initial experiment in this respect is the observation that one apparently robust characteristic of the ground truth matches in the professionally-recorded corpus that are not found by our current features is that they are executed in the recordings with the sustain pedal on (which has previously been identified as a problem in other MIR tasks [5, 10]); designing a feature to cope with this would be very desirable, but the distinction between the performance practice with sustain and without was new information to our co-author pianist.

We expect to go through several more design-and-test iterations for our implementation of a user interface; known currently-missing features include: a quasi-live interface for rapid, experimental search; and a means for navigation between regions [1, 13]. However, we believe that what we have already developed is good enough for a sophisticated user to be able to explore his own performance practice, or for a composer to use as a thesaurus. The software will be available to download from the OMRAS website [4] shortly after publication, and we welcome feedback from users.

6. REFERENCES

[1] Dominikus Baur, Tim Langer, and Andreas Butz. Shades of Music: Letting Users Discover Sub-song Similarities. In *Proc. ISMIR*, pages 111–116, 2009.

[2] Donald Byrd. A Similarity Scale for Content-Based Music IR. Available at `http://www.informatics.indiana.edu/donbyrd/MusicSimilarityScale.html`, 2008.

[3] M. Casey, C. Rhodes, and M. Slaney. Analysis of Minimum Distances in High-Dimensional Musical Spaces. *IEEE Transactions on Audio, Speech and Language Processing*, 16(5):1015–1028, 2008.

[4] M. Casey and M. Slaney. The Importance of Sequences in Music Similarity. In *Proc. ICASSP*, volume V, pages 5–8, 2006.

[5] Arshia Cont. Realtime Multiple Pitch Observation using Sparse Non-negative Constraints. In *Proc. ISMIR*, 2006.

[6] Mark d'Inverno, Christophe Rhodes, Michael Casey, and Michael Jewell. Content-based Search for Time-based Media. in preparation.

[7] Alexander Duda, Andreas Nürnberger, and Sebastian Stober. Towards Query by Singing/Humming on Audio Databases. In *Proc. ISMIR*, pages 331–334, 2007.

[8] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming: musical information retrieval in an audio database. In *Proc. ACM Conference on Multimedia*, pages 231–236, 1995.

[9] Masataka Goto and Takayuki Goto. Musicream: New Music Playback Interface for Streaming, Sticking, Sorting, and Recalling Musical Pieces. In *Proc. ISMIR*, pages 404–411, 2005.

[10] Maarten Grachten and Gerhard Widmer. Who is who in the end? Recognizing pianists by their final ritardandi. In *Proc. ISMIR*, pages 51–56, 2009.

[11] Kjell Lemström and Geraint A. Wiggins. Formalizing invariances for content-based music retrieval. In *Proc. ISMIR*, pages 591–596, 2009.

[12] M. Magas, M. Casey, and C. Rhodes. mHashup: fast visual music discovery via locality sensitive hashing. In *SIGGRAPH '08: ACM SIGGRAPH 2008 new tech demos*, pages 1–1, Los Angeles, 2008. ACM.

[13] Michela Magas and John Wood. A More User-Centric Approach to the Retrieval of Music Data. submitted to JNMR, 2010.

[14] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag, Berlin Heidelberg, 2007.

[15] Dominik Schnitzer, Arthur Flexer, and Gerhard Widmer. A Filter-and-Refine Indexing Method for Fast Similarity Search in Millions of Music Tracks. In *Proc. ISMIR*, pages 537–542, 2009.

[16] Yuen-Hsien Tseng. Content-based retrieval for music collections. In *Proc. ACM SIGIR*, pages 176–182, 1999.

[17] George Tzanetakis, Andreye Ermonlinskyi, and Perry Cook. Beyond the Query-By-Example Paradigm: New Query Interfaces for Music Information Retrieval. In *Proc. ICMC*, pages 177–183, 2002.

[18] Hugues Vinet, Perfecto Herrera, and François Pachet. The CUIDADO Project. In *Proc. ISMIR*, pages 197–203, 2002.

[4] `http://www.omras2.org`

REAL-TIME POLYPHONIC MUSIC TRANSCRIPTION WITH NON-NEGATIVE MATRIX FACTORIZATION AND BETA-DIVERGENCE

Arnaud Dessein, Arshia Cont, Guillaume Lemaitre
IRCAM – CNRS UMR 9912, Paris, France
{dessein,cont,lemaitre}@ircam.fr

ABSTRACT

In this paper, we investigate the problem of real-time polyphonic music transcription by employing non-negative matrix factorization techniques and the β-divergence as a cost function. We consider real-world setups where the music signal arrives incrementally to the system and is transcribed as it unfolds in time. The proposed transcription system is addressed with a modified non-negative matrix factorization scheme, called non-negative decomposition, where the incoming signal is projected onto a fixed basis of templates learned off-line prior to the decomposition. We discuss the use of non-negative matrix factorization with the β-divergence to achieve the real-time decomposition. The proposed system is evaluated on the specific task of piano music transcription and the results show that it can outperform several state-of-the-art off-line approaches.

1. INTRODUCTION

The task of *music transcription* consists in converting a raw music signal into a symbolic representation such as a score. Considering polyphonic signals, this task is closely related to the problem of multiple-pitch estimation which has been largely investigated for music as well as speech, and for which a wide variety of methods have been proposed [8]. Non-negative matrix factorization has already been used in this context, with off-line approaches [1, 3, 20, 22–24] as well as on-line approaches [4, 6, 7, 17, 21].

Generally speaking, *non-negative matrix factorization* (NMF) is a technique for data analysis where the observed data are supposed to be non-negative [16]. The main philosophy of NMF is to build up these observations in a constructive additive manner, what is particularly interesting when negative values cannot be interpreted (*e.g.* pixel intensity, word occurrence, magnitude spectrum).

In this paper, we employ NMF techniques to develop a real-time system for polyphonic music transcription. This system is thought as a front-end for musical interactions in live performances. Among applications, we are interested in computer-assisted improvisation for instruments such as the piano. We do not discuss such applications in the paper but rather concentrate on the system for polyphonic music transcription and invite the curious reader to visit the companion website [1] for complementary information and additional resources. The proposed system is addressed with an NMF scheme called *non-negative decomposition* where the signal is projected in real-time onto a basis of note templates learned off-line prior to the decomposition.

In this context, the price to pay for the simplicity of the standard NMF is the overuse of templates to construct the incoming signal, resulting in note insertions and substitutions such as octave and harmonic errors. In [6, 7], the issue has been tackled with the standard Euclidean cost by introduction of a sparsity constraint similar to [14]. We here investigate the use of more complex costs by using the β-divergence. This is in contrast to previous systems for real-time audio decomposition which have either considered the Euclidean distance or the Kullback-Leibler divergence. NMF with the β-divergence has recently proved its relevancy for off-line applications in speech analysis [18], music analysis [11] and music transcription [3, 23]. We adapt these approaches to a real-time setup and propose a tailored multiplicative update to compute the decomposition. We also give intuition in understanding how the β-divergence helps to improve transcription. The provided evaluation show that the proposed system can outperform several off-line algorithms at the state-of-the-art.

The paper is organized as follows. In Section 2, we introduce the related background on NMF techniques. In Section 3, we focus on NMF with the β-divergence, provide a multiplicative update tailored to real-time decomposition, and discuss the relevancy of the β-divergence for the decomposition of polyphonic music signals. In Section 4, we depict the general architecture of the real-time system proposed for polyphonic music transcription, and detail the two modules respectively used for off-line learning of note templates and for on-line decomposition of music signals. In Section 5, we perform evaluations of the system for the specific task of piano music transcription.

In the sequel, uppercase bold letters denote matrices, lowercase bold letters denote column vectors, lowercase plain letters denote scalars. $\mathbb{R}_+$ and $\mathbb{R}_{++}$ denote respectively the sets of non-negative and of positive scalars. The element-wise multiplication and division between two matrices $\mathbf{A}$ and $\mathbf{B}$ are denoted respectively by $\mathbf{A} \otimes \mathbf{B}$ and $\frac{\mathbf{A}}{\mathbf{B}}$. The element-wise power p of $\mathbf{A}$ is denoted by $\mathbf{A}^{\cdot p}$.

[1] http://imtr.ircam.fr/imtr/Realtime_Transcription

2. RELATED BACKGROUND

This section introduces the NMF model, the standard NMF problem, and the popular multiplicative updates algorithm used to solve it. We then present the relevant literature in sound recognition with NMF.

2.1 NMF model

The NMF model is a low-rank approximation for unsupervised multivariate data analysis. Given an $n \times m$ non-negative matrix $\mathbf{V}$ and a positive integer $r < \min(n, m)$, NMF tries to factorize $\mathbf{V}$ into an $n \times r$ non-negative matrix $\mathbf{W}$ and an $r \times m$ non-negative matrix $\mathbf{H}$ such that:

$$\mathbf{V} \approx \mathbf{WH} \tag{1}$$

In this model, the multivariate data are stacked into $\mathbf{V}$, whose columns represent the different observations, and whose rows represent the different variables. Each column $\mathbf{v}_j$ of $\mathbf{V}$ can be expressed as $\mathbf{v}_j \approx \mathbf{W}\mathbf{h}_j = \sum_i h_{ij}\mathbf{w}_i$, where $\mathbf{w}_i$ and $\mathbf{h}_j$ are respectively the i-th column of $\mathbf{W}$ and the j-th column of $\mathbf{H}$. The columns of $\mathbf{W}$ then form a *basis* and each column of $\mathbf{H}$ is the *decomposition* of the corresponding column of $\mathbf{V}$ into this basis.

2.2 Standard problem and multiplicative updates

The standard NMF model of Equation 1 provides an approximate factorization $\mathbf{WH}$ of $\mathbf{V}$. The aim is then to find the factorization which optimizes a given goodness-of-fit measure called *cost function*. In the standard formulation, the Euclidean distance is used, and the NMF problem amounts to minimizing the following cost function subject to non-negativity of both $\mathbf{W}$ and $\mathbf{H}$:

$$\frac{1}{2}\|\mathbf{V} - \mathbf{WH}\|_F^2 = \frac{1}{2}\sum_j \|\mathbf{v}_j - \mathbf{W}\mathbf{h}_j\|_2^2 \tag{2}$$

For this particular cost function, factors $\mathbf{W}$ and $\mathbf{H}$ can be computed with the popular *multiplicative updates* introduced in [16]. These updates are derived from a gradient descent scheme with judiciously chosen steps, as follows:

$$\mathbf{H} \leftarrow \mathbf{H} \otimes \frac{\mathbf{W}^T\mathbf{V}}{\mathbf{W}^T\mathbf{WH}} \qquad \mathbf{W} \leftarrow \mathbf{W} \otimes \frac{\mathbf{V}\mathbf{H}^T}{\mathbf{WHH}^T} \tag{3}$$

The updates are applied in turn until convergence, and ensure both non-negativity and decreasing of the cost, but not necessarily local optimality of factors $\mathbf{W}$ and $\mathbf{H}$.

A flourishing literature exists about extensions to the standard NMF problem and their algorithms [5]. These extensions can be thought of in terms of modified cost functions (*e.g.* using divergences or adding penalty terms), of modified constraints (*e.g.* imposing sparsity), and of modified models (*e.g.* using tensors). For example, the cost function defined in Equation 2 is often replaced with the Kullback-Leibler divergence for which specific multiplicative updates have been derived [16].

2.3 Applications in sound recognition

NMF algorithms have been applied to various problems in vision, sound analysis, biomedical data analysis and text classification among others [5]. In the context of sound analysis, the matrix $\mathbf{V}$ is in general a time-frequency representation of the sound to analyze. The rows and columns represent respectively different frequency bins and successive time-frames. The factorization $\mathbf{v}_j \approx \sum_i h_{ij}\mathbf{w}_i$ can then be interpreted as follows: each basis vector $\mathbf{w}_i$ contains a spectral template, and the decomposition coefficients h_{ij} represent the activations of the i-th template $\mathbf{w}_i$ at the j-th time-frame.

NMF has already been used in the context of polyphonic music transcription (*e.g.* see [1, 22]). Several problem-dependent extensions have been developed to this end such as a source-filter model [24], an harmonic constraint [20], an harmonic model with temporal smoothness [3], or an harmonic model with spectral smoothness [23]. These approaches rely in general on the off-line nature of NMF, but some authors have used NMF in an on-line setup.

A real-time system to identify the presence and determine the pitch of one or more voices is proposed in [21]. This system is also adapted for sight-reading evaluation of solo instrument in [4]. Concerning automatic transcription, a similar system is used in [17] for transcription of polyphonic music, and in [19] for drum transcription. A real-time system for polyphonic music transcription with sparsity considerations is proposed in [6]. The approach is further developed in [7] for real-time coupled multiple-pitch and multiple-instrument recognition. Yet, all these approaches are based on NMF with the Euclidean distance or the Kullback-Leibler divergence. We discuss the use of the more general β-divergence as a cost function and its relevancy for decomposition of music signals in Section 3.

3. NON-NEGATIVE DECOMPOSITION WITH THE BETA-DIVERGENCE

In this section, we define the β-divergence, give some of its properties, and review its use as a cost function for NMF. We finally formulate the non-negative decomposition problem with the β-divergence and give multiplicative updates tailored to real-time for solving it.

3.1 Definition and properties of the beta-divergence

The β-divergences form a parametric family of distortion functions [9]. For any $\beta \in \mathbb{R}$ and any points $x, y \in \mathbb{R}_{++}$, the β-divergence from x to y is defined as follows:

$$d_\beta(x|y) = \frac{1}{\beta(\beta-1)}\left(x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}\right) \tag{4}$$

As special cases when $\beta = 0$ and $\beta = 1$, taking the limits in the above definition leads respectively to the well-known Itakura-Saito and Kullback-Leibler divergences:

$$d_{\beta=0}(x|y) = d_{IS}(x|y) = \frac{x}{y} - \log\frac{x}{y} - 1 \tag{5}$$

$$d_{\beta=1}(x|y) = d_{KL}(x|y) = x\log\frac{x}{y} + y - x \tag{6}$$

For $\beta = 2$, the β-divergence specializes to the widely used half squared Euclidean distance:

$$d_{\beta=2}(x|y) = d_E(x|y) = \frac{1}{2}(x-y)^2 \tag{7}$$

Concerning their properties, all β-divergences are non-negative and vanish iff $x = y$. However, they are not necessary distances in the strict terms since they are not symmetric and do not satisfy the triangle inequality in general. A property of the β-divergences relevant to the present work is that for any scaling factor $\lambda \in \mathbb{R}_{++}$ we have:

$$d_\beta(\lambda x|\lambda y) = \lambda^\beta d_\beta(x|y) \tag{8}$$

We discuss further the interest of this scaling property for decomposition of polyphonic music signals in Section 3.3.

3.2 NMF and the beta-divergence

The β-divergence was first used with NMF to interpolate between the Euclidean distance and the Kullback-Leibler divergence [15]. Starting with the scalar divergence in Equation 4, a matrix divergence can be constructed as a *separable* divergence, *i.e.* by summing the element-wise divergences. The NMF problem with the β-divergence then amounts to minimizing the following cost function subject to non-negativity of both $\mathbf{W}$ and $\mathbf{H}$:

$$\mathcal{D}_\beta(\mathbf{V}|\mathbf{WH}) = \sum_{i,j} d_\beta(v_{ij} \,|\, [\mathbf{WH}]_{ij}) \tag{9}$$

For $\beta = 2$, this cost function specializes to the cost defined in Equation 2 for standard NMF.

As for standard NMF, several algorithms including multiplicative updates have been derived for NMF with the β-divergence and its extensions [5, 15]. The β-divergence has also proved its relevancy as a cost function for audio off-line applications in speech analysis [18], music analysis [11] and music transcription [3, 23].

3.3 Problem formulation and multiplicative update

We now formulate the problem of non-negative decomposition with the β-divergence. We assume that $\mathbf{W}$ is a fixed dictionary of note templates onto which we seek to decompose the incoming signal $\mathbf{v}$ as $\mathbf{v} \approx \mathbf{Wh}$. The problem is therefore equivalent to minimizing the following cost function subject to non-negativity of $\mathbf{h}$:

$$\mathcal{D}_\beta(\mathbf{v}|\mathbf{Wh}) = \sum_i d_\beta(v_i \,|\, [\mathbf{Wh}]_i) \tag{10}$$

To solve this problem, we update $\mathbf{h}$ iteratively by using a vector version of the corresponding multiplicative update proposed in the literature [5, 15]. As $\mathbf{W}$ is fixed, we never apply its respective update. The algorithm thus amounts to repeating the following update until convergence:

$$\mathbf{h} \leftarrow \mathbf{h} \otimes \frac{\mathbf{W}^T\big((\mathbf{Wh})^{\cdot\beta-2} \otimes \mathbf{v}\big)}{\mathbf{W}^T(\mathbf{Wh})^{\cdot\beta-1}} \tag{11}$$

This scheme ensures non-negativity of $\mathbf{h}$, but not necessarily local optimality. Unfortunately, no proof has been found yet to show that the cost function is non-increasing under this update for a general parameter β, even if it has been observed in practice [11]. However, even if such theoretical issues need to be investigated further, the simplicity of this scheme makes it suitable for real-time applications and gives good results in practice.

Concerning implementation, we can take advantage of $\mathbf{W}$ being fixed to employ a multiplicative update tailored to real-time decomposition. Indeed, after some matrix manipulations, we can rewrite the updates as follows:

$$\mathbf{h} \leftarrow \mathbf{h} \otimes \frac{\big(\mathbf{W} \otimes (\mathbf{ve}^T)\big)^T(\mathbf{Wh})^{\cdot\beta-2}}{\mathbf{W}^T(\mathbf{Wh})^{\cdot\beta-1}} \tag{12}$$

where $\mathbf{e}$ is a vector full of ones. This helps to reduce the computational cost of the update scheme as the matrix $\big(\mathbf{W} \otimes (\mathbf{ve}^T)\big)^T$ needs only to be computed once.

The scaling property in Equation 8 may give an insight in understanding the relevancy of the β-divergence in our context. For $\beta = 0$, the Itakura-Saito divergence is the only β-divergence to be scale-invariant as it was remarked in [11]. This means that the corresponding NMF problem gives the same relative weight to all coefficients, and thus penalizes equally a bad fit of factorization for small and large coefficients. Considering music signals, this amounts to giving the same importance to high-energy and to low-energy frequency components. When $\beta > 0$, more emphasis is put on the frequency components of higher energy, and the emphasis augments with β. When $\beta < 0$, the effect is the converse. In our context of music decomposition, we try to reconstruct an incoming music signal by addition of note templates. In order to avoid common octave and harmonic errors, a good reconstruction would have to find a compromise between focusing on the fundamental frequency, the first partials and higher partials. The parameter β can thus help to control this trade-off.

4. GENERAL ARCHITECTURE OF THE SYSTEM

In this section, we present the real-time system proposed for polyphonic music transcription. The general architecture is shown schematically in Figure 1. The right side of the figure represents the music signal arriving in real-time, and its decomposition onto notes whose descriptions are provided *a priori* to the system as templates. These templates are learned off-line, as shown on the left side of the figure, and constitute the dictionary used during real-time decomposition. We describe the two modules hereafter.

4.1 Note template learning

The learning module aims at building a dictionary $\mathbf{W}$ of note templates onto which the polyphonic music signal is projected during the real-time decomposition phase.

In the present work, we use a simple rank-one NMF with the standard cost function as a learning scheme. We suppose that the user has access to isolated note samples of the instruments to transcribe, from which the system learns characteristic templates. The whole note sample k is first

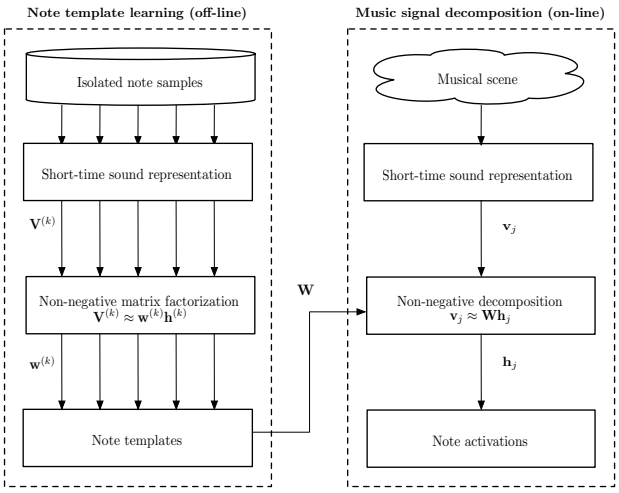


Figure 1. Schematic view of the general architecture.

processed in a short-time sound representation supposed to be non-negative and approximatively additive (*e.g.* a short-time magnitude spectrum). The representations are stacked in a matrix $\mathbf{V}^{(k)}$ where each column $\mathbf{v}_j^{(k)}$ is the sound representation of the j-th time-frame. We then solve standard NMF with $\mathbf{V}^{(k)}$ and a rank of factorization $r = 1$, using the multiplicative updates in Equation 3. This learning scheme simply gives a template $\mathbf{w}^{(k)}$ for each note sample (the information in the row vector $\mathbf{h}^{(k)}$ is discarded).

4.2 Music signal decomposition

Having learned the templates, we stack them in columns to form the dictionary $\mathbf{W}$. The problem of real-time transcription then amounts to projecting the incoming music signal $\mathbf{v}_j$ onto $\mathbf{W}$, where $\mathbf{v}_j$ share the same representational front-end as the note templates. The problem is thus equivalent to a non-negative decomposition $\mathbf{v}_j \approx \mathbf{W}\mathbf{h}_j$ where $\mathbf{W}$ is kept fixed and only $\mathbf{h}_j$ is learned. The learned vectors $\mathbf{h}_j$ would then provide successive activations of the different notes in the music signal. Following the discussion in Section 3, we learn the vectors $\mathbf{h}_j$ by employing the β-divergence as a cost function and the multiplicative update tailored to real-time decomposition in Equation 11.

As such, the system reports only a frame-level activity of the notes. Some post-processing is thus needed to extract more information about the eventual presence of the notes, and provide a symbolic representation of the music signal for transcription. This post-processing potentially includes activation thresholding, onset detection, temporal modeling, etc. It is however not thoroughly discussed in this paper where we use a simple threshold-based detection followed by a minimum duration pruning.

5. EVALUATION AND RESULTS

In this section, we evaluate the system on polyphonic transcription of piano music. We provide a subjective evaluation with musical excerpts synthesized from MIDI references. We also perform an objective evaluation with a real piano music database and standard evaluation metrics.

5.1 Subjective evaluation

As sample examples, we transcribed two musical excerpts synthesized from MIDI references with real piano samples from the Real World Computing (RWC) database [12].

For the non-negative decomposition, β was set to 0.5 since this value was shown optimal for music transcription in [23] and provided good results in our tests. The threshold for detection was set to 2 and no minimum duration pruning was applied. For the dictionary, one note template was learned and max-normalized for each of the 88 notes of the piano using corresponding samples taken from RWC. We used a simple short-time magnitude spectrum representation, with a frame size of 50 ms leading to 630 samples at a sampling rate of 12600 Hz, and computed with a zero-padded Fourier transform of 1024 bins. The frames were windowed with a Hamming function, and the hopsize was set to 25 ms for template learning and refined to 10 ms for decomposition. The decomposition was computed in real-time simulation under MATLAB on a 2.40 GHz laptop with 4.00 Go of RAM, and was about three times faster than real-time.

The results of the decomposition are shown in Figure 2. Figures 2(a) and 2(b) depict the piano-roll representations of the two piano excerpts. The ground-truth references are represented with rectangles and the transcriptions with black dots. Overall, this shows that the system is able to match reliably the note templates to the music signals. During note attacks, more templates are used due to transients but some post-processing such as minimum duration pruning would help to remove these errors. We also remark a tendency to shorten sustained notes which may be due to a different spectral content during note releases.

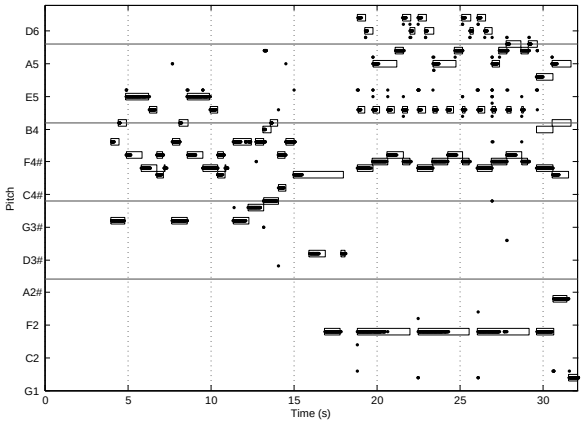
5.2 Objective evaluation

For a more rigorous evaluation, we considered the standards of the Music Information Retrieval Evaluation eXchange (MIREX) [2] and focused on two subtasks: (1) a frame-level estimation of the present events in terms of musical pitch, and (2) a note-level tracking of the present notes in terms of musical pitch, onset and offset times.

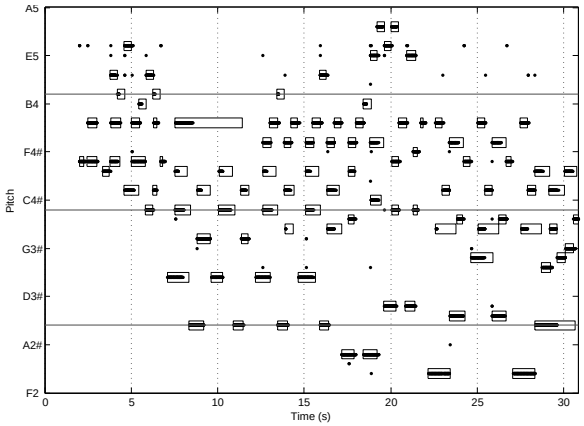
For the evaluation dataset, we chose the MIDI-Aligned Piano Sounds (MAPS) database [10]. MAPS contains real recordings of piano pieces with ground-truth references. We selected 25 pieces and truncated each of them to 30 s.

Concerning parameters, β was set to 0.5. The thresholds for detection were set empirically to 1 and 2 for the frame and note levels respectively. The minimum duration for pruning was set to 50 ms. The templates were learned from MAPS with the same representation front-end as above. This algorithm is referenced by BND.

In addition, we tested the system with the standard Euclidean decomposition algorithm referenced by END, and with the sparse algorithm of [14] with projection onto the cone of sparsity $s = 0.9$. For these two algorithms, the detection thresholds were set to 2 and 4 for the frame and note levels respectively. To compare results, we also performed the evaluation for two off-line systems at the state-of-the-art: one based on NMF but with an harmonic model



(a) 1st movement, *Pavane de la Belle au bois dormant*.



(b) 4th movement, *Les entretiens de la Belle et de la Bête*.

Figure 2. Transcription of two piano excerpts from *Ma mère l'Oye, Cinq pièces enfantines pour piano à quatre mains* (1908-1910), Maurice Ravel (1875-1937).

and spectral smoothness [23], and another one based on a sinusoidal analysis with a candidate selection exploiting spectral features [25].

We report the evaluation results per algorithm in Tables 1 and 2 at the frame and note levels respectively. Standard evaluation metrics from the MIREX are used as described in [2]: precision $\mathcal{P}$, recall $\mathcal{R}$, F-measure $\mathcal{F}$, accuracy $\mathcal{A}$, total error $\mathcal{E}_{tot}$, substitution error $\mathcal{E}_{subs}$, missed error $\mathcal{E}_{miss}$, false alarm error $\mathcal{E}_{fa}$, mean overlap ratio $\mathcal{M}$. At the note level, the subscripts 1 and 2 represent respectively the onset-based and the onset/offset-based results.

Overall, the results show that the proposed real-time system performs comparably to the state-of-the-art off-line algorithms of [23, 25]. Using the β-divergence, the system BND even outperforms the other algorithms. The sparse algorithm of [14] reduces insertions and substitutions, but augments the number of missed notes so that it actually does not perform better than the standard scheme END. The standard Euclidean cost also shows its limits for transcription where more complex costs with the β-divergence give better results. We finally remark that the mean overlap ratio scores corroborate the observation that sustained notes tend to be shortened.

Alg.	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{A}$	$\mathcal{E}_{tot}$	$\mathcal{E}_{subs}$	$\mathcal{E}_{miss}$	$\mathcal{E}_{fa}$
BND	63.9	67.3	**65.5**	**48.7**	**58.9**	11.9	20.8	26.2
END	55.3	58.6	**56.9**	**39.8**	71.4	17.3	24.1	29.9
[14]	58.5	55.2	56.8	39.7	**67.1**	16.8	28.0	22.3
[23]	61.0	66.7	63.7	46.8	65.6	10.4	22.9	32.3
[25]	60.0	70.8	**65.0**	**48.1**	**60.0**	16.3	12.8	30.8

Table 1. Frame-level transcription results per algorithm.

Alg.	$\mathcal{P}_1$	$\mathcal{R}_1$	$\mathcal{F}_1$	$\mathcal{A}_1$	$\mathcal{M}_1$	$\mathcal{P}_2$	$\mathcal{R}_2$	$\mathcal{F}_2$	$\mathcal{A}_2$
BND	75.5	67.1	**71.1**	**55.1**	**56.7**	30.0	26.6	**28.2**	**16.4**
END	57.9	58.2	**58.1**	**40.9**	53.9	21.4	21.6	**21.5**	**12.0**
[14]	57.2	56.3	56.8	39.6	**54.1**	21.0	20.7	20.8	11.6
[23]	58.1	73.7	**65.0**	**48.1**	**57.7**	20.7	26.3	**23.2**	**13.1**
[25]	33.0	58.8	42.3	26.8	55.1	11.6	20.7	14.9	8.0

Table 2. Note-level transcription results per algorithm.

6. CONCLUSION

This paper addressed the problem of real-time polyphonic music transcription by employing NMF techniques. We discussed the use of the β-divergence as a cost function for non-negative decomposition tailored to real-time transcription. The obtained results show that the proposed system can outperform state-of-the-art off-line approaches, and are encouraging for further development.

A problem in our approach is that templates are inherently considered as stationary. One way to tackle this is to consider representations that capture variability over a short time-span as in [7]. We could also combine NMF with a state representation and use templates for each state.

The template learning method can be further improved by using extended NMF problems and algorithms to learn one or more templates for each note. Such issues have not been developed but interesting perspectives include learning sparse or harmonic templates. Using the β-divergence during template learning in our experience did not improve the results. Further considerations are needed on this line.

In a live performance setup such as ours, the templates can be directly learned from the corresponding instrument. Yet in other setups, the issue of generalization must be carefully considered and will be discussed in future work. We think of considering adaptive templates by adapting an approach proposed in [13] to real-time decomposition.

We would like also to improve the robustness against noise, by keeping information from the activations during template learning, or by using noise templates as in [7]. In addition, we want to develop more elaborate sparsity controls than in [6, 7, 14]. In our approach, sparsity is controlled implicitly during decomposition. Yet in some applications, specially for complex problems such as auditory scene analysis, controlling explicitly sparsity becomes crucial. A forthcoming paper will address this issue.

Last but not least, the proposed system is currently under development for the Max/MSP real-time computer music environment and will be soon available for free download on the companion website.

7. ACKNOWLEDGMENTS

This work was partially funded by a doctoral fellowship from the UPMC (EDITE). The authors would like to thank C. Yeh and R. Badeau for their valuable help, V. Emiya for kindly providing the MAPS database, as well as P. Hoyer and E. Vincent for sharing their source code.

8. REFERENCES

[1] S. A. Abdallah and M. D. Plumbley. Polyphonic music transcription by non-negative sparse coding of power spectra. In *Proc. of ISMIR 2004*, pages 318–325, Barcelona, Spain, October 2004.

[2] M. Bay, A. F. Ehmann, and J. S. Downie. Evaluation of multiple-$F0$ estimation and tracking systems. In *Proc. of ISMIR 2009*, Kobe, Japan, October 2009.

[3] N. Bertin, R. Badeau, and E. Vincent. Enforcing harmonicity and smoothness in Bayesian non-negative matrix factorization applied to polyphonic music transcription. *IEEE Transactions on Audio, Speech and Language Processing*, 18(3):538–549, March 2010.

[4] C.-C. Cheng, D. J. Hu, and L. K. Saul. Nonnegative matrix factorization for real time musical analysis and sight-reading evaluation. In *Proc. of ICASSP 2008*, pages 2017–2020, Las Vegas, NV, USA, March/April 2008.

[5] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley-Blackwell, 2009.

[6] A. Cont. Realtime multiple pitch observation using sparse non-negative constraints. In *Proc. of ISMIR 2006*, Victoria, Canada, October 2006.

[7] A. Cont, S. Dubnov, and D. Wessel. Realtime multiple-pitch and multiple-instrument recognition for music signals using sparse non-negative constraints. In *Proc. of DAFx 2007*, Bordeaux, France, September 2007.

[8] A. de Cheveigné. *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, chapter Multiple $F0$ Estimation, pages 45–72. Wiley-IEEE Press, 2006.

[9] S. Eguchi and Y. Kano. Robustifying maximum likelihood estimation. Technical report, Institute of Statistical Mathematics, Tokyo, Japan, 2001.

[10] V. Emiya, R. Badeau, and B. David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech and Language Processing*, To appear.

[11] C. Févotte, N. Bertin, and J.-L. Durrieu. Nonnegative matrix factorization with the Itakura-Saito divergence: with application to music analysis. *Neural Computation*, 21(3):793–830, March 2009.

[12] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: popular, classical, and jazz music databases. In *Proc. of ISMIR 2002*, pages 287–288, October 2002.

[13] M. Heiler and C. Schnörr. Learning sparse representations by non-negative matrix factorization and sequential cone programming. *J. of Machine Learning Research*, 7:1385–1407, July 2006.

[14] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. of Machine Learning Research*, 5:1457–1469, November 2004.

[15] R. Kompass. A generalized divergence measure for nonnegative matrix factorization. *Neural Computation*, 19(3):780–791, 2007.

[16] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999.

[17] B. Niedermayer. Non-negative matrix division for the automatic transcription of polyphonic music. In *Proc. of ISMIR 2008*, pages 544–549, Philadelphia, PA, USA, September 2008.

[18] P. D. O'Grady and B. A. Pearlmutter. Discovering speech phones using convolutive non-negative matrix factorisation with a sparseness constraint. *Neurocomputing*, 72:88–101, 2008.

[19] J. Paulus and T. Virtanen. Drum transcription with non-negative spectrogram factorisation. In *Proc. of EUSIPCO 2005*, Antalya, Turkey, September 2005.

[20] S. A. Raczyński, N. Ono, and S. Sagayama. Harmonic nonnegative matrix approximation for multipitch analysis of musical sounds. In *Proc. of ASJ Autumn Meeting*, pages 827–830, September 2007.

[21] F. Sha and L. K. Saul. Real-time pitch determination of one or more voices by nonnegative matrix factorization. In *Proc. of NIPS 2004*, volume 17, pages 1233–1240, Cambridge, MA, USA, 2005.

[22] P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *Proc. of WASPAA 2003*, pages 177–180, New Paltz, NY, USA, October 2003.

[23] E. Vincent, N. Bertin, and R. Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio, Speech and Language Processing*, 18(3), March 2010.

[24] T. Virtanen and A. Klapuri. Analysis of polyphonic audio using source-filter model and non-negative matrix factorization. In *Proc. of NIPS Workshop AMAC 2006*, 2006.

[25] C. Yeh. *Multiple fundamental frequency estimation of polyphonic recordings*. PhD thesis, Université Pierre et Marie Curie, Paris, France, June 2008.

Recognizing Classical Works in Historical Recordings

Tim Crawford
Goldsmiths, University of London, Centre for Cognition, Computation and Culture
t.crawford@gold.ac.uk

Matthias Mauch
Queen Mary, University of London, Centre for Digital Music
matthias.mauch@elec.qmul.ac.uk

Christophe Rhodes
Goldsmiths, University of London, Department of Computing
c.rhodes@gold.ac.uk

ABSTRACT

In collections of recordings of classical music, it is normal to find multiple performances, usually by different artists, of the same pieces of music. While there may be differences in many dimensions of musical similarity, such as timbre, pitch or structural detail, the underlying musical content is essentially and recognizably the same. The degree of divergence is generally less than that found between 'cover songs' in the domain of popular music, and much less than in typical performances of jazz standards. MIR methods, based around variants of the chroma representation, can be useful in tasks such as work identification especially where disco/bibliographical metadata is absent or incomplete as well as for access, curation and management of collections. We describe some initial experiments in work-recognition on a test-collection comprising c. 2000 digital transfers of historical recordings, and show that the use of *NNLS chroma*, a new, musically-informed chroma feature, dramatically improves recognition.

1. INTRODUCTION

As was pointed out by Richard Smiraglia in a paper at ISMIR 2001, "musical works (as opposed to musical documents, such as scores or recordings of musical works) form a key entity for music information retrieval. … However, in the [general] information retrieval domain, the work, as opposed to the document, has only recently received focused attention." [1] This largely remains true today; despite a steady advance in content-based MIR techniques, we have hardly begun to realize the potential power of using them to extract higher-level musical knowledge corresponding to what is embedded in bibliographical metadata, hitherto the exclusive domain of music-librarianship. In this paper we use the term 'work' simply to refer to the musical composition as represented by the notes in a musical score (though we acknowledge that the concept is much more complex than this naïve definition assumes). The importance of the work concept in classical music becomes immediately apparent when one is confronted with the kind of confused or inaccurate metadata that often results from the use of online CD-recognition systems which rely on the ID3 tagging scheme [2] used for identifying mp3 tracks, which is rarely applied correctly to classical music. Further problems arise when a track becomes isolated from its original media (e.g. by digital copying or 'ripping' from a CD). The situation is even more problematic when works are segmented differently in different recorded manifestations: there is, for example, no standard way to divide up the continuous music of an opera into CD tracks; although there exist musicological conventions about the navigation through numbered acts and scenes, even these can break down when, for example, it is not clear from the score whether an introductory recitative forms part of an aria or forms an independent number.

In the controlled environment of the digital music library these issues can be addressed by adopting cataloguing standards such as FRBR [3], which deals comprehensively with the musical work concept and its various manifestations in physical and recorded form. The correct identification of classical works (for example, on uncatalogued archive tapes), or fragments from them (as frequently encountered on movie or advertisement sound-tracks) remains a time-consuming task demanding considerable expertise. The solution to some of these problems may lie in a system built around content-based work-recognition, operating over the internet on well-documented 'authority' collections of recorded works whose metadata can be trusted.

For much of the mainstream classical repertory, however, the work concept is fairly straightforward. The collection investigated here can be claimed to be fairly representative of the taste of classical-music record buyers in the years before the Second World War. This paper deals with the particular case of historical recordings of classical music, much of which is still in the mainstream repertory, but in which the integrity of the work may be compromised by the restrictions of the recording process itself.

Music recorded before about 1960 almost exclusively exists in the form of 78-rpm gramophone recordings. Many of these, by famous artists from Caruso to Glenn Miller, are available in modern commercial transfers,

often painstakingly enhanced using a variety of digital technologies.[2] But there remains a vast heritage of recorded music performed by less well-known artists that is unlikely ever to prompt the investment necessary to make commercial release viable. Digitization initiatives in a number of countries are increasingly making these recordings available to scholars investigating the history of recording and of musical performance as well as to the general public.[3] Music information retrieval (MIR) techniques offer rich possibilities for the curation and management of, as well as the access to, such collections. Professional metadata standards used by music librarians for cataloguing mainstream classical music, whether in the form of scores or recordings, universally make use of the work concept, and it is natural to seek ways to aid them using MIR techniques such as those described in this paper.

The task we employ as a use-case in this paper, Classical Work Recognition, is described in Section 2. Early recordings present special problems and are not suitable for many MIR methods, which are usually developed with modern commercial recordings of popular music in mind. We discuss some of these special features of early recordings and our approach to them in section 2.1.

In section 3 we discuss the chroma features we use on the historical recordings, introducing a new feature, the *NNLS chroma* (Non-Negative Least Squares chroma), which proves to offer great advantages for this task. Here we also discuss certain aspects of the search method we adopt in using the OMRAS2 specialized audio search system, audioDB.

Section 4 gives further details of our test collection and some of its special features. We describe the retrieval experiment we carried out on the test, which demonstrates clearly the advantage of a musically-informed approach, as is the case with *NNLS chroma*; this is followed in Section 5 by a discussion of the results and mentions further work we shall be doing in the near future.

2. CLASSICAL WORK RECOGNITION

We situate the research described here as a step towards automatic metadata enrichment. The long-term aim, simply put, is to develop a system which can help to identify classical works in a collection of digital audio whose descriptive metadata is either incomplete or inaccurate; the more modest task reported here is the identification of classical works that appear more than once in a collection of digitized historical recordings. Such duplicates may range from identical repetitions of the same digital file, through multiple digitizations (with or without different parameter settings) of the same 78-rpm disc, different performances by the same or different artists, to re-scorings, arrangements, extracts and medleys, examples of all of which occur in our test collection.

In order to evaluate our method's performance on this task, we have to establish 'ground-truth' in the form of a list of duplicates and 'covers' within the test collection. In principle, we should be able to process the accompanying machine-readable metadata for this, but, for various reasons, this was not possible, so this has inevitably been a largely manual process (see 4.2, below). Since almost all commercial historical recordings carry clearly-printed labels, in general it should not be hard naively to identify the works performed on a 78-rpm disc or set of discs. However, once the music has been digitized and separated from this graphical information (as was the situation for us), the problem becomes potentially more complex. In general, for example, we cannot identify tracks with works in a one-to-one correspondence, as will be discussed below.

Furthermore, classical works often – perhaps usually – comprise more than one movement. In the experiment reported here we actually treat work-*movements* as if each was a separate 'work'; we make no attempt to categorise different movements as belonging to the same work, an exercise that would presuppose a degree of musical unity which cannot be said to apply universally. In a different use-case, matching music between different movements of a work may be of great interest to musicologists, as may close matches between different works, or even works by different composers. Similarly, we ignore multiple matches of musical sequences within a single track, although this is of central importance for musical structure analysis.

If classical work-recognition could be robustly achieved with historical recordings despite their technical drawbacks (discussed below) this would offer a useful tool for metadata enrichment when used online in conjunction with a standard reference collection of recordings with high-quality metadata.

2.1 Early Recordings

Some of the special features of early recordings which can cause problems in audio analysis, and thus in audio MIR, are: limited frequency range, surface noise, distortion, variability of pitch (both global and local) and the problem of side-breaks. We briefly mention some of these in this section, though space precludes a full discussion here.

The frequency range attainable in 78-rpm recordings ranged from 168–2000 Hz in early acoustic recordings to 100–5000 Hz in electrical recordings from 1925. However, this is complicated by the various degrees of equalization that were applied to compensate for the fact that mechanical recording systems respond much more

[2] For a detailed overview of the special features of early recordings that need to be borne in mind, see [4].

[3] Useful lists of URLs for online collections of historical recordings are [5, 6]; to these [7, 8, 9] should be added.

strongly to low-frequency sounds, leading to various kinds of distortion when global gain levels are adjusted to capture the higher frequencies [9]. In this research, we take on trust the work of the professional transfer engineers who carried out the digitizations.[4]

The most immediately obvious difference between a 78-rpm recording and a modern digital one is the amount of broadband background noise known as *'surface noise'*; this has various causes, usefully summarized in [10]. There is often other noise present, usually due to mechanical aspects of the recording process. Not all of this can be completely eliminated by digital techniques, especially when it has a more-or-less definite 'pitch'. Problems due to broadband noise can mostly be avoided by using chroma features such as *NNLS chroma*, designed to ignore the non-harmonic components from percussion instruments. Distortion is a common feature in early recordings, like noise due to a variety of causes, and it is a problem that cannot easily be sidestepped. We have observed that highly-modulated loud passages in certain recordings tend to be distorted and often behave anomalously in content-based matching. This will need to be the subject of future research.

As Daniel Leech-Wilkinson demonstrates[5], the pitch of early recordings is by no means reliable; in general we can neither be sure of the global pitch-standards used by the performers (e.g. A=440Hz) nor of the actual frequencies sounding in the studio during recording. We mention some strategies for overcoming this problem in Section 3, below. The problem of side-breaks is addressed in Section 4.

3. FEATURE SELECTION & SEARCH

The classical work-recognition problem is close, though not identical, to the well-known MIR 'cover song' problem. In fact, in some respects it is somewhat simpler, since cover songs vary from their original model in ways that are generally unpredictable and can occur in several directions simultaneously. In general, we can be fairly sure that sequences of pitch-based data will be more-or-less invariant between recorded instances of the same work. This is more likely to be true where the scoring and instrumentation are the same, and both performers are working from the same (or a similar) score; where more radical re-arrangement or rearranging of the music has taken place there will be less similarity. For this reason, we match sequences of chroma features, rather than whole-track features; unless the latter embody some notion of sequence (as might be the case in an n-gram model) the number of false positives is likely to be high, since many work-movements in the same key and using the same general harmonic language are likely to share similar overall pitch-class content. Furthermore, chromas are robust to variation in timbre, which allows us to match radically-differing instrumentations (subject to limits of noise caused by percussive sounds or distortion).

In this paper we compare the performance of two chroma features in our classical work-recognition task. These are: (a) 36-bin chroma features extracted using fftExtract [11] (FE); (b) the new *NNLS chroma* features described in the following section (NNLS).

3.1 NNLS Chroma

In this section we give a brief description of the new 12-dimensional *NNLS chroma* feature which has been developed for the purpose of chord transcription [12]. But first we explain our reasons for comparing this with the performance of 36-d chromas in the same task, as this may not be immediately obvious.

One commonly-observed feature of early recordings is that they were often recorded on machines operating at different speed than the standard 78 revolutions per minute that was normal. An additional complication here is that we cannot always be sure what pitch-standard was being used by the performers; a variety of pitch standards have co-existed across the world of music in the past, some flatter, some sharper than today's accepted standard of A=440Hz. While there is little we can do to reconcile these conflicting sources of error, we can allow some tolerance in matching covers recorded at different global pitch standards by using three (or more) bins per equal-temperament semitone bin in the chroma feature, rotating the query by plus or minus a single bin at query time, and choosing the best match from these three queries. We present results using non-rotated queries and also rotated by ± one semitone below.

Although our new *NNLS chroma* features have only 12-dimensions, corresponding to the 12 chromatic pitch classes of conventional music theory, they are derived from a spectrogram with three bins per semitone, with the intention of achieving a similar invariance to small pitch deviation; the most important practical difference is that a single exhaustive search of a collection of 12-dimensional features will inevitably be more efficient than three searches of one of 36-dimensional features.

NNLS chroma features are obtained using a prior NNLS-based approximate note transcription [12, 13]. We first calculate a log-frequency spectrogram (similar to a constant-Q transform), with a resolution of a three bins per semitone. We derive the tuning of the piece in a quartertone neighbourhood of 440 Hz and adjust the log-frequency spectrogram by linear interpolation such that the centre bin (of the three) of every note corresponds to the fundamental frequency of that note in equal temperament, as is frequently done in chord- and key-estimation [e.g. 14], we adjust the chromagram to compensate for differences in the tuning pitch. First, the tuning is estimated from the relative magnitude of the three bin classes. Using this estimate, the log-frequency

[4] http://www.charm.kcl.ac.uk/history/p20_4_4_1.html
[5] [4], chapter 3.1, heading 'Misrepresentations in early recordings'

spectrogram is updated by linear interpolation to ensure that the centre bin of every note corresponds to the fundamental frequency of that note in equal temperament. The spectrogram is then updated again to attenuate broadband noise and timbre. This is done using a kind of running standardization combining the removal of the background spectrum and a form of spectral whitening.

We assume a linear generative model in which every frame Y of the log-frequency spectrogram can be expressed approximately as the linear combination $Y \approx Ex$ of note profiles in the columns of a dictionary matrix E, weighted by the activation vector x. Finding the note activation pattern x that approximates Y best in the least-squares sense subject to $x \geqq 0$ is called the non-negative least squares problem (NNLS). We choose a semitone-spaced note dictionary with exponentially declining partials, and use the NNLS algorithm proposed by Lawson and Hanson [15] to solve the problem and obtain a unique activation vector. This vector is then mapped to the twelve pitch classes C,...,B by summing the values of the corresponding pitches.

In the work reported here, our feature-vectors are all averaged into one-second frames; future work will investigate the effect on retrieval of using finer granularity. Similarly, we do not consider here the effect of low-level DSP parameters such as FFT window-length, using default values in most cases.

3.2 Search

Searching was carried out using the audioDB software developed at Goldsmiths College in the OMRAS2 project [16]. Independent audioDB databases for each feature-set were searched for best matches by Euclidean distance between queries and items in the database specified as feature-vector sequences of a given length.

4. TEST COLLECTION & EXPERIMENT

The collection of audio files we used is a subset of one provided by the King's Sound Archive (KSA) which represented the set of their digitisations completed by February 2009. The current KSA is considerably bigger, numbering over 4,500 sides with highly-reliable metadata, and free download access to most of the collection is available via a metadata-searchable web interface.[6]

4.1 King's Sound Archive (KSA)

The King's Sound Archive is based on the BBC's donation of their holdings of duplicate 78-rpm records in 2001; KSA now holds over 150,000 discs including classical and popular music as well as spoken-word and sound-effect recordings from c. 1900 to c. 1960.[7]

Our test collection comprises digitizations (undertaken in the CHARM project [17]) of 2,017 78-rpm sides, mostly classical but including some jazz, spoken-word and sound-effect recordings. This number was arrived at by chance, being the number of sound files that we could process conveniently and reconcile with the metadata provided by the KSA. We received the sound-files before the detailed discographical data now on the CHARM web-site[8] was ready; we thus had to rely on the technical production metadata, which was not primarily concerned with work identification, although it did include catalogue numbers from the disc-labels as well as the 78-rpm matrix numbers. This necessitated a lot of manual metadata editing.

4.2 Relevance judgements

In the metadata editing process we have identified a 'cover list' of around 88 works for which duplicates or multiple performances exist in the test collection. This list forms the basis for relevance judgments in our experiments. We are aware that there are often other 'relevant' tracks for a given query, but since our experiments are comparative in nature we do not regard this as a problem; in fact their effect is likely to be detrimental to our precision results.

The existence of multi-side recordings of work-movements in the collection complicates the issue of establishing reliable relevance judgments. (The various possibilities for the disposition of work-movements and side-breaks is shown diagramatically in Figure 1.) While it is often the case that the same musical material is repeated or alluded to throughout a single movement of a classical work, we cannot be sure that such repetitions are distributed evenly so that each 78-rpm side over which a movement is spread contains a roughly-equal proportion of similar musical material. Furthermore, side-breaks do not always occur at the same point in the music.

There are two basic approaches that can be taken to solve this problem, both of which present some difficulty: post-processing of results or pre-manipulation of the data. Given a list of tracks in the database (each of which corresponds to a 78-rpm side), we can post-process the search results so as to regard as mutually-relevant all matches between a query and tracks that come from anywhere in the same movement. Alternatively, we can in a preprocessing stage digitally concatenate tracks that we suspect are from the same movement. Clearly the latter procedure does not fairly represent the case where we cannot rely on our metadata, and the exact correspondence between sides and movements is unclear. In our experiment we adjusted the lists to consider as relevant only sections from similar sections of a work-movement (so 'side 1' of a given recording of a work-movement, say, is not considered relevant to 'side 2' of another recording of the same movement); while we acknowledge the limitations of this approach it does not affect our comparative evaluation.

[6] www.charm.kcl.ac.uk/sound/sound_search.html
[7] www.kcl.ac.uk/schools/humanities/depts/music/res/ksahistory.html
[8] http://www.charm.kcl.ac.uk/discography/disco.html

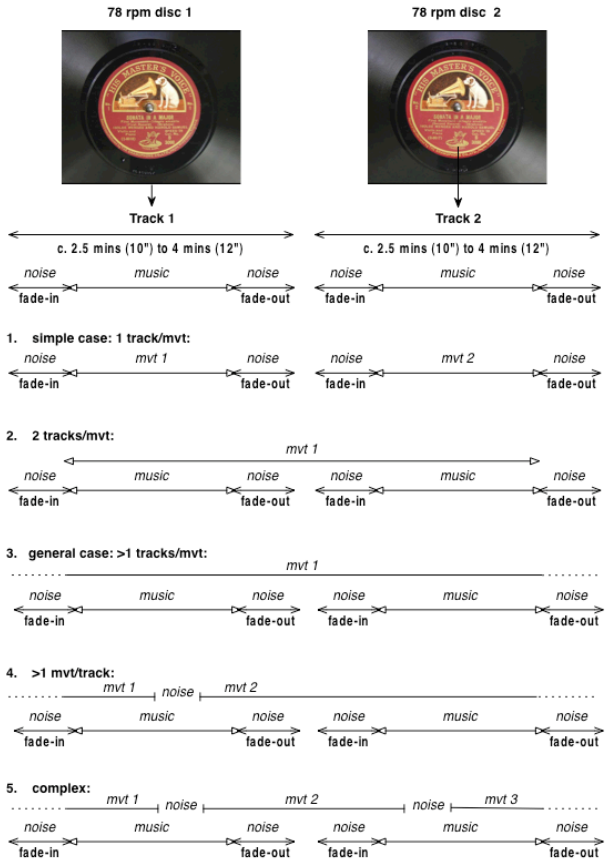


Figure 1. Disposition of work-movements on the sides of 78-rpm gramophone records.

One problem that is not solved either way is that many of our database tracks contain material from more than one movement (as in Figure 1, cases 4 and 5); furthermore, because all performances are not necessarily at the same tempo, or do not observe the same repeats, or are recorded on 78-rpm discs of different size (and consequent time-duration) the same pattern of side-breaks is generally not duplicated. This is one good reason why we use sequence-based matching, rather than using whole-track features in which material extracted from the whole of a track – even if some of it comes from a different movement in a different key – are consolidated into a single feature-vector.

4.3 Experiment

4.3.1 Method

We extracted one-second features as described in Section 3; we built an audioDB database with each set of features corresponding to the 2,017 tracks; with each pre-discovered 'cover' track in turn as query[9], we searched the database for best query/track matches of various-length sequences of feature vectors. We found that a sequence of 25 vectors consistently gave the best retrieval performance for this task across all tested features. We repeated the search with the queries rotated by up to a semitone flat and sharp (± 1 bin for NNLS; ± 1 and 2 bins for FE) taking the best result for each search.

4.3.2 Results

The 11-point recall/precision graph in Fig. 2 and the average precision values in Table 1 show a dramatic improvement (20%) in performance in this particular task brought about by the use of the *NNLS chroma* feature as opposed to the 'standard' chroma we used. While query-rotation for both features significantly improved retrieval performance, NNLS still did far better than FE. Bearing in mind the generally poor acoustic quality of the recordings in the collection, this is particularly encouraging and suggests that the new feature will be generally useful for classical work-recognition tasks on collections of higher recording quality, though as yet this remains to be tested.

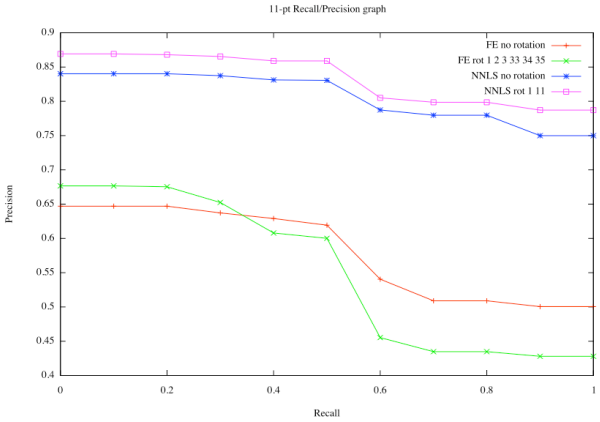


Figure 2. 11-point interpolated Recall/Precision graph for the classical work-recognition task.

	NNLS chroma	NNLS rotated	FE chroma	FE rotated
Average precision over all rel docs	0.80	**0.83**	0.57	0. 54

Table 1. Average precision (non-interpolated) for non-rotated and rotated queries over all 322 relevant tracks.

4.3.3 Discussion

The improvement in the retrieval performance of the system with the *NNLS chroma* feature (and no other changes) is striking; particularly since the feature was not designed with search or similarity judgments in mind. The model underlying it (described in section 3.1) does attempt to capture similar note content (in a way that generic chroma features attempt to capture similar acoustic pitch content) but there is potential to perform even better than our current results by tuning the NNLS features to better reflect perceptual musical similarity.

[9] The query track will, of course, be in the database at search-time; since the identity-match is always returned at the top of the ranked result-list we adjust the result-lists accordingly for our evaluation.

The fact that a query-sequence length of 25 seconds/vectors gave the best retrieval results with all features may be useful in distinguishing complete 'covers' of the kind we are dealing with here from works by classical composers which contain references to, or quotations of, other music. In general these are most likely to be short. However, this interesting topic needs to be the subject of further investigation.

5. CONCLUSIONS AND FUTURE WORK

We have demonstrated that using a chroma feature based on prior NNLS approximate transcription gives a 20% improvement in retrieval performance over conventional chroma for the work-recognition task that is the main focus of this paper. Since this copes with historical recordings of varying quality and a number of the special features of the collection (such as the arbitrary distribution of movements across 78-rpm sides), we are encouraged to hope that it will prove a particularly effective feature for general musical work recognition in other MIR contexts.

Amongst other work, then, we plan to characterize the details of the *NNLS chroma* feature in order to be able to align it better to human judgments of note-content musical similarity, as well as designing other audio features which reflect other aspects of musical sound such as timbre or rhythm.

6. ACKNOWLEDGMENTS

This work was supported by EPSRC grant EP/E02274X/1 and the NEMA (Networked Environment for Music Analysis) project funded by the Andrew S. Mellon Foundation. We are most grateful to Daniel Leech-Wilkinson, Andrew Hallifax and Martin Haskell for providing data from the King's Sound Archive. Thanks also to Ben Fields for help in various ways.

7. REFERENCES

[1] R. P. Smiraglia: "Musical works as Information Retrieval Entities: Epistemological Perspectives," Proceedings of the 2nd Annual Symposium on Music Information Retrieval (ISMIR 2001, Bloomington, Indiana, U.S.A.), 85-91

[2] Anon, "ID3", *Wikipedia*, website: http://en.wikipedia.org/wiki/ID3

[3] Anon, "Functional Requirements for Bibliographic Records", *Wikipedia*, website: http://en.wikipedia.org/wiki/Functional_Requirements_for_Bibliographic_Records

[4] D. Leech-Wilkinson: *The Changing Sound of Music: Approaches to Studying Recorded Musical Performance*, online publication, London, 2009: http://www.charm.kcl.ac.uk/studies/chapters/intro.html

[5] Thomas Edison National Historical Park, Links to Online Recordings, web-site: http://www.nps.gov/edis/photosmultimedia/links-to-online-recordings.htm

[6] Anon, 'Historic Sound Recordings Around the Web', web-site: http://www.phonozoic.net/listening.html

[7] DISMARC web-site: http://www.dismarc.eu/

[8] British Library, 'Archival Sound Recordings', web-site: http://sounds.bl.uk/

[9] Anon, "Gramophone record", *Wikipedia*, website: http://en.wikipedia.org/wiki/Gramophone_record

[10] R. Wilmut, "Reproduction of 78rpm records", website: http://home.clara.net/rfwilmut/repro78/repro.html#s/n

[11] fftExtract website: http://omras2.doc.gold.ac.uk/software/fftextract/

[12] M. Mauch and S. Dixon: "Approximate Note Transcription for the Improved Identification of Rare Chords," Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010, Utrecht, The Netherlands)

[13] M. Mauch: "Automatic Chord Transcription from Audio Using Computational Models of Musical Context", unpublished PhD dissertation (Queen Mary University of London, 2010)

[14] C. Harte and M. Sandler: "Automatic Chord Identification using a Quantised Chromagram," Proceedings of 118th Convention of the Audio Engineering Society, 2005

[15] C. L. Lawson and R. J. Hanson: *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974

[16] audioDB website: http://omras2.doc.gold.ac.uk/software/audiodb/

[17] AHRC Centre for the History and Analysis of Recorded Music (CHARM); website: http://www.charm.kcl.ac.uk/

RECOGNITION OF VARIATIONS USING AUTOMATIC SCHENKERIAN REDUCTION

Alan Marsden
Lancaster Institute for the Contemporary Arts, Lancaster University, UK
A.Marsden@lancaster.ac.uk

ABSTRACT

Experiments on techniques to automatically recognise whether or not an extract of music is a variation of a given theme are reported, using a test corpus derived from ten of Mozart's sets of variations for piano. Methods which examine the notes of the 'surface' are compared with methods which make use of an automatically derived quasi-Schenkerian reduction of the theme and the extract in question. The maximum average F-measure achieved was 0.87. Unexpectedly, this was for a method of matching based on the surface alone, and in general the results for matches based on the surface were marginally better than those based on reduction, though the small number of possible test queries means that this result cannot be regarded as conclusive. Other inferences on which factors seem to be important in recognising variations are discussed. Possibilities for improved recognition of matching using reduction are outlined.

1. SCHENKERIAN REDUCTION

Earlier work [6] has shown that Schenkerian analysis by computer is possible, though not easy. (Currently only short segments of music can be analysed, and confidence in the analyses produced cannot be high.) The aim of the research reported here is a first attempt at testing whether these automatic analyses produce information which is useful for information retrieval.

Schenkerian analysis is a technique, with a long pedigree in music theory, which aims to discover the structural 'framework' which is believed to underlie the 'surface' of a piece of music (see [1], for example). Reduction according to the theory of Lerdahl & Jackendoff, which has also been subject to computational implementation [2], is broadly similar. Figure 1 shows the first four bars of the theme of a set of variations for piano by Mozart, and its reduction as derived by the software used here. (This is by far the simplest of the themes used here; to show other themes and their reductions would take more space than is available.) Schenker's reductions were notated in a different fashion, and also included information not given here, but the basic information of which pitches are regarded as more 'structural', and so included in the higher levels, is similar.

The research reported here fits into that body of MIR research which aims to improve MIR procedures through the application of ideas from music theory.

2. VARIATIONS

A common type of composition in classical music is 'theme and variations'. In this kind of piece, a theme is presented, followed by a number of variations of that theme. There is no single and established definition of what constitutes a variation of a theme, but in the Classical period (the period of Haydn, Mozart and Beethoven) it is clear that a variation is not simply the presentation of the same melody in different arrangements (as it was for some later composers) but rather a composition which has the same structural features as the theme. This is particularly clear in Mozart's variations: they are almost always the same length as the theme, have the same number of phrases, and have matching cadences for those phrases (at least in their harmony; often in other features also). The

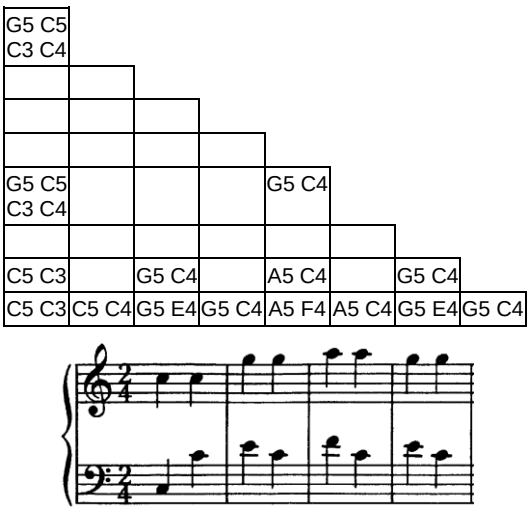


Figure 1. The first four bars of the theme of Mozart's variations K.265 (using a tune known in English as 'Twinkle, twinkle little star'), and the highest-scoring reduction derived from these bars by the software.

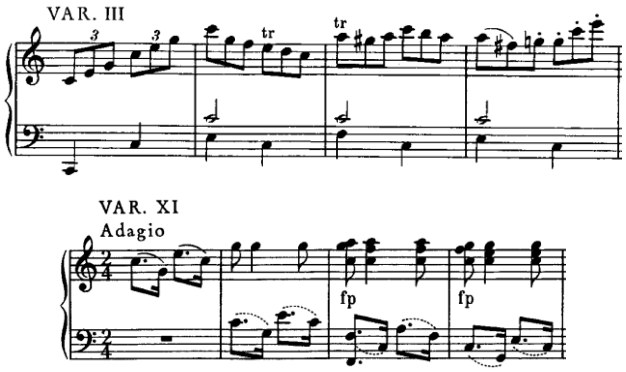


Figure 2. The beginning of two variations of the theme shown in Figure 1.

internal structure of those phrases can also show common features: the harmony is often similar; there can be common notes, especially in important positions like beginnings and endings, and the variation sometimes clearly gives a decorated version of the melody and/or bass of the original. Figure 2 shows the first four bars of two variations of the theme shown in Figure 1.

If Schenkerian analysis validly reveals musical structure, then the analysis of each variation should, to some degree, match the analysis of the theme. To test this requires analyses of variations and themes which are unbiased in the sense that the analyses of each variation should be made with no knowledge of the theme. To achieve unbiased analyses with human analysts would be very difficult: expert analysts are required, and one would have to recruit as many analysts as there are variations in a set. Furthermore, it is well known that different analysts produce different analyses, and it would be difficult to neutralise these personal differences. The computer software described below gives a means for generating unbiased analyses, and so allows this kind of empirical test of the validity of Schenkerian analysis.

3. REDUCTION SOFTWARE

The method of reduction used here is described more fully in [6]. There is space here only to give a brief outline. An analysis of a piece is a binary tree whose leaves are the 'segments' of the surface of the music (the notes of the score). A segment is a span of music, containing all the notes sounding at that time. At least one note begins at the start of the segment and at least one note finishes at its end. No notes begin or finish at other points within the span. A note is defined by its pitch and by whether or not it is tied to a note in the preceding segment. A single note in the score can be split into a series of tied notes across several segments.

Segments above the surface are related to a pair of 'child' segments through a set of 'atomic elaborations'. These define how a note in a higher-level segment can be elaborated to become two shorter notes (or a note and a rest) in the child segments. The set of atomic elaborations is derived from Schenkerian theory and consists of such things as repetitions, neighbour notes, anticipations, consonant skips, etc. Atomic elaborations can imply that certain pitches are consonant, and the implications of the set of atomic elaborations relating a higher-level segment to its children must be consistent (i.e., the consonant pitches must form an acceptable harmony).

An analysis is therefore a kind of parse tree employing a grammar defined by the atomic elaborations. The software used here effectively employs a chart parser [4] as a step towards generating such a tree, but the computational complexity of the algorithm is of order $O(n^4)$ time. With typical computing resources, it is therefore possible to derive a parse chart from extracts of simple piano music up to only four to eight bars in length.

The parse chart is a triangular matrix whose cells contain the possible reductions at each stage of reduction. The bottom (longest) row contains the segments of the surface. The first row above contains segments which result from reduction of each of the pairs of consecutive segments below. Rows further above contain segments which result from reductions of those with other segments, etc., until the top row, with just one cell, contains the segments which derive from reduction of the entire extract. The top part of Figure 1 shows a reduction chart in which the best-scoring analysis has been selected (see below). Most of the cells of this chart are empty and those that are not contain just one segment, each containing two to four notes. Before an analysis is selected from a chart, its cells are generally fuller, and each contains a number of segments corresponding to the different ways in which a group of surface segments may be reduced. Each derived segment has an associated score, intended to suggest how likely that segment is to be a part of a complete 'good' analysis of the entire extract.

An analysis can be derived from the chart by selecting a high-scoring segment in the top cell, and then recursively selecting its highest-scoring children until a complete tree to all the segments of the surface has been derived. However, complications of context-sensitivity mean that selecting the locally highest-scoring children at each stage does not guarantee the highest-scoring complete analysis. The current procedure to ensure derivation of the highest-scoring analysis from the chart is of exponential complexity, so in some cases a chart containing information on possible analyses can be derived, but it is not practical, by current means, to derive a single best analysis from this chart.

The research reported in [6] derived some scoring mechanisms by comparing the output of the analysis-derivation software with pre-existing analyses of the same pieces. One can therefore have some confidence in the scores the software derives, but because of a lack of read-

ily available test material, the research so far has been based on a very small quantity of music (just five short themes by Mozart). At this stage, therefore, results from research in this general area can only really be regarded as provisional.

That earlier research also showed that low-scoring possible reductions can be omitted from the chart, vastly reducing the computation time required for its derivation, without jeopardising the derivation of a good analysis. This project has used the same limits as outlined in [6]. In deriving the reduction chart, no more than 25 segments were recorded in each cell, discarding lower-scoring possibilities if necessary. In [6] scores were computed from comparison of good analyses with random analyses containing an *Ursatz* (a structure Schenker regarded as indicating a complete musical statement). In this project, the extracts of music do not constitute complete statements (most importantly they often do not end on the tonic), so new scores were computed from the same raw data from comparisons of good analyses with random analyses, regardless of the presence of an *Ursatz*. The new scores were similar to the old ones.

Small changes were also made to the set of possible atomic reductions because certain configurations not found in the five themes used in [6] were found in the material used here. An 'échappée' (a following incomplete neighbour note) elaboration was added, with tight harmonic constraints. New harmonic constraints, looser in some respects but tighter in others, were defined for some elaborations to allow situations where a dissonant note can be elaborated by 'repetition', 'delay' or 'shortening' (i.e., being preceded or followed by a rest).

4. RECOGNISING VARIATIONS

The objective of the research reported here is to explore mechanisms for recognising whether or not a passage of music is a variation of a given theme, and in particular to test whether or not a procedure using reduction yields better recognition than one relying only on the 'surface' of the music. To be precise, if a procedure which uses reductions of the theme and variations produces better results than a similar procedure which does not use reductions, then we can conclude with some confidence that the reduction software does produce useful information concerning musical structure.

4.1 Materials

The materials used in this project are encodings made by myself of four bars from the theme and most of the variations of 10 sets of variations for piano by Mozart: K. 179, 180, 264, 265, 352, 354, 398, 455, 573 and 613. These are all the sets of variations in section 26 of the Neue Mozart Ausgabe—the source used—with the exception of two sets written when Mozart was nine years old, and which cannot therefore safely be regarded as mature compositions, one set in the metre 6/8, and one which has a theme beginning and ending half-way through a bar. In all but one case it is the first four bars which are used. In K. 613 the first four bars are taken from the theme proper, which begins after an introduction. In each case the four bars form a coherent phrase. Variations in a minor key, or in a different metre from the theme, were omitted. Some small changes to the music were required in order to facilitate successful reduction by the software: all anacruses (pickups) were omitted as the reduction software cannot cope with these; all grace notes, and trills plus any terminating turn, were omitted; in a very few cases notes from some middle voices were omitted because the software operated with a limit of 4 notes in a segment; notes at the end of the last bar which clearly led into the following bar rather than belonging to the first phrase were omitted.

The encoding gave the pitch of each note (the pitch spelling of the score is used in the encoding, but pitches are converted to MIDI values in the software) and its duration. Voices are indicated, and were determined by hand when the encoding was made. This information is used only when matching surfaces as the reduction procedure changes the composition of voices.

To neutralise differences of key, each theme and variation was transposed to the key of F major, a key selected because it allowed each entire set of theme and variations to be transposed in the same direction and still remain in range for the software. It is not so simple to neutralise differences of metre, so themes in a triple metre were only compared with variations in a triple metre, and similarly for themes in a duple metre. This made a corpus in two parts, for duple and triple metres, of 5+5 themes and 41+36 variations. This is not a sufficiently large corpus for definitive results, but further materials are not readily available.

4.2 Procedure

A reduction 'chart' (i.e., a matrix of the possible reductions) was derived from each of the extracts of themes and variations, using the software as described above. (This took about 24 hours of computing time.) The best-scoring analyses were derived for each of the themes. (This was not possible for the variations because of the excessive demand of computing time in some cases.)

There has been considerable research on techniques of measuring melodic similarity (see, for example [3]), but to ask if some extract of music is a variation of another, at least in the case of 'Classical' variations as described above, is not the same as to ask if two extracts are similar. Some work in measuring melodic similarity has attempted to make use of concepts of structure from music theory [5, 7], with encouraging results. Unlike that work, the research reported here is concerned with full textures rather than just melodies, and unlike [7], which shares some of

the underlying concepts of this work, the comparison method requires no manual intervention (though it does make use of an encoding which gives the key and metre). Instead a large number of methods specialised to comparing extracts to determine if one is a variation of the other, both at the surface and comparing a best analysis to a reduction chart, were implemented in software. Each method resulted in a single match value for each pair compared. If a comparison method is successful, it will consistently yield higher values for comparisons between a theme and a variation of that theme than between a theme and a variation of a different theme.

4.2.1 Comparison Methods

Similar principles were used in the design of the methods for comparing both surfaces and reductions, as follows.

1. **Pitch-matching: pitches/pitch classes.** Some methods count exactly matching pitches; some methods accept matching pitch classes (i.e., the matched pitch can be transposed up or down any number of octaves).
2. **Voices to test: all/melody+bass/melody/bass.** There are four different kinds of match under this heading: those which seek to match all notes of each segment from the theme, those which match only the melody and bass, those which match only the melody, and those which match only the bass. For reduction matches, the lowest note of a segment is taken to belong to the bass and the highest to the melody.
3. **Voice-matching: yes/no.** Some methods only accept matches of pitches in the same voice; some accept matches no matter in which voice the note occurs in the variation. The concept of voice used here is only 'melody', 'middle' and 'bass'. The middle contains all the notes which are not in the melody and bass.
4. **Match tied notes: yes/no.** Some methods seek to match only notes which are not tied to a preceding note, while others seek to match all notes.
5. **Weighting by duration: yes/no.** Some methods weight matches in proportion to the duration of the segment in the theme to be matched.
6. **Weighting by metre/level: yes/no.** In surface-matching methods, matches can or cannot be weighted by the metrical level of the beginning of the note, giving notes at the beginning of the bar the greatest weight. (The metre of a piece is specified in the encoding.) In reduction-matching methods, the corresponding weight is determined by the level of the segment in the analysis tree. Weight steadily decreases from the root to the leaves.
7. **Limiting by parent match: yes/no (reduction only).** Some matching methods for reductions limit the level of match found for child segments to be no greater than the level of match found for their parents, on the grounds that matches of children when the parent is not matched are accidental.
8. **Values: present/proportion/bar; maximum/average/score-weighted/score-weighted*2.** Different values can be recorded for any individual segment. In the case of surface matches, some methods only look for a matching pitch to be present within the time span occupied by the original pitch. In other cases, the proportion of the original time span during which a matching pitch is sounding in the variation is used. In yet others, it is sufficient merely for a matching pitch to be present somewhere with the same bar, since variations clearly sometimes involve changes in rhythm. For reduction-matching measures, a segment of the theme can be matched with up to 25 possible segments in the reduction chart for the variation. In different methods, four different values are recorded: the maximum match; the average match; the average match weighted by the score of the matching segment; and the average match weighted by the square of the score of the matching segment. Score weights are computed in relation to the maximum weight in a reduction chart so as to always fall in the range 0 to 1 and decrease exponentially in relation to decreases in score.

The combination of all these parameters results in 384 comparison methods for surfaces and 1024 for reductions. In each case, the match value for a segment is based on the number of notes from the segment of the theme which are matched in the corresponding segments of the variation, divided by the number of notes to be matched, weighted as appropriate by proportion for surfaces or score for reductions with parent-match limiting applied if appropriate. The overall result of a comparison between a theme and a variation is the average of the results from matching each segment of the theme (and its reduction, if appropriate) with the corresponding segments of the variation (and its reduction), weighted by duration and/or metre/level as appropriate.

4.2.2 Testing Methods

Every theme was compared with every variation in the same class of metre—those which were variations of this theme and those which were variations of another theme—using each of the comparison methods outlined above. Each test can be thought of as retrieval from a database using a theme as the query. A perfect response would retrieve all the variations of that theme, and none of the variations of other themes. An appropriate measure of success is therefore the F-measure, the harmonic mean of 'precision' (the proportion of correctly retrieved variations to the total retrieved) and 'recall' (the proportion of correctly retrieved variations to the total number of variations for that theme).

A simple query mechanism would retrieve all variations whose comparison with the theme yields a value above a certain threshold. Possible values for this threshold lie between the lowest value for any comparison between a theme and one of its variations, and the highest value for any comparison between a theme and a variation of a different theme. For each comparison method, the average F-measure, using each theme as a query, was computed, at each candidate value of the threshold. The best possible F-measure (on this corpus) using each comparison method was thus be computed.

An alternative test is to ask, for each variation, of which of the five candidate themes is it a variation. The simple answer would be the theme which yields the highest comparison value. This test will be called the 'recognition measure, and for each comparison method the value recorded is the percentage of variations whose theme is correctly recognised.

5. RESULTS

The main hypothesis of this study, that reduction will lead to better recognition of variations, is not confirmed by the results, as shown in Table 1. In fact twelve of the 384 methods comparing surfaces produced a better average F-measures than the best reduction-comparing method, and two produced better recognition measures. The difference is small, however. It is impossible to know without further research whether this is because the fundamental idea that variations share common reductions is mistaken, or whether it is because the reductions produced by this reduction software are incorrect. Currently there is no simple way of determining the correctness of an analysis.

The values of match between the analysis of a theme and the reductions of its variations are generally high, but they can also be high for reductions of variations of other themes. This is illustrated in Figure 3, which shows a graph of the match values for K. 265, using the best reduction-matching method (matching pitch classes from the melody and bass in the appropriate voice in the variation, but not matching tied notes; weighted by duration but not level and not limited; taking the maximum match among alternative segments). The best threshold value for this comparison method is 0.78, which causes one variation of this theme not to be recognised, and a number of false positives from variations of other themes. According to Schenkerian theory, pieces of tonal music become more alike each other the higher up the structural tree one looks, until all (proper) pieces share one of only three possible *Ursätze*. Perhaps the reduction-matching methods have been confounded by this underlying universal similarity.

The match values for surface matches are typically lower and more spread out, as illustrated in Figure 4, which shows the results for the same theme using the best surface-matching method (matching all pitch classes in the appropriate voice in the variation, including tied notes; weighted by duration but not metre; taking the proportion a pitch class is present in a segment's span). The best threshold for this method is 0.36, causing all variations of this theme to be correctly recognised but also a false positive.

	Surface methods		**Reduction methods**	
	Average F-meas.	Recog. measure	Average F-meas.	Recog. measure
Best	0.867	94.8%	0.842	90.9%
Average	0.776	74.8%	0.748	70.3%
Worst	0.540	42.9%	0.671	35.1%

Table 1. Summary results.

5.1 Factors leading to better recognition

Analysis of the results indicates that many of the factors listed above make little difference to the quality of a recognition method. One notable exception is that weighting by level in the case of reduction matches generally leads to worse results. This is consistent with the general conclusion above that reduction does not lead to better recognition of variations. Also consistent with this is a weaker result that weighting by duration does not improve recognition in the case of reduction matches, probably because higher-level segments are likely to have longer durations. In the case of surface matches, however, weighting by duration, but not by metre, leads to a slight improvement.

On average, counting a surface match simply by the presence of the required pitch or pitch class within the span of a segment gives slightly better results than measuring the proportion of the span in which it is present, and both give better results than counting matches anywhere within the bar. However, there are interdependencies among the various parameters. For example, when pitch classes are matched within voices, measuring the proportion gives consistently better results.

In the case of reduction-based methods, taking the maximum match among alternative segments yields the best results, on average. This is consistent with the idea that variations should have reductions which match the reductions of the theme. The listener hears the theme first, and so ambiguities in the structure of variations can be resolved by reference to the structure of the theme. It is therefore sufficient that there be some *possible* reduction of the variation which matches the theme.

In both surface- and reduction-based methods, the worst results come from matching only the bass, followed by matching only the melody. The difference between matching all notes and just the melody and bass is small. In every case, if pitch classes are matched, the best results come from matching them in the appropriate voices,

whereas if pitches are matched, the best results come from ignoring the voice in which they occur in the variation. This might be because sometimes Mozart writes a new part *above* the melody, and in such cases the melody often occurs at its original register.

5.2 Possible Improvements

A half-way house has been tested, which looked for matches of segments at higher levels only if there was no match at a level below. However, this produced no better results than those given above. Better results might come from matching melody and bass voices separately, possibly at different levels, but this has not yet been tested.

In examination of some of the false negatives and false positives, similarities and dissimilarities are revealed in the reductions which are not present at the surface, but as yet no consistent pattern has been discerned which would lead to a consistently better variation-recognition method. It is possible that harmony should be taken into account. (Harmonic analysis is a bi-product of the reduction procedure.) Matching on harmony alone, however, would not produce good results because many of the themes have similar harmonic structures; it would have to be combined with other factors.

Overall, variation has been found to be more complicated than first thought. The quantitative results do not show reduction to reveal the relationship between theme and variations, but examination of false results suggests that further research might yet show this to be the case.

6. REFERENCES

[1] Forte, A. & Gilbert, S.E. *Introduction to Schenkerian Analysis*, Norton, New York, 1982.

[2] M. Hamanaka, K. Hirata, and S. Tojo: "Implementing 'A Generative Theory of Tonal Music'", *Journal of New Music Research*, Vol. 35 No. 4, pp. 249–277, 2007.

[3] W. Hewlett and E. Selfridge-Field (eds.): *Melodic Similarity* (Vol. 11 of *Computing in Musicology*), MIT Press, Cambridge MA, 1998.

[4] D. Jurafsky and J.H. Martin: *Speech and Natural Language Processing* (2nd edition), Pearson, Upper Saddle River NJ, 2009.

[5] P. Kranenburg, A. Volk, F. Wiering and R.C. Veltkamp: "Musical Models for Folk-Song Melody Alignment", *Proceedings of the International Symposium on Music Information Retrieval*, pp. 507–512, 2009.

[6] A. Marsden: 'Schenkerian Analysis by Computer: A Proof of Concept', *Journal of New Music Research*, forthcoming.

[7] N. Orio and A Rodà: "A Measure of Melodic Similarity Based on a Graph Representation of the Music Structure", *Proceedings of the International Symposium on Music Information Retrieval*, pp. 543–548, 2009.

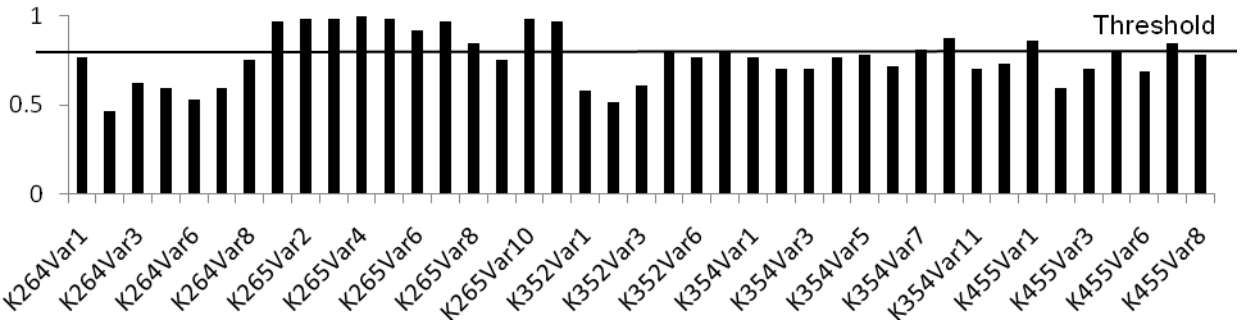


Figure 3. Match values for the theme of K. 265 using a reduction-based comparison method.

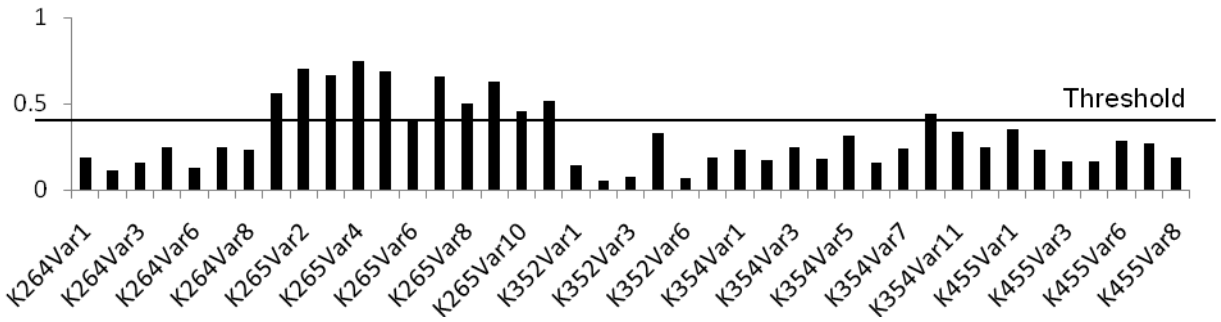


Figure 4. Match values for the theme of K. 265 using a surface-based comparison method.

SCALABLE GENRE AND TAG PREDICTION WITH SPECTRAL COVARIANCE

James Bergstra
University of Montreal
bergstrj@iro.umontreal.ca

Michael Mandel
University of Montreal
mandelm@iro.umontreal.ca

Douglas Eck
University of Montreal
eckdoug@iro.umontreal.ca

ABSTRACT

Cepstral analysis is effective in separating source from filter in vocal and monophonic [pitched] recordings, but is it a good general-purpose framework for working with music audio? We evaluate covariance in spectral features as an alternative to means and variances in cepstral features (particularly MFCCs) as summaries of frame-level features. We find that spectral covariance is more effective than mean, variance, and covariance statistics of MFCCs for genre and social tag prediction. Support for our model comes from strong and state-of-the-art performance on the GTZAN genre dataset, MajorMiner, and MagnaTagatune. Our classification strategy based on linear classifiers is easy to implement, exhibits very little sensitivity to hyper-parameters, trains quickly (even for web-scale datasets), is fast to apply, and offers competitive performance in genre and tag prediction.

1. INTRODUCTION

Many features for music classification have a longer history in speech recognition. One of the first steps taken by most speech recognizers is to transform audio containing speech into a sequence of phonemes. A phoneme is the smallest segmental unit of sound, for example the /b/ in "boy". For a speech recognizer to work with multiple speakers, it needs to generalize over a range of voice types (adult verus child, male verus female). To achieve this generalization it can be useful to separate the audio signal into two parts: the source excitation at the vocal cords and the transfer function (filtering) of the vocal tract. Cepstral analysis is commonly used to achieve this separation. The cepstrum C is defined as

$$C = |G\log(|Fx|^2)|^2 \quad (1)$$

where F is the discrete Fourier transform and G is the inverse discrete Fourier transform or the discrete cosine transform. One important property of the cepstrum is that convolution of two signals can be expressed as the addition of their cepstra.

In general cepstral analysis for speech recognition or music analysis is done on a power spectrum $|Fx|^2$ that has been downsampled (compressed) non-linearly to better model human perception of equidistant pitches (the Mel scale). The resulting cepstral-domain values are called Mel-Frequency Cepstral Coefficients (MFCCs). See Section 2 for details.

In the domain of music, cepstral analysis can be used to model musical sounds as a convolution of a pitched source (perhaps a vibrating string) and an instrument body that acts as a filter. This deconvolution of musical source (pitch) from musical filter (instrument) can be seen for several instruments in Figure 1. If our goal is to generalize across different pitches for a single instrument timbre, it is clear that the cepstral domain has advantages over the spectral domain.

The main use of source / filter deconvolution in speech is allow for the *elimination* of the source. This is achieved by only retaining the first few cepstral coefficients (usually 12 MFCCs for speech recognition). However, music is not speech. The assumption that recorded music consists of a filter with it's own sound quality (instrument timbre) acting on an irrelevant source is certainly false. For many instruments, it is difficult to distinguish between pitch and timbre, and certainly the assumption breaks down in polyphonic recordings.

This paper presents segment covariance features as an alternative to means and variances in MFCCs for condensing spectral features for linear classification. The covariance, together with mean and variance in spectral features provides a better basis for genre and tag prediction than those same statistics of MFCCs. These features are quick and easy to compute (pseudo-code given below) work well with linear classification. Together they offer a viable approach to web-scale music classification, and a competitive null model for research on new datasets.

2. FEATURES

We follow [1] in distinguishing two levels, or stages, of feature extraction. Firstly, at the *frame level* (typically 20-50 milliseconds) we extract features such as Mel Frequency scale Cepstral Coefficients (MFCCs). Secondly, at the *segment level* (typically 3-10 seconds) we summarize frame-level features via statistics such as the mean, variance and covariance. For our frame-level features we partitioned the audio into contiguous *frames* of either 512,

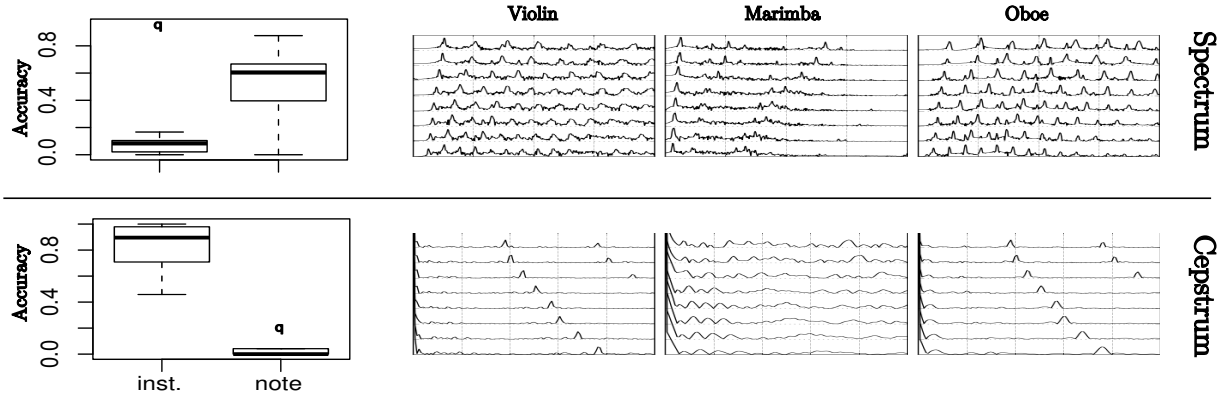


Figure 1. Comparison of spectral and cepstral analysis for eight notes from twelve instruments. **Right:** The six panels show spectral and cepstral anslysis for three of the twelve instruments. The horizontal axis is frequency for the three spectral panels, and time for the three cepstrum panels. The vertical axis for each of the eight subplots is coefficient magnitude. Spectral analysis (above) highlights the overtone series. Cepstral analysis separates pitch and timbre for pitched instruments, separation less clear for marimba. **Left:** Nearest neighbor classifiers using spectral and cepstral features to predict note and instrument labels–cepstrum predicts instruments, spectrum predicts notes.

1024, or 2048 samples, depending on the samplerate – whichever corresponded to a duration between 25 and 50 ms.

We compare two kinds of frame-level features in this work: MFCCs and Log-scaled Mel-Frequency Spectrograms (denoted LM features). MFCCs have been used extensively for music classification [1, 3, 6–8, 12, 14]. As our null model, we implemented MFCCs as in Dan Ellis' Rastamat toolbox's implementation of HTK's MFCC. [1] Our MFCC frame-level feature was 16 coefficients computed from 36 critical bands spanning the frequency range 0-4kHz using the HTK Mel-frequency mapping

$$\mathrm{Mel}(\mathrm{freq}) = 2595 \log_{10}(1 + \mathrm{freq}/700), \qquad (2)$$

using a type-II DCT, and with Hamming windowing. Here the variable freq denotes the frequency of an FFT band, and it is in units of Hz.

The definition of our LM features differs subtly from the MFCCs in ways that make the feature computation faster and the implementation simpler. The audio is scaled to the range (-1.0,1.0). No windowing is applied to the raw audio frames prior to the Fourier transform. A small constant value (10^{-4}) is added in the Fourier domain to represent a small amount of white noise and prevent division by zero. The Fourier magnitudes (not the power of each band) were projected according to a Mel-scale filterbank (the `M` in Table 2). The loudness in each critical band was approximated by the logarithm of its energy. Our experiments used from 16 to 128 critical bands (LM coefficients) to cover the frequencies from $0Hz$ to $16kHz$.

$$\mathrm{LM} = \log_{10}(|FA|M + 10^{-4}) + 4 \qquad (3)$$

$$\mathrm{MFCC} = G(\mathrm{LM}) \qquad (4)$$

Equations 3 and 4 describe the computation of the MFCC and LM features in terms of an audio matrix A that has a row for each frame in a segment, and a column for each sample in a frame. The FFT is taken over columns. The constant 10^{-4} quantifies our uncertainty in $|FA|$ and prevents taking a logarithm of zero. The subtraction of 4 from the logarithm ensures that LM features are non-negative, but take value zero when the audio is silent. The MFCCs are the discrete cosine transform G of LM.

[1] HTK: `http://htk.eng.cam.ac.uk/`

```
sr <-samplerate
nf <-number of Fourier bins
nm <-number of mel-scale coefficients

nyq = sr/2
nyq_mel = 2595 * log10(1 + nyq/700.)

# Build a Mel-Frequency Scale
# Filterbank matrix M
M = zero_matrix(rows=nf, cols=nf)
for i in 0..(nf-1):
  f     = i * nyq / nf
  f_mel = 2595 * log10(1 + f/700.)
  m_idx = f_mel/nyq_mel*nm
  j     = floor(m_idx)
  M(j+1,i) = m_idx-j
  M(j+0,i) = 1.0-m_idx+j
```

Table 1. Pseudo-code for building a Mel-Frequency Scaled filterbank in the frequency domain. This code assumes 0-based array indexing.

3. PERFORMANCE

We used three datasets to explore the value of these features in different descriptor prediction settings: tag frequency, tag presence, and genre. Segments were summarized by the mean, variance, and/or covariances of frame-level features. The summaries were classified by either a

logistic regression model, or where noted, by a Support-Vector Machine (SVM) with an RBF kernel.

In linear models, we trained binary and multi-class logistic regression to maximize the expected log-likelihood

$$-\sum_{x\in X}\sum_{l\in L} P_{true}(l|x)\log(P_{predicted}(l|x)) \qquad (5)$$

where X is the set of examples, and L the set of labels. The linear (technically affine) prediction was normalized across the classes using a Gibbs distribution.

$$P_{predicted}(l|x) = \frac{e^{Z_l f(x)+b_l}}{\sum_{k\in L} e^{Z_k f(x)+b_k}} \qquad (6)$$

where $f(x)$ is the features we extract, Z_l is the l^{th} row of the linear model, and b_l is a scalar bias for the l^{th} predictor. We fit this model by gradient descent with a small learning rate and regularize it by early stopping on held-out data. Our SVM experiments were performed using LIBSVM with the RBF kernel to implement all-pairs multiclass prediction. Held-out data was used to choose the best kernel and cost parameters (typically called γ and C in SVM literature). Features [by dimension] were normalized to have zero mean and unit variance prior to training a classifier.

3.1 Genre Prediction

Genre classification performance was estimated using the GTZAN dataset of 1000 monophonic audio clips [12], each of which is 30 seconds long. The dataset features 100 clips each of the 10 genres: blues, classical, country, disco, hiphop, pop, jazz, metal, reggae, and rock. Although this collection is relatively small, it has been used in several studies of genre classification [1, 5, 10, 13]. Classification was performed by partitioning the audio into 3-second segments and voting over the segments, as in [1]. Following standard practice, performance was measured by standard 10-fold cross-validation. For each fold, the training set was divided into a hold-out set (of 100 randomly chosen songs) and fitting set (of the remaining 800). The results of our models and some results from the literature are shown in Table 2

Our best results with both the linear and SVM classifiers were with the mean and covariance (or correlation) in LM features rather than MFCC features. Our linear model using covariance in LM features was approximately 77% accurate, while the more expensive SVM was 81% accurate. The small size of the GTZAN dataset leaves considerable variance in these performance estimates (scores are accurate to within about 4 percentage points with 95% probability). To our knowledge, the only systems to surpass our baseline linear classifier are the AdaBoost-based model of [1] and the model of [10] based on L1-regularized inference and non-negative tensor factorization. Both of these superior models are significantly more complicated and CPU intensive than our baseline. In larger datasets, the capacity of the linear model to use more data in a tractable amount of time should make its performance improve in comparison to the SVM.

Algorithm	Acc.(%)
Sparse rep. + tensor factor. [10]	92
AdaBoost + many features [1]	83
*RBF-SVM with LM (m32,r32)	81
*RBF-SVM with LM (m32,c32)	79
*Log. Reg. with LM (m32,r32)	77
*RBF-SVM with MFCC (m32,c32)	76
*Log. Reg. with LM (m32,c32)	76
*RBF-SVM with MFCC (m32,r32)	74
*Log. Reg. with MFCC (m32,r32)	72
*Log. Reg. with MFCC (m32,c32)	70
RBF+MFCC [11]	72
LDA+MFCC(m5,v5) + other [5]	71
GMM+MFCC(m5,v5) + other [13]	61

Table 2. Classification accuracies on the GTZAN dataset of our algorithms (*) and others in the literature. 'm32' is the means of 32 frame-level features, 'c32' is the upper triangle in their covariance matrix, 'r32' is the upper triangle in their correlation matrix. These scores are accurate to within about 4 percentage points with 95% probability.

3.2 Tag Frequency

We estimated our models' performance at tag frequency prediction using the MajorMiner dataset. The MajorMiner dataset consists of short audio clips labeled by players of the MajorMiner web game [7]. In this game, players listen to 10-second clips of songs and describe them with free-form tags. They score points for using tags that agree with other players' tags, but score more points for using original tags that subsequent players agree with. There are 1000 unique tags that have been verified by two players and there are a total of 13,000 such verified usages on 2600 clips. The average clip has been seen by 7 users and described with 30 tags, 5 of which have been verified. The tags describe genres, instruments, the singer (if present), and general musical or sonic qualities.

The MajorMiner dataset includes the number of times each tag was applied to each clip, which gives a graded relevance for each (tag, clip) pair. After removing clips that were tagged 'silent','end','nothing' or had a length less than eight seconds, 2578 remained. Clips are typically nine seconds in length but we used the first 8 seconds. As second-level features, we summarized each clip as a single 8-second segment. We followed [7] in using the top 25 tags (drum, guitar, male, synth, rock, electronic, pop, vocal, bass, female, dance, techno, piano, jazz, hip hop, rap, slow, beat, voice, 80s, electronica, instrumental, fast, saxophone, keyboard) which accounted for about half of the tag usages in the dataset. To ensure that each clip had a valid distribution over these tags, we added to every clip an extra virtual tag with a usage of 0.1. For most clips this usage accounted for about 1.5% of the total tag usage, but for a small number of clips with none of the top 25 tags it accounted for 100%. The clips in the dataset were sorted by their order of occurrence in the "three_columns.txt" file. The dataset was partitioned into train, validation, and test

sets by taking the 6^{th} of every 10 clips for validation, and by taking the 7^{th} and 8^{th} for testing.

We interpreted the tag usages in the MajorMiner dataset as being samples from clip-conditional multinomial distributions. We would like a function to infer that *whole distribution* by conditioning on the audio of the clip. This learning setup differs from multiple independent label prediction, for that case see the Tagatune results below.

Our results are summarized in Figure 2. Our best results were obtained by using LM features and keeping many coefficients: 128 means, 128 variances, covariance in a 32-dimensional downsampling of all 128 features. The best model (m128,v128,c32) had an average (across tag) AUC-ROC of 0.78, which is competitive with the best models from the MIREX 2009 MajorMiner evaluation. The average AUC-ROC values across the 25 tags used here for the MIREX 2009 submissions range from 0.72 (submission "HBC") to 0.79 (submission "LWW2"). [2]

There are two interesting things to note about the effect of covariance features in the MajorMiner experiments. Firstly, the covariance in MFCC features was strictly harmful in this prediction task, when used alongside the mean and variance. It has been argued that MFCC features are already relatively decorrelated compared with spectral magnitudes [9]. Perhaps when we normalize the MFCC covariance features we add features with a very low signal to noise ratio. Secondly, the LM features without covariance are poorer than a like number of MFCC features. This is similar to the simple experiment in the introduction that demonstrated the superior ability of cepstral features to generalize across pitch—that simple experiment corresponds to using the mean frame-level feature. Spectral features do not generalize as well across pitch without covariance information, but with covariance information they are better at it.

All the models were trained in just a few minutes on a desktop PC. Extracting features and training the model were fast enough that the decoding of the MP3 files was a significant part of the overall experiment time.

3.3 Tag Presence

We also tested spectral covariance features in a larger and sparser descriptor-based retrieval setting using the Magna Tagatune (Tagatune) dataset. Tagatune contains 25863 30-second audio clips [4]. Each clip is labeled with one or more binary attributes from a collection of 188 potential descriptors such as 'guitar', 'classical', 'no voices', 'world', 'mellow', 'blues', 'harpsicord', 'sitar'. These descriptors were collected from an online game in pairs of players use these words/phrases to determine whether they are listening to the same song or not. The descriptors generally refer to instrumentation, genre, and mood (see Table 4). Attributes in the dataset are binary and do not reflect the degree to which any attribute applies to a song. Furthermore, the nature of the data-collection game is such that the non-occurrence of an attribute is weak evidence that the attribute does not apply. There are many descriptors which might apply to a song, but no player thought to use them. Still, the Tagatune dataset provides a great deal of human-verified annotations and audio–despite these minor problems it is a great resource.

[2] MIREX 2009 Results:
http://www.music-ir.org/mirex/2009/index.php

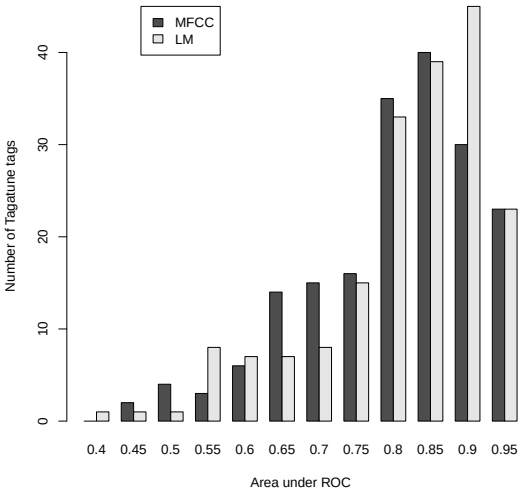


Figure 3. Histogram of the success of a linear classifier at predicting Tagatune attributes from LM (light) and MFCC (dark) means and covariance. AUC-ROC reflects both precision and recall; 0.5 is expected from a random predictor, and 1.0 from a perfect one. The MFCC features get 3 more tags 80% right, but the LM features get 16 more tags 90% right. Overall, LM features give better performance.

For the Tagatune dataset an independent logistic regression model was used to predict each potential attribute. Only logistic regression was used because the size of the dataset was prohibitive of the SVM's quadratic training algorithm. Song classification was performed by partitioning each clip into 3-second segments and voting over the segments, as in [1]. The dataset was partitioned into train, validation, and test sets by taking the 6^{th} of every 10 clips for validation, and by taking the 7^{th} and 8^{th} for testing. A plateau in validation-set accuracy (averaged across all tasks) was reached after about 20 minutes of CPU (wall) time, after about 10 passes through the training data.

The results of a linear classifier applied to the LM and MFCC mean and covariance feature are shown in Figure 3 and Table 4. Again, the LM features with covariance outperformed statistics of MFCC features. For the LM-based model, 140 of 188 descriptors were over .8 ROC and 68 of those are over .9. The MIREX 2009 Tagatune evaluation used a test protocol almost identical to ours, and found that the participants' average ROC scores ranged from 0.67 to 0.83. Our simple model appears actually to score slightly better than the MIREX 2009 participants did.

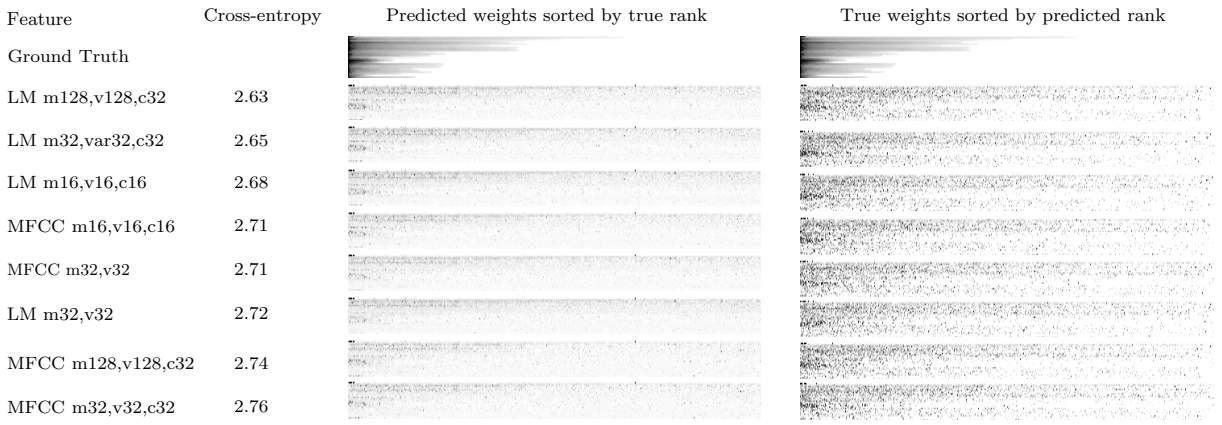


Figure 2. LM features outperform MFCC features for predicting tag distributions in the MajorMiner dataset. 'm32' (similarly 'm16','m128') denotes the mean in 32 coefficients, 'v32' the variance, and 'c32' the covariance (the upper triangle of the 32x32 matrix). When covariance information is used, LM features outperform MFCCs. Images in the middle column illustrate the recall of each model for each of the 25 most popular tags: the greater the density toward the left, the higher the recall of the model. Images in the right column illustrate the corresponding precision. The best model has AUC-ROC of 0.78.

4. DISCUSSION

Our covariance and correlation features help to summarize (in a time-independent way) the different kinds of timbres observed during a segment. The covariance (or correlation) in LM frame-level features summarizes the way that energy/loudness in different frequency bands co-occur in a signal. Recall that the timbre of a sustained instrument note can be crudely approximated by the shape of the spectral envelope. The difficulty in recognizing instruments in a segment after it has been summarized by the mean or variance is that when different instruments are present in one segment, the mean envelope can be indistinguishable from the mean that would come from other instrument combinations. The covariance is an improvement in this respect. If different instruments in a segment do not play simultaneously, then the covariance will encode activity corresponding to the (pairs of) peaks in the envelope of each instrument. To the extent that these instruments have timbres with different pairs of peaks, the instruments will not interfere with one another in the segment-level feature. Still, when instruments are played simultaneously (as often happens!) there is more interference.

4.1 MAP vs. ROC

Table 4 lists the mean average precision (MAP) and classification error rate (ERR) for some of the most popular and least popular attributes. The classification accuracies for most descriptors is quite close to the baseline of random guessing. That is because most descriptors are so rare that when a trained model is forced to take a hard classification decision, it is very difficult for the evidence from the song feature to outweigh the overwhelming prior that a rare descriptor does not apply. A similar finding is described in [2]. Since the attributes are rare, even the best models rank many negative-labeled examples also near the beginning of the clip collection, so precision and classification error are low. But the high rate of false-negative labels (see Table 3) biases the MAP and ERR measures more than the ROC. In the case of MagnaTagatune, where there are many false-negative labels (true instances, labeled as non-instances) MAP and ERR criteria are potentially very biased estimators of model performance. The ROC measure a more appropriate criterion in this context.

Attr	Count
female vocals	386
male vocals	465
female vocal	644
no vocal	995
male vocal	1002
no vocals	1158
vocals	1184
vocal	1729

Table 3. The number of applications of several Tagatune attributes over all 25863 clips. One of the difficulties with Tagatune is the frequency of false negative attributes in the dataset. For example, although vocal was applied 1729 times, the 'no vocal' attribute was applied only 995 times; 90% of the clips are labeled as neither 'vocal' nor 'no vocal'.

5. CONCLUSION

The covariance of log-magnitude Mel frequency scale *spectral* coefficients (LM features) offer a superior alternative to statistics of MFCCs when summarizing frame-level audio features for genre and tag prediction. We have demonstrated the advantage of our LM features on three standard genre and tag prediction datasets: GTZAN, MajorMiner, and Tagatune. Furthermore, these features make state of the art performance available with just a linear classifier (such as L1- or L2-regularized logistic regression, or lin-

Attr	Freq	ERR	MAP	ROC
guitar	0.187	0.159	0.62	0.85
classical	0.169	0.157	0.63	0.91
slow	0.135	0.135	0.32	0.77
techno	0.110	0.090	0.58	0.92
strings	0.110	0.110	0.44	0.87
drums	0.102	0.102	0.34	0.84
electronic	0.096	0.097	0.33	0.84
rock	0.093	0.057	0.71	0.96
fast	0.093	0.093	0.31	0.79
piano	0.077	0.074	0.54	0.84
repetitive	0.001	0.001	0.15	0.77
scary	0.001	0.001	0.02	0.97
woodwind	0.001	0.001	0.01	0.82
viola	0.001	0.001	0.08	0.95
quick	0.001	0.001	0.00	0.56
soprano	0.001	0.001	0.17	0.97
horns	0.001	0.001	0.00	0.68
soft rock	0.001	0.001	0.00	0.70
monks	0.001	0.001	0.31	0.99
clasical	0.001	0.001	0.00	0.83
happy	0.001	0.001	0.00	0.61

Table 4. Test set performance of best model on the 10 most popular (above) and least popular (below) descriptors in Tagatune. **Freq** is application frequency, **ERR** is classification error, **MAP** is mean average precision, and **ROC** is the area under the ROC.

ear SVM). Our results on the GTZAN dataset suggest that RBF SVMs may offer slightly better performance when time allows for training.

The LM features are straightforward to implement, computationally cheap, and the use of a linear classifier makes our model viable on any size of genre or tag-prediction dataset. We believe that the model presented here has great potential for working with industrial-scale audio datasets.

Finally, the results presented here demonstrate that the discrete cosine transform (or inverse Fourier transform) responsible for the cepstrum's deconvolution property actually hinders performance in some circumstances. One explanation of this behavior is that our model learns a better deconvolution-like transform of the spectral data than is provided by the cepstrum. We admit that this is only one possible explanation of these results and that further analysis is necessary in order to make conclusions. We believe that this is one fruitful direction for future research.

6. REFERENCES

[1] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kegl. Aggregate features and AdaBoost for music classification. *Machine Learning*, 65:473–484, 2006.

[2] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger: A model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, 37(2):115–135, 2008.

[3] Thibault Langlois and Gonalo Marques. A music classification method based on timbral features. In *ISMIR*, Kobe, Japan, October 2009.

[4] Edith L. M. Law and Luis von Ahn. Input-agreement: A new mechanism for data collection using human computation games. In *CHI*, pages 1197–1206, 2009.

[5] Tao Li and George Tzanetakis. Factors in automatic musical genre classification. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, 2003.

[6] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *ISMIR*, Plymouth, Mass., October 2000.

[7] M. Mandel and D. Ellis. A web-based game for collecting music metadata. *J. New Music Research*, 37(2):151–165, 2008.

[8] Michael I. Mandel and Daniel P.W. Ellis. Song-level features and support vector machines for music classification. In *ISMIR*, pages 594–599, London, UK, September 2005.

[9] P. Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, pages 374–388, 1976.

[10] Yannis Panagakis, Constantine Kotropoulos, and Gonzalo R. Arce. Music genre classification using locality preserving non-negative tensor factorization and sparse representations. In *ISMIR*, Kobe, Japan, October 2009.

[11] Douglas Turnbull and Charles Elkan. Fast recognition of musical genres using RBF networks. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):1–5, April 2005.

[12] G. Tzanetakis, G. Essl, and P. Cook. Automatic musical genre classification of audio signals. In *ISMIR*, Bloomington, Indiana, October 2001.

[13] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002.

[14] Kris West and Stephen Cox. Finding an optimal segmentation for audio genre classification. In *ISMIR*, London, UK, October 2005.

Similarity Measures for Chinese Pop Music Based on Low-Level Audio Signal Attributes

Chun-Man Mak; Tan Lee; Suman Senapati; Yu-Ting Yeung; Wang-Kong Lam
Department of Electronic Engineering
The Chinese University of Hong Kong
{cmmak, tanlee, ssuman, ytyeung, wklam}@ee.cuhk.edu.hk

ABSTRACT

In this article a method of computing similarity of two Chinese pop songs is presented. It is based on five attributes extracted from the audio signal. They include music instrument, singing voice style, singer gender, tempo, and degree of noisiness. We compare the computed similarity measures with similarity scores obtained with subjective listening by over 200 human subjects. The results show that rhythm and mood related attributes like tempo and degree of noisiness are most correlated to human perception of Chinese pop songs. Instrument and singing style are relatively less relevant. The results of subjective evaluation also indicate that the proposed method of similarity computation is fairly correlated with human perception.

1. INTRODUCTION

With the rapidly increasing popularity of digital music and related technologies, thousands of new songs are made available over the Internet everyday. The convenience of low-cost digital storage also promotes the increase in personal collection of music files. However, a user may find it more difficult to look for a song that he or she likes to listen. This calls for music recommendation systems to sort and find music efficiently.

A recommendation system aims to identify songs that match a user's taste and recommend these songs to the user. There are two types of music recommendation systems: Content-based and metadata-based recommendation. A content-based system mainly exploits audio features extracted from the music file itself to make recommendation [2], [6]. Metadata-based systems, like the *Last.fm* music network [11], make use of textual metadata associated with music documents, such as the artist's name, the song title, or the album name. These systems are often combined with collaborative filtering techniques to capture users' preferences. Lyrics of a song can also be used [1]. There are also hybrid systems that combine audio content and metadata for recommendation, [3],[4],[10].

In content-based recommendation system, the recommended songs are those that sound similar to the existing songs that the user likes to listen to. Recommendation becomes a task of matching in this case. A kind of similarity between a query song and a set of candidate songs needs to be computed. The candidates with the highest similarity measure will be returned as the results of recommendation. The computation of an effective similarity measure is therefore a crucial step in many recommendation systems.

Similarity of two songs can be computed from the contents or textual metadata embedded in the songs. In this article, we focus on studying content-based music similarity computed from songs in a specific genre, namely, Chinese pop songs. Most recommendation systems previously proposed were developed in a cross-genre environment [2],[10]. The song database contains a large number of songs of different genres, and genre classification is performed to put songs into the right group. Recommendation of songs within the same genre also has important applications. For example, a pop music producer may wish to promote his/her new productions by matching the taste of potential listeners. It is necessary to understand the aspects that humans consider when they decide if two songs are similar in intra-genre case. In addition, to the best of authors' knowledge, there does not exist any content-based recommendation system for Chinese pop songs. Our investigation results can provide valuable information in building such system in the future.

To facilitate the computation of a similarity measure, the important attributes of a song must be represented numerically. There are many attributes that can be used to represent a song, such as melody structure and chord. Such features, however, are difficult to extract accurately. Therefore, in the proposed method, we use a set of low-level audio descriptors, i.e., instrument identity, singing style, gender of the singer, tempo, and degree of noisiness (DoN) to represent the songs. The usefulness of these features is verified via subjective evaluation.

The paper is organized as follows. Section 2 describes the audio attributes that are used to represent Chinese pop songs, and the method of computing similarity from these attributes. Section 3 explains the process of subjective evaluation. Section 4 gives the experimental results, and the correlation between the proposed similarity measure and the findings of subjective evaluation. Conclusions are drawn in section 5.

2. MUSIC ATTRIBUTES

Prior to signal analysis, all audio files are down-sampled to 16 KHz. Most digital music are recorded at sampling rate above 16 KHz. By unifying to this common frequency, we try to minimize the discrepancies of signal quality among the many possible data sources. In the case of stereo recording, the two tracks are averaged to obtain a monaural wave signal. In short, all signal analysis operations are performed based on monaural wave signal at 16 kHz sampling rate.

2.1 Vocal / Non-vocal Segmentation

A typical Chinese pop song is about 3 to 4 minutes in duration and is composed of four parts. It starts with an instrument intro, followed by vocal verse, and instrument interlude, and finally the chorus. The vocal part is sung in the Cantonese dialect or Mandarin. Assuming this song structure, we divided each song into the vocal part (with human singing voice) and the non-vocal part (instruments only) in our analysis. The extraction of instrument attribute is done on the non-vocal part, while the attributes of singing style and gender are extracted from the vocal part. Tempo and DoN are estimated from the whole song. This is illustrated in Figure 1.

The wave signal is divided into short-time frames of 100ms long with frame shift of 50ms. Short-time Fourier Transform (STFT) of 2048 points is applied to each frame and 13 Mel-frequency cepstral coefficients (MFCC) are computed. A statistical classifier is built for vocal / non-vocal segmentation. About 500 songs are manually segmented into vocal and non-vocal parts. These songs, along with other songs used in our experiments, are all Chinese pop songs purchased from CD stores. These segments are used as the training data for the vocal / non-vocal classifier. The minimum segment length was set to be 1 second. A vocal class and a non-vocal class are modeled by two Gaussian mixture models (GMM), each with 64 mixtures.

The segmentation is performed by dividing the audio signals into non-overlapping segments of 1 second long, i.e., 20 frames, and classifying these segments as either vocal or non-vocal. For each segment, the log-likelihood with respect to the vocal class is computed as

$$L_{vocal} = \sum_{n=1}^{20} \log\left(Prob\left(\boldsymbol{\Theta}_n \middle| \mathbf{w}_{vocal}, \boldsymbol{\mu}_{vocal}, \boldsymbol{\sigma}_{vocal}\right)\right) \qquad (1)$$

where $\boldsymbol{\Theta}_n$ is the MFCC feature vector of the n^{th} frame in the segment, and $\mathbf{w}_{vocal}, \boldsymbol{\mu}_{vocal}, \boldsymbol{\sigma}_{vocal}$ are respectively the mixture weights, means, and covariance matrices of the vocal GMM. The log-likelihood with respect to the non-vocal class, denoted by $L_{non\text{-}vocal}$, is computed in the same way. The segment is classified to the class with the higher log-likelihood. This method of statistical classification is also used in the extraction of instrument, singing style, and gender attributes. The classification accuracy of the vocal / non-vocal classifier is 90%. The test data used in our experiments are not included in the training set and this applies to all other classification accuracy reported in this article. Readers may also notice that the numbers of mixtures used vary for different classifiers described in latter sections. These numbers are determined empirically to achieve the best trade-off between computation time and accuracy.

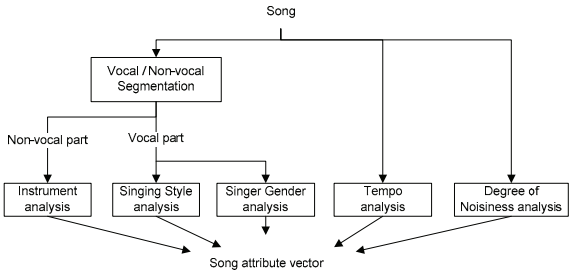


Figure 1. Segments of songs

2.2 Extraction of Music Attributes

Five attributes are used to describe a Chinese pop song: instrument, singing style, gender of singer, tempo, and degree of noisiness. Instrument, singing style, and gender are computed based on MFCC features and the statistical classification technique.

2.2.1 Instrument

For simplicity, we assume that only a single instrument is present or dominant at a particular time instant. Eight instruments commonly used in Chinese pop songs are modeled: reedy, electronic-guitar, piano, strings, synthesizer, guitar, flute, and percussion. Each instrument is represented by a GMM of 256 mixtures. The training data includes the 500 manually annotated songs and part of the RWC database [7]. Following an approach similar to vocal/non-vocal segmentation, each non-vocal segment in a song is assigned to an instrument class. Then the instrument attribute is represented by a vector with eight elements, i.e.,

$$\mathbf{F}_{inst} = \left\{I_1, I_2, ..., I_8\right\} \qquad (2)$$

where I_i is the percentage of segments in the song that are assigned to the i^{th} instrument class. In our experiments, the classification accuracy was 72%.

2.2.2 Singing Style

It is not trivial to describe the singing style in a Chinese pop song. In our study, singing style is defined with reference to a few famous singers of Chinese pop songs. For a given song, we try to measure the degree of similarity between the singing voice in the song and the voice of each reference singer. Among the 500 training songs, we choose 6 male and 6 female singers with distinct voices and styles. Each singer has about 30 training songs. A separate class is established for children's voice. As a result, we have a total of 13 singing style classes. Each class is represented by a GMM of 64 mixtures. The singing style attribute, $\mathbf{F}_{sing}$, is defined as

$$\mathbf{F}_{\text{sing}} \in \{S_1, S_2, ..., S_{13}\} \quad (3)$$

where S_i is the percentage of segments in the song that are assigned to the i^{th} class. In our experiments, the classification accuracy of singing style is 82%.

2.2.3 Gender

An explicit gender classifier is built. A male voice and a female voice model are trained to be 128–mixture GMM. The gender attribute, $\mathbf{F}_{gend}$, is defined as:

$$\mathbf{F}_{gend} \in \{G_{male}, G_{female}\} \quad (4)$$

where G_{male} and G_{female} are the percentage of male and female voice segments in the song, respectively. The song-level gender classifier has an accuracy of 97%.

2.2.4 Tempo

The tempo detection method proposed in [9] is used to generate the tempo information we need. The tempo of a given song is estimated by the Fourier analysis of the beat onset pattern. Complex domain spectral difference is used as the detection function for the beat onset. To find the spectral difference, we first estimate the instantaneous spectral change $\eta(m)$ at the m^{th} short-time frame, which is defined as

$$\eta(m) = \sum_k \left| \hat{Y}_m(k) - Y_m(k) \right| \quad (5)$$

where $Y_m(k)$ is the spectral value (DFT coefficient) at frame m and frequency bin k, and $\hat{Y}_m(k)$ is the corresponding value predicted from the immediately preceding frame, i.e.

$$\hat{Y}_m(k) = \left| Y_{m-1}(k) \right| e^{j\hat{\Phi}_m(k)}. \quad (6)$$

The expected phase $\hat{\Phi}_m(k)$ is predicted from phase in previous two frames as follows:

$$\hat{\Phi}_m(k) = \varphi_{m-1}(k) + (\varphi_{m-1}(k) - \varphi_{m-2}(k)) \quad (7)$$

where $\varphi_{m-1}(k)$ and $\varphi_{m-2}(k)$ are the observed phases for frame m-1 and m-2 respectively. Frame size of 100ms and frame shift of 10ms are used. The frame shift is much shorter than the 50ms shift as used in the extraction of other attributes because of the need to detect fast tempo. To obtain a beat onset pattern with clear and well-defined peaks, $\eta(m)$ is subtracted by the moving average threshold $\overline{\eta}(m)$ with window size $W = 10$, i.e.

$$\overline{\eta}(m) = \frac{1}{W} \sum_{i=m-\frac{W}{2}}^{m+\frac{W}{2}} \eta(i). \quad (8)$$

Half-wave rectification is then performed on the difference to obtain $\hat{\eta}(m)$, the beat onset value of frame m, i.e.

$$\hat{\eta}(m) = HWR(\eta(m) - \overline{\eta}(m)) \quad (9)$$

where $HWR(x) = (x + |x|)/2$.

To handle tempo variations over the entire duration of a song, we divide a song into segments of 12s long, with time shift of 4s. Each of these segments contains 1200 frames. Fourier analysis is performed on beat onset pattern of each segment, and the frequency axis is mapped into tempo values. The analysis result is represented by a tempogram, which is a two-dimensional time-tempo representation of the strength of tempo values in local segments. An example of tempogram is shown in Figure 2. The tempo value is limited to the range of 30 beat per minute (bpm) to 300 bpm, which covers most of the pop music. The tempo value with the strongest impulse is picked as the local tempo value for each segment. The local tempo values of all segments in the song form a distribution, from which the tempo attribute is derived as

$$\mathbf{F}_{tempo} = \{T_1, T_2, T_3\} \quad (10)$$

where T_1, T_2, and T_3 corresponds to the 25%, 50%, and 75% percentile tempo values.

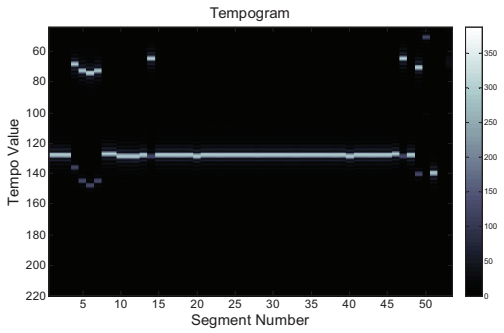


Figure 2. An illustration of the tempogram

2.2.5 Degree of Noisiness

The mood of a song is an important factor to be considered when computing similarity of songs. The energy of a song is in some way related to the mood of the song. According to Thayer's emotion model [8], songs with low intensity are usually associated with calm, relax, or de-

pressed emotions, while songs with high intensity, are associated with exciting or angry emotion. This is illustrated in Figure 3. In addition, songs with thick texture, i.e., many instruments and voices playing simultaneously, are generally associated with more intense and excited feeling. On the other hand, songs with thin texture, i.e., a few instruments or voices, are commonly found in calm and relax music. Audio signals in thick-texture music are likely to have flatter spectra, compared to those in the thin-texture music. Thus we propose to use both the signal intensity and spectral flatness features to compute a "degree of noisiness" (DoN) attribute, which is related to the mood of the song.

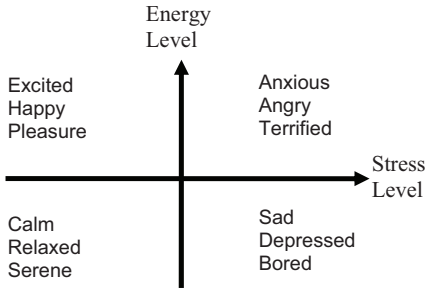


Figure 3. Thayer's model of mood [8]

Three values are defined as the DoN label: 0. 0.5, and 1, which correspond to "*low*", "*medium*", and "*high*" respectively. A low DoN means that the song is perceived as quiet and calm, while a high DoN means loud and noisy. We found that in many pop songs, the DoN level varies noticeably between the first and the second half of the song. Therefore, two DoN labels are used to describe a song, one for the first half of the song and one for the second half.

Prior to the DoN analysis, the audio signal is normalized by its maximum amplitude. Since the beginning and ending of the songs usually contains silence, the first 10% and the last 10% of the signal in each song are discarded. The remaining signal is then divided into two halves as described above. For each frame of 100 ms long, the signal intensity is computed by

$$P(m) = 10\log_{10}\left(\frac{1}{T}\sum_{t=0}^{T-1} X_m^2(t)\right) \tag{11}$$

where $P(m)$ is the power of m^{th} frame, X_m is the signal of m^{th} frame in time domain, and T is equal to 1600 for 16KHz sampling and 100ms frame size. From the frame-level signal intensity values, we compute the mean and standard deviation of the whole segment (half of the song), which are denoted by M_{power} and SD_{power}, respectively. Similarly, we compute the mean and standard deviation of the spectral flatness over each segment, which are denoted by M_{SF} and SD_{SF}, respectively. The spectral flatness, $SF(m)$, is defined by

$$SF(m) = \frac{\sqrt[K]{\prod_{k=1}^{K}|F_m(k)|}}{\frac{1}{K}\sum_{k=1}^{K}|F_m(k)|} \tag{12}$$

where $|F_m(k)|$ is the magnitude spectrum of X_m computed by 2048-point DFT, K is equal to 1023. The DC term (k=0) is ignored in the computation.

Each DoN class is represented by a single-mixture Gaussian model, which models the four-element feature vector $\{M_{power}, SD_{power}, M_{SF}, \text{and } SD_{SF}\}$. The DoN attribute vector of a song is defined as:

$$\mathbf{F}_{DoN} = \{N_1, N_2\} \tag{13}$$

where N_1 and N_2 corresponds to the label in the first and second half of the song, respectively.

2.3 Computation of Similarity Score

Let A and B denote two songs. The overall similarity between A and B is the weighted sum of the similarities computed for the five audio attributes. The similarity value of each attribute has the range of 0 (most dissimilar) to 1 (identical). The superscript in the attribute vectors and elements denotes the song from which the attribute is computed. For instrument and singing style attributes, the similarities $s_{inst}(A,B)$ and $s_{sing}(A,B)$ are computed by cosine similarity, i.e.

$$s_{inst}(A,B) = \frac{\mathbf{F}_{inst}^A \bullet \mathbf{F}_{inst}^B}{\left\|\mathbf{F}_{inst}^A\right\|\left\|\mathbf{F}_{inst}^B\right\|} \tag{14}$$

and

$$s_{sing}(A,B) = \frac{\mathbf{F}_{\text{sing}}^A \bullet \mathbf{F}_{\text{sing}}^B}{\left\|\mathbf{F}_{\text{sing}}^A\right\|\left\|\mathbf{F}_{\text{sing}}^B\right\|}. \tag{15}$$

For gender, the similarity, $s_{gend}(A,B)$, is defined as:

$$s_{gend}(A,B) = \frac{\min\left(G_{male}^A, G_{male}^B\right)}{\max\left(G_{male}^A, G_{male}^B\right)} \tag{16}$$

Tempo similarity, $s_{tempo}(A,B)$, is computed in a similar way as the gender, except that we take the average over the three components in the tempo attribute vector, i.e.,

$$s_{tempo}(A,B) = \frac{1}{3}\sum_{i=1}^{3}\frac{\min\left(T_i^A, T_i^B\right)}{\max\left(T_i^A, T_i^B\right)} \tag{17}$$

For the discrete class labels in the DoN attribute vectors, a normalized form of Euclidean distance is used for $s_{DoN}(A,B)$, which is defined as,

$$s_{DoN}(A,B) = 1 - \frac{\left\|\mathbf{F}_{DoN}^A - \mathbf{F}_{DoN}^B\right\|}{\sqrt{2}} \tag{18}$$

The overall similarity score is the weighted sum of the similarities by all attributes,

$$s(A,B) = \sum_{x \in music\ attribute} w_x s_x(A,B) \qquad (19)$$

where w_x denotes the weight for attribute x. The weights are determined empirically based on the observations from subjective evaluation.

3. SUBJECTIVE EVALUATION

Subjective evaluation is carried out to obtain a set of reference similarity scores that can be considered as the ground truth. These similarity scores will be used for two purposes: (a) to analyze which attributes are more important to human listeners when comparing two songs, and (b) to determine the optimal weights in the proposed objective similarity computation.

A total of 215 subjects participated in the listening tests. 43 Chinese pop songs, each with duration of about 4 minutes, were selected as the test materials. These songs were not included in the training set of 500 songs. They are chosen in a way that within this limited number of test songs, the varieties of instruments, singers, tempo, and mood are maximized. For the test song Q, 10 candidate songs were manually selected from the remaining songs in the test set so that we have various degrees of differences in the attributes between the candidates and the query songs. Five subjects were asked to listen to Q and the 10 candidate songs, denoted as C_i, i={1,2,…,10}. The subject was asked to rate the similarity between Q and C_i, on the scoring scale as shown in Figure 4.

A test session was divided into two parts, in the first part, a subject was asked to first listen to song Q, and 5 of the candidate songs. The subject was allowed to repetitively listen to Q if he/she liked to. There was a short break after the first part. Then the second part starts by listening to Q again, and then the remaining 5 candidates. The orders of playing the candidate songs are different from one listener to another.

4. EXPERIMENTAL RESULTS

We first investigated on the importance of each music attribute in the proposed similarity measure. This is done by assigning the weight of one attribute to be 1 and the others to be zero. For the query song Q in the set of subjective test songs, we compute the objective similarity score of Q and each of the 10 candidates using (19) and obtain a similarity vector $\mathbf{S}_Q$ = {$s(Q,C_1)$, $s(Q,C_2)$, … , $s(Q,C_{10})$}. From the results of subjective evaluation, we compute the average subjective similarity scores from all subjects that were tested with this set of songs. The resulted scores form the subjective similarity vector denoted as $\mathbf{E}_Q$={$e(Q,C_1)$, $e(Q,C_2)$, … , $e(Q,C_{10})$}. The Spearman's rank correlation between $\mathbf{S}_Q$ and $\mathbf{E}_Q$, denoted

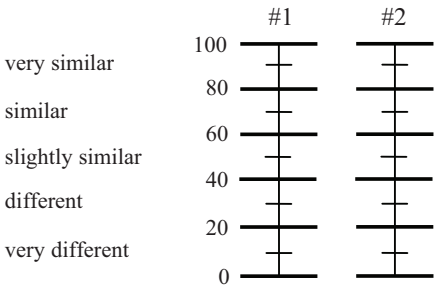


Figure 4. Scoring scale

by ρ_Q, is computed for each query song. Subsequently, the overall average of correlation is computed over the 43 songs. The results are shown in Table 1. It is observed that the importance of the attributes in human listening is highly uneven. Instrument and gender are the least relevant attributes. The correlation values are close to zero, indicating that subjective scores are almost uncorrelated to the objective similarity. Singing style and tempo are more important, with a small but positive correlation. DoN is the most important attribute with the highest correlation between objective similarity and subjective scores.

Attribute	Average Correlation
Instrument	0.07
Singing style	0.20
Gender	0.06
Tempo	0.24
Degree of Noisiness	0.49

Table 1. Correlation of each attribute to subjective scores

Next we try to determine a set of optimal weights for similarity computation in (19) by maximizing the correlation to subjective listening. Exhaustive search is performed. The attribute weights are varied from 0 to 10 with an increment step size of 1. It was found that the optimal set of weights are 1, 2, 0, 3, and 4 for instrument, singing style, gender, tempo, and DoN, respectively. The average correlation achieved with this set of weights is equal to 0.544. Although it is not a very high correlation, the result shows that these audio attributes are still useful in modeling subjective similarity judgment of Chinese pop songs. The weights are in agreement with our observations on the study of single attribute.

Figure 5 shows the distribution of the correlation values for all test songs when the optimal weights are applied. Among the 43 songs, 5 songs have negative values of correlation, and 27 have correlations higher than 0.6. This indicates that the proposed objective similarity measure can fairly model the subjective similarity for most of the songs.

As part of the subjective test, each human subject was asked to fill in a survey questionnaire. The subject had to

rank from 1 (most important) to 5 (least important) the following attributes when he/she considers the similarity of a pair of songs: rhythm, accompaniment instrument, singing style, lyrics, and languages. The average ranks of these attributes are tabulated in Table 2. The results match with what we found from the study of correlation between objective and subjective scores. Tempo and DoN, which somehow affect the perception of the rhythm, are the most important factors. Instrument and singing style are more or less the same in importance. Lyrics and languages (Cantonese or Mandarin) are the least important factors. At the current stage, we only use attributes like tempo and DoN to model the rhythm. More complex attributes such as melody and chord information should give us a better model for the rhythm and obtain an objective similarity metric that better correlates with subjective scores. One interesting note is that some subject suggests that attributes like atmosphere of the song, style of songs in different generations, and harmony of the songs may also be important. However these attributes are more abstract and difficult to extract from low-level audio features. Metadata such as publication year, genre, etc. may be used in future system to better describe these attributes.

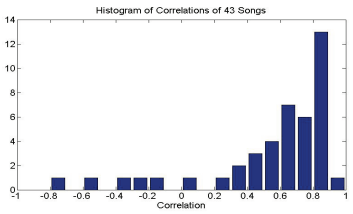


Figure 5. Histogram of correlations of 43 songs

Attributes	Average rank
Rhythm	1.3
Instrument	2.7
Singing Style	2.9
Lyrics	4.1
Language	4.4

Table 2. Average rank of attributes from survey

5. CONCLUSIONS

In this article we proposed five audio signal attributes that can be used to generate an attribute vector for a song in the Chinese pop song genre. The attribute vectors can then be used to compute similarity between songs, which is a fundamental process in content-based music recommendation system. We found that among these attributes, tempo and degree of noisiness play the most important role in approximating the subjective scores, followed by singing style and instrument. The results also indicate that rhythmic and mood information is crucial in objective similarity computation.

6. ACKNOWLEDGEMENTS

This work was supported by the Innovation and Technology Commission of the HKSAR Government.

7. REFERENCES

[1] Y. Xia, L. Wang, and K.-F. Wong: “Sentiment Vector Space Model for Lyric-Based Song Sentiment Classification,” International Journal of Computer Processing Of Languages, Vol. 21, No. 4, pp. 309-330, Dec. 2008.

[2] X. Zhu, Y.-Y. Shi, H.-G. Kim, K.-W. Eom: "An integrated music recommendation system," IEEE Transactions on Consumer Electronics,Vol.52, No.3, pp.917-925, Aug. 2006.

[3] T. Yoon, S. Lee, K.H. Yoon, D. Kim, J.-H. Lee: "A personalized music recommendation system with a time-weighted clustering," Proceedings of the 4th International IEEE Conference Intelligent Systems, Vol.2, pp.10-48-10-52, 6-8 Sept. 2008.

[4] K. Yoshii, M. Goto, K. Komatani, T. Ogata, H.G. Okuno: "An Efficient Hybrid Music Recommender System Using an Incrementally Trainable Probabilistic Generative Model," IEEE Transactions on Audio, Speech, and Language Processing, Vol.16, No.2, pp.435-447, Feb. 2008.

[5] W. Cohen and W. Fan: “Web-collaborative filtering: Recommending music by crawling the web,” Computer Networks, Vol. 33, No. 1–6, pp. 685–698, 2000.

[6] JJ Aucouturier, F Pachet, "Music similarity measures: What's the use," Proceedings of the ISMIR, 2002.

[7] RWC database: available at: http://staff.aist.go.jp/m.goto/RWC-MDB/

[8] R.E. Thayer: *The Biopsychology of Mood and Arousal*, Oxford University Press, 1989.

[9] P. Grosche and M. Muller: “A Mid-level Representation for Capturing Dominant Tempo and Pulse Information in Music Recordings,” Proceedings of the International Society for Music Information Retrieval Conference (ISMIR 2009), pp. 189-194, 2009.

[10] A. Berenzweig, B. Logan, D.P.W. Ellis, and B. Whitman: "A Large-Scale Evaluation of Acoustic and Subjective Music-Similarity Measures," Computer Music Journal, Vol. 28, No. 2, pp. 63-76, 2004.

[11] *Last.fm* music radio website: http://www.last.fm/home

SINGING / RAP CLASSIFICATION OF ISOLATED VOCAL TRACKS

Daniel Gärtner
Fraunhofer Institute for Digital Media Technology IDMT
daniel.gaertner@idmt.fraunhofer.de

ABSTRACT

In this paper, a system for the classification of the vocal characteristics in HipHop / R&B music is presented. Isolated vocal track segments, taken from acapella versions of commercial recordings, are classified into classes singing and rap. A feature-set motivated by work from song / speech classification, speech emotion recognition, and from differences that humans perceive and utilize, is presented. An SVM is used as classifier, accuracies of about 90% are achieved. In addition, the features are analyzed according to their contribution, using the IRMFSP feature selection algorithm. In another experiment, it is shown that the features are robust against utterance-specific characteristics.

1. INTRODUCTION

According to the IFPI Digital Music Report 2010 [11] , the catalogue of digital music from the licensed music services contained more then 11 million tracks in 2009. For some years, researchers have been working on tools that simplify the handling of this large amount of data. Automatic content-based analysis is now part of a multitude of different applications. The algorithms help people to visualize their music collections and generate playlists. Music lovers can discover new music with the help of music recommendation engines. DJs use software for automatic tempo and beat detection.

This work is about automatically labeling short snippets of isolated vocal tracks according to their vocal characteristics. The segments are classified into two classes, rap and singing. These two classes are the dominant vocal styles in HipHop and contemporary R&B music. A successful labeling could further be useful in urban sub-genre classification, serve as a basis for vocal characteristics song segmentation, and help analyzing the song structure. Also, intelligent audio players could be designed, that automatically skip all sung or all rapped parts in R&B and HipHop music songs, depending on the preferences of their users.

Rap is a form of rhythmically speaking, typically to accompaniment music. As pointed out in [7], singing contains a larger percentage of voiced sounds than speaking. For Western music, the singing voice also covers a wider range of fundamental frequencies. In addition, singing tends to have a much wider dynamic range in terms of amplitude. According to [8], singing voice tends to be piecewise constant with abrupt changes of pitch in between. In natural speech, the pitch frequencies slowly drift down with smooth pitch change in an utterance. This peculiarity can also often be observed in rap passages. While rapping, artists are quite free in their choice of the pitches, while the fundamental frequencies in singing are usually related to the harmonic or melodic structure of the accompaniment.

In a survey conducted in [5], subjects had to label vocal utterances with a value from 1 (speaking) to 5 (singing), and explain their decision. For one utterance, 5 subjects used the word ”rap” in their explanation. The mean score of this utterance was 3.74. Rap seems to be perceived somewhere in between singing and speaking, in that special case even a bit more singing than speaking. Different subjects mentioned melody, rhythm, or rhyming combined with musical scales as features to discriminate singing from speaking. However, rhythm descriptions might be less important for rap / singing classification, since rap and singing are both rhythmically while speech is not. Further, repetitions, the clarity and sharpness of pitch, or the presence of vibrato have been identified to be present in singing rather than speaking. Another feature for the discrimination of speech and song as denoted in [9] is stress. It is stated that in English language speech, stress affects the meaning of the utterance. This is another one of the points where speech and rap differ. In rap, where the voice is used as instrument, accentuation often is part of the rhythm.

In previous work [4] the classification into singing and rap has been investigated on full songs (vocals + accompaniment), using common low-level features and a Gaussian mixture model based classifier. One of the outcomes of this work has been, that, although the classifier produced reasonable results, the classification was highly influenced by the accompaniment music. We therefore suggest to build the system composed of two major components: vocal track isolation and the classification of isolated tracks, using a feature set designed towards this task. This paper focuses on the second objective.

To the knowledge of the authors, automatic content-based discrimination of isolated singing and rap tracks has not yet been investigated elsewhere. However, research has been carried out on the task of singing and speaking classification. Investigations on the rap voice in a musicology context have been carried out though, e.g., [6].

In [5], a set of features is presented to discriminate between singing and speaking including statistics over F0 and ΔF0, vibrato detection, repetition detection, and the proportions of voiced frames, unvoiced frames, and silent frames, and repetition features.

Another system is presented in [19]. Based on features like the rate of voiced sounds, the standard deviation of the duration of voiced sounds, and the mean silence durations, an SVM is trained for singing / speaking classification. This classifier is used to forward sung queries to a query by humming system, and spoken queries to a speech recognition system.

[12] present another study on speaking / singing discrimination. The first part addresses human performance. They find that already 1 second of audio signal is enough to classify with an accuracy of 99.7% accuracy. Still 70% are reported on signals of 200 ms length. Further, it is investigated, that the performance drops, when either spectral or prosodic information is distorted in the audio signal. In the second part, the task is performed using Mel frequency cepstral coefficients (MFCCs), ΔMFCCs, and ΔF0 as features and a maximum likelihood classifier based on Gaussian mixture models (GMM).

Another field working with energy-based and pitch-features on vocal signals is speech emotion recognition (e.g., [17, 18]).

The remainder of this paper is organized as follows. In Section 2, the features and classifier are described. Section 3 deals with the experiments that have been conducted. There, also the used data and the general experimental setup is introduced. The results and their meaning are discussed in Section 4. Finally, the conclusions and an outlook are given in Section 5.

2. APPROACH

In this section, the used features and the classifier that has been utilized, are explained.

2.1 Features

The features contain the information about the audio signal, that is accessed by the classifier. Therefore, it is important, that the features are well designed with respect to the task.

Some of the features are calculated from the pitch of the vocal segment. YIN [3] has been used as F0-estimator. In addition to an F0-estimation in octaves over time, YIN's output also includes the instantaneous power (IP) and the ratio of aperiodic power to the total power (ATR).

All F0-estimations are transformed in the relative pitch representation (RPR), which is a mapping into an interval of one octave width around the dominant frequency. First, a histogram with 100 bins is calculated over the estimated F0 values. The center frequency of the bin with the highest value in the histogram is used as dominant frequency. Too large or small frequencies are halved or doubled respectively, until they fit into the chosen interval. By doing so, octave-errors are removed. Of course, also absolute pitch information is removed, but absolute pitch is mainly artist depended, and a contribution to rap / singing classification is not expected. The resolution of the YIN features is 1378 samples per second. Figure 1 and Figure 2 show the

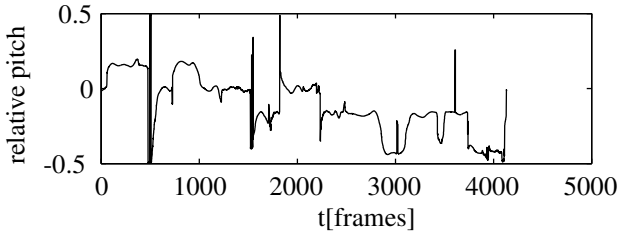


Figure 1. RPR progression of a singing snippet.

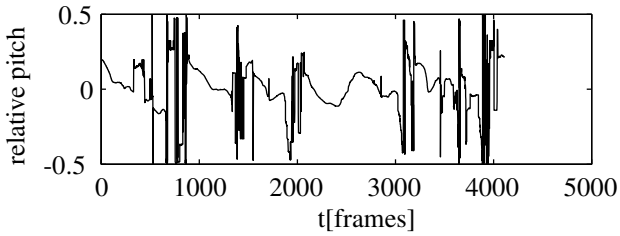


Figure 2. RPR progression of a rap snippet.

RPR progression for an exemplary singing and rap snippet respectively. One difference between the two examples is, that for singing, regions of almost constant pitch (RPR values of approximately 0.2, 0.0, -0.2, and -0.45 in Figure 1) can be observed, while for rap the RPR values are permanently changing. Based on RPR, IP, and ATR, a set of features is extracted.

First of all, the number of non-silent frames is determined, based on thresholding of IP. The ratio of non-silent frames to the number of overall frames will be denoted **ERatio**.

Next, from the non-silent frames, the number of voiced frames is determined, using a threshold on ATR. The ratio of voiced frames to the number of non-silent frames will be denoted **VRatio**. As already stated, rap is supposed to contain less voiced frames than song.

In another in-house study it has been discovered, that song segments have a lower syllable-density than rap segments. IP can be used as onset detection function. Based on adaptive thresholding, the number of onsets is estimated, which is then divided by the length of the segment. This feature is denoted **ORatio**.

As another step, from the voiced frames the segments are determined, during which $|\Delta\text{RPR}|$ is below a threshold. Segments of a length smaller then 10 frames are discarded. The ratio of frames that contribute to such a segment and the number of voiced frames is denoted **CRatio**. All the following calculations are performed on the RPR frames, that belong to a segment.

The mean of ΔRPR and the mean of ΔΔRPR also serve as features, denoted **PitchDiff** and **PitchDDiff**. Further, the mean of $|\text{RPR}|$, **MeanC**, and the variance of RPR, **VarC** are calculated.

The ratio of the number of frames with negative ΔRPR and the number of frames with positive ΔRPR is denoted **SLRatio**. In sung segments, either constant or with vibrato, both components are balanced. However, in rap segments a decreasing pitch can often be observed, and as a consequence, the SLRatio would be larger than 1.

A histogram over RPR with a resolution of 3 bins for each note is calculated, for a coarse approximation of the shape of the pitch distribution. Rap segments tend to have an unimodal RPR-distribution (Figure 3). Sung segments often have multimodal RPR-distributions, depending on the number of different notes that are sung in an utterance, as depicted in the example of Figure 4. Further, the RPR-distribution of a sung segment tends to have much sharper peaks than the distribution of a rap segment. The distance of the two bins with the largest values, divided by the width of the histogram will be denoted **NoteDist**. Dividing the second largest value in the histogram by the largest one, leads to the **NRatio**.

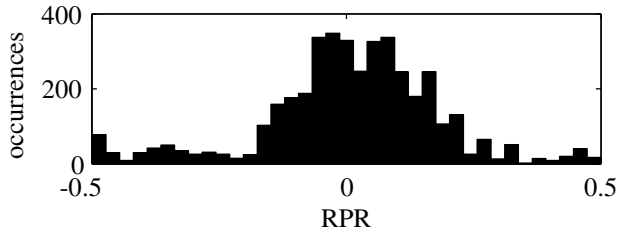


Figure 3. RPR-histogram for a rap snippet.

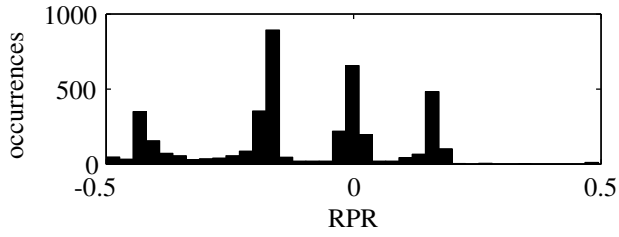


Figure 4. RPR-histogram for a singing snippet.

In addition, MFCCs are extracted from the audio signal. MFCCs are a popular feature in speech recognition and describe the spectral envelope. In [10], their applicability to modeling music has been shown, and as a consequence they have been successfully used in different music information retrieval tasks since then. For each snippet, the mean of all contributing frames is calculated. MFCCs are not part of the feature-set, they are used in a system for comparison reasons to describe the robustness of the feature-set in terms of utterance-sensitivity.

2.2 Classifier

Support vector machines (SVM, [2]) have been used as classifier. An SVM consists of a set of support vectors, that span a hyperplane in the feature space. This hyperplane separates two classes. The class of a test-observation depends on on which side of the hyperplane the test-observation is located in the feature-space. This can be calculated incorporating the dot-products of the feature vector and the support vectors. In the training stage, the support vectors are determined based on training observations.

In order to use non-linear hyperplanes, the feature space is transformed in a higher-dimensional space by the use of a kernel function. Computational costs for the transformation and the calculation of the dot-products can be reduced by selecting the kernel in a way that the dot-product can also be expressed in the original feature space. A radial basis function (RBF) kernel has been used, that is parameterized by γ.

Another parameter of the SVM is C, the weight of the error term during training. LibSVM [1] has been used as SVM implementation.

3. EXPERIMENTS

Following the approach section, the system setup, including the data, and the performed experiments are explained.

3.1 Data

A dataset of 62 songs from 60 different artists has been used in this study. Acapella versions of commercial Hip-Hop and contemporary R&B songs, performed in English language have been used. In these genres, songs are often released including an acapella and instrumental version. Other artists or DJs then can make remixes. For all songs, the segments containing only monophonic singing or monophonic rap have been determined.

Each segment is cut into 3s snippets, that overlap by 0.5s. The influence of the segment length is not evaluated in this paper. Although [12] reports that already snippets of 1 second length contain enough information for humans to accurately classify speech and singing, a larger snippet size has been chosen, since it is then more likely to observe notes with different pitches in singing snippets. The final dataset consists of 815 rap-snippets and 584 singing-snippets.

3.2 System setup and evaluation

Training and evaluation is performed using 5-fold cross-validation. All snippets are randomly distributed amongst the 5 folds using an utterance filter, which means, that all snippets from one song (belonging to one utterance) are distributed in the same fold. Each of the folds serves as test-data once and is part of the training-data in the other cases. The training data is used to determine the parameters of the SVM, i.e., the support vectors, C, and γ. It is crucial in SVM training / classification that all the features have approximately the same range. Therefore the data has to be normalized. Variance-normalization is used, in order to make the data zero mean unit variance. The mean μ and the standard deviation σ have to be estimated.

A reasonable choice of C and γ is important for good classification results. Both parameters are estimated using 3-fold cross-validation on the training data. This stage

will later be referred to as development stage. The distribution into the folds is done randomly again. However, at this point it is possible to decide whether an utterance filter should be applied or not. A three-stage grid-search has been employed. Since this process itself also consists of training and evaluation, μ and σ have to be determined every time the training-data changes due to recomposition from different folds.

Having determined C and γ, μ and σ are estimated based on the whole training-data, the training-data is normalized, and the SVM is trained with the previously determined C and γ. Finally, the performance is measured using the test-data, which is the test-observations from the one out of five folds, that has not been used for training and development.

The performance of a trained system both in evaluation A_t and development A_d is measured in accuracy. The accuracy of a classifier on given data is calculated by dividing the number of correctly classified test-observations by the number of all test-observations. Accuracy can be sensitive to imbalanced test-data. So if for example the test-data contains 80% observations from one class and only 20% observations from the other class, a classifier, that would always choose the same class would lead to a performance of either 80% or 20%, depending on which class he always choses. Therefore the test data is made balanced during the evaluation by randomly picking 584 observations from the 815 rap snippets.

The whole process, incorporating the random distribution into five folds, the development and training of the classifier, and its evaluation, is performed multiple times (denoted #runs), since this process contains random elements and is therefore indeterministic. The mean and variance of the accuracies in the test series are given as final measure, denoted $\mu_{A,t}$ and $\sigma^2_{A,t}$. Further, in Table 2 also $\mu_{A,d}$ and $\sigma^2_{A,d}$, are given, which are the accuracies during development for the chosen C and γ. Matlab is used as experimental framework.

3.3 Feature selection

A feature selection algorithm (FSA) has been used to give an estimation of the contribution of each of the features. Inertia ratio maximization using feature space projection [16] is a filter FSA, where the criteria of choosing features is distinct from the actual classifier. For each feature dimension an r-value is determined, which is the ratio of the between-class inertia to the total-class inertia. The feature with the largest r is chosen, then an orthogonalization process is applied to the feature space, in order to avoid the choice of redundant dimensions during following iterations. These steps are repeated until a stop criterium applies. The order of the features after feature selection reflects their importance according to the feature selection criterion, that should be correlated to the classification performance to a certain extend.

3.4 Utterance filter

One of the goals in machine learning is to build systems that are able to generalize. Also, performances of classifiers should be compared based on unseen test data. In order to achieve this, it is necessary to strictly discriminate training-data and testing-data during development and evaluation of the system. The distribution of the data in training and test-set can be even more restricted. It is common practice to put all the segments of a song in the same dataset, to for example avoid that the system is trained with a segment from the song and tested with a similar segment from the same song. In [15], it is suggested to put all pieces of an artist in the same dataset in a genre classification task. With experiments it is shown, that the performance of a system decreases significantly, if this so called artist filter is used. A possible reason for this is, that the system might focus on perceptually not so relevant information such as production effects [14].

As described in 3.2, an utterance filter is always applied in the 5-fold cross-validation setup, since it is possible, that the suggested feature set also reflects utterance-specific characteristics. In the 3-fold cross-validation development stage however, the utterance-filter can be either applied or omitted. Comparing performances based on systems with and without utterance-filter helps in describing the robustness towards utterance-specific characteristics. If a system generalizes well, $\mu_{A,t}$ and $\mu_{A,d}$ should be approximately equal.

The mean over the MFCC-frames of a snippet is a feature, that is supposed to be utterance-specific. In 4.3, the use of an utterance-filter is analyzed for the proposed feature-set and the mean-MFCC feature.

4. RESULTS AND DISCUSSION

The results of the performed experiments are listed and discussed in this section.

4.1 Feature contribution

In Table 1, the outcome of the FSA is denoted. Overall, feature selection has been performed 69425 times. In all runs, the VRatio feature has been selected first, as can be seen in column 2, belonging to rank 1. Further important features are CRatio, SLRatio and ORatio, that have been chosen 54989, 8175, and 6240 times as second feature respectively. The most unimportant features according to the IRMFSP are PitchDDiff and VarC (often chosen on rank 10 or 11 according to the values in column 11 and column 12).

The mean r-value of the first selected feature is 0.52, followed by 0.47 for the second selected feature. r decreases drastically from the second to the third selected feature. In [16], it is suggested to stop the iterative feature selection process as soon as r of the current iteration is below 1/100 of r in the first iteration. Following this criterion, the 6 top features would have been selected.

Rank	1	2	3	4	5	6	7	8	9	10	11
$\bar{r}$	0.5191	0.4716	0.0459	0.0151	0.0094	0.0068	0.0046	0.0034	0.0018	0.0004	0.0001
CRatio	0	54989	0	2	91	54	891	4511	5932	1678	1277
ERatio	0	0	0	359	2218	6062	13300	24101	17819	3644	1922
MeanC	0	0	0	1367	21200	9070	8650	13060	12116	3257	705
NoteDist	0	21	6432	50987	8455	2720	686	120	4	0	0
NRatio	0	0	0	3505	12729	8077	12597	9916	21221	1305	75
ORatio	0	6240	1880	10822	11029	9791	20787	7385	1376	112	3
PitchDDiff	0	0	0	0	5	54	794	3593	7596	39962	17421
PitchDiff	0	0	12249	2342	12096	27695	9244	4585	1195	17	2
SLRatio	0	8175	48864	41	1601	5902	2464	1999	357	22	0
VarC	0	0	0	0	1	0	12	155	1809	19428	48020
VRatio	69425	0	0	0	0	0	0	0	0	0	0

Table 1. Ranks of different features in the feature selection process.

4.2 System Performance

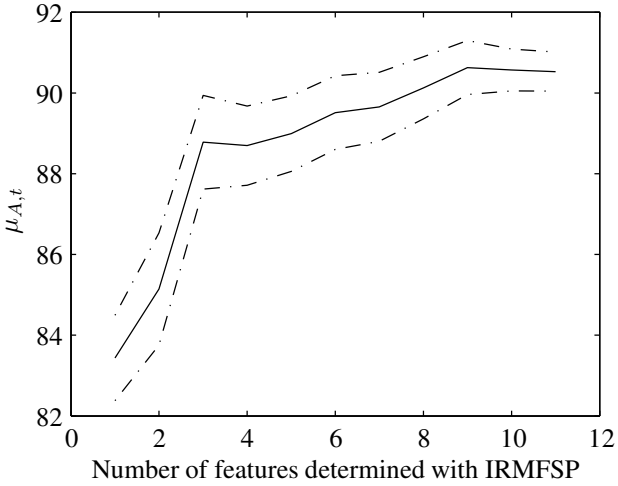


Figure 5. Performance subject to the number of features.

The final performance of the system is plotted against the number of features after IRMFSP in Figure 5. The top performance, 90.62% is achieved using 9 features. Using the feature-set consisting of all 11 features leads to a mean accuracy of 90.53%. The largest gain in performance is reported from 2 features (85.14%) to 3 features (88.78%).

4.3 Influence of the utterance filter

Table 2 contains the results of the investigation of utterance-sensitivity. For the suggested feature-set (full) the performance decrease is 1.09% (91.63% down to 90.54% from development to testing) with utterance filter. Without utterance-filter 3.07% (from 93.88% down to 90.81%) are observed. These small decreases originate in the fact that $\mu_{A,d}$ is result of the optimization of C and γ, while $\mu_{A,t}$ is not. Further, during development, imbalanced test-data is used for the evaluation, which can also lead to differences between both values. On the full feature-set, $\mu_{A,t}$ is almost similar for both systems, with and without utterance filter.

Feat.	u.filter	$\mu_{A,t}$	$\sigma^2_{A,t}$	$\mu_{A,d}$	$\sigma^2_{A,d}$	#runs
full	yes	90.54	0.53	91.63	0.16	1588
full	no	90.81	0.49	93.88	0.03	1583
MFCC	yes	67.71	6.71	72.84	1.17	1084
MFCC	no	65.08	3.85	96.36	0.04	934

Table 2. Influence of the utterance-filter.

Applying an utterance-filter to the MFCC-feature results to an decrease from 5.13% (from 72.84% down to 67.71%), which again can be explained with the optimization procedure. If the system is trained with the MFCC-feature without using an utterance-filter, the development-performance is 96.36%, which is the highest one achieved in the experiments. But on new utterances, the performance drastically decreases to 65.08%. In our data, artists that rap do not sing and vice versa. Without the utterance-filter, different parts of the same utterance are in the test-set and the training-set during the system-development, and a task like that can also be performed by an artist-detection or utterance-detection system. MFCCs are well known for their capabilities to capture speaker characteristics, and are therefore often used in speaker recognition systems. So in the development stage, the system is trained to classify into rap and singing by actually identifying utterances. A $\mu_{A,d}$-value of 96.36% shows, that, MFCCs are an appropriate feature for this task. On the contrary, $\mu_{A,t}$ is determined classifying snippets from unknown utterances. An utterance detection system cannot do that well, which leads to a low accuracy of 65.08%. For the MFCC-system with utterance filter, as already reported the difference is much smaller. For the full feature-set, no large difference between $\mu_{A,t}$ and $\mu_{A,t}$ could be observed. This set therefore is not sensitive to utterance-specific characteristics.

Comparing $\mu_{A,t}$ for the MFCC-systems with and without utterance-filter, one can see that the system trained with utterance-filter performs 2.63% better. A possible reason is that MFCCs seem to be able to also classify based the vocal characteristics to a certain extend, but when trained

without utterance-filter, the classifier seems to "learn the task that is easier to perform", which might be utterance-identification instead of vocal characteristics classification. When trained with utterance-filter, there is no utterance-identification development data provided. But since the difference is so small, there might be other reasons.

5. CONCLUSIONS AND OUTLOOK

A system for the classification of isolated vocal tracks into the classes singing and rap has been presented. A feature set, motivated by differences perceived by human is developed. Accuracies of over 91% are achieved on 3 second snippets of isolated vocal tracks from commercial urban music recordings. Further, it has been shown in experiments with an utterance-filter, that the suggested feature-set is not sensitive to utterance-specific characteristics.

As a next step, the application on full tracks, where no isolated vocal tracks are available, will be investigated. Since the described system is not designed to also work on mixtures of vocal tracks and accompaniment, the vocal track has to be separated from the song. Methods for the separation of the vocal track as described in, e.g., [13, 20, 21] are currently investigated. The system that has been described in this paper can also serve as benchmark for the source separation algorithms. Further, a study incorporating listening test is intended, in order to evaluate human performance on this task.

6. REFERENCES

[1] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273:297, 1995.

[3] Alain de Cheveigné and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustic Society of America*, 111 (4):1917:1930, 2002.

[4] Daniel Gärtner and Christian Dittmar. Vocal characteristics classification of audio segments: An investigation of the influence of accompaniment music on low-level features. In *Proceedings of the ICMLA*, 2009.

[5] David Gerhard. *Computationally measurable differences between speech and song*. PhD thesis, Simon Fraser University, Canada, 2003.

[6] Ferdinand Hörner and Oliver Kautny. *Die Stimme im HipHop*. transcript Verlag, 2009.

[7] Youngmoo E. Kim. *Singing Voice Analysis/Synthesis*. PhD thesis, Massachusetts Institute of Technology, 2003.

[8] Yipeng Li and DeLiang Wang. Separation of singing voice from music accompaniment for monaural recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1475–1487, May 2007.

[9] George List. The boundaries of speech and song. *Ethnomusicology*, 7(1):1:16, January 1963.

[10] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of ISMIR*, 2000.

[11] International Federation of the Phonographic Industry. IFPI Digital Music Report 2010. Available at http://www.ifpi.org/content/library/DMR2010.pdf.

[12] Yasunori Ohishi, Masataka Goto, Katunobu Itou, and Kazuya Takeda. On human capability and acoustic cues for discriminating singing and speaking voices. In *Proceedings of ICMPC*, 2006.

[13] Alexey Ozerov, Pierrick Philippe, Frédéric Bimbot, and Rémi Gribonval. Adaptation of bayesian models for single-channel source separation and its application to voice/music separation in popular songs. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5):1564–1578, July 2007.

[14] Elias Pampalk. *Computational Models of Music Similarity and their Application to Music Information Retrieval*. PhD thesis, Vienna University of Technology, Austria, March 2006.

[15] Elias Pampalk, Arthur Flexer, and Gerald Widmer. Improvements of audio-based music similarity and genre classification. In *Proceedings of ISMIR*, London, UK, 2005.

[16] Geoffroy Peeters and Xavier Rodet. Hierarchical gaussian tree with inertia ratio maximization for the classification of large musical instruments databases. In *Proceedings of DAFx*, 2003.

[17] Thomas S. Polzin. Verbal and non-verbal cues in the communication of emotions. In *Proceedings of ICASSP*, 2000.

[18] Björn Schuller, Gerhard Rigoll, and Manfred Lang. Hidden markov model-based speech emotion recognition. In *Proceedings of ICASSP*, 2003.

[19] Björn Schuller, Gerhard Rigoll, and Manfred Lang. Discrimination of speech and monophonic singing in continuous audio streams applying multi-layer support vector machines. In *Proceedings of ICME*, volume 3, pages 1655–1658, 2004.

[20] Shankar Vembu and Stephan Baumann. Separation of vocals from polyphonic audio recordings. In *Proceedings of ISMIR*, 2005.

[21] Tuomas Virtanen, Annamaria Mesaros, and Matti Ryynänen. Combining pitch-based inference and non-negative spectrogram factorization in separating vocals from polyphonic music. In *Proceedings of the ISCA Tutorial and Research Workshop on Statistical And Perceptual Audition*, 2008.

SINGING PITCH EXTRACTION BY VOICE VIBRATO/TREMOLO ESTIMATION AND INSTRUMENT PARTIAL DELETION

Chao-Ling Hsu **Jyh-Shing Roger Jang**

Multimedia Information Retrieval Laboratory
Computer Science Department, National Tsing Hua University
Hsinchu, Taiwan
{leon, jang}@mirlab.org

ABSTRACT

This paper proposes a novel and effective approach to extract the pitches of the singing voice from monaural polyphonic songs. The sinusoidal partials of the musical audio signals are first extracted. The Fourier transform is then applied to extract the vibrato/tremolo information of each partial. Some criteria based on this vibrato/tremolo information are employed to discriminate the vocal partials from the music accompaniment partials. Besides, a singing pitch trend estimation algorithm which is able to find the global singing progressing tunnel is also proposed. The singing pitches can then be extracted more robustly via these two processes. Quantitative evaluation shows that the proposed algorithms significantly improve the raw pitch accuracy of our previous approach and are comparable with other state of the art approaches submitted to MIREX.

1. INTRODUCTION

The pitch curve of the lead vocal is one of the most important elements of a song as it represents the melody. Hence it is broadly used in many applications such as singing voice separation, music retrieval, and auto-tagging of the songs.

Lots of work which focuses on extracting the main melody of songs has been proposed in the literature. Poliner et al. [1] comparatively evaluated different approaches and found that most of the approaches roughly follow the general framework as follows: Firstly, the pitches of different sound sources are estimated at a given time and some of them are then selected as the candidates. The melody identifier then chooses one, if any, of these pitch candidates as a constituent of the melody for each time frame. Finally the output melody line is formed after smoothing the raw pitch line. Since the goal of most of these approaches is to extract the melody line carried by not only the singing voice but also the music instruments, they do not consider the different characteristics between the human singing voice and instruments: formants, vibrato and tremolo. More related work can be found in our previous work [3].

In the present study, we apply the method suggested by Regnier and Peeters [2], which was originally used to detect the presence of singing voice. This method utilizes the vibrato (periodic variation of pitch) and tremolo (periodic variation of intensity) characteristics to discriminate the vocal partials from the music accompaniment partials. We apply this technique to the singing pitch extraction so that the singing pitches can be tracked with less interference of instrument partials.

The rest of this paper is organized as follows. Section 2 describes the proposed system in detail. The experimental results are presented in section 3, and section 4 concludes this work with possible future directions.

2. SYSTEM DESCRIPTION

Fig. 1 shows the overview of the proposed system. The sinusoid partials are first extracted from the musical audio signal. The vibrato and tremolo information is then estimated for each partial. After that, the vocal and instrument partials can be discriminated according to a given threshold, and the instrument partials can be therefore deleted. With the help of instrument partials deletion, the trend of the singing pitches can be estimated more accurately. This trend is referred to as global progressing path and indicates a series of time-frequency regions (T-F regions) where the singing pitches are likely to be present. Since the T-F regions consider relatively larger periods of time and larger ranges of frequencies, they are able to provide robust estimations of the energy distribution of the extracted sinusoidal partials.

On the other hand, the normalized sub-harmonic summation (NSHS) map [3] which is able to enhance the harmonic components of the spectrogram is computed, and the instrument partials which are discriminated with lower thresholds are deleted from NSHS map. After that, the global trend is applied to the instrument-deleted NSHS map.

The energy at each semitone of interest (ESI) [3] is then computed from the trend-confined NSHS map. Finally, the continuous raw pitches of the singing voice are

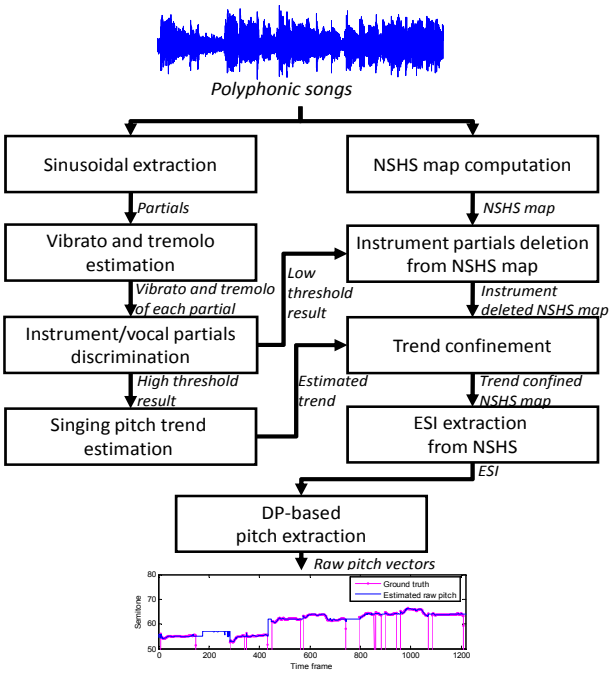


Figure 1. System overview

estimated by tracking the ESI values using the dynamic programming (DP) based pitch extraction.

An example is shown in the evaluation section (3.2). The following subsections explain these blocks in detail.

2.1 Sinusoidal Extraction

This block extracts the sinusoidal partials from the musical audio signal by employing the multi-resolution FFT (MR-FFT) proposed by Dressler [4]. It is capable of covering the fast signal changes and maintaining an adequate discrimination of concurrent sounds at the same time. Both of these properties are extremely well justified for the proposed approach.

The extracted partials with short duration are excluded in this stage because they are more likely to be produced by some percussive instruments or unstable sounds.

2.2 Vibrato and Tremolo Estimation

After extracting the sinusoidal partials, the vibrato and tremolo information of each partial are estimated by this block by applying the method suggested by Regnier and Peeters [2].

Vibrato refers to the periodic variation of pitch (or frequency modulation, FM) and tremolo refers to the periodic variation of intensity (or amplitude modulation, AM). Due to the mechanical aspects of the voice production system, human voice contains both types of the modulations at the same time, but only a few musical instruments can produce them simultaneously [5]. In general, wind and brass instruments produce AM dominant sounds, while string instruments produce the FM dominant sounds.

Two features are computed to describe vibrato and tremolo: frequencies (the rate of vibrato or tremolo) and amplitudes (the extent of vibrato or tremolo). For human singing voice, the average rate is around 6Hz [6]. Hence we determine the relative extent values around 6Hz by using the Fourier transform for both vibrato and tremolo.

More specifically, to compute a relative extent value of vibrato for a partial $p_k(t)$ existing from time t_i to t_j, the Fourier transform of its frequency values $f_{p_k}(t)$ is given by:

$$F_{p_k}(f) = \sum_{t=t_i}^{t_j} (f_{p_k}(t) - \mu_{f_{p_k}}) e^{-2i\pi f \frac{t}{L}},$$

where $\mu_{f_{p_k}}$ is the average frequency of $p_k(t)$ and $L = t_j - t_i$. The relative extent value in Hz is given by:

$$\Delta f_{relp_k}(f) = \frac{F_{p_k}(f)}{L\mu_{f_{p_k}}}.$$

Lastly, the relative extent value around 6Hz is computed as follow:

$$\Delta f_{p_k} = \max_{f \in [4,8]} \Delta f_{relp_k}(f).$$

The relative extent value for tremolo can be computed in the same way except that amplitude a_{p_k} is used instead of f_{p_k}.

2.3 Instrument/Vocal Partials Discrimination

The instrument and vocal partials are discriminated according to the given thresholds of the relative extent of vibrato and tremolo. The instrument partials can then be deleted if both the relative extents are lower than specified values. By selecting the thresholds, we can adjust the trade-off between instrument partials deletion rate and vocal partials deletion error rate. The higher thresholds are, the more instrument partials are deleted, but the more deletion errors of the vocal partials are. Usually a lower threshold is applied for instrument partials deletion from NSHS map, while a higher threshold is applied for the singing pitch trend estimation. The reasons will be explained in the following subsections.

2.4 Singing Pitch Trend Estimation

One of the major error types of singing pitch extraction is the doubling and halving errors where the harmonics or sub-harmonics of the fundamental frequency are erroneously recognized as the singing pitches. Here we refer the harmonic partials to those partials whose frequencies are multiples of the F0 partials. And we use "vocal partials" to indicate the union of the disjoint sets of "vocal F0 partials" and "vocal harmonic partials". Although the error can be handled by considering the time and frequency smoothness of the pitch contours, most of the approaches only consider the local smoothness during a short period of time. However, there are many 'gaps' between successive vocal partials such as the non-vocal pe-

riod between two segments of lyrics where instrument partials may be predominant in these gaps. These instrument partials often act like 'bridges' which may mislead the pitch tracking algorithm to connect two vocal partials erroneously.

To deal with this problem, we propose a method to estimate the trend of the singing pitches. Firstly, higher thresholds are applied to delete more instrument partials. This might also delete some vocal partials, but it will not affect the pitch trend estimation as long as we still have enough vocal partials. Secondly, the harmonic partials are deleted based on the assumption that the lowest-frequency partial within a frame is the vocal F0 partial. Moreover, these deleted harmonic partials are accumulated into their vocal F0 partials. This process is repeated until we have only several low-frequency partials representing potential vocal F0 partials. As a result, most of the harmonic partials are deleted and the energy of the vocal F0 partials is strengthened. The energy of the remaining partials is then max-picked for each frame and summed up within a time-frequency region (T-F region). More precisely, given a spectrogram $x[t,f]$ computed from the previous MR-FFT, the strength $s_{T,F}$ of the T-F region is defined as:

$$s_{T,F} = \sum_{t=0}^{M_{time}-1} \max_{f\in[0,M_{freq}-1]} x[t+TL_{time}, f+FL_{freq}],$$
$$T = 0,1,...n-1 \text{ and } F = 0,1,...m-1$$

where

t	is the index of the time frame.
f	is the index of the frequency bin.
n	is the number of T-F regions in the time axis
m	is the number of T-F regions in the frequency axis
T , F	are the indices of the T-F region in time and frequency axes respectively.
L_{time} , L_{freq}	are the time and frequency advance of the T-F region (hop-size) respectively.
M_{time} , M_{freq}	are the number of the time frames and the number of the frequency bins of a T-F region respectively.

The size of the T-F region should be large enough so that the global trend of the singing pitches can be acquired. On the other hand, the T-F region should also be small enough so that the harmonics of the singing pitches can be separated in different frequency bands and the pitch changes can be captured in different time periods. Note that although M_{freq} is fixed for all T-F regions, the frequency ranges are different for the T-F regions in different frequency bands. This is because the frequency bins in the result of sinusoidal extraction via MR-FFT are spaced by 0.25 semitone. In other words, the lower frequency T-F region has smaller frequency range since the frequency differences between low fundamental frequency partials and their harmonics are relatively smaller than that of high fundamental frequency partials.

Because the singing pitch trend should be smooth, the problem is defined as the finding of an optimal path $[F_0,\cdots,F_i,\cdots,F_{n-1}]$ that maximizes the score function:

$$score(F,\theta) = \sum_{T=0}^{n-1} s_{T,F_T} - \theta \times \sum_{T=1}^{n-1} |F_T - F_{T-1}|,$$

where s_{T,F_T} is the strength of the T-F region at the time index T and frequency index F_T. The first term in the score function is the sum of strength of the T-F region along the path, while the second term controls the smoothness of the path with the use of a penalty coefficient θ. If θ is larger, the computed path is smoother.

The dynamic programming technique is employed to find the maximum of the score function, where the optimum-valued function $D(T,l)$ is defined as the maximum score starting from time index 1 to T, with $F_T = l$:

$$D(T,l) = s_{T,l} + \max_{k\in[0,m-1]} \{D(t-1,k) - \theta \times |k-l|\},$$

where $t = [1,n-1]$, and $l = [0,m-1]$. The initial condition is $D(0,l) = s_{0,l}$, and the optimum score is equal to $\max_{l\in[0,m-1]} D(n-1,l)$. At last, this optimal path is applied to the instrument-deleted NSHS map described in section 2.6.

2.5 NSHS Computation

Instead of simply extracting the singing pitches by tracking the remaining vocal partials, the NSHS proposed by our previous work [3] is used since the non-peak values of the spectrum are also useful for the later DP-based pitch extraction algorithm. The NSHS is able to enhance the partials of harmonic sound sources, especially the singing voice. It is modified from the sub-harmonic summation [7] by adding a normalizing term. The reason of the modification is based on the observation that most of the energy in a song locates at the low frequency bins, and the energy of the harmonic structures of the singing voice decays slower than that of instruments [8]. It is therefore that, when more harmonic components are considered, energy of the vocal sounds is further strengthened.

2.6 Instrument partials deletion and trend confinement

In these two blocks, the instrument partials detected with the lower thresholds in the previous block are first removed from the NSHS map by setting their magnitude to zero (within the range of neighboring local minima). For extracting singing pitches, the thresholds are set to be lower in order to delete the instrument partials without deleting too many vocal partials. After that, the instrument deleted NSHS map can be further confined to the estimated pitch trend (section 2.4). In other words, only the energy along the trend will be retained.

2.7 ESI Extraction from NSHS

The ESI computed from the trend-confined NSHS map in the time frame t can be obtained as follows [3]:

$$v_t(n) = \max_{p_n - \frac{p_n - p_{n-1}}{2} \le p < p_n + \frac{p_{n+1} - p_n}{2}} \left(A_t(f)\right),$$

where $A_t(*)$ is the NSHS map calculated in the previous stage, $n = 0,1,..,N-1$, N is the total number of semitones that are taken into account, and p_n is the frequency of the n-th semitone in the selected pitch range.

Note that we also need to record the maximal frequency within each frequency range of ESI in order to reconstruct the most likely pitch contours.

2.8 DP-based Pitch Extraction

The DP-based pitch tracking algorithm is previously proposed in [3]. It is very similar to the algorithm described in section 2.4. The most likely pitch contour can be finally acquired by tracking the ESI computed in the previous block. Note that we do not perform vocal/non-vocal detection since it is not the focus of this study. In addition, the vocal/non-vocal detection can be implemented by various methods such as [2][3].

3. EVALUATION

Two datasets were used to evaluate the proposed approach. The first one, MIR-1K, is a publicly available dataset proposed in our previous work [9]. It contains 1000 song clips recorded at 16 kHz sample rate with 16-bit resolution. The duration of each clip ranges from 4 to 13 seconds, and the total length of the dataset is 133 minutes. These clips were extracted from 110 karaoke songs which contain a mixed track and a music accompaniment track. These songs were selected (from 5000 Chinese pop songs) and sung, consisting of 8 females and 11 males. Most of the singers are amateurs with no professional training. The music accompaniment and the singing voice were recorded at the left and right channels respectively. The ground truth of the pitch values of the singing voices were first estimated from the pure singing voice and then manually corrected. All songs are mixed at 0 dB SNR, indicating that the energy of the music accompaniment is equal to the singing voice. Note that the SNRs for commercial pop songs are usually larger than zero, indicating that our experiments were set to deal with more adversary scenarios than the general cases. The second dataset, ADC2004, is one of the testing dataset for audio melody extraction task in MIREX. It contains 20 song clips and the average length of the clips is around 20 seconds. Only the 12 vocal songs of ADC2004 are used for testing in this study. Although the size of ADC2004 is much smaller than that of MIR-1K, it is convenient for comparing the performance of different algorithms which were submitted to MIREX.

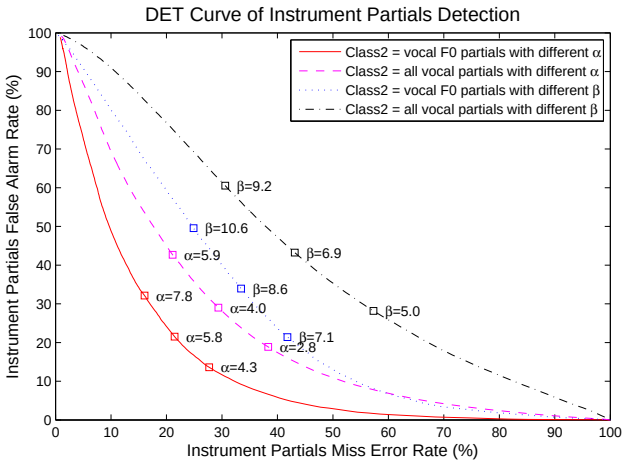


Figure 2. The DET curves of instrument partials false alarm rate versus instrument partials miss error rate by using different values of α and β as the thresholds alone, respectively. (Here we assume class 1 is instrument partials, and class 2 is either vocal F0 partials or all vocal partials.)

3.1 Evaluation for Instrument Partials Detection

The frame size and hop size used in the sinusoidal extraction by MR-FFT are 64 ms and 8 ms respectively. The frequency bins in MR-FFT are spaced by 0.25 semitone from 80Hz to 1280Hz, resulting a total of 192 bins. The partials whose durations are less than 56 ms are removed since they are more likely to be generated by percussive instruments or unstable sounds. With regard to the relative vibrato and tremolo extent estimation, the parameters are set to be the same as those suggested by [2].

Figure 2 shows the DET (detection error tradeoff) curves of instrument partials false alarm rate versus instrument partials miss error rate by using different relative vibrato extent (α) and relative tremolo extent (β) as the thresholds alone, respectively. A higher instrument partials false alarm rate indicates more vocal partials are erroneously recognized as instrument partials. On the other hand, a higher instrument partials miss error rate indicates more instrument partials are recognized as vocal partials. Here we assume class 1 is instrument partials, and class 2 is either vocal F0 partials or all vocal partials. The solid line and dotted line show the results of using vocal F0 partials as class 2 with different α and β respectively. The dashed line and dash-dot line show the results of using all vocal partials as class 2 with different α and β respectively. We want to show the results of using vocal F0 partials as class 2 because the goal of this study is to extract the singing pitches carried by these vocal F0 partials. In contrast, the harmonic partials of the singing voice are comparably not as important. All of these partials are extracted from the MIR-1K dataset. Since the MIR-1K has separated tracks of singing voice and accompaniment, the sources of the partials can be distinguished.

From Figure 2, it is obvious that α has better discriminative capability to detect instrument partials than β.

	Vocal F0	Non-vocal F0
Partials remaining in the pitch trend tunnel	82.47 %	19.19 %
Partials remaining in the pitch trend tunnel but deleted by instrument partial deletion	8.07 %	66.18 %
Final partials remaining	75.82%	6.49%
Vocal pitches remaining in the pitch trend tunnel	86.30%	

Table 1. Performance of singing pitch trend estimation

This is because the pop music in MIR-1K has less wind and brass instruments than string instruments. We have found in our preliminary experiment[1] that β has better vocal/instrument discriminative power for wind and brass instruments.

The instrument partials deletion block applied α = 0.1125 and β = 3. The vocal F0 remaining rate is around 94.3% (or equivalently, 5.7% instrument partials false alarm rate) and instrument partial deletion rate is around 60.4% (or equivalently, 39.6% instrument partials miss error rate). On the other hand, singing pitch trend estimation applied α = 0.3 and β = 5.5 as the thresholds. The vocal F0 partials remaining rate is 72.9% and instrument partials deletion rate is 82.8%.

3.2 Evaluation for Singing Pitch Trend Estimation

The parameters for this experiment were set as follows. The sizes along time and frequency axes for each T-F region were 3 seconds and 13.5 semitones, respectively. Their hop sizes were 1.5 seconds and 4 semitones, respectively. The penalty coefficient θ for the dynamic programming step was set to 1 empirically.

Table 1 shows the results of the singing pitch trend estimation. More than 82% of vocal F0 partials remain in the pitch trend tunnel and the singing pitches remaining rate is 86%. On the other hand, only 19.19% of instrument and vocal harmonic partials are retained within the pitch trend tunnel. In addition, 66.18% of the non-vocal F0 partials left in the pitch trend tunnel are deleted by the NSHS computation stage, and 8.07% of the remaining vocal F0 partials are deleted erroneously at the same time. Finally, 75.82% of vocal F0 partials remain while only 6.49% of non-vocal F0 partials are kept in both deletion procedures.

Figure 3 shows the stage-wise results in singing pitch extraction. Figure 3(a) shows all the partials after sinusoidal extraction. Figure 3(b) and 3(c) applies different thresholds on 3(a) to delete instrument partials for different purposes. Because 3(b) applies lower thresholds than those of 3(c), more instrument partials are removed in 3(c). The harmonic partials in Figure 3(c) are then further deleted in 3(d). Figure 3(f) is obtained by subtracting the

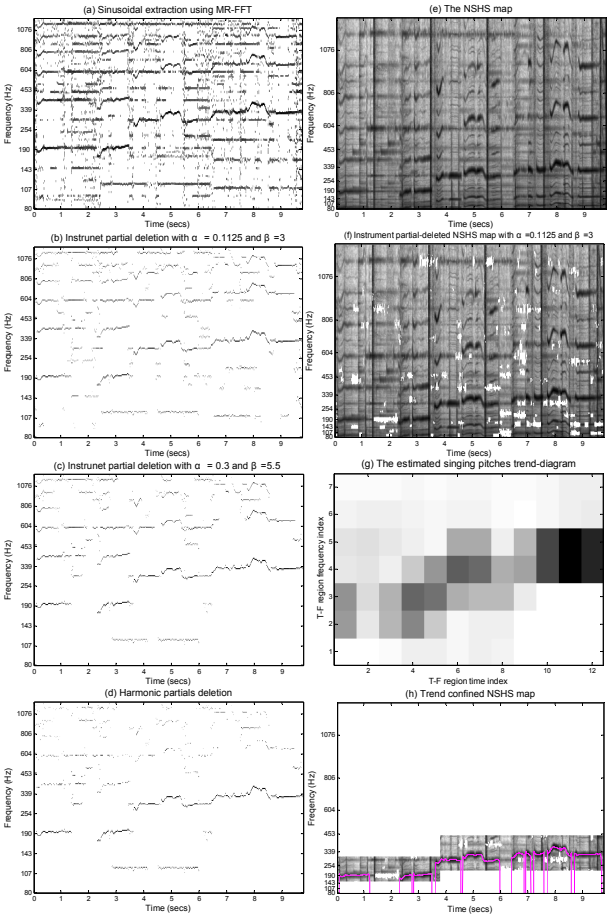


Figure 3. Stage-wise results of singing pitch extraction for the clip 'Ani_4_05.wav' in MIR-1K. (a) Results after sinusoidal extraction using MR-FFT. (b) The remaining partials after instrument partial deletion thresholds of α = 0.1125 and β = 3. (c) The remaining partials after instrument partial deletion after threshold of α = 0.3 and β = 5.5. (d) The result after harmonic partials deletion. (e) The NSHS map. (f) Instrument partial-deleted NSHS map with threshold of α = 0.1125 and β = 3. (g) The estimated singing pitches trend-diagram. (h) Trend confined NSHS map, where the solid line represents the ground truth of the singing pitches.

detected instrument partials in Figure 3(b) from the NSHS map in 3(e). Figure 3(g) illustrates the T-F regions computed from Figure 3(d), with color depth indicating the strength each T-F region. Finally, Figure 3(h) is the NSHS map (Figure 3(f)) confined by the pitch trend tunnel. As can be seen in this example, the identified pitch trend tunnel is capable of covering the vocal F0 partials (represented by solid lines) while most of the instrument partials are deleted.

3.3 Evaluation for Singing Pitch Extraction

Figure 4 shows the results of singing pitch extraction. The raw pitch accuracy is computed over the frames which were labeled as voiced in the ground truth. An estimated singing pitch is considered as correct if the deviation from the ground truth is small than 1/4 tone (or 1/2

[1] The experiment was also performed on the University of Iowa Musical Instrument Samples which is available at http://theremin.music.uiowa.edu/

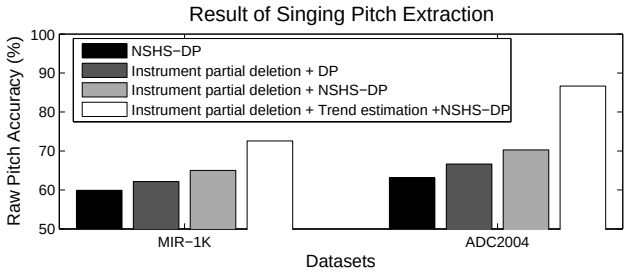


Figure 4. The results of singing pitch extraction.

semitone). The black bars show the performance of the

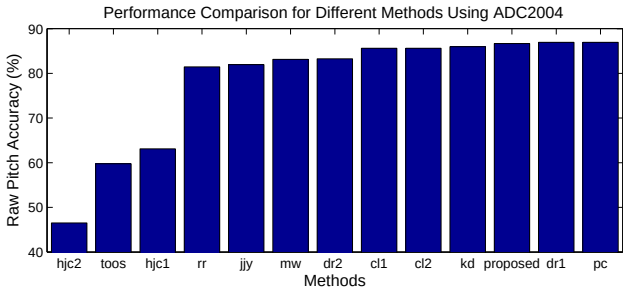


Figure 5. Performance comparison.

previous NSHS-DP method [3] (ranked 5-th out of 12 in MIREX2009). The dark gray bars show the result of combining the proposed instrument partial deletion and dynamic programming without using the NSHS. The light gray bars are the same as the dark gray bar except that the NSHS map is applied. The light gray bars perform better than the ones without using the NSHS map, which confirms the argument that the non-peak values of the spectrum are also useful. Lastly the white bars show the performance of the proposed approach where instrument partial deletion, singing pitch trend estimation, and NSHS are applied.

It is clear that the proposed instrument partial deletion and singing pitch trend estimation facilitate extracting singing pitches since its performance improves significantly over the rest of the compared methods in both datasets. The raw pitch accuracy of proposed approach achieves 72.57% and 86.67% for MIR-1K and ADC2004, respectively, with the same setting of the parameters described in previous subsections. Comparing to the MIREX 2009 results shown in Figure 5, the performance of the proposed approach is comparable to the state of the art approaches.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel approach for singing pitch extraction by deleting instrument partials. It is surprising that the vocal and instrument partials can be discriminated by only two simple features, and the performance is also encouraging. Besides, a singing pitch trend estimation algorithm is proposed to enhance the pitch extraction accuracy.

Since only the features suggested in [2] were used in this study, other characteristics of voice vibrato and tremolo could be use as new features for improving the performance. Moreover, it is worth noting that the proposed instrument partial deletion and singing trend estimation techniques are general for pitch extraction, in the sense that they can be applied to any other spectrum-based methods to delete the unlikely pitch candidates. Our immediate future work is to explore the use of the proposed techniques on top of existing methods to confirm their feasibility in further improving the performance.

5. ACKNOWLEDGEMENT

This work was conducted under the ”Digital Life Sensing and Recognition Application Technologies Project” of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs of the Republic of China.

6. REFERENCES

[1] G. E. Poliner, D. P. W. Ellis, A. F. Ehmann, E. Gomez, S. Streich, and B. Ong, "Melody transcription from music audio: approaches and evaluation," *IEEE TASLP*, vol. 15, pp. 1247-1256, 2007.

[2] L. Regnier and G. Peeters, “Singing voice detection in music tracks using direct voice vibrato detection,” *IEEE ICASSP*, pp. 1685-1688, 2009.

[3] C. L. Hsu, L. Y. Chen, J. S. Jang, and H. J. Li, “Singing pitch extraction from monaural polyphonic songs by contextual audio modeling and singing harmonic enhancement”, *ISMIR*, pp. 201-206, 2009.

[4] K. Dressler, “Sinusoidal extraction using an efficient implementation of a multi-resolution FFT,” *DAFx*, pp. 247–252, 2006

[5] V. Verfaille, C. Guastavino, and P. Depalle, “Perceptual evaluation of vibrato models,” *Proceedings of Conference on Interdisciplinary Musicology*, 2005.

[6] E. Prame, “Measurements of the vibrato rate of ten singers,” *JASA,* vol. 96, pp. 1979, 1994.

[7] D. J. Hermes, “Measurement of Pitch by Subharmonic Summation,” *JASA*, vol.83, pp. 257-264, 1988.

[8] Y. Li and D. L. Wang, “Detecting pitch of singing voice in polyphonic audio,” *IEEE ICASSP*, pp. 17–20, 2005.

[9] C. L. Hsu and J. S. Jang, “On the improvement of singing voice separation for monaural recordings using the MIR-1K dataset,” *IEEE TASLP*, volume 18, pp. 310-319, 2010.

SONGWORDS: EXPLORING MUSIC COLLECTIONS THROUGH LYRICS

Dominikus Baur, Bartholomäus Steinmayr, Andreas Butz
Media Informatics Group
University of Munich (LMU), Munich, Germany
{dominikus.baur,andreas.butz}@ifi.lmu.de, steinmayr@cip.ifi.lmu.de

ABSTRACT

The lyrics of a song are an interesting, yet underused type of symbolic music data. We present SongWords, an application for tabletop computers that allows browsing and exploring a music collection based on its lyrics. SongWords can present the collection in a self-organizing map or sorted along different dimensions. Songs can be ordered by lyrics, user-generated tags or alphabetically by name, which allows exploring simple correlations, e.g., between genres (such as *gospel*) and words (such as *lord*). In this paper, we discuss the design rationale and implementation of SongWords as well as a user study with personal music collections. We found that lyrics indeed enable a different access to music collections and identified some challenges for future lyrics-based interfaces.

1. INTRODUCTION

Lyrics are an important aspect of contemporary popular music. They are often the most representative part of a song. They verbally encode the songs general message, thereby strongly contributing to its mood. For most people, singing along is one of the easiest ways to actively participate in the music experience. Lyrics are also regularly used for identifying a song, since the first or most distinct line of the chorus often also is the song's title. This importance of lyrics makes purely instrumental pieces rather rare in contemporary popular music.

Despite this central role of lyrics, computer interfaces mostly still ignore them. Media player software for personal computers mostly only shows lyrics after installing additional plug-ins, and although the ID3 metadata standard for digital music contains a field for lyrics, it is rarely used. More complex operations, such as browsing and searching based on lyrics, are even further away and scarcely touched in research (e.g., [6]). We therefore think that looking at music from the perspective of lyrics can allow users a fresh view on their collection, reveal unknown connections between otherwise different songs and allow them to discover new patterns between the lyrics and other aspects of the music.

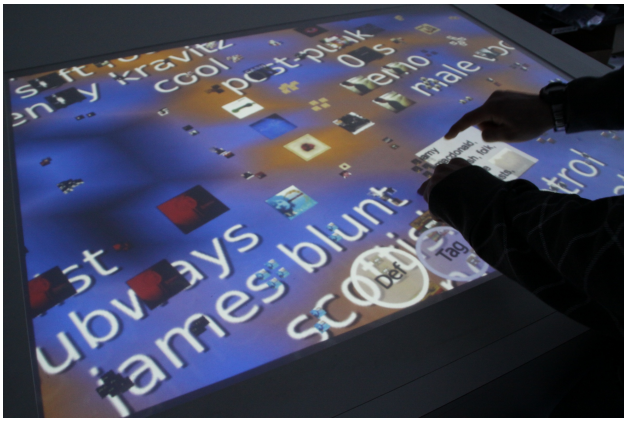


Figure 1. Browsing a music collection through its lyrics on a tabletop

In this paper, we give an overview of SongWords (see figure 1 and video [1]), an application for tabletop computers which supports navigating music collections and investigating correlations based on the lyrics of songs. We present related research on browsing and tabletop interfaces, describe and explain our interface and interaction design decisions, talk about the implementation of SongWords and present the results of a user study.

2. RELATED WORK

Content-based MIR often uses not only the instrumental but also the vocal parts of a song. However, since extracting the words of a song directly from the audio signal has proven to be difficult, a common approach is to gather lyrics from the internet based on available metadata (e.g., [14]). These lyrics then enable tasks that go beyond pure retrieval, such as semantic or morphologic analysis (topic detection [13], rhymes in hip hop lyrics [9], genre classification from rhyme and style features [16]). Other work is approaching the problem of mapping textual lyrics to an audio signal ([12], [7]). Combining an ontology with lyrics enables even more sophisticated tasks: Baumann et al. used natural language processing and mapped text to a vector space model to calculate a lyrical similarity value for pairs of songs [1]. Fujihara et al. presented an approach for creating bi-directional hyperlinks between words in songs that could be applied not only to textual lyrics but also to the actual audio data [6]. They

[1] *http://www.youtube.com/watch?v=FuNPhN6zyRw*

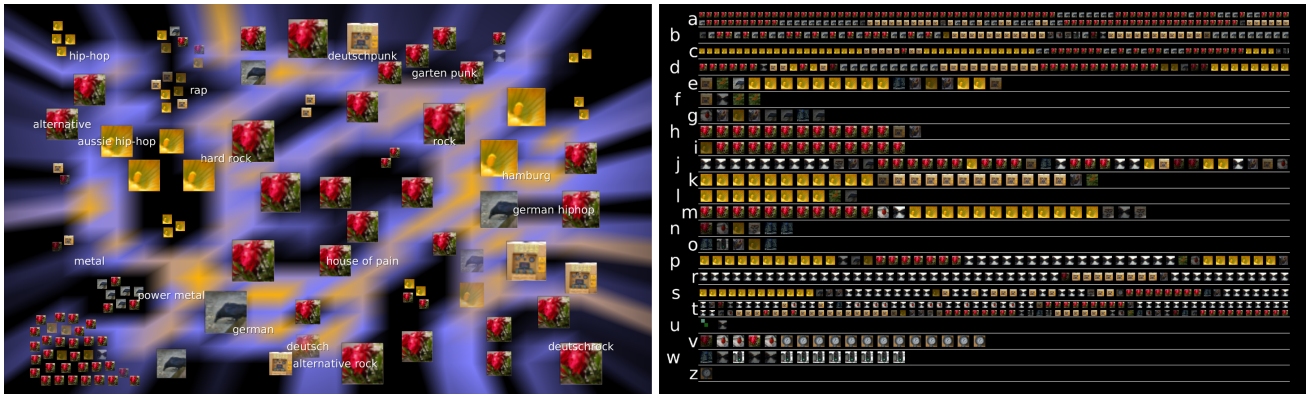


Figure 2. Songs are organized on a map based on lyrics or tags (left), or sorted alphabetically by their artist's name (right)

also describe an application called *LyricSynchronizer* [8] that allows browsing collections by navigating through the aligned song lyrics. There is, however, no work on visualizing a complete music collection based on lyrics.

In order to make complex music collections accessible, a multitude of browsing interfaces are available. Beyond the sorted lists commonly used in media player software, abstraction and filtering capabilities are useful, e.g., by applying techniques from information visualization [23] or by providing views based on different facets [2]. Since music content provides a very high-dimensional data set, complexity also has to be reduced for visualization. Pampalk's Islands of Music [17] is the best known example for this approach. It has also been extended to incorporate multiple views on different acoustic aspects [18]. Self-organizing maps have also widely been used for visualizing text documents (e.g., [5]). In a similar vein, several projects allow browsing a music collection on tabletop displays using self-organizing maps of different low- and high-level audio features (SongExplorer [11], MarGrid [10], DJen [3], MusicTable [22]). Lyrics, however, haven't been used for browsing so far.

3. INTERFACE DESIGN

When designing SongWords we started from two user tasks: First, users should be able to easily browse and search through their personal collections based on lyrics. SongWords should give them a new perspective on their own songs and let them browse through the collection from word to word (similar to [7]). Second, we wanted to allow users to corroborate or disprove hypotheses about connections between lyrics and genres. It should be easy to discover correlations between different genres and words, such as *"Hip hop lyrics often use cuss words"* or *"Pop songs often revolve around 'love' and 'baby'"*.

Since such patterns are hard to discover by scrolling through a text-based list, we decided to map the high-dimensional information space to a two-dimensional canvas using Self-Organizing Maps [15]. Furthermore, as the resulting map at a reasonable level of detail largely exceeded the screen size, we also implemented a Zoomable User Interface to navigate the large virtual canvas on a physical display. With a potentially very large number of items, we finally chose to use an interactive tabletop display for its advantages regarding screen space [24] and its potential for multi-user interaction. In addition, zooming and panning was found to work better using direct touch and bi-manual interaction than using mouse input [4].

3.1 Visualization and Interaction

SongWords analyzes a given music collection and displays it on a two-dimensional canvas. The visualization consists of two self-organizing maps for lyrics and for tags, as well as an alphabetical list by artist's names for direct access to songs (see figure 2). In addition, there is a view for the results of text searches (see below). The user can switch between these different views by pressing one of a number of buttons at the border of the screen.

All songs of the collection are represented on the virtual canvas by their cover art. To optimize the use of screen space, each item is displayed as large as possible without overlapping with other songs. The underlying self-organizing map guarantees spatial proximity between similar items regarding the currently chosen aspect (lyrics or tags). The map contains black areas in the background that connect clusters of items and displays the most relevant words or tags next to the song items to give overview and allow orientation. A common interaction that is possible in each context is pan and zoom (see figure 3). Panning is triggered by putting the finger to the canvas outside of a song icon and dragging, with the canvas sticking to the finger. Zooming and rotation are controlled by two or more fingers and the system calculates the geometric transformation of the canvas from their movements.

In addition to this geometric zoom for the virtual canvas, SongWords also implements a semantic zoom for song icons (see figure 4): At the least detailed zoom level, songs are represented as colored squares to reduce screen clutter with thousands of items. The item's colors represent the home collection of the song when several collections are available. When zooming in, the solid colors are replaced by the artwork of the corresponding record. By zooming further in (or tapping once on the song icon) the artist, title and lyrics of the song become available. Here, the user

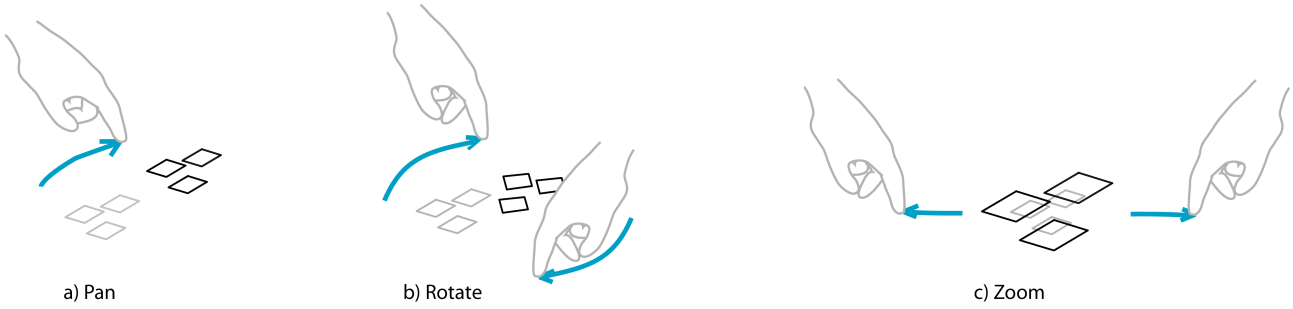


Figure 3. Uni- and bi-manual gestures for panning, rotation and zoom

can scroll through the text if the screen space does not suffice, mark sections of it and search for these sections in the collection. Despite SongWords' focus on text, we decided against using an on-screen keyboard for text search. Not only would it have taken up screen space (or required an explicit mode switch) and suffered from typical problems of virtual keyboards such as missing tactile feedback, it would also have allowed erroneous search terms. Search results in turn are displayed on a spiral based on their relevance (see below). If multiple song collections are visible (e.g., from different users), each song icon has a colored border that represents its home collection.

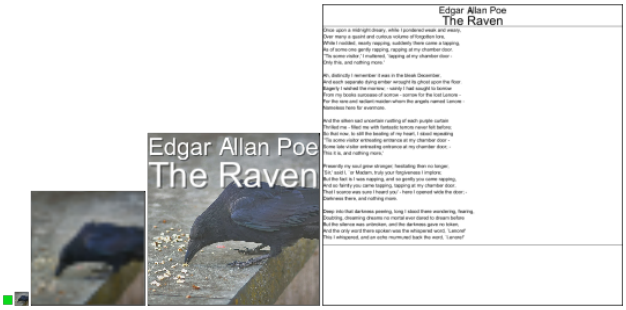


Figure 4. Different semantic zoom levels for song icons

Touching a song with two fingers displays the artist, title and tags for this song. Touching a section of the map with one finger displays the relevant tags or words from the lyrics. Songs can be played by holding a finger on their icon for a short time. In order to allow the discovery of new music based on lyrics, songs with similar lyrics are retrieved from the internet and displayed alongside the songs from the collection. They are rendered slightly transparent in order to distinguish them from local songs. If the user wants to listen to them, a thirty-second sample is downloaded and played.

One challenge in designing for tabletop displays is the so-called orientation problem. While PC screens have a fixed 'up' direction, users can interact with a tabletop computer from any side. The straightforward two-finger rotation of SongWords prevents problems of readability (for a single user) and lets the user quickly change position. When the canvas' orientation changes, the view buttons at the bottom of the screen move along to always remain at the bottom and thus well reachable.

Figure 5. Search results are arranged on a spiral based on their relevance

3.2 User tasks

SongWords enables several distinct user tasks from simple browsing to gaining novel insight and testing hypotheses. By their working principle, the self-organizing maps visualize the similarity between different songs regarding either lyrics or tags. While user-generated keywords can be expected to be relatively consistent for one artist, the lyrics map can bear greater surprises: When songs by one artist are spread widely across the map, this means that this artist produces very diverse lyrics (or employs different songwriters). Similarly, the (in)consistency between songs from different collections can also be seen from their colored borders: If each color is dominant in a different corner of the map, the overlap between the lyrics of the collections is not very high. Discovery of new music based on lyrics is supported in SongWords, as the lyrics and preview clips of related but unknown songs are automatically downloaded and added to fill the map.

The user can navigate from song to song using the text search. By selecting a portion of the lyrics and double-tapping, the system switches to the search view, in which

all songs containing this sequence of words are arranged by relevance. To use the two-dimensional space most efficiently, the linear result list is furled into a spiral (see figure 5). Thereby, the user can quickly find all songs that contain a favorite text section.

To keep track of one or more songs across different views, they can be selected. Songs are selected by pressing the select-button and drawing one or more arbitrary polygons around them. This causes all contained songs to be highlighted with an overlay color, and when switching from view to view their movement can thus be followed effortlessly.

SongWords' different views also allow the user to verify hypotheses. To confirm the hypothesis *Hip hop lyrics often use cuss words* a possible workflow is to switch to the lyrics view, select songs that gather around cuss words, then switch to the tag-view (where the genre of a song is usually the most prominent tag) and see how many songs appear near to the hip hop area. Similarly, other hypotheses regarding the lyrics of one artist (selected from the alphabetical view) or songs containing a certain word can be examined.

4. IMPLEMENTATION

The SongWords prototype has been implemented in C++ for Windows XP. We use the OpenGL framework for rendering and the BASS library for audio playback. SongWords was deployed on a custom-built FTIR tabletop display (99 x 74 cm, 1024 x 768 resolution) and multi-touch input is handled by the Touchlib library. We rely on the internet for all information besides the actual artist, title and album information of a song: After extracting this information from the MP3's ID3 tag, we use the search capabilities of various internet lyrics databases (e.g., lyrics.wikia.com) and parse the resulting HTML pages (similar to, for example, [14]) to retrieve the lyrics. Album covers and user-generated tags are accessed through the API of Last.FM. In order to find the most representative words for a song, we filter the text for stop words and afterwards perform a term frequency inverse document frequency (TF/IDF) analysis [21] to find discriminative terms. The resulting word lists are stemmed with Porter's stemming algorithm [20], to merge related words. These stemmed words are also shown on the map, and in some cases look misspelled. The list of all discriminative stemmed terms forms the feature vector which is used for computing the self-organizing map [15].

For creating the self-organizing map of tags, each song item again receives a feature vector consisting of the tags and their popularity. Our self-organizing maps are based on the classical machine learning approach by Kohonen [15] with one tweak: Since we wanted to make sure that items do not appear at completely different positions in the lyrics and tag views of SongWords, we don't initialize the learning phase of the tag map with random values, but with the results from the lyrics map. Therefore, the chances that identical items appear at similar positions on the two maps are much higher without disturbing the dimensionality reduction capabilities. We also use a relatively low number of 400 iterations for training in order to generate the visualization sufficiently fast.

To allow discovery and fill the map with relevant related music, for every five songs in the collection a random artist is picked, and a related artist is acquired from Last.FM's API beforehand (related artists are calculated based on collaborative filtering of their massive user base). For this artist, a search on Amazon.com is performed and for each resulting song an audio sample is downloaded. SongWords then tries to find lyrics on the aforementioned online databases and once it succeeds, the complete song is added to the training set of the map.

5. USER STUDY

After implementing SongWords, we evaluated it in order to verify whether it properly supported the tasks for which it was designed. As evaluating a complete visualization system is difficult and an active field of research [19], we decided to rely on qualitative user feedback and a small number of participants.

5.1 Study Design

The main objectives of the study were to check usability of the application and identify possible design flaws that could prevent the user from fulfilling the two main tasks. In addition, we wanted to test the application under realistic conditions and therefore asked participants to select a sample of roughly thousand songs from their personal collections. For this set of songs we gathered covers and lyrics before the study and presented the participants with them. As a third aspect of the evaluation we wanted to verify the choice of using a tabletop display compared to a desktop PC. Therefore, we ported SongWords to a PC and mapped the input to the mouse: A left mouse-click was used for panning and selection, a right click for displaying contextual information and the scroll wheel for zooming.

5.2 Study Tasks

The participants were asked to fulfill tasks of increasing complexity to find potential shortcomings of the user interface. Basic tasks were "Play three different songs" or "Choose a song and find out the last line of the lyrics" which could be easily completed using basic interaction techniques. The more complex compound tasks required participants to combine multiple aspects of the application: "Find your favorite song, pick a word from the text that you regard as special and find other songs that contain it" and "Find words that are typical for a certain genre" were two of them. For each task, we let our participants play around with SongWords without previous instructions in order to gather spontaneous feedback and see how self-explanatory the interaction techniques were. If users weren't able to finish the task on their own the experimenter explained how to do it after a few minutes. Initially, all participants worked on and explored the desktop version of SongWords. After they had fulfilled all tasks, they moved on to the

Tabletop version and completed the same tasks again to find the differences between the two setups. We were only interested in the differences they perceived between the two conditions interaction-wise and not quantitative data like the required time, so using the same tasks and a predefined order of desktop and tabletop led to no problems.

Finally, we also wanted to examine the influence of potential multi-user interaction on SongWords: Therefore, we matched our participants to pairs after each of them had worked on their own, displayed their two collections at the same time with color coding and presented them with multi-user tasks. Exemplary tasks were "Find genres that appear in both collections" and "Find the song from the other's collection that you like best". We thereby wanted to identify potential problems in coordinating the access to the interface and in collaboration between the pairs. In the end, the participants were asked to fill out a questionnaire that collected demographic information and their opinions for the desktop and tabletop version of SongWords.

5.3 Participants

We recruited six participants from the undergraduate and graduate students of the University of *[Removed for anonymous submission]* (age: 24-30 years, one female). We supplied them with a small Java application beforehand that allowed them to conveniently choose a thousand songs from their collections, retrieved required meta-data and sent the results to us via e-mail. Only one of the participants was recruited on short notice and was not able to provide us with his song set, so we used a different participant's collection and adapted the tasks accordingly.

5.4 Results

Using both the tabletop version and the desktop version of SongWords showed that the concepts work similarly well on both platforms. Also, the participants mostly were able to transfer their knowledge from one to the other. One notable exception was "hovering" by using two fingers. None of the users figured this out by themselves. We also observed an artifact from desktop computing: Participants kept trying to double click for starting song playback. The change from a map view to the results view after searching often went by unnoticed as the song's text filled the screen and occluded the switch. Additionally, none of the participants actually discovered new music, as the slight transparency of the suggested items obviously wasn't enough to make them stand out. Most of these flaws can easily be fixed. Besides them, we didn't discover any major usability problems.

Finally, we also observed the participants while interacting in pairs with SongWords. Observations were that the color-coding of items from collections worked, even though clear textual or tag distinctions between the collections were not visible. Also as expected from previous research on tabletop displays, participants quickly began taking turns when interacting with the system in order not to get tangled up.

6. DISCUSSION AND LIMITATIONS

One principal limitation of this approach is the fact that it doesn't apply to purely instrumental music. As discussed in the introduction, this is not a very strong limitation for contemporary popular music, but entirely excludes much of the classical music genre, for example. One of the major challenges in working with actual music collections is their size. The hardware-accelerated graphics of SongWords currently produce interactive frame rates for collections of several thousands of songs, but for a practical deployment there are other restrictions: Gathering the lyrics for a song takes several seconds and has to be performed sequentially (in order not to flood the web servers with requests). The time for training the self-organizing map grows linearly with the number of songs and has to happen twice (once for the lyrics, once for the tags) when the program first reads a new collection. Fortunately, the map can be incrementally updated when new songs are added.

The text analysis capabilities of SongWords are currently limited to the most discriminative terms from each song. These most important words can be seen in the maps at first glance and spatial organization is correctly based on these statistical relationships. As the analysis uses the TF/IDF approach [21], it works especially well when the collection is quite dissimilar regarding the words to produce clear distinctions. Subtler differences will go unnoticed, and would require more sophisticated methods from Natural Language Processing.

7. CONCLUSION AND FUTURE WORK

We have presented SongWords, a user interface for browsing music collections based on their lyrics. The visualization using self-organizing maps, combined in a zoomable user interfaces with interactions for searching, marking and reordering, allows a new perspective on music collections. In particular, we observed that users were able to explore correlations between fragments of the lyrics and genre or other user-generated tags. These correlations would be impossible to discover with current list-based interfaces or visualizations purely based on audio data analysis.

In an evaluation we identified a number of minor design flaws of our current prototype, which we will fix in a future version. We will also explore more sophisticated natural language processing and visualization methods, for example involving synonyms and hierarchical clusters of similarity in order to create an even more meaningful similarity measure on lyrics.

8. REFERENCES

[1] S. Baumann and A. Klüter. Super Convenience for Non-Musicians: Querying MP3 and the Semantic Web. In *Proceedings of the International Conference on Music Information Retrieval*, pages 297–298. ISMIR, 2002.

[2] R. Dachselt and M. Frisch. Mambo: a facet-based zoomable music browser. In *Proceedings of the 6th in-*

ternational conference on Mobile and ubiquitous multimedia, pages 110–117. ACM, 2007.

[3] D. Diakopoulos, O. Vallis, J. Hochenbaum, J. Murphy, and A. Kapur. 21st Century electronica: MIR techniques for classification and performance. In *Proc. 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 465–469. ISMIR, 2009.

[4] Clifton Forlines, Daniel Wigdor, Chia Shen, and Ravin Balakrishnan. Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*, pages 647–656, New York, New York, USA, 2007. ACM Press.

[5] B. Fortuna, M. Grobelnik, and D. Mladenić. Visualization of text document corpus. In *Informatica*, volume 29, pages 497–502, 2005.

[6] H. Fujihara, M. Goto, and J. Ogata. Hyperlinking Lyrics: A Method for Creating Hyperlinks Between Phrases in Song Lyrics. In *Proc. 9th International Society for Music Information Retrieval Conference (ISMIR '08)*, pages 281–286. ISMIR, 2008.

[7] H Fujihara, M Goto, J Ogata, K Komatani, T Ogata, and H G Okuno. Automatic synchronization between lyrics and music CD recordings based on Viterbi alignment of segregated vocal signals. In *Proceedings of the Eighth IEEE International Symposium on Multimedia (ISM'06)*, pages 257–264, 2006.

[8] Masataka Goto. Active Music Listening Interfaces Based on Signal Processing. In *IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 2007. IEEE, April 2007.

[9] H. Hirjee and D.G. Brown. Automatic Detection of Internal and Imperfect Rhymes in Rap Lyrics. In *Proc. 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 711–716. ISMIR, 2009.

[10] S Hitchner, J Murdoch, and G Tzanetakis. Music Browsing Using a Tabletop Display. In *Proc. 8th International Conference on Music Information Retrieval (ISMIR'07)*, pages 175–176. ISMIR, 2007.

[11] Carles F. Julià and Sergi Jordà. Songexplorer: a tabletop application for exploring large collections of songs. In *Proc. 10th International Society for Music Information Retrieval Conference (ISMIR '09)*, pages 675–680. ISMIR, 2009.

[12] M.Y. Kan, Y. Wang, D. Iskandar, T.L. Nwe, and A. Shenoy. LyricaAlly: Automatic synchronization of textual lyrics to acoustic music signals. *IEEE Transactions on Audio Speech and Language Processing*, 16(2):338–349, 2008.

[13] F. Kleedorfer, P. Knees, and T. Pohle. Oh oh oh whoah! towards automatic topic detection in song lyrics. In *Proc. 9th International Society for Music Information Retrieval Conference (ISMIR '08)*, pages 287–292. ISMIR, 2008.

[14] P. Knees, M. Schedl, and G. Widmer. Multiple lyrics alignment: Automatic retrieval of song lyrics. In *Proceedings of 6th international conference on music information retrieval (ISMIR 05)*, pages 564–569. ISMIR, 2005.

[15] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[16] R. Mayer, R. Neumayer, and A. Rauber. Rhyme and style features for musical genre classification by song lyrics. In *Proc. 9th International Society for Music Information Retrieval Conference (ISMIR '08)*, pages 337–342. ISMIR, 2008.

[17] E. Pampalk. Islands of music: Analysis, organization, and visualization of music archives. *Journal of the Austrian Soc. for Artificial Intelligence*, 22(4):20–23, 2003.

[18] E. Pampalk, S. Dixon, and G. Widmer. Exploring music collections by browsing different views. *Computer Music Journal*, 28(2):49–62, Juni 2004.

[19] C. Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, pages 109–116. ACM, 2004.

[20] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[21] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[22] Ian Stavness, Jennifer Gluck, Leah Vilhan, and Sidney Fels. The MUSICtable: A Map-Based Ubiquitous System for Social Interaction with a Digital Music Collection. *Lecture Notes in Computer Science*, 3711/2005(Entertainment Computing - ICEC 2005):291–302, Juni 2005.

[23] M. Torrens, P. Hertzog, and J.L. Arcos. Visualizing and exploring personal music libraries. In *Proc. 5th International Conference on Music Information Retrieval (ISMIR '04)*, pages 421–424. ISMIR, 2004.

[24] Stephen Voida, Matthew Tobiasz, Julie Stromer, Petra Isenberg, and Sheelagh Carpendale. Getting Practical with Interactive Tabletop Displays: Designing for Dense Data, Fat Fingers, Diverse Interactions, and Face-to-Face Collaboration. In *Proc. ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 109–116. ACM, 2009.

STRING QUARTET CLASSIFICATION WITH MONOPHONIC MODELS

Ruben Hillewaere and **Bernard Manderick**
Computational Modeling Lab
Department of Computing
Vrije Universiteit Brussel
Brussels, Belgium
{rhillewa,bmanderi}@vub.ac.be

Darrell Conklin
Department of Computer Science and AI
Universidad del País Vasco
San Sebastián, Spain
IKERBASQUE, Basque Foundation or Science
Bilbao, Spain
darrell_conklin@ehu.es

ABSTRACT

Polyphonic music classification remains a very challenging area in the field of music information retrieval. In this study, we explore the performance of monophonic models on single parts that are extracted from the polyphony. The presented method is specifically designed for the case of voiced polyphony, but can be extended to any type of music with multiple parts. On a dataset of 207 Haydn and Mozart string quartet movements, global feature models with standard machine learning classifiers are compared with a monophonic n-gram model for the task of composer recognition. Global features emerging from feature selection are presented, and future guidelines for the research of polyphonic music are outlined.

1. INTRODUCTION

In the field of music information retrieval, much research has been done in symbolic music genre classification, where a model has to assign an unseen score to a certain class, for example style, period, composer or region of origin. There are two main categories of models that have been widely investigated: *global feature models* and *n-gram models*. Global feature models express every piece as a feature vector and use standard machine learning classifiers, whereas n-gram models rely on sequential event features.

In a recent paper [10] the results of a thorough comparison of these types of models are reported for the task of classifying folk songs based on their region of origin on a large monophonic data set. That study demonstrates that the n-gram models are always outperforming the global feature models for this classification task. It is an interesting question whether this result still holds in a polyphonic setting.

In the literature, it appears that most research has been investigating classification or characterization of melodies (monophonic) [5, 14, 16], but only few efforts have been made to develop polyphonic models. In [15], orchestrated polyphonic MIDI files are classified using global features, including some features about musical texture and chords. A set of polyphonic features based on counterpoint properties was developed by [19], and applied to the task of composer classification. They find that the distinction between Haydn and Mozart string quartets, which is very interesting from a musicological point of view, is a hard classification task.

When considering polyphonic music, it is essential to qualify the form of input. Two formats can be considered:

voiced: a fixed and persistent number of parts; and,

unvoiced: free polyphony that is not available in, or cannot be easily divided into parts.

A typical example of voiced polyphony is a string quartet, consisting of 4 well-defined voices (Violin 1, Violin 2, Viola and Cello). Unvoiced polyphony is common, for example, in piano music.

Another way to view this dichotomy of polyphonic music is in terms of a MIDI file type: voiced (type 1), or unvoiced (type 0), realizing of course the grey area where tracks within a type 1 file may contain internal polyphony, and where type 0 files identify voices by use of channel numbers.

This paper investigates how monophonic global feature and n-gram models perform on the classification of Haydn and Mozart string quartets in their original voiced format. The voiced structure is exploited since these monophonic models are applied to separate voices. The initial database used in [19] containing 107 string quartet movements was extended to a total of 207 movements in order to measure statistically more relevant differences.

Two tentative hypotheses from previous work [11] are being verified in this paper:

1. n-gram models also perform better than global feature models on monophonic parts extracted from the polyphonic texture.

2. the first violin is the most distinctive voice of the string quartets.

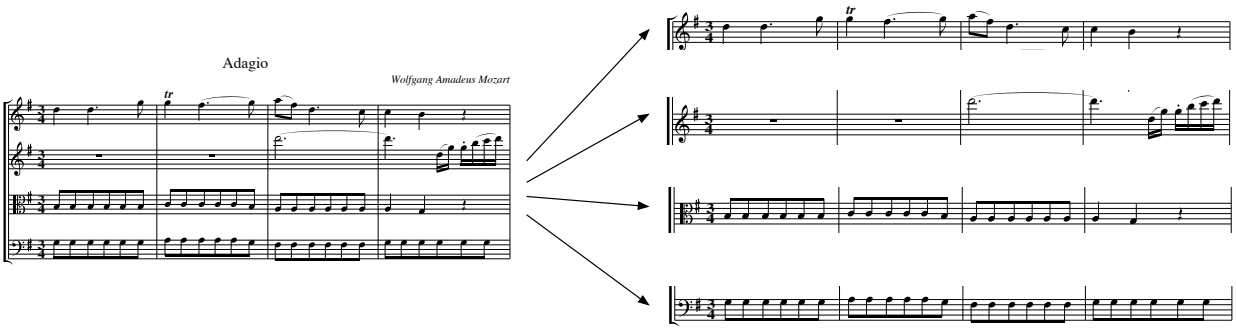


Figure 1. The voice separation of the string quartets into Violin 1, Violin 2, Viola and Cello.

For the global feature models, special care has been taken to apply feature selection within the inner loop of the cross validation scheme, in order to avoid overoptimistic evaluation estimates [8, 17]. A similar procedure has been set up to tune the parameters of the learning algorithms during the training phase. Features that emerge recurrently in the feature selection process are highlighted.

The remainder of this paper is structured as follows. We start by introducing the dataset and the music representations, global feature and n-gram models, and the classification methodology in the next section. Then, we give the results of the described models on the Haydn/Mozart dataset. We end with a discussion and some directions for future work.

2. METHODS

In this section we describe the dataset used for our experiments and we will give a short overview of both global feature and n-gram models. Furthermore, we introduce our classification methodology outlining the cross validation setup combined with supervised feature selection and SVM parameter tuning.

2.1 Dataset and music representation

The Haydn/Mozart dataset is composed of 112 string quartet movements from Haydn and 95 string quartet movements from Mozart, including most of the pieces from the dataset used in [19], but extending it as much as possible to nearly double its size. We chose to focus on the period 1770-1790 in which both composers were active, discarding early or late Haydn quartets which might be easy to classify. In order to maximize the number of quartet movements from Mozart, we included 8 movements from two flute quartets (K.285, K.298) and one oboe quartet (K.370), which are written for flute/oboe, violin, viola and cello, and thereby very similar to the string quartets. The original scores in **kern format were found on the website of the Center for Computer Assisted Research in the Humanities at Stanford University [1]. We transformed these to clean MIDI files, ensuring that the four voices appear on separate tracks and that all barlines are correctly synchronized in all voices by correcting several errors in note durations. We retained only the highest note of double stops, thereby reducing each voice to a purely monophonic sequence of notes. To enable the use of monophonic classification techniques, we created four monophonic data sets called Violin 1, Violin 2, Viola and Cello by isolating each voice of every string quartet movement, as illustrated in Figure 1.

2.2 Global feature models

In this section we introduce global features and the corresponding global feature models. A global feature summarizes information about a whole piece into a single attribute, which can be a real, nominal or boolean value, for example “average note duration”, “meter” or “major/minor”. With a set of global features, pieces can be simply re-expressed as feature vectors, and a wide range of standard machine learning algorithms can then be applied to evaluate the feature set.

Voiced polyphony presents the advantage of having a fixed number of monophonic parts, which enables us to isolate these parts and apply monophonic models. In this paper three global feature sets are used to represent the monophonic parts. These features describe melody characteristics, mainly derived from pitch or duration, whereby we mean that at least one pitch or duration value is inspected for the feature computation.

The global feature sets chosen are the following :

- The *Alicante* set of 28 global features, designed by P.J. Ponce de Léon and J.M. Iñesta in [16]. This set was applied to classification of MIDI tunes in jazz, classical, and pop genres. From the full set, we implemented the top 12 features that they selected for their experiments.

- The *Jesser* set, containing 39 statistics proposed by B. Jesser [13]. Most of these are derived from pitch, since they are basic relative interval counts, such as “amajsecond”, measuring the fraction of melodic intervals that are ascending major seconds. Similar features are constructed for all ascending and descending intervals in the range of the octave.

- The *McKay* set of 101 global features [15], which were used in the winning 2005 MIREX symbolic genre classification experiment and computed with McKay's software package jSymbolic [2]. These features were developed for the classification of orchestrated polyphonic MIDI files, therefore many features, for example those based on dynamics, instrumentation, or glissando, were superfluous for this analysis of monophonic single voices and we were able to reduce the set down to 61 features.

These global feature sets do not show many overlapping features, only some very basic ones occur in maximum two of the sets, such as the "pitch range". Therefore it is interesting to join the three feature sets to form the *Joined* set, which means every piece is represented as a data point in a 112-dimensional feature space. We are interested in finding out which features are relevant for this specific task of composer classification, therefore we will apply feature selection on this Joined set.

2.3 n-gram models

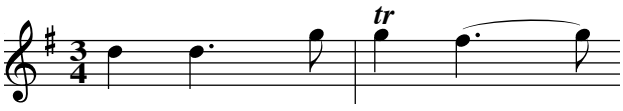
In this section we introduce n-gram models and how they can be used for classification of music pieces using event features. n-gram models capture the statistical regularities of a class by modeling the probability of an event given its preceding context and computing the probability of a piece as a product of event probabilities. This technique is particularly well-known for language modeling, a word in language being roughly analogous to an event in music. The context $\overline{e_{i-1}} = [e_1, e_2, \ldots, e_{i-1}]$ of an event e_i is usually limited to a short suffix $[e_{i-n+1}, \ldots, e_{i-1}]$, meaning the probability of the current event only depends on the $n-1$ previous events. The n-gram counts of the training data are used to estimate the conditional event probabilities $p(e_i \,|\, \overline{e_{i-1}})$, and the probability of a new piece $\overline{e_\ell}$ is obtained by computing the joint probability of the individual events in the piece:

$$p(\overline{e_\ell}) = \prod_{i=1}^{\ell} p(e_i \,|\, \overline{e_{i-1}}) \qquad (1)$$

To use an n-gram model for music classification, for each class a separate model is built, and a new piece is then simply assigned to the class with the highest piece probability.

For monophonic music, n-gram models and more powerful extensions are naturally applicable [6, 10], but polyphonic music needs first to be converted into a sequential form. One way to do this is to simply extract a voice (e.g., Violin 1) from the polyphonic texture.

To reduce the sparseness of the n-gram counts, we do not model the pitch or duration directly, but we first transform the music events by means of event features. An event feature assigns a feature-value to every event, in our case to every note in the music piece. The chosen event feature determines the level of abstraction of the data representation. The event feature we will use is the melodic interval. Models are constructed for a class by extracting the same voice (e.g., Violin 1) for every piece in a corpus, and viewing that piece as a sequence of melodic intervals.



pitch	74	74	79	79	78	79
melodic interval	⊥	0	+5	0	-1	+1
melodic contour	⊥	r	u	r	d	u
duration ratio	⊥	3/2	1/3	2	3/2	1/3
average pitch	77.1667					
rel. freq. m2	0.4					
Huron contour	ascending					

Figure 2. First measures of the first violon of the Adagio K.080 of W.A. Mozart, illustrating the contrast between global features (lower three) and event features (upper four).

Figure 2 illustrates the difference between global features and event features on an excerpt of the first violon of the Adagio K.080 of W.A. Mozart. A global feature describes a constant property of the whole piece, whereas an event feature is associated to one particular note. A global feature summarizes the data much more, but one uses a whole collection of global features to build a global feature model, whereas n-gram models are constructed using one single event feature.

2.4 Classification methodology

In this paper, two fundamentally different types of models are applied to the task of composer classification. In order to present any comparative results, we have to find a common way of evaluating the performance of these models. It is common practice to set up a cross validation scheme to obtain classification accuracies that generalize reasonably well.

Our data set is very small from a general machine learning point of view, only 207 samples, it is therefore preferable to do *leave-one-out cross validation*, where one uses as much training data as possible to train the model, discarding only one instance for testing purposes. For both global feature and n-gram models, a leave-one-out cross validation scheme was implemented.

Since global features represent every instance as a multidimensional feature vector, any standard machine learning classifier can be applied to get a performance accuracy. Simple classifiers such as Naive Bayes and k-nearest neighbours can give us a good indication, but in this work we opt for the more sophisticated Support Vector Machine, shortly SVM, which has been proven to be a state-of-the-art classifier [7]. An SVM makes use of a so-called kernel function to determine non-linear decision boundaries between classes, and a well-known kernel function, the

RBF-kernel, is used in this paper [4]. In this setting, an SVM has two parameters that need to be trained. The first is related to the softness of the decision margin, expressing the tradeoff between generality and classification accuracy, commonly denoted as C. The second is a parameter σ specific to the RBF kernel function. In practice, these parameters can simply be tuned by doing a "grid-search" over a large search-space of pairs (C, σ) as described in [12].

Another common machine learning technique is *feature selection*, which is often used to reduce the dimensionality of the data or to discover which features are highly correlated with the target class. In principle, feature selection is decreasing the size of the hypothesis space, which leads to a faster and more effective search for the learning algorithms and tends to avoid overfitting. Therefore, it has led to improved classification accuracies in some cases, or to a compact feature set that describes the data in a more interpretable, summarized way.

However, there is a subtlety in both feature selection and SVM parameter optimization, a pitfall to avoid when one uses *supervised* learning methods in combination with a cross validation scheme [8, 17]. In the simple case where a separate training and test set are given, one has to apply supervised preprocessing methods followed by the learning algorithm on the training set only, before testing the resulting model on the test set. Expanding this to a cross validation scheme, this means one must take care to apply these methods within the inner cross validation loop. As pointed out by [17], it is a common mistake to use both training and test set for supervised feature selection, which leads to overoptimistic and exaggerated performance results.

In this paper, SVM* denotes the model in which parameter tuning with a grid search has been done within the inner loop of the cross validation scheme. Feature selection is also implemented taking this special consideration.

3. RESULTS

In this section we describe the experimental results for the classification of the Haydn and Mozart string quartet movements. As a baseline, we keep in mind that the classes are quite equally distributed (112 Haydn, 95 Mozart), which means that 54.1% classification accuracy can be achieved by always predicting Haydn.

To evaluate the global feature approach, the SVM* classifier method is applied to the Joined set. As described above, this includes an SVM parameter tuning by doing a grid search within each loop of the leave-one-out cross validation. Furthermore, a supervised feature selection method called Correlation-based Feature Selection (CFS) is also applied. CFS is a filter method aiming to find a subset of features that are highly correlated with the class but have few intercorrelation among them [9]. The implementation of SVM* and the CFS make use of the Weka machine learning toolbox [3, 20].

For the n-gram model, we use a simple trigram model of the melodic intervals. For each Haydn and Mozart a separate model is built on the training set and a test piece is assigned to the class of which the model generates it with the highest probability according to Equation 1. A global classification accuracy is also computed with leave-one-out cross validation. The results for both the global feature models and the trigram models on the separate voices are reported in Table 1.

Voices	SVM*	SVM*+feat.sel.	3-grams
Violin 1	74.4	73.4	63.8
Violin 2	66.2	66.2	61.4
Viola	62.8	57.0	61.4
Cello	65.7	59.4	**75.4**

Table 1. The l.o.o. classification accuracies of the Joined global feature set and the trigram model on the separate voices extracted from the voiced string quartet movements.

It appears immediately that the results of previous work done on a smaller database of 107 pieces do not hold up [11]. Previously, we noticed a consistent tendency for n-gram models to perform better than global feature models regardless of the voice. Now we observe that the n-gram models perform well on the Cello dataset with an accuracy of 75.4%, but poorly on the other voices, whereas the global feature models achieve an almost equally high accuracy of 74.4% on the Violin 1. Our second hypothesis, about the first violin being the most predictive one for a composer, is also weakened because of this surprising result with n-gram models on the Cello. However, the global feature result on Violin 1 is still an indication of its predictive value. Additional computation of global feature models on the separate Alicante, Jesser and McKay sets confirm this indication, and show that we can order the voices according to there predictiveness with global feature models as follows: Violin 1, Cello, Violin 2 and Viola.

The second column of Table 1 is showing the results of the SVM* with CFS feature selection. These classification accuracies are slightly lower than without applying feature selection, which confirms that supervised feature selection does not necessarily lead to an improvement when it is applied in the inner loop of the cross validation scheme. Nevertheless, it is interesting for musicological reasons to see which features emerge in the selected feature subsets for each voice. Below we give three short examples of features that are selected in one or more voices.

- "dmajsec", i.e. the fraction of melodic intervals that are descending major seconds, is selected for both Violin 1 and Violin 2. Looking at the relative frequencies of this feature, it appears that Mozart uses more descending major seconds than Haydn for the two violins.

- "shortestlength" emerges in both the Violin 2 and the Viola. This is the shortest duration such that all durations are a multiple of this shortest duration (except for triplets). Again by looking at the relative distributions, one notices that Mozart tends to use smaller shortest lengths in these voices.

- "ImportanceMiddleRegister" is one of the features selected for the Cello. This denotes simply the fraction of notes with MIDI pitches between 55 and 72, which is roughly the upper range of the instrument. It seems that Haydn uses the cello more often in this range than Mozart in these string quartets.

4. CONCLUSIONS AND FUTURE WORK

This paper has applied monophonic classification models to the task of composer recognition in voiced polyphonic music, specifically Haydn and Mozart string quartets written in the period 1770-1790. An earlier dataset of string quartet movements is extended to a total of 207 pieces to obtain more statistical significance. The voiced structure is exploited by extracting the separate voices to enable the use of monophonic models. Several conclusions emerge: that a simple trigram model of melodic interval performs very well on the cello, achieving the best classification accuracy of 75.4%, but is outperformed by the global feature models on the other voices. Therefore, we are also unable to conclude that the first violin is indeed the most predictive voice for a composer, even though the results on Violin 1 are consistently best with the global feature approaches.

At first sight, these observations are rather disappointing, but they confirm the necessity of having a sufficiently large dataset before making any claims. Learning algorithms in symbolic music have to cope with this kind of challenge, what shows there is still room for improvement on a machine learning level.

Currently, we are investigating what causes this remarkable result with the trigram model on the cello and the low accuracy on the first violin, by looking carefully which pieces are correctly classified by one method and not by another, or correctly by both. Perhaps there is a core part of this dataset that is 'easy' to classify, or else we might consider using an ensemble model where one combines the global feature models and the n-gram models in order to improve the overall accuracies. One could also wonder how the so-called Haydn Quartets, six quartets written by Mozart but famously inspired by and dedicated to Haydn, influence these results. So far we have only found an indication that these particular movements are slightly harder to recognize, this topic will be part of further research.

Further future work will address the issue of polyphonic music in different ways. Figure 3 illustrates the global structure of these future directions. As we detailed earlier in this paper, polyphonic music can be voiced, like the string quartets used for this study, or unvoiced, for example piano sonatas. Each of these types of polyphony can be modelled by monophonic or polyphonic models. The models from this work were monophonic models, which are situated in the outer left branch of the tree. Polyphonic models for voiced polyphony can for example be based on polyphonic global features taking into account voice information or harmonic global features, such as those used in [15, 19]. To apply monophonic models to unvoiced polyphonic music, one has to apply some voice extraction algorithm first, for example the *skyline* method [18], which simply slices polyphony at each distinct onset and takes the highest pitch of every slice. The outer right branch of the tree is dealing with unvoiced polyphonic music by means of polyphonic models. One can easily imagine global features representing this kind of data, for example by computing relative frequencies of vertical intervals, i.e. intervals between simultaneous notes. However, building a truly polyphonic n-gram model remains a challenge, as one has to deal with segmentation and representation issues to cope with sparsity.

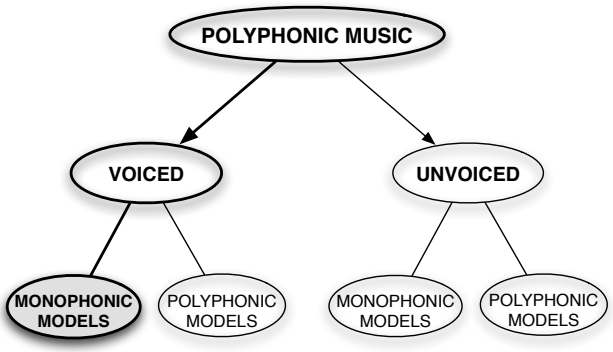


Figure 3. Tree structure outlining the possible ways to approach the classification of polyphonic music.

5. ACKNOWLEDGEMENTS

Darrell Conklin is supported by IKERBASQUE, Basque Foundation for Science, Bilbao, Spain. Ruben Hillewaere is supported by the project Messiaen Weerspiegeld in collaboration with the Royal Conservatory of Brussels. Special thanks go to Stijn Meganck and Jonatan Taminau for their useful comments and support during this research.

6. REFERENCES

[1] `http://www.ccarh.org`.

[2] `http://jmir.sourceforge.net/jSymbolic.html`.

[3] `http://www.cs.waikato.ac.nz/ml/weka/`.

[4] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[5] D. Conklin. Melodic analysis with segment classes. *Machine Learning*, 65(2):349–360, 2006.

[6] D. Conklin and I. H. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[7] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[8] R. Fiebrink and I. Fujinaga. Feature selection pitfalls and music classification. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 340–341, Victoria, Canada, 2006.

[9] M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 359–366, Stanford, USA, 2000.

[10] R. Hillewaere, B. Manderick, and D. Conklin. Global feature versus event models for folk song classification. In *ISMIR 2009: 10th International Society for Music Information Retrieval Conference*, pages 729–733, Kobe, Japan, 2009.

[11] R. Hillewaere, B. Manderick, and D. Conklin. Melodic models for polyphonic music classification. In *Second International Workshop on Machine Learning and Music*, Bled, Slovenia, 2009.

[12] C.W. Hsu, C.C. Chang, C.J. Lin, et al. A practical guide to support vector classification. Technical report, 2003.

[13] B. Jesser. *Interaktive Melodieanalyse*. Peter Lang, Bern, 1991.

[14] M. Li and R. Sleep. Melody classification using a similarity metric based on Kolmogorov complexity. In *Sound and Music Computing*, Paris, France, 2004.

[15] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *Proceedings of the International Conference on Music Information Retrieval*, pages 525–530, Barcelona, Spain, 2004.

[16] P. J. Ponce de Léon and José M. Iñesta. Statistical description models for melody analysis and characterization. In *Proceedings of the 2004 International Computer Music Conference*, pages 149–156, Miami, USA, 2004.

[17] P. Smialowski, D. Frishman, and S. Kramer. Pitfalls of supervised feature selection. *Bioinformatics*, 26(3):440, 2010.

[18] A.L. Uitdenbogerd and J. Zobel. Matching techniques for large music databases. In *Proc. ACM Multimedia Conference*, pages 57–66, Orlando, Florida, 1999.

[19] P. van Kranenburg and E. Backer. Musical style recognition - a quantitative approach. In *Proceedings of the Conference on Interdisciplinary Musicology (CIM)*, pages 106–107, Graz, Austria, 2004.

[20] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques. 2nd edition*. Morgan Kaufmann, San Francisco, 2005.

SUPERVISED AND UNSUPERVISED WEB DOCUMENT FILTERING TECHNIQUES TO IMPROVE TEXT-BASED MUSIC RETRIEVAL

Peter Knees, Markus Schedl, Tim Pohle, Klaus Seyerlehner, and Gerhard Widmer
Department of Computational Perception
Johannes Kepler University Linz, Austria
peter.knees@jku.at

ABSTRACT

We aim at improving a text-based music search engine by applying different techniques to exclude misleading information from the indexing process. The idea of the original approach is to index music pieces by "contextual" information, more precisely, by all texts to be found on Web pages retrieved via a common Web search engine. This representation allows for issuing arbitrary textual queries to retrieve relevant music pieces. The goal of this work is to improve precision of the retrieved set of music pieces by filtering out Web pages that lead to irrelevant tracks. To this end we present two unsupervised and two supervised filtering approaches. Evaluation is carried out on two collections previously used in the literature. The obtained results suggest that the proposed filtering techniques can improve results significantly but are only effective when applied to large and diverse music collections with millions of Web pages associated.

1. MOTIVATION AND CONTEXT

Searching for music by issuing "semantic" and descriptive queries has become a hot research topic recently [2, 4, 8, 11–13, 15]. While typical *query-by-example* systems require the user to have a specific piece of music at hand (or at least in mind) when searching for other music, *query-by-description* systems allow for typing in a short characterisation or a related term to find desired music. Moreover, it is generally desirable to build systems that are capable of linking music to meaningful textual descriptions (i.e., bridging what is often misleadingly called "semantic gap" [17]). For instance, this capability can be used to recommend music based on other textually represented information, e.g., by analysing the user's currently viewed Web page [7].

For the dedicated task of building a music search engine, several approaches have been presented. In [4], Baumann et al. describe a system incorporating various kinds of meta-data, lyrics, and acoustic properties. To analyse queries, natural language processing methods and knowledge from a semantic ontology are applied to map the query tokens to the corresponding concepts. In [8], Celma et al. propose usage of a Web crawler focused on audio blogs to obtain textual descriptions for music. Blog entries are extracted and the associated music pieces are indexed based on this information. From a text-based retrieval result, also acoustically similar songs can be discovered. Yang et al. [18] index a music collection using lyrics and apply a combination of text and audio descriptors to cluster results.

In [15,16], Turnbull et al. present a method for semantic retrieval. Based on the CAL500 data set – a collection of 500 songs manually labelled with descriptions representing music-relevant properties – models of these properties are learned from audio features. The system can then be used to retrieve relevant songs based on queries consisting of the words used for annotation. In [2], this approach is extended by incorporating multiple sources of features (i.e., acoustic features related to timbre and harmony, social tags, and Web documents). These largely complementary sources are combined to improve prediction accuracy.

In [13], we propose an unsupervised strategy for music retrieval that is capable of dealing with a large and arbitrary vocabulary. Contrary to learning a pre-defined set of labels (cf. [2, 15]), music pieces are represented in a vector space constructed from related Web documents. An improved version of this approach is presented in [11]. Instead of aggregating Web pages to construct term vectors, the retrieved Web documents are stored in an index. A given query is processed by passing the query to this index and applying a technique called *rank-based relevance scoring* to the resulting document ranking. This scoring is based on the associations between music tracks and Web documents (as we further extend this approach in this paper, a more detailed description can be found in Section 2). In [12], we propose unsupervised methods to improve search results by integrating audio similarity. Results show that the combinations can raise performance slightly but mainly introduce noise.

With this work, we aim at enhancing the approach from [11] by constructing filters that remove misleading information from the Web document index and raise precision of the retrieved music piece rankings (cf. [3]). Two of these filters are built in an unsupervised manner, whereas the other two make use of external annotations for learning to distinguish between informative and noisy content.

2. WEB-BASED MUSIC INDEXING

In the approach from [11, 12], music is indexed by using "contextual" meta-information about the pieces under consideration. This context-data is assumed to be found on related Web pages retrieved via Google by issuing three queries for each music piece m. Constructed from the meta-information categories *artist name*, *album name*, and *track title*, these queries are *"artist" music*, *"artist" "album" music review -lyrics*, and *"artist" "title" music review -lyrics*. For each of these queries, the top-100 Web pages are retrieved and joined into D_m, the set of pages associated with m. All retrieved documents are also stored in an index I. Relevance of a music piece m wrt. a given query q is assessed by querying I with q and applying *rank-based relevance scoring (RRS)* to the n most relevant Web documents in I (Equation 1).

$$RRS_n(m,q) = \sum_{p \in D_m \cap D_{q,n}} 1 + |D_{q,n}| - rnk(p, D_{q,n}) \quad (1)$$

In this equation, n denotes the maximum number of top-ranked documents when querying I, $D_{q,n}$ the ordered set (i.e., the ranking) of the n most relevant Web documents in I with respect to query q, and $rnk(p, D_{q,n})$ the rank of document p in $D_{q,n}$. For retrieval, the final ranking of music tracks is obtained by sorting the music pieces according to their RRS value.

In the published evaluations [11, 12], precision hardly ever exceeds 30% using this scoring approach, i.e., rankings usually contain three times more irrelevant music pieces than relevant. Based on this, also subsequent steps such as combination with audio similarity may suffer from erroneous input. Clearly, the reason for the high number of irrelevant pieces has to be searched for in the underlying Web pages. For indexing, all pages returned by Google are considered relevant, irrespective of whether they actually contain information about or descriptions of the corresponding music piece or artist. Furthermore, the page indexer does not distinguish between text that occurs in the "main part" of the Web page and text that is used for navigation or links to stories about other, completely unrelated artists. Thus, to improve precision of the retrieved set of music pieces, in the next section, we propose four different filtering approaches to remove noisy information and documents.

3. DOCUMENT FILTERING TECHNIQUES

This section describes the proposed filtering methods to exclude noisy information from the indexing process. We explore two types: Unsupervised and supervised filters.

3.1 Unsupervised Filtering

The characteristic of these filters is that they aim at identifying misleading texts without information from external sources. Hence, they can be applied to the index directly after building it. The first filter does not remove full documents from the index, but tries to identify those portions within the indexed text that do not contain specific information. The second approach identifies and removes complete documents.

3.1.1 Alignment-Based Noise Removal

As mentioned earlier, most indexed Web pages do not only contain relevant and interesting information (if any at all). Almost every page contains a site-specific header, navigation bar, links to related pages, and copyright disclaimers, frequently automatically generated by a content management system, cf. [9, 19]. Especially on music pages, these segments often feature lists of other music artists, genres, or tag clouds to facilitate browsing. This surrounding information is usually not relevant to the associated music piece and should thus be ignored.

Removal of this kind of text is the aim of this filter. Since large parts of the surrounding text remain the same for most pages within a Web domain, we can identify redundant segments by comparing several texts from the same domain. Coherent parts are most likely to be non-specific for a given music piece and can therefore be removed. To this end, we adopt the *multiple lyrics alignment* technique originally used to extract lyrics from multiple Web sources by matching coherent parts and preserving overlapping segments [14]. In the current filtering scenario, the overlapping segments are going to be removed.

To apply the filter, we collect all documents belonging to the same domain. Since for many blogs, the domain alone does not indicate similarly structured pages – different blogs are typically accessible via separate subdomains (e.g., for blogspot.com) – we keep the subdomain if the host section of the URL contains the word "blog". For domains that occur only up to five times in the page index, no filtering is performed. For all other domains up to eight documents are chosen randomly and used for alignment. From the alignment, we choose all aligned tokens occurring in at least 60% of the aligned texts and finally select all text sequences consisting of at least 2 tokens. The resulting sequences are then removed in all Web pages originating from the domain.

3.1.2 Too-Many-Artists Filtering

With this filter, the goal is to detect pages that do not deal with only one type of music, i.e., pages that provide an ambiguous content and are therefore a potential source of error. Some of these pages can be identified easily, since they contain references to many artists. Hence, we query the page index with all artist names from the music collection and count the occurrences of each page in the result sets. Constructing the filter simply consists in selecting a threshold for the maximum number of allowed artists per page. By systematically experimenting with this threshold, we yielded most promising results when removing all pages containing more than 15 distinct artists. Throughout the rest of this paper, *too-many-artists filtering* refers to the removal of pages containing more than 15 artists.

3.2 Supervised Filtering

As already mentioned in [12], automatic optimisation of the (unsupervised) Web-based indexing approach is difficult, since for arbitrary queries, there is no learning target known in advance (in contrast, for instance, to the approaches presented in [2, 15], where the set of possible queries is limited). However, in terms of identifying sources of noise, automatic optimisation approaches are somewhat more promising, provided that a set of potential queries with corresponding relevance judgements is available. The idea is that by observing performance on a given set of queries, it should be possible to learn to identify and exclude misleading Web pages and therefore yield better results also on other, previously unseen queries. This is based on the assumption that documents responsible for introducing noise to a music piece ranking contain erroneous (at least ambiguous) information and are likely to introduce noise to other queries too.

3.2.1 Query-Based Page Blacklisting

Following the general idea outlined in Section 3.2, we construct a simple filter that blacklists (i.e., excludes) Web pages contributing more negatively than positively to query results. Hence, based on RRS we calculate a simple score to rate a page p:

$$S_n(p) = \sum_{q \in Q} \left(\sum_{m \in M_p \cap T_q} RRS_n(m,q) - \sum_{m \in M_p \cap \overline{T_q}} RRS_n(m,q) \right) \quad (2)$$

where Q denotes the set of all available queries/annotations, M_p the set of all music pieces associated with page p, T_q the set of all pieces annotated with q (i.e., relevant to query q), and $\overline{T_q}$ its complement (i.e., all music pieces not relevant to q). Informally speaking, over all queries, we subtract the sum of RRS scores contributed to negative examples from the sum of RRS scores contributed to positive examples. We then remove all Web documents p with $S_n(p) < 0$, i.e., all documents that contributed more negatively than positively over the course of all queries.

3.2.2 Query-Trained Page Classification

While the *query-based page blacklisting* filter represents (if any) just the "laziest" form of machine learning (i.e., merely recognising instances without any kind of generalisation), this filter aims at learning to automatically classify Web pages as either "positive" (keep) or "negative" (remove). Hence, it should be better suited to deal with new queries that provoke previously unseen (and thus unrated) Web pages. To get positive and negative examples as training instances for the classifier, only pages that have either contributed exclusively positively or exclusively negatively are considered. Positive examples are defined as $\{p \,|\, p \in D_{q,n}, \forall q \in Q : M_p \cap \overline{T_q} = \emptyset\}$ and negative as $\{p \,|\, p \in D_{q,n}, \forall q \in Q : M_p \cap T_q = \emptyset\}$ (cf. Eq. 2). As a further requirement, only pages that appear in at least two query result sets are considered. As feature representation for Web pages, we incorporate characteristic values such as the length of the page's (unparsed) HTML content, the length of the parsed content, the number of different terms occurring on the page, the number of associated music pieces (i.e., $|M_p|$), the number of contained artist names (cf. 3.1.2), as well as ratios between these numbers. Furthermore, we utilise title and URL of the pages as very short textual representations that are converted into a term vector space (using the functions provided by WEKA [10]) and added as numerical features.

For classification, we decided to use the *Random Forest Classifier* [5] from the WEKA package (with 10 trees). Since there are usually significantly more negative than positive examples, we also apply a cost-sensitive meta-classifier to raise importance of positive instances (misclassification of positive instances is penalised by the ratio of negative to positive examples).

4. EVALUATION

For evaluation of the different filtering approaches, we use both test collections from [12]. The first collection, called c35k, is a large real-world collection and contains 35,000 mostly popular pieces. For evaluation purposes, a benchmarking set consisting of 200 queries and relevance judgements has been created from Last.fm tags [1] . The second collection is the CAL500 set, a collection of 500 songs manually labelled with words representing various music-relevant properties [15]. For comparison, we adopted the 139 category subset used in [12]. To test effectiveness of the retrieval approaches, annotations are used as queries to the system. They also serve as relevance indicator, i.e., a track is considered to be relevant for query q if it has been tagged with tag q. For evaluation of the supervised filtering approaches, a 10-fold cross validation is performed on the test collections, i.e., in each fold, 90% of the queries are used to train the filters which are then applied and evaluated on the remaining 10%.

To measure the quality of the obtained rankings, standard evaluation measures for retrieval systems are calculated, cf. [1]. Additionally to the "global" measures *precision* and *recall*, ranking measures like *precision@10 documents*, *r-precision* (i.e., precision at the r^{th} returned document, where r is the number of tracks relevant to the query), and *(mean) average precision* (*MAP*, i.e., the arithmetic mean of precision values at all encountered relevant documents) are used for evaluation. To further compare different retrieval strategies, we calculate *precision at 11 standard recall levels*. For each query, precision $P(r_j)$ at the 11 standard recall levels $r_j, j \in \{0.0, 0.1, 0.2, ..., 1.0\}$ is interpolated according to $P(r_j) = max_{r_j \leq r \leq r_{j+1}} P(r)$. This allows averaging over all queries and results in characteristic curves for each retrieval algorithm, enabling comparison of distinct settings. To obtain a single value for comparison of these curves, we calculate the area under the curve (*prec@11std.recall - AUC*). For presentation of the c35k collection, we decided to use tables instead of graphs to show more detailed results (including significance tests).

[1] `http://www.last.fm`

	Recall						**Precision**					
	UNF	ANR	2MA	A+2	QPB	QPC	UNF	ANR	2MA	A+2	QPB	QPC
$n = 10$	**2.18**	**2.01**	**2.07**	1.95	**2.01**	**2.69**	30.15	**31.89**	**35.72**	30.71	**34.32**	**36.98**
$n = 20$	**3.74**	**3.95**	**3.93**	**3.66**	**3.89**	**4.93**	29.02	31.30	**32.86**	30.91	**33.91**	**37.11**
$n = 50$	**7.17**	7.48	**7.87**	7.34	**8.15**	**9.15**	27.61	29.50	**31.13**	28.56	**32.50**	**34.79**
$n = 100$	**12.72**	12.09	12.58	**11.92**	**12.80**	**14.26**	25.99	27.64	29.05	27.25	**31.75**	**32.42**
$n = 200$	**18.67**	18.22	18.13	**17.95**	19.08	**20.73**	23.77	25.77	**26.21**	25.26	**28.60**	**28.74**
$n = 500$	29.31	29.60	29.84	28.16	29.96	**32.00**	20.12	21.69	21.71	21.00	**24.76**	**23.19**
$n = 1,000$	40.38	40.31	40.11	36.41	40.43	**41.52**	16.88	17.86	18.19	18.00	**20.75**	18.92
$n = 10,000$	**80.50**	**79.56**	76.80	57.55	73.50	73.32	7.29	7.42	7.87	**11.90**	**9.63**	8.62

	Prec@10						**r-Precision**					
	UNF	ANR	2MA	A+2	QPB	QPC	UNF	ANR	2MA	A+2	QPB	QPC
$n = 10$	**31.19**	**34.94**	**37.76**	**32.74**	**36.45**	**37.32**	**2.16**	**1.99**	**2.05**	1.79	**2.01**	**2.68**
$n = 20$	**32.40**	**34.96**	**37.05**	**31.93**	**36.75**	**37.89**	**3.63**	**3.68**	**3.67**	**3.39**	**3.69**	**4.57**
$n = 50$	**38.45**	36.20	**41.70**	36.52	**40.25**	**43.90**	**6.52**	6.85	**7.08**	6.29	**7.30**	**8.38**
$n = 100$	**44.10**	39.05	**46.65**	40.52	**43.40**	**48.40**	**10.24**	10.41	10.78	10.27	**11.52**	**12.48**
$n = 200$	47.75	42.15	**48.90**	43.82	**46.95**	**49.60**	14.22	14.54	14.68	14.27	**16.03**	**16.83**
$n = 500$	**50.30**	45.95	**51.20**	47.32	**49.75**	**52.45**	19.84	21.01	20.35	19.85	**22.54**	**23.52**
$n = 1,000$	**52.55**	48.75	**52.85**	48.47	**53.15**	**54.20**	24.22	25.43	25.00	23.66	**27.48**	**27.85**
$n = 10,000$	**57.45**	57.20	**56.70**	50.12	**62.35**	**58.00**	**35.20**	**35.77**	**35.03**	28.28	**35.69**	**34.70**

	Avg. Prec (MAP)						**Prec@11Std.Recall - AUC**					
	UNF	ANR	2MA	A+2	QPB	QPC	UNF	ANR	2MA	A+2	QPB	QPC
$n = 10$	**1.19**	**1.32**	**1.38**	**1.12**	**1.39**	**1.83**	**3.05**	**3.15**	**3.34**	**2.95**	**3.23**	**3.61**
$n = 20$	1.84	2.14	**2.24**	1.86	**2.26**	**2.95**	**3.64**	3.98	**4.07**	**3.74**	**4.06**	**4.78**
$n = 50$	3.24	3.79	**4.06**	3.58	**4.49**	**5.22**	4.99	5.63	**5.86**	**5.44**	**6.39**	**6.67**
$n = 100$	5.54	5.93	**6.29**	5.67	**7.00**	**7.64**	7.11	7.56	**7.91**	7.34	**8.51**	**9.14**
$n = 200$	8.23	8.61	8.78	8.26	**10.24**	**10.65**	9.62	10.12	10.19	9.75	**11.72**	**12.20**
$n = 500$	12.39	13.30	13.19	12.38	**15.06**	**15.93**	13.76	14.69	14.57	13.89	**16.45**	**17.43**
$n = 1,000$	16.10	17.37	17.01	15.45	**19.41**	**19.84**	17.22	18.84	18.36	16.82	**20.82**	**21.03**
$n = 10,000$	**29.98**	**30.60**	**29.80**	21.86	**30.92**	29.22	**31.25**	**31.84**	**31.07**	23.07	**32.05**	30.39

Table 1. Comparison of *unfiltered RRS* (UNF) vs. *Alignment-Based Noise Removal* (ANR), *Too-Many-Artists Filtering* (2MA), *ANR+2MA* (A+2), *Query-Based Page Blacklisting* (QPB), and *Query-Trained Page Classification* ($\text{QPC}_{n=200}$) for the c35k collection and different values of n, i.e., the maximum number of retrieved Websites incorporated in RRS. QPB and QPC are performed upon ANR. Values (given in %) are obtained by averaging over 200 evaluation queries (for supervised approaches via 10-fold Cross Validation). Entries in bold face indicate that there is no significant difference between this entry and the best performing, i.e. bold entries indicate the "best group" (Friedman test, $\alpha = 0.01$). Note that due to the rank-based nature of the non-parametric Friedman test, results may belong to the best group even with lower average values than significantly worse results.

Table 1 shows evaluation results on the c35k collection for different values of n (number of top ranked Web documents when querying the page index). For the alignment-based noise removal (ANR), we observe slight improvements for the averaged results especially for precision, r-precision, average precision and the area under the standardized precision-recall curve. However, in the Friedman test these results are not significant. For recall and precision@10 we can see a significant drop in performance.

The too-many-artists filter (2MA) outperforms the unfiltered RRS significantly in terms of precision and average precision for smaller values of n. A decrease is most clearly visible for recall. In addition, we evaluated also the combination of both unsupervised filters (A+2). In most cases, this combination worsens results significantly which is rather surprising, considering that these filters target different levels of noise removal. However, it seems that too much information is excluded when using both.

Except for recall, both *supervised approaches* are constantly in the best performing group, superiority is clearly visible for precision. For the query-trained page classification filter (QPC), it has to be mentioned that for values of $n > 500$ the number of training instances gets very high, slowing down the evaluation progress. For this reason, we decided to use the QPC filter with the $n = 200$ setting also for experiments with $n \neq 200$. This explains also the slight drop for $n \geq 1,000$. Still, results are more than acceptable for QPC.

	Recall						Precision					
	UNF	ANR	2MA	A+2	QPB	QPC	UNF	ANR	2MA	A+2	QPB	QPC
$n = 10$	5.96	6.10	5.85	6.00	6.10	6.69	25.77	27.17	25.72	26.99	27.22	28.35
$n = 20$	10.19	9.68	9.46	9.49	9.83	10.69	24.87	25.39	24.69	25.44	25.07	25.73
$n = 50$	17.99	18.09	17.20	17.61	18.23	17.89	22.84	23.14	22.44	22.94	23.22	22.68
$n = 100$	26.80	26.34	25.48	25.90	27.11	***24.34***	21.02	21.05	20.80	21.10	21.69	20.16
$n = 200$	38.63	38.62	37.24	37.21	37.95	***31.75***	19.15	19.30	19.11	19.25	19.67	18.54
$n = 500$	56.31	55.65	54.26	***53.31***	***51.21***	***38.09***	16.86	16.95	16.92	17.05	17.74	17.24
$n = 1,000$	66.91	66.48	63.78	***62.69***	***53.10***	***40.10***	***15.54***	***15.81***	***15.74***	***16.01***	17.36	16.74
$n = 10,000$	73.27	72.93	***69.06***	***68.02***	***37.98***	***40.76***	***14.56***	***14.84***	***14.82***	***15.17***	20.75	***16.56***

Table 2. Comparison of unfiltered RRS vs. the filter approaches for the CAL500 set averaged over 139 queries (cf. Table 1). Note that in contrast to Table 1, in this table, bold and italic appearing entries indicate a significant difference to the group of best approaches, i.e., worse results are marked. For all experiments with QPC, the setting $\text{QPC}_{n=50}$ is used.

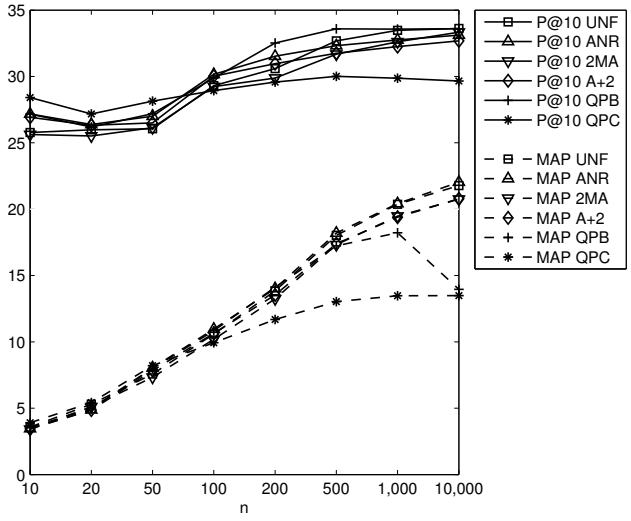


Figure 1. *Precision@10* (upper curves) and *Avg. Prec (MAP)* (lower curves) for the CAL500 set and different values of n (cf. Table 2).

For the CAL500 set, results are very disappointing (Table 2). No proposed filter can significantly improve results (except for precision of the supervised filters with high values of n, which go along with a dramatic loss in recall due to a very high number of excluded pages). The reasons are not directly comprehensible. One possibility could be that in the case of the c35k set with associated Last.fm tags, the approaches benefit from the inherent redundancies in the tags/queries (e.g., *metal* vs. *black metal* vs. *death metal*). In the case of the CAL500 set, queries exhibit no redundancy, as the set is constructed to describe different dimensions of music. However, this would only affect the supervised filters.

Another explanation could be that the CAL500 page index contains considerably less pages than the c35k index (approx. 80,000 vs. approx. 2 million pages). First, and also in the light that the CAL500 set has been carefully designed, it seems possible that the index does not contain so much noise. Hence, the proposed noise removal strategies don't work here. Second, since the index is rather small, removal of a relatively high number of pages has a higher impact on the overall performance. This becomes especially apparent when examining the results of the supervised approaches for high n. Apart from the results, it should be noted that the CAL500 set is without doubt very valuable for research (high quality annotations, freely available, etc.) but at the same time, it is a highly artificial corpus which can not be considered a "real-world" collection. Hence, some "real-world" problems maybe can not be tested with such a small set.

5. CONCLUSIONS AND FUTURE WORK

We have demonstrated the usefulness of two unsupervised and two supervised filtering approaches for Web-based indexing of music collections. Evaluation showed inconsistent results for two collections with very different characteristics and suggests that the proposed filtering techniques can improve results significantly when applied to large and diverse music collections with millions of Web pages associated.

Regarding the proposed filtering techniques, more or less all of them proved to be useful and could improve not only the overall precision but also the ranking of music pieces. By introducing supervised optimisation into this originally unsupervised technique, there is still more potential to tweak performance. For instance, we are convinced that a more carefully selected feature set can easily improve results of page classification further. Using annotated sets for learning, also proper combination with audio similarity, e.g., to raise recall, could be possible.

Instead of finding redundant portions in Web pages from the same domain by aligning and matching their content, techniques like *vision page segmentation* [6] could help in identifying the relevant parts of a Web page. By extracting smaller segments from Web pages, the principle of the RRS weighting could be transferred to "blocks" and scoring could be designed more specifically.

Another aspect not directly related to filtering pages became apparent during experiments with the Too-many-artists filter. When querying the page index with the names

of the contained artists, artists with common speech names can be easily identified. As each artist only has a limited number of associated pages in the index, true occurrences are somehow normalised. For artists that occur much more often than expected (outlier), it can be assumed that they have common speech names. This finding could be interesting for related tasks in future work.

6. ACKNOWLEDGMENTS

This research is supported by the Austrian "Fonds zur Förderung der Wissenschaftlichen Forschung" (FWF) under project number L511-N15.

7. REFERENCES

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, Reading, Massachusetts, 1999.

[2] Luke Barrington, Douglas Turnbull, Mehrdad Yazdani, and Gert Lanckriet. Combining audio content and social context for semantic music discovery. In *Proceedings of the 32nd ACM SIGIR*, Boston, MA, USA, 2009.

[3] Stephan Baumann and Oliver Hummel. Using Cultural Metadata for Artist Recommendation. In *Proceedings of the 3rd International Conference on Web Delivering of Music (WEDELMUSIC 2003)*, September 2003.

[4] Stephan Baumann, Andreas Klüter, and Marie Norlien. Using natural language input and audio analysis for a human-oriented MIR system. In *Proceedings of the 2nd International Conference on Web Delivering of Music (WEDELMUSIC 2002)*, Darmstadt, Germany, 2002.

[5] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Extracting content structure for web pages based on visual representation. In *Fifth Asia Pacific Web Conference (APWeb 2003)*, 2003.

[7] Rui Cai, Chao Zhang, Chong Wang, Lei Zhang, and Wei-Ying Ma. Musicsense: contextual music recommendation using emotional allocation modeling. In *Proceedings of the 15th ACM Multimedia*, 2007.

[8] Oscar Celma, Pedro Cano, and Perfecto Herrera. Search Sounds: An audio crawler focused on weblogs. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR'06)*, Victoria, B.C., Canada, 2006.

[9] Sandip Debnath, Prasenjit Mitra, and C. Lee Giles. Automatic extraction of informative blocks from webpages. In *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC'05)*, 2005.

[10] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

[11] Peter Knees, Tim Pohle, Markus Schedl, Dominik Schnitzer, and Klaus Seyerlehner. A Document-centered Approach to a Natural Language Music Search Engine. In *Proceedings of the 30th European Conference on Information Retrieval (ECIR'08)*, Glasgow, Scotland, UK, March 30-April 3 2008.

[12] Peter Knees, Tim Pohle, Markus Schedl, Dominik Schnitzer, Klaus Seyerlehner, and Gerhard Widmer. Augmenting Text-Based Music Retrieval with Audio Similarity. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan, October 2009.

[13] Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer. A Music Search Engine Built upon Audio-based and Web-based Similarity Measures. In *Proceedings of the 30th ACM SIGIR*, Amsterdam, the Netherlands, July 23-27 2007.

[14] Peter Knees, Markus Schedl, and Gerhard Widmer. Multiple Lyrics Alignment: Automatic Retrieval of Song Lyrics. In *Proceedings of 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, September 2005.

[15] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Towards Musical Query-by-Semantic-Description using the CAL500 Data Set. In *Proceedings of the 30th ACM SIGIR*, Amsterdam, the Netherlands, July 23-27 2007.

[16] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, February 2008.

[17] Geraint Wiggins. Semantic Gap?? Schemantic Schmap!! Methodological Considerations in the Scientific Study of Music. In *Proceedings of the 11th IEEE International Symposium on Multimedia (ISM'09): Workshop on Advances in Music Information Research (AdMIRe)*, 2009.

[18] Yi-Hsuan Yang, Yu-Ching Lin, and Homer Chen. Clustering for music search results. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2009.

[19] Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In *Proceedings of the 9th ACM SIGKDD Conference*, 2003.

Symbol Classification Approach for OMR of Square Notation Manuscripts

Carolina Ramirez
Waseda University
ramirez@akane.waseda.jp

Jun Ohya
Waseda University
ohya@waseda.jp

ABSTRACT

Researchers in the field of OMR (*Optical Music Recognition*) have acknowledged that the automatic transcription of medieval musical manuscripts is still an open problem [2, 3], mainly due to lack of standards in notation and the physical quality of the documents. Nonetheless, the amount of medieval musical manuscripts is so vast that the consensus seems to be that OMR can be a vital tool to help in the preserving and sharing of this information in digital format.

In this paper we report our results on a preliminary approach to OMR of medieval plainchant manuscripts in square notation, at the symbol classification level, which produced good results in the recognition of eight basic symbols. Our preliminary approach consists of the preprocessing, segmentation, and classification stages.

1. INTRODUCTION

Several groups are currently working to build digital archives and catalogues using digital technologies [10, 11, 12, 13, 14], of the huge number of early musical manuscripts accessible from multiple sources. The lines of research of these groups in early music information retrieval range from the design of web protocols for digital representation of scanned early music sources to the automatic transcription of those sources through adaptive techniques [2, 5, 9, 10]. Given the physical and semantic characteristics of many of these documents (degradation, non-standard notation, etc.), great variability is introduced to the data, and the subsequent analysis can be a quite difficult and time consuming task, usually requiring advanced expert knowledge. So, until very recently, those mentioned efforts were restricted mostly to build text catalogues and repositories of scanned images.

In the case of standard modern music notation, OMR has achieved high levels of accuracy, and there are several OMR systems commercially available [1, 15]. In the case of early music manuscripts, attempts to achieve good OMR results become more challenging as our sources go back in time. Still, researchers have extended their work to early music manuscripts, and in the past years we have observed advances in renaissance printed music and handwritten music [4, 5, 17], but still little has been reported about experimental results with western plainchant medieval sources [2]. The work done by the NEUMES project [10], and most recently by Burgoyne et al. [3], are among the few experimental results with this particular type of source. In [2] the problem of non-standard notation is mentioned as the most critical issue for early manuscript OMR. For this reason, we start our research by restricting the manuscripts in square notation to belong to the XIV century and later, when square notation was already an established practice and basic symbols were more standardized than in previous neumatic alphabets [16].

In this paper we aim to successfully classify the eight basic characters of western square notation, see Figure 1, using relatively simple and widely known image processing and pattern recognition algorithms. If this proves successful, we believe that more complex models, context information, and adaptive techniques can be used in the future to minimize the errors at the classification stage, to extend the span of examples that can be analyzed, i.e. less standard documents, and to include a whole semantic analysis.

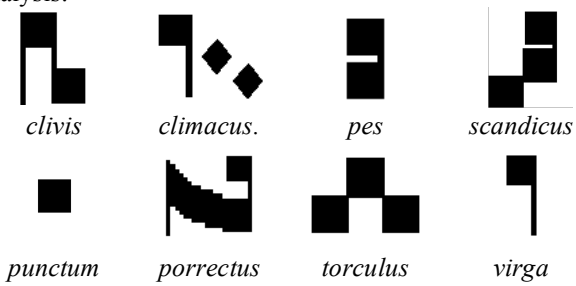


Figure 1: Square notation basic symbols.

Finally, it is necessary to mention that a big concern in this research area is the evaluation methods to be used. Symbol classification can be evaluated using the usual techniques, but creating a ground-truth for a full manuscript (where even the experts sometimes disagree) would require an effort that is beyond the scope of this paper.

2. OUTLINE

In section 3 we describe the preprocessing stage, which includes binarization of the manuscript image, location of staff lines and staves that define our ROI (*Region of Interest*), and stave deskewing. In section 4 we describe our segmentation and classification strategy. Lastly, in section 5 we present our conclusions and delineate some future work ideas.

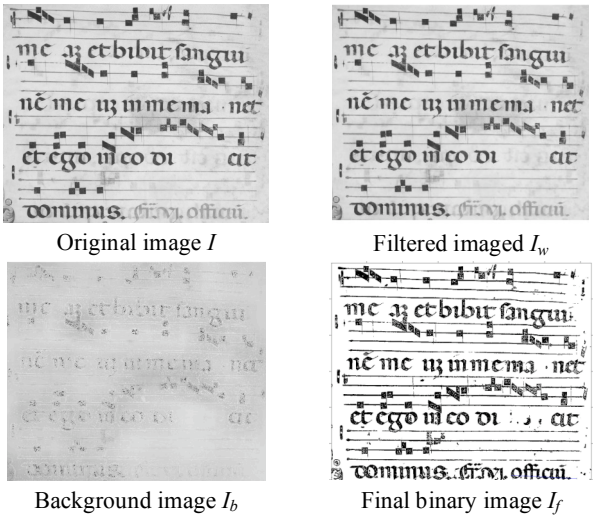
3. PREPROCESSING

3.1 Binarization and ROI Extraction

As we said above, one of the biggest difficulties in analyzing early music manuscripts comes from the high variability on the image data introduced by the deteriorated state of the documents [9]. Besides dealing with a non-standard notation or non-standard scanning methods, the physical condition of some documents (high degradation, discoloration, missing parts, etc.) calls for an adequate amount of preprocessing. Some possibilities for the preprocessing stage include filtering, spatial transforms (Hough transform has been proposed to correct staff line positions [5]), and adaptive thresholding.

In order to binarize and extract the ROI we implement the adaptive approach proposed by Gatos et al. in [6]. The main advantage of this method is that it is able to deal with degradations due to shadows, non-uniform illumination, low contrast, smear, and strain. The disadvantage is that it is a parametric method, and in order to obtain good results some amount of parameter tuning is required [4]. The steps include an initial denoising using a 3x3 Wiener filter, a rough foreground estimation using Sauvola's Local Adaptive Threshold, a background estimation, and a final local thresholding using the distance between the Wiener filtered image and the background estimation. We did not implement the up-sampling stage in [6], because preliminary tests showed that it was not critical to detect our ROI.

The original image I, the filtered image I_w, the background image I_b, and the final binary image I_f are shown in Figure 2.



Original image I — Filtered imaged I_w

Background image I_b — Final binary image I_f

Figure 2: Binarization stages.

We use the binary image I_f to detect our region of interest, the area of the image where the relevant symbols are located, which in a musical document is a stave, i.e. a group of staff lines. There can be many staves in one document and we want to extract each one of them separately. This also helps to minimize the presence of text and drawings in the analyzed images, elements that could make our analysis more difficult.

As an initial approach, we perform a rough localization of the staff lines by first detecting the positions of all the lines in the document using polar Hough transform. After the lines are extracted, we use another feature to decide if a group of lines is a stave. This feature is the space between lines, which can be also estimated from the Hough transform. Here we use the hypothesis that spaces between staff lines on the same stave are relatively smaller than the space between staves. We use a k-means classifier to group the spaces and detect the staves. Figure 3 shows an example of stave detection. Only whole staves will be extracted, so staff lines that do not form a complete stave are not considered as part of the ROI.

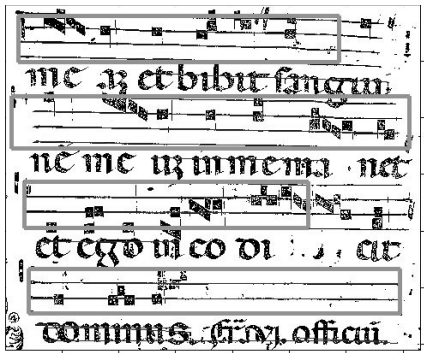


Figure 3: Stave detection.

In Figure 3 it can be noticed that the whole length of the stave is not detected. To solve this problem we use heuristics based on the inter-staff line and inter-staves spaces and the dimensions of the image.

3.2 Staves Deskewing

Many OMR algorithms assume that staff lines are horizontal, but this is not necessarily true in old manuscripts.

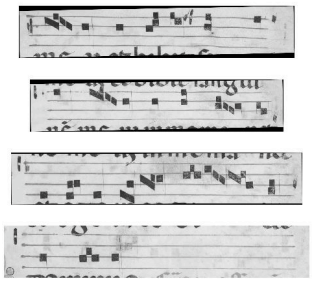


Figure 4: Aligned staves.

In order to facilitate the analysis, and in case we want to apply standard OMR techniques, it is useful to horizontally align the images as much as possible. This can be done with the information already obtained from the Hough transform, by rotating against the Hough angle. The result of applying this rotation can be seen in Figure

4. Note that this approach does not address the issue of deformed staff lines.

4. SEGMENTATION AND CLASSIFICATION

As explained in the introduction, we aim to obtain good symbol classification results while at the same time using a relatively simple methodology. In general, the standard approach is to binarize the document and then segment and classify the symbols using binary representations. We cannot use this approach because, even though the binarization we used above allows us to find the region of interest in the image, it is not accurate enough to conserve all the pixel information of the symbols across all the documents in our database. Hence, we carry out the segmentation directly from the extracted staves in grayscale. Due to the difficulty in removing lines from heavily degraded and deformed documents, we decided to skip the staff lines removal stage, and thus avoid a pixel-wise approach for symbol segmentation. Instead, we detect and segment whole symbols using pattern matching via correlation, and then we use a SVM (*Support Vector Machine*) to classify the symbols from gradient-based features.

4.1 Segmentation

We use normalized correlations on each stave image to match an artificially generated binary pattern of each symbol to the regions where that symbol potentially appears. Some of the binary patterns can be seen in Figure 1, but the classes that present more variability in size and geometrical distribution (*pes*, *torculus*, *porrectus*, *clivis*) are also divided in subclasses. These patterns were applied in 3 different scales, based in the height of the stave, to each stave image. After this process, a set of detected candidates is obtained. These candidates are the input for the SVM. An example of this process is shown in Figures 5, 6, and 7.

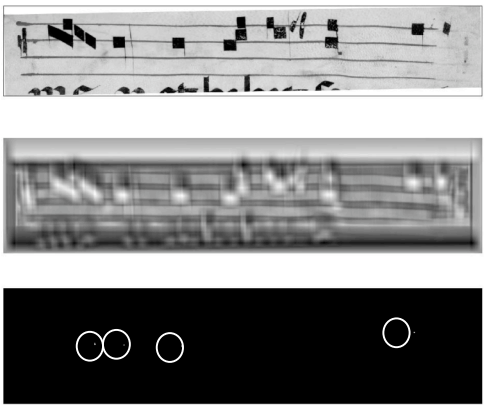


Figure 5: From top to bottom. Grayscale stave, normalized correlation image, and peaks of the correlation image.

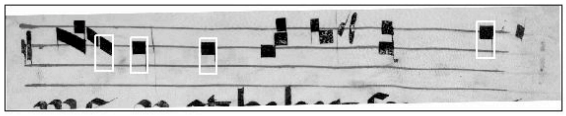


Figure 6: Pattern Detection, class *virga*.

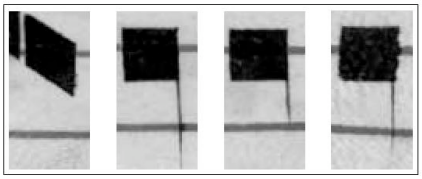


Figure 7: Segmented symbols, class *virga* (left, false detection).

After testing our detection algorithm in real documents [14], we observe that all basic symbols were detected with the binary patterns, but also many "false" candidates were extracted. These "false" candidates were mainly due to two causes: first, a basic pattern is actually part of another one, and second, a geometrical configuration similar to the basic pattern is formed by certain elements in the document. Examples of both conditions are shown in Figure 8.

Figure 8: Left, false *pes* detection (part of *scandicus*). Right, false *torculus* detection (part of *porrectus flexus*).

4.2 Classification

For Classification purposes, 1334 sample images of the 8 basic symbols were manually segmented and labeled from 47 sheets of music available at the Digital Scriptorium [14]. These sources are square notation manuscripts from the XIV to the XVII centuries (to avoid transitional times [16]), and from different geographical locations (Spain, Germany, Italy, etc.).

A size and position normalization using aspect ratio was performed on the samples [7], and 4 directional Sobel masks were applied to them (horizontal, vertical, left-diagonal, and right-diagonal) to obtain the gradient-based features used for classification. These Sobel images were divided in 96 blocks, and the mean gradient for each block was calculated. Finally, all the values were stacked in a feature vector [8].

We trained a SVM with a quadratic kernel function, and we tested it using cross-validation. The training was made using a *one-against-all* approach, thus obtaining a classifier for each of the eight classes. A simple voting algorithm is used to decide the final class from the outputs of the eight independent classifiers. Three experi-

ments were conducted, each with a different type of input. In the first experiment, we used grayscale samples without any quality enhancement, in the second experiment we used grayscale samples with contrast enhancement, and in the third experiment we used binary samples. Results are shown in Table 1.

Sample	Recall
Binary	0.8453
Grayscale	0.9208
Contrast enhanced	0.9610

Table 1: Classification rates for SVM cross-validation experiments. Values range from 0 to 1.

Table 2 shows the test results from 3000 independent examples, by class, for contrast-enhanced samples.

Class	Precision	Recall
Clivis	0.9331	0.9914
Climacus	0.9429	0.9519
Pes	0.9646	0.9542
Punctum	0.9674	0.9132
Porrectus	0.8476	0.8580
Scandicus	0.8667	0.8228
Torculus	0.9261	0.9482
Virga	0.9744	0.9311

Table 2: Classification results for contrast-enhanced samples. Values range from 0 to 1.

The candidates extracted from Section 4.1 were tested in the most successful of the three SVMs, with good classification rates. In the case of "false" candidates, the classifier is currently not capable of discern them as a different class, i.e. a class of "wrong" samples independent of the 8 basic classes.

5. SUMMARY AND DISCUSSION

We believe that our results, while not being completely conclusive, show that using a gradient-based feature generates good classification results of square notation at the symbol level provided the results from both detection and segmentation stages are good. When combining the detection stage with the classification stage, the performance is degraded by the presence of "false" detections obtained with the normalized correlation pattern matching. However, even if these results are not ideal, we consider that the errors in the classification of the "false" candidates can be reduced if we introduce two valuable elements into the analysis. The first element is the use of the redundancy in the detection, i.e. when two or more candidates are extracted from similar or overlapping positions in the image; the second element is the use of the context in which the symbol is found. In the first case, the sole presence of redundancy will alert us to the occurrence of an abnormal situation, and therefore allow us to act on it accordingly. In the second case, context information can be used to minimize errors: think of a basic pattern being part of another (for the worst case scenario, think of a *punctum*!). In that case, observing the context is essential to obtain complete information about the symbol under analysis, and be able to determine its correct class.

In terms of future work, our first concern is to improve the segmentation via pattern matching, without renouncing to other segmentation techniques. It is quite intuitive to imagine that some classes are more difficult to deal with. For instance, we observed that in many cases the classes *virga* and *punctum* were detected as the other, which makes us think that the characteristic stem of the *virga* has a weak influence in the normalized correlation pattern matching.

Finally, we believe that a robust analysis of these manuscripts cannot be completely achieved without also taking in account semantic context information. In general terms, plainchant is a sequence of sounds and rhythmic patterns evolving in time, and as such, models or techniques that deal with time sequences look like an attractive alternative to complement the symbol-based analysis and improve error management strategies. We know that certain rules are observed in Gregorian Chant, so, if some probabilistic rules can be derived from its semantics, even soft ones, we would like to undertake that direction of research.

6. ACKNOWLEGMENTS

We would like to thank the Free Library of Philadelphia, Rare Book Department, for granting their permission to reproduce images from their repository [18].

7. REFERENCES

[1] Bainbridge, D. and Bell, T. *The Challenge of Optical Music Recognition.* Computers and the Humanities, No 35, pp95-121. 2001.

[2] Barton, L.W. G., Caldwell, J. A. and Jeavons, P. G. *ELibrary of Medieval Chant Manuscript Transcriptions*. Proceedings of the 5yth ACM/IEEE Joint Conference on Digital Libraries (Digital Libraries Cyberinfraestructure for Research and Education). Association for Computing Machinery. 2005, pp320-329.

[3] Burgoyne, J.A., Y. Ouyang, T. Himmelman, J. Devaney, L. Pugin, and I. Fujinaga. *Lyric extraction and recognition on digital images of early music sources*. Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009) 2009.

[4] Burgoyne, J. A., L. Pugin, G. Eustace, and I. Fujinaga. 2007. A comparative survey of image binarisation algorithms for optical recognition on degraded musical sources. *Proceedings of International Conference on Music Information Retrieval*. Vienna. 509–12.

[5] Fornes, A., Llados, J. & Sanchez, G. *Primitive Se mentation in Old Handwritten Music Scores*. Lecture Notes in Computer Science, vol. 3926, pp. 279-290. 2006.

[6] Gatos, B., Pratikakis, I.E., Perantonis, S.J. *Adaptive degraded document image binarization*. Pattern Recognition, Vol.39, No. 3, pp. 317-327. March 2006.

[7] CL Liu, K Nakashima, H Sako, H Fujisawa. Handwritten Digit Recognition: Investigation of Normalization and Feature Extraction Techniques. Pattern Recognition, vol. 37, pp. 265-279.2004 .

[8] CL Liu, K Nakashima, H Sako, H Fujisawa. *Handwritten Digit Recognition: Benchmarking of State_of_the_Art Techniques*. Pattern Recognition, vol. 36, pp. 2271-2285.2003

[9] Pugin, L., Burgoyne, J.A. & Fujinaga, I. *MAP Adaptation to Improve Optical Music Recognition of Early Music Documents Using Hidden Markov Models*. Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007), pp. 513-16. Vienna, Austria.

[10] NEUMES Project
http://www.scribeserver.com/NEUMES/index.html

[11] CANTUS Database
http://publish.uwo.ca/~cantus/

[12] The CAO-ECE Project
http://www.zti.hu/earlymusic/cao-ece/cao-ece.html

[13] Cantus Planus Study group
http://www.cantusplanus.org/

[14] Digital Scriptorium
http://www.digital-scriptorium.org

[15] OMR Systems
http://www.informatics.indiana.edu/donbyrd/OMRSystemsTable.html

[16] Nota Quadrata
http://notaquadrata.ca/index.html

[17] Aruspix
http://www.aruspix.net/project.html

[18] Lewis E M 73:13v. Used by permission of the rare Book Department, Free Library of Philadelphia.

TEMPO INDUCTION USING FILTERBANK ANALYSIS AND TONAL FEATURES

Aggelos Gkiokas[1,2], Vassilis Katsouros[1] and George Carayannis[2]
[1]Institute for Language and Speech Processing
[2]National Technical University of Athens
{agkiokas, vsk, gcara}@ilsp.gr

ABSTRACT

This paper presents an algorithm that extracts the tempo of a musical excerpt. The proposed system assumes a constant tempo and deals directly with the audio signal. A sliding window is applied to the signal and two feature classes are extracted. The first class is the log-energy of each band of a mel-scale triangular filterbank, a common feature vector used in various MIR applications. For the second class, a novel feature for the tempo induction task is presented; the strengths of the twelve western musical tones at all octaves are calculated for each audio frame, in a similar fashion with Pitch Class Profile. The time-evolving feature vectors are convolved with a bank of resonators, each resonator corresponding to a target tempo. Then the results of each feature class are combined to give the final output.

The algorithm was evaluated on the popular ISMIR 2004 Tempo Induction Evaluation Exchange Dataset. Results demonstrate that the superposition of the different types of features enhance the performance of the algorithm, which is in the current state-of-the-art algorithms of the tempo induction task.

1. INTRODUCTION

Tempo Induction has gained a great interest within the Music Information Retrieval community the past few years. Although in most systems, the tempo induction is made simultaneously with the beat tracking process as a unified task, the need for an individual handling of these tasks is apparent. An example can be found in Gouyon and Dixon in [1], where a genre classifier for 8 different music genres, based solely on the tempo of each excerpt has given remarkable results.

Beyond the scope of music classification, tempo induction and beat tracking are essential in many diverse applications, such as music similarity and recommendation, automatic transcription, audio editing, music to MIDI synchronization, and automatic accompaniment. They almost always serve as an inter-step in algorithms handling more complicated problems such as meter extraction [2] and rhythm description.

The algorithms that extract tempo can be divided into two main categories. The first consists of algorithms that use onset lists as input (either extracted directly from MIDI or audio). Indicative work can be found in [3], [4]. Most of these algorithms extract periodicities from the *inter-onset intervals* (IOIs) or by applying the autocorrelation function (ACF) to the onsets list in order to extract the tempo. In the latter belong the algorithms that search for periodicities directly from the audio (e.g. the ACF applied to frame features). Respective work can be found in [5], [6]. Although the former have the advantage of generalization (handling both MIDI and audio), evidence that the latter achieves better results is reported in [7]. An extensive review on the rhythm description algorithms can be found in [8].

A first step to systemize the tempo extraction task was the evaluation exchange organized during the 5th International Conference on Music Information Retrieval [7]. Seven participants submitted twelve different algorithms, tested on a collection of 3199 tempo-annotated music excerpts. The data was hidden from the participants. After the contest was conducted, the data were made available online (except of the *Loops* data that are available under a fee). Detailed description can be found in [7]. In a similar fashion, MIREX 2005[1] and MIREX 2006[2] Audio Tempo Extraction evaluation exchanges were conducted, with the difference that the evaluation procedure was more focused on the perceived than actual tempo. Unfortunately, the data is still not available except of a small portion that was used as training data.

Although a benchmark collection was created, few tempo induction algorithms have been tested on this dataset. A remarkable exception is Seyerlehner, Widmer and Schnitzer's work [9]. They proposed two versions of an algorithm that extracts rhythmic patterns using the autocorrelation function (ACF) as described in [10] and the Fluctuation Pattern as described in [11], respectively, in order to determine the tempo. Their approach is based on the assumption that pieces with similar rhythmic patterns are more likely to have similar tempo as well. The rhythmic patterns of excerpts are compared with those of a tempo-annotated music database. Their results showed that the proposed algorithm outperformed all the algorithms presented in the ISMIR 2004 evaluation exchange

[1] http://www.music-ir.org/mirex/2005/index.php/Audio_Tempo_Extraction
[2] http://www.music-ir.org/mirex/2006/index.php/Audio_Tempo_Extraction

on the *ballroom* data, and had similar results with the top performing algorithm [2] on the *songs excerpts* data. Note that the results presented are based on *accuracy1*, and not on *accuracy2*, where correct tempi are considered the fractions of the ground-truth tempo (half, double, three times, 1/3) which partly can be considered perceptually more relevant. Another example is the work of Alonso et al. in [12]. They proposed a system that estimates the tempo by decomposing the music signal sub-bands into harmonic and noise components. Then musical events are extracted with an "accentuation" weighting and periodicities are estimated. They tested the proposed algorithm on a corpus consisting of the *songs excerpts* collection and excerpts from the author's private collection. The evaluation measures were *accuracy1* and *accuracy2*, but with a 5% tolerance. Thus, their algorithm cannot be compared directly with the aforementioned.

In this paper we present a system that extracts tempo without onset detection, in a similar fashion that Scheirer does in [5]. The difference is that additionally to the filter-bank analysis, we incorporate a novel feature for the tempo induction task, similar to Pitch Class Profile, introduced by Fujishima [13]. The proposed algorithm assumes no significant tempo variation within the music excerpt.

The rest of the paper is organized as follows. In Section 2 we describe the architecture of our system. Section 3 focuses on details concerning the algorithm and individual processes of the implemented system. Comparable results on the *ballroom* and *songs excerpts* data of the ISMIR 2004 tempo induction evaluation exchange are provided in Section 4. Conclusions, drawbacks and future work conclude this paper in Section 5.

2. ALGORITHM OVERVIEW

The overall architecture and the individual components of the proposed algorithm are shown in Figure 1. Initially a moving Gaussian window is applied on to the input signal. For each frame a filterbank of equally spaced triangular filters in the mel-scale is applied, and the log-energy of each bank is calculated, in order to produce a vector **m** for each frame. Simultaneously, a similar process takes place, using a larger window. Each frame is convolved with twelve filters, each one corresponding to one of the twelve musical tones, forming the vector **t**.

Then a larger window of 8secs length is applied to each time-evolving feature, with a 1sec shift. Afterwards, the features are differentiated and convolved with a bank of resonators, with frequencies corresponding to the target tempi, and the $\|\bullet\|_\infty$ of each convolution is calculated. Then the norms are summed across features, independently for each feature type, forming two vectors $\mathbf{SC}_m$ and $\mathbf{SC}_t$ of length *T*, where *T* denotes the plurality of the target tempi. Each vector indicates the strength of each target tempo to the specific frame. Finally $\mathbf{SC}_m$ and $\mathbf{SC}_t$ are summed across the segments of the whole excerpt, to get the final tempo strengths for each feature class. The two vectors are combined to get the final output.

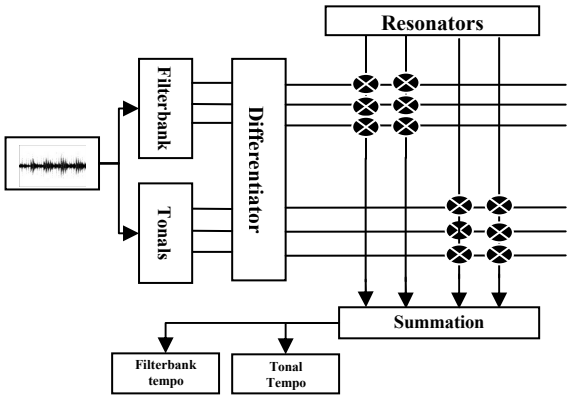


Figure 1. System Overview

3. ALGORITHM DETAILS

3.1 Extracting Filterbank Features

A moving Gaussian window of 20ms length with 5ms shift is applied to the input signal. Each segment is analyzed by a mel-scale triangular filterbank consisting of 12 bands, and the log-energy for each band is computed. This process forms a 12-dimensional feature vector **m**, for each frame.

3.2 Extracting Tonal Features

In a similar fashion with filterbank analysis, a Gaussian window is applied to the signal. In order to have better frequency resolution, the window is chosen much larger than the case before. A window with 80ms length and 5ms shift was chosen after experiments.

Each segment is convolved with 12 reference signals of same length with the sliding window. Each signal represents one the 12 western musical tones. The reference signals are the sum of cosines of equal amplitudes, at frequencies equal to the F_0 of each musical tone, at all octaves within a range from 27.5Hz to 10 kHz. No harmonic partials are considered. Formally, the reference signals are given by the following formula:

$$R_{tone(k)}(n) = \sum_{f_i \in \Omega_k} cos(2\pi f_i n), \quad k = 1..12 \qquad (1)$$

where Ω_k denotes the set of fundamental frequencies of tone *k* in the range of 27.5Hz to 10 kHz and *n* the time index. Afterwards the $\|\bullet\|_2$ of the twelve convolutions are calculated to form the tonal feature vector **t** for each frame. Formally

$$t_k(l) = \left\| (s(l) * R_{tone(k)})(n) \right\|_2, \quad k = 1...12 \quad (2)$$

where $s(n,l)$ is the signal frame l and $R_{tone(k)}$ the reference signal defined in Equation 1.

3.3 Convolving with the Bank of Resonators

The feature extraction process is followed by the convolution of the feature vectors with a bank of resonators. Firstly, we segment each time-evolving feature using a rectangular window of 8 secs with 1 sec overlap.

In order to compute the rhythmic periodicities of the signal, we convolve each feature segment with a bank of resonators, each resonator representing a specific tempo. We consider resonators with impulse response support that equals to the segment length for every integer value from 40 up to 280 bpm. The resonators can be any system with periodic impulse response, making our algorithm flexible to adapt to different signals. Motivated by the mathematical model of entrainment, as it was presented by Large and Kolen in [14], we adopted the basic oscillatory unit as the impulse response of the resonator. The equation of the oscillatory unit is given by

$$r(l) = 1 + \tanh(\gamma \cdot (\cos(2\pi\psi_t l) - 1)) \quad (3)$$

where ψ_t denotes the frequency of tempo *t*. Parameter γ is called the output gain.

After this process the $\|\bullet\|_\infty$ is computed for the convolution of every feature-resonator pair, resulting a vector indicating the strength of all tempi for each specific feature. Formally, we can write

$$S_{f_i}(t) = \left\| (r_t * f_i)(l) \right\|_\infty, \quad t \in T \quad (4)$$

where $S_{f_i}(t)$ is the strength of feature f_i at tempo *t* and T is the target tempi set. ($f_i \in \{\mathbf{m}_i, \mathbf{t}_i, i = 1..12\}$)

3.4 Combining the Feature Vectors

To compute the tempo for a specific segment, we summate the tempo strengths $S_{f_i}(t)$ across the features, *individually* for each feature class, thus taking two vectors $\mathbf{SC}_m$ and $\mathbf{SC}_t$, for filterbank and tonal features respectively. Formally, we can write

$$\mathbf{SC}_m(t) = \sum_{i=1}^{12} \mathbf{S}_{m_i}(t), \quad \mathbf{SC}_t(t) = \sum_{i=1}^{12} \mathbf{S}_{t_i}(t), \quad t \in T \quad (5)$$

where T is the target tempi set.

Finally, to combine the results of the two tempo detectors, we summate $\mathbf{SC}_m(t), \mathbf{SC}_t(t)$ across the segments of the music excerpt. Then by point-wise multiplication we compute the final vector **SC,** indicating the tempo strengths within the excerpt. The tempo with the maximum strength is considered as the correct tempo.

4. EXPERIMENTAL RESULTS

In this section we present the evaluation of the proposed algorithm. The data we used for the experiments consists of 1163 excerpts from the *ballroom* and *songs excerpts* datasets of the ISMIR 2004 Tempo Induction evaluation exchange. Details on the statistics, collection and annotation of the corpus can be found in [9].

Firstly, we evaluated our algorithm for each feature class individually. Afterwards we combined the outputs of the individual features as described in the previous section. The results on both *ballroom* and *songs excerpts* datasets are presented in Table 1.

	Ballroom		**Songs**	
Feature Type	**Acc1**	**Acc2**	**Acc1**	**Acc2**
Filterbank	56.34	93.33	23.01	88.39
Tonals	50.32	81.08	46.45	73.33
Combination	61.08	93.98	42.15	90.11

Table 1. Results (%) of the algorithm for the Ballroom and Songs Excerpts datasets, using feature classes individually and in combination .

It is clear that the algorithm yields better results using the filterbank features in the Ballroom dataset for the *accuracy1* measure. On the other hand, the algorithm performed poorly in the songs data based on *accuracy1* (only 23%). The above can be explained by the fact that the Ballroom data consists of more "percussive" excerpts, thus the filterbank energies represent sufficiently the data. Additionally, the experimental results demonstrate that by using solely the filterbank features, the proposed system "tends" to capture tempi double of the groundtruth tempo. For most of the excerpts classified correctly using *accuracy2* and misclassified using *accuracy1*, the detected tempo was double of the correct tempo.

When we used solely tonal features as input to the system, the *accuracy1* on the songs data increased significantly (from 23% to 46.5%) for the *Songs* data. This can be explained by the more "melodic" nature of the excerpts consisting *Songs* data, which prove tonal features to be more suitable for that case. On the other hand *accuracy2* measure degraded from 88.39% to 73.3%. A possible explanation is the large window used in the preprocessing stage, which "cuts off" frequencies double or triple of the actual tempo.

When combining the results from the two versions, we observe that in both Ballroom and Songs excerpts data, the superposition increases the algorithm performance, especially in the Songs Data. Considering the filterbank features as base features, tonal features provide additional information about the rhythm periodicities of the signal. Comparative results of the presented algorithm, namely GK, with the best five performing algorithms in [7], namely *Miguel Alonso* (AL), *Simon Dixon* (DI), *Anssi*

Klapuri (KL), *Christian Uhle* (UH) and *Eric Scheirer* (SC), plus *Klaus Seyerlehner* (SE1,SE2)[9] are presented in Table 2.

	Ballroom		Songs	
Method	**Acc1**	**Acc2**	**Acc1**	**Acc2**
GK	61.08	93.98	42.15	90.11
AL	34.1	69.48	37.42	68.6
DI	43.12	86.96	16.99	76.99
KL	63.18	90.97	58.49	91.18
UH	56.45	81.09	41.94	71.83
SC	51.86	75.07	37.85	69.46
SE1	78.51	-	40.86	-
SE2	73.78	-	60.43	-

Table 2. Comparative results (%) on *Ballroom* and *Songs* datasets.

5. CONCLUSION AND FURTHER WORK

In this paper we presented a system that extracts the tempo of a music signal. The proposed algorithm was evaluated on the benchmark corpus of the ISMIR 2004 with encouraging results. Without taking into consideration any high-level musical information, our system performed within the current state-of-the-art algorithms of the tempo induction task.

The tonal features introduced in this work prove to capture additional aspects of rhythmic periodicity in a musical signal. It is evident that underlying rhythmic periodicities of a musical signal can be found beyond the filterbank energies, in a more "pitched context". Without any multi-pitch estimation or chord detection process, we observe that the simpler and more abstract tonal features presented in this paper similar to Pitch Class Profile, contain rhythmic information that can enhance the performance of a tempo induction system that does not take into account any tonal information.

However, during the experiments we observed that the performance of the presented algorithm is sensitive to the window length and shift during the extraction process of tonal features, an effect that will be investigated in the future. Moreover we intend to extend tonal features in a more sophisticated way, such as chords, and incorporate harmonic partials information. Finally, the superposition of the output for the features classes is a subject for future research.

6. REFERENCES

[1] Gouyon F. and Dixon S., "Dance Music Classification: A Tempo-Based Approach", *Proceedings of the 5th International Conference on Music Information Retrieval,* Barcelona, Spain 2004

[2] Klapuri A., Eronen A. and Astola J., "Analysis of the Meter of Music Acoustic Signals", *IEEE Trans. Audio, Speech, and Language Processing, 14(1),* January 2006.

[3] Alonso M., David B., Richerd G., "Tempo and Beat Estimation of Musical Signals", *Proceedings of the 5th International Conference on Music Information Retrieval,* Barcelona, Spain 2004.

[4] Davies M., Plumbley M., "Context-Dependent Beat Tracking of Musical Audio", *IEEE Transactions on Audio, Speech, and Language Processing, Vol. 15, No. 3*, March 2007.

[5] Scheirer E., "Tempo and Beat Analysis of Acoustic Musical Signals.", *The Journal of the Acoustical Society of America, Vol. 103, No. 1,* January 1998.

[6] Dannenberg R., "Toward Automated Holistic Beat Tracking, Music Analysis, and Understanding", *Proceedings of the 6th International Conference on Music Information Retrieval, ,* London, UK, 2005.

[7] Gouyon F., Klapuri A., Dixon S., Alonso M., Tzanetakis G., Uhle C., and Cano P., "An Experimental Comparison of Audio Tempo Induction Algorithms", *IEEE Transactions on Audio, Speech, and Language Processing, Vol. 14, No. 5,* September 2006.

[8] Gouyon F. and Dixon S., "A Review of Automatic Rhythm Description Systems", *Computer Music Journal, 29:1, pp 34-54,* Spring 2005.

[9] Seyerlehner K., Widmer G., and Schnitzer D., "From Rhythm Patterns to Perceived Tempo", *Proceedings of the 8th International Conference on Music Information Retrieval, ,* Vienna, Austria, 2007.

[10] Ellis D.P.W. "Beat Tracking with Dynamic Programming", *Journal of New Music Research,* vol. 36 no. 1, March 2007, pp 51-60.

[11] Pampalk E., Rauber A., Merkl D., "Content-Based Organization and Visualization of Music Archives", *Proceedings of the 10th ACM International Conference on Multimedia*, Juan les Pins, France, 2002.

[12] Alonso M., Richard G., David B., "Accurate Tempo Estimation Based on Harmonic + Noise Decomposition", *EURASIP Journal on Applied Signal Processing Volume 2007, Issue 1,* January 2007.

[13] Fujishima T. "realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music".

[14] Large E. and Kolen J., "Resonance and the Perception of Musical Meter", *Connection Science 6(1), pp 177-208,* 1994.

THE STANDARDIZED VARIOGRAM AS A NOVEL TOOL FOR AUDIO SIMILARITY MEASURE

Simone Sammartino, Lorenzo J. Tardón, Cristina de la Bandera, Isabel Barbancho, Ana M. Barbancho
Dept. Ingeniería de Comunicaciones, E.T.S. Ingeniería de Telecomunicación,
Universidad de Málaga, Campus Universitario de Teatinos s/n, 29071, Málaga, Spain
ssammartino@ic.uma.es

ABSTRACT

Most of methods for audio similarity evaluation are based on the Mel frequency cepstral coefficients, employed as main tool for the characterization of audio contents. Such approach needs some way of data compression aimed to optimize the information retrieval task and to reduce the computational costs derived from the usage of cluster analysis tools and probabilistic models. A novel approach is presented in this paper, based on the standardized variogram. This tool, inherited from Geostatistics, is applied to MFCCs matrices to reduce their size and compute compact representations of the audio contents (song signatures), aimed to evaluate audio similarity. The performance of the proposed approach is analyzed in comparison with other alternative methods and on the base of human responses.

1. INTRODUCTION

The computation of the degree of similarity among songs is one of the most demanded tasks in the field of multimedia processing, and its interest is still growing with the increasing popularity of on line services and databases. Music Information Retrieval (MIR) stands for the tools to access audio contents with the aim to reorder, search and classify them [1]. In MIREX 2006 [3], the term 'Audio Similarity' was introduced for the first time in the tasks list and, consequently, a human evaluation system (Evalutron6000) was created to make quantitative evaluations of the proposed algorithms.

The main task of audio similarity evaluation is based on the definition of some form of representation of the songs (signatures) to compare them and measure the closeness of the signature songs. The base of most of the known algorithms for audio similarity evaluation are the Mel frequency cepstral coefficients (MFCCs) [10]. The spectral information supplied by the MFCCs is proposed to be compressed in a wide variety of different approaches by different authors [13] [4] [11] [1]. In this work, the standardized variogram [8] is presented as a novel tool to conveniently compress the MFCCs vectors for sorting similar songs.

The outline of the paper follows: in Section 2, a general description of the MFCCs is presented. In Sections 3 and 3.2, the details of the approach based on the variogram and its application to signal processing are described. In Section 4, the use of MFCCs matrices is presented and in Section 5 the application of the variogram is discussed in detail. Finally, in Section 6, the experimental results are presented and in Section 7, the conclusions and future proposals are discussed.

2. MEL COEFFICIENTS AND AUDIO SIMILARITY

The MFCCs are short-term spectral-based features, originally developed for speech recognition and successfully adapted to music information retrieval [10]. The computation of MFCCs follows some crucial steps [13]: 1) the calculus of the short-term spectrum of the signal, 2) the transformation of the spectrum into the Mel scale (through a triangular filter bank), 3) the calculus of the logarithm of the Mel spectrum and 4) the compression of the resulting matrix through the application of the DCT (Discrete Cosine Transform). MFCCs are widely used to generate compact spectral representations of the song: the signal is framed into short fragments (usually some tens of milliseconds) and their coefficients are computed frame by frame [13]. In order to conveniently represent the global spectral behavior of the song in a compact way, the MFCCs vectors have to be clustered. For this task, several approaches have been proposed by different authors. Pampalk [13] uses GMM and EM approach, by modelling the probability distribution functions of the coefficients vectors. Foote [4] proposes a supervised tree-structured quantizer as discriminant approach for the sequential labeling of the coefficients. Aucouturier and Pachet [1] present a combination of GMM/EM and Monte Carlo approaches to evaluate the likelihood between the MFCCs of two different songs. Finally, Logan and Salomon [11] propose the popular K-means method for MFCCs clustering. In this article, an alternative approach is proposed based on the computation of the variogram of the MFCCs, allowing for a computationally low-cost compression of the coefficients and a simple calculus of the distance among the spectral signatures.

3. THE STANDARDIZED VARIOGRAM

The term 'variogram', inherited from Geostatistics, stands for the function describing the evolution of the spatial dependence of a random field [16]. Empirically used by the mine engineer D.G. Krige in South Africa mines [9], and later formalized by G. Matheron in its pioneer works [12], the variogram or 'semivariance function' is widely employed in spatial statistics to perform uncertainty modeling in a spatial framework. It is often used as characteristic weighting function for the spatial interpolation technique known as *Kriging* [9].

3.1 Some mathematical issues

A formal definition of the variogram is now provided. Let z_α, with $\alpha = 1, \cdots, n$ represents a set of n sampled observations of a spatial phenomenon. The variogram is defined as half the variance of the increment $[z_\alpha - z_{\alpha+h}]$ [16]:

$$\gamma(\alpha, h) = \frac{1}{2} E\{[z_\alpha - z_{\alpha+h}]^2\} - \{E[z_\alpha - z_{\alpha+h}]\}^2 \quad (1)$$

Assuming the intrinsic stationarity of order two [16], the mean of the variable $E[z]$ is invariant for any translation, that is $E[z_\alpha] = E[z_{\alpha+h}]$, the second term of equation (1) can be neglected and the variance of the increment is said to be depending only on the distance vector h and not on the position α [8]:

$$\gamma(h) = \frac{1}{2} E\{[z_\alpha - z_{\alpha+h}]^2\} \quad (2)$$

where z_α and $z_{\alpha+h}$ are two different samples of the random variable z separated by a distance h.

Given a set of spatially distributed samples, the variogram can be estimated empirically [16]:

$$\gamma^*(h) = \frac{1}{2N(h)} \sum_{\alpha=1}^{N_h} [z_\alpha - z_{\alpha+h}]^2 \quad (3)$$

where the number of pairs $N(h)$ depends on the value of h. For its mathematical relation with the variance, the variogram is also known as semivariance function (or semivariogram).

The variogram is strictly related with the auto-covariance function of the increment. In particular the covariance of the increment $Cov(z_\alpha, z_{\alpha+h})$, in condition of translation invariance of the mean, can be expressed as follows:

$$Cov(z_\alpha, z_{\alpha+h}) = Cov(h) = E[z_\alpha \cdot z_{\alpha+h}] - E[z_\alpha]^2 \quad (4)$$

where $E[z_\alpha] = E[z_{\alpha+h}]$. When h is zero, $Cov(0)$ is maximum and it corresponds to the variance of the variable:

$$Cov(0) = E\{[z_\alpha]^2\} - \{E[z_\alpha]\}^2 = Var(z_\alpha) \quad (5)$$

Note that equation (2) can be written as:

$$\gamma(h) = \frac{1}{2} E\{[z_\alpha]^2 - 2 \cdot z_\alpha \cdot z_{\alpha+h} + [z_{\alpha+h}]^2\} \quad (6)$$

and using equations (4) and (5), we can express the variogram in term of the covariance function:

$$\gamma(h) = Cov(0) - Cov(h) \quad (7)$$

The last equation shows the relation between the variogram and the covariance function [14]. Under the condition of translation invariance of the mean, at $h = 0$, the covariance is just the variance of the variable, $Cov(0) = Var(z)$, and the variogram is zero, $\gamma(0) = 0$. Conversely, when the pair of elements, z_α and $z_{\alpha+h}$, are too far away to show any kind of relation, their covariance is zero and the variogram is the variance of the variable, $\gamma(h) = Var(z)$. In general, the covariance function shows a behavior opposed to the behavior of the variogram (see Fig. 1).

The empirical variogram is usually fitted by a theoretical model to obtain a continuous function, modeling the covariance exhaustively in the whole domain. The models are chosen within a group of *admissible models* that must be positive-definite [6]. Moreover, the theoretical models can be characterized by few shape parameters [6]: the *Sill*, the asymptotic variance value the function tends to when the lag distance, h, increases, the *Range*, the lag value at which the theoretical variogram reaches the sill, and the *Nugget effect*, the discontinuity of the function at the origin.

When semivariance values are normalized by the global variance, the variogram is reported as *standardized variogram* [8] and its correspondence covariance function is the correlation function. Both the empirical and its correspondent theoretical standardized variogram are shown in Fig. 1. The correspondent correlation function is shown too.

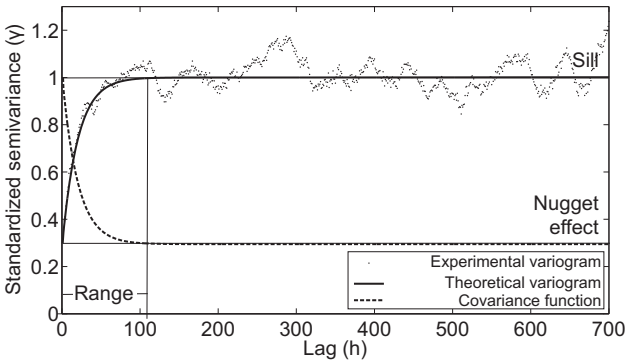


Figure 1. An example of a typical standardized variogram. The empirical variogram (dotted line) is fitted by the theoretical model (solid line). The correlation function (dashed line) is shown too.

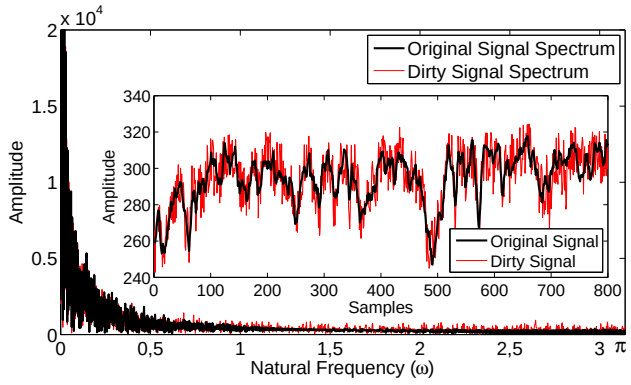
3.2 The variogram in signal processing

Many authors have dealt with the use of the variogram coupled to classical signal processing techniques, as a tool for periodicity analysis of signals and time series analysis. Khachatryan and Bisgaard [8] employ the variogram as tool for estimating the stationarity of industrial time series data. Haslett [5] proposes the use of the variogram as a functional approach for time estimation in case of fault of the stationarity conditions. Kacha et al. [7] apply the generalized variogram to the linear prediction in disordered speech analysis.

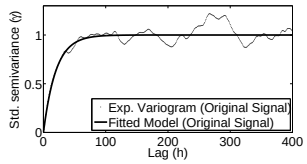
In spite of the different origins of the spatial variogra-

phic approach and the time series analysis in signal processing, the former can be successfully applied as an alternative tool for spectral analysis. In the case of time-signal processing, the parameter h is unidimensional and it represent the time lag among the samples.

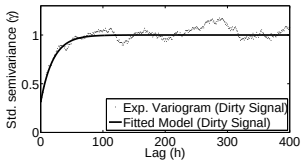
If we take a signal and we add an uncorrelated noise component with known mean and variance (see box inside Fig. 2(a)), we can observe that it is well reflected both in the waveform and in the frequency spectrum. In the variogram, the added signal leads to a very small change in the general shape of the curve (Fig. 2(b) and 2(c)), while a marked increase of the variance at the origin (nugget effect) is noticeable. Such value corresponds to a contribution of about the 31% of the total variance of the dirty signal. However, in the frequency spectrum, this change is mainly reflected in central-high frequency bands, where only a diffused increase in amplitude is apparent (Fig. 2(a)).



(a) Spectra of original (darker thick line) and dirty signal (lighter thin line). Inner box: raw signal.



(b) Standardized Variogram of original signals.



(c) Standardized Variogram of dirty signals.

Figure 2. Spectra and standardized variograms of a clean signal and of the same signal corrupted with additive noise. Experimental variograms are represented with a dotted line and the theoretical fitted model with a solid line.

The variogram can also be conveniently used as tool for sound analysis applications. Dillon et al. [2] noted as the variogram fluctuations, once reached the sill, are strictly related with signal spectrum. He remarks that the variogram can be especially useful for fundamental frequency detection, by taking into account the variance pseudoperiodic pattern known as *hole effect* [6].

4. COMPRESSION OF MATRICES

The standardized variogram described in Section 3 is proposed here as a novel method for reducing the dimensionality of the MFCCs matrices. It is computed on each vector of coefficients throughout the frames of the song, to obtain a function describing the evolution of the covariance through the time. With the aim to compress the MFCCs information, the variogram is computed only on a reduced number of lags (values of distance h for which the variogram is calculated). As shown in Fig. 1, the variogram typically presents a logarithmic-like rising behavior at the lowest lags and an asymptotic trend to the global variance (equal to one in the case of standardized variogram) from lags approaching the range, forward. Taking into account these two factors, a total amount of ten lags values are sampled logarithmically from 1 to half the length of Mel coefficients.

For each row of the MFCCs matrices, the semivariance is computed for all the pairs of samples located at distances equal to the lags selected, and the values are normalized by the variance of the MFCCs row data. The outcome is a compact function keeping enough information to characterize the signal.

The experimental standardized variogram can be characterized on the basis of two parameters extracted from its correspondent theoretical function (although the latter is not explicitly calculated in this application): the range and the nugget effect. In this case, these parameters can be interpreted on the base of their spectral meaning.

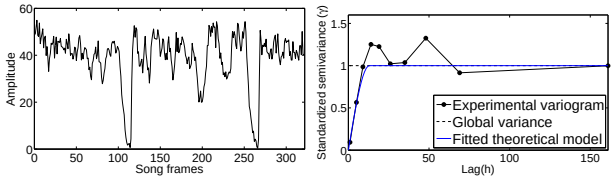
The range can be interpreted as the time scale at witch the periodicity of the signal begins to be evident. Up to the range, the structured variability of the variable masks its periodicity, while, when the pairwise covariance starts to be weak enough (from range forward), that periodic behavior rises and it becomes evident. Clearly, due to the strong reduction of lags, sometimes the range can be poorly detected by the reduced variogram.

The nugget effect is very important to understand the small-scale behavior of the Mel coefficients. The discontinuity at lag $h \rightarrow 0$ explains the variation of the signal at very small-scale. In terms of spectral analysis, it stands for the high frequency contribution to the total variance in the Mel spectra.

An example of the application of the variogram to the MFCC spectra is shown in Fig. 3. The first song (Fig. 3(a)) is a piece from the genre 'Classic' [3], its spectrum shows a rather clear periodicity, with some peculiar patterns repeating with a certain regularity. The second song, belonging to the genre 'Heavy metal', shows a more fuzzy spectrum with higher frequency variations and a less evident periodicity. Such differences are well reflected in their correspondent variograms.

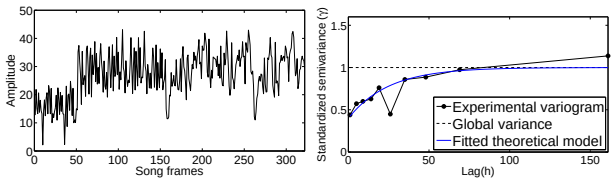
The clearer regularity of the classic piece is reflected by a certain degree of periodicity in the variogram (although not exhaustively revealed by the reduced number of lags). Moreover, the very low nugget variance value reflects the high degree of regularity with reduced high frequency oscillations (Fig. 3(b)).

In the case of the heavy metal piece, the very high frequency oscillations (Fig. 3(c)) are well reflected by a notable nugget variance (about the 50% of the total variance) and by a larger range indicating the lack of a structured



(a) The second Mel coefficient of a classical music song.

(b) Standardized reduced variogram of the signal in fig. 3(a) (dotted line).



(c) The second Mel coefficient of a heavy metal song.

(d) Standardized reduced variogram of the signal in fig. 3(c) (dotted line).

Figure 3. A comparison between the variograms of the MFCCs spectra of two very different songs, a classical music song (top) and a heavy metal song (bottom). Fitted theoretical model (thick lighter line) and global variance (dashed line) are shown too. The pieces are 35 seconds length.

variability. Note that, although poorly reflected by the reduced availability of lags, some degree of non-stationarity, expressed as the lack of a well definite asymptoticity of the variogram [16], is present in this case.

5. STANDARDIZED VARIOGRAM FOR AUDIO SIMILARITY ASSESSMENT

As mentioned before, the variogram has been employed for audio similarity assessment. For each piece, the Mel coefficients are calculated and the standardized variogram is computed for each coefficient, obtaining a compact signature of the track. Successively, the signatures are compared, by computing a weighted difference of their elements.

After computing the standardized variogram (10 lags) of the 19 Mel coefficients for each song (the first one has been neglected [13]), the resulting 10×19 matrices (signatures) are compared by averaging the weighted absolute value of their difference, according to the following equation:

$$D = \frac{1}{I \cdot J} \sum_{j=1}^{10} \sum_{i=1}^{19} \omega_j \left| V_a(i,j) - V_b(i,j) \right| \qquad (8)$$

where V_a and V_b are the signature matrices for song a and b, respectively and the indexes i and j are referred to the 19 coefficients and the 10 lags, respectively. Differences are linearly weighted in order to give more importance to the small-scale lags of the variogram vectors. The vector $\Omega = [\omega_1, \ldots, \omega_{10}]$ contains the 10 linearly decreasing weights ω_j, such that $\sum_{j=1}^{10} \omega_j = 1$. Each j-th weight is computed as follows:

$$\omega_j = \frac{11 - j}{D} \qquad (9)$$

where $D = \sum_{j=1}^{10} j$. Audio similarity is simply evaluated by sorting the songs with respect to a reference piece, according to their reciprocal distance, computed using equation (8).

6. EXPERIMENTAL RESULTS AND DISCUSSION

An objective evaluation of the sorting capability of the method is very hard to achieve because of the subjectiveness of the concept of 'audio similarity'. Actually, one song can be judged as more similar to another one depending on a series of parameters (rhythm, spectral content, melody etc.) that are subconsciously evaluated by the listeners.

In order to obtain a robust and objective estimate of the performance of the method, a series of tests performed by a group of users, have been carried on. A total of 5 lists of songs have been submitted to 10 users who sorted them with respect to a set of reference songs, without any previous knowledge about any tagging or taxonomy of the dataset. The test songs are sampled by the Audio Description Context database of the ISMIR2004 [3] and belong to all the genres presented in the database.

Successively, the lists created manually have been compared with the outcomes of 4 automatic methods, the variogram-based method and other three methods that can be found in the literature:

1. Fluctuation Patterns [13]
2. MFCCs with GMM/EM clustering approach [13]
3. MFCCs with K-means clustering approach

The fluctuation patterns, describing the amplitude modulation of the loudness of the frequency bands, are used by Pampalk [13] to briefly characterize the song spectral content. The Gaussian Mixture Models coupled with Expectation/Maximization approach are employed by the same author to cluster the Mel coefficients in 30 vectors (G30) of 19 elements. The third method is the same approach used by Logan and Salomon [11], based on the calculus of the MFCCs clustered by the popular K-means, with the Euclidean distance instead of the Kullback-Leibler distance.

A total amount of some tens of lists have been obtained by the manual sorting by the users. A rapid look at these lists reveals a strong lack of homogeneity among them. It is related to the high subjectiveness of the sorting process and the variability of the human perception of the 'audio similarity'. This leads to the lack of a representative list for each reference song. Instead of trying to extract a unique reference list among the users, the authors turned to derive a measure of the agreement among the users.

A weighted matching score has been computed, taking into account the reciprocal distance of the songs (in terms of position index in the list). Such distances have been linearly weighted, such that the first songs in the lists reflected more importance than the last ones. Actually, it is easier to

define the order of few very similar songs, than to sort the very different ones.

Let L_α and L_β represent two different lists of n songs, for the same reference song, the matching score S has been computed using the following equation:

$$S = \sum_{i=1}^{n} |i - j| \cdot \omega_i \tag{10}$$

where i and j are the indexes for lists L_α and L_β, respectively. In particular, j is the index of the j-th song in list L_β, such that $L_\alpha(i) \equiv L_\beta(j)$. In practice, the i-th song in the list L_α is searched in L_β and their correspondent indexes are compared. The absolute difference is linearly weighted by the weights ω_i as referred in equation (9).

Finally, the scores are transformed to be represented as percentage of the maximum score attainable.

For each reference song, the matching scores have been computed among all the available lists, both among the users lists and among the users lists and the ones returned by the automatic methods. Thus, two different sets of scores have been obtained: the inter-users scores and the users-automatic scores. The measure of the performance of the automatic method is drawn by the degree of similarity of the two sets, that is, how close are the scores computed among the users lists and the lists returned by the automatic method. In order to have an estimation of such closeness, the coherence among the two sets of scores is computed by a statistical test. The Kolmogorov-Smirnov test [15] has been used to measure the correspondence between the two distributions of the two sets of scores, before and after the inclusion of the automatic list.

In Table 1, the basic statistics for both the distributions of the inter-users scores set and the users-automatic scores sets are shown. The results of the statistical test (H) is shown too.

The degree of similarity among the songs is a very subjective response and only a high number of cases can guarantee a reliable response. Nevertheless, the statistical results are enough to have an idea of the performance of the automatic methods. The response of the users can be seen as some form of quantifying the difficulty level of the sorting task. When the songs are easily sortable, the users show a high degree of agreement (high mean scores). Conversely, when the similarity among the songs is not very clear, the discrepancy among the users increases and, together with a decrease of the centrality measures (mean and median), an increase of the variance is appreciable. Actually, the standard deviation is an index of the disagreement among the users and can be related with the complexity of the sorting procedure.

In the test results, the discordance among the users is well reflected by high values of the standard deviation in most of the cases. The mean standard deviation for the 5 cases is about the 14% of the mean score.

In general, a wide variety of performances are shown by the different methods. The method based on the clustering of the Mel coefficients by the GMM/EM approach reaches the best score in 3 cases, for songs B,C and D, while it fails the test for song B. The method based on the clustering of the Mel coefficients by the variogram returns the highest scores for songs A and E. It also returns the second highest score for song B, although failing the test, but with the highest p-value (not shown in the table).

Ref.song	Method	Mean	Median	Min	Max	Skewness	St.Dev.	H
Song A	**Users**	72.3	74.9	33.4	90.9	-1.0	13.2	-
	MFCC-Var	**71.5**	**75.2**	42.8	82.5	-1.8	11.3	0
	FP	71.2	72.5	**43.5**	**85.3**	-1.3	11.6	0
	MFCC-EUC	70.8	71.8	41.8	84.8	-1.4	12.1	0
	MFCC-G30	55	54.3	41.3	69.4	0.1	**9**	1
Song B	**Users**	81.8	83.9	52.7	99.2	-1	9.6	-
	MFCC-Var	75.4	76.8	54.4	87.8	-1.2	8.7	1
	FP	66.6	66.1	58	81.8	0.9	6.9	1
	MFCC-EUC	67.5	66.6	**61.5**	72.9	0.1	**4.3**	1
	MFCC-G30	**76.7**	**82.3**	41.5	**99.2**	-0.7	14.2	1
Song C	**Users**	84.3	85.6	66.6	96.2	-0.2	6.6	-
	MFCC-Var	71.3	70.8	66.1	76.2	0.2	**3.5**	1
	FP	70	69.9	58	82.8	0.1	6.5	1
	MFCC-EUC	81.8	83.2	71.4	90.9	-0.4	6.1	0
	MFCC-G30	**86.9**	**89.1**	**73.7**	**92.9**	-1.1	6.2	0
Song D	**Users**	77.2	77.7	57.5	96.2	0	8.2	-
	MFCC-Var	60.8	60.1	57.2	71.1	1.4	**4.5**	**1**
	FP	57.3	54.6	50.9	73.4	1.3	7	1
	MFCC-EUC	66.1	65.1	59.7	84.6	1.9	7	1
	MFCC-G30	**77.8**	**83.0**	**63.0**	**86.6**	-0.5	8.8	0
Song E	**Users**	65.7	66.1	22.8	93.2	-0.4	15.4	-
	MFCC-Var	**67.8**	**68.5**	**56.2**	**82.3**	0.1	**8.1**	**0**
	FP	59.2	60.6	42.3	70.1	-0.5	9.8	0
	MFCC-EUC	34.1	34.2	11.6	62	0.1	15.4	1
	MFCC-G30	49.9	49.4	30.9	63.3	-0.4	9.7	1
Mean	**Users**	76.3	77.6	-	-	-	10.6	-
	MFCC-Var	69.4	70.3	-	-	-	**7.2**	-
	FP	**64.9**	64.7	-	-	-	8.4	-
	MFCC-EUC	64.0	64.2	-	-	-	9.0	-
	MFCC-G30	69.3	**71.6**	-	-	-	9.6	-

Table 1. Basic statistics of the distributions of the inter-users scores set and the users-automatic scores set. Values are in percent. Results of statistical test are shown too: $H = 0$ means that the two distributions are coherent, while $H = 1$ stands for a distributions mismatch. The codes for the automatic methods stand for: MFCC/Var = MFCCs clustered by standardized variogram, FP = fluctuation patterns, MFCC-EUC = MFCCs clustered by K-means, MFCC-G30 = MFCCs clustered by GMM/EM method. Best results in bold.

The best results are attained for the song A, where three of the four methods pass the test, while, for song B, none of them return a sufficient matching with the inter-users distribution. This last issue is basically related with the high agreement shown by the users (about 82%) that is hardly attained by the automatic methods. Quite the same situation occurs for song C and D, with high mean scores among the users lists (more than 84% and 77%, for songs C and D, respectively) approached by only two of the four methods proposed. Finally, the song E reveals a very low mean inter-users score (about 66%), well reflected by all the methods. Globally, all the methods show a good perfor-

mance, with averaged mean values higher than 64%. The variogram-based approach shows the highest mean value, with 69.4%, very close to the result by the GMM/EM based method.

7. CONCLUSIONS AND FUTURE WORKS

A new approach based on the use of the standardized variogram for the clustering of the Mel coefficients for audio similarity evaluation has been proposed. The variogram is calculated on a reduced vector of ten lag elements and it is standardized by the global variance, in order to obtain comparable signature matrices for different songs. The method capability is evaluated on the base of a statistical comparison among the distributions of the matching scores computed among a set of users lists and the ones returned by the automatic method. Moreover, for a more complete assessment of the method performance, other three known methods employed in literature for audio similarity evaluation are computed, and their correspondent scores are compared.

Performances vary from quite poor to very good for all the methods, with mean matching scores varying from the lowest mean value of about 34% for the method based on the Euclidean distance and the highest value of about 87% for the method based on the clustering by GMM/EM. The averaged mean values reveal a good global performance of the method based on the variogram and on the GMM/EM approach, with quite poorer results by the other two ones. In practice, the variogram-based method proposed here works quite well and its performance can be compared with the one of other more popular methods that, in some cases, show a higher degree of computational complexity.

The capability of the method can be improved, by optimizing some calculation parameters, as the sampling of the distance lags values. Moreover, the theoretical variogram can be evaluated and its shape parameters can be taken into account to optimize the modeling of the spectral content of the song to improve the audio similarity assessment task. The evaluation task can be improved by increasing the number of users and broadening the test samples.

8. ACKNOWLEDGMENTS

This work was supported by the Ministerio de Educación y Ciencia of the Spanish Government, under Project No. TSI2007-61181, by the Ministerio de Industria, Turismo y Comercio of the Spanish Government, under Project No. TSI020501-2008-0117 and by the Junta de Andalucía, under Project No. P07-TIC-02783.

9. REFERENCES

[1] Jean-Julien Aucouturier and Francois Pachet. Music similarity measures: What's the use ? http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.5609, 2002.

[2] C. G. Dillon, C. Lloyd, and L. Philip. Identifying short-range and long-range structural components of a compacted soil: an integrated geostatistical and spectral approach. *Computers and Geosciences*, 29:1277–1290, December 2003.

[3] Stephen J. Downie. The music information retrieval evaluation exchange (mirex). http://www.dlib.org/dlib/december06/downie/12-downie.html.

[4] Jonathan T. Foote. Content-based retrieval of music and audio. In *Multimedia Storage and Archiving Systems II, Proc. of SPIE*, pages 138–147, 1997.

[5] John Haslett. On the sample variogram and the sample autocovariance for non-stationary time series. *The Statistician*, 46(4):475–485, 1997.

[6] Edward H. Isaaks and Mohan R. Srivastava. *An Introduction to Applied Geostatistics*. Oxford University Press, USA, January 1990.

[7] A. Kacha, F. Grenez, J. Schoentgen, and K. Benmahammed. Dysphonic speech analysis using generalized variogram. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 1, pages 917–920, 2005.

[8] Davit Khachatryan and Sren Bisgaard. Some results on the variogram in time series analysis. *Quality and Reliability Engineering International*, March 2009.

[9] Daniel G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, 52(6):119–139, December 1951.

[10] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *In International Symposium on Music Information Retrieval*, 2000.

[11] Beth Logan and Ariel Salomon. A content-based music similarity function. Technical report, Processing Languages – Document Style Semantics and Specification Language (DSSSL). Ref. No. ISO/IEC 10179:1996(E), 2001.

[12] George Matheron. The theory of regionalized variables and its applications. *Les cahiers du CMM de Fontainebleau*, 5, 1971.

[13] E. Pampalk. *Computational Models of Music Similarity and their Application to Music Information Retrieval*. PhD thesis, Vienna University of Technology, Vienna, March 2006.

[14] S. Sammartino. *Geostatistical models for environmental datasets*. PhD thesis, University of Napoli "Federico II", Napoli (Italy), March 2006.

[15] M. A. Stephens. Edf statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69(347):730–737, 1974.

[16] Hans Wackernagel. *Multivariate Geostatistics: An Introduction With Applications*. Springer-Verlag Telos, January 1999.

THUMBNAILDJ: VISUAL THUMBNAILS OF MUSIC CONTENT

Ya-Xi Chen
Media Informatics, University of Munich
Amalienstr. 17, 80333 Munich, Germany
yaxi.chen@ifi.lmu.de

René Klüber
Media Informatics, University of Munich
Amalienstr. 17, 80333 Munich, Germany
klueber@cip.ifi.lmu.de

ABSTRACT

Musical perception is non-visual and people cannot describe what a song sounds like without listening to it. To facilitate music browsing and searching, we explore the automatic generation of visual thumbnails for music. Targeting an expert user groups, DJs, we developed a concept named ThumbnailDJ: Based on a metaphor of music notation, a visual thumbnail can be automatically generated for an audio file, including information of tempo, volume, genre, aggressiveness and bass. We discussed ThumbnailDJ and other 3 selected concepts with DJs, and our concept was preferred most. Based on the results of this interview, we refined ThumbnailDJ and conducted an evaluation with DJs. The results confirmed that ThumbnailDJ can facilitate expert users browsing and searching within their music collection.

1. INTRODUCTION

People can easily gain an overview of a photo by glimpsing at its thumbnail. With assistance of multiple thumbnails, people can browse many photos in parallel and locate the desired ones quickly. On the contrary, music carries no visual information and people cannot describe what a song sounds like without listening to it. Cover art is commonly used as a visual assistance for music. However, it only visually encodes the relevant artist and album, and has no reflection on the intrinsic music content. Will a visualization of musical content help, and who will benefit from it? To answer these questions, we explore the automatic generation of visual thumbnails for music content and develop a concept named ThumbnailDJ. We conducted several rounds of survey and interview, and the results confirmed that our concept can help expert users browsing and searching within their music collections.

2. RELATED WORK

A map-based representation is widely used to display music collections. Islands of Music [6] and MUSICtable [9] cluster songs based on their acoustic similarity. In Artist Map [13] the user can chose any two of the four criteria of mood, genre, year and tempo to display music on a map. In iCandy [3], songs are displayed in a grid layout and their order is determined by the selected criteria such as genre, most played artist or album. Besides map-based representation, there are other visualizations of music collections. Torrens et al. [10] presented three visualization concepts: a disc, a rectangle and a tree. In MusicRainbow [7] artists are displayed in rings of a rainbow.

The aforementioned visualizations focus on the representation of an entire music collection, in which single songs are either displayed with the cover art or as the name of artist, album or song, and none of them reflect the intrinsic music content. Some work addressed this issue by producing visualization for the file content. Semanticons [8] produce semantically meaningful icons for different file types. Music Icon [5] is a similar concept using a blossom metaphor: Each music file is represented as a blossom icon with two rings of petals (see Figure 1d for a simplified version). The music feature is reflected by the color, shape and number of petals.

Besides these metaphoric visualizations, some researchers focus on the sequential representation. Beat Histogram [11] is a temporal representation of beat strength in audio signals (see Figure 1a). It helps to gain an overall impression of how beat strength changes over time. TimbreGrams [12] represents an audio file as sequential stripes. Bright colors correspond to speech or singing, and purple and blue ones associate to classical music (see Figure 1c). In Arc Diagrams [14], instances of the identical notes are connected by arcs, thus depicting the repetitive structure in a music file (see Figure 1b).

3. ONLINE SURVEY

In order to assess the understandability and suitability of the concept of visualizing music content, we conducted an online survey with four selected concepts (see Figure

1): Beat Histogram, Arc Diagram, TimbreGrams and simplified Music Icon.

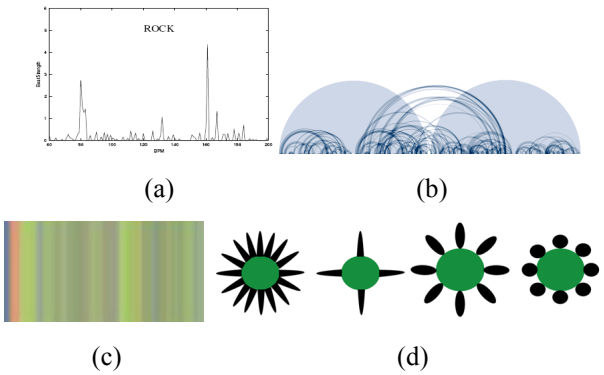


Figure 1. Four selected concepts. a) Beat Histogram [11]. b) Arc Diagram [14]. c) TimbreGrams [12]. d) A simplified Music Icon has one ring of petals. The number and shape of petals represent tempo and aggressiveness respectively.

3.1 Survey Design

The participants first filled out a questionnaire about their personal information and general experience with music. Then the four visualization concepts were briefly introduced. The participants answered 15 questions about these concepts: We chose 8 popular songs and asked the participants about their familiarity with these songs. They could follow the corresponding links to listen to these songs online. For each concept, they were required to map one visualization to a correct song out of 4 candidates, then map one song to a correct visualization out of 4 candidates. After this, they were asked about information that can be derived from each concept. Then we asked them about their preference between and comments about these concepts. All scores were rated on a 5-point Linkert-scale where 5 represented the highest score.

3.2 Participants

In total we received 38 complete questionnaires, 9 female and 29 male. Their age ranged from 18 to 55 with an average age of 26.6 years. 31 out of 38 participants were students and employees from Europe.

3.3 Results

Regarding the reflection on music content, the participants thought attributes such as melody, mood, rhythm, instrument and genre were more important than the general information of lyrics, length and release year. Concerning the usefulness of different information in helping gaining an overall impression of a song, the 30-second preview clip received the highest score (M=3.84, SD=0.17). Similar artists/tracks (M=3.16, SD=0.19) and top tags (M=2.78, SD=0.19) were scored lower. Unfortunately, our concept of visual thumbnails of music content was rated lowest (M=2.22, SD=0.16).

Although the participants were generally familiar with the tested songs (M=3.50, SD=0.74), the correctness of their answers was quite low: To map a visualization to a correct song out of 4 candidates, 14 participants (36.8%) chose the correct song for Music Icon, 12 (31.6%) for TimbreGrams, 9 (23.7%) for BeatHistogram and 8 (21.1%) for Arc Diagrams. To map a song to a correct visualization out of 4 candidates, the performance was slightly better but still low: 17 (44.7%) for Music Icon, 16 (42.1%) for BeatHistogram, 10 (26.3%) for Arc Diagrams and 8 (21.1%) for TimbreGrams.

Concerning the information that can be derived from each concept, the results illustrated that BeatHistogram helped to learn tempo and volume, and Music Icon tempo and volume. TimbreGrams and Arc Diagrams facilitated gaining song structure and information of harmony. In general, the scores for easiness of deriving each information were rather low (all below 3). The usefulness of each concept was also rated quite low: BeatHistogram (M=2.17, SD=1.40), Arc Diagrams (M=2.09, SD=1.17), Music Icon (M=1.97, SD=1.32) and TimbreGrams (M=1.40, SD=0.55).

3.4 Discussion

All the participants commented that these visualizations were overall too complicated for them. Although they agreed that attributes such as melody and rhythm are important features of a song, they did not think such information can help them to gain the overall impression of a song. Instead, they would prefer direct and non-trivial assistance, such as 30-second preview clips.

4. CONCEPT DEVELOPMENT

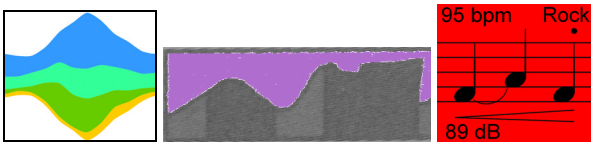
The results of the survey revealed an overall low appreciation for the concepts of visualizing musical content. The normal music listeners seem not be unsuitable as the consumers of those thumbnails, as they have neither requirements nor efficient knowledge in understanding technical details of a song. Therefore, we shift our focus to more professional users, Disk Jockeys (DJs). We develop a concept named ThumbnailDJ. We discussed ThumbnailDJ and 3 other concepts with DJs. ThumbnailDJ was preferred most, and we derived implications for the refinement of this concept.

4.1 Four Tested Concepts

We first conducted a preliminary discussion with 7 DJs with the four concepts presented in Figure 1. They were overall appreciated the idea of visualizing musical content. Arc Diagram and TimbreGrams were commented as helpful to gain an overall impression of a song, but lacking of precise values, such as those shown in Beat Histogram. Beyond the single criterion displayed in Beat Histogram, more attributes were requested, such as aggres-

siveness and volume. Music Icon was generally preferred most among these four tested concepts.

The results of the preliminary test illustrated that expert users require multiple attributes with precise values, and simplicity is vital for mental perception [1]. Therefore, we selected four simple and compact visualizations and tested them with DJs. Havre et al. [4] introduced a river metaphor to represent topical changes within a document collection. We employ this concept to describe temporal changes of attributes (see Figure 2a). Border Community[1] offers hand-drawn graphs to show the composition of songs. The amplitude depicts intensity and gray color represents bass strength. We name this visualization TensionDiagram and map its background color to genre (see Figure 2b). The third concept is Music Icon, which has already been tested in the previous online survey (see Figure 1d).



(a) (b) (c)

Figure 2. Tested concepts with DJs. a) ThemeRiver [4]. b) TensionDiagram[1]. c) Initial concept of ThumbnailDJ.

Besides these three selected visualizations, we develop our own concept ThumbnailDJ (see Figure 2c). Our design is built on the metaphor of music notation, the most common symbolic representation of music, from which abundant information can be read out: Pitch is shown as the vertical position of notes on a five-line staff. Duration is illustrated as the note value and additional symbols such as dot and tie. A dot extends the value of a note and a tie connects two notes with same pitch. Tempo and dynamics are shown above or below the staff. Tempo is normally represented as Beats per Minute (BPM), and dynamics as the overall volume of the whole piece. By reading the staff from left to right, the overall temporal impression can be gained.

In our initial concept, we made some modification of the original metaphor. The lengthy notation contradicts to the fundamental characteristic of compactness, and thus we decided to employ only three notes, each representing 1/3 of the song. The vertical location of a note in the staff depicts the aggressiveness. Two notes are connected by a tie if they share similar aggressiveness. On the contrary, a dot on the top of a note stands for change in aggressiveness. Tempo and volume are shown in the top and bottom left corners respectively. Crescendo (<) and decrescendo (>) describe the increase/decrease of volume. The genre is shown in the top right corner and associated with the background color of the entire graphics. Figure 2c shows a medium-tempo calm rock song which gets louder over time. Aggressiveness keeps constant in the first two parts and changes in the third part.

[1]http://www.bordercommunity.com

4.2 Discussion with DJs

In order to test the suitability of the four selected concepts, we conducted a second round of discussion with the same 7 DJs. The open questions covered mainly their routine tasks and general impression of the tested concepts, which were introduced in a blind fashion. The participants have DJ experience about 10 years in average. They are all male, and their age range from 24 to 37 with an average age of 28.4 years. Two of them play mainly Hip-Hop music, three play Electronic, and the other two play diverse genres. Two DJs play analogue music and the others play digital. All DJs can read music notation.

Analogue DJs organize their collections on shelves, either sorting them by alphabet or genre, or no ordering at all. Digital DJs store their collections on hard disk and sort them by folders and ID3tags. They rely heavily on the search functionality in DJ software to look for music.

Concerning the desired music attributes, DJs playing multiple genres required a general impression of a song. DJs playing only one or few genres requested detailed temporal information. The generally important attributes were genre, tempo, aggressiveness and volume.

The idea of visualizing music content was overall appreciated. TensionDiagram was well accepted, as it is similar to the signal histogram shown in most of the DJ software. Consistent with the preliminary test, Music Icon and ThemeRiver were commented as lacking of precise values. ThumbnailDJ was preferred most, as it uses the general metaphor of music notation and reflects both overview and precise information for the most desired attributes of genre, tempo, aggressiveness and volume. The participants' comments indicated further improvement of ThumbnailDJ. Representing the entire piece of song as three separated parts was commented as simple and easy to learn. But a continuous flow description was desired, which helps to gain temporal changes and facilitate observation of the representative parts, such as peaks and gaps. Bass value was desired, as it helps to achieve smooth transition between songs. The symbols of crescendo, decrescendo, dot and tie were commented as less important and poor readable in a size-limited thumbnail.

5. CONCEPT REFINEMENT

Based on the implications derived in the discussion with DJs, we refined ThumbnailDJ (see Figure 3a): We excluded the symbols of crescendo, decrescendo, dot and tie. We associated pitch (namely the vertical location of a note in the staff) with bass value. For example, a note on the bottom line implies a higher bass value. The bass value is also represented by the position of note head: A note with head on the bottom represents higher bass

value, and head on the top lower bass value. Aggressiveness is depicted as note value, for example, half note for calm and sixteenth for aggressive (see Figure 3b). A light gray flow is drawn under the staff, indicating changes of aggressiveness. Genre is displayed in the top right corner and also represented as the background color of the entire graphics. Tempo is represented as BPM in the top left corner, and volume as decibel (dB) in the bottom left corner. Currently we define 6 main categories of genres (see Figure 3c), and more genres can be easily included. Figure 3a shows an example thumbnail for a rock song, with average tempo of 108 BPM and volume of 94 dB. It is quite aggressive and very heavy on bass in both beginning and end parts. The background flow illustrates that aggressiveness descends constantly in the middle part, rises again and becomes quite fluctuate in the last part.

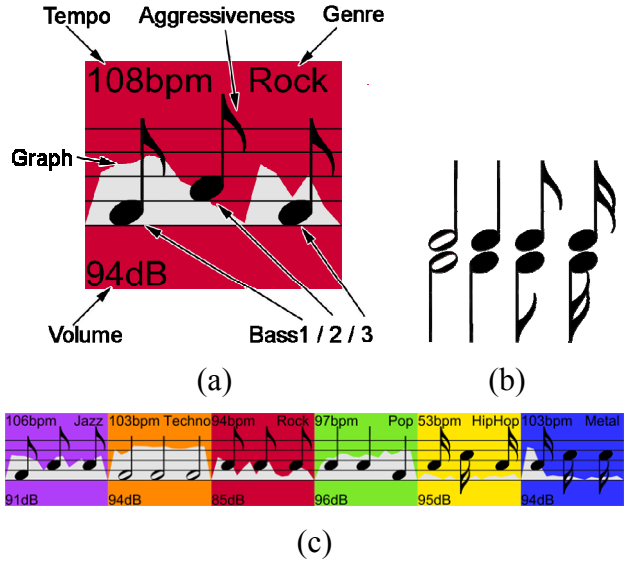


Figure 3. The refined ThumbnailDJ. a): Rock song" Biffy Clyro-As dust dances". b) 8 kinds of notes. The values of aggressiveness and bass are both mapped to a discrete value between 1 and 4. The note with head on the bottom represents bass value of 3 or 4, and on the top 1 or 2. Half to sixteenth notes indicate aggressive values from 1 to 4 respectively. c) Some example thumbnails for different genres.

5.1 Implementation

ThumbnailDJ is implemented in Java. All songs and their relevant data are saved in a SQLite database. We use Tritonus[1] to read in the ID3tags of a song. JLayer[2] is used to decode the audio file. After comparing the performance of different audio features, we decided to use zero crossings and Fast Fourier Transformation (FFT) to calculate values of aggressiveness and bass. We first use mp3splt[3] to cut the audio file into 15 snippets with equal length. We then apply jAudio[4] to extract the corresponding low-level features from each snippet. We compute the value of zero crossings over each snippet. The average value of each successive 5 snippets represents aggressiveness for each 1/3 part of the song. Bass is determined by the FFT frequencies. For each snippet, the frequencies are sorted from low to high. The bass value is represented as the sum of lower 1/4 frequencies divided by the sum of all frequencies. The average value of each successive 5 snippets represents bass value for each 1/3 part of the song. Both aggressiveness and bass are normalized to a discrete value between 1 and 4, in order to map them to one of the 8 note shapes (see Figure 3b). Volume is determined by the average value of Root Mean Square (RMS) and tempo by the average value of beats in the beat histogram.

6. EVALUATION

We conducted a user study with DJs to evaluate the performance of ThumbnailDJ. We were specifically interested in how it helps gaining an overall impression of a song, and facilitating browsing and searching in a music collection.

6.1 Settings and Procedure

As DJ equipments were required, the evaluation was conducted in the work places of the participants. Each participant was asked to offer a collection of 100 songs. For each collection, the thumbnails were generated before hand and shown in Windows Explorer in another laptop. On average the user study lasted about 90 minutes per participant. It was recorded on video using the Think-Aloud protocol was applied. All scores were rated on a 5-point Linkert-scale where 5 represented the highest score.

Since the participants already joined the former interview and they preferred ThumbnailDJ most, in this evaluation we focused on ThumbnailDJ and did not compare it with other concepts. After a brief introduction of refined ThumbnailDJ, the participants were asked to describe their impression of two unfamiliar songs by viewing the corresponding thumbnails. Then they were shown thumbnails of two familiar songs and asked to rate how well these thumbnails describe these songs. After that, they executed a routine task with their own methods (two participants with analogue music and the other three with digital) and through browsing the corresponding thumbnails respectively: Finding some appropriate songs as intro, bridge and outro respectively for an X (the genre the participant often plays) party. The order of their own methods and ThumbnailDJ was counterbalanced between the participants to minimize learning effects. After com-

[1]http://www.tritonus.org/
[2]http://www.javazoom.net/javalayer/javalayer.html
[3]http://mp3splt.sourceforge.net/mp3splt_page
[4]http://jmir.sourceforge.net/jAudio.html

pleting all tasks, they filled out a questionnaire concerning their overall impression of ThumbnailDJ.

6.2 Participants

We recruited 5 DJs, who took part in the earlier discussions. Their age ranged from 24 to 31, with an average age of 27.4 years. They are all experience DJs with average experience about 10 years. Two participants play mainly Electronic music, two play Hip-Hop, and the other one plays multiple genres. Two participants play analogue music and the other three digital. All participants can read musical notation.

6.3 Results

The meaning of attributes shown in the thumbnails was clear to all participants. They claimed that these thumbnails reflected efficiently the features of the tested songs. However, the combination of multiple attributes was still insufficient to help gaining the sense of an unfamiliar song. The participants were reluctant to "guess" the feeling of an unfamiliar song without listening to it. ThumbnailDJ was thus considered rather useful as a quick visual reminder of a familiar song. Some additional attributes were desired, such as vocal and instrument.

Song selection was overall very subjective, which was also influenced by the context, such as the audience feedback, the performance duration, the order of DJs in the same show and music played by the previous DJ. Songs played as intro were characterized as moody and relaxing, and thus slow and calm songs were selected. Outro songs were similar to the intros, and those with similar tempo were composed in a block. Bridge song should fit the tempo and bass intensity of the connecting songs.

Using their own methods and ThumbnailDJ respectively, there were on average 5 songs selected in each session. Among these selections each DJ picked 2 to 3 same songs and other songs were quite similar. When asked about their selection criteria, the participants pointed out the decisive factors such as song attributes and other contextual considerations. However, they could not formulate formally why they chose a specific song, as "music is kind of sense that can not be precisely described" (DJ 6). Therefore, with the general open tasks of song selection, we could not collect details about how ThumbnailDJ assisted search and browsing in certain aspects.

Considering the completion time, digital DJs were overall faster. Their tag-based search was comparable with browsing thumbnails. Analogue DJs were slightly slower by flipping through the analogue collections. They claimed the time difference would become more noticeable with a larger collection.

The concept of visualizing music content was generally appealing (M=4.20, SD=0.84). The impression of ThumbnailDJ was quite positive in the aspects of ease of use (M=4.0, SD=1.40), learnability (M=4.80, SD=1.84) and understandablity (M=4.20, SD=1.27). Enjoyment was rated lower (M=3.40, SD=0.63). Most participants believed that the performance of ThumbnailDJ was promising, but needed more graphical and acoustical appealing effects. All participants expressed high willingness to have ThumbnailDJ as plug-in in their DJ software, (M=4.80, SD=0.45).

6.4 Discussion

The final feedback was quite encouraging, and we received valuable implications for further improvement. Besides the included attributes of tempo, volume, genre, aggressiveness and bass, some additional information is desired, such as vocal and instrument, which are important for a smooth transition between songs. Currently, attributes are directly extracted from low-level features, and more elaborate mapping algorithms should be integrated to achieve a better association between low-level features and high-level perception. Music taste is subjective and different users may have different requirements. Besides, it is not practical to display too much information in a compact thumbnail. Therefore, we suggest employing a personalization mechanism, and thus the user can produce personalized thumbnails, for example, by defined the desired attributes shown in the thumbnail.

Concerning the evaluation method, the collected data was mainly qualitative. Detailed information about the performance of ThumbnailDJ could not be collected with the open tasks. To derive deeper understanding of how such visualization facilitates browsing and decision making, more controlled tasks should be considered. With a possible integration in existing DJ software, a field study in a real DJ working environment will help to gain more insights on the practical usage of such a tool.

7. CONCLUSION AND FUTURE WORK

We explore how to facilitate browsing and searching within music collections with the assistance of visual thumbnails of music content. Based on the metaphor of music notation, we developed ThumbnailDJ to generate thumbnails for music content, which include information of tempo, volume, genre, aggressiveness and bass. We discussed ThumbnailDJ and other 3 selected concepts with DJs, and our concept was preferred most. We then refined ThumbnailDJ and conducted an evaluation with DJs. Our concept received overall positive feedback, especially towards its helpfulness for quickly recalling of a familiar song. Moreover, DJs showed high willingness to have such a plug-in in their DJ software.

We discussed with a senior developer of Traktor[1] the potential integration of ThumbnailDJ into their DJ software. The feedback was quite encouraging. The different

[1] http://www.native-instruments.com/en/products/dj/traktor-pro

representations of the same information, such as genre, bass and aggressiveness, were especially appreciated. Considering a possible integration into a commercial product, some modification of ThumbnailDJ is necessary, for example, displaying more meaningful information for tempo and volume beyond the currently discrete values.

Besides DJs, online audio searchers might be another potential user group. For example, in iStockPhoto[1], to determine which audio file to buy, the user has to listen to many retrieval results, which is quite time-consuming. We expect that browsing and decision-making can be facilitated by assisting the audio files with visual thumbnails of their content. Then the user can exclude unmatching files quickly, or scan all thumbnails in parallel while looking for candidates satisfying certain graphical features. We discussed ThumbnailDJ with 10 online audio searchers. They were offered a set of 20 thumbnails, covering diverse genre, tempo, volume and aggressiveness. They first categorized these thumbnails and sorted them in each category. They then selected an audio file for a coffee advertisement. All participants could manage these tasks and commented the assistance of ThumbnailDJ as helpful.

We agree that "Music is more of an art than science" [2]. We also believe that visualizations can help people gaining more musical insights. We hope our exploration can shed some light on facilitating browsing and searching within music collections by bridging the perceptions of vision and acoustics.

8. ACKNOWLEDGEMENT

This research was funded by the Chinese Scholarship Council and the German state of Bavaria. We would like to thank the participants of our user study, and Anreas Butz, Friedemann Becker and Dominikus Baur for their valuable feedback.

9. REFERENCES

[1] M. D. Byrne: "Using Icons to Find Documents: Simplicity is Critical," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 446–453, 1993.

[2] S. J. Cunningham, D. Bainbridge and A. Falconer, " 'More of an art than a science': Supporting the Creation of Playlists and Mixes, " *Proceedings of the International Symposium on Music Information Retrieval*, 2006.

[3] J. Graham and J. J. Hull: "iCandy: A Tangible User Interface for iTunes," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Ext. Abstracts,* pp. 2343-2348, 2008.

[4] S. Havre, B. Hetzler and L. Nowell: "ThemeRiver: Visualizing Theme Changes over Time," Proceedings *of the IEEE symposium on Information Visualization*, pp. 115-123, 2000.

[5] P. Kolhoff, J. Preuß and J. Loviscach: "Music Icons: Procedural Glyphs for Audio Files," *Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing*, pp. 289-296, 2006.

[6] E. Pampalk: "Islands of Music: Analysis, Organization, and Visualization of Music Archives," *Journal of the Austrian Society for Artificial Intelligence*, 2003.

[7] E. Pampalk and M. Goto: "MusicRainbow: A New User Interface to Discover Artists Using Audio-based Similarity and Web-based Labeling," *Proceedings of the International Symposium on Music Information Retrieval*, 2006.

[8] V. Setlur, C. Albrecht-Buehler, A. A. Gooch, S. Rossoff and B. Gooch: "Semanticons: Visual Metaphores as File Icons," *Computer Graphics Forum*, Vol. 24, pp. 647-656, 2005.

[9] I. Stavness, J. Gluck, L. Vilhan and S. Fels: "The MUSICtable: A Map-based Ubiquitous System for Social Interaction with a Digital Music," *Proceedings of the International Conference on Entertainment Computing*, pp. 291-302, 2005.

[10] M. Torrens, P. Hertzog and J. L. Arcos: "Visualizing and Exploring Personal Music Libraries," *Proceedings of the International Symposium on Music Information Retrieval*, 2004.

[11] G. Tzanetakis, G. Essl and P. Cook: "Human Perception and Computer Extraction of Musical Beat Strength," *Proceedings of the International Conference on Digital Audio Effects*, pp. 257-261, 2002.

[12] G. Tzanetakis and P. Cook: "3D Graphics Tools for Sound Collections," *Proceedings of the International Conference on Digital Audio Effects*, 2000.

[13] R. van Gulik and F. Vignoli: "Visual Playlist Generation on the Artist Map," *Proceedings of the International Symposium on Music Information Retrieval*, 2005.

[14] M. Wattenberg: "Arc Diagrams: Visualizing structure in strings," *Proceedings of the IEEE symposium on Information Visualization*, pp. 110-116, 2002.

[1] http://www.istockphoto.com

TIMBRAL QUALITIES OF SEMANTIC STRUCTURES OF MUSIC

Rafael Ferrer and Tuomas Eerola
Finnish Centre of Excellence in Interdisciplinary Music Research
rafael.ferrer-flores@jyu.fi; tuomas.eerola@jyu.fi

ABSTRACT

The rapid expansion of social media in music has provided the field with impressive datasets that offer insights into the semantic structures underlying everyday uses and classification of music. We hypothesize that the organization of these structures are rather directly linked with the "qualia" of the music as sound. To explore the ways in which these structures are connected with the qualities of sounds, a semantic space was extracted from a large collection of musical tags with latent semantic and cluster analysis. The perceptual and musical properties of 19 clusters were investigated by a similarity rating task that used spliced musical excerpts representing each cluster. The resulting perceptual space denoting the clusters correlated high with selected acoustical features extracted from the stimuli. The first dimension related to the high-frequency energy content, the second to the regularity of the spectrum, and the third to the fluctuations within the spectrum. These findings imply that meaningful organization of music may be derived from low-level descriptions of the excerpts. Novel links with the functions of music embedded into the tagging information included within the social media are proposed.

1. INTRODUCTION

Attempts to craft a bridge between acoustic features and the subjective sensation they provoke [3] have usually started with concepts describing instrument sounds, using adjectives or bipolar scales (e.g., bright-dark, static-dynamic) and matching these with acoustic descriptors (such as shape of the envelope and energy distribution) [11, 20].

In this study, we present a purely bottom-up approach to the conceptual mapping between sound qualities and emerging meanings. We utilized social media to obtain a wide sample of music and extract an underlying semantic structure of this sample. Next, we evaluated the validity of the obtained mapping by investigating the acoustic features underlying the semantic structures. This was done by an analyzing of the examples representing the semantic space, and by having participants to rate the similarity of random spliced sound examples representing the semantic space.

Social tagging is an activity, where descriptive verbal characterizations are given to items of interest, such as songs, images, or links as a part of the normal use of the popular online services. Tags can be considered as semantic representations of abstract concepts created essentially for mnemonic purposes and used typically to organize items [14]. Tagging music is not a novel idea, as any labeling scheme such as musical genres may be considered as tags themselves, but in recent years in the context of social networks, tagging has acquired a new relevance and meaning [1].

Despite all the possibilities offered by large databases containing tags, a central problem remains on how to derive an ontology from them [19]. Starting with the assumption of an underlying structure existing in an apparently unstructured set, we consider a sample of tags to extract a semantic structure, explained next.

2. ANALYSIS OF TAGS

2.1 Material

A collection of 6372 songs [7] representing 15 musical genres (Alternative, Folk, Finnish Iskelmä, Pop, World, Blues, Gospel, Jazz, Rock, Classical, Heavy, Soul, Electronic, Hip-Hop, Soundtrack) served as the initial database of music. Musical genres were used in establishing the sample in order to maximize musical variety in the collection and to be compatible with a host of music preference studies (e.g., [6, 22]) that have provided lists of 13 to 15 broad musical genres relevant for most Western adult listeners. The tags related to the songs in this collection were retrieved from an online music service (*last.fm* [1]) with a dedicated API (Application programming interface) named *Pylast* [2].

2.2 Description of the corpus

The retrieved *corpus* consists of 5,825 lists of tags (mean length of 62.27 tags), each list (*document* in this context) is associated with a piece of music. The number of times each tag had been used in the system until the time of the retrieval was also obtained, representing a measure of "popularity".

[1] http://www.last.fm
[2] http://code.google.com/p/pylast/

In total, the corpus contains 362,732 tags, from which 77,537 are distinct. Each tag is formed by one or more words (M=2.48, SD=1.86), a small proportion of the distinct tags in the corpus contain long expressions (e.g. 6% of the distinct tags are formed by 5 words or more). In this study a tag is considered as a unit representing an element of the *vocabulary*, disregarding the number of words that compose it. Treating tags as *collocations* (i.e. frequent juxtaposition of words) shifts the focus from data processing to concept processing [2], also allowing the tags to function as conceptual expressions [23] instead of words or phrases.

2.3 Lexical layers of the vocabulary

Preprocessing is necessary in any text mining application because retrieved data does not follow any particular set of rules, and there are not standard steps to follow [13].

Three filtering rules where applied to the corpus in the quantitative domain. First, *hapax legomena* (i.e. tags used only once in the corpus), are removed under the rationale of discarding unrelated data. To capture the most prevalent and relevant tags, a second filter uses the associated popularity measure of each tag to eliminate the tags below the mean popularity index of the vocabulary. The third step eliminates tags with three or more words to prune short sentence-like descriptions from the corpus. The subset resulting from such reductions represents 46.6% of the corpus (N=169,052, Vocabulary=2,029 tags).

At this point, data has been de-noised but for the extraction of a meaningful semantic ontology from the tags, a semantic analysis and qualitative filtering is necessary. To categorize the tags at a functional level [24] (e.g. musicological and lexicological), an analysis was performed by using the Brown Corpus [9] as parts-of-speech (POS) tagger, Wordnet database [8] for word sense disambiguation, and Urban Dictionary online [3] and Last.fm database for general reference. Tags are looked-up in these sources and the selection of a category is decided by reviewing each case. The criteria applied in this process favors categories closely related to music, such as genre, artist, instrument, form and company, then adjectives, and finally other types. For instance, "Acid" is a noun but it is also a term extensively used to describe certain musical genres, so it was classified according to its musical function. Proposed categories, percentage of the vocabulary, definition and examples are shown in Table 1. The resulting layers were used to make a finer discrimination of the tags to uncover the semantic structure. Since one of the main motivations of this project was to obtain prototypical timbral descriptions, we focused on tags related to adjectives, nouns, instruments, temporal and verbs.

2.4 Semantic structure

Tag structure (or folksonomy) is obtained by using *latent semantic analysis* (LSA) as a framework [5], a method that has been used before in the domain of musical tags [17, 18]. In this study, detection of semantic structure has three stages: 1) construction of a *Term-Document Matrix*, 2) calculation of similarity coefficients, and 3) cluster analysis. First, a Term-Document Matrix $\mathbf{X} = \{x_{ij}\}$ is constructed. Where each song i, corresponds to a "Document" and each unique tag (or item of the vocabulary) j, to a "Term". The result is a binary matrix $\mathbf{X}(0,1)$ containing information about the presence or absence of a particular tag to describe a given song. Second, a similarity matrix $n \times n$ $\mathbf{D}$ with elements d_{ij} where $d_{ii} = 0$ for all i, is created by computing similarity indexes between tag vectors x_{i*j} of $\mathbf{X}$ with:

$$d_{ij} = \frac{ad}{\sqrt{(a+b)(a+c)(d+b)(d+c)}} \quad (1)$$

where *a* is the number of (1,1) matches, *b* for (1,0), *c* for (0,1) and *d* for (0,0).

There are several methods to compute similarity coefficients between binary vectors (c.f., [10]). This coefficient was selected because of its *symmetric* quality, which considers the double absence (0,0) as important as (1,1), that presumably has positive impact on ecologic applications [10]. A hierarchical clustering algorithm was used to transform the similarity matrix into a sequence of nested partitions. The method used in the hierarchical clustering was Ward's minimum variance, to find compact, spherical clusters [21] and because it has demonstrated its proficiency in comparison to other methods [12].

After obtaining a hierarchical structure, the clusters are derived from the resulting dendrogram by "pruning" the branches with an algorithm that uses a partitioning around medioids (PAM) clustering method in combination with the height of the branches [15]. Figure 1 shows a two dimensional projection (obtained with multidimensional scaling) of the similarity matrix used in the hierarchical clustering. Each dot represents a tag, and the numbers show the centers of their corresponding clusters. Each number is enclosed in a circle that shows the relative size of the cluster in terms of the number of tags contained in it. A more detailed reference on the content of the clusters can be consulted in Table 2.

2.5 Ranking of musical examples in the clusters

In order to explore any acoustic or musical aspects of the clusters, we need to link the clusters with the specific songs represented by the tags. For this, a $m \times n$ *Term Document Matrix* (TDM) $\mathbf{X} = \{x_{ij}\}$ is constructed, where lists of tags attributed to a particular song are represented as m, and preselected tags as n. A list of tags is a finite set $\{1, ..., k\}$, where $1 \leq k \leq 96$. Each element of the matrix contains a value of the normalized rank of a tag if found on a list, and it is defined by:

$$x_{ij} = \left(\frac{r_k}{k}\right)^{-1} \quad (2)$$

Where r_k is the cardinal rank of the tag j if found in i, and k is the total length of the list. To obtain a cluster profile,

[3] http://www.urbandictionary.com

Categories	%	Definition	Examples
Genre	36.72%	Musical genre or style	Rock, Alternative, Pop
Adjective	12.17%	General category of adjectives	Beautiful, Mellow, Awesome
Noun	9.41%	General category of nouns	Love, Melancholy, Memories
Artist	8.67%	Artists or group names	Coldplay, Radiohead, Queen
Locale	8.03%	Geographic situation or locality	British, American, Finnish
Personal	6.80%	Words used to manage personal collections	Seen Live, Favourites, My Radio
Instrument	4.83%	Sound source	Female vocalists, Piano, Guitar
Unknown	3.79%	Unclassifiable gibberish	aitch, prda, <3
Temporal	2.41%	Temporal circumstance	80's, 2000, Late Romantic
Form	2.22%	Musical form or compositional technique	Ballad, Cover, Fusion
Company	1.72%	Record label, radio station, etc.	Motown, Guitar Hero, Disney
Verb	1.63%	General category of verbs	Chillout, Relax, Wake up
Content	1.03%	Emphasis in the message or literary content	Political, Great lyrics, Love song
Expression	0.54%	Exclamations	Wow, Yeah, lol

Table 1. Main categories of tags, their prevalence, definition and examples.

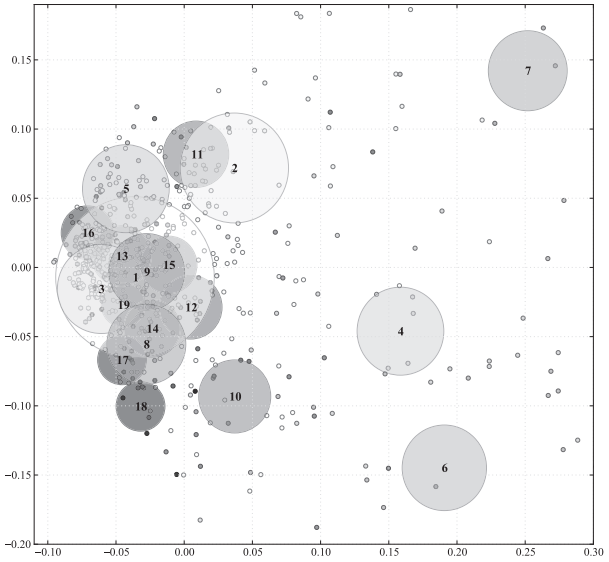


Figure 1. 19 clusters obtained with hierarchical clustering and hybrid pruning.

mean rank of the tag across the TDM is calculated with:

$$\bar{r}_j = \frac{\sum_{i=1}^{m} x_{ij}}{m} \tag{3}$$

Thus the cluster profile or mean ranks vector is defined as:

$$\mathbf{p}_l = \bar{r}_{j \in C_l} \tag{4}$$

C_l denotes a given cluster l for $1 \leq l \leq 19$ (optimal number of clusters for this dataset), and $\mathbf{p}$ is a vector $\{5, ..., k\}$, where $5 \leq k \leq 334$.

Last step aims to obtain ranked lists of songs ordered in terms of its closeness to each cluster profile. This is carried out by calculating the euclidean distance between each song rank vector $x_{i,j \in C_l}$ and the cluster profile $\mathbf{p}_l$:

$$d_i = \sqrt{\sum_{j \in C_l} (x_{ij} - \mathbf{p}_l)^2} \tag{5}$$

The examples of the results can be seen in Table 2, where top artists of each cluster are displayed below central tags of the cluster.

3. EXPERIMENT

In order to explore whether the obtained clusters are perceptually meaningful and to further understand what kinds of acoustic and musical attributes they consist of, empirical data unrelated to the existing structures about the clusters is needed. A similarity rating experiment was designed to assess the timbral qualities of songs pertaining to each of the clusters. We chose to emphasize the low-level, non-structural qualities of music since we wanted to minimize the confounding factors caused by recognition of songs, artists and the subsequent associations with these as well as the lyrical contents of the music. To this end, the stimuli for the experiment consisted of semi-randomly spliced, brief excerpts, explained in detail below.

3.1 Experiment details

3.1.1 Stimuli

Initially, 5-second audio samples were taken from a random middle part (25% after the beginning and 25% before the end) of the 25 top ranked songs (see ranking procedure in section 2.5) from each cluster. For each sample, the temporal position of notes onsets were estimated based on *spectral flux* using MIRToolbox [16]. The highest onset was selected as a reference point from which slices of random length ($150ms \leq t \leq 250ms$) were taken from $10ms$ before the peak onset of each sample, then equalized in loudness, and finally mixed together using a fade in-out of $50ms$ with an overlap window of $100ms$ This resulted in 19 excerpts (each representing a cluster) of variable length, that were finally trimmed to $1750ms$, with a fade in-out of $100ms$ To prepare these 19 excerpts for a similarity rating, the 171 paired combinations were mixed with a silence of $600ms$. between them.

3.1.2 Participants

12 females and 9 males (age M=26.8, SD=4.15) participated to the experiment. 9 of them possessed least one year of musical training. 12 reported listening to music attentively between one and 10 hours per week.

Cluster ID	*Tags proximate to cluster centroids*	Top artists in the cluster
1	*Energetic, Female vocal, Powerful, Hot, Sex*	Amy Adams, Fred Astaire, Kelly Clarkson
2	*Dreamy, Chill out, Haunting, Sleep, Moody*	Nick Drake, Radiohead, Massive Attack
3	*Sardonic, Sarcastic, Cynical, Humorous, Funny*	Alabama 3, Yann Tiersen, Tom Waits
4	*Awesome, Amazing, Male vocalist, Loved, Great*	Guns N' Roses, U2, Metallica
5	*Composer, Cello, Piano, Cello rock, Violin*	Camille Saint-Saëns, Tarja Turunen, Franz Schubert
6	*Female vocalist, Female vocalists, Female, 00s, Sexy*	Fergie, Lily Allen, Amy Winehouse
7	*Mellow, Beautiful, Chillout, Chill, Sad*	Katie Melua, Phil Collins, Coldplay
8	*Hard, Angry, Loud, Aggressive, Rock out*	System of a Down, Black Sabbath, Metallica
9	*60s, 70s, Guitar virtuoso, Sixties, Guitar solo*	Simon & Garfunkel, Janis Joplin, The Four Tops
10	*Feelgood, Summer, Feel good, Cheerful, Gute laune*	Mika, Goo Goo Dolls, Shekinah Glory Ministry
11	*Autumnal, Wistful, Intimate, Sophisticated, Reflective*	Soulsavers, Feist, Leonard Cohen
12	*High school, 90's, 1990s, 1995, 1996*	Fool's Garden, The Cardigans, No Doubt
13	*50s, Saxophone, Trumpet, Tenor sax, Sax*	Miles Davis, Thelonious Monk, Charles Mingus
14	*1980s, 80's, Eighties, 80er, Voci maschili*	Ray Parker Jr., Alphaville, Michael Jackson
15	*Affirming, Lyricism, Life song, Vocalization*	Lisa Stansfield, KT Tunstall, Katie Melua
16	*Choral, A capella, Acapella, Choir, A cappella*	Mediæval Bæbes, Alison Krauss, Blackmore's Night
17	*Voce femminile, Femmina, Voci femminili, Femmine*	Avril Lavigne, The Cranberries, Diana Krall
18	*Tangy, Coy, Sleek, Attitude, Flirty*	Kylie Minogue, Ace of Base, Solange
19	*Rousing, Exuberant, Confident, Playful, Passionate*	James Brown, Does It Offend You, Yeah?, Tchaikovsky

Table 2. Most representative tags and typical artists of each of the 19 clusters.

3.1.3 Procedure

Participants were presented with pairs of sound excerpts in random order using a computer interface and high-quality headphones. Their task was to rate the similarity of sounds on a 9-level Likert scale, whose extremes were labeled as *dissimilar* and *similar*. Before the actual experimental trials, they were given instructions and practice trials to familiarize themselves with the task.

3.1.4 Audio features

To explore the acoustic and musical features underlying the perceptual similarities of the clusters, 41 audio features (listed on Table 3) were extracted from each spliced stimuli using MIR toolbox [16]. The choice of features was restricted to those which would be applicable to spliced examples and would not require high-level feature analysis such as structural repetition or tonality. The extraction was carried out using frame-based approach with $50ms$ analysis frame using 50% overlap.

Category	No.	Feature
Dynamics	1-2	RMS energy
	3-4	Attack time (M, SD)
Rhythm	5-6	Fluctuation peak pos. (M, SD)
	7	Fluctuation centroid (M, SD)
Pitch	8-9	Pitch (M, SD)
	10-11	Chromagram (unwr.) centr. (M, SD)
Harmony	12	Entropy (oct. collap. spectr.) (M)
	13	Roughness (M)
	14	Inharmonicity (M, SD)
Timbre	15-16	Brightness (cut-off 110 Hz) (M, SD)
	17-18	Spectral centroid (M, SD)
	19-20	Zerocross (M, SD)
	20-21	Spread (M)
	22	Spectral entropy (M)
	23	Spectral flux (M)
	24	Flatness (M)
	25	Kurtosis (M)
	26-27	Regularity (M, SD)
	28-29	1st MFCC (M, SD)
	⋮	⋮
	30-41	7th MFCC (M, SD)

Table 3. List of extracted audio features (M= mean, SD= standard deviation)

3.2 Results

Highly consistent pattern of similarities between the 21 participants were obtained (Cronbach $\alpha = 0.94$). For this reason, a mean similarity matrix of the individual ratings was subjected to metric multidimensional scaling (MDS) analysis based on stress minimization by means of majorization (SMACOF) [4]. This yielded adequate low - dimensional projections of the data, from which we focus on 2 - dimensional (stress=0.065) and 3 - dimensional (stress=0.027) solutions.

The organization of the clusters (represented with sliced samples) illustrates a clear organization in terms of the semantic qualities of the clusters (see Figure 2), showing the *Awesome* and *Hard* examples on the left uppermost corner, and the semantically distant, *Autumnal* and *Dreamy* in the lower right-hand corner.

To investigate the perceived organization of the semantic clusters in terms of the acoustic qualities, the 3 dimensions were correlated with the extracted audio features. Highly significant correlations, top five shown in Table 4, were observed for dimensions 1 and 2. We may interpret these correlations in terms of the qualities of the sound spectrum: The first dimension is related to the distribution of energy along the frequency (spectral centroid, flatness, brightness, MFCC1, etc.), where the items in the MDS solution are arranged from the high-frequency energy content in the left to the prevalence of low-frequency energy content in the right. The second dimension may be interpreted as the periodic organization of the spectrum, i.e., whether the spectrum is harmonic (roughness,skewness, spread and fluctuation centroid). The clusters represented by the items in the lower part of the MDS solution possess clearer organization of the spectrum in comparison with the the items high on the MDS solution. The third dimension seem to be related the temporal fluctuation of the spectrum (MFCC6 [SD], Fluctuation position [M], MFCC22 [M]).

Dimension 1			Dimension 2			Dimension 3		
Acoustic feature	***r***		**Acoustic feature**	***r***		**Acoustic feature**	***r***	
MFCC 1 (M)	0.94	***	Fluctuation centroid (M)	-0.72	***	MFCC 6 (SD)	0.51	*
Flatness (M)	-0.86	***	Roughness (M)	0.68	**	Fluctuation position (M)	-0.50	*
Centroid (M)	-0.83	***	Skewness (M)	0.67	**	MFCC 2 (M)	-0.46	*
Brightness (M)	-0.81	***	Spread (M)	-0.65	**	Fluctuation peak (M)	0.45	
Spectral entropy (M)	-0.80	***	Kurtosis (M)	0.57	*	Irregularity (SD)	0.44	

$*** = p < .001$, $** = p < .01$, $* = p < .05$

Table 4. Correlations between the dimensions of the multidimensional scaling solution and acoustic descriptors.

Figure 2. Dimensions 1 and 2 of the MDS with behavioural responses and associated tags

3.3 Discussion

In sum, when brief and spliced excerpts taken from the clusters representing semantic structures of the music descriptions are presented to listeners, they are able to form coherent distances between them. An acoustic analysis of the excerpts was used to label the dimensions embedded in the cluster similarities. This analysis showed clear correlations between the dimensional and timbral qualities of music. However, it should be emphasized that the high relevance of many timbral features is only natural since the timbral characteristics of the excerpts were preserved and structural aspects were masked by the semi-random splicing.

We are careful in not taking these early results to mean literally that the semantic structure of the initial sample would be explainable by means of the same timbral features. This is of course another question which is easily empirically approached using feature extraction of the typical examples representing each cluster and either classify the clusters based on features, or predict the coordinates of the clusters within a low dimensional space by means of regression using a larger set of acoustic features (including those that are relevant for full excerpts such as tonality and structure). However, we are positively surprised at the level of coherence from the part of the listener ratings and their explanations in terms of the acoustic features despite the limitations we imposed on the setting (i.e. discarding tags connected with musical genres), splicing and having a large number of clusters to test. Our intention is to follow this analysis with more rigorous selection of acoustic features (PCA and other data reduction techniques) and use multiple regression to assess whether linear combinations of the features would be necessary for explaining the perceptual dimensions.

4. CONCLUSIONS

The present work provided a bottom-up approach to semantic qualities of music descriptions, which capitalized social media, natural language processing, similarity ratings and acoustic analysis. Semantic structures of music descriptions have been extracted from the social media previously [18] but the main difference here was the careful filtering of such data. We used natural language processing to focus on categories of tags that are meaningful but do not afford immediate categorization of music in a way that, for example, musical genre does.

Although considerable effort was spent on finding the optimal way of teasing out reliable and robust structures of the tag occurrences using cluster analysis, several other techniques and parameters within clustering could also have been employed. We realize that other techniques would probably have led to different structures but it is an open empirical question whether the connections between the similarities of the tested items and their acoustic features would have been entirely different. A natural continuation of the current study would be to predict the typical examples of the clusters with the acoustic features by using either classification algorithms or mapping of the cluster locations within a low dimensional space using correlation and multiple regression. However, the issue at stake here was the connection of timbral qualities with semantic structures.

The implications of the present findings are related to several open issues. The first one is the question whether structural aspects of music are required in explaining the semantic structures or whether the low-level, timbral characteristics are sufficient, as was indicated by the present findings. Secondly, what new semantic layers (as indicated by categories of tags) can be meaningfully connected with the acoustic properties of the music? Finally, if the timbral

characteristics are indeed strongly connected with such semantic layers as *adjectives*, *nouns* and *verbs*, do these arise by means of learning and associations, or are the underlying regularities connected with emotional, functional and gestural cues of the sounds?

5. REFERENCES

[1] J.J. Aucouturier and E. Pampalk. Introduction-from genres to tags: A little epistemology of music information retrieval research. *Journal of New Music Research*, 37(2):87–92, 2008.

[2] J. Brank, M. Grobelnik, and D. Mladenic. Automatic evaluation of ontologies. In Anne Kao and Stephen R.Poteet, editors, *Natural Language Processing and Text Mining*. Springer, USA, 2007.

[3] O. Celma and X. Serra. Foafing the music: Bridging the semantic gap in music recommendation. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):250–256, 2008.

[4] J. de Leeuw and P. Mair. Multidimensional scaling using majorization: SMACOF in R. *Journal of Statistical Software*, 31(3):1–30, 2009.

[5] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[6] M.J. Delsing, T.F. ter Bogt, R.C. Engels, and W.H. Meeus. Adolescents music preferences and personality characteristics. *European Journal of Personality*, 22(2):109–130, 2008.

[7] T. Eerola and R. Ferrer. Setting the standards: Normative data on audio-based musical features for musical genres. In *Proceedings of the 7th Triennial Conference of European Society for the Cognitive Sciences of Music, ESCOM*, 2009.

[8] Christiane Fellbaum, editor. *WordNet: An electronic lexical database*. Language, speech, and communication. MIT Press, Cambridge, Mass, 1998.

[9] W.N. Francis and H. Kucera. *Brown corpus. A Standard Corpus of Present-Day Edited American English, for use with Digital Computers*. Department of Linguistics, Brown University, Providence, Rhode Island, USA, 1979.

[10] J.C. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of classification*, 3(1):5–48, 1986.

[11] J.M. Grey. Multidimensional perceptual scaling of musical timbres. *Journal of the Acoustical Society of America*, 61(5):1270–1277, 1977.

[12] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[13] Anne Kao and Stephen R. Poteet, editors. *Natural Language Processing and Text Mining*. Springer Verlag, 2006.

[14] P. Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2):101–114, 2008.

[15] P. Langfelder, B. Zhang, and S. Horvath. *dynamicTreeCut: Methods for detection of clusters in hierarchical clustering dendrograms.*, 2009. R package version 1.20.

[16] O. Lartillot, P. Toiviainen, and T. Eerola. A matlab toolbox for music information retrieval. *Data Aalysis, Machine Learning and Applications*, pages 261–8, 2008.

[17] C. Laurier, M. Sordo, J. Serra, and P. Herrera. Music mood representation from social tags. In *Proceedings of the 10th International Society for Music Information Conference, Kobe, Japan*, 2009.

[18] M. Levy and M. Sandler. Learning latent semantic models for music from social tags. *Journal of New Music Research*, 37(2):137–150, 2008.

[19] H. Lin, J. Davis, and Y. Zhou. An integrated approach to extracting ontological structures from folksonomies. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, page 668. Springer, 2009.

[20] S. McAdams, S. Winsberg, S. Donnadieu, G. De Soete, and J. Krimphoff. Perceptual scaling of synthesized musical timbres: Common dimensions, specificities and latent subject classes. *Psychological Research*, 58(3):177–192, 1995.

[21] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. ISBN 3-900051-07-0.

[22] P.J. Rentfrow and S.D. Gosling. Message in a ballad: the role of music preferences in interpersonal perception. *Psychol Sci*, 17(3):236–242, 2006.

[23] J.M. Siskind. Learning word-to-meaning mappings. *Models of language acquisition: inductive and deductive approaches*, pages 121–153, 2000.

[24] B. Zhang, Q. Xiang, H. Lu, J. Shen, and Y. Wang. Comprehensive query-dependent fusion using regression-on-folksonomies: a case study of multimodal music search. In *Proceedings of the seventeen ACM international conference on Multimedia*, pages 213–222. ACM, 2009.

TOWARDS MORE ROBUST GEOMETRIC CONTENT-BASED MUSIC RETRIEVAL

Kjell Lemström
Department of Computer Science
University of Helsinki

ABSTRACT

This paper studies the problem of transposition and time-scale invariant (*ttsi*) polyphonic music retrieval in symbolically encoded music. In the setting, music is represented by sets of points in plane. We give two new algorithms. Applying a search window of size w and given a query point set, of size m, to be searched for in a database point set, of size n, our algorithm for exact *ttsi* occurrences runs in $O(mwn \log n)$ time; for partial occurrences we have an $O(mnw^2 \log n)$ algorithm. The framework used is flexible allowing development towards even more robust geometric retrieval.

1. INTRODUCTION

Query-by-humming type problems have been under study for over fifteen years. First, the music under investigation was assumed to be monophonic [3], later the term has been used with a wider meaning addressing problems where the task is to search for excerpts of music, resembling a given query pattern, in a large database. Moreover, both the query pattern and the database may be polyphonic, and the query pattern constitutes only a subset of instruments appearing in the database representing possibly a full orchestration of a musical piece. Although current audio-based methods can be applied to rudimentary cases where queries are directed to clearly separable melodies, the general setting requires methods based on symbolic representation that are truly capable of dealing with polyphonic subset matching.

To this end, several authors have recently used geometric-based modeling of music [1, 7–9]. Geometric representations usually also take into account another feature intrinsic to the problem: the matching process ignores extra intervening notes in the database that do not appear in the query pattern. Such extra notes are always present because of the polyphony, various noise sources and musical decorations. There is, however, a notable downside to the current geometric methods: they do not usually allow distortions in tempo (except for individual outliers that are not even discovered) that are inevitable in the application. Even if the query could be given exactly on tempo, the occurrences in the database would be time-scaled versions of the query (requiring *time-scale invariance*). If the query is to be given in a live performance, local jittering will inevitably take place and a stronger invariance, namely the *time-warping invariance* [4], would be a desired property for the matching process.

In this paper, new time-scale invariant geometric algorithms that deal with symbolically encoded, polyphonic music will be introduced. We use the pitch-against-time representation of note-on information, as suggested in [9] (see Fig 1). The musical works in a database are concatenated in a single geometrically represented file, denoted by T; $T = t_0, t_1, \ldots, t_{n-1}$, where $t_j \in \mathbb{R}^2$ for $0 \leq j \leq n-1$. In a typical retrieval case the query pattern P, $P = p_0, p_1, \ldots, p_{m-1}$; $p_i \in \mathbb{R}^2$ for $0 \leq i \leq m-1$, to be searched for is often monophonic and much shorter than the database T to be searched. Sometimes a search window w is applied and typically $w < m$, that is $w < m \ll n$. It is assumed that P and T are given in the lexicographic order. If this is not the case, the sets can be sorted in $O(m \log m)$ and $O(n \log n)$ times, respectively.

The problems under consideration are modified versions of two problems originally represented in [8]. The following list gives both the original problems and the modifications under consideration; for the partial matches in P2 and S2, one may either use a threshold α to limit the minimum size of an accepted match, or to search for maximally sized matches only.

(P1) Find translations of P such that each point in P matches with a point in T.

(P2) Find translations of P that give a partial match of the points in P with the points in T.

(S1) Find time-scaled translations of P such that each point in P matches with a point in T.

(S2) Find time-scaled translations of P that give a partial match of the points in P with the points in T.

Fig. 2 gives 4 query patterns to be searched for in the excerpt of Fig. 1, exemplifying these 4 problems.

In [6], Romming and Selfridge-Field gave a geometric hashing-based algorithm for S2. Without windowing, it works in $O(n^3)$ space and $O(n^2m^3)$ time. This paper studies another way to solve problems S1 and S2. The new algorithms to be introduced resemble Ukkonen et al's PI and PII algorithms. The algorithm for S1 runs in time $O(mn^2 \log n)$ and $O(mn^2)$ space; the algorithm for S2 in $O(m^2n^2logn)$ time and $O(m^2n^2)$ space. An advantage of our method over the one by Romming and Selfridge-Field is that the performance can be spedup by applying an index-filter preprocessing [5]. Our method also seems to be adaptable to time-warping invariant cases. Thus, the method is an important step towards more robust geometric content-based music retrieval.

2. RELATED WORK

Let α denote a similarity threshold for P2, and let $p_0, p_1, \ldots, p_{m-1}$ and $t_0, t_1, \ldots, t_{n-1}$ be the pattern and database points, respectively, *lexicographically sorted* according to their co-ordinate values: $p_i < p_{i+1}$ iff $p_i.x < p_{i+1}.x$ or ($p_i.x = p_{i+1}.x$ and $p_i.y < p_{i+1}.y$), and $t_j < t_{j+1}$ iff $t_j.x < t_{j+1}.x$ or ($t_j.x = t_{j+1}.x$ and $t_j.y < t_{j+1}.y$). In our application the elapsing time runs along the horisontal axes, represented by x, the perceived height, the *pitch*, is represented by y. A translation of P by vector f is denoted $P + f$: $P + f = p_0 + f, \ldots, p_{m-1} + f$. Using this notation, problem P1 is expressible as the search for a subset I of T and some f such that $P + f = I$. Note that decomposing translation f into horisontal and vertical components $f.x$ and $f.y$, respectively, captures two musically distinct phenomena: $f.x$ corresponds to aligning the pattern time-wise, $f.y$ to *transposing* the musical excerpt to a lower or higher key. Note also that a musical time-scaling σ, $\sigma \in \mathbb{R}^+$, has an effect only on the horisontal translation, the vertical translation stays intact.

Example 2.1. *Let* $p = \langle 1, 1\rangle$, $f = \langle 2, 2\rangle$ *and* $\sigma = 3$. *Then* $p + \sigma f = \langle 7, 3\rangle$.

A straight-forward algorithm solves P1 and P2 in $O(mn \log(mn))$ time. The algorithm first exhaustively collects all the translations mapping a point in P to another point in T. The set of the collected translation vectors are then sorted in lexicographic order. In the case of P1, a translation f that has been used m times corresponds to an occurrence; in the case of P2, any translation f that has been used at least α times would account for an occurrence. Thoughtful implementations of the involved scanning (sorting) of the translation vectors, will yield an $O(mn)$ ($O(mn \log m)$) time algorithm for P1 (P2) [8].

Indeed, the above $O(mn \log m)$ time algorithm is the fastest online algorithm known for P2. Moreover, any significant improvement in the asymptotic running time, exceeding the removal of the logarithmic factor, cannot be seen to exist, for P2 is known to be a `3SUM`-hard problem [2]. It is still possible that P2 is also a `Sorting X+Y` -hard problem, in which case Ukkonen et al's PII algorithm would already be an optimal solution. In [2], Clifford et al introduced an $O(n \log n)$ time approximation algorithm for P2.

To make the queries more efficient, several indexing schemes have been suggested. The first indexing method using geometric music representation was suggested by Clausen et al. [1]. Their sublinear query times were achieved by using inverted files, adopted from textual information retrieval. The performance was achieved with a lossy feature extraction process, which makes the approach non-applicable to problems P1 and P2. Typke [7] proposed the use of metric indexes that works under robust geometric similarity measures. However, it is difficult to adopt his method to support translations and partial matching at the same time. Lemström et al's approach [5] combines sparse indexing and (practically) lossless filtering. Their index is used to speed up a filtering phase that charts all the promising areas in the database where real occurrences could reside. Once a query has been received, the filtering phase works in time $O(g_f(m) \log n + n)$ where function $g_f(m)$ is specific to the applied filter f. The last phase involves checking the found c_f ($c_f \leq n$) candidate positions using Ukkonen et al's PI or PII algorithm executable in worst-case time $O(c_f m)$ or $O(c_f m \log m)$, respectively.

The only non-brute-force solution known for S1 and S2 is by Romming and Selfridge-Field [6]. It is based on geometric hashing and works in $O(n^3)$ space and $O(n^2m^3)$ time. By applying a window on the database such that w is the maximum number of events that occur in any window, the above complexities can be restated as $O(w^2n)$ and $O(wnm^3)$, respectively.

3. MATCHING ALGORITHMS

Our matching algorithms for the time-scale invariant problems S1 and S2 resemble somewhat Ukkonen et al's PI and PII algorithms in that they all use a priority queue as a focal structure. Ukkonen et al's PI and PII work on trans-set translations, or *trans-set vectors*, $f = t - p$, where p and t are points in a given query pattern, of length m, and in the underlying database, of length n, respectively. Let us assume (without loss of generality) that all the points, both in the pattern and in the database, are unique. The number of trans-set vectors is within the range $[n+m-1, nm]$. In order to be able to build an index on the database in an offline phase, Lemström et al's method [5] is based on *intra-set vectors*. For instance, translation vector f is an *intra-pattern vector*, if there are two points p and p' ($p, p' \in P$) such that $p + f = p'$. *Intra-database vectors* are defined accordingly. Naturally, the number of intra-pattern and intra-database vectors are $O(m^2)$ and $O(n^2)$, respectively.

The set of *positive intra-pattern vectors* include translations $p_{i'} - p_i$ where in the case of S1: $0 \leq i < m$

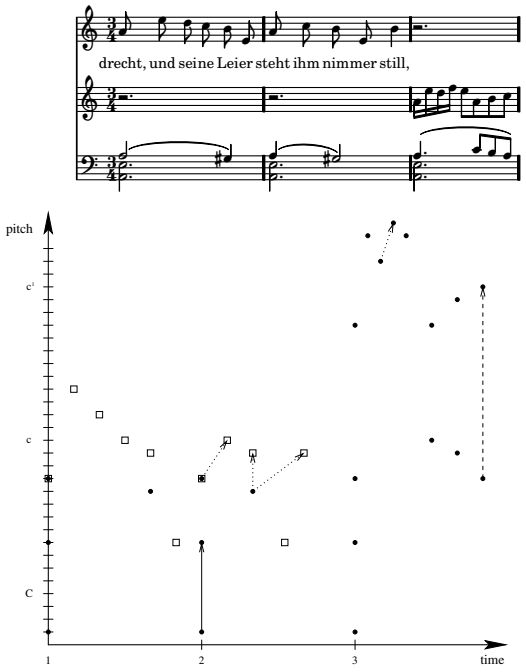


Figure 1. On top, an excerpt from Franz Schubert's song cycle Winterreise. Below, the related geometric, point-set representation. The points associated with the vocal part are represented distinctly (by squares). The depicted 6 intra-database vectors will be discussed later.

and $i' = i + 1$, and in the case of S2: $0 \leq i < i' \leq m$. The set of *positive intra-database vectors* include translations $t_{k'} - t_k$ where, independently of the case, $0 \leq k < k' \leq n$. To reduce the search space, one may apply a window that restates the bounds for i' (in the case of S2) and k' in the obvious way: $0 \leq i < i' \leq \min\{i + w, m\}$ and $0 \leq k < k' \leq \min\{k + w, n\}$.

For the convenience of the algorithms, we pretend that there is an extra elements p_m in the pattern and t_n in the database. The matching algorithms take as input intra-set vectors, stored in tables $K[i], 0 \leq i < m$. Table $K[i]$ stores intra-database translations that may match [1] the positive intra-pattern vectors $p_{i'} - p_i$, i.e., translation vectors starting at point p_i. See Fig. 3 for an illustration on tables $K[i]$.

The entries in our main data structures will be sorted in a lexicographic order. We will specify the underlying order by an ordered set $\aleph$. $\aleph$ is formed by members of $\{a, b, s\}$, where a, b and s correspond to the columns named accordingly in tables $K[i]$. For instance, lexicographic order $\langle a, s\rangle$ is firstly based on the values on column a (the starting point of the associated intra-database vector), secondly on the values on column s (the associated scaling value). A main loop that goes exhaustively through all the possibilities of positive intra-pattern and positive intra-database vectors to initialise the tables $K[i]$ is needed. To this end, let a positive intra-database vector $g = t_{k'} - t_k$ be such that there is a positive intra-pattern vector $f = p_{i'} - p_i$

[1] Please note the distinction between an occurrence and a match. An *occurrence* involves as many *matching* pairs of intra-database and intra-pattern vectors as is required.

$\langle 0, 7\rangle$	$\langle 0, 12\rangle$	$\langle 2, 15\rangle$	$\langle 0, 5\rangle$	$\langle 2, 8\rangle$	$\langle 4, 4\rangle$
$\langle \mathbf{2}, \mathbf{3}\rangle$	$\langle 4, -1\rangle$	$\langle 4, 2\rangle$	$\langle 2, -4\rangle$	$\langle 2, -1\rangle$	$\langle 4, -8\rangle$
$\langle \mathbf{0}, \mathbf{3}\rangle$	$\langle 2, -4\rangle$	$\langle \mathbf{4}, \mathbf{3}\rangle$	$\langle 2, -7\rangle$	$\langle 4, 0\rangle$	$\langle 6, -14\rangle$
$\langle 2, 7\rangle$	$\langle 4, -7\rangle$	$\langle 4, 0\rangle$	$\langle 2, -14\rangle$	$\langle 2, -7\rangle$	$\langle 2, -2\rangle$
$\langle 0, 7\rangle$	$\langle 0, 12\rangle$	$\langle 0, 24\rangle$	$\langle 0, 5\rangle$	$\langle 0, 17\rangle$	$\langle 1, 24\rangle$
$\langle 0, 12\rangle$	$\langle 1, 19\rangle$	$\langle 2, 17\rangle$	$\langle 1, 7\rangle$	$\langle 2, 5\rangle$	$\langle 3, 8\rangle$
$\langle 1, -2\rangle$	$\langle 2, 1\rangle$	$\langle 3, 0\rangle$	$\langle \mathbf{1}, \mathbf{3}\rangle$	$\langle 2, 2\rangle$	$\langle 4, -14\rangle$
$\langle 1, -1\rangle$	$\langle 3, -17\rangle$	$\langle 3, -8\rangle$	$\langle 2, -16\rangle$	$\langle 2, -7\rangle$	$\langle 4, -18\rangle$
$\langle 0, 9\rangle$	$\langle 2, -1\rangle$	$\langle 2, 11\rangle$	$\langle 2, -10\rangle$	$\langle 2, 2\rangle$	$\langle 4, -12\rangle$
$\langle 0, 12\rangle$	$\langle 2, -2\rangle$	$\langle 2, 13\rangle$	$\langle 2, -14\rangle$	$\langle 0, 1\rangle$	$\langle 0, 15\rangle$

Table 1. The intra-database vectors generated by the example given in Fig. 1 when ignoring the first bar and setting $w = 3$. The first and the last intra-database vectors under consideration are depicted in Fig. 1 as arrows with solid and dashed stems, respectively.

for which $g.y = f.y$ (ie. the pitch intervals of the two vectors match). Because g may be part of an occurrence, a new row, let it be the hth, in $K[i]$ is allocated and the following updates are conducted:

$$K[i]_h.a \leftarrow k; \quad K[i]_h.b \leftarrow k'; \tag{1}$$

$$K[i]_h.s \leftarrow \frac{t_{k'}.x - t_k.x}{p_{i'}.x - p_i.x}; \tag{2}$$

$$K[i]_h.y \leftarrow \mathbf{nil}; \quad K[i]_h.w \leftarrow 1; \tag{3}$$

$$K[i]_h.c \leftarrow i'; \quad K[i]_h.z \leftarrow 0. \tag{4}$$

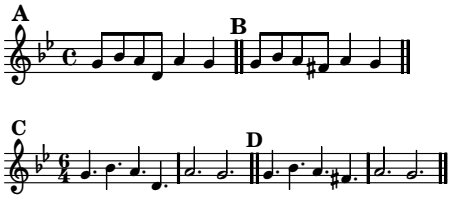
Above, in (1), the associated starting and ending points of the matching intra-database vector are stored in $K[i]_h.a$ and $K[i]_h.b$, respectively. The required time scaling for the intra-vectors to match is stored in $K[i]_h.s$ (2); here extra carefulness is needed to avoid zero division: If both the numerator and the denominator equal zero, we set $K[i]_h.s = 1$. If only one of them equals zero or both equal infinity, the whole row is deleted from the table (they would represent impossible time scalings). Columns y and w, initialised in (3), are used for backtracking a found occurrence and storing the length of a candidate occurrence, respectively. The last columns, initialised in (4), will be needed when searching for partial occurrences (in Section 3.2): column c stores the ending point of the associated intra-pattern vector, z is used for identifying an occurrence.

Denoting by Σ_{p_i} the number of rows generated above for table $K[i]$, $0 \leq i < m$, for the aforementioned extra elements we set:

$$K[i]_{\Sigma_{p_i}}.a \leftarrow K[i]_{\Sigma_{p_i}}.b \leftarrow \infty;$$
$$K[i]_{\Sigma_{p_i}}.s \leftarrow K[i]_{\Sigma_{p_i}}.w \leftarrow 0;\ K[i]_{\Sigma_{p_i}}.c \leftarrow i + 1$$

As each iteration of the main loop takes constant time, this exhaustive initialisation process runs in time $O(nmw^2)$. Finally, the columns in $K[i]$ are sorted in lexicographic order $\langle a, s\rangle$. The matching algorithms have an associated priority queue Q_i for each table $K[i]$, $0 < i \leq m$ [2]. For Q_i, a lexicographic order $\langle b, s\rangle$ is used. As a reminder, the order is given in the superscript of a priority queue (e.g. $Q_i^{\langle b,s\rangle}$).

[2] A single priority queue would suffice, but the algorithm would become more complicated.



$\langle \mathbf{6}, \mathbf{3} \rangle \quad \langle 6, -2 \rangle \quad \langle 6, -7 \rangle \quad \langle 6, 7 \rangle \quad \langle 12, -2 \rangle$

Figure 2. On top, 4 example queries. For query **A** an occurrence in the excerpt given in Fig. 1 would be found in all four cases P1, P2, S1 and S2; for **B** in cases P2, S1 and S2; for **C** in S1 and S2; and for **D** in S2 only. At the bottom, the positive intra-pattern vectors, associated with the query **C** in case S1.

3.1 S1: Time Scaled Exact Matching

Once the tables $K[i]$ have been initialised and their columns have been sorted in lexicographic order $\langle a, s \rangle$, the transposition-invariant time-scaled exact occurrences can be found using the matching algorithm given in Fig. 4. The algorithm works by observing *piecewise matches* between positive intra-database and intra-pattern vectors

$$t_{k_{i'}} - t_{k_i} = \sigma_i(p_{i+1} - p_i) \tag{5}$$

that are stored in the associated $K[i]$. Above $\sigma_i \in \mathbb{R}^+$ is the time-scaling factor (recall Example 2.1). The piecewise matches may form a chain $T_{\tau_0 \ldots \tau_{m-1}} = t_{\tau_0}, t_{\tau_1}, \ldots, t_{\tau_{m-1}}$, where $\tau_0, \tau_1, \ldots, \tau_{m-1}$ is an increasing sequence of indices in T; $t_{\tau_{i+1}} - t_{\tau_i} = \sigma(p_{i+1} - p_i)$ for $0 \leq i < m-1$ and $\sigma \in \mathbb{R}^+$ is a time-scaling factor common to all the piecewise matches in the chain. As such chains represent transposition-invariant, time-scaled exact occurrences, the task is to look for them.

A chain $T_{\tau_0 \ldots \tau_{m'}}$ $m' < m-1$, is called a *prefix occurrence* (of length m'); $T_{\tau_{m'-1}, \tau_{m'}}$ is the *final suffix* of the prefix occurrence $T_{\tau_0 \ldots \tau_{m'}}$. Let $t_{\tau_{i+1}} - t_{\tau_i}$ (that, by definition, equals $\sigma(p_{i+1} - p_i)$) be the final suffix of a prefix occurrence $T_{\tau_1 \ldots \tau_{m'}}$. The prefix occurrence is *extensible* if there is a piecewise match $t_{k'_{i+1}} - t_{k_{i+1}} = \sigma(p_{i+2} - p_{i+1})$ such that

$$t_{\tau_{i+1}} = t_{k_{i+1}} \tag{6}$$

and scaling factor σ is the one that was used in forming $T_{\tau_1 \ldots \tau_{m'}}$. The binding in Equation 6 is called the *binding of extension*, $t_{\tau_{i+1}} - t_{\tau_i}$ the *antecedent* and $t_{k'_{i+1}} - t_{k_{i+1}}$ the *postcedent of the binding.*

Lemma 3.1. *If a prefix occurrence is extensible, its final suffix is also extensible.*

Proof. Immediate. □

To be more efficient, at point $i+1$, the algorithm actually considers any piecewise match $t_{k'_i} - t_{k_i} = \sigma_i(p_{i+1} - p_i)$ as an antecedent to the binding and tries to extend it. Because in this case the piecewise matches in an occurrence chain have to be consecutive in P, the antecedents of the binding are all found in $K[i]$ and their possible extensions, postcedents, in $K[i+1]$. To process all the bindings of extension at point $i+1$, therefore, involves going through all the entries both in $K[i]$ and in $K[i+1]$. To make this process efficient, no entry of either of the tables should be observed more than once for one iteration. In order for this to be possible, both sides of the binding of extension (associated with antecedents and postcedents) should be enumerated in the same (increasing) order. However, the lefthand side of the binding involves end points of the intra-database vectors in $K[i]$ and the righthand side the start points of the intra-database vectors in $K[i+1]$. Therefore, we use a priority queue $Q_i^{\langle b,s \rangle}$ whose entries are addresses to rows associated with the antecedents of the binding at i. In this way, the binding of extension at i can be done efficiently by enumerating the entries in Q_i and $K[i]$. Note that the set of piecewise matches extended this way also includes all the final suffixes, and therefore, according to Lemma 3.1, also all the prefix occurrences.

The binding of extension takes place in line (8) of the algorithm. If a piecewise match is extensible, its length is updated (line 9) and a backtracking link is stored (line 10). The latter becomes useful if any of the extended piecewise matches extends into a proper occurrence, and the whole occurrence is to be revealed (instead of just reporting it).

Let us now demonstrate the main idea of the algorithm by using a musical example.

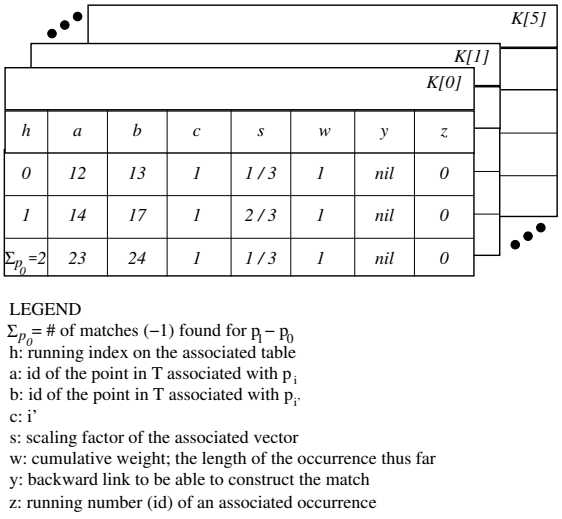


Figure 3. Illustration of tables $K[i]$ when considering problem (S1) and searching for occurrences for the query **C** of Fig. 2 within the two last bars of Fig. 1, $w = 3$. The intra-database vectors under consideration are the ones given in Table 1. Having initialized the tables K in equations (1-4), $K[0]$ contains the depicted 3 rows.

Example 3.2. *The vocal line in Fig. 1 ends in a suspension that is dissolved at the beginning of the third*

TimeScaledExactOccurrence($K[i]$)
(0) $K[0]_{\Sigma_{p_0}}.s \leftarrow \infty$
(1) **for** $j \leftarrow 0, \ldots, \Sigma_{p_0}$ **do**
(2) $\quad Q_1^{\langle b,s \rangle} \leftarrow$ **push**$(\&K[0]_j)$
(3) **for** $i \leftarrow 1, \ldots, m-2$ **do**
(4) $\quad q \leftarrow$ **pop**$(Q_i^{\langle b,s \rangle})$
(5) $\quad$ **for** $j \leftarrow 0, \ldots, \Sigma_{p_i} - 1$ **do**
(6) $\quad\quad$ **while** $[q.b, q.s] < [K[i]_j.a, K[i]_j.s]$ **do**
(7) $\quad\quad\quad q \leftarrow$ **pop**$(Q_i^{\langle b,s \rangle})$
(8) $\quad\quad$ **if** $[q.b, q.s] = [K[i]_j.a, K[i]_j.s]$ **then**
(9) $\quad\quad\quad K[i]_j.w \leftarrow q.w + 1$
(10) $\quad\quad\quad K[i]_j.y \leftarrow q$
(11) $\quad\quad\quad Q_{i+1}^{\langle b,s \rangle} \leftarrow$ **push**$(\&K[i]_j)$
(12) $\quad\quad\quad q \leftarrow$ **pop**$(Q_i^{\langle b,s \rangle})$
(13) $\quad K[i]_{\Sigma_{p_i}}.s \leftarrow \infty$
(14) $\quad Q_{i+1}^{\langle b,s \rangle} \leftarrow$ **push**$(\&K[i]_{\Sigma_{p_i}})$
(15) **if** $K[m-2]_j.w = m-1$ **for some** $0 \leq j \leq \Sigma_{p_{m-2}}$
(16) $\quad$ **then** `report an occurrence`

Figure 4. Algorithm for finding transposition-invariant time-scaled exact occurrences.

bar. As the expectation for the dissolution is strong and the lower part of the accompaniment resides in the same register as the vocal part, if suitably arranged, the listener perceives the higher "a" of the lower part of the accompaniment to belong to the vocal melody. The queries in Fig. 2 look for occurrences with such a dissolution. To solve S1 with the query **C** *of Fig. 2, the algorithm first fills Table* $K[0]$ *with rows corresponding to the intra-database vectors that match the interval of the first intra-pattern vector* $\langle \mathbf{6}, \mathbf{3} \rangle$ *(bolded in the figure). The matching vectors are depicted in Fig. 1 (arrows with a dotted stem) and given bolded in Table 1. Note that the vector* $\langle \mathbf{0}, \mathbf{3} \rangle$ *is not accepted since the associated scaling would actually squeeze any melody to a chord. The three accepted piecewise matches are stored in* $K[0]$ *(see Fig. 3) and the algorithm continues by looking for piecewise matches in* $K[1]$ *that could extend them.*

Analysis. Let us denote by $|Q_i|$ and $|K[j]|$ the number of entries in Q_i and $K[j]$, respectively. Clearly, in this case, $|Q_i| = |K[i-1]|$ for $1 \leq i \leq m$. Moreover, let $\Sigma = max_{i=1}^{m}(|Q_i|, |K[i-1]|)$. The outer loop (line (3)) is iterated m times. Within the inner loop (line (5)), all the entries in Q_i and in $K[i]$ are processed exactly once, resulting in $O(\Sigma)$ entry processing steps. The only operation taking more than a constant time is the updating of the priority queue; it takes at most $O(\log \Sigma)$ time. Thus, the algorithm runs in time $O(m\Sigma \log \Sigma)$. Moreover, the tables $K[i]$ and priority queues Q_i require $O(m\Sigma)$ space.

In this case $\Sigma = O(wn)$, because each table $K[i]$ contains the piecewise matches for the positive intra-pattern vector $p_{i+1} - p_i$, and there are $O(wn)$ possibilities to this end. Naturally $w = n$ if no windowing has been applied. □

TimeScaledPartialOccurrence($K[i]$)
(0) $\ell \leftarrow 0$; $K[0]_{\Sigma_{p_0}}.s \leftarrow \infty$
(1) **for** $i \leftarrow 0, \ldots, m-2$
(2) $\quad$ **for** $j \leftarrow 0, \ldots, \Sigma_{p_0}$
(3) $\quad\quad Q_{K[i]_j.c}^{\langle b,s \rangle} \leftarrow$ **push**$(\&K[i]_j)$
(4) **for** $i \leftarrow 1, \ldots, m-2$ **do**
(5) $\quad q \leftarrow$ **pop**$(Q_i^{\langle b,s \rangle})$
(6) $\quad$ **for** $j \leftarrow 0, \ldots, \Sigma_{p_i} - 1$ **do**
(6,5) $\quad\quad$ **if** $[q.b, q.s] > [K[i]_{\Sigma_{p_i}}.a, K[i]_{\Sigma_{p_i}}.s]$ **break**
(7) $\quad\quad$ **while** $[q.b, q.s] < [K[i]_j.a, K[i]_j.s]$ **do**
(8) $\quad\quad\quad q \leftarrow$ **pop**$(Q_i^{\langle b,s \rangle})$
(9) $\quad\quad$ **if** $[q.b, q.s] = [K[i]_j.a, K[i]_j.s]$ **then**
(10) $\quad\quad\quad$ **while min**$(Q_i^{\langle b,s \rangle}) = [q.b, q.s]$ **do**
(11) $\quad\quad\quad\quad r \leftarrow$ **pop**$(Q_i^{\langle b,s \rangle})$
(12) $\quad\quad\quad\quad$ **if** $r.w > q.w$ **then** $q \leftarrow r$
(13) $\quad\quad\quad K[i]_j.w \leftarrow q.w + 1$
(14) $\quad\quad\quad K[i]_j.y \leftarrow q$
(15) $\quad\quad\quad$ **if** $K[i]_j.w = \alpha$ **then**
(16) $\quad\quad\quad\quad \ell \leftarrow \ell + 1$
(17) $\quad\quad\quad\quad K[i]_j.z = \ell$
(18) $\quad\quad\quad\quad \kappa[\ell] \leftarrow \&K[i]_j$
(19) $\quad\quad\quad$ **if** $K[i]_j.w > \alpha$ **then**
(20) $\quad\quad\quad\quad K[i]_j.z = q.z$
(21) $\quad\quad\quad\quad \kappa[q.z] \leftarrow \&K[i]_j$
(22) $\quad\quad\quad Q_{K[i]_j.c}^{\langle b,s \rangle} \leftarrow$ **push**$(\&K[i]_j)$
(23) $\quad K[i]_{\Sigma_{p_i}}.s \leftarrow \infty$
(24) $\quad Q_{i+1}^{\langle b,s \rangle} \leftarrow$ **push**$(\&K[i]_{\Sigma_{p_i}})$
(25) ReportOccurrences(κ)

Figure 5. Algorithm for finding transposition-invariant time-scaled partial occurrences. The optimizer at line (6,5) can be omitted without breaking functionality; it can also be used to optimize the previous algorithm (as line (5,5)).

3.2 S2: Time Scaled Partial Matching

In order to be able to find transposition-invariant time-scaled partial occurrences, we need the two extra columns c and z, that were initialised in Equation 4, for tables $K[i]$. Recall that $K[i]_h.c$ stores the ending point i' for an intra-pattern vector $p_{i'} - p_i$ that is found to match an intra-database vector $t_{k'} - t_k$ with some scaling factor σ_i. Column z is used for storing a running number that is used as an id, for a found partial occurrence. Furthermore, we use an extra table, denoted by κ, for storing all the found occurrences.

The structure of the algorithm (see Fig. 5) is similar to the previous algorithm. Again, at point i, the antecedents in Q_i are to be extended by postcedents found in $K[i]$. However, as we are looking for partial occurrences this time, we cannot rely on piecewise

matches that are consecutive in P but any piecewise match associated with a positive intra-pattern vector

$$t_{k_{i'}} - t_{k_i} = \sigma_i(p_{i'} - p_i) \quad (7)$$

has to be considered. Here $0 \leq k_i < k_{i'} \leq \min\{k_i + w, n\}$; $0 \leq i < i' \leq \min\{i+w, m\}$ and $\sigma_i \in \mathbb{R}^+$. Given a threshold α, a chain $T_{\tau_0 \ldots \tau_{\beta-1}}$, such that $t_{\tau_j} - t_{\tau_{j-1}} = \sigma(p_{\pi_j} - p_{\pi_{j-1}})$, for $0 < j \leq \beta - 1$, $\beta \geq \alpha$, where $\tau_0 \ldots \tau_{\beta-1}$ and $\pi_0 \ldots \pi_{\beta-1}$ are increasing sequences of indices in T and P, respectively, would constitute for a transposition-invariant time-scaled partial occurrence (of length β).

That piecewise matches can now be between any two points in the pattern makes the problem somewhat more challenging. This has the effect that, at point i, pushing a reference to a priority queue (lines (2) and (21) of the algorithm) may involve any future priority queue, from Q_{i+1} to Q_m, not just the successive one as in the previous case; the correct priority queue is the one that is stored in $K[i]_j.c$ (recall that it stores the end point of the intra-pattern vector associated with the piecewise match). Conversely, the antecedents at point i (stored in Q_i) may include references to any past tables within the window size, expanding the size of the priority queue Q_i.

The two remaining differences to the algorithm above, are in lines (11) and (14-20). In line (11), the algorithm chooses to extend the piecewise match that is associated with the longest prefix occurrence. This is a necessary step, once again, because we are no more dealing with piecewise matches that are consecutive in P. In lines (14-20) the algorithm deals with a found occurrence. Lines (14-17) deal with a new occurrence: generate a new running number, ℓ, for it (that is used as its id) and store a link to the found occurrence to the table of occurrences κ. Lines (18-20) deal with extending a previously found occurrence.

Analysis. Let $\Sigma = max_{i=1}^{m}(|Q_i|, |K[i-1]|)$. With an analogous reasoning to that of the previous analysis, we arrive at similar complexities: the algorithm runs in $O(m\Sigma \log \Sigma)$ time and $O(m\Sigma)$ space. Let us now analyse the order of Σ in this case. Still it holds that for a positive intra-pattern vector, $p_{i+1} - p_i$, there are $O(wn)$ possible piecewise matches. However, the table $K[i]$ may contain entries associated with piecewise matches with any positive intra-pattern vector ending at point $i+1$. Thus, $max_{i=1}^{m}(|K[i-1]|) = O(\min\{m, w\}wn)$. As $|Q_i| = |K[i-1]|$ for $0 < i \leq m$ and assuming $w < m$, we conclude that the algorithm has an $O(mnw^2 \log n)$ running time and works in a space $O(mnw^2)$. □

4. CONCLUSIONS

In this paper we suggested novel content-based music retrieval algorithms for polyphonic, geometrically represented music. The algorithms are both transposition and time-scale invariant. Given a query pattern $P = p_0, \ldots, p_{m-1}$ to be searched for in a music database $T = t_0, \ldots, t_{n-1}$ and applying a search window of size w, the algorithms run in $O(m\Sigma \log \Sigma)$ time and $O(m\Sigma)$ space, where $\Sigma = O(wn)$ when searching for exact occurrences under such a setting, and $\Sigma = O(nw^2)$ when searching for partial occurrences. Whether this is an improvement in practice over the existing algorithm by Romming and Selfridge-Field [6], working in space $O(w^2n)$ and time $O(wnm^3)$, is left for future experiments on real data.

However, the framework seems to be very flexible: it is currently under modification to a more complex case, where an uneven time deformation is known just to preserve the order of the notes; there are no known solutions for this time-warping invariant problem [4]. Moreover, it seems that with some modifications to the data structures and ideas presented in [5] it would be possible to adopt the idea of using a three-phase searching process (indexing, filtering and checking) resulting in a smaller search space and a better running time to those presented here.

Acknowledgements This work was supported by the Academy of Finland (grants #108547 and #218156).

5. REFERENCES

[1] M. Clausen, R. Engelbrecht, D. Meyer, and J. Schmitz. Proms: A web-based tool for searching in polyphonic music. In *Proc. ISMIR'00*, Plymouth, MA, October 2000.

[2] R. Clifford, M. Christodoulakis, T. Crawford, D. Meredith, and G. Wiggins. A fast, randomised, maximal subset matching algorithm for document-level music retrieval. In *Proc. ISMIR'06*, pp. 150–155, Victoria, BC, October 2006.

[3] A. Ghias, J. Logan, D. Chamberlin, and B. Smith. Query by humming - musical information retrieval in an audio database. In *Proc. ACM Multimedia*, pages 231–236, San Francisco, CA, 1995.

[4] K. Lemström and G. Wiggins. Formalizing invariances for content-based music retrieval. In *Proc. ISMIR'09*, pp. 591–596, Kobe, October 2009.

[5] K. Lemström, N. Mikkilä, and V. Mäkinen. Filtering methods for content-based retrieval on indexed symbolic music databases. *Journal of Information Retrieval*, 13(1):1–21, 2010.

[6] C. Romming and E. Selfridge-Field. Algorithms for polyphonic music retrieval: The hausdorff metric and geometric hashing. In *Proc. ISMIR'07*, pp. 457–462, Vienna, September 2007.

[7] R. Typke. *Music Retrieval based on Melodic Similarity*. PhD thesis, Utrecht University, Netherlands, 2007.

[8] E. Ukkonen, K. Lemström, and V. Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In *Proc. ISMIR'03*, pp. 193–199, Baltimore, MA, October 2003.

[9] G. Wiggins, K. Lemström, and D. Meredith. SIA(M)ESE: An algorithm for transposition invariant, polyphonic content-based music retrieval. In *Proc. ISMIR'02*, pp. 283–284, Paris, October 2002.

TUNEPAL - DISSEMINATING A MUSIC INFORMATION RETRIEVAL SYSTEM TO THE TRADITIONAL IRISH MUSIC COMMUNITY

Bryan Duggan, Brendan O' Shea

School of Computing
Dublin Institute of Technology
Kevin St
Dublin 8
Ireland
{bryan.duggan, brendan.oshea}@dit.ie

ABSTRACT

In this paper we present two new query-by-playing (QBP) music information retrieval (MIR) systems aimed at musicians playing traditional Irish dance music. Firstly, a browser hosted system - tunepal.org is presented. Secondly, we present Tunepal for iPhone/iPod touch devices - a QBP system that can be used *in situ* in traditional music sessions. Both of these systems use a backend corpus of 13,290 tunes drawn from community sources and "standard" references. These systems have evolved from academic research to become popular tools used by musicians around the world. 16,064 queries have been logged since the systems were launched on 31 July, 2009 and 11 February, 2010 respectively to 18 May 2010. As we log data on every query made, including geocoding queries made on the iPhone, we propose that these tools may be used to follow trends in the playing of traditional music. We also present an analysis of the data we have collected on the usage of these systems.

1. INTRODUCTION

There exist approximately seven thousand unique traditional Irish dance tunes [1]. Musicians playing traditional music have a personal repertoire of up to a thousand tunes. Many of these tunes are known by multiple names, while many more are known simply as "gan anim" (without name). In the past, commercial recordings of traditional music were accompanied by extensive sleeve notes providing biographic information on the tunes in the recording. In the modern age two trends have emerged. Firstly, the use of digital audio formats and digital downloading of music has meant that personal music collections do not contain this biographic data and many musicians are unfamiliar with the history and background to the tunes they are playing. This fact is compounded by the fact that although traditional tunes often have colourful and memorable titles (Table 1), there is nothing to link the title of a tune with its melody [2].

The second trend is the development of extensive; crowd sourced biographic references and discographies for tunes on websites such as thesession.org [3]. Linking the melodies of traditional Irish dance tunes to biographic data about the tune, including names, is the goal of an ongoing project at the DIT School of Computing.

Name
The Bucks Of Oranmore
Come West Along The Road
Repeal Of The Union
The Chicken That Made The Soup
More Power To Your Elbow
If It's Sick You Are Tea You Wants
The Night We Made The Match
Last Night's Fun
My Former Wife
The First Night In America

Table 1: Tune names taken from [4]

In our previous work [5-7], we described a proof of concept music information retrieval (MIR) system adapted to the characteristics of traditional Irish dance music that addressed this very problem. In this paper, we present follow up work in developing this research into robust and reliable tools that are now being used by thousands of musicians around the world. Specifically we present tunepal.org – a browser hosted query-by-playing (QBP) system and Tunepal for the iPhone/iPod touch – a QBP system that can be used *in situ* in traditional music sessions. As these systems log details of every query being made (including geotagging queries made on the iPhone), they represent a unique opportunity to analyse the *zeitgeist* of traditional music. In other words, to identify trends, popular tunes and tune types being played around the world.

Section 2 of this paper presents a brief overview of our previous work in this area. Section 3 presents the architecture of tunepal.org. Section 4 presents Tunepal for iPhone. Section 5 presents a summary of usage data collected from the two systems. Section 6 presents a summary and conclusions.

2. RELATED WORK

Our previous work describes Tunepal for Windows Mobile devices such as smartphones and PDA's [8,9]. This is a symbolic MIR system that allows musicians to search for tunes by name, retrieve the ABC notation [10] for the tune and playback the tune. Figure 1 presents screenshots of Tunepal running on a Windows Mobile smartphone. Our aim with this system was to facilitate musicians to

start tunes that they could recall the name of, but not the melody.

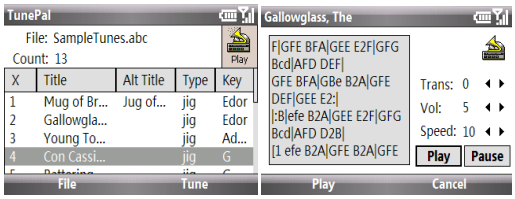


Figure 1: Screenshot of Tunepal for Windows Mobile

MATT2 (Machine Annotation of Traditional Tunes) is a standalone QBP MIR system for traditional Irish dance tunes initially developed for the tin-whistle and wooden flute [7] and subsequently enhanced to support queries on a range of traditional instruments including the uilleann pipes, concertina and fiddle [5,11]. MATT2 is based on two subsystems – a transcription subsystem and a matching subsystem. The transcription subsystem uses an onset detection function based on comb filters (ODCF) developed especially for the transcription of traditional music [12]. A harmonicity, pitch detection algorithm based on Klapuri's [13] multi-pitch estimator is used to extract frequencies from the FFT (Fast Fourier Transform) of a note frame. MATT2 incorporates Ornamentation Filtering (OF) to remove expressiveness from the transcription. The corpus used in MATT2 is Norbeck's reel and jig collection [14], which is pre-processed to expand parts, separate variations, remove ornamentation and normalise for register. This collection contains 1582 reels and jigs, with variations. Matching is achieved using the substring edit distance algorithm [15], with a cost function modified to take account of breath marks in the transcription. An evaluation of this system is presented in [11].

An enhancement to MATT2 is the TANSEY (Turn ANnotation from SEts using SimilarityY profiles) algorithm, named after the traditional flute player Seamus Tansey [6]. TANSEY is a segmentation algorithm to annotate tunes played in set (sequences of multiple tunes repeated multiple times and played segue). TANSEY makes use of *melodic similarity profiles* and can retrieve the start and end of each repetition of a tune, count the repetitions and retrieve the name and associated biographic data associated with each tune in a recording of a set of tunes.

3. TUNEPAL.ORG

Our first task in disseminating the work described in section 2 was to expand the corpus used in the experiments described in [5,11] to include a comprehensive collection of traditional Irish music from definitive sources available in ABC notation. The tunepal.org database contains 13,290 tunes drawn from community sources, such as the website thesession.org [3] and "standard" references including O'Neills Dance Music of Ireland [16] and Brendan Breathneach's Ceol Rince Na hÉireann series in five volumes [17]. Our corpus also includes collections of Welsh, Scottish and Breton music in addition to several different transcriptions of the same tune from the canon of Irish traditional music. Table 2 presents an analysis of sources of the tunes in the tunepal.org corpus.

Source	Count
thesession.org	9,310
Henrik Norbeck	1,474
O'Neills Dance Music of Ireland	994
Ceol Rince na hÉireann 1	73
Ceol Rince na hÉireann 2	192
Ceol Rince na hÉireann 3	37
Ceol Rince na hÉireann 4	220
Jonny O'Leary	196
Nigel Gatherer	794
Total:	13,290

Table 2: Sources of Tunepal tunes

In order to make the system easily accessible to traditional musicians without the necessity of installing software, a browser hosted version of MATT2 – tunepal.org was developed. For this version, the transcription algorithms were deployed in a Java applet, while the tune corpus and matching subsystems were hosted on a server. Figure 2 presents a screenshot of tunepal.org.

To find a tune, a musician records a query played on an instrument such as the concert flute, tin-whistle, uilleann pipes, accordion or concertina. An energy based

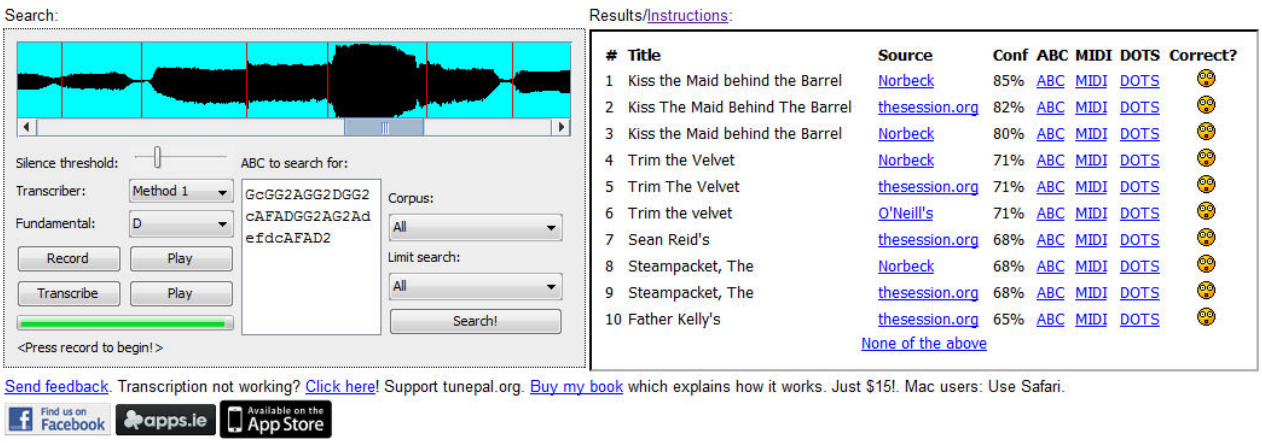


Figure 2: A screenshot of tunepal.org

silence detection algorithm removes silence at the start of recorded queries, which would affect the evaluation of the quaver length (a core element of our system). The user can then click the transcribe button and the system will extract the melody spelled in ABC notation from the recording [10]. tunepal.org differs from similar web based QBP systems such as Musipedia [18] in that traditional instrument queries are explicitly supported. Although Musipedia contains traditional Irish dance tunes as part of its corpus, it does not generate positive results when queries are played on the tin-whistle or wooden flute (as tested by the author).

Users are also offered the ability to change the transcription fundamental. This changes the frequencies used by the pitch spelling algorithm, so that tunepal.org can work with differently pitched instruments, such as Eb flutes and uilleann pipes pitched in B and C.

The query is then submitted to the matching engine, a J2EE web application; hosted on tunepal.org. The matching engine uses the substring edit distance algorithm against the corpus of *search keys* - strings of musical notes extracted from the tunes and normalised as described in [5,11]. These are stored in a MySQL database. For each submitted query, tunepal.org presents the ten closest matches in order of descending distance. MATT2 gives the correct tune as the closest match for 93% of queries in experiments using real-world field recordings of traditional musicians from sessions, classes, concerts and commercial recordings including solo and ensemble playing on traditional instruments recorded in a variety of real-world settings such as noisy public sessions [11]. In tunepal.org therefore we log the closest matching tune for a query in the database. tunepal.org incorporates a feedback system, so users can however proof listen to the results and give feedback as to which (if any) of the returned tunes was the correct one. We also store a confidence score for the match calculated as per (1), where q is the query length and ed is the minimum substring edit distance between the query and the closest match [6].

$$c = 1 - \left(\frac{ed}{q}\right) \quad (1)$$

Each tune in the database can be played, displayed in ABC notation or stave notation. Stave notation display uses ABCJS, an open source, browser hosted rendering engine for ABC notation [19].

tunepal.org was launched on 31 July, 2009. It runs on Windows, Mac and Linux systems. We promoted tunepal.org on popular traditional music discussion forums such as thesession.org and the chiff and fipple forum. tunepal.org has been quite successful and the site is now well known amongst traditional musicians having been profiled in a national newspaper [20]. At the time of writing (18 May 2010), 7,885 queries have been logged. A more detailed analysis of the usage of tunepal.org is presented in section 5.

4. TUNEPAL FOR IPHONE

Traditional Irish music is most commonly played by groups of musicians in a community setting known as a *session* [21]. Sessions usually take place in shared public spaces. It was felt important therefore that for this work to become ubiquitous, it had to be made available on a mobile handheld device. We therefore ported the functionality of tunepal.org to the iPhone platform. Figure 3 presents screenshots of Tunepal running on an iPhone.

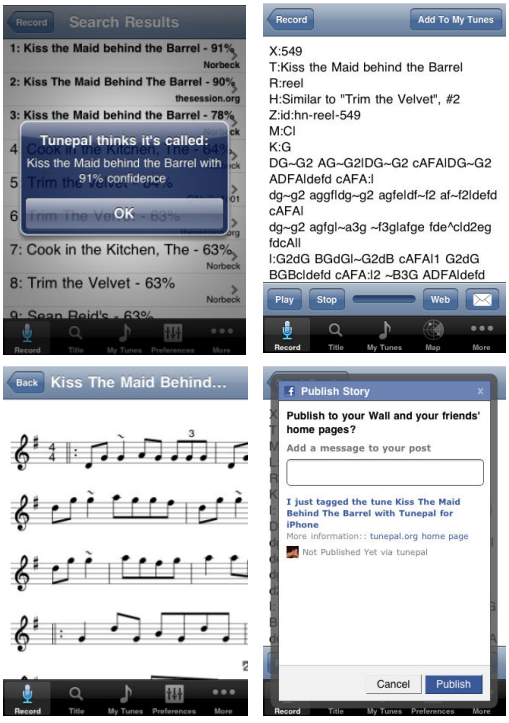


Figure 3: Screenshots of Tunepal running on an iPhone

Certain compromises were necessary in the iPhone version of Tunepal in order to make transcription speed acceptable. Firstly queries are limited to twelve seconds of audio (similar to Shazam [22]). Secondly, the sample rate is reduced to 22.05KHz and finally, onset detection is achieved using a combination of an STFT (Short-time Fourier Transform) with a Hanning window and a pitch speller instead of using ODCF. An STFT is carried out on the signal using a frame size of 2,048 samples, with a 50% overlap. This gives a frequency resolution of 10.76Hz, discriminant enough to detect pitches of traditional instruments without interpolation. Our harmonicity based, pitch detection algorithm [5] that analyses peak intervals in the frequency spectrum was ported to C++ for use in Tunepal for iPhone. Identified frequencies are then assigned pitch classes using the pitch spelling algorithm. A note onset is annotated when the pitch class changes in the time domain. The quaver length is determined using the fuzzy histogram clustering algorithm described in [5,7,11]. Ornamentation notes are removed from the transcription and long notes (crochets, dotted crochets) are

split into multiple quaver notes. The transcription string (a sequence of pitch classes) is then submitted to tunepal.org for matching.

Tunepal for iPhone uses the same back end database and infrastructure as tunepal.org and so has access to a corpus of 13,290 tunes. The iPhone version of Tunepal, returns the top ten closest matching tunes for a query with confidence scores. Similar to tunepal.org, we log each query, with the closest matching tune and confidence score. When a tune is matched both tunepal.org and Tunepal for iPhone offer the option to link back to the original source of the ABC notation on the internet. In the case of tunes indexed from the website thesession.org, this often includes extensive discussions on the origin of the tune, the source of transcription and recordings on which the tune appears (Figure 4).

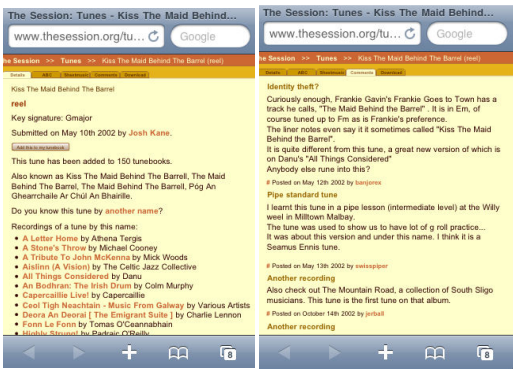


Figure 4: Biographic reference for the tune "Kiss the Maid Behind the Barrel" from the website thesession.org displayed on an iPhone

Retrieved tunes are stored in a “My Tunes” tab on the user’s device, in order of most recently tagged to facilitate future retrieval for learning purposes. Playback is achieved using ABC2MIDI [23] and the FMOD audio engine [24]. The iPhone version of Tunepal has one major advantage over tunepal.org and that is the ability for accurate geocoding (Figure 5).

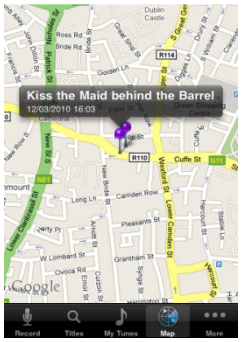


Figure 5: Geotagged tunes displayed within Tunepal on the iPhone

Therefore with the users permission, we geotag each query on the iPhone and store the longitude and latitude with each query in the tunepal.org database. This makes it possible for a user to track their queries on a map. Tunepal for iPhone was released on 11 February, 2010 and at the time of writing (18 May 2010), 5,866 QBP queries have been made, while 2,313 title searches were made (title searches were added as a feature on 13 February 2010). As the iPhone does not support programs written in Java, it was necessary to port the transcription subsystem of MATT2 and tunepal.org to a combination of C++ and Objective C. Tunepal for iPhone was listed in the top twenty cultural apps available on the iPhone by the Sunday Times (an Irish national newspaper) [25].

5. RESULTS

To date (18 May 2010) tunepal.org and Tunepal for iPhone have logged 16,064 queries since being released (Table 3).

Client	QBP
tunepal.org QBP	7,885
iPhone QBP	5,866
iPhone Title	2,313
Total:	16,064

Table 3: Queries logged from tunepal.org and Tunepal for iPhone

Table 4 gives the top ten tune types queried by users of tunepal.org and Tunepal for iPhone. The tunepal.org count was generated by counting the user verified tunes for each query. The iPhone count was generated by selecting the closest matching tune for each query.

	tunepal.org (Verified)		iPhone (QBP)	
#	Type	Count	Type	Count
1	Reel	521	Reel	1,594
2	Jig	240	Jig	913
3	Hornpipe	68	Hornpipe	211
4	Polka	57	Polka	116
5	Slip Jig	28	Waltz	111
6	Slide	23	Slip Jig	89
7	Waltz	20	Slide	56
8	Double Jig	13	Barndance	46
9	Barndance	9	Double Jig	38
10	Strathspey	7	Strathspey	18

Table 4: Top ten tune types queried by users of tunepal.org and Tunepal for iPhone

In order to minimise the effect of false positives on the iPhone counts, tunes with a confidence of < 65% are excluded. The cut-off of 65% was derived by stochastic sampling and proof listening. While this undoubtedly removes many true positives, it does eliminate most of the

false positives. The scores in Table 4 correspond broadly with the profile of tunes in most traditional musicians' repertoire, where reels and jigs assume prominence [26]. While it would be interesting to analyse the frequency that particular tunes appear in search results, more data is needed to make this analysis significant as the profile of tune appearances is in fact mostly flat, with the majority of tunes appearing only once or twice and even the top tunes appearing less than twenty times.

Table 5 gives a breakdown of QBP queries submitted by day of the week, though as these are in the local time of the server (the server is hosted in Ireland), there will be "bleed" from day to day due to the different time zones of users. Nevertheless, it is significant that weekends are more popular than weekdays for uses of tunepal.org, playing music being a leisure activity for many musicians. Tunepal for iPhone however demonstrates consistent usage across the week, which could be attributed to its portability.

	tunepal.org	iPhone	Total
Mon	999	793	1,792
Tue	1,039	862	1,901
Wed	985	728	1,713
Thurs	860	957	1,817
Fri	743	887	1,630
Sat	1,773	744	2,517
Sun	1,486	895	2,381
Total:	7,885	5,866	1,3751

Table 5: Analysis of queries by day of the week

Figure 6 better illustrates the trend towards high volumes of usage over the weekend, with significant usage on Monday and Tuesday, dropping off on Wednesday and Thursday to peak at the weekends.

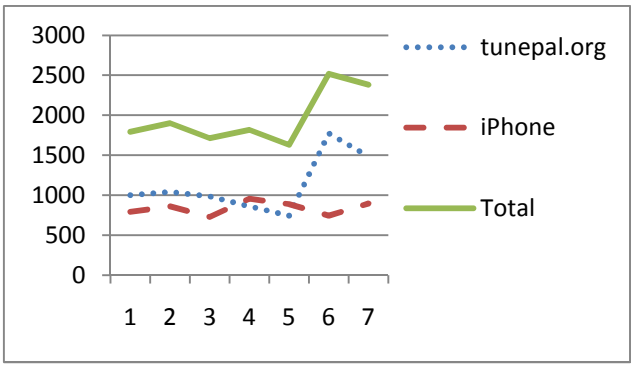


Figure 6: Plot of daily usage

We geotag queries generated by Tunepal for iPhone. An extract of this plot is given in Figure 7.

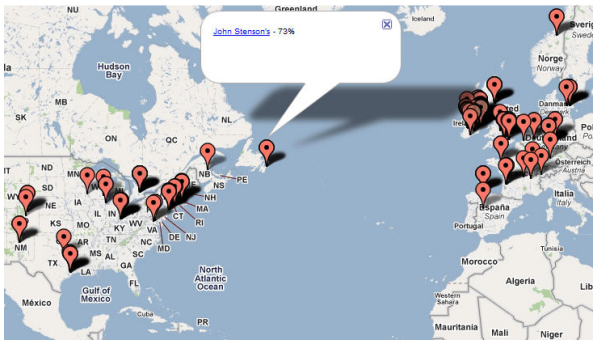


Figure 7: An extract from the worldwide geotagged query map

This is an optional feature that users must agree to; however 74% of queries made on an iPhone are geotagged. The realtime worldwide map of geotagged QBP queries can be viewed on a google map at the website http://tunepal.org.

Table 6 was generated by reverse geocoding the longitude and latitude from tagged queries to generate a profile of usage by country.

Country	Count
Ireland	1,276
United States	1,092
United Kingdom	393
Germany	179
Canada	122
Sweden	91
Spain	89
France	73
Netherlands	44
Australia	20

Table 6: Top ten countries for Tunepal for iPhone QBP queries

Although the amount of data collected is insufficient to draw any firm conclusions, it is nonetheless interesting to observe that the United States and the United Kingdom are significant sources for queries, these being major centers for the Irish Diaspora. This is a correlation we hope to explore in more detail in future work.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented two new QBP MIR systems for traditional music that developed from academic research. These tools have become popular, being used by musicians around the world to connect playing with tune names and biographic data. To achieve this, we use a corpus of 13,290 compositions collected by both the traditional music community and noted collectors such as O'Neill and Breathneach. Further we presented an analy-

sis of the data we have collected on the usage of these systems since being launched.

It is our aim to further disseminate these query-by-playing systems to the traditional music community by making them available on a greater variety of platforms such as the iPad, Android, Symbian, Maemo and Windows Phone 7 platforms. Usage of Tunepal is growing as are our usage logs. Once sufficient data is collected we hope to be able to mine these to gather new insights into musical trends and correlations that we hope to present in future work.

7. ACKNOWLEDGEMENTS

We are grateful for the support the School of Computing at the Dublin Institute of Technology who fund this work.

8. REFERENCES

[1] S. Driscoll, "A Trio of Internet Stars: ABC's," *Fiddler Magazine*, vol. 11, Summer. 2004.

[2] C. Carson, *Last Night's Fun: A Book about Irish Traditional Music*, North Point Press, 1997.

[3] "The Session."

[4] F. O'Neill, *The Music of Ireland*, 1903.

[5] B. Duggan, "Machine Annotation of Traditional Irish Dance Music," Dublin Institute of Technology, 2009.

[6] B. Duggan, M. Gainza, B. O'Shea, and P. Cunningham, "Machine Annotation of Sets of Traditional Irish Dance Tunes," *Ninth International Conference on Music Information Retrieval (ISMIR), Drexel University, Philadelphia, USA*, Sep. 2008.

[7] B. Duggan, B. O'Shea, and P. Cunningham, "A System for Automatically Annotating Traditional Irish Music Field Recordings," *Sixth International Workshop on Content-Based Multimedia Indexing, Queen Mary University of London, UK*, Jun. 2008.

[8] B. Duggan, "Enabling Access to Irish Traditional Music Archives on a PDA," *Eight Annual Irish Educational Technology Users Conference, DIT Bolton St, Ireland*, May. 2007.

[9] B. Duggan, "Learning Traditional Irish Music using a PDA," *IADIS Mobile Learning Conference, Trinity College Dublin*, Jul. 2006.

[10] C. Walshaw, "The ABC home page" Available: http://www.walshaw.plus.com/abc/.

[11] B. Duggan, M. Gainza, B. O'Shea, and P. Cunningham, "Compensating for Expressiveness in Queries to a Content Based Music Information Retrieval System," *2009 International Computer Music Conference*, Aug. 2009.

[12] M. Gainza, E. Coyle, and B. Lawler, "Onset Detection Using Comb Filters," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY: 2005.

[13] A. Klapuri, "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness," *Speech and Audio Processing, IEEE Transactions on*, vol. 11, 2003, pp. 804-816.

[14] H. Norbeck, "ABC Tunes" Available: http://www.norbeck.nu/abc/index.html,

[15] G. Navarro and M. Raffinot, *Flexible Pattern Matching in Strings: Practical On-Line Search Algorithms for Texts and Biological Sequences*, Cambridge University Press, 2002.

[16] F. O'Neill, *The Dance Music of Ireland – 1001 Gems*, Chicago, USA: 1907.

[17] B. Breathnach, "Ceol Rince na hÉireann Cuid V [Dance Music of Ireland] Vol V," 1999.

[18] L. Prechelt and R. Typke, "An interface for melody input," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 8, 2001, pp. 133-149.

[19] G. Dyke and P. Rosen, "abcjs - Project Hosting on Google Code" Available: http://code.google.com/p/abcjs/.

[20] S. Long, "Film body makes case against bord snips," *The Irish Times*, Aug. 2009.

[21] H. O'Shea, "Getting to the Heart of the Music: Idealizing Musical Community and Irish Traditional Music Sessions," *Journal of the Society for Musicology in Ireland*, vol. 2, 2006, p. 1.

[22] Shazam, "Shazam - The amazing music discovery engine. Join our Community" Available: http://www.shazam.com/music/portal.

[23] S. Shlien, "The ABC Music project - abcMIDI" Available: http://abc.sourceforge.net/abcMIDI/.

[24] "fmod - interactive audio middleware" Available: http://www.fmod.org/.

[25] "The Arts World on iPhone," *The Sunday Times*, Mar. 2010.

[26] F. Vallely, *The Companion to Irish Traditional Music*, New York University Press, 1999.

UNIVERSAL ONSET DETECTION WITH BIDIRECTIONAL LONG SHORT-TERM MEMORY NEURAL NETWORKS

Florian Eyben, Sebastian Böck, Björn Schuller
Institute for Human-Machine Communication
Technische Universität München
eyben@tum.de, sb@minimoog.org, schuller@tum.de

Alex Graves
Institute for Computer Science VI
Technische Universität München
graves@in.tum.de

ABSTRACT

Many different onset detection methods have been proposed in recent years. However those that perform well tend to be highly specialised for certain types of music, while those that are more widely applicable give only moderate performance. In this paper we present a new onset detector with superior performance and temporal precision for all kinds of music, including complex music mixes. It is based on auditory spectral features and relative spectral differences processed by a bidirectional Long Short-Term Memory recurrent neural network, which acts as reduction function. The network is trained with a large database of onset data covering various genres and onset types. Due to the data driven nature, our approach does not require the onset detection method and its parameters to be tuned to a particular type of music. We compare results on the Bello onset data set and can conclude that our approach is on par with related results on the same set and outperforms them in most cases in terms of F_1-measure. For complex music with mixed onset types, an absolute improvement of 3.6% is reported.

1. INTRODUCTION

Finding onset locations is a key part of segmenting and transcribing music, and therefore forms the basis for many high level automatic retrieval tasks. An onset marks the beginning of an acoustic event. In contrast to music information retrieval studies which focus on beat and tempo detection via the analysis of periodicities (e. g. [7, 9]), an onset detector faces the challenge of detecting single events, which need not follow a periodic pattern. Recent onset detection methods (e. g. [5, 16, 17]) have matured to a level where reasonable robustness is obtained for polyphonic music. However, the methods are specialised or tuned to specific kinds of onsets (e. g. pitched or percussive) and lack the ability to perform well for music with mixed onset types. Thus, multiple methods need to be combined or a method has to be selected depending on the type of onsets to be analysed.

In this paper we propose a novel, robust approach to onset detection, which can be applied to any type of music. Our approach is based on auditory spectral features and Long Short-Term Memory (LSTM) [13] recurrent neural networks. The approach is purely data driven, and as we will see, yields a very high temporal precision as well as detection accuracy.

The rest of this paper is structured as follows. A brief overview of the state of the art in onset detection is given in Section 2, and Section 3 provides an introduction to LSTM neural networks. Section 5 describes the Bello onset data set [2] as well as introducing a new data set. Experimental results for both data sets are provided in Section 6, along with a comparison to related systems.

2. EXISTING METHODS

Most onset detection algorithms are based on the three step model shown in Figure 1. Some methods include a preprocessing step. The aim of preprocessing is to emphasise relevant parts of the signal. Next, a reduction function is applied, to obtain the detection function. This is the core component of an onset detector. Some of the most common reduction functions found in the literature are summarised later in this section.

Figure 1. Traditional onset detection workflow

The last stage is to extract the onsets from the detection function. This step can be subdivided into post processing (e. g. smoothing and normalising of the detection function), thresholding, and peak picking. If fixed thresholds are used, the methods tend to pick either too many onsets in louder parts, or miss onsets in quieter parts. Hence, adaptive thresholds are often used. Finally the local maxima above the threshold, which correspond to the detected onsets, are identified by a peak picking algorithm.

Early reduction functions, such as [14], operated in the time domain. This approach normalises the loudness of the signal before splitting it into multiple bands via bandpass

filters. Onsets are then detected in each band as peaks in the first order difference of the logarithm of the amplitude envelope. These band-wise onsets are then combined to yield the final set of detected onsets. More recent systems employ spectral domain reduction functions. We describe the most common ones in the following paragraphs.

2.1 Spectral domain reduction functions

Since onsets are often masked in the time domain by higher energy signals, many reduction functions operate on a spectral representation of the audio signal. The methods listed below are all based on a short-time Fourier transform (STFT) of the signal.

2.1.1 High Frequency Content

Percussive sounds have a high energy in the upper frequency bands. This is exploited by weighting each STFT bin with a factor proportional to its frequency. Summing all weighted bins yields a measure called the high frequency content (HFC), which is used as a detection function. Although this method works well for percussive onsets, it shows weaknesses for other onset types [2].

2.1.2 Spectral difference

For computation of the spectral difference function (SD), the difference of two consecutive short-time spectra is computed bin by bin. All positive differences are then summed up across all bins. Some approaches use the L_2-norm [2] for calculating the difference, whereas others use the L_1-norm [5], in which case the function is referred to as spectral flux (SF). Onset detection methods based on these methods are among the best overall performers so far.

2.1.3 Phase deviation

The methods mentioned so far rely on the spectral magnitudes. In [2] a method utilising phase information is described. The change of the phase in a STFT frequency bin is a rough estimate of its instantaneous frequency. A change of this frequency is an indicator of a possible onset. To reduce the chance of a missed onset due to phase wrap around, the mean phase change over all frequency bins is used. Dixon proposes an improvement to the phase deviation (PD) detection function called normalised weighted phase deviation (NWPD) [5], where each frequency bin's contribution to the phase deviation function is weighted by its magnitude. The result is normalised by the sum of the magnitudes.

2.1.4 Complex Domain

Another way to incorporate both magnitude and the phase information is proposed in [6]. First, the expected amplitude and phase is calculated for the current frame based on the two previous frames, assuming constant amplitude and phase change rate. The sum of the magnitude of the complex differences between the actual values for each frequency bin and the estimated values is then computed and used as a detection function. A variant of this method is called the rectified complex domain (RCD) [5]. Observing that increases of the signal amplitude are generally more relevant than decreases for onset detection, RCD modifies the original algorithm by only summing over positive amplitude changes.

2.2 Probabilistic reduction functions

An alternative approach is to base the description of signals on probabilistic models. The negative log-likelihood method [1] defines two different statistical models and observes whether the signal follows the first or the second model. A sudden change from the first model to the second can be an indication of an onset. This method shows good results for music with soft onsets, e. g. non-percussive sounds [2].

2.3 Pitch-based onset detection techniques

Collins describes an onset detection function based on a pitch detector front-end [4]. Zhou presented a combination of pitch and energy based detection functions [17]. In principle pitch-based onset detection is based on identification of discontinuities and perturbations in the pitch contour, which are assumed to be indicators of onsets.

2.4 Data-driven reduction functions

To build general detection functions, which are capable of detecting onsets in a wider range of audio signals, classifier based methods emerged. In [15] an onset detection algorithm based on a feed forward neural network, namely a convolutional neural network, is described. This system performed best in the MIREX 2005 audio onset detection evaluation.

3. NEURAL NETWORKS

Motivated by the high performance of the onset detection method of Lacoste and Eck, we investigate a novel artificial neural network (ANN) based approach. Instead of a simple feed forward neural network we use a bidirectional recurrent neural network with Long Short-Term Memory [13] hidden units. Such networks were proven to work well on other audio detection tasks, such as speech recognition [10].

This section gives a short introduction to ANN with a focus on bidirectional Long Short-Term Memory (BLSTM) networks, which are used for the proposed onset detector.

3.1 Feed forward neural networks

The most commonly used form of feed forward neural networks (FNN) is the multilayer perceptron (MLP). It consists of a minimum of three layers, one input layer, one or more hidden layers, and an output layer. All connections feed forward from one layer to the next without any backward connections. MLPs classify all input frames independently. If the context a frame is presented in is relevant, this context must be explicitly fed to the network, e. g. by using a fixed width sliding window, as in [15].

3.2 Recurrent neural networks

Another technique for introducing past context to neural networks is to add backward (cyclic) connections to FNNs. The resulting network is called a recurrent neural network (RNN). RNNs can theoretically map from the entire history of previous inputs to each output. The recurrent connections form a kind of memory, which allows input values to persist in the hidden layer(s) and influence the network output in the future. If future context is also necessary required, a delay between the input values and the output targets can be introduced.

3.3 Bidirectional recurrent neural networks

A more elegant incorporation future context is provided by bidirectional recurrent networks (BRNNs). Two separate hidden layers are used instead of one, both connected to the same input and output layers. The first processes the input sequence forwards and the second backwards. The network therefore has always access to the complete past and the future context in a symmetrical way, without bloating the input layer size or displacing the input values from the corresponding output targets. The disadvantage of BRNNs is that they must have the complete input sequence at hand before it can be processed.

3.4 Long Short-Term Memory

Although BRNNs have access to both past and future information, the range of context is limited to a few frames due to the vanishing gradient problem [11]. The influence of an input value decays or blows up exponentially over time, as it cycles through the network with its recurrent connections and gets dominated by new input values.

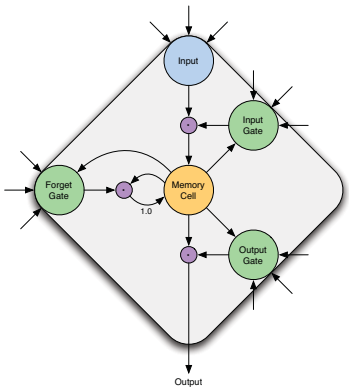


Figure 2. An LSTM block with one memory cell

To overcome this deficiency, a method called Long Short-Term Memory (LSTM) was introduced in [13]. In an LSTM hidden layer, the nonlinear units are replaced by LSTM memory blocks (Figure 2). Each block contains one or more self connected linear memory cells and three multiplicative gates. The internal state of the cell is maintained with a recurrent connection of constant weight 1.0. This connection enables the cell to store information over long periods of time. The content of the memory cell is controlled by the multiplicative input, output, and forget gates, which – in computer memory terminology – correspond to write, read, and reset operations. More details on the training algorithm employed, and the bidirectional LSTM architecture in general can be found in [10].

4. PROPOSED APPROACH

This section describes our novel approach for onset detection in music signals, which is based on bidirectional Long Short-Term Memory (BLSTM) recurrent neural networks. In contrast to previous approaches it is able to model the context an onset occurs in. The properties of an onset and the amount of relevant context are thereby learned from the data set used for training. The audio data is transformed to the frequency domain via two parallel STFTs with different window sizes. The obtained magnitude spectra and their first order differences are used as inputs to the BLSTM network, which produces an onset activation function at its output. Figure 3 shows this basic signal flow. The individual blocks are described in more detail in the following sections.

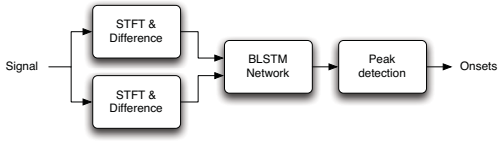


Figure 3. Basic signal flow of the new neural network based onset detector

4.1 Feature extraction

As input, the raw PCM audio signal with a sampling rate of $f_s = 44.1$ kHz is used. To reduce the computational complexity, stereo signals are converted to a monaural signal by averaging both channels. The discrete input audio signal $x(t)$ is segmented into overlapping frames of W samples length ($W = 1024$ and $W = 2048$, see Section 4.2), which are sampled at a rate of one per 10 ms (onset annotations are available on a frame level). A Hamming window is applied to these frames. Applying the STFT yields the complex spectrogram $X(n, k)$, with n being the frame index, and k the frequency bin index. The complex spectrogram is converted to the power spectrogram $S(n, k) = |X(n, k)|^2$.

The dimensionality of the spectra is reduced by applying psychoacoustic knowledge: a conversion to the Mel-frequency scale is performed with openSMILE [8]. A filterbank with 40 triangular filters, which are equidistant on the Mel scale, is used to transform the spectrogram $S(n, k)$ to the Mel spectrogram $M(n, m)$. To match human perception of loudness, a logarithmic representation is chosen:

$$M_{log}(n, m) = log\left(M(n, m) + 1.0\right) \quad (1)$$

The positive first order difference $D^+(n,m)$ is calculated by applying a half-wave rectifier function $H(x) = \frac{x+|x|}{2}$ to the difference of two consecutive Mel spectra:

$$D^+(n,m) = H\left(M_{log}(n,m) - M_{log}(n-1,m)\right) \quad (2)$$

4.2 Neural Network stage

As a neural network, an RNN with BLSTM units is used. As inputs to the neural network, two log Mel-spectrograms $M^{23}_{log}(n,m)$ and $M^{46}_{log}(n,m)$ (computed with window sizes of 23.2 ms and 46.4 ms ($W = 1024$ and $W = 2048$ samples), respectively) and their corresponding positive first order differences $D^+_{23s}(n,m)$ and $D^+_{46s}(n,m)$ are applied, resulting in 160 input units. The network has three hidden layers for each direction (6 layers in total) with 20 LSTM units each. The output layer has two units, whose outputs are normalised to both lie between 0 and 1, and to sum to 1, using the softmax function. The normalised outputs represent the probabilities for the classes 'onset' and 'no onset'. This allows the use of the cross entropy error criterion to train the network [10]. Alternative networks with a single output, where a value of 1 represents an onset frame and a value of 0 a non-onset frame, which are trained using the mean squared output error as criterion, were not as successful.

4.2.1 Network training

For network training, supervised learning with early stopping is used. Each audio sequence is presented frame by frame (in correct temporal order) to the network. Standard gradient descent with backpropagation of the output errors is used to iteratively update the network weights. To prevent over-fitting, the performance (cross entropy error, cf. [10]) on a separate validation set is evaluated after each training iteration (epoch). If no improvement of this performance over 20 epochs is observed, the training is stopped and the network with the best performance on the validation set is used as the final network. The gradient descent algorithm requires the network weights to be initialised with non zero values. We initialise the weights with a random Gaussian distribution with mean 0 and standard deviation 0.1. The training data, as well as validation and test sets are described in Section 5.

4.3 Peak detection stage

A network obtained after training as described in the previous section is able to classify each frame into two classes: 'onset' and 'no onset'. The standard method of choosing the output node with the highest activation to determine the frame class has not proven effective. Hence, only the output activation of the 'onset' class is used. Thresholding and peak detection is applied to it, which is described in the following sections:

4.3.1 Thresholding

One problem with existing magnitude based reduction functions (cf. Section 2) is that the amplitude of the detection function depends on the amplitude of the signal or the magnitude of its short time spectrum. Thus, to successfully deal with high dynamic ranges, adaptive thresholds must be used when thresholding the detection function prior to peak picking. Similar to phase based reduction functions, the output activation function of the BLSTM network is not affected by input amplitude variations, since its value represents a probability of observing an onset rather than representing onset strength. In order to obtain optimal classification for each song, a fixed threshold θ is computed per song proportional to the median of the activation function (frames $n = 1 \ldots N$), constrained to the range from $\theta_{min} = 0.1$ to $\theta_{max} = 0.3$:

$$\theta^* = \lambda \cdot \text{median}\{a_o(1), \ldots, a_o(N)\} \quad (3)$$

$$\theta = \min\left(\max\left(0.1, \theta^*\right), 0.3\right) \quad (4)$$

with $a_o(n)$ being the output activation function of the BLSTM neural network for the onset class, and the scaling factor λ chosen to maximise the F_1-measure on the validation set. The final onset function $o_o(n)$ contains only the activation values greater than this threshold:

$$o_o(n) = \begin{cases} a_o(n) & \text{for } a_o(n) > \theta \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

4.3.2 Peak picking

The onsets are represented by the local maxima of the onset detection function $o_o(n)$. Thus, using a standard peak search, the final onset function $o(n)$ is given by:

$$o(n) = \begin{cases} 1 & \text{for } o_o(n-1) \leq o_o(n) \geq o_o(n+1) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

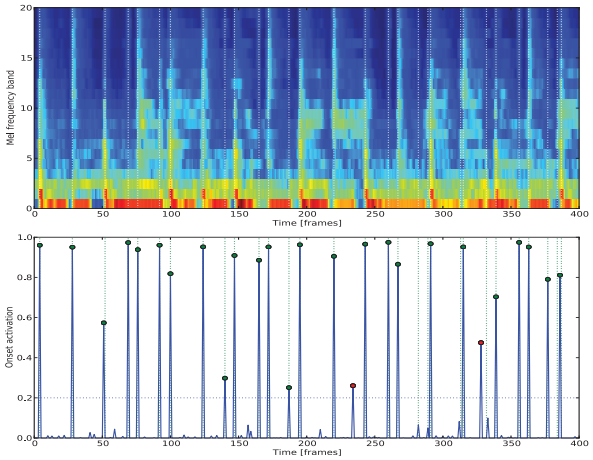


Figure 4. *Top*: log Mel-spectrogram with ground truth onsets (vertical dashed lines). *Bottom*: network output with detected onsets (marked by dots), ground truth onsets (dotted vertical lines), and threshold θ (horizontal dashed line). 4 s excerpt from 'Basement Jaxx - Rendez-Vu'.

5. DATA SETS

We evaluate our onset detector using the data set introduced by Bello in [2], which consists of 23 sound excerpts with lengths ranging from a few seconds to one minute (cf. Table 1). the data set is divided into four categories: pitched percussive (PP), pitched non-percussive (PNP), non-pitched percussive (NPP), and complex music mixes (MIX). The set includes audio synthesised from MIDI files as well as original recordings.

In order to effectively train the BLSTM network, the onset annotations had to be corrected in a few places: missing onsets were added and onsets in polyphonic pieces were properly aligned to match the annotation precision of the MIDI based samples. For rule-based onset detection approaches, minor inaccuracies of a few frames are not crucial since these are levelled out by the detection window during evaluation. For the BLSTM network, however, it is necessary to have temporally precise data for training. Nonetheless, the original, unmodified transcriptions are used for evaluation, to ensure a fair comparison.

To increase the size of the training data set, 87 10 s excerpts of ballroom dance style music (BRD_o in the ongoing) from the ISMIR 2004 tempo induction contest [1] [9] were included (cf. Table 1). A part of the annotation work was done by Lacoste and Eck for their neural network approach [2]. The remaining parts were manually labelled by an expert musician [3]. As with the Bello data set, all annotations have been revised for network training.

Set	# files	# onsets	min/max/mean length [s]
BRD_o	87	5474	10.0 / 10.0 / 10.0
PNP	1	93	13.1 / 13.1 / 13.1
PP	9	489	2.5 / 60.0 / 10.5
NPP	6	212	1.4 / 8.3 / 4.3
MIX	7	271	2.8 / 15.1 / 8.0

Table 1. Statistics of the onset data sets.

For network training, the full set (BRD_o and Bello set) is initially randomly split on the file level into eight disjunctive folds. Next, in an 8-fold cross validation, results for the full set are obtained. Thereby for each fold six subsets are used for training, one for validation, and one for testing. Since the initial weights of the neural nets are randomly distributed, the 8-fold cross validation is repeated 10 times (using the exact same folds) and the means of the output activation functions are used for the final evaluation.

6. RESULTS

In [2] and [5], an onset is reported as correct if it is detected within a 100 ms window ($\pm$50 ms) around the annotated ground truth onset position. In [3] a smaller window of $\pm 25ms$ was used for percussive sounds. We therefore decided to report two results for each set, one using a 100 ms window ω_{100} for comparison with results in [2] and [5], and the second using a 50 ms window ω_{50}. All results were obtained with a fixed threshold scaling factor of $\lambda = 50$.

Table 2 shows the results of our BLSTM network approach for each set of onsets in comparison to six other onset detection methods as reported in [2] and [5].The PNP data set consists of 93 onsets from only one audio file of string sounds. As a consequence, the results are not as representative as the others, and can vary a lot, depending on the used parameters, as shown by [5]. The number of onsets of the PP set has changed from originally 489 (used in [2, 5]) to 482 now, due to modifications by its author. The new results are therefore slightly worse (up to max. 1.4%) than the original results but can still compete.

BRD_o & Bello-set	Precision	Recall	$\mathbf{F_1}$**-measure**
BLSTM (ω_{100})	0.945	0.925	0.935
BLSTM (ω_{50})	0.920	0.901	0.911
BLSTM ($comb, \omega_{100}$)	0.938	0.916	0.927
BLSTM ($comb, \omega_{50}$)	0.911	0.890	0.900

Table 3. 8-fold cross validation results for BLSTM on the full data set with 100 ms and 50 ms detection windows (ω). $comb$: all onsets within 30 ms combined.

Table 3 shows the results obtained by cross validation for the full data set. The first two rows reflect the results obtained with the same settings as for the individual Bello sets. It has been shown that two onsets are perceived as one if they are not more than 30 ms apart [12]. Hence we also report results, where all onsets less than 30 ms apart have been combined to a single one. There are 6 605 onsets in the original annotations and 5 861 after combining.

6.1 Discussion

The results show that our algorithm can compete with, and in most cases outperform, a range of existing methods for all types of onsets. However, we must temper this conclusion by adding that we were not able to compare to the latest MIREX participants (e.g. [16]), since the MIREX test data is not publicly available and the authors did not publish results on the Bello data set. Perhaps the most exciting aspect of our approach is that it does not require adaptation to specific onset types to achieve good results. This is an important step towards a universal onset detector.

If a detection window of only 50 ms is chosen our approach even outperforms the reference algorithms in some cases. This shows the excellent temporal precision of the BLSTM onset detector. In our opinion the results given for a detection window of 50 ms with all onsets less than 30 ms apart combined to a single one should be used in the future, as they better reflect the temporal precision of the algorithm and the perception of the human ear.

7. CONCLUSION

We have presented a novel onset detector based on BLSTM-RNN, which – on the Bello onset data set – achieves results

[1] http://mtg.upf.edu/ismir2004/contest/tempoContest/node5.html
[2] http://w3.ift.ulaval.ca/~allac88/dataset.tar.gz
[3] Data available at: http://mir.minimoog.org/

	PNP			PP			NPP			MIX		
	P	R	F	P	R	F	P	R	F	P	R	F
HFC [2]	0.844	0.817	0.830	0.947	0.941	0.944	**1.000**	0.967	0.983	0.888	0.845	0.866
SD [2]	0.910	0.871	0.890	0.983	0.949	0.966	0.935	0.816	0.871	0.886	0.804	0.843
NLL [2]	**0.968**	0.968	**0.968**	0.968	0.924	0.945	0.980	0.929	0.954	0.889	0.860	0.874
SF [5]	0.938	0.968	0.952	0.981	**0.988**	0.984	0.959	0.975	0.967	0.882	0.882	0.882
NWPD [5]	0.909	0.968	0.938	0.961	0.981	0.971	0.950	0.966	0.958	0.916	0.845	0.879
RCD [5]	0.948	**0.978**	0.963	0.983	0.979	0.981	0.944	0.983	0.963	**0.945**	0.819	0.877
BLSTM (ω_{100})	**0.968**	0.968	**0.968**	**0.987**	0.987	**0.987**	0.991	**0.995**	**0.993**	0.941	**0.897**	**0.918**
BLSTM (ω_{50})	0.918	0.957	0.937	0.955	0.981	0.968	0.982	**0.995**	0.989	0.844	0.865	0.855

Table 2. Results for the Bello data sets PNP, PP, NPP, and MIX. Precision (P), Recall (R), and F_1-measure (F) (as used in [5]). BLSTM with 100 ms and 50 ms detection windows (ω) in comparison to other approaches: high frequency content (HFC), spectral difference (SD), negative log-likelihood (NLL), spectral flux (SF), normalised weighted phase deviation (NWPD), and rectified complex domain (RCD).

on par with or better than existing results on the same data (wrt. F_1-measure), regardless of onset type. We have also introduced a new thoroughly annotated data set of onsets in ballroom dance music.

The average improvement on the whole Bello data set, is 1.7% F_1-measure absolute. The improvement was best (3.6% F_1-measure, absolute) for complex music mixes, reflecting the adaptivity of our method to different musical genres. Competitive results are obtained even if the detection window is halved in size (50 ms instead of 100 ms).

In future work we will investigate whether the approach is suitable for identifying the onset type (e. g. instrument type, vocal, etc.) via detectors trained on respective data.

8. ACKNOWLEDGMENT

We would like to thank Juan Bello, Alexandre Lacoste, and Douglas Eck for sharing their annotated onset data sets.

9. REFERENCES

[1] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall, 1993.

[2] J. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, Sept. 2005.

[3] N. Collins. A comparison of sound onset detection algorithms with emphasis on psychoacoustically motivated detection functions. In *Proc. of the AES Convention 118*, pages 28–31, 2005.

[4] N. Collins. Using a pitch detector for onset detection. In *Proc. of ISMIR*, pages 100–106, 2005.

[5] S. Dixon. Onset detection revisited. In *Proc. of DAFx-06, Montreal, Canada*, pages 133–137, Sept. 2006.

[6] C. Duxbury, J. P. Bello, M. Davies, M. Sandler, and M. S. Complex domain onset detection for musical signals. In *Proc. DAFx-03 Workshop*, 2003.

[7] F. Eyben, B. Schuller, and G. Rigoll. Wearable assistance for the ballroom-dance hobbyist - holistic rhythm analysis and dance-style classification. In *Proc. of ICME 2007*, pages 92–95. IEEE, July 2007.

[8] F. Eyben, M. Wöllmer, and B. Schuller. openEAR - Introducing the Munich Open-Source Emotion and Affect Recognition Toolkit. In *Proc. of ACII 2009*, pages 576–581. IEEE, September 2009.

[9] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, Sept. 2006.

[10] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. PhD thesis, Technische Universität München. Munich, Germany. 2008.

[11] S. Hochreiter, Y. Bengio, P. Frasconi and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks* IEEE Press, 2001.

[12] S. Handel. *Listening: an introduction to the perception of auditory events*. MIT Press, 1989.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computing*, 9(8):1735–1780, 1997.

[14] A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *Proc. of ICASSP'99*, vol. 6, pages 3089–3092, 1999.

[15] A. Lacoste and D. Eck. Onset detection with artificial neural networks. MIREX, 2005.

[16] A. Röbel. Onset Detection By Means Of Transient Peak Classification In Harmonic Bands. MIREX, 2009

[17] R. Zhou and J. Reiss. Music onset detection combining energy-based and pitch-based approaches. MIREX, 2007.

UNSUPERVISED ACCURACY IMPROVEMENT FOR COVER SONG DETECTION USING SPECTRAL CONNECTIVITY NETWORK

Mathieu Lagrange
Analysis-Synthesis team,
IRCAM-CNRS UMR 9912,
1 place Igor Stravinsky, 75004 Paris, France
mathieu.lagrange@ircam.fr

Joan Serrà
Music Technology Group,
Universitat Pompeu Fabra,
Roc Boronat 138, 08018 Barcelona, Spain
joan.serraj@upf.edu

ABSTRACT

This paper introduces a new method for improving the accuracy in medium scale music similarity problems. Recently, it has been shown that the raw accuracy of query by example systems can be enhanced by considering priors about the distribution of its output or the structure of the music collection being considered. The proposed approach focuses on reducing the dependency to those priors by considering an eigenvalue decomposition of the aforementioned system's output. Experiments carried out in the framework of cover song detection show that the proposed approach has good performance for enhancing a high accuracy system. Furthermore, it maintains the accuracy level for lower performing systems.

1. INTRODUCTION

Expressing the similarity between music streams is of interest for many multimedia applications [3]. Though, in many tasks in music information retrieval (MIR), one can observe a glass ceiling in the performance achieved by current methods and algorithms [5]. Several research directions can be considered for tackling this issue. In this paper, we focus on the cover song detection task, but most of the argumentation may be transferred to more general similarity tasks involving a query by example (QBE) system.

One option to boost the accuracy of current QBE systems is to use an enhanced description of the musical stream using the segregation principle [2]. Intuitively, a lot can be gained if an audio signal is available for each instrument. This way, one can easily focus on the stream of interest for each MIR task. In this line, Foucard et al. [8] show that considering a dominant melody removal algorithm as a pre-processing step is a promising approach for observing more robustly the harmonic progression and, in this way, achieve a better accuracy in the cover song detection task. However, it may be a long way until such pre-processing based on segregation will be beneficial for managing medium to large scale musical collections.

An efficient alternative is to consider post-processing approaches exploiting the regularities found in the results of a QBE system for a given music collection. Indeed, music collections are usually organized and structured at multiple scales. In the case of cover detection, songs naturally cluster into so-called cover sets [17]. Therefore, if those cover sets can be approximately estimated, one can gain significant retrieval accuracy, as evidenced by Serrà et al. [17] and Egorov & Linetsky [6]. A different and very interesting post-processing alternative is the general classification scheme proposed by Ravuri & Ellis in [15], where they employ the output of different cover song detection algorithms and a z-score normalization scheme to classify pairs of songs.

Unsupervised post-processing methods that have been introduced so far are rooted on (a) the knowledge of an experimental similarity threshold defining whether two songs are covers or not [17], or (b) the potential number of or cardinality of clusters of the dataset being considered [6]. Thus, these methods are either algorithm or data-dependant. The scheme in [15] is a supervised system trained on different algorithms outputs for some ground truth data. Therefore, it might potentially fail into one or both of the aforementioned dependencies [1].

In this paper, we focus on improving the output of a single QBE system in an unsupervised way. In contrast with the aforementioned references, we propose to consider "more global" approaches in order to alleviate their needs and in order to advance towards unsupervised parameter-free post-processing steps for QBE systems. To this extent we introduce spectral connectivity network (SCN). In addition, we focus on the benefits this technique might provide if the raw accuracy of the QBE system is rather low. This could be the case of a particularly difficult dataset, of a more simple and efficient system (or merely a suboptimal one), or a combination of both cases.

The remaining of the paper is organized as follows: after a presentation of previous work in Sec. 2, we introduce our new accuracy improvement scheme in Sec. 3. In this section, the algorithm is motivated and illustrated on

[1] Furthermore, issues could arise with the employed z-score normalization for some intricate data structures or algorithm outputs (e.g., binomially distributed classifier inputs).

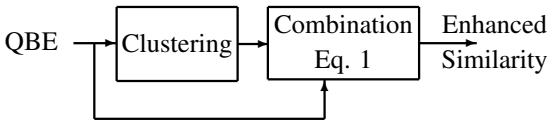


Figure 1. *Combination scheme used for clustering based systems.*

generic artificial datasets. In Sec. 4 we use the evaluation methodology considered in [17] to show the potential of the proposed approach.

2. PREVIOUS WORK

There exist different proposals for the unsupervised post-processing of the output of a single QBE cover song detection system [6, 17]. Most promising strategies so far consist in first estimating the cover sets and use this clustering information in order to increase the overall accuracy as shown in Fig. 1. This can be achieved by considering a classical agglomerative hierarchical clustering algorithm such as the well-known group average linkage (UPGMA) method [10, 19] or alternatively the Community Clustering method (CC) presented in [17], which looks for connected components in a complex network built upon the results of the considered QBE system. Once a clustering solution is obtained, the output distance for a couple of song entries (e_i, e_j) given by a QBE system can be modified to increase the overall accuracy [17]:

$$d'_{i,j} = \begin{cases} \frac{d(e_i,e_j)}{\max(d)} & \text{if } e_i, e_j \in E_k, \\ \frac{d(e_i,e_j)}{\max(d)} + \beta & \text{otherwise.} \end{cases} \quad (1)$$

We denote $d_{i,j}$ as the raw dissimilarity output of the QBE system between two songs e_i and e_j, E_k represents a given cluster, and $\beta > 1$.

Both UPGMA and CC depend on the setting of a threshold similarity value that overall discriminates between cover and non-cover song pairs. This parameter is usually algorithm-dependent. Therefore, for different music collections analyzed through the same QBE system, one should expect similar values for the similarity threshold. That seems to be the case for the algorithm presented in [16] when analyzing different datasets [2] (Fig. 2).

At a first glance one could screen Fig. 2 and set a dissimilarity threshold for roughly separating between covers and not covers. In the present case this threshold could be around 0.6 (or below, if we want to have less false positives). The threshold then would provide the necessary information to the post-processing clustering stage. However, this dissimilarity threshold might not directly correspond to what the clustering algorithm is using internally (e.g., intra-cluster cophenetic distances [10, 19]). In the end one might better perform a grid search for the involved parameter.

In a more general scenario, one might not always be sure about the data or algorithm dependencies of the problem. So, to be on the safe side, some data exploration, algorithm analysis, and/or parameter optimization needs to be done. To avoid those tedious steps is what motivates us to consider unsupervised parameter-free post-processing strategies.

[2] We notice however that both datasets have some similarities, e.g., in terms of genres.

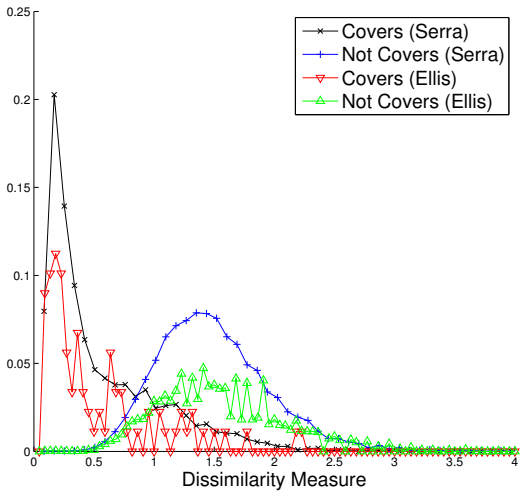


Figure 2. *Normalized histograms for the dissimilarity measure [16] on the music collection of [17] (lines with crosses) and on the "covers80" dataset [7] (lines with triangles).*

3. SPECTRAL CONNECTIVITY NETWORK (SCN)

Without any a priori knowledge about the problem at hand, one needs to root the method on a statistical analysis that is able to identify the underlying structure of the observation, being in our case the output of a QBE system over a large music collection.

Spectral graph clustering has gained popularity in many information retrieval areas, specially in gene, web, image, and audio processing [1, 11, 18]. The interested reader may be referred to [12] for a tutorial introduction.

If S is a square matrix encoding the similarities of all the entries e_i of our music collection E, it can be shown [14] that the eigenvectors of the corresponding Laplacian are relevant clustering indicators for determining the k disjoint set of clusters $E_1, ..., E_k$ (see Fig. 3). We propose to consider this property in order to increase the overall accuracy of QBE systems using the processing scheme shown in Fig. 4. Each of the steps are further detailed in the remaining of this section.

3.1 Similarity Computation

As most QBE systems output a dissimilarity value $d_{i,j}$ measuring how "far" a given couple of entries (e_i, e_j) are, one needs to convert this distance into a similarity value $s_{i,j}$. This is performed using the traditional radial basis function

$$s_{i,j} = e^{\left(\frac{-d^2(e_i,e_j)}{\sigma^2}\right)}, \quad (2)$$

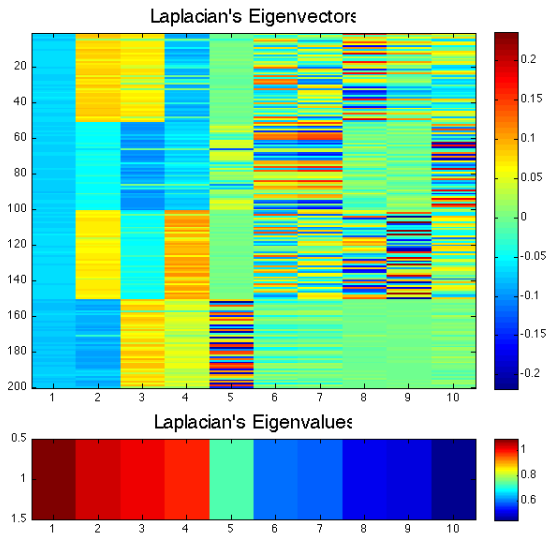


Figure 3. *Eigenvalues and eigenvectors of the Laplacian graph corresponding to a dataset made of 4 bi-dimensional sets of 50 components with low overlapping.*

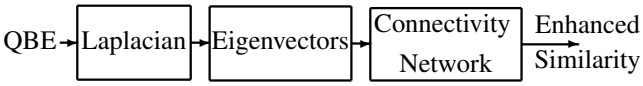


Figure 4. *Processing scheme used for the proposed system based on SCN.*

with σ determined using the local scaling procedure proposed in [20]. The similarities $s_{i,j}$ lead to a matrix S, which is further normalized as [18]:

$$S_d = \mathcal{D}^{-1/2} S \mathcal{D}^{-1/2}, \tag{3}$$

where $\mathcal{D}$ is a diagonal matrix with the degrees $\lceil_1, ..., \lceil_n$ along the diagonal, $\lceil_k = \sum_j s_{k,j}$.

3.2 Eigenvalue Decomposition

As proposed in [14] and illustrated in Fig. 3, the eigenvectors corresponding to the k highest eigenvalues of S_d can be considered as cluster indicators. For that purpose, the contribution of each eigenvector is first normalized with respect to each of the entries, (i.e., per rows).

For a clustering task, any traditional clustering algorithm may then be considered. The k-means algorithm is usually considered in the literature. Though, in the case of cover set detection, the number of clusters is high and their cardinality is low, which makes the algorithm rather slow and highly sensitive to the random initialization. In pre-analysis, it was found more suitable to use the aforementioned UPGMA algorithm. However, in this scenario, one still needs to perform the clustering decision based on a prior, be it the number of clusters or the similarity threshold and consider Eq. 1 for accuracy improvement.

3.3 Connectivity Network

An alternative approach is to consider the Connectivity Network (CN) as our enhanced dissimilarity $d'(i,j)$ by using the projection matrix of the normalized eigenvectors:

$$P = \sum_{k=1}^{N_q} q_k q_k^T, \tag{4}$$

where q_k is the eigenvector corresponding to the k highest eigenvalue λ_k and N_q is the number of eigenvectors to consider. This principle has been originally used for correspondence analysis of contingency tables [9] and reintroduced later in the context of spectral clustering [4].

The usual procedure is to set $N_p = k$ in order to retain only the relevant eigenvectors. If k cannot be considered as a prior (which is the case for cover set detection), one has to consider a method that can robustly estimate k. Unfortunately, no standard estimation procedure gave satisfying results both in terms of accuracy and complexity.

However, notice that in Fig. 3 the eigenvalues are high for the first k eigenvalues and lower afterwards. Considering the eigenvalues as weights in the computation leads us to the so-called Green's function

$$G = \sum_{k=2}^{N_q} q_k \lambda_k q_k^T, \tag{5}$$

where N_g can more safely be set to a high value. An alternate formulation was proposed in [4]:

$$SPCA = D \sum_{k=2}^{N_q} q_k \lambda_k q_k^T D. \tag{6}$$

In the experiments reported in this paper, the Green's function outperformed significantly the two others in the case of unknown k, *i.e.* when N_g is set to the total number of eigenvectors. Since we are interested in a parameter-free system, only the results obtained using this function are reported. Fig. 5 illustrates the use of the Green's function while considering a dataset made of four bi-dimensional Gaussian clusters with significant overlap. Fig. 5(b) is obtained by a bi-dimensional scaling of the Green's function.

4. RESULTS

We split our results into two parts. The first part concerns accuracy improvements related to QBE systems expected to have already a good accuracy and the second part relates to what might happen to systems with worse raw accuracies before the post-processing stages applied in this paper.

4.1 High accuracy QBE systems

In this subsection we attempt to improve a QBE system with quite high raw accuracy. We exactly use the same methodology and input data as in [17].

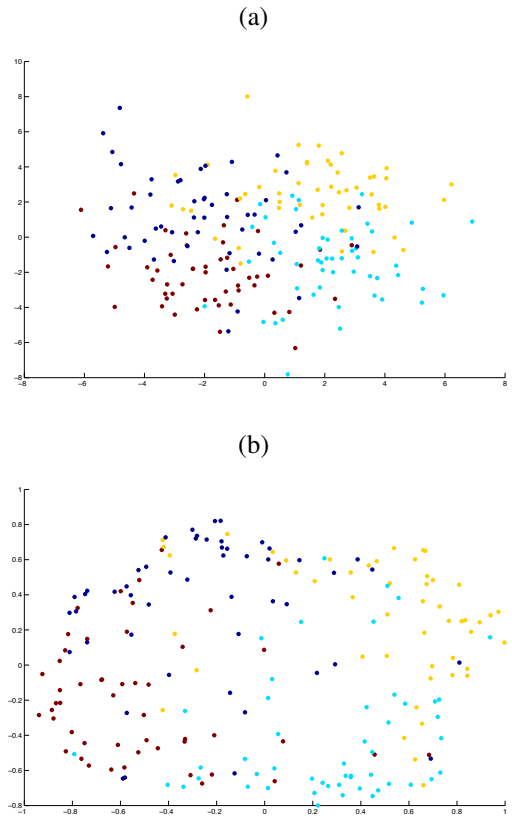


Figure 5. *Four bi-dimensional Gaussian clusters with significant overlap (a) and bi-dimensional scaling plot of the corresponding Green's function (b).*

4.1.1 Methodology

In order to replicate those experiments we use both the same synthetic and real data. Synthetic data is generated by considering a Gaussian noise font $\mathcal{N}(0, 0.25)$ with zero mean and 0.25 standard deviation. A dissimilarity measure between songs i and j is then defined as:

$$d_{i,j} = \begin{cases} 0 & \text{if } i = j, \\ |\mathcal{N}(0, 0.25)| & \text{if } i \text{ and } j \text{ are covers,} \\ 1 - |\mathcal{N}(0, 0.25)| & \text{otherwise.} \end{cases} \quad (7)$$

Real data is provided by the $Q_{\max}$ measure presented in [16] and sampled from the 2125 song collection of [17].

We also employ the same data setups as in [17] (Table 1, where var. means that cover sets have a variable cardinality). Different number of cover sets (n_C) and cardinalities are considered, as well as the fact of adding a different number of noise songs (n_N). For setups 1.1 to 2.4 we repeat the experiments 20 times.

To evaluate QBE systems we employ the mean of average precisions (MAP) over all queries. The MAP is routinely employed in a wide variety of tasks in the IR [13] and MIR communities, including the MIREX cover song identification task [5]. The average precision (AP) for a query i is calculated from the retrieved answer A_i as

Setup	Parameters			
	n_C	Card.	n_N	Trials
1.1	25	4	0	20
1.2	25	var.	0	20
1.3	25	4	100	20
1.4	25	var.	100	20
2.1	125	4	0	20
2.2	125	var.	0	20
2.3	125	4	400	20
2.4	125	var.	400	20
3	525	var.	0	1

Table 1. *Setup summary.*

$$\mathrm{AP}_i = \frac{1}{C_i} \sum_{r=1}^{N} \mathrm{P}_i(r)\mathrm{I}_i(r), \quad (8)$$

where C_i is the total number of covers for the i-th query, N is the total number of songs in the dataset, P_i is the precision of the sorted list A_i at rank r,

$$\mathrm{P}_i(r) = \frac{1}{r} \sum_{l=1}^{r} \mathrm{I}_i(l), \quad (9)$$

and I_i is a relevance function such that $\mathrm{I}_i(z) = 1$ if the song with rank z in A_i is a cover of the i-th song, $\mathrm{I}_i(z) = 0$ otherwise. A relative MAP increase is then computed just dividing the post-processed MAP by the raw one, subtracting 1, and multiplying by 100. For further details about methodology we resort to [17]. In the case of UPGMA and CC we report results with the optimal threshold found, independently for each data source.

4.1.2 Results

As it can be seen in Table 2, a significant accuracy improvement can be gained over the synthetic dataset. UPGMA performs best, followed by SCN which is handicapped by the cluster size variability (setups 2.2 and 2.4).

On the real dataset, UPGMA and CC perform equally well (Table 3). SCN achieves lower performance, probably due to the fact that real data has less intrinsic regularity

	UPGMA	CC	SCN
1.1	10.17	5.49	6.17
1.2	9.76	4.31	4.08
1.3	10.01	3.88	10.20
1.4	9.54	3.73	3.27
2.1	20.95	5.33	20.00
2.2	20.70	4.95	5.98
2.3	21.54	4.62	25.20
2.4	20.35	5.08	10.90

Table 2. *Accuracy improvement (expressed as relative MAP-improvement %) for the synthetic dataset processed using the QBE proposed in [17] as input.*

	UPGMA	CC	SCN
1.1	5.49	4.91	3.55
1.2	4.31	4.00	3.15
1.3	3.88	3.97	3.26
1.4	3.73	4.05	3.45
2.1	5.33	6.44	2.82
2.2	4.95	5.02	2.47
2.3	4.62	6.08	2.43
2.4	4.77	5.06	1.70
3	5.08	5.57	1.14

Table 3. *Accuracy improvement (expressed as relative MAP-improvement %) for the real dataset processed using the QBE proposed in [17] as input.*

than the synthetic one. Actually, no post-processing improves more than 5-6%. This may be explained by the fact that the MAP achieved by the considered system over this concrete dataset is rather high. As a consequence, setting a threshold distance can be done reliably (recall Fig. 2). Therefore, one can speculate that the best MAP that can be achieved given this configuration is in that range.

As a conclusion, it seems that approaches focusing on locality (UPGMA and CC) are more relevant than global approaches (SCN) for improving the performance of a QBE system with rather high raw accuracy provided that their clustering threshold can be set reliably.

4.2 Lower Accuracy QBE systems

In light of the previous results, we are interested in seeing how these clustering schemes perform on lower accuracy systems. Motivations for that could be that we either do not have a good, high performing QBE system for a given task, but a more modest one, or either that we are using a faster and more efficient version of the original system. Furthermore, we could be dealing with a particularly difficult dataset where our (otherwise reliable) QBE system performs more poorly.

In these cases, the accuracy improvement provided by the post-processing steps outlined in this paper could be more significant than with the original high accuracy system. It could even be the case that, with a (in principle) lower performing QBE system, we reached the same (or a higher) final MAP.

For lower accuracy systems it is theoretically relevant to consider more global approaches, as setting a dissimilarity threshold is more difficult due to the noise level. However, the overall structure of the dataset might not be completely lost, and therefore we can still take benefit of this fact by using a method like SCN. This can be asserted by comparing the MAP increase achieved by the studied methods when considering as input a lower accuracy system [7] (Table 4).

4.2.1 Methodology

We propose to further verify the previous assertion by simulating a QBE system with a controllable accuracy. For

	MAP	UPGMA	CC	SCN
Serrà et al. [16]	0.73	4.01	1.27	1.14
Ellis & Cotton [7]	0.42	8.06	3.04	19.70

Table 4. *MAP and MAP increase (%) for two QBE systems over the "covers80" dataset [7]. UPGMA and CC thresholds were specifically optimized for this dataset (however no significant difference was observed, c.f. Sec. 2).*

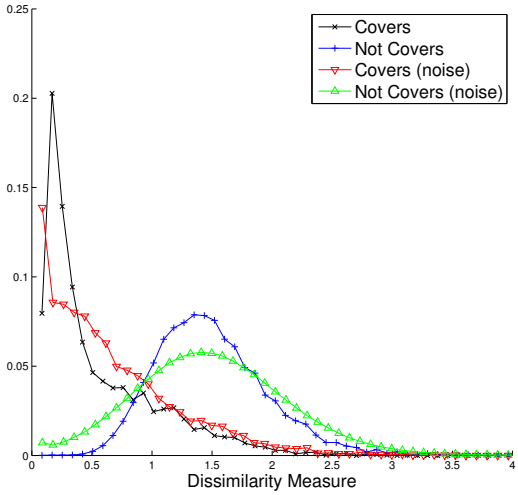


Figure 6. *Normalized histograms for real data with no noise (lines with crosses) and with $\sigma = 0.45$ (lines with triangles).*

that purpose, noise is added to our real data $d_{i,j}$ (the output of the high accuracy reference QBE system) such that

$$\tilde{d}_{i,j} = |d_{i,j} + \mathcal{N}(0, \sigma d_{\text{mx}})|\,, \qquad (10)$$

where σ is the noise level and d_{mx} is a normalization factor set to the maximal dissimilarity found (see Fig. 6 for the corresponding histograms).

4.2.2 Results

As it can be seen in Fig. 7, CC does not maintain its initial MAP increase when the noise level raises up. In contrast, UPGMA maintains or slightly increases its relative MAP. We finally see that SCN really boosts the MAP increase as more noise is added. This confirms our hypothesis and leads us to speculate that these methods are more robust for low accuracy QBE systems.

5. CONCLUSION

We proposed a global approach for improving the accuracy of query-by-example (QBE) systems based on spectral connectivity network. Contrasting with other state-of-the-art approaches, it does not rely on any parameter setting such as a dissimilarity threshold or the expected number of or cardinality of clusters within the data.

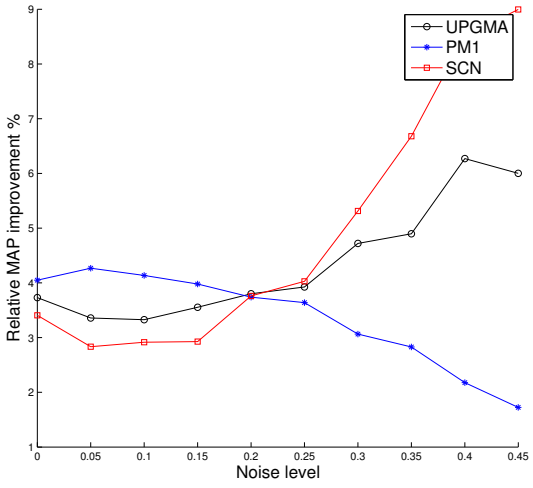


Figure 7. *Relative accuracy increase as a function of the noise level for setup 2.4 using Serra's data/QBE combination as input.*

The experiments showed that the proposed approach exhibits comparable results for improving high accuracy QBE systems and becomes highly competitive for improving lower accuracy QBE systems. Future research will include a more in depth study upon the selection of the relevant eigenvectors (a problem closely linked to the estimation of the number of clusters in a dataset).

6. ACKNOWLEDGEMENTS

M.L. has been partially funded by the Quaero project within the task 6.4: "Music Search by Similarity". J.S. has been partially funded by the Music 3.0 project TSI-070100-2008-318 of the Spanish Ministry of Industry, Tourism, and Trade.

7. REFERENCES

[1] F. Bach and M. I. Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7:1963–2001, 2006.

[2] A. S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. The MIT Press, 1990.

[3] M. Casey, R. C. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, April 2008.

[4] C. Ding, X. He, H. Zha, and H. Simon. Unsupervised learning: Self-aggregation in scaled principal component space. In *European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2002.

[5] J. S. Downie. The music information retrieval evaluation exchange (2005–2007): a window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[6] A. Egorov and G. Linetsky. Cover song identification with IF-F0 pitch class profiles. *MIREX extended abstract*, September 2008.

[7] D. P. W. Ellis and C. Cotton. The 2007 LabROSA cover song detection system. *MIREX extended abstract*, September 2007.

[8] R. Foucard, J.-L. Durrieu, M. Lagrange, and G. Richard. Multimodal similarity between musical streams for cover version detection. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP'10)*, Dallas, Texas, USA, March 2010.

[9] M. J. Greenacre. *Theory and Applications of Correspondence Analysis*. Academic Press, 1984.

[10] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[11] M. Lagrange, L. G. Martins, J. Murdoch, and G. Tzanetakis. Normalized cuts for predominant melodic source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):278–290, Feb. 2008.

[12] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.

[13] C. D. Manning, R. Prabhakar, and H. Schutze. *An introduction to Information Retrieval*. Cambridge University Press, 2008.

[14] M. Meila and J. Shi. Learning segmentation by random walks. In *Advance on Neural Information Processing Systems*, 2000.

[15] S. Ravuri and D. P. W. Ellis. Cover song detection: from high scores to general classification. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 55–68, March 2010.

[16] J. Serrà, X. Serra, and R. G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11:093017, September 2009.

[17] J. Serra, M. Zanin, C. Laurier, and M. Sordo. Unsupervised detection of cover song sets: Accuracy improvement and original identification. In *International Society for Music Information Retrieval Conference*, 2009.

[18] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[19] R. Xu and D. C. Wunsch. *Clustering*. IEEE Press, 2009.

[20] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Annual Conference on Neural Information Processing Systems*, 2004.

USERS' RELEVANCE CRITERIA IN MUSIC RETRIEVAL IN EVERYDAY LIFE: AN EXPLORATORY STUDY

Audrey Laplante
École de bibliothéconomie et des sciences de l'information
Université de Montréal, Montréal, QC, Canada
audrey.laplante@umontreal.ca

ABSTRACT

The paper presents the findings of a qualitative study on the way young adults make relevance inferences about music items when searching for music for recreational purposes. Data were collected through in-depth interviews and analyzed following the constant comparative method. Content analysis revealed that participants used four types of clues to make relevance inferences: bibliographic metadata (e.g., names of contributors, labels), relational metadata (e.g., genres, similar artists), associative metadata (e.g., cover arts), and recommendations/reviews. Relevance judgments were also found to be influenced by the external context (i.e., the functions music plays in one's life) and the internal context (i.e., individual tastes and beliefs, state of mind).

1. INTRODUCTION

In recognition of the need to provide researchers with an infrastructure for the evaluation of MIR systems and algorithms, Music Information Retrieval Evaluation eXchange (MIREX) was established in 2005. Contests have been held annually since then. Like Text Retrieval Conference (TREC) experiments from which it is inspired, MIREX uses precision-recall measures to evaluate system performance. These are used to measure "the probability of agreement between what the system retrieved or failed to retrieve as relevant (systems relevance) and what the user assessed as relevant (user relevance) where user relevance is the gold standard on the basis of which evaluations are made" [1]. Hence, to establish 'ground truth' for the evaluation of MIR tasks that called for human judgment (e.g., audio music similarity), user surrogates (as opposed to the real users who originated the search queries) were asked to judge *a posteriori* whether the results retrieved were relevant [2].

Although this approach has the advantage of taking into account the human dimension of the process, it also presents limitations. The validity of relevance judgments made in experimental setting is questionable since the criteria used by participants might not correspond to those used by people in real situations. Studies on user-defined relevance conducted in naturalistic settings show that, apart from content-based criteria, criteria pertaining to the user and the user's situation play a significant role in the evaluation of relevance [3, 4]. Therefore, failing to take into consideration the situation within which relevance judgments occur raises concerns and stresses the importance of studying how relevance judgments are made in real life. While research on relevance criteria used in textual information retrieval can have some utility for the MIR community, studies on video and image information retrieval suggest that differences in information type can result in differences in the criteria used to make relevance judgments [5, 6], hence the need to conduct research on user-based relevance in the context of MIR. Unfortunately, this area of research has hitherto remained essentially unexplored.

The present study was designed to bridge this gap by investigating how young adults make relevance inferences about music items when searching for music for recreational purposes. More specifically, it aims to address the following research questions: (1) What clues do young adults use to make relevance inferences about music items? (2) How do individual characteristics (e.g., knowledge, experience) influence their relevance judgments? (3) How does the context influence their relevance judgments? By providing a rich understanding of relevance judgments in context, this study will be beneficial in many ways. It will provide the MIR community with a better understanding of the behavior of current and potential MIR systems users, which may translate into improvements in MIR system design and evaluation measures.

2. RELATED RESEARCH

Since the 1990s, information scientists have conducted numerous empirical studies on user-based relevance. This has led to a redefinition of the concept of relevance and to an increased knowledge of the criteria used by people when making relevance judgments.

2.1 Concept of Relevance

Researchers distinguish system-oriented (or objective) relevance from user-oriented (or subjective) relevance [7]. According to the former, a document is considered relevant if it is topically related to the search query, a measure that has the useful property of being objective. From the user's point of view, however, topicality was found to be the most important but not necessarily the only relevance criterion. Therefore, a user-oriented definition of relevance was proposed where a document is considered relevant if the user who originated the query judges that it meets his/her information need. This conception of relevance implies that relevance judgments are interpreta-

tional: they can only be made by end-users and are closely tied to the context within which they occur.

User-oriented relevance can be studied from different standpoints: some researchers have focused on its cognitive aspects (*cognitive relevance*) [8]; some on its psychological aspects (*psychological relevance*) [9]; some on its dynamic nature (*dynamic relevance*) [10]; and others on its relation with the situation or task at hand (*situational relevance* or *utility*) [11]. Hence, in an attempt to encompass all these dimensions, Barry and Schamber [12] describe relevance in the following terms:

> "[…] relevance is (1) cognitive and subjective, depending on users' knowledge and perceptions; (2) situational, relating to users' information problems; (3) complex and multidimensional, influenced by many factors; (4) dynamic, constantly changing over time; and yet (5) systematic, observable and measurable at a single point in time."

It is with this multidimensional and situational perspective of relevance in mind that the present research project was designed.

2.2 Studies on User-Defined Relevance

In 1998, Barry and Schamber compare the findings of two studies on user-defined relevance and conclude that a high degree of overlap exists in the relevance criteria used by the participants in both studies. Subsequent studies have confirmed it since: there seems to exist a core set of criteria people use to make relevance judgments regardless of the context [13]. However, depending on the nature of the information, the situation or the user, the weight people attribute to each criterion varies and additional criteria may be employed. Of particular importance to MIR is the research on relevance criteria used when searching for non-textual documents. Choi and Rasmussen [6] found that in the context of image information retrieval, authority was less important than in textual information retrieval, whereas subjectivity and affectiveness—the emotional reaction to an image—played a significant role in the selection stage. Yang and Marchionini [5] found that users of video retrieval systems used textual criteria to start their search but mostly employed visual criteria (e.g., style, color, motion) in the final selection stage. In both cases, as in textual information retrieval, topicality was the most common and important criterion.

Also of interest for MIR is the research on relevance in non-problem-solving contexts, which correspond more closely to situations where people search for music for recreational purposes. Xu [14], who studied how users make relevance judgments when searching for information for its epistemic or entertainment value, found that novelty displaces topicality as the most commonly used relevance criterion.

3. RESEARCH DESIGN

The purpose of the present study was to provide a *rich description* of the way young adults make relevance judgments when seeking music for recreational purposes. Considering the complex and subjective nature of the phenomenon, a qualitative approach was considered best suited.

3.1 Data Collection

The subjective and interpretational nature of relevance called for a method that would allow us to gain insights into the internal behavior of participants (e.g., thoughts, feelings, intentions). In-depth interviewing was deemed the most appropriate method to attain this objective. The literature review on user-oriented relevance provided a useful theoretical background for the development of an interview guide. This guide enabled us to ensure consistency in the topics covered in the interviews while facilitating comparison between participants. During the interviews, participants were asked to talk about their preferred music information sources, to discuss the strengths and weaknesses of these sources, and to explain how, why and in which contexts they use them. Participants were also asked to relate in detail a recent music information-seeking experience.

3.2 Participants

Since music behavior is known to vary according to age and culture, we decided to reduce the heterogeneity of the population by limiting our study to the French-speaking young adults (18-29 years) of the Montreal metropolitan community. Participants (*n=15*) were selected following the maximum variation sampling strategy as described in [11]. Recruitment continued until the saturation point was reached, that is when the information obtained through interviews started to be redundant so that no new themes or patterns were emerging from the analysis.

Among the fifteen participants, ten were male. At the time of the interview, five were full-time students, seven were full-time workers, and three were unemployed. All had a high school diploma, 13 had a college diploma (or the equivalent), and ten had a university degree or were currently enrolled in a university program. None of them were professional musicians but six played at least one musical instrument. The group comprised a majority of avid music listeners, although the sample also included a few light or moderate music consumers.

3.3 Data Analysis

The interviews were recorded and transcribed. Each interview lasted between 38 and 62 minutes, for a total of 724 minutes of recording and over 120,000 words of transcriptions and notes. The software package NVivo by QSR International was used to facilitate the encoding and analysis process.

The data were analyzed inductively using the constant comparative method (CCM) as defined in [12]. CCM consists in a step-by-step method according to which the researchers (1) prepare the data for analysis by subdividing the transcripts into units of meaning (in this case into paragraphs); (2) read through the data to identify emerging themes and patterns in order to create a provisional set of categories; (3) categorize each unit of meaning into a category, forming new categories as needed; (4) refine the categories by comparing all units comprised into each category in order to identify the common properties or characteristics, merging, subdividing, or restating categories as needed; and (5) explore relationships and patterns across categories.

4. RESULTS

The interviews elicited a wealth of information about participants' music information-seeking behavior, including their likes and dislikes in terms of music information sources and the way they interact with these sources. The analysis revealed that the participants' relevance judgments were the results of a combination of different criteria and factors: criteria pertaining to the music itself (e.g., quality), the physical document (e.g., disk), the external context (e.g., intended use), the internal context (e.g., disposition), and their personal knowledge and experience. To determine the likelihood that a music item meets these criteria, participants used a variety of clues.

The results presented here cover the clues used to make relevance inferences and the criteria associated with the internal and external contexts of the search. Quotes were translated from French to English, while maintaining as much as possible the level of language used by the participants.

4.1 Relevance Clues

For all participants, listening to the music retrieved played a crucial role in determining its relevance. This, however, requires time and effort that, in some conditions, participants did not consider worthwhile. Moreover, the content of the music was not the only criterion. Therefore, at least in the first stage of their search, participants reported employing a variety of clues or extra-musical information to make inferences about the type of experience a music item offers, and assess the probability that it meets their desired criteria. As these clues are not always self-explanatory, previous knowledge and experience was often called upon to interpret them.

4.1.1 Bibliographic Metadata

Bibliographic metadata refer to the information used to describe an item, which, for music recordings, includes performers, composers, authors of lyrics, titles of songs/pieces and albums, labels, etc. This type of information appeared to be commonly used by participants in the selection process. The names of the main contributors (e.g., singers, bands) seemed to have the greater impact on their selection. A good experience with an artist could even transform some of them into committed fans having an almost unwavering faith in any new project to which this artist contributes. This explains why a few participants admitted buying CDs of their favorite artists without even listening to them beforehand. Conversely, disappointing experiences with an artist also increase the likelihood of discarding an item without listening to it. This speaks of the notion of authority, a relevance criterion also used in other contexts of information retrieval.

In the same way, for the most avid music listeners, trusted labels represented a guarantee of quality. When asked whether he would borrow an album by an unknown artist from the library, one participant explains: "if it's on a label, maybe. […] if it's a bluegrass band, for instance, because it's on Smithsoninan, it must be good." Another affirms that he "pretty much trust[s] what [Matador] release." Of course, metadata regarding contributors or labels are only useful if those are familiar. The fact that labels were only used by heavy music listeners suggests that extensive music knowledge is required to interpret this type of information.

Other bibliographic metadata proved to be of lesser importance for relevance judgments. Composers and authors of lyrics were explicitly mentioned by none of the participants, which might be due to the fact that many of the participants' favorite bands and singers composed their own songs. Album titles did not seem to affect selection either and only one participant affirmed relying on song titles to determine if an album was worth borrowing from the library.

4.1.2 Relational Metadata

Lee and Downie [15] define relational metadata as data regarding relationships between music items (e.g., music genres and similarity). Relationship between artists (e.g., collaborations, influences) could be added to this category. For several participants, links between artists allow them to situate an unknown artist in the music sphere, thus helping them assess the probability that the music of this artist corresponds to their taste. Hence, one participant related having discovered a group by reading an interview in a music magazine in which the group members were citing "their inspirations", which turned out to be "things [he] like[s]", which convinced him to listen to their music. This could also explain why *MySpace Music* and *allmusic*, two sources that incorporate a plethora of relational metadata, reached the top of the list as the most popular music information sources on the Web among participants. In *MySpace Music*, the "Friend Space" that connects artists and regular users was considered the source's most useful feature. Browsing the list of friends of a group one loves was a common strategy to discover new (but similar) groups as exemplified by this quote from a participant: "It's also a good way […] to find groups that make similar music and groups with which they do concerts… It's really, really useful!" Likewise, in *allmusic*, participants greatly appreciated the links to influencers, followers and similar artists in the articles, although the 'similar artists' links, which are created by music experts, were not unanimously considered reliable. Indeed, two participants complained about links leading to artists that "were not really similar" or "not similar at all."

While links between artists are primarily used to make inferences about individual objects, music genres, which are meant to bring similar or somewhat similar albums together, were mostly employed at an earlier stage, for instance to discard several items at once in order to get a manageable set of items to browse. Hence, one participant reported going directly to the "Film Music" section at the library, whereas another mentioned regularly searching for "rock" in online music stores. Even though participants widely used genres to search or browse music collections, they also frequently complained about them. Common criticisms included (1) some groups do not clearly fit into one genre ("It's oversimplified. You can't always categorize a group into something."); (2) genres are too broad ("[searching by genres on the Web] generates endless lists", or "[In music stores] they put everything that is rock, alternative, punk, metal together!); and (3) genres are too narrow ("[On *allmusic*] they have something like 600 styles of music, it's not concise

enough. […] They make up terms I've never seen and there are basically two groups that make this type of music, maybe even just one!"). As illustrated by this last quote, genres that are too specific also often sound unfamiliar and are therefore useless ("I could tell you there are 40-50% [of the genres on Limewire] I have no idea where it comes from or what it is"). The apparent contradiction between participants' complains regarding the level of specificity of genres can be explained by the source used to find music: genres in brick-and-mortar music stores are usually very broad, whereas specialized music sources on the Web tend to use very specific genres. But this does not explain entirely their dissatisfaction. Their knowledge of music or of a particular genre also influenced their perception of genres. While one participant admits knowing "seven genres, eight at most!" another explains that "there are five, six styles of punk."

4.1.3 Associative Metadata

Lee and Downie [15] define associative metadata as data about the relationship of a music item with events or other forms of art, which includes cover arts. While some appreciated cover arts for their intrinsic beauty, participants also appeared to attach importance to them for what they tell about the music. When little or no costs are involved, assessing the potential value of an album on the basis of its cover art could even represent a valid alternative to sampling the music: "At Cheap Thrills [a music store], they have a one-dollar rack. There are groups I don't know. The only thing I can look at, really, is the cover art. If it looks cool… And it often happens to fit. I find music I like in jackets I like!" Most of the time, however, participants employed cover arts only to make a first selection among albums, the following step being to listen to the music. Hence, when asked how she selects CDs in music stores, one participant describes that she looks at CDs "one by one" and selects those who have a nice cover art because "it says a lot" and "it has to represent something." She will then wait to be at home to download the album and see if it is worth buying it. Another participant, whose previous experiences had led him to conclude that cardboard jewel cases often contained music he liked, had come to use that criterion to make a first selection, a method that seemed to pay off ("Often, it happens to be albums I like!"). For two participants, however, cover arts had no influence on their selection. One participant explains that "[shopping for] music is not really visual" and cannot be done "simply by wandering around, looking at album covers." Another mentions that her previous experiences have convinced her that cover arts should not be considered as indicators of quality ("I've bought so many good albums in hideous jackets in my life!").

4.1.4 Recommendations and Reviews

A majority of participants affirmed that they attached importance to recommendations, reviews and ratings. The trustworthiness of the source was determinant in the value they ascribed to the information. Hence, recommendations from friends or colleagues perceived has having discriminating judgment and tastes that are similar to theirs had the greatest influence on their relevance judgments. A few participants also relied occasionally on record store staff. In these cases, as they did not know them personally, the decision to trust a person was based on (1) his/her general look: the person has to look like someone who "pretty much listens to what I listen"); and/or (2) where that person works: people working in small and specialized music stores ("a small, underground CD store") tended to be perceived as especially trustworthy. In contrast, because of past disagreements with critics' reviews, a majority of participants affirmed not being influenced by them.

Recommendations and reviews provided information that could be useful at different stages of a search. Participants sought recommendations to begin a search, to obtain information that has been filtered specifically for them ("It's like a filter [or] a bit like a shortcut"). It saves them time, while increasing their chances of finding something interesting ("You're more likely to come across something good right away"). This information can also be useful to make relevance inferences *en route*. Hence, they were more likely to pick up an album if they had heard of the artist before. Star ratings or popularity sorting, on the contrary, were used at a later stage, to identify albums or songs "that best represent the career of an artist." Reviews or recommendations could even change their initial relevance judgment. Indeed, one participant admitted having changed her mind about some music artists because of friends who "had found arguments" that had allowed her to listen to the music "from a new angle."

To conclude this section on the clues participants used to make relevance inferences about music items, we shall mention the types of information that had little or no influence. Interviews or biographies were mentioned only by a two, who were mostly interested in the professional life of the artists and in the relational metadata (influences, collaborators, etc.) they find in them. Related to that, participants seemed to favor information that could be scanned quickly: ratings or editor's picks, for instance, were far more popular than long, written reviews. Also absent from the picture were the lyrics (or the topic of the lyrics). Although the topic of a song or an album could sometimes be determinant in deciding what one would listen to in a specific situation, the lyrics appeared to be of little importance in determining the relevance of an item during the search process. This might be explained by the fact that although the participants spoke French, a majority mostly listened to Anglophone music.

4.2 Context

Saracevic [16] maintains that relevance "cannot be considered without a context" and defines the context as the result of a "dynamic interaction between a number of external and internal aspects." In line with this definition, our analysis revealed that the context in which a search occurs affected the relevance judgments of the participants in various ways.

4.2.1 Situation

We usually define situational relevance as the relationship between an information object and the user's information problem or task. This definition, however, did not seem entirely appropriate for this study since, according to our participants' accounts, searching for music for recrea-

tional purposes is rarely a task-oriented activity. But this does not mean that the situation had no influence on their relevance judgments. In reality, the roles music plays in their lives affect their information-seeking behavior. Participants used music for a variety of functions. While some reported listening to the same type of music regardless of the context, most affirmed selecting different genres of music in different situations. Thus, when seeking music, they always bore in mind the potential functions the music encountered could potentially fulfill.

It is easy to see how this can affect relevance judgments. Music, for instance, was the soundtrack of mental or artistic work for 11 participants. The reasons for listening to music while working were various: music can (1) contribute to inducing concentration ("when I forget my headphones, I'm kind of not productive"); (2) make the work more pleasant, especially a tedious or repetitive task ("it helps me get through the day"); or (3) provide inspiration for artistic activities ("[it] helps me get into a mood"). When used in this context, the primary selection criterion was that music did not interfere with their thinking, which usually meant instrumental and/or repetitive music such as classical, techno, or electronic; music genres they would not necessarily listen to otherwise. Hence, for one participant who usually listened to old French music, the music that played in the background when she was working was totally different: "If I'm at work and need a lot of concentration, I will put on techno music with no lyrics."

When music is used to maintain or establish interpersonal relationships, the selection is once again affected. As a matter of fact, some participants mentioned listening to different genres of music with different persons so that it would please everyone, maybe even music they wouldn't have listened alone, as illustrated by this quote: "My mother hates Pink Floyd. […] So when I'm with my father, we listen to Pink Floyd, but when I'm with my mother, I'll listen to something else. I can even listen to some Luce Dufault."

Music is also used to manage one's mood. Whereas some participants used music to modulate or enhance their mood ("If I get up on the wrong side of the bed, I put on music that will make me happy for the rest of the day"); others sought music that matched their current —usually depressed— mood ("If you're broken-hearted [… and you listen to] *Rose* by Portishead, you clearly know that they feel like shit, just like you.").

4.2.2 Individual Tastes and Beliefs

Not surprisingly, one of the main criteria employed to make relevance judgments was affectiveness or the emotional response to music. In other words, music usually has to meet one's taste to be considered relevant. Indeed, although participants reported occasionally selecting music outside of their regular tastes to fulfill specific functions, they still wanted this music to be as good as it could be. This explains why participants believed listening to the music was an essential step in formulating relevance judgments unless, as mentioned before, they considered that the risk incurred was low (e.g., highly trusted artist, music is free or almost free). For that purpose, participants frequently visited the *MySpace* profiles of artists or downloaded music illegally to be able to listen to entire songs or albums ("it allows you to really see what the song is like, the melody, see if you like it or not"). Indeed, although most online music stores or other music information sources propose 30-second excerpts, this was considered insufficient to make inferences about the work of an artist.

As seen in Section 4.2.1, music serves different functions, one of which being to help people define their identity, an area of research that has been widely studied by sociologists and psychologists. Through their music tastes, people express who they are—their attitudes, values, and opinions. Of course, people also use music preferences to make inferences about others. This has repercussions on their information-seeking behavior: people want the music they retrieve to correspond to their values and beliefs. Such behavior was common among participants. One participant said that he liked music that "has meaning" and "would feel guilty if [he] liked the music of someone [he] hated." Another admitted attributing a lot of importance to finding underground groups "because I tend to want to be unique." In fact, many participants showed a strong penchant for non-commercial music, which could be explained by their desire to distinguish themselves from others by having unique music taste.

Related to that, the geographical provenance of music artists seemed to influence the perception of a few since five participants showed a marked preference for local artists (one confessed that she was more "opened" to Quebec groups, while another explained that she really liked following "what's happening on the Quebec scene").

4.2.3 State of Mind

One's state of mind also affects music perception. As a matter of fact, nine participants admitted that discovering new music artists or genres required mental effort and an openness of mind they only had in certain contexts. Four said they could only appreciate unfamiliar music if they had "time to waste" so that they could settle down and concentrate on the music. Three affirmed they needed to be receptive to novelty ("I need to be in a different state of mind. […] You really have to say 'Ok, I need to adapt'"). Moreover, since music is often used for mood management (as seen in Section 4.2.1), one's current mood also influenced music selection.

5. CONCLUSION

This study sheds light on some of the particularities of relevance judgment in the context of MIR, more specifically in situations where people seek music information for recreational purposes. Although we found that a significant set of criteria used in textual information retrieval were also applicable in this context (e.g., quality, authority, familiarity, situation, user's knowledge and experience), some unique characteristics also emerged. Findings suggest that criteria pertaining to the user, especially individual tastes and beliefs, have a greater impact on selection than in other contexts. This could be due to the subjective nature of music perception and to the fact that mu-

sic tastes often act as a 'social badge' that conveys information about people. Moreover, while topicality was found to be the most common and important relevance criterion in textual, image and video information retrieval, our study revealed that it was not used by our participants, possibly because they attached little importance to lyrics. Therefore, genre, which has the property of allowing one to obtain rapidly a manageable set of items, displaced topicality as the most commonly used criterion to start a search. Our analysis also uncovered the importance of recommendations and reviews from trustworthy sources at different stages of the music selection process. On the other hand, this study reinforces findings from previous studies by confirming the importance of affectiveness as a criterion for making relevance judgments about non-textual information [5]; and the importance of novelty in recreational contexts [14].

This study provides indications for the design of MIR systems and interfaces that support users in their relevance judgments. It reiterates the need to provide rich metadata, including links between artists, not only to facilitate the search but also to better assist users in their selection. However, the fact that some metadata were useless to people who did not have the required knowledge to interpret them suggest that systems should also assist the user in this task, for instance by providing descriptions for music genres or music labels. The importance ascribed to recommendations from people one knows and trusts indicates that systems should include social networking tools that facilitate the sharing of information between users. The study also revealed that music preferences could change depending on the context (e.g., one's current mood, functions music plays in one's life). A successful recommender system should therefore be able to handle this complexity and allow people to have multiple 'music personalities,' thus recognizing the dynamic nature of relevance.

6. ACKNOWLEDGEMENTS

This work has been supported by the Social Sciences and Humanities Research Council of Canada. I am grateful to Dr. John Leide and Dr. J. Stephen Downie for their contribution to this project.

7. REFERENCES

[1] T. Saracevic, "Effects of inconsistent relevance judgments on information retrieval test results: A historical perspective," *Library Trends,* vol. 56, pp. 763-783, Spring 2008.

[2] M. C. Jones*, et al.*, "Human similarity judgments: implications for the design of formal evaluations," in *Proceedings of the 8th International Conference on Music Information Retrieval, Vienna, Austria, September 23-27*, S. Dixon*, et al.*, Eds., ed Vienna: Österreichische Computer Gesellschaft, 2007, pp. 539-542.

[3] L. Schamber, "Relevance and information behavior," *Annual Review of Information Science and Technology (ARIST),* vol. 29, pp. 3-48, 1994.

[4] C. L. Barry, "User-defined relevance criteria: An exploratory study," *Journal of the American Society for Information Science,* vol. 45, pp. 149-159, 1994.

[5] M. Yang and G. Marchionini, "Exploring users' video relevance criteria: a pilot study " in *ASIST 2004, Proceedings of the 67th ASIST Annual Meeting*, L. Schamber and C. L. Barry, Eds., ed Medford, NJ: Information Today, 2004, pp. 229-238.

[6] Y. Choi and E. Rasmussen, "Users' relevance criteria in image retrieval in American history," *Information processing and management,* vol. 38, pp. 695-726, 2002.

[7] D. Swanson, "Subjective versus objective relevance in bibliographic retrieval systems," *Library Quarterly,* vol. 56, pp. 389-398, 1986.

[8] P. Wang and D. Soergel, "A cognitive model of document use during a research project. Study I. Document selection," *Journal of the American Society for Information Science,* vol. 49, pp. 115-133, 1998.

[9] S. P. Harter, "Psychological relevance and information science," *Journal of the American Society for Information Science,* vol. 43, pp. 602-615, 1992.

[10] L. Schamber*, et al.*, "A re-examination of relevance: toward a dynamic, situational definition," *Information Processing & Management,* vol. 26, pp. 755-776, 1990.

[11] P. Wilson, "Situational Relevance," *Information storage and retrieval,* vol. 9, pp. 457-471, 1973.

[12] C. L. Barry and L. Schamber, "Users' criteria for relevance evaluation: A cross-situational comparison," *Information Processing & Management,* vol. 34, pp. 219-236, 1998.

[13] T. Saracevic, "Relevance: A review of the literature and a framework for thinking on the notion in information science. Part III: behavior and effects of relevance," *Journal of the American Society for Information Science and Technology,* vol. 58, pp. 2126-2144, 2007.

[14] Y. Xu, "Relevance judgment in epistemic and hedonic information searches," *Journal of the American Society for Information Science and Technology,* vol. 58, pp. 179-189, 2007.

[15] J. H. Lee and J. S. Downie, "Survey of music information needs, uses, and seeking behaviours: Preliminary findings," in *5th International Conference on Music Information Retrieval*, Barcelona, Spain, 2004, pp. 441-446.

[16] T. Saracevic, "Relevance: A review of the literature and a framework for thinking on the notion in information sicence. Part II: nature and manifestations of relevance," *Journal of the American Society for Information Science and Technology,* vol. 58, pp. 1915-1933, 2007.

USING JWEBMINER 2.0 TO IMPROVE MUSIC CLASSIFICATION PERFORMANCE BY COMBINING DIFFERENT TYPES OF FEATURES MINED FROM THE WEB

Gabriel Vigliensoni
CIRMMT
McGill University
gabriel@music.mcgill.ca

Cory McKay
CIRMMT
McGill University
cory.mckay@mail.mcgill.ca

Ichiro Fujinaga
CIRMMT
McGill University
ich@music.mcgill.ca

ABSTRACT

This paper presents the jWebMiner 2.0 cultural feature extraction software and describes the results of several musical genre classification experiments performed with it. jWebMiner 2.0 is an easy-to-use and open-source tool that allows users to mine the Internet in order to extract features based on both Last.fm social tags and general web search string co-occurrences extracted using the Yahoo! API. The experiments performed found that the features based on social tags were more effective at classifying music into a small (5-genre) genre ontology, but the features based on general web co-occurrences were more effective at classifying a moderate (10-genre) ontology. It was also found that combining the two types of features resulted in improved performance overall.

1. INTRODUCTION

The field of music information retrieval (MIR) has benefited greatly from the explosion of information that is available on the Internet. The musical information that can be mined from the web is extremely rich in both depth and breadth, and the on-line contributions of both musical experts and general listeners has provided music researchers with a rich resource of cultural information. This information can be accessed not only via traditional web mining approaches like web scraping and crawling, but also via powerful APIs provided by a variety of on-line organizations, such as Last.fm [19] and Yahoo! [20]. This information is of particular value to researchers in automatic music classification, as they can harvest it in the form of numerical features that can then be processed by machine learning algorithms in order to automatically label music with categories associated with domains of interest such as genre, mood or listening scenario.

This paper has two main foci. The first is an investigation of the relative utility of features mined from the web in general and features mined from listener tags, in this case from Yahoo! and Last.fm, respectively. This investigation involves genre classification experiments based on features derived using the APIs provided by these two organizations. The classification effectiveness of each of these two groups of features is analysed both individually and in combination. Results derived from several different feature extraction configurations are also studied, as different configurations can have an important impact on results, but this area has not been methodically investigated to date in the MIR literature.

The emphasis on genre classification in this inquiry is due to the fact that it can be a particularly difficult type of classification that examines the effectiveness of various classification approaches. Although genre classification can certainly have value in and of itself [11], the ultimate goal of this research is to evaluate approaches that can, hopefully, be effectively extended to other types of music classification as well.

The second primary focus of this paper is the presentation of jWebMiner 2.0, a feature extraction tool for mining data from the Internet. This tool has been expanded since the publication of the original jWebMiner 1.0 [12], and the updated version is presented here to the MIR community for their research use. jWebMiner 2.0 was used to perform all of the experiments described in this paper.

2. BACKGROUND INFORMATION

2.1 Mining the Internet for Musical Features

There has been too much research on mining useful information from the Internet to cite with any completeness here. It is, however, valuable to emphasize certain particularly influential papers, namely [2, 5–10, 16–18].

2.2 Social Tags

As noted in by McKay and Fujinaga [11], genre (and other types of musical categories) can be strongly characterized by how an audience understands and perceives music and musicians, not just on objective content-based characteristics. This has important implications for genre classification. “True” class labels are essentially specified by the opinion of millions of listeners and evolve over time. These labels are influenced by many cultural factors, some of which may

be independent of purely sonic musical characteristics. The labels that one might assign to a song are based not only on the song itself, but one's overall perception of an artist. Furthermore, it might be said that there is no one ground truth, as is the "truth" is indeed the sum of the opinions of all listeners.

"Social tags" are unstructured text labels assigned by non-expert users to an entity, such as a song, an album or the collected work of an artist. A variety of on-line services, such as Last.fm, permit users to aggregate their tags for a variety of musical resources. There are typically no restrictions on the choice of words or phrases that users can tag resources with, although users do in practice often seem to select tags that other users have already used, thus creating a kind of shared and navigable system [10]. Tagged resources can therefore be said to, in some way, share certain characteristics with other resources that have been tagged with the same category or categories in the perception of users [1]. The value of social tags increases when they are aggregated into such a large public access community repository, as they provide access to information on how all the users of the system perceives and organize the resources [8]. In the context of musical communities and libraries, social tags are used by people for playlist organization; personal song retrieval; the expression of taste and opinion; and general contribution to public knowledge [10].

It is clear that social tags are a source of valuable human-generated contextual knowledge about music. They provide researchers with information about mood, emotion, genre and other types of categories that are based on the subjective perceptions of millions of users [3]. Mining this data from the web can thus be effective in acquiring information that can be used in the evaluation and training of MIR systems, and at least in the short term, as the aggregation of cultural perceptions that are in constant flux as culture and individual opinions change [12].

2.3 Last.fm

Last.fm is a music service that has been in operation since 2003. Internet radio is the core service that it offers to its users, but it also provides them with a broad range of additional functionality. For example, Last.fm allows users to create personal profiles and contribute social tags to songs, albums and artists. Of particular interest, Last.fm automatically generates custom radio playlists using recommendation algorithms based primarily on collaborative filtering. These algorithms consider both user tags and listening behaviour, as mined by Last.fm's "Scrobbler" software, which monitors the music played by listeners both on the Last.fm site and on their enabled local media players.

The amount of information managed by Last.fm is enormous, consisting of more than 39 billion tracks scrobbled to date, a number that is increasing at a rate of over 400 million tracks per week.[1] The company provides free access to portions of its data through an API [19], something that permits developers to build their own tools.

2.4 jWebMiner and jMIR

jWebMiner [12] is part of the jMIR automatic music classification research suite [14]. In addition to jWebMiner, jMIR also includes tools for extracting features from audio files and MIDI files; a machine learning engine based on metalearning; datasets to serve as ground truth for training and testing classification models; and software for profiling and detecting metadata errors in music collections. jMIR is designed specifically to facilitate the integration of information extracted from different types of musical data, and jWebMiner is the component that provides access to social context features (i.e., cultural musical information) available on-line.

jWebMiner, like all of the jMIR software components, is designed to be usable by users with both technical and non-technical backgrounds, and as such includes an easy-to-use and flexible graphical interface. All of the jMIR components, including jWebMiner, are open-source and available for free at http://jmir.sourceforge.net.

At its most basic level, the original jWebMiner 1.0 operates by accessing Yahoo's web search API to acquire hit counts for various search strings. For example, calculations measuring how often the names of different musicians or composers co-occur on the same web pages (taking into account how often they occur individually) can provide insights on the relative similarity of the musicians to each other. Similarly, the cross tabulation of song or artist names with musical class labels associated with genre, or mood, for example, can be used to classify music. Of course, basic hit counts can result in noisy data, so it is necessary to include additional processing to improve results.

jWebMiner begins by parsing either iTunes XML, ACE XML, Weka ARFF or text files in order to acquire strings to use in searches. Users may also manually enter search strings in the GUI. The software then accesses Yahoo's API to either measure the co-occurrence of each value in one field with other values in the same field, or to measure the cross tabulation of values in different fields.

The optimal statistical procedure for processing hit counts and dealing with noise contained in them can vary depending on the task at hand. One must consider not only the accuracy of an approach, but also its search complexity, as web services typically involve daily limits on queries. jWebMiner therefore allows users to choose between a variety of metrics and scoring systems.

One option offered by jWebMiner is the ability to allow users to specify "string synonyms" so that hit counts will be combined for linked synonyms. This would be useful, for example, in a genre classification task where the class names "R&B" and "RnB" are equivalent.

jWebMiner also allows user-definable "filter strings." This permits the software to be set to ignore all web pages that do not contain general filter terms such as "music," for example, or application-specific terms such

[1] http://www.last.fm/community

as "genre" or "mood." This can be useful in avoiding irrelevant and noisy hit counts. For instance, a feature extraction should not count co-occurrences of "The Doors" with "Metal" or "Rap" unless they refer to music rather than unrelated topics such as the building industry or door knockers.

It is also possible to set jWebMiner to limit searches to particular sites, such as the All Music Guide or Pitchfork, in order to emphasize musically relevant and reliable sites. jWebMiner also allows users to assign varying weights to particular sites as well as to the web as a whole when feature values are calculated.

The feature values generated by jWebMiner essentially consist of relative similarities measured between various specified search strings, after appropriate statistical processing. These feature values can be exported to ACE XML, Weka ARFF, delimited text or HTML files. Feature values may also be browsed directly via the GUI.

3. MINING LAST.FM WITH JWEBMINER 2.0

The most significant improvement incorporated into the new jWebMiner 2.0, in addition to the existing functionality described in Section 2.4, is the ability to extract social tag-based information using the Last.fm API [19]. For example, users can specify artist names and have the software extract the most common tags for that artist from the Last.fm API, ranked by popularity. The user can also specify class labels of interest, and have jWebMiner derive features based on whether each artist has been tagged by Last.fm with any of these labels and, if so, have the feature value reflect the tag's relative Last.fm ranking.

jWebMiner 2.0 can extract just the Last.fm-derived features, just the Yahoo!-derived features or both. If the latter option is selected, then jWebMiner will not only extract each of the two feature sets individually, but will also provide the user with levels of support associated with each class label based on the normalized combination of the Last.fm-derived and Yahoo!-derived features. This normalization process is performed to level all queries to the same number-base in the case of the Yahoo!-mined features, and to represent the position of a given artist's Last.fm tag on a normalized scale. For an artist and genre we define a scoring function $S(a,g)$, where $P(a,g)$ is the Last.fm position of the queried tag and $P(a,i)$ the position for all genre tags:

$$S(a,g) = \frac{1}{P(a,g)\sum_{i=1}^{n}\frac{1}{P(a,i)}} \quad (1)$$

To exemplify the normalization process let us query the german band *Tarwater* with the tags *indie, post-rock* and *electronic*. These tags appear respectively in the position 6, 7 and 1 of the top tags for that artist. Being n equals 3, the value of the sum is 1/6 + 1/7 + 1/1, which is 55/42. Thus, the scoring function values are 7/55 for *indie*, 6/55 for *post-rock,* and 42/55 for *electronica.*

jWebMiner automatically bases its score on only the Yahoo!-derived features if a particular artist is not on Last.fm, or if an artist has not been tagged with any of the queried class names. In addition, jWebMiner can show the web search normalized feature score, the Last.fm normalized ranking score, and the averaged results. These values can be processed independently afterwards.

It is hoped that the combination of social tag-based feature extraction with more general web search-based feature extraction will provide MIR researchers with a unified and accessible cultural feature extraction tool that provides access to two different kinds of valuable cultural musical information available on-line. Although jWebMiner 2.0 can certainly be used alone, it also carries the significant advantage of allowing features extracted with it to be easily processed using jMIR's ACE machine learning tool, or combined with features extracted from audio or MIDI by, respectively, jMIR's jAudio and jSymbolic feature extractors [13].

4. EXPERIMENTS

4.1 Overview

A series of experiments were performed in order to investigate the relative performance of features derived from Last.fm social tags, features derived from Yahoo! web searches and the combination of features derived from both sources. Attention was also given to various possible web search feature extraction configurations, involving the use of various different filter words and site weightings (see Section 2.4).

The feature groups and extraction configurations were evaluated based on their performance in genre classification. As noted in Section 1, genre classification was chosen because it can be a particularly difficult task, and is thus a good stress test for features.

All experiments were performed using jWebMiner 2.0, which harvested features using the Last.fm and Yahoo! web services, as described above.

4.2 Dataset used

The experiments were conducted using the SAC (Symbolic, Audio and Cultural) dataset [13]. This dataset consists of 250 matching MIDI files and audio recordings, as well as accompanying metadata (e.g., title, artist, etc.). This metadata was stored in an iTunes XML file, which was parsed by jWebMiner in order to extract cultural features from the web [13].

The files of the SAC dataset are divided into 10 different genres with equal numbers of artists per genre (*Modern Blues, Traditional Blues, Baroque, Romantic, Bebop, Swing, Hardcore Rap, Pop Rap, Alternative Rock* and *Metal*). It is clear upon observation that these 10 genres consist of 5 pairs of similar genres. This arrangement makes it possible to perform 5-class genre classification experiments as well as 10-class experiments on the same dataset simply by combining each pair of related genres into one class. An additional advantage is that it becomes possible to measure an indication of how serious misclassification errors are in 10-class

experiments by examining how many misclassifications are in an instance's partner genre rather than one of the other 8 genres. The ground truth was created using expert sources, such as AllMusic.com, combined with the personal expertise of the authors of the dataset.

SAC was chosen partly because it provides two tiers of genre classification; partly because the similarity of each of the 5 genre pairs makes 10-class classification particularly difficult, and thus a good test of jWebMiner's effectiveness; and partly because it can be used in other research to investigate the utility of combing the cultural features extracted by jWebMiner with other kinds of features, such as features extracted from audio, symbolic and lyrical data. This latter application was previously investigated with jWebMiner 1.0 in [13], and an update to this research using jWebMiner 2.0 is presented in [15] .

4.3 Text filtering and site weighting

In order to optimize classification accuracy using jWebMiner's filtering capabilities, we designed and tested several sample filters for the *Required Filter Words* and *Excluded Filter Words* fields. Sources such as [2], [4], [7], and [17] have recommended certain required filter words, such as *music*, *review*, *like*, *work* and *artist* x *played* y *music*, and have also recommended certain excluded filter words, such as *mp3*, *download*, *videos*, *cart*, *prices* and *login*. In general, our results matched those obtained in [7], which is to say that better performance was achieved when only simple filters were used. Thus, only *mp3* and *store* were used as excluded filter words, and no required filter words were used. It was found through informal experimentation that too much noise was otherwise introduced by web pages in which many of these terms co-occur with in non-specific ways with many other artists and genres.

We also developed a set of synonyms for different genres and artist names in order to take into account the variety that one finds in practice. So, for example, we used the terms *Bebop* and *Be-bop* as synonyms for *Bop*. On the other hand, an artist such as *Derek and the Dominos* could be found as *Derek and the Dominoes.*

We also tested different site weighting schemes. To begin with, we tried simply querying the whole network (NC, as described in Table 1) using the 5-genre taxonomy. We then queried the web as a whole as well as three predetermined websites (*wikipedia.org*, *allmusic.com* and *amazon.com*), using weight values of 0.5 for the whole network and 0.166 for each one of the sites (*W1*). For 10-genre classification, we tried a third arrangement that did not take into account the whole network, and where each of the above three sites was assigned a weight of 0.333 (*W2*). Experiments with these these different arrangements were performed in order to gain insights into how data mined from the web as a whole performed relative to specialized websites.

MK08	Previous experiment with jWebMiner performed in [13].
NC	No constraints involving weighting, required filter words or excluded filter words.
F/W1	*MP3* and *store* as excluded filter words. Site weightings of 1/6 for *wikipedia.org, allmusic.com, and amazon.com; the whole web weighted by 1/2.*
F/W1/S	Same settings as F/W1, plus a set of synonyms for the genres and artist names.
ST	Classification results when using only social tags.
C NC	Combined and averaged results using web search with no constraints, as well as social tags.
C F/W1	Combined and averaged results using social tags and web search in F/W1 configuration.
C F/W2	Combined and averaged results of social tags and web search, *MP3* and *store* as excluded filter words, and site weightings of 1/3 for *wikipedia.org, allmusic.com, and amazon.com.*

Table 1. The different configurations tested on the SAC dataset. Some experiments involved 5-genre classification, while others involved 10-genre classification.

4.4 Results

4.4.1 SAC dataset 5-genre classification

Table 1 provides brief descriptions of each experimental setup, as well as specifications of the identifying notation used. The average classification accuracy rates for experiments involving only web search-based 5-genre classification are shown in Table 2.

MK08	NC	F/W1	F/W1/S	ST	C F/W1
87.2	82.4	90.1	93.1	95.4	96.9

Table 2. Average classification accuracy rates for 5-genre classification experiments on the SAC dataset.

It can be seen that the *NC* experiment classification accuracy was worse than *MK08*, which is not surprising because the *MK08* experiments used some filtering constraints, namely the use of "music" as a required filter word. However, with the *F/W1* and *F/W1/S* configurations we observed improvements of 2.9% and 5.2% over *MK08*, and 7.7% and 10.7% over *NC,* respectively. These results suggest that highlighting particular music-related sites can provide better results than simply extracting information from the web as a whole.

On the other hand, retrieving social tags alone resulted in improvements in genre classification performance of

8.2% over the *MK08* mark, and the combined approach of retrieving social tags as well as querying the web resulted in the highest classification rate of 96.9%. Hence, it appears that combining the features from web searches and social tags can increase accuracy and diminish the problems associated with each method, such as noisy web search hits (as reviewed in [13] and [6]) and the *cold start*, *polysemy*, *annotation accuracy*, *popularity bias*, and *malicious behaviour* issues associated with social tags (as reviewed in [8] and [14]).

4.4.2 SAC dataset 10-subgenre classification

For 10-genre classification, we performed the same experiments but separated synonyms, filter words and site weights in order to gain insights on how each one of them affect results. The same SAC genre-pair weighting scheme used in [13] and described in Section 4.2 was also used to evaluate the seriousness of those classification errors that did occur Specifically, if a misclassification is within a genre pair, the error is reduced to 0.5 of an error, and if the misclassification is outside of a pair, then the error is increased to 1.5. Table 3 shows the weighted (W) and unweighted (*UW*) Yahoo! web search-only accuracy rates that we found.

	MK08	NC	F/W1/S	F/W1	F/W2	ST
UW	61.2	56.5	50.4	67.2	77.9	43.5
W	67.4	63.7	51.9	76.7	82.8	43.9

Table 3. Classification accuracy rates in 10-genre classification experiments, including both unweighted (*UW*) and weighted (*W*) results.

As in the 5-genre experiments, *NC* performed worse than *MK08*. On the other hand, *F/W1* and especially *F/W2* were more accurate than *MK08* by 6.0% and 16.7% respectively in terms of unweighted classification accuracy, and by 9.3% and 15.4% respectively in terms of weighted classification accuracy. Also, in the 5-genre classification experiments social tags gave excellent results. In contrast, in the 10-genre classification experiments using social tags (*ST*) social tags actually performed the worst amongst all configurations.

To understand this phenomenon, we studied the tags that Last.fm users use and found that, in general, they are very well defined for broad genres, such as those used in the 5-genre experiment. However, when one delves into finer classification, there are many subtle differences in the ways that different people perceive genres. For example, how substantially different is the genre *Rock* from *Punk* if we think of a band such as *Green Day*? In addition, a number of problems related to tagging such as polysemy, synonymy, accuracy, and spam are present in large collections of social tags [8]. Furthermore, tags do not represent only genres or styles, but anything that individual users want, from *mood* to *BPM*, so they can be very noisy if one wishes to extract detailed class labels.

To overcome these problems, we tried combining the features extracted from both Last.fm social tags and Yahoo! web searches. The obtained results are shown in Table 4.

	C NC	C F/W1	C F/W2
UW	75.6	74.8	77.9
W	78.6	79.8	81.3

Table 4. Classification accuracy rates for 10-genre classification experiments using combined data from both web search and social tags, unweighted and weighted.

It can be seen that for the *C NC* experiment, the combined data results in improvements of 19.1% (*UW*) and 14.9% (*W*) relative to using only web searches without any constraints. For the *C F/W1* experiment, the results were 7.6% (*UW*) and 3.1% (*W)* better. The genre classification accuracy was thus improved by using the combined approach, despite the fact that social tags alone performed very poorly. However, for the last experiment *C F/W2*, the results were the same for the *W* scale (as *F/W2)*, and slightly worse in the case of the *UW* scale. These last results make some sense because the *F/W2* experiment weights only the three predetermined websites mentioned above, rather than querying the whole web.

5. CONCLUSIONS

We have provided jWebMiner with new functionality, namely the ability to extract features based on Last.fm social tags. These features can be used alone or combined with jWebMiner's already existing Yahoo! search-based features, and the feature values can be summarized and saved in convenient formats for machine learning processing in future research.

We also performed web search experiments on different sets of excluded filter words and site weightings, in order to investigate their effect on genre classification accuracy. The experiments were performed on the SAC ground truth dataset, and improvements of 5.9% and 16.7% were achieved respectively for 5- and 10-genre classification compared to the results from earlier published experiments [13].

When performing the same experiments using social tags, we achieved an improvement of 8.2% for the 5-genre taxonomy, but also observed a 17.7% decrease in performance with the 10-genre taxonomy. It was thus found that social tags performed very well for broad genres, but lacked sufficient precision for more detailed sub-genre classes. However, the best results overall were achieved when both social tags and the web search data were combined. We conclude that the information obtained by each approach is at least partially, complementary.

In summary, the genre classification results obtained using both Last.fm social tags and Yahoo! web search-co-occurences were the highest observed amongst all

configurations experimented with. Indeed, this combined approach actually achieved results comparable to those found in [13], where symbolic and audio data were also available and more sophisticated machine learning-based classification methodologies were used. The combined approach described in this paper was also extremely successful in a just-published followup to the [13] experiments, as described in [15].

6. FUTURE RESEARCH

Although the music classification results obtained in our experiments with jWebMiner 2.0 are very promising, we believe that there is still more space for improvement in accuracy. First, in order to properly filter music tags used by contributors to social sites, we will work on the development of a thesaurus of terms that can group together related words, something that should result in more precise tag rankings. Secondly, we will develop software for customizing web search site weightings through the automated application of genetic algorithms. A third research step will be to explicitly experiment with other types of classification, such as mood classification. Finally, efforts will be made to take advantage of any open-source web services in order to reduce the dependency on the excellent but proprietary Last.fm and Yahoo! services.

7. REFERENCES

[1] Aucouturier, J.-J., and F. Pachet. 2003. Representing musical genre: A state of the art. *Journal of New Music Research*. 32 (1): 83–93.

[2] Baumann, S., and O. Hummel. 2003. Using cultural metadata for artist recommendations. *Proceedings of the Third International Conference on Web Delivering of Music*. 138–41.

[3] Casey, M.A., R. Veltkamp, M. Goto, M. Rhodes, C. Rhodes, and M. Slaney. 2008. Content-based music information retrieval: current directions and future challenges. *Proceedings of the IEEE*. 96 (4): 668–96.

[4] Geleijnse, G., and J. Korst. 2006. Learning effective surface text patterns for information extraction. *Proceedings of the EACL Workshop on Adaptative Text Extraction and Mining*. 1–8.

[5] Geleijnse, G., M. Schedl, and P. Knees. 2007. The quest for ground truth in musical artist tagging in the social web era. *Proceedings of the 8th International Conference on Music Information Retrieval*. 525–30.

[6] Knees, P., E. Pampalk, and G. Widmer. 2004. Artist classification with web-based data. *Proceedings of the 5th International Conference on Music Information Retrieval*. 517–24.

[7] Knees, P., M. Schedl, and T. Pohle. 2008. A deeper look into web-based classification of music artists. *Proceedings of the 2nd Workshop on Learning the Semantics of Audio Signals*. 31–44.

[8] Lamere, P. 2008. Social tagging and music information retrieval. *Journal of New Music Research*. 37 (2): 101–14.

[9] Levy, M., and M. Sandler. 2007. A semantic space for music derived from social tags. *Proceedings of the 8th International Conference on Music Information Retrieval*. 411–6.

[10] Levy, M., and M. Sandler. 2009. Music information retrieval using social tags and audio. *IEEE Transactions on Multimedia.* 11 (3): 383–95.

[11] McKay, C., and I. Fujinaga. 2006. Musical genre classification: Is it worth pursuing and how can it be improved? *Proceedings of the 7th International Conference on Music Information Retrieval.*101–6.

[12] McKay, C., and I. Fujinaga. 2007. jWebMiner: a web-based feature extractor. *Proceedings of the 8th International Conference on Music Information Retrieval*. 113–4.

[13] McKay, C., and I. Fujinaga. 2008. Combining features extracted from audio, symbolic and cultural sources. *Proceedings of the 9th International Conference on Music Information Retrieval*. 597–602.

[14] McKay, C., and I. Fujinaga. 2009. jMIR: Tools for automatic music classification. *Proceedings of the International Computer Music Conference*. 65–8.

[15] McKay, C., J. A. Burgoyne, J. Hockman, J. Smith, G. Vigliensoni, and I. Fujinaga. 2010. Evaluating the genre classification performance of lyrical features relative to audio, symbolic and cultural features. Accepted for publication at the *Int. Society for Music Information Retrieval Conference*. Utrecht, Netherlands.

[16] Turnbull, D., L. Barrington, and G. Lanckriet. 2008. Five approaches to collecting tags for music. *Proceedings of the 9th International Conference on Music Information Retrieval*. 225–30.

[17] Whitman, B., and S. Lawrence. 2002. Inferring descriptions and similarity for music from community metadata. *Proceedings of the 2002 International Computer Music Conference*. 591–8.

[18] Zadel, M., and I. Fujinaga. 2004. Web services for music information retrieval. *Proceedings of 5th International Conference on Music Information Retrieval*. 478–83.

[19] Last.fm Web Services. Retrieved 19 March 2010, from http://www.last.fm/api.

[20] Yahoo! Developer Network. Retrieved 19 March 2010, from http://developer.yahoo.com.

VOCALIST GENDER RECOGNITION IN RECORDED POPULAR MUSIC

Björn Schuller, Christoph Kozielski, Felix Weninger, Florian Eyben and Gerhard Rigoll
Institute for Human-Machine Communication
Technische Universität München
Munich, Germany
{lastname}@tum.de

ABSTRACT

We introduce the task of vocalist gender recognition in popular music and evaluate the benefit of Non-Negative Matrix Factorization based enhancement of melodic components to this aim. The underlying automatic separation of drum beats is described in detail, and the obtained significant gain by its use is verified in extensive test-runs on a novel database of 1.5 days of MP3 coded popular songs based on transcriptions of the Karaoke-game UltraStar. As classifiers serve Support Vector Machines and Hidden Naive Bayes. Overall, the suggested methods lead to fully automatic recognition of the pre-dominant vocalist gender at 87.31 % accuracy on song level for artists unkown to the system in originally recorded music.

1. INTRODUCTION

Determination of the gender of the (main) vocalist(s) is an astonishingly untouched task in the field of Music Information Retrieval (MIR): while there is a substantial body of literature dealing with gender in spoken language, e. g. to improve automatic speech recognition systems by switching or adapting acoustic models (e. g. [1]) or accordingly to improve emotion recognition systems (e. g. [22]), only some works consider singer identification (on artificial signals) [3, 10]. However, explicit recognition of the gender of the main performing vocal artist in original audio recordings of e. g. contemporary popular music has apparently not been addressed in MIR research, yet, which is to overcome, as like genre, mood or style, it can be an important feature for organizing and querying music collections, for example to find a song whose artist's name is unknown to the user, or for recommendation systems in on-line stores. In addition, it might be considered interesting as mid-level attribute for other MIR tasks as audio mood classification [14] or transcription of the sung lyrics with gender-adapted models – shown to be beneficial in [9].

Apart from finding suitable features and an appropriate classification method, as is the pre-concern for gender identification in (spoken) speech analysis, a setting dealing with the named original audio recordings of music demands for reasonable enhancement of the singer(s) voice given the background 'noise' of musical and rhythmic accompaniment. It is comparably easy to *eliminate* the main singer's voice in stereophonic recordings, e. g. for Karaoke application: often stereophonic channel subtraction killing the mid-panned parts suffices, as the lead singer's voice is usually panned there to be well audible at any position carrying the main melody (in fact the bass is usually panned there as well, which can be by-passed first). However, to *separate* these vocals is a non-trivial and challenging task – 'intelligent', i. e. data-driven, spectral decomposition is usually required to this end.

Non-negative Matrix Factorization (NMF) is one of the increasingly popular algorithms used within blind source separation of audio signals. Among other fields (e. g. speech recognition [18] or non-linguistic vocalisation recognition [16]), it has been successfully used in speaker separation [12], instrument separation [17, 23], especially drum beat separation [4, 15, 20], and vocal separation [10, 21]. While these methods provide audible results of great quality, it is not fully clear to which extent blind source separation can aid in general Music Information Retrieval tasks [15]. Here, we employ it to separate drum-beats from the rest of a popular music piece. While one could directly aim at separation of the vocals, this is considerably more difficult giving the large spectral overlap with other instruments. We thus decided to remove the relatively easier separable drum and percussion part and recognize gender in combination with general vocal presence in the remaining audio.

In this work the recognition system has to identify the gender of the performing artist particularily on 'real-world' data, i. e. originally recorded music without any preselection of 'friendly cases', which is a challenging task not only due to the above-named instrumental accompaniment, but also to the variety of genre and singing styles. In our experiments we introduce a database of popular songs from the Karaoke game UltraStar, which includes annotations of the tempo and the location of the sung parts. Gender is additionally labelled for the following experiments.

In the remainder of this paper, we first introduce the UltraStar database in section 2, then explain the acoustic features and the classifiers used for vocalist gender recognition in music in section 3, and the methodology applied for separating the drum beat with NMF in section 4. Then, sec-

tion 5 presents the data partitioning throughout test-runs and the experimental results before finally deriving conclusions in section 6.

2. ULTRASTAR DATABASE

We first introduce a data set of songs with annotations in the file format of the open-source Karaoke game UltraStar [2] for vocalist gender evaluation, referred to as *UltraStar database* in the ongoing. This set contains 582 complete songs and is encoded in MP3 (MPEG-1 Audio Layer 3) format with 44.1 kHz PCM and variable bit rate with a minimum of 128 kbit/s. In total length, this set corresponds to 37 h 06 min of music, i. e. 1.5 days of continuous music. The set covers generations from the 1960s until today and is a good example of typical popular music from diverse genres like Rock, Pop, Electronic Music, Ballads or Musical.

As annotation, we use the tempo and the information on the location of vocal presence in a song. In addition, we carried out a gender annotation on song level for this data set: per song we assigned the gender of the vocalist that is perceived as pre-dominant over the total run-time after listening to each full song. This was done by two labellers individually and in random order without any disagreement. Overall, 178 songs were labelled as female, and 404 songs as male, respectively 11 h 08 min female and 25 h 58 min male playtime. Prior to the processing all songs were down-mixed to monophonic by non-clipping stereo channel-addition.

Since every user of the UltraStar Karaoke game has the possibility to contribute annotations to the game's website, we chose songs according to their popularity among users, assuming that high popularity of a song indicates that a robust ground truth can be established.

3. GENDER RECOGNITION

3.1 Acoustic Features

For the actual gender recognition we consider the short-time energy, zero-, and mean-crossing rate known to indicate vocal presence [24]. In addition we extract values from the normalized autocorrelation sequence of the DFT coefficients, namely voicing probability, F-zero, and harmonics-to-noise ratio (HNR). F-zero is the location of the highest peak of the autocorrelation sequence aside from the maximum at zero. HNR is computed by the value of this peak. We further calculate Mel frequency cepstral coefficients (MFCC) 0–13 and their respective first-order delta regression coefficients. MFCC are known to capture the characteristic qualities of individual voices in speech and music for singer identification [8, 10, 11] and have proven highly meaningful in various speech gender recognition tasks [1, 22]. Thus, altogether we employ a set of 32 features.

Vocals in popular music are synchronous to the beats of a song most of the time. For every quarter beat we know from the annotations in the UltraStar database whether sung vocals are present or not. From this we derived an annotation on beat level by a non-ambiguous majority vote procedure: we judged vocals to be present in a beat if they are present in at least two of the quarter beats.

Based on this, every song in our data set is divided into analysis frames corresponding to the beats of the song. As the tempo and the locations of vocal presence are known for each song, beat synchronous chopping is possible for the training section and test section, to focus on the problem at hand. However, using the highly reliable automatic beat-tracker as introduced in [13] led to non-significant (one-tailed test, for testing conditions cf. below) differences in accuracy on song level. We divide the signal into non-overlapping frames with a Hamming window function of the length of a beat of the particular song – a strategy found beneficial over smaller units in previous tests. Per likewise beat-synchronous frame the above mentioned features are computed.

For easy reproducibility of the results we decided for open-source feature extraction by using the real-time toolkit for 'Speech and Music Interpretation by Large Space Extraction' (openSMILE) [1] .

3.2 Classifiers

We evaluate Support Vector Machines (SVM) with polynomial Kernel, sequential minimal optimization learning, and pairwise multi-class decision, as well as different Bayesian classifiers for our gender recognition task to be more independent of classifier influence.

A Bayesian network in general is a directed graph in which nodes represent attributes and branches represent attribute dependencies. It is quantified by conditional probabilities for each node dependent on its parents. In naive Bayes, each attribute node has the class node as its parent only, without any relation to other attributes, so it is the simplest Bayesian network [6]. In structure learned Bayesian networks every attribute node can have other attribute nodes as its parents, thus all dependencies between attributes are considered. However, achieving an optimal structure by learning from data is often impracticable in reasonable time. Hidden naive Bayes represents attribute dependencies by creating a hidden parent for each attribute node. This parent combines all influences of other attributes. Although attribute dependencies are considered, it keeps the structure of naive Bayes and does not need to be structure learnt [25].

For Bayes classification we found discretization by Kononenko's minimal description length criterion [5] based on the training instances beneficial (significant gain in average accuracy of 5.54 % on beat level with and without enhancement in the experiments as follows, for testing conditions cf. below), and Hidden Naive Bayes (HNB) superior to the considered alternatives (significant gain as before of 2.41 % over structure learned Bayesian networks, and of 7.68 % over Naive Bayes).

[1] http://www.openaudio.eu

4. DRUM BEAT SEPARATION USING NON-NEGATIVE MATRIX FACTORIZATION

4.1 Definition of NMF

Given a matrix $\mathbf{V} \in \mathbb{R}_{\geq 0}^{m \times n}$ and a constant $r \in \mathbb{N}$, non-negative matrix factorization (NMF) computes two matrices $\mathbf{W} \in \mathbb{R}_{\geq 0}^{m \times r}$ and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{r \times n}$, such that

$$\mathbf{V} \approx \mathbf{W} \cdot \mathbf{H} \tag{1}$$

For information reduction one generally chooses r such that $(m+n)r \ll mn$.

4.2 Application to Blind Source Separation

An important application area of NMF in signal processing is blind source separation. In the particular field of music processing, NMF has been successfully used to separate drum from harmonic sounds [4, 15, 20].

NMF-based blind source separation is usually realized in the frequency domain. Thereby the signal is split into overlapping frames of constant size. In our experiments, a frame size of 60 ms and an overlap of 50 % produced best results. Each frame is multiplied by a window function and transformed to the frequency domain using Discrete Fourier Transformation (DFT), with transformation size equal to the number of samples in each frame. We use the square root of the Hann function for windowing, as this helps to reduce artifacts when transforming back to the time domain [4].

Only the magnitudes of the DFT coefficients are retained, and the frame spectra are put in the columns of a matrix. Denoting the number of frames by n and the frame size by T, and considering the symmetry of the coefficients, this yields a $(\lfloor T/2 \rfloor + 1) \times n$ real matrix.

To exploit NMF for blind source separation, one assumes a *linear signal model*. Note that Eq. 1 can be written as follows (the subscripts $:,t$ and $:,j$ denote the t^{th} and j^{th} matrix columns, respectively):

$$\mathbf{V}_{:,t} \approx \sum_{j=1}^{r} \mathbf{H}_{j,t} \mathbf{W}_{:,j}, \quad 1 \leq t \leq n \tag{2}$$

Thus, if $\mathbf{V}$ is the magnitude spectrogram of a signal (with short-time spectra in columns), the factorization from Eq. 1 represents each short-time spectrum $\mathbf{V}_{:,t}$ as a linear combination of spectral basis vectors $\mathbf{W}_{:,j}$ with non-negative coefficients $\mathbf{H}_{j,t}$ $(1 \leq j \leq r)$.

We define the j^{th} *component* of the signal to be the pair $(\mathbf{w}_j, \mathbf{h}_j)$ of a spectrum $\mathbf{w}_j := \mathbf{W}_{:,j}$ along with its time-varying gains $\mathbf{h}_j := \mathbf{H}_{j,:}$ (the subscript $j,:$ denotes the j^{th} matrix row).

It has turned out that the non-negativity constraint on the coefficients alone is sufficient to decompose a signal into the underlying sources [17, 20]. Note that a 'source' in the intuitive sense, e. g. an instrument, can consist of multiple components.

When there is no prior knowledge about the number of spectra that can describe the source signal, the number of components r has to be chosen empirically. In our experiments, best results were achieved by setting $r = 30$.

4.3 Factorization Algorithm

A factorization according to Eq. 1 is usually achieved by iterative minimization of cost functions. For the purpose of drum beat separation, an extended form of the Kullback-Leibler (KL) divergence has been shown to yield good results [15, 20]:

$$c_d(\mathbf{W}, \mathbf{H}) = \sum_{i=1}^{m} \sum_{t=1}^{n} \left(\mathbf{V}_{i,t} \log \frac{\mathbf{V}_{i,t}}{(\mathbf{WH})_{i,t}} - (\mathbf{V} - \mathbf{WH})_{i,t} \right) \tag{3}$$

Eq. 3 can be enhanced by a term that enforces temporal continuity of the gains, improving separation quality [20] at the expense of increased computational costs. Because the perceived audio quality of components is of minor relevance for our task, we chose c_d from Eq. 3 as cost function and minimized it using Lee and Seung's multiplicative update algorithm [7]. It performs the following iterative updates of the matrices $\mathbf{W}$ and $\mathbf{H}$:

$$\mathbf{H}_{j,t} \leftarrow \mathbf{H}_{j,t} \frac{\sum_{i=1}^{m} \mathbf{W}_{i,j} \mathbf{V}_{i,t} / (\mathbf{WH})_{it}}{\sum_{i=1}^{m} \mathbf{W}_{i,j}} \tag{4}$$

for $j = 1, \ldots, r; t = 1, \ldots, n$ and

$$\mathbf{W}_{i,j} \leftarrow \mathbf{W}_{i,j} \frac{\sum_{t=1}^{n} \mathbf{H}_{j,t} \mathbf{V}_{i,t} / (\mathbf{WH})_{i,t}}{\sum_{t=1}^{n} \mathbf{H}_{j,t}} \tag{5}$$

for $i = 1, \ldots, m; j = 1, \ldots, r$.

Since in our scenario the spectral characteristics of the drum and harmonic sources in the signal are not known beforehand, we initialize $\mathbf{W}$ and $\mathbf{H}$ with random numbers drawn from a uniform distribution on the interval $]0, 1]$.

To reduce computational cost, instead of detecting convergence by computing the cost function (Eq. 3) after each iteration step, we run the algorithm for 100 iterations, after which in separation of popular music a reasonable separation quality is reached and convergence slows down considerably [15].

4.4 Synthesis of Harmonic Signals

Our goal is to obtain a drum-free signal from the NMF representation computed according to the previous section. To this end, we first classify the signal components $(\mathbf{w}_j, \mathbf{h}_j)$, $1 \leq j \leq r$ into two classes, 'drum' and 'harmonic'. Note that the gains $\mathbf{h}_j$ may also contain valuable features for discrimination of drum and harmonic sounds, since e. g. drum sounds are expected to be more periodic than harmonic sounds. The exact feature set and parameters used for classification will be described in the next section.

After classification, we compute a magnitude spectrogram $\mathbf{V}_{\text{harm}}$ of a signal that contains only harmonic sounds: Let $J_{\text{harm}} = \{j : (\mathbf{w}_j, \mathbf{h}_j) \text{ classified as harmonic}\}$. Then,

$$\mathbf{V}_{\text{harm}} = \sum_{j \in J_{\text{harm}}} \mathbf{w}_j \mathbf{h}_j \tag{6}$$

We transfer $\mathbf{V}_{\text{harm}}$ back to the time domain applying a column-wise inverse DFT, using the phase matrix from the

# beats	training	develop	test	sum
female	39 267	25 354	12 856	77 477
male	60 210	55 429	40 805	156 444
no voice	73 512	64 568	37 447	175 527
sum	**172 989**	**145 351**	**91 108**	**409 448**

Table 1: Number of beats per set of the UltraStar database.

original signal. Finally, we obtain a 'harmonic' time signal by windowing each time frame with the square root of the Hann function, then employing an overlap-add procedure.

4.5 Discrimination of Drum and Harmonic Components

For discrimination of drum and harmonic components, we use a linear SVM classifier with a feature set similar to the ones proposed in [4] and [15].

From the spectral vectors $\mathbf{w}_j$, we compute MFCC, using 26 triangular filters on a bank ranging from 20 Hz to 8 kHz. 10 first MFCC plus the zeroth (energy) coefficient are considered. Furthermore, we add sample standard deviation (using the common unbiased estimator), spectral centroid, 95 % roll-off point, and noise-likeness [19]. While these spectral features are quite common in pattern recognition, the temporal features computed from the gains vectors $\mathbf{h}_j$ are more specific to the drum beat separation task. In detail, we use percussiveness [19], periodicity, average peak length, and peak fluctuation [4, 15].

For NMF computation and extraction of the named features the open source 'Blind Source Separation for Audio Retrieval Tasks' (BliSSART) [2] package is used for reproducibility reasons. To train the SVM classifier, we used a data set generated from a popular music collection which is presented in detail in [15]. It consists of 95 drum and 249 harmonic components computed by NMF on song extracts. In 10-fold stratified cross validation of this data set, a classification accuracy of 96.2 % was achieved.

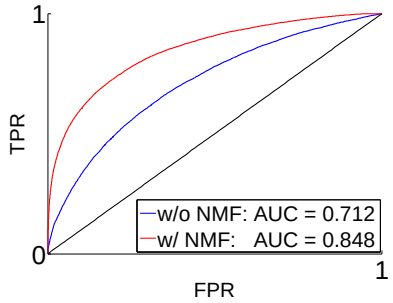
5. EXPERIMENTS

5.1 Data Partitioning

We partitioned the UltraStar database introduced in section 2 into the three groups: training, develop, and test. The training set (241 songs) contains all artists beginning with A,D,G,..., the develop set (207 songs) artists beginning with B,E,H,..., and the test set (134 artists) those beginning with C,F,I,...,0-9. Note that this dividing setup provides that all songs of one artist are in the same set, thus processing is strictly independent of the artist. See Table 1 for the gender distribution on beat level.

The features for the actual gender determination as introduced in section 3 were extracted from the original song files. In addition we created the same sets with the features extracted from the discriminated harmonic segments after applying our NMF algorithm and drum beat separation as introduced in section 4 on the original song.

[2] http://www.openaudio.eu



(a) SVM, 2-class

Figure 1: ROC by true (TPR) over false positive rate (FPR), and the area under the curve (AUC) for the two-(female / male) class task.

We evaluate two different tasks: first, three classes (no voice / female / male) to evaluate vocal presence localization in combination with gender recognition; second, two classes (female / male) where we only consider beats with vocal presence to judge whether performance is increased particularly in gender discrimination by NMF-based drum separation.

We trained with our training set and evaluated on the develop set to verify feature relevance before and after drum-beat separation. For optimization, we applied random down-sampling – i. e. elimination of instances – to the training set to achieve a balanced distribution of instances among classes.

5.2 Results

While training with the training and evaluating with the develop set, we found that every extracted feature was relevant, as classification performance could not be improved by removing features of a certain type. We thus kept the full feature-set as described in section 3 for the oncoming evaluations. For our final results we next merged the training and develop sets for classifier training, and evaluated on the test set for representative performances.

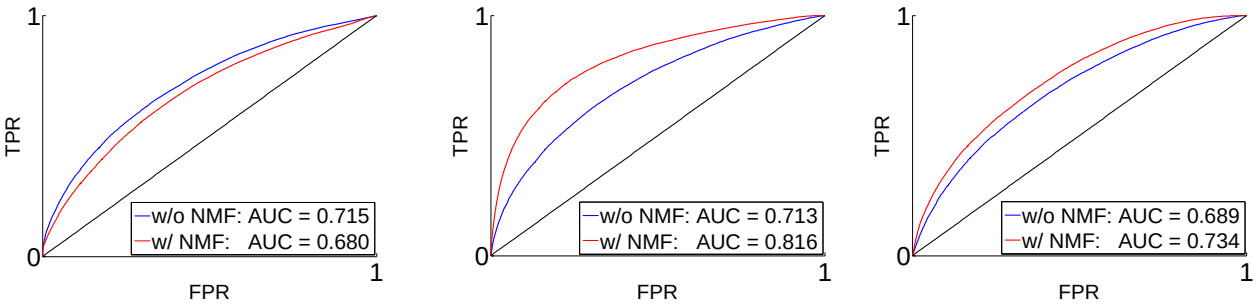
First we consider the results with three classes on beat level (cf. Table 2, columns 'beat'): performance is improved for SVM and HNB by NMF-based drum-beat separation, and a maximum accuracy of 58.84 % is reached by HNB. Next looking at the obtained performances with only two classes (female / male) on beat level, one again notices a considerable boost by drum-beat separation for all classifiers. Again, SVM benefit most, while HNB reaches highest level at 80.22 % accuracy.

Next, we shift to the level of the whole song, and identify the gender of the mainly heard vocalist(s): after classification per beat we estimate a song's overall gender either by majority vote or alternatively by adding the prediction scores per gender and choosing the maximum overall score.

Table 2 (columns 'song') shows the according results. Minor differences – and only for HNB – are observable between majority vote and the maximum added score. The

Accuracy [%]	classification scheme	w/o NMF beat	w/o NMF song	w/ NMF beat	w/ NMF song
-/f/m	HNB vote	58.54	79.85	58.84	84.33
	HNB added score		79.85		85.07
	SVM vote / added score	52.06	79.85	56.54	**87.31**
f/m	HNB vote	70.35	82.09	80.22	85.82
	HNB added score		83.58		86.57
	SVM vote / added score	67.52	82.09	79.97	**89.55**

Table 2: *Accuracies of vocalist gender recognition on beat level and on song level by majority voting or maximum added score for HNB and SVM as classifier – once with (w/) and once without (w/o) separation of drum-beats by NMF. Considered are no voice (-), female (f), and male (m) discrimination on all beats or only gender on those with vocal presence.*



(a) no voice (SVM, 3-class) (b) female (SVM, 3-class) (c) male (SVM, 3-class)

Figure 2: ROC by true (TPR) over false positive rate (FPR), and the area under the curve (AUC) for the three-class (female / male / no voice) task.

latter is found beneficial – as one would assume – as information on certainty is preserved prior to the song level decision. The results further indicate that the accuracy of classification on beat level is sufficiently above chance level to allow for repairing of mispredictions over the duration of a song. Here, SVM perform slightly better with a maximum of 87.31 % accuracy for the three classes, and the difference to the two-class task is drastically reduced. Overall, the statistical significance on song level for the improvement gained by NMF utilization is 0.05. Thus, we can state that drum separation by NMF helps recognizing gender even on the song level.

To shed light on this effect per class, according Receiver Operating Characteristics (ROC) are depicted in Figure 1 for the two-class task of gender recognition, and in Figure 2 for the three-class task with additional determination of positions that contain vocals by SVM. To provide a single value rather than a curve, one can calculate the area under the ROC curve, called AUC. The highest possible AUC is 1.0, equal to the whole graph area, and achievable only by a perfect classifier. Random guessing has an AUC of 0.5 since it corresponds to the diagonal line in the ROC space. A reasonable classifier should therefore have an AUC that is significantly greater than 0.5, with better classifiers yielding higher values. The values obtained are also shown in Figure 1: For the two-class task (female / male) the difference in the AUC with and without NMF is highly significant at the 10^{-3} level. In the three-class problem clear differences are observable: the highest benefit is reached for female vocals, next come male vocals, and interestingly the recognition of parts without vocal presence is negatively affected by reduction of the drum beat presence.

6. CONCLUSION

It was our primary goal to predict the vocalist gender in originally recorded popular music, and our secondary to analyze whether NMF usage for separation of the drum-beat can help improve on this task. The results clearly demonstrate the significant improvement obtained, and we are by that able to fully automatically identify the gender of the main vocalist in popular music at a high and reasonable accuracy for system unknown artists and songs. On beat level NMF application slightly impairs vocals presence estimation, but increases the overall performance of gender classification explaining the better results on song level.

Considering the choice of classifier, no clear tendency was found, apart from the fact that the overall best result was obtained by SVM.

Future refinement can be invested in improved annotation: as mentioned, the UltraStar annotations were created by members of the game community. Therefore errors among the ground truth tempo and vocals' locations might be present though we chose the most frequently used files. A widespread verification of the annotations would minimize the error rate and maybe reduce false classifications. But that would need a huge investment of time.

Further, we assigned the main vocalist gender. How-

ever, alternatively local labeling and consideration of mixed gender or choir passages could be provided.

Finally and self-evident, tailored vocal instead of drum separation should be targeted now that the more robustly obtainable separation of drum-beats was already found significantly beneficial.

7. ACKNOWLEDGMENT

This work was supported by the Federal Republic of Germany through the German Research Foundation (DFG) under the grant no. SCHU 2508/2-1 ("Non-Negative Matrix Factorization for Robust Feature Extraction in Speech Processing").

8. REFERENCES

[1] W. H. Abdulla and N. K. Kasabov. Improving speech recognition performance through gender separation. In *Proc. of ANNES*, pages 218–222, Dunedin, New Zealand, 2001.

[2] P. Cebula. UltraStar - PC conversion of famous karaoke game SingStar, 2009.

[3] H. Fujihara, T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno. Singer identification based on accompaniment sound reduction and reliable frame selection. In *Proc. of ISMIR*, pages 329–336, 2005.

[4] M. Helen and T. Virtanen. Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine. In *Proc. of EUSIPCO*, Antalya, Turkey, 2005.

[5] I. Kononenko. On biases in estimating multi-valued attributes. In *Proc. of IJCAI*, pages 1034–1040, Montreal, Quebec, Canada, 1995.

[6] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proc. of AAAI*, pages 223–228, San Jose, CA, USA, 1992.

[7] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proc. of NIPS*, pages 556–562, Vancouver, Canada, 2001.

[8] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Proc. of MUSIC-IR*, Plymouth, MA, USA, 2000.

[9] A. Mesaros and T. Virtanen. Automatic recognition of lyrics in singing. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009:24 pages, 2009.

[10] A. Mesaros, T. Virtanen, and A. Klapuri. Singer identification in polyphonic music using vocal separation and pattern recognition methods. In *Proc. of ISMIR*, pages 375–378, 2007.

[11] D. A. Reynolds. Experimental evaluation of features for robust speaker identification. *IEEE Transactions on Speech and Audio Processing*, 2(4):639–643, 1994.

[12] M. N. Schmidt and R. K. Olsson. Single-channel speech separation using sparse non- negative matrix factorization. In *Proc. of Interspeech*, Pittsburgh, Pennsylvania, USA, 2006.

[13] B. Schuller, F. Eyben, and G. Rigoll. Tango or Waltz?: Putting ballroom dance style into tempo detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2008(846135):12 pages, 2008.

[14] B. Schuller, C. Hage, D. Schuller, and G. Rigoll. 'Mister D.J., Cheer Me Up!': Musical and Textual Features for Automatic Mood Classification. *Journal of New Music Research*, 38:33 pages, 2009.

[15] B. Schuller, A. Lehmann, F. Weninger, F. Eyben, and G. Rigoll. Blind enhancement of the rhythmic and harmonic sections by NMF: Does it help? In *Proc. of International Conference on Acoustics (NAG/DAGA 2009)*, pages 361–364, Rotterdam, The Netherlands, 2009.

[16] B. Schuller and F. Weninger. Discrimination of speech and non-linguistic vocalizations by non-negative matrix factorization. In *Proc. of ICASSP*, Dallas, TX, 2010.

[17] P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *Proc. of WASPAA*, pages 177–180, 2003.

[18] V. Stouten, K. Demuynck, and H. van Hamme. Discovering phone patterns in spoken utterances by non-negative matrix factorization. *IEEE Signal Processing Letters*, 15:131–134, 2008.

[19] C. Uhle, C. Dittmar, and T. Sporer. Extraction of drum tracks from polyphonic music using independent subspace analysis. In *Proc. of ICA*, Nara, Japan, 2003.

[20] T. Virtanen. Monaural sound source separation by non-negative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech and Language Processing*, 15(3):1066–1074, March 2007.

[21] T. Virtanen, A. Mesaros, and M. Ryynänen. Combining pitch-based inference and non-negative spectrogram factorization in separating vocals from polyphonic music. In *Proc. of SAPA*, Brisbane, Australia, 2008.

[22] T. Vogt and E. Andre. Improving automatic emotion recognition from speech via gender differentiation. In *Proc. of LREC*, Genoa, Italy, 2006.

[23] B. Wang and M. D. Plumbley. Musical audio stream separation by non-negative matrix factorization. In *Proc. of DMRN Summer Conference*, Glasgow, Scotland, 2005.

[24] C. Xu, N. C. Maddage, and X. Shao. Automatic music classification and summarization. *IEEE Transactions on Speech and Audio Processing*, 13(3):441–450, 2005.

[25] H. Zhang, L. Jiang, and J. Su. Hidden naive Bayes. In *Proc. of AAAI*, pages 919–924, Pittsburgh, PA, USA, 2005.

WHEN LYRICS OUTPERFORM AUDIO FOR MUSIC MOOD CLASSIFICATION: A FEATURE ANALYSIS

Xiao Hu **J. Stephen Downie**
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
xiaohu@illinois.edu jdownie@illinois.edu

ABSTRACT

This paper builds upon and extends previous work on multi-modal mood classification (i.e., combining audio and lyrics) by analyzing in-depth those feature types that have shown to provide statistically significant improvements in the classification of individual mood categories. The dataset used in this study comprises 5,296 songs (with lyrics and audio for each) divided into 18 mood categories derived from user-generated tags taken from last.fm. These 18 categories show remarkable consistency with the popular Russell's mood model. In seven categories, lyric features significantly outperformed audio spectral features. In one category only, audio outperformed all lyric features types. A fine grained analysis of the significant lyric feature types indicates a strong and obvious semantic association between extracted terms and the categories. No such obvious semantic linkages were evident in the case where audio spectral features proved superior.

1. INTRODUCTION

User studies in Music Information Retrieval (MIR) have found that music mood is a desirable access point to music repositories and collections (e.g., [1]). In recent years, automatic methods have been explored to classify music by mood. Most studies exploit the audio content of songs, but some studies have been using song lyrics in music mood classification as well [2-4].

Music mood classification studies using both audio and lyrics consistently find that combining lyric and audio features improves classification performance (See Section 2.3). However, there are contradictory findings on whether audio or lyrics are more useful in predicting music mood, or which source is better for individual mood classes. In this paper, we continue our previous work on multi-modal mood classification [4] and go one step further to investigate these research questions: 1) Which source is more useful in music classification: audio or lyrics? 2) For which moods is audio more useful and for which moods are lyrics more useful? and, 3) How do lyric features associate with different mood categories? Answers to these questions can help shed light on a profoundly important music perception question: How does the interaction of sound and text establish a music mood?

This paper is organized as follows: Section 2 reviews related work on music mood classification. Section 3 introduces our experimental dataset and the mood categories used in this study. Section 4 describes the lyric and audio features examined. Section 5 discusses our findings in light of our research questions. Section 6 presents our conclusions and suggests future work.

2. RELATED WORK

2.1 Music Mood Classification Using Audio Features

Most existing work on automatic music mood classification is exclusively based on audio features among which spectral and rhythmic features are the most popular (e.g., [5-7]). Since 2007, the Audio Mood Classification (AMC) task has been run each year at the Music Information Retrieval Evaluation eXchange (MIREX) [8], the community-based framework for the formal evaluation of MIR techniques. Among the various audio-based approaches tested at MIREX, spectral features and Support Vector Machine (SVM) classifiers were widely used and found quite effective [9].

2.2 Music Mood Classification Using Lyric Features

Studies on music mood classification solely based on lyrics have appeared in recent years (e.g., [10,11]). Most used bag-of-words (BOW) features in various unigram, bigram, trigram representations. Combinations of unigram, bigram and trigram tokens performed better than individual n-grams, indicating higher-order BOW features captured more of the semantics useful for mood classification. Features used in [11] were novel in that they were extracted based on a psycholinguistic resource, an affective lexicon translated from the Affective Norm of English Words (ANEW) [12].

2.3 Multi-modal Music Mood Classification Using Both Audio and Lyric Features

Yang and Lee [13] is often regarded as one of the earliest studies on combining lyrics and audio in music mood classification. They used both lyric BOW features and the 182 psychological features proposed in the General Inquirer [14] to disambiguate categories that audio-based classifiers found confusing. Besides showing improved classification accuracy, they also presented the most salient psychological features for each of the considered mood categories. Laurier et al. [2] also combined audio and lyric BOW features and showed that the combined features improved classification accuracies in all four of their categories. Yang et al. [3] evaluated both unigram and bigram BOW lyric features as well as three methods for fusing lyric and audio sources and concluded that le-

veraging lyrics could improve classification accuracy over audio-only classifiers.

Our previous work [4] evaluated a wide range of lyric features from n-grams to features based on psycholinguistic resources such as WordNet-Affect [15], General Inquirer and ANEW, as well as their combinations. After identifying the best lyric feature types, audio-based, lyric-based as well as multi-modal classification systems were compared. The results showed the multi-modal system performed the best while the lyric-based system outperformed the audio-based system. However, our reported performances were accuracies ***averaged*** across all of our 18 mood categories. In this study, we go deeper to investigate the performance differences of the aforementioned feature types on ***individual*** mood categories. More precisely, this paper examines, in some depth, those feature types that provide statistically significant performance improvements in identifying individual mood categories.

2.4 Feature Analysis in Text Sentiment Classification

Except for [13], most existing studies on music mood classification did not analyze or compare which specific feature values were the most useful. However, feature analysis has been widely used in text sentiment classification. For example, a study on blogs, [16] identified discriminative words in blog postings between two categories, "happy" and "sad" using Naïve Bayesian classifiers and word frequency thresholds. [17] uncovered important features in classifying customer reviews with regard to ratings, object types, and object genres, using frequent pattern mining and naïve Bayesian ranking. Yu [18] presents a systematic study of sentiment features in Dickenson's poems and American novels. Besides identifying the most salient sentiment features, it also concluded that different classification models tend to identify different important features. These previous works inspired the feature ranking methods examined in this study.

3. DATASET AND MOOD CATEGORIES

3.1 Experimental Dataset

As mentioned before, this study is a continuation of a previous study [4], and thus the same dataset is used. There are 18 mood categories represented in our dataset, and each of the categories comprises 1 to 25 mood-related social tags downloaded from last.fm. A mood category consists of tags that are synonyms identified by WordNet-Affect and verified by two human experts who are both native English speakers and respected MIR researchers. The song pool was limited to those audio tracks at the intersection of being available to the authors, having English lyrics available on the Internet, and having social tags available on last.fm. For each of these songs, if it was tagged with any of the tags associated with a mood category, it was counted as a positive example of that category. In this way, one single song could belong to multiple mood categories. This is in fact more realistic than a single-label setting since a music piece may carry multiple moods such as "happy and calm" or "aggressive and depressed".

A binary classification approach was adopted for each of the mood categories. Negative examples of a category were songs that were not tagged with any of the tags associated with this category but were heavily tagged with many other tags. Table 1 presents the mood categories and the number of positive songs in each category. We balanced equally the positive and negative set sizes for each category. This dataset contains 5,296 unique songs in total. This number is much smaller than the total number of examples in all categories (which is 12,980) because categories often share samples.

Category	No. of songs	Category	No. of songs	Category	No. of songs
calm	1,680	angry	254	anxious	80
sad	1,178	mournful	183	confident	61
glad	749	dreamy	146	hopeful	45
romantic	619	cheerful	142	earnest	40
gleeful	543	brooding	116	cynical	38
gloomy	471	aggressive	115	exciting	30

Table 1. Mood categories and number of positive examples

3.2 Mood Categories

Music mood categories have been a much debated topic in both MIR and music psychology. Most previous studies summarized in Section 2 used two to six mood categories which were derived from psychological models. Among the many emotion models in psychology, Russell's model [19] seems the most popular in MIR research (e.g., [2, 5]).

Russell's model is a *dimensional* model where emotions are positioned in a continuous multidimensional space. There are two dimensions in Russell's model: *valence* (negative-positive) and *arousal* (inactive-active). As shown in Figure 1, this model places 28 emotion-denoting adjectives on a circle in a bipolar space subsuming these two dimensions.

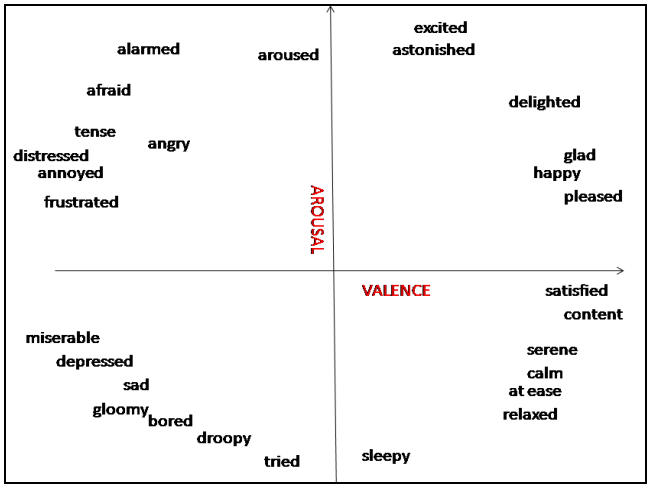


Figure 1. Russell's model with two dimensions

From Figure 1, we can see that Russell's space demonstrates relative distances or similarities between moods. For instance, "sad" and "happy", "calm" and "angry" are at opposite places while "happy" and "glad" are close to each other.

The relative distance between the 18 mood categories in our dataset can also be calculated by co-occurrence of

songs in the positive examples. That is, if two categories share many positive songs, they should be similar. Figure 2 illustrates the relative distances of the 18 categories plotted in a 2-dimensional space using Multidimensional Scaling where each category is represented by a bubble in a size proportional to the number of positive songs in this category.

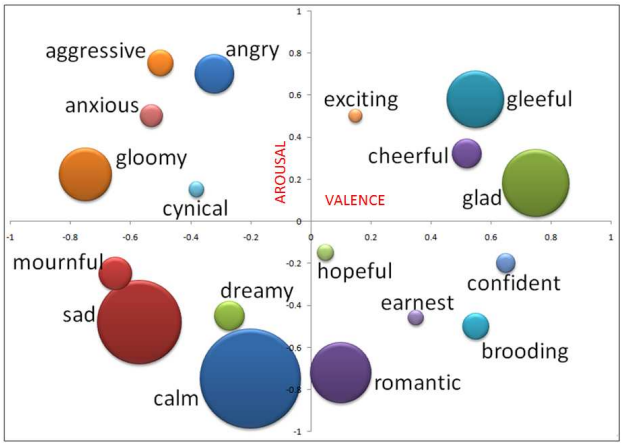


Figure 2. Distances between the 18 mood categories in the experimental dataset

The patterns shown in Figure 2 are similar to those found in Figure 1: 1) Categories placed together are intuitively similar; 2) Categories at opposite positions represent contrasting moods; 3) The horizontal and vertical dimensions correspond to *valence* and *arousal* respectively. Taken together, these similarities indicate that our 18 mood categories fit well with Russell's mood model which is the most commonly used model in MIR mood classification research.

4. LYRIC AND AUDIO FEATURES

In [4], we systematically evaluated a range of lyric feature types on the task of music mood classification, including: 1) basic text features that are commonly used in text categorization tasks; 2) linguistic features based on psycholinguistic resources; and, 3) text stylistic features. In this study, we analyze the most salient features in each of these feature types. This section briefly introduces these feature types. For more detail, please consult [4].

4.1 Features based on N-grams of Content Words

"Content words" (CW) refer to all words appearing in lyrics except function words (also called "stop words"). Words were not stemmed as our earlier work showed stemming did not yield better results. The CW feature set used was a combination of unigrams, bigrams and trigrams of content words since this combination performed better than each of the n-gram types individually [4]. For each n-gram, features that occurred less than five times in the training dataset were discarded. Also, for bigrams and trigrams, function words were not eliminated because content words are usually connected via function words as in "I love you" where "I" and "you" are function words. There were totally 84,155 CW n-gram features.

4.2 Features based on General Inquirer

General Inquirer (GI) is a psycholinguistic lexicon containing 8,315 unique English words and 182 psychological categories [14]. Each of the 8,315 words in the lexicon is manually labeled with one or more of the 182 psychological categories to which the word belongs. For example, the word "happiness" is associated with the categories "Emotion", "Pleasure", "Positive", "Psychological well being", etc. GI's 182 psychological features were a feature type evaluated in [4], and denoted as "GI".

Each of the 8,315 words in General Inquirer conveys certain psychological meanings and thus were evaluated in [4]. In this feature set (denoted as "GI-lex"), feature vectors were built using only these 8,315 words.

4.3 Features based on ANEW and WordNet

Affective Norms for English Words (ANEW) is another specialized English lexicon [12]. It contains 1,034 unique English words with scores in three dimensions: *valence* (a scale from unpleasant to pleasant), *arousal* (a scale from calm to excited), and *dominance* (a scale from submissive to dominated). As these 1,034 words are too few to cover all the songs in our dataset, we expanded the ANEW word list using WordNet [20] such that synonyms of the 1,034 words were included. This gave us 6,732 words in the expanded ANEW. We then further expanded this set of affect-related words by including the 1,586 words in WordNet-Affect [15], an extension of WordNet containing emotion related words. Therefore, this set of 7,756 affect-related words formed a feature type denoted as "Affe-lex".

4.4 Text Stylistic Features

The text stylistic features evaluated in [4] included such text statistics as number of unique words, number of unique lines, ratio of repeated lines, number of words per minute, as well as special punctuation marks (e.g., "!") and interjection words (e.g., "hey"). There were 25 text stylistic features in total.

4.5 Audio Features

In [4] we used the audio features selected by the MARSYAS submission [21] to MIREX because it was the leading audio-based classification system evaluated under both the 2007 and 2008 Audio Mood Classification (AMC) task. MARSYAS used 63 spectral features: means and variances of Spectral Centroid, Rolloff, Flux, Mel-Frequency Cepstral Coefficients (MFCC), etc. Although there are audio features beyond spectral ones, spectral features were found the most useful and most commonly adopted for music mood classification [9]. We leave it as our future work to analyze a broader range of audio features.

5. RESULTS AND DISCUSSIONS

5.1 Feature Performances

Table 2 shows the accuracies of each aforementioned feature set on individual mood categories. Each of the accu-

racy values was averaged across a 10-fold cross validation. For each lyric feature set, the categories where its accuracies are significantly higher than that of the audio feature set are marked as bold (at $p < 0.05$). Similarly, for the audio feature set, bold accuracies are those significantly higher than all lyric features (at $p < 0.05$).

Category	CW	GI	GI-lex	Affe-lex	Stylistic	Audio
calm	0.5905	0.5851	0.5804	0.5708	0.5039	**0.6574**
sad	0.6655	0.6218	0.6010	0.5836	0.5153	0.6749
glad	0.5627	0.5547	0.5600	0.5508	0.5380	0.5882
romantic	**0.6866**	0.6228	**0.6721**	0.6333	0.5153	0.6188
gleeful	0.5864	0.5763	0.5405	0.5443	0.5670	0.6253
gloomy	0.6157	0.5710	0.6124	0.5859	0.5468	0.6178
angry	**0.7047**	0.6362	0.6497	**0.6849**	0.4924	0.5905
mournful	0.6670	0.6344	0.5871	0.6615	0.5001	0.6278
dreamy	0.6143	0.5686	0.6264	0.6269	0.5645	0.6681
cheerful	**0.6226**	0.5633	0.5707	0.5171	0.5105	0.5133
brooding	0.5261	0.5295	0.5739	0.5383	0.5045	0.6019
aggressive	**0.7966**	**0.7178**	**0.7549**	0.6746	0.5345	0.6417
anxious	**0.6125**	0.5375	0.5750	0.5875	0.4875	0.4875
confident	0.3917	0.4429	0.4774	0.5548	0.5083	0.5417
hopeful	**0.5700**	0.4975	**0.6025**	**0.6350**	**0.5375**	0.4000
earnest	0.6125	0.6500	0.5500	0.6000	0.6375	0.5750
cynical	0.7000	0.6792	0.6375	0.6667	0.5250	0.6292
exciting	0.5833	0.5500	**0.5833**	0.4667	**0.5333**	0.3667
AVERAGE	0.6172	0.5855	0.5975	0.5935	0.5290	0.5792

Table 2.Accuracies of feature types for individual categories

From the averaged accuracies in Table 2, we can see that whether lyrics are more useful than audio, or vice versa depends on which feature sets are used. For example, if using CW n-grams as features, lyrics are more useful than audio spectral features in terms of overall classification performance averaged across all categories. However, the answer is reversed if text stylistics is used as lyric features (i.e., audio works better).

The accuracies marked in bold in Table 2 demonstrate that lyrics and audio have their respective advantages in ***different*** mood categories. Audio spectral features significantly outperformed *all* lyric feature types in only one mood category: "calm". However, lyric features achieved significantly better performance than audio in seven divergent categories: "romantic", "angry", "cheerful", "aggressive", "anxious", "hopeful" and "exciting".

In the following subsections, we will rank (by order of influence), and then examine, the most salient features of those lyric feature types that outperformed audio features in the seven aforementioned mood categories. Support Vector Machines (SVM) were adopted as the classification model in [4] where a variety of kernels were tested and a linear kernel was finally chosen. In a linear SVM, each feature was assigned a weight indicating its influence in the classification model, and thus the features in this study were ranked by the assigned weights in the same SVM models trained in experiments in [4].

5.2 Top Features in Content Word N-Grams

There are six categories where CW n-gram features significantly outperformed audio features. Table 3 lists the top-ranked content word features in these categories. Note how "love" seems an eternal topic of music regardless of the mood category! Highly ranked content words seem to have intuitively meaningful connections to the categories, such as "with you" in "romantic" songs, "happy" in "cheerful" songs, and "dreams" in "hopeful" songs. The categories, "angry", "aggressive" and "anxious" share quite a few top-ranked terms highlighting their emotional similarities. It is interesting to note that these last three categories sit in the same top-left quadrant in Figure 2.

romantic	cheerful	hopeful	angry	aggressive	anxious
with you	i love	you ll	baby	fuck	hey
on me	night	strong	i am	dead	to you
with your	ve got	i get	shit	i am	change
crazy	happy	loving	scream	girl	left
come on	for you	dreams	to you	man	fuck
i said	new	i ll	run	kill	i know
burn	care	if you	shut	baby	dead
hate	for me	to be	i can	love	and if
kiss	living	god	control	hurt	wait
let me	rest	lonely	don t know	but you	waiting
hold	and now	friend	dead	fear	need
to die	all around	dream	love	don t	i don t
why you	heaven	in the eye	hell	pain	i m
i ll	met	coming	fighting	lost	listen
tonight	she says	want	hurt you	i ve never	again and
i want	you ve got	wonder	kill	hate	but you
love	more than	waiting	if you want	have you	my heart
give me	the sun	i love	oh baby	love you	hurt
cry	you like	you best	you re my	yeah yeah	night

Table 3. Top-ranked content word features for moods where content words significantly outperformed audio

5.3 Top-Ranked Features Based on General Inquirer

"Aggressive" is the only category where the GI set of 182 psychological features outperformed audio features with a statistically significant difference. Table 4 lists the top GI features for this category.

GI Feature	Example Words
Words connoting the physical aspects of well being, including its absence	blood, dead, drunk, pain
Words referring to the perceptual process of recognizing or identifying something by means of the senses	dazzle, fantasy, hear, look, make, tell, view
Action words	hit, kick, drag, upset
Words indicating time	noon, night, midnight
Words referring to all human collectivities	people, gang, party
Words related to a loss in a state of well being,	burn, die, hurt, mad

Table 4. Top GI features for "aggressive" mood category

It is somewhat surprising that the psychological feature indicating "hostile attitude or aggressiveness" (e.g., "devil", "hate", "kill") was ranked at 134 among the 182 features. Although such individual words ranked high as content word features, the GI features were aggregations of certain kinds of words. The mapping between words and psychological categories provided by GI can be very helpful in looking beyond word forms and into word meanings.

By looking at rankings on specific words in General Inquirer, we can have a clearer understanding about which GI words were important. Table 5 presents top GI word features in the four categories where "GI-lex" features significantly outperformed audio features.

romantic	aggressive	hopeful	exciting
paradise	baby	i'm	come
existence	fuck	been	now
hit	let	would	see
hate	am	what	up
sympathy	hurt	do	will
jealous	girl	in	tear
kill	be	lonely	bounce
young	another	saw	to
destiny	need	like	him
found	kill	strong	better
anywhere	can	there	shake
soul	but	run	everything
swear	just	will	us
divine	because	found	gonna
across	man	when	her
clue	one	come	free
rascal	dead	lose	me
tale	alone	think	more
crazy	why	mine	keep

Table 5. Top-ranked GI-lex features for categories where GI-lex significantly outperformed audio

5.4 Top Features Based on ANEW and WordNet

According to Table 2, "Affe-lex" features worked significantly better than audio features on categories "angry" and "hopeful". Table 6 presents top-ranked features.

Category	Top Features (in order of influence)
angry	one, baby, surprise, care, death, alive, guilt, happiness, hurt, straight, thrill, cute, suicide, babe, frightened, motherfucker, down, misery, mad, wicked, fighting, crazy
hopeful	wonderful, sun, words loving, read, smile, better, heart, lonely, friend, free, hear, come, found, strong, letter, grow, safe, god, girl, memory, happy, think, dream

Table 6. Top Affe-lex features for categories where Affe-lex significantly outperformed audio

Again, these top-ranked features seem to have strong semantic connections to the categories, and they share common words with the top-ranked features listed in Tables 3 and 5. Although both Affe-lex and GI-lex are domain-oriented lexicons built from psycholinguistic resources, they contain different words, and thus each of them identified some novel features that are not shared by the other.

5.5 Top Text Stylistic Features

Text stylistic features performed the worst among all feature types considered in this study. In fact, the average accuracy of text stylistic features was significantly worse than each of the other feature types ($p < 0.05$). However, text stylistic features did outperform audio features in two categories: "hopeful" and "exciting". Table 7 shows the top-ranked stylistic features in these two categories.

Note how the top-ranked features in Table 7 are all text statistics without interjection words or punctuation marks. These kinds of text statistics capture very different characteristics of the lyrics from other word-based features, and thus combining these statistics and other features may yield better classification performance. Also noteworthy is that these two categories both have relatively low positive valence (but opposite arousal) as shown in Figure 2.

hopeful	exciting
Std of number of words per line	Average number of unique words per line
Average number of unique words per line	Average repeating word ratio per line
Average word length	Std of number of words per line
Ratio of repeating lines	Ratio of repeating words
Average number of words per line	Ratio of repeating lines
Ratio of repeating words	Average number of words per line
Number of unique lines	Number of blank lines

Table 7. Top-ranked text stylistic features for categories where text stylistics significantly outperformed audio

5.6 Top Lyric Features in "Calm"

"Calm", which sits in the bottom-left quadrant and has the lowest arousal of any category (Figure 2), is the only mood category where audio features were significantly better than all lyric feature types. It is useful to compare the top lyric features in this category to those in categories where lyric features outperformed audio features. Top-ranked words and stylistics from various lyric feature types in "calm" are shown in Table 8.

CW	GI-lex	Affe-lex	Stylistic
you all look	float	list	Standard derivation (std) of repeating word ratio per line
all look	eager	moral	
all look at	irish	saviour	Repeating word ratio
you all i	appreciate	satan	Average repeating word ratio per line
burning	kindness	collar	
that is	selfish	pup	Repeating line ratio
you d	convince	splash	Interjection: "Hey"
control	foolish	clams	Average number of unique words per line
boy	island	blooming	
that s	curious	nimble	Number of lines per minute
all i	thursday	disgusting	Blank line ratio
believe in	pie	introduce	Interjection: "ooh"
be free	melt	amazing	Average number of words per line
speak	couple	arrangement	
blind	team	mercifully	Interjection: "ah"
beautiful	doorway	soaked	Punctuation: "!"
the sea	lowly	abide	Interjection: "yo"

Table 8. Top lyric features in "calm" category

As Table 8 indicates, top-ranked lyric words from the CW, GI-lex and Affe-lex feature types do not present much in the way of obvious semantic connections with the category "calm" (e.g., "satan"!). However, some might argue that word repetition can have a calming effect, and if this is the case, then the text stylistics features do appear to be picking up on the notion of repetition as a mechanism for instilling calmness or serenity.

6. CONCLUSIONS AND FUTURE WORK

This paper builds upon and extends our previous work on multi-modal mood classification by examining in-depth those feature types that have shown statistically significant improvements in correctly classifying individual mood categories. While derived from user-generated tags found on last.fm, the 18 mood categories used in this study fit well with Russell's mood model which is commonly used in MIR mood classification research. From our 18 mood categories we uncovered seven divergent categories where certain lyric feature types significantly outperformed audio and only one category where audio

outperformed all lyric-based features. For those seven categories where lyrics performed better than audio, the top-ranked words clearly show strong and obvious semantic connections to the categories. In two cases, simple text stylistics provided significant advantages over audio. In the one case where audio outperformed lyrics, no obvious semantic connections between terms and the category could be discerned.

We note as worthy of future study the observation that no lyric-based feature provided significant improvements in the bottom-left (negative valence, negative arousal) quadrant (Figure 2) while audio features were able to do so (i.e., "calm"). This work is limited to audio spectral features and thus we also plan on extending this work by considering other types of audio features such as rhythmic and harmonic features.

7. ACKNOWLEDGEMENT

We thank The Andrew Mellon Foundation for their financial support.

8. REFERENCES

[1] J. S. Downie and S. J. Cunningham: "Toward a Theory of Music Information Retrieval Queries: System Design Implications." In *Proceedings of the 1st International Conference on Music Information Retrieval (ISMIR'02).*

[2] C. Laurier, J. Grivolla and P. Herrera: "Multimodal Music Mood Classification Using Audio and Lyrics," In *Proceedings of the International Conference on Machine Learning and Applications*, 2008.

[3] Y.-H. Yang, Y.-C. Lin, H.-T. Cheng, I.-B. Liao, Y.-C. Ho, and H. H. Chen: "Toward multi-modal music emotion classification," In *Proceedings of Pacific Rim Conference on Multimedia (PCM'08).*

[4] X. Hu and J. S. Downie: "Improving mood classification in music digital libraries by combining lyrics and audio," In *Proceedings of Joint Conference on Digital Libraries, (JCDL2010).*

[5] L. Lu, D. Liu, and H. Zhang: "Automatic Mood Detection and Tracking of Music Audio Signals," *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1): 5-18, 2006.

[6] T. Pohle, E. Pampalk, and G. Widmer: "Evaluation of Frequently Used Audio Features for Classification of Music into Perceptual Categories," In *Proceedings of the 4th International Workshop on Content-Based Multimedia Indexing*, 2005.

[7] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas: "Multi-Label Classification of Music into Emotions," In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08).*

[8] J. S. Downie: "The Music Information Retrieval Evaluation Exchange (2005-2007): A Window into Music Information Retrieval Research," *Acoustical Science and Technology* 29 (4): 247-255, 2008. Available at: http://dx.doi.org/10.1250/ast.29.247

[9] X. Hu, J. S. Downie, C. Laurier, M. Bay, and A. Ehmann: "The 2007 MIREX Audio Music Classification Task: Lessons Learned," *Proceedings of the International Conference on Music Information Retrieval (ISMIR'08).*

[10] H. He, J. Jin, Y. Xiong, B. Chen, W. Sun, and L. Zhao: "Language Feature Mining for Music Emotion Classification via Supervised Learning From Lyrics," In *Proceedings of Advances in the 3rd International Symposium on Computation and Intelligence (ISICA'08).*

[11] Y. Hu, X. Chen, and D. Yang: "Lyric-Based Song Emotion Detection with Affective Lexicon and Fuzzy Clustering Method," In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR'09).*

[12] M. M. Bradley and P. J. Lang: "Affective Norms for English Words (ANEW): Stimuli, Instruction Manual and Affective Ratings," Technical report C-1. University of Florida, 1999.

[13] D. Yang, and W. Lee: "Disambiguating Music Emotion Using Software Agents," In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR'04).*

[14] P. J. Stone: *General Inquirer: a Computer Approach to Content Analysis.* Cambridge: M.I.T. Press, 1966.

[15] C. Strapparava and A. Valitutti: "WordNet-Affect: an Affective Extension of WordNet," In *Proceedings of the International Conference on Language Resources and Evaluation*, pp. 1083-1086, 2004.

[16] R. Mihalcea and H. Liu: "A Corpus-based Approach to Finding Happiness," In *AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW'06).*

[17] X. Hu and J. S. Downie: "Stylistics in Customer Reviews of Cultural Objects," In *Proceedings of the 2nd SIGIR Stylistics for Text Retrieval Workshop*, pp.37-42. 2006.

[18] B. Yu: "An Evaluation of Text Classification Methods for Literary Study," *Literary and Linguistic Computing*, 23(3): 327-343, 2008.

[19] J. A. Russell: "A Circumplex Model of Affect," *Journal of Personality and Social Psychology*, 39: 1161-1178, 1980.

[20] C. Fellbaum: *WordNet: An Electronic Lexical Database*, MIT Press, 1998.

[21] G. Tzanetakis: "Marsyas Submissions to MIREX 2007", available at http://www.music-ir.org/mirex/2007/abs/AI_CC_GC_MC_AS_tzanetakis.pdf

AUDIO-BASED MUSIC STRUCTURE ANALYSIS

Jouni Paulus*
Fraunhofer Institute for Integrated Circuits IIS
Erlangen, Germany
jouni.paulus@iis.fraunhofer.de

Meinard Müller
Saarland University and MPI Informatik
Saarbrücken, Germany
meinard@mpi-inf.mpg.de

Anssi Klapuri
Queen Mary Univ. of London
Centre for Digital Music
London, UK
anssi.klapuri@elec.qmul.ac.uk

ABSTRACT

Humans tend to organize perceived information into hierarchies and structures, a principle that also applies to music. Even musically untrained listeners unconsciously analyze and segment music with regard to various musical aspects, for example, identifying recurrent themes or detecting temporal boundaries between contrasting musical parts. This paper gives an overview of state-of-the-art methods for computational music structure analysis, where the general goal is to divide an audio recording into temporal segments corresponding to musical parts and to group these segments into musically meaningful categories. There are many different criteria for segmenting and structuring music audio. In particular, one can identify three conceptually different approaches, which we refer to as repetition-based, novelty-based, and homogeneity-based approaches. Furthermore, one has to account for different musical dimensions such as melody, harmony, rhythm, and timbre. In our state-of-the-art report, we address these different issues in the context of music structure analysis, while discussing and categorizing the most relevant and recent articles in this field.

1. INTRODUCTION

The difference between arbitrary sound sequences and music is not well-defined: what is random noise for someone may be ingenious musical composition for somebody else. What can be generally agreed upon is that it is the structure, or the relationships between the sound events that create musical meaning. This structure starts from the level of individual notes, their timbral characteristics and pitch and time intervals. Notes form larger structures, phrases, chords, and chord progressions, and these again form larger constructs in a hierarchical manner. At the level of entire musical pieces the subdivision can be made to musical sections, such as *intro*, *chorus*, and *verse* in popular music. Recovering a description of this structure, often referred to as *musical form*, is what is here meant by *music structure analysis*. In this paper, we mainly focus on Western popular music in terms of the musical structures and acoustic assumptions we make, even though many of the employed principles can be utilized to analyze other kinds of music as well. For a tutorial and a review of earlier methods for music structure analysis, we refer to the book chapter by Dannenberg and Goto [16]. Our objective is to give an updated overview on this important topic by discussing a number of new trends and recent research articles. Computational analysis of the structure of recorded music constitutes a very active research field within the area of music information retrieval. Here we focus on music structure analysis at the largest temporal scale, and assume that the musical form can be expressed as a sequence of musically meaningful parts at this level. [1] The musical form is of great importance for both understanding as well as processing music and is often characteristic to the particular genre.

Structure in music signals arises from certain relationships between the elements—notes, chords, and so forth—that make up the music. The principles used to create such relationships include *temporal order*, *repetition*, *contrast*, *variation*, and *homogeneity*. Obviously, the *temporal order* of events, as also emphasized by Casey and Slaney [11], is of crucial importance for building up musically and perceptually meaningful entities such as melodies or harmonic progressions. Also, the principle of *repetition* is central to music, as Middleton [51] states: *"It has often been observed that repetition plays a particularly important role in music—in virtually any sort of music one can think of, actually. [. . .] In most popular music, repetition processes are especially strong."* Recurrent patterns, which may be of rhythmic, harmonic, or melodic nature, evoke in the listener the feeling of familiarity and understanding of the music. The principle of *contrast* is introduced by having two successive musical parts of different character. For example, a quiet passage may be contrasted by a loud one, a slow section by a rapid one, or an orchestral part by a solo. A further principle is that of *variation*, where motives and parts are picked up again in a modified or transformed

*This work was performed when the author was at the Department of Signal Processing, Tampere University of Technology, Tampere, Finland.

[1] One of the few methods aiming at a hierarchical description of the structure at various time scales is the approximate string matching method by Rhodes and Casey [70].

form [39]. Finally, a section is often characterized by some sort of inherent *homogeneity*, for example, the instrumentation, the tempo, or the harmonic material being similar within the section.

In view of the various principles that crucially influence the musical structure, a large number of different approaches to music structure analysis have been developed. One can roughly distinguish three different classes of methods. Firstly, *repetition-based* methods are employed to identify recurring patterns. From a technical point of view, these methods are also often referred to as *sequence approaches*, see also Sec. 5. Secondly, *novelty-based* methods are used to detect transitions between contrasting parts. Thirdly, *homogeneity-based* methods are used to determine passages that are consistent with respect to some musical property. Note that novelty-based and homogeneity-based approaches are two sides of a coin: novelty detection is based on observing some surprising event or change after a more homogenous segment. From a technical point of view, the homogeneity-based approach has often been referred to as *state approach*, see also Sec. 5. Finally, in all the method categories, one has to account for different musical dimensions, such as melody, harmony, rhythm, or timbre. To this end, various feature representations have been suggested in the literature.

The remainder of this paper is organized as follows. In Sec. 2, we approach the structure analysis task from different angles and give a problem definition used in this paper. In Sec. 3, we discuss feature representations that account for different musical dimensions. In Sec. 4, we introduce the concept of a self-distance matrix often used in music structure analysis, and show how the various segmentation principles are reflected in this matrix. Then, in Sec. 5, we discuss the principles of repetition-based, novelty-based, and homogeneity-based structure analysis methods. Here, we also discuss and categorize the most relevant and recent articles in this field. In Sec. 6, we address the issue of evaluating analysis results, which in itself constitutes a non-trivial problem. Finally, in Sec. 7, we conclude with a discussion of open problems.

2. PROBLEM SPECIFICATION

As mentioned before, the task of *music structure analysis* refers to a range of problems, and different researchers have pursued slightly different goals in this context. A common theme, however, is that the temporal scale of the analysis has been approximately the same in all the cases. In the rest of the paper, we use the following terminology. A *part* is understood to be a musical concept that loosely refers to either a single instance or all the instances of a musical section, such as chorus or verse, whereas a *segment* is understood to be a technical concept that refers to the temporal range of a single occurrence of a musical part. The term *group* is used to denote one or more segments that represent all the occurrences of the same musical part.

The methods discussed in the following take an acoustic music signal as the input and produce some information about the structure. The output of the discussed methods varies from images created for visualization purposes to representations that specify the time range and musically meaningful label of each found part. In the simplest form, no explicit structural analysis is performed, but some transformation of the acoustic features of the piece are used to yield a visual representation of structural information, e.g., the self-similarity matrix visualization by Foote [24].

The next category of methods aim to specify points within a given audio recording where a human listener would recognize a change in instrumentation or some other characteristics. This problem, which is often referred to as *novelty detection*, constitutes an important subtask [25]. For example, as we explain later, having computed novelty points in a preprocessing step may significantly speed up further structure analysis [62].

Another and yet more complex task level involves grouping the sections that represent the same underlying musical part: sections that can be seen as repetitions of each other [59, 64, 56]. Finding and grouping all repeated sections provides already a fairly complete description of the musical form, by considering the non-repeated segments as separate and mutually unrelated parts.

Some structure analysis methods have been motivated by finding only one representative section for a piece of music, a "thumbnail" that provides a compact preview of the piece [31, 8, 23, 64]. For this purpose, the most often repeating section is typically suitable.

In this paper, we focus on the structure analysis problem where the objective is to determine a description that is close to the musical form of the underlying piece of music. Here, the description consists of a *segmentation* of the audio recording as well as of a *grouping* of the segments that are occurrences of the same musical part. The groups are often specified by letters $A, B, C, \ldots$ in the order of their first occurrence. Since some of the musical parts have distinct "roles" in Western music, some methods aim to automatically assign the groups with *labels*, such as *verse* or *chorus* [61].

3. FEATURE REPRESENTATION

Since the sampled waveform of an acoustic signal is relatively uninformative by itself, some feature extraction has to be employed. The first question to be addressed concerns the acoustic and musical features that humans observe when determining the musical form of a piece. Bruderer et al. [10] conducted experiments to find the perceptual cues that humans use to determine segmentation points in music. The results suggest that "global structure," "change in timbre," "change in level," "repetition," and "change in rhythm" indicated the presence of a structural boundary to the test subjects. We now summarize how some of these aspects can be accounted for by transforming the music signal into suitable feature representations.

3.1 Frame Blocking for Feature Extraction

The feature extraction in audio content analysis is normally done in relatively short, 10-100 ms frames. In music struc-

ture analysis each frame of a piece is usually compared to all other frames, which can be computationally intensive. Many of the proposed methods employ a larger frame length in the order of 0.1-1 s. Not only does this reduce the amount of data, but it also allows focusing on a musically more meaningful time scale [63]. The importance of the temporal resolution of feature extraction on the final structure analysis results has been emphasized in [52, 62].

The idea of a musically meaningful time scale has been taken even further in some methods that propose the use of event-synchronized feature extraction. In other words, instead of a fixed frame length and hop size, the division is defined by the temporal locations of sound events [36] or the occurrences of a metrical pulse, e.g., tatum or beat [47, 72, 42, 48, 14, 59]. Using a signal-adaptive frame division has two benefits compared to the use of a fixed frame length: tempo-invariance and sharper feature differences. Tempo-invariance is achieved by adjusting the frame rate according to the local tempo of the piece, which facilitates the comparison of parts performed in different tempi. Event-synchronized frame blocking also allocates consecutive sound events to different frames, which prevents them from blurring each others' acoustic features. In practice, one often calculates the features in short frames and then averages the values over the length of the event-synchronized frames [23, 60, 62, 50].

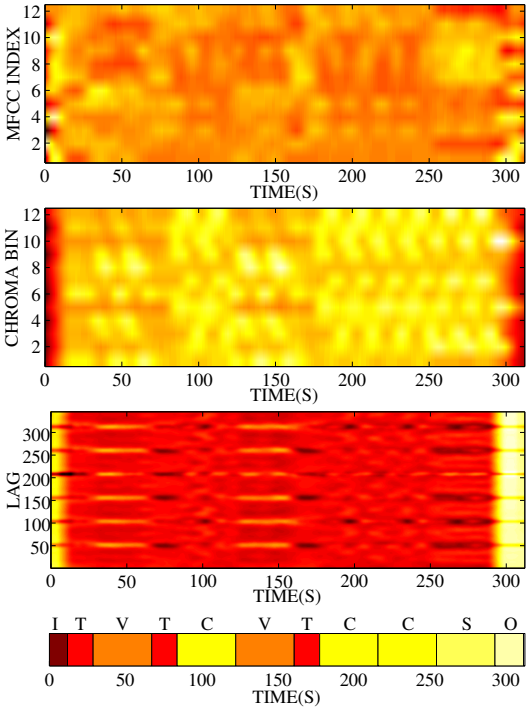


Figure 1: Acoustic features extracted from the piece "Tuonelan koivut" by Kotiteollisuus. The three feature matrices correspond to MFCCs (first panel), chroma (second panel), and rhythmogram (third panel). The annotated structure of the piece is given at the bottom panel, and the parts are indicated with: intro (I), theme (T), verse (V), chorus (C), solo (S), and outro (O).

3.2 Features

The instrumentation and timbral characteristics are of great importance for the human perception of music structure [10]. Perceptually, timbre is closely related to the recognition of sound sources and depends on the relative levels of the sound at critical bands as well as their temporal evolution. Therefore, a majority of the timbre-based structure analysis methods use *mel-frequency cepstral coefficients* (MFCCs), which parametrize the rough shape of the spectral envelope and thus encode timbral properties of the signal [18]. MFCCs are obtained by discrete cosine transforming (DCT) log-power spectrum on the mel-frequency scale:

$$\mathrm{MFCC}(k) = \sum_{b=0}^{N-1} E(b) \cos\left(\frac{\pi(2b+1)k}{2N}\right), \quad (1)$$

where the subbands b are uniformly distributed on the mel-frequency scale and $E(b)$ is the energy of band b. A generally accepted observation is that the lower MFCCs are closely related to the aspect of timbre [3, 74].

As an alternative to using MFCCs as a timbre parametrization, Maddage proposed replacing the mel-spaced filter bank with 4-12 triangular filters in each octave for the task [46]. Other parametrisations omit the DCT step and use some non-mel spacing in band definitions. For example, the MPEG-7 *AudioSpectrumEnvelope* descriptor [35] has been used [78, 41], or very similar constant-Q spectrograms [2, 11]. Aucouturier and Sandler [5] compared different parametrisations of timbral information in music structure analysis and found MFCCs to outperform other features such as linear prediction coefficients. MFCCs calculated from an example piece are illustrated in the top panel of Fig. 1.

Another important aspect of music is its pitched content on which harmonic and melodic sequences are built upon. In the context of music structure analysis, *chroma features* or *pitch class profiles* have turned out to be a powerful mid-level representation for describing harmonic content [8, 29, 52, 13, 48]. Assuming the equal-tempered scale, the chroma correspond to the set $\{\mathrm{C}, \mathrm{C}^\sharp, \mathrm{D}, \ldots, \mathrm{B}\}$ that contains the twelve pitch classes used in Western music notation. A normalized chroma vector describes how the signal's spectral energy is distributed among the 12 pitch classes (ignoring octave information), see Fig. 1 for an illustration.

Several methods for calculating chroma-based audio features have been proposed. Most approaches first compute a discrete Fourier transform (DFT) and then suitably pool the DFT coefficients into chroma bins [8, 29, 31]. Müller et al. [52, 56] propose to use a multirate filter bank consisting of time-domain band-pass filters that correspond to the semitone bands before the chroma projection. Ryynänen and Klapuri replace the DFT analysis by a multipitch estimation front-end [71]. Other chroma-like features are compared in a music structure analysis application by Ong et al. in [58]. Recently, Müller et al. [54] proposed a method to increase the timbre-robustness of chroma by removing some information correlating with the timbre before the octave folding. Some timbre-robustness

is also achieved by the spectral whitening as described in [71]. For an overview of other variants of chroma and pitch-based features, see Müller [52] and Gómez [29].

In contrast to timbral and harmonic content, there has been comparatively little effort in exploiting beat, tempo, and rhythmic information for music structure analysis. To extract such information from audio recordings, most approaches proceed in two steps. In the first step, a detection function, here called *onset accent curve*, is calculated, where high values correlate with the positions of note onsets in the music. The calculation typically relies on the fact that note onsets tend to cause a sudden change of the signal energy and spectrum [9, 80]. In the second step, the accent curves are analyzed with respect to quasiperiodic patterns. Important for the analysis is to obtain a shift-invariant representation that is immune to the exact temporal position of the pattern. Autocorrelation-based analysis allows for detecting periodic self-similarities by comparing an accent curve with time-shifted copies itself [19, 22, 65]. Alternatively, one can use a short-time Fourier transform and then omit the phase in order to derive a shift-invariant representation of the accent curve [65, 32]. Both methods reveal rhythmic properties, such as the tempo or beat structure. These properties typically change over time and are therefore often visualized by means of spectrogram-like representations referred to as *tempogram* [12], *rhythmogram* [38], or *beat spectrogram* [26].

Rhythmic features have not been used in music structure analysis very widely. For example, Jehan [36] used loudness curves, and Jensen [37, 38] included rhythmograms [2] for the structure analysis task. Paulus and Klapuri noted in [60] that the use of rhythmic information in addition to timbral and harmonic features provides useful information to structure analysis, see also Fig. 1. Finally, Peeters [63] has introduced *dynamic features* that aim to parametrize the rhythmic content by describing the temporal evolution of features.

Even though different features describe different musical properties, to date very few methods have utilized more than one feature at a time (except the methods with a large number of more simple features combined with feature vector concatenation [79, 57]). In some approaches MFCC and chroma features have been used to define a single, overlaid self-distance matrix [23, 64], see also Sec. 4. Levy et al. [40] combined information from timbral and harmony related features by feature vector concatenation. A similar approach was adopted by Cheng et al. [14]. Paulus and Klapuri [62] combine the information obtained from MFCCs, chroma features, and rhythmograms using a probabilistic framework.

[2] Recently, Grosche et al. [33] suggested a cyclic variant of a tempogram, which may be a low-dimensional alternative in the structure analysis context. Similar to the concept of cyclic chroma features, where pitches differing by octaves are identified, the cyclic tempogram is obtained by identifying tempi that differ by a power of two.

4. SELF-DISTANCE MATRIX

As the musical structure is strongly implied by repetition, a useful strategy is to compare each point of a given audio recording with all the other points, in order to detect self-similarities. The general idea is to convert a given audio recording into a suitable feature sequence, say $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, and then to compare all elements of the sequence with each other in a pairwise fashion. More precisely, given a distance function d that specifies the distance between two feature vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, it is possible to compute a square *self-distance matrix* (SDM) $D(i,j) = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $i, j \in \{1, 2, \ldots, N\}$. Frequently used distance measures include the Euclidean distance $d_{\mathrm{E}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \| \boldsymbol{x}_i - \boldsymbol{x}_j \|$, and the cosine distance

$$d_{\mathrm{C}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0.5 \left(1 - \frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle}{\| \boldsymbol{x}_i \| \| \boldsymbol{x}_j \|} \right), \qquad (2)$$

where $\| \cdot \|$ denotes vector norm and $\langle \cdot, \cdot \rangle$ dot product. If the distance measure d is symmetric, i.e., $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$, the resulting SDM is also symmetric along the main diagonal.

The origins of an SDM representation stems from recurrence plots proposed by Eckmann et al. [21] for the analysis of chaotic systems. The concept of a self-distance matrix [3] has been introduced to the music domain by Foote [24] in order to visualize the time structure of a given audio recording. Naturally, the properties of an SDM crucially depend on the chosen distance measure and the feature representation.

The distance measures are usually defined to compare single frames. Often, it is beneficial to also include the local temporal evolution of the features in order to enhance the structural properties of an SDM. To this end, Foote [24] proposed to average the distance values from a number of consecutive frames and to use that as the distance value. This results in a smoothing effect of the SDM. Müller and Kurth [55] extended these ideas by suggesting a contextual distance measure that allows for handling local tempo variations in the underlying audio recording. Instead of using sliding windows of several consecutive frames, other approaches calculate the average distance from the feature vectors within non-overlapping musically meaningful segments such as musical measure [72, 59]. Jehan [36] calculated SDMs at multiple levels of a temporal hierarchy, starting from individual frames to musical patterns. Each higher level in the hierarchy was calculated based on the SDM of the finer temporal structure.

Recurring patterns in the feature vector sequence $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ are visible in the SDM. The two most important patterns induced by the feature patterns are illustrated in an idealized SDM in Fig. 2. If the features capture musical properties (e.g., instrumentation) that stay somewhat constant over the duration of a musical part, *blocks* of low distance are formed. In case the features describe sequential properties instead of remaining constant within a

[3] The dual of SDMs are self-similarity matrices in which each element describes the *similarity* between the frames instead of distance. Most of the following operations can be done with either representation, although here we discuss only SDMs.

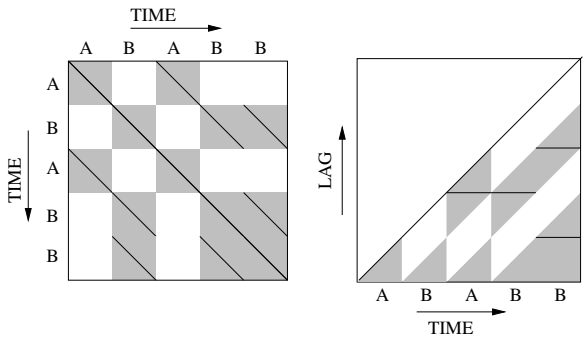


Figure 2: **Left:** An example of the patterns formed in SDMs. The sequence consists of two parts, A and B, repeating as indicated, and darker element denotes lower distance. **Right:** Corresponding time-lag matrix of the SDM The non-main diagonal stripes will be transformed into horizontal lines with the vertical position describing the interval (lag) between the occurrences.

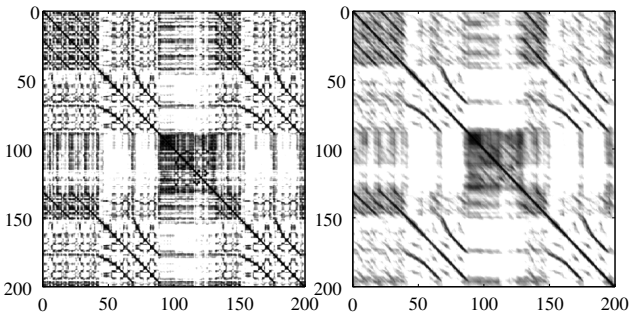


Figure 3: **Left:** Self-distance matrix of a piece with tempo variations. **Right:** Path-enhanced version. Darker pixels denote lower distances. Note that some of the stripes are curved expressing relative tempo differences in the repeating parts.

part, diagonal *stripes* of low distance are formed. If such a part is repeated, one finds stripes in the SDM that run parallel to the main diagonal. This is often the case when using chroma features, which then reveal repeated harmonic progressions within a piece. Locating and interpreting these patterns with various methods is the main approach employed in many of the structure analysis methods described in the literature.

As Peeters [63] noted, the features alone do not determine whether blocks or stripes are formed, but the temporal parameters of the feature extraction process are also important. In other words, the longer the temporal window is that the feature vector describes, the more likely it is that blocks are formed in the SDM. Therefore, working with low resolutions may not only be beneficial for computational, but also for structural reasons [56, 60]. The effect of the time scale parameter used in the feature computation on the resulting SDMs is also illustrated by Fig. 4.

Often a musical part is repeated in another key. Using chroma features, Goto [31] simulates transpositions by cyclically shifting the chroma. Adopting this idea, Müller and Clausen [53] introduced the concept of transposition-invariant SDMs, which reveals the repetitive structure even in the presence of key transpositions.

Another way to present repetitive information is to transform an SDM into a time-lag format [31]. In an SDM

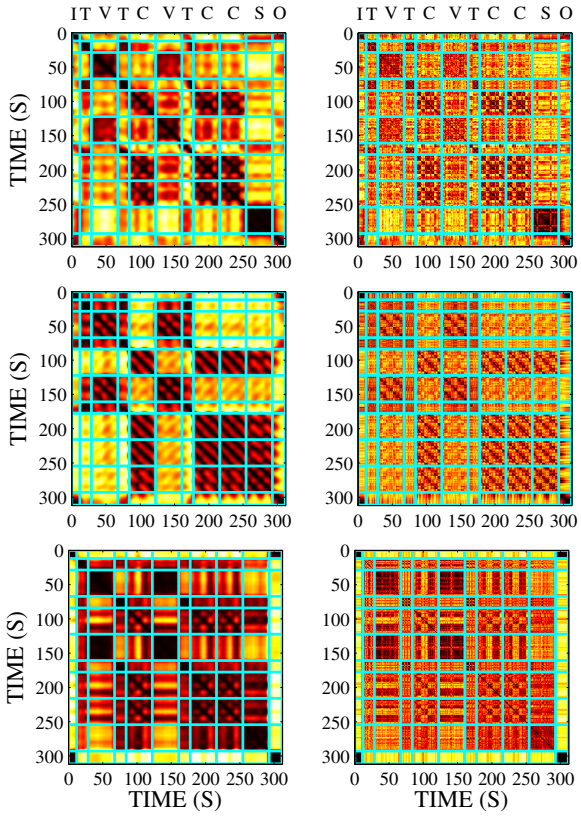


Figure 4: Example SDMs from features of Fig. 1 at a coarse (**Left**) and fine (**Right**) time scale. **Top:** MFCCs. **Middle:** Chroma features. **Bottom:** Rhythmogram. Darker pixels denote lower distances. The annotated structure of the piece is indicated by the overlay grid, and the part labels are indicated in the top of the figure with: intro (I), theme (T), verse (V), chorus (C), solo (S), and outro (O). The figure shows how different parts share some of the perceptual aspects, but not all, e.g., chorus and solo have similar harmonic but differring timbral content.

D both the axes represent absolute time, whereas in the time-lag matrix ***R*** one axis is changed to represent time difference (lag) instead

$$R(i, i-j) = D(i,j), \quad \text{for } i-j > 0. \tag{3}$$

The ordinate transformation discards the duplicate information of a symmetric SDM, see Fig. 2. The diagonal stripes formed by repeated sequences appear as horizontal lines in the time-lag representation, and may be easier to extract. Even though a time-lag representation transforms the stripe information into a more easily interpretable form, the block information is transformed into parallelograms and may now be more difficult to extract. Furthermore, the time-lag representation only works when repeating parts occur in the same tempo, which is, in particular for classical music, often not the case. Structure analysis in the presence of temporal variations is discussed in [52, 56], see also Fig. 3 for an illustration.

5. STRUCTURE ANALYSIS APPROACHES

As mentioned before, there are a variety of different methods proposed for music structure analysis. An overview of the operational entities of the proposed methods is shown

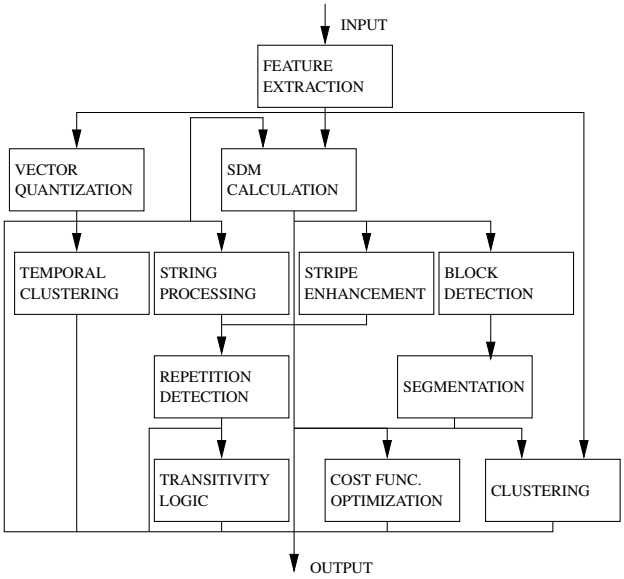


Figure 5: An overview block diagram of various operational entities employed in music structure analysis methods.

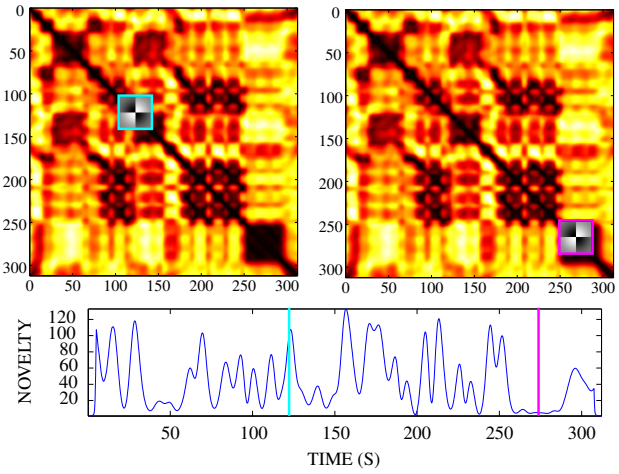


Figure 6: **Top:** Two instances of the SDM using MFCCs from Fig. 4. The checkerboard-like kernel that is correlated along the main diagonal is shown at two different positions on the left and right. **Bottom:** Resulting novelty curve.

in Fig. 5. Furthermore, relevant literature along with a classification of the involved methods is summarized by Table 1. In this section, we describe the main approaches as well as the interconnections between the operational entities in more detail. The first categorization of music structure analysis methods was proposed by Peeters [63] dividing them into *sequence* and *state* approaches. The *sequence approaches* assume that there are sequences of events that are repeated several times in the given musical signal, thus forming diagonal stripes in the corresponding SDM. The *state approaches* in turn consider the piece to be produced by a finite state machine, where each state produces some part of the signal. Considering the SDM representation, the state approaches can be thought to form the blocks [4] . As mentioned in Sec. 1, we use the more semantically motivated term *repetition-based approach* instead of the more technically motivated term *sequence approach*. Similarly, we use the term *homogeneity-based approach* instead of the term *state approach*. Furthermore, we add a third category referred to as *novelty-based approach*. In the following, we describe some instantiations of each of the categories in more detail and then discuss some combined approaches.

5.1 Novelty-based Approaches

An important principle in music is that of change and contrast introducing diversity and attracting the attention of a listener. The goal of *novelty-based* procedures is to automatically locate the points where these changes occur. A standard approach for *novelty detection* introduced by Foote [25] tries to identify segment boundaries by detecting 2D corner points in an SDM of size $N \times N$ using a kernel matrix of a lower dimension. The kernel consists of an $M \times M$ matrix (with $M < N$) which has a 2×2 checkerboard-like structure and is possibly weighted by a Gaussian radial function. The kernel is illustrated within the small rectangles on top of the two SDMs in Fig. 6. The kernel is then correlated along the main diagonal of the SDM. This yields a *novelty function*, the peaks of which indicate corners of blocks of low distance. Using MFCCs, these peaks are good indicators for changes in timbre or instrumentation. For an illustration, we refer to Fig. 6. Similarly, using other feature representation such as chroma features or rhythmograms, one obtains indicators for changes in harmony, rhythm, or tempo.

Jensen uses a different approach for locating the main diagonal blocks in an SDM [38] by formulating the segmentation as an optimization problem. The cost function to be optimized tries to minimize the average distance within blocks (defined by neighboring segment boundaries) of the SDM while keeping the number of segments small. Tzanetakis and Cook [76] propose to segment a signal by first extracting a set of features from the signal and then calculating a Mahalanobis distance between successive frames. Large differences in the distance values indicate possible segmentation points. For other methods to music segmentation, we refer to the publication by Turnbull et al. [75], in which several acoustic features and both supervised as well as unsupervised segmentation methods are evaluated.

5.2 Homogeneity-based Approaches

A direct continuation of the *novelty-based* procedure is to analyze the content of the created segments and to classify them building up *homogenous* clusters. Such an approach was introduced by Cooper and Foote in [15], where, after a novelty-based segmentation, the content of each segment is modeled by a normal distribution. Then, the similarity between two segments is computed using the Kullback-Leibler divergence between two multivariate normal distributions [28]. Having the distances for all segment pairs,

[4] In principle a state is capable of emitting also a feature sequence forming stripes in SDM when repeated. However, the name "state approach" is more often used of methods that utilize principles of homogeneity.

Author / publication	Task	Acoustic features	Approach	Method
Aucouturier et al. [4]	full structure	spectral envelope	homogeneity	HMM
Barrington et al. [7]	full structure	MFCC / chroma	homogeneity	dynamic texture model
Bartsch & Wakefield [8]	thumbnailing	chroma	repetition	stripe detection
Chai [13]	full structure	chroma	repetition	stripe detection
Cooper & Foote [15]	summarisation	magnitude spectrum	homogeneity	segment clustering
Dannenberg & Hu [17]	repetitions	chroma	repetition	dynamic programming
Eronen [23]	chorus detection	MFCC+chroma	repetition	stripe detection
Foote [24]	visualization	MFCC		self-similarity matrix
Foote [25]	segmentation	MFCC	novelty	novelty vector
Goto [31]	repetitions	chroma	repetition	stripe detection (*RefraiD*)
Jehan [36]	pattern learning	MFCC+chroma+loudness	homogeneity	hierarchical SDMs
Jensen [38]	segmentation	MFCC+chroma+rhythmogram	novelty	diagonal blocks
Levy & Sandler [41]	full structure	MPEG-7 timbre descriptor	homogeneity	temporal clustering
Logan & Chu [43]	key phrase	MFCC	homogeneity	HMM / clustering
Lu et al. [44]	thumbnailing	constant-Q spectrum	repetition	stripe detection
Maddage [46]	full structure	chroma	homogeneity	rule-based reasoning
Marolt [48]	thumbnailing	chroma	repetition	RefraiD
Mauch et al. [50]	full structure	chroma	repetition	greedy selection
Müller & Kurth [56]	multiple repetitions	chroma statistics	repetition	stripe search & clustering
Ong [57]	full structure	multiple	repetition	RefraiD
Paulus & Klapuri [59]	repeated parts	MFCC+chroma	repetition	cost function
Paulus & Klapuri [62]	full description	MFCC+chroma+rhythmogram	combined	fitness function
Peeters [63]	full structure	dynamic features	homogeneity	HMM, image filtering
Peeters [64]	repeated parts	MFCC+chroma+spec. contrast	repetition	stripe detection
Rhodes & Casey [70]	hierarchical structure	timbral features	repetition	string matching
Shiu et al. [72]	full structure	chroma	repetition	state model stripe detection
Turnbull et al. [75]	segmentation	various	novelty	various
Wellhausen & Höynck [78]	thumbnailing	MPEG-7 timbre descriptor	repetition	stripe detection

Table 1: A summary of discussed methods for music structure analysis.

the segments are grouped with spectral clustering [77]. Logan and Chu [43] used a similar Gaussian parametrization on segments of fixed length and applied agglomerative hierarchical clustering. The method proposed by Goodwin and Laroche [30] performs the segmentation and clustering at the same time. The method itself resembles the optimization procedure described by Jensen [38], with the difference that the searched path can now return to a state defined earlier if it is globally more efficient for the structure description.

The concept of *state* is taken more explicitly in methods employing hidden Markov models (HMMs) for the analysis, see, e.g., [5, 27]. Here, the basic assumption is that each musical part can be represented by a state in an HMM, and the states produce observations from the underlying probability distribution. In an HMM, the probability of a state sequence $\mathbf{q} = (q_1, q_2, \ldots, q_N)$ given the observation sequence $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ can be calculated by

$$P(\mathbf{q}|\mathbf{X}) \propto P(\boldsymbol{x}_1|q_1) \prod_{n=2}^{N} P(\boldsymbol{x}_n|q_n) p(q_n|q_{n-1}), \qquad (4)$$

where $P(\boldsymbol{x}_n|q_n)$ is the likelihood of observing $\boldsymbol{x}_n$ if the state is q_n, and $p(q_n|q_{n-1})$ is the transition probability from state q_{n-1} to state q_n. The analysis operates by training the HMM with the piece to be analyzed, and then by decoding (finding the most probable state sequence) the same signal with the model. Effectively this implements vector quantization of the feature vectors with some temporal dependency modeling expressed by the state transition probabilities. Though this model has a certain appeal, it does not work very well in practice because the result is often temporally fragmented, as noted by Peeters et al. [68]. The fragmentation is due to the fact that the individual states tend to model individual sound events rather than longer musical parts.

To alleviate the problem of temporal fragmentation, several post-processing methods have been proposed. Here, the state sequence produced by an HMM is only used as a mid-level representation for further analysis, where each state represents a certain context-dependent short sound event [41]. Fig. 7 shows the resulting state sequences of an example piece after analyzing it with fully connected HMMs with 8 and 40 states, respectively. The state sequence representation is included also for general audio parametrization in the MPEG-7 standard as the *SoundModelStatePathType* descriptor [35]. Abdallah et al. [1] proposed to calculate histograms of the states with a sliding window over the entire sequence and then to use the resulting histogram vectors as new feature representation. Based on these state histograms, probabilistic clustering is applied. This method was extended to include statistical modeling of the cluster durations [2]. Levy et al. [42] increased the amount of the contextual knowledge using a variant of a fuzzy clustering approach applied on the histograms. This approach was formalized by Levy and Sandler [41] using a probabilistic framework. Despite the relatively simple approach, the temporal clustering method [42] has proven to work quite well.

A slightly different approach to reduce the resulting fragmentation was proposed by Peeters [68]. He performed initial segmentation based on an SDM and then used the average feature value over each individual segment as initial cluster centroids that he further updated using k-means clustering. The obtained cluster centroids were then used to initialize the training of an HMM which produced the final clustering. In a recent publication Barrington et al. [7] propose to use dynamic texture mixture

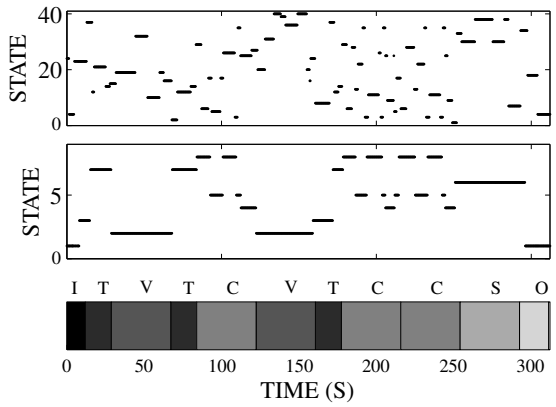


Figure 7: State sequences resulting from a fully connected HMM using 40 (**Top**) and 8 (**Middle**) states applied to the MFCC feature sequence of Fig. 1. The bottom panel shows the annotated ground truth structure.

models (DTM) for the structure analysis. DTM is basically a state model, where each (hidden) state produces observations that have a temporal structure. The main novelty of the method compared to the HMM-based state methods is that the observation model itself takes the temporal behavior of the produced observations into account, and there will be less need for heuristic post-processing.

5.3 Repetition-based Approaches

The repetition of musical entities, as already noted in Sec. 1, is an important element in imposing structure on a sequence of musical sounds. Here, the temporal order in which the sound events occur is crucial to form musically meaningful entities such as melodies or chord progressions. Therefore, the task of extracting the repetitive structure of a given audio recording of a piece of music amounts to first transform the audio into a suitable feature sequence and then to find repeating subsequences in it.

As was explained in Sec. 4, one possible approach is to compute an SDM and to search for diagonal stripes parallel to the main diagonal. Even though it is often easy for humans to recognize these stripes, the automated extraction of such stripes constitutes a difficult problem due to significant distortions that are caused by variations in parameters such as dynamics, timbre, execution of note groups (e.g., grace notes, trills, arpeggios), modulation, articulation, or tempo progression [56, 52]. To enhance the stripe structure, many approaches apply some sort of low-pass filtering to smooth the SDM along the diagonals [78, 8]. A similar effect can be achieved by averaging the distance values from a number of consecutive frames and to use that as the distance value [24]. Marolt [48] proposed to enhance the stripes by calculating multiple SDMs with different sliding window lengths and then by combining them with element-wise multiplication. Lu et al. [44] employed multiple iterations of erosion and dilation filtering along the diagonals to enhance the stripes by filling small breaks and removing too short line segments. Ong [57] extended the erosion and dilation filtering into two-dimensional filter to enhance the entire SDM. Goto [31] employed a two-dimensional local filter to enhance the stripes; similar enhancement was later utilized by Eronen [23]. Peeters [64] proposed to low-pass filter along the diagonal direction, and high-pass filter along the anti-diagonal direction to enhance the stripes.

Most of the above approaches assume that the repeating parts are played in the same tempo, resulting in stripes that run exactly in parallel to the main diagonal. However, this assumption may not hold in general. For example, in classical music there are many recordings where certain parts are repeated in different tempi or where significant tempo changes (e.g. riterdando, accelerando, rubato) are realized differently in repeating parts. Here, the stripes may be even curved paths as indicate by Fig. 3. Müller et al. [55, 52] introduced smoothing techniques that can handle such situations by incorporating contextual information at various tempo levels into a single distance measure.

After enhancing the stripe structure, the stripe segments can be found, e.g., by thresholding. The *RefraiD* approach proposed by Goto [31] has later been employed by several studies [48, 57]. It uses the time-lag version of SDM to select the lags that are more likely to contain repeats, and then detect the line segments along the horizontal direction of the lags. Each of the found stripes specifies two occurrences of a sequence: the original one and a repeat. For chorus detection, or simple one-clip thumbnailing, selecting a sequence that has been repeated most often has proven to be an effective approach. In the case that a more comprehensive structural description is wanted, multiple stripes have to be detected as well as some logical reasoning to deduce the underlying structure as proposed by Dannenberg [17].

Similar to the dynamic programming approaches used for segmentation [30, 38], some of the stripes can be found by a path search. Shiu et al. [73] interpret the self-similarity values as probabilities and define a local transition cost to prefer diagonal movement. Then, Viterbi search is employed to locate the optimal path through the lower (or upper) triangle of the SDM. The stripes have large similarity values, thus the probability values are also large and the path is likely to go through the stripe locations. Another method to locate stripe segments by growing them in a greedy manner was proposed by Müller and Kurth [56]. These approaches are advantageous in that they are able to handle tempo differences in the repeats.

Rhodes and Casey [70] employed a string matching method to the HMM state sequence representation to create a hierarchical description of the structure. Though the algorithm was presented to operate on a finite alphabet formed by the HMM states, the authors suggest that similar operations could be accomplished with feature vectors after modifying the matching algorithm to accept vector inputs. Aucouturier and Sandler [6] proposed another method for inspecting the HMM state sequences with image processing methods. The main idea is to calculate a binary co-occurrence matrix (resembling an SDM) based on the state sequence, which elements have the value 1, if the two frames have the same state assignment, and the value 0 otherwise. Then a diagonal smoothing kernel is

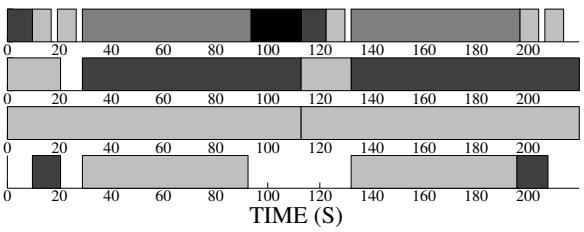


Figure 8: Effect of differently weighting the terms in the cost function of [59] on the final structure description. **Top:** Annotated ground truth. **Second row:** Analysis result with some reasonable values for the weights. **Third row:** Result with increased weight of the *complexity* term. **Bottom:** Result with a decreased weight for the term *amount unexplained*.

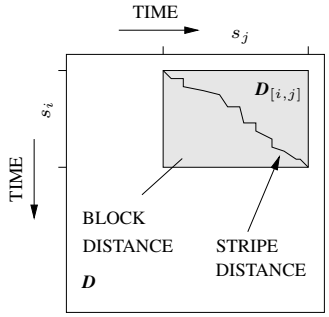


Figure 9: Illustration of the basic ideas behind the *stripe* and *block* distances between two segments s_i and s_j of a piece. The stripe distance is based on the path of least cost through the submatrix $\boldsymbol{D}_{[i,j]}$ while the block distance is based on the average distance value within the submatrix.

applied on the matrix to smooth out small mismatches between sequences. Finally, stripes are searched from the resulting matrix with Hough transform, which is claimed to be relatively robust against bad or missing data points.

5.4 Combined Approaches

Most methods for music structure analysis described so far rely on a single strategy. For example, homogeneity-based approaches try to locate blocks of low distance on the SDM main diagonal and then to classify them. Or, repetition-based approaches try to extract stripes from the SDM and then to deduce the repetitive structure. An alternative approach is to focus on modeling the properties of a good structural description, and in doing so, to combine different segmentation principles. This is the idea of Paulus and Klapuri [59, 62], who proposed a cost function for structural descriptions of a piece that considers all the desired properties, and then, for a given acoustic input, minimized the cost function over all possible structural descriptions. A similar approach was also suggested by Peiszer [69]. In [59], the cost function included terms representing the *within-group dissimilarity* (repeats should be similar), the *amount unexplained* (the structural description would cover as much of the piece as possible), and the *complexity* (the structure should not be fragmented). The effect of the balancing of these three terms is illustrated in Fig. 8.

The main weakness of the cost function based method described above—as well as with most of the other methods relying on locating individual stripes or blocks in the SDM—is that they operate only on parts of the SDM. In other words, when locating stripes, each of the stripes is handled separately without any contextual information. Considering structure analysis as a data clustering problem, each of the formed clusters should be compact (having small within-group distances), and the clusters should be well-separated (having large between-group distances). Paulus and Klapuri [62] formalized these ideas using a probabilistic framework. Here, replacing the cost function, a *fitness measure* is defined for jointly measuring within-group distance (which should be small) and between-group distance (which should be large). To this end, for each segment pair, two distances were calculated: a *stripe distance* that measures the distance of the feature sequences corresponding to the two segments (using dynamic time warping) and a *block distance* that measures the average distance over all frame pairs of the two segments, see also Fig. 9. Maximizing the fitness measure then resulted in a reasonable trade-off between these two types of complementary information. Multiple feature representations (MFCCs, chroma features, rhythmogram) were integrated into the fitness measure to account for the various musical dimensions, see Sec. 3.

In [62], the combinatorial optimization task over all descriptions was approximately solved by limiting the set of possible segments. To this end, a set of candidate segmentation points was created using a novelty-based method [25], and then a greedy algorithm over the remaining search space was applied. As a result, the method combines all the segmentation principles discussed in Sec. 5: a *novelty-based approach* was used to reduce the number segment candidates, and *homogeneity-based* and *repetition-based* approaches were integrated in the fitness measure. One drawback of the described approach is that the final structure description crucially depends on the first novelty detection step, which was found to be a bottle-neck in some cases.

6. EVALUATION

Music is multi-faceted and complex. Even though it is structured and obeys some general rules, music also lives from expanding and even breaking these rules. Therefore it can be problematic to give a concise and unique structural description for a piece of music. As a consequence, evaluating the performance of an automated structure analysis method is not as simple as it may initially seem. We now briefly discuss some of the evaluation metrics proposed in the literature.

To evaluate the accuracy of segmentation boundaries, most evaluation procedures involve some sort of recall rate, precision rate, and F-measure while accepting a small temporal deviation [75]. An alternative is to calculate the mean (or median) time between a claimed and annotated segmentation point [75]. The evaluation of music thumbnailing requires user studies, since the quality of the output is usually measured subjectively instead of an objective met-

ric, as described by Chai [13] and Ong [57].

Evaluating the result of a method producing a description of the full structure of a piece is less straightforward. Many of the evaluation metrics adopt an approach similar to evaluating clustering results: pairs of frames are inspected, and if they belong to any occurrence of the same musical part, they are considered to belong to the same cluster, denoted by the set $\mathbb{F}_A$ in case of ground truth and the set $\mathbb{F}_E$ in the case of analysis result. Based on these two sets, it is possible to calculate the pairwise precision rate $R_P = |\mathbb{F}_A \cap \mathbb{F}_E|/|\mathbb{F}_E|$, the pairwise recall rate $R_R = |\mathbb{F}_A \cap \mathbb{F}_E|/|\mathbb{F}_A|$, and the F-measure

$$F = \frac{2R_P R_R}{R_P + R_R}. \quad (5)$$

Using the above evaluation metric was proposed by Levy and Sandler [41]. Another closely related metric is the Rand index [34], used by Barrington et al. [7]. Abdallah et al. [1] proposed to match the segments in the analysis result and ground truth and to calculate a directional Hamming distance between frame sequences after the match. A similar approach with a differing background was proposed by Peeters [64]. A second evaluation metric proposed by Abdallah et al. [1] treats the structure descriptions as symbol sequences and calculates the mutual information between the analysis result and the ground truth. The mutual information concept was developed further by Lukashevich [45], who proposed an over- and under-segmentation measures based on the conditional entropies of the sequential representations of structures.

A property that can be considered to be a weakness in the metrics relying on pairs of frames, is that they disregard the order of the frames. In other words, they do not penalize hierarchical level differences between the computed parts such as splittings of segments into smaller parts. Chai [13], and Paulus and Klapuri [59] proposed heuristics finding a common hierarchical level for the computed structure result and the ground truth structure. However, the evaluation method is rather complicated, and the results are still subject for discussion.

Finally, it should be noted that most of the suggested evaluation metrics only consider one type of provided ground truth annotation. As the experiments by Bruderer et al. [10] suggest, the perception of musical structures is generally ambiguous. Thus the descriptions provided by two persons on the same piece may differ. A small-scale comparison of descriptions made by two annotators was presented by Paulus and Klapuri [62], and slight differences in the hierarchical levels as well as in the grouping were noted (using the F-measure (5) as the metric, human vs. human result was 89.4% whereas the employed computational method reached 62.4%). Peeters and Deruty [67] proposed a more well-defined ground truth annotation scheme that allows annotating the structure of a piece from several different aspects and temporal scales at the same time. The annotation can then be transformed to focus on the aspect relevant to the current application, e.g., by reducing it to be a temporal segmentation and grouping, as with earlier data sets.

The first systematic evaluation of different structure analysis methods methods took place in the Music Structure Segmentation task at the Music Information Retrieval Evaluation eXchange (MIREX) 2009 [5]. MIREX itself is a framework for evaluating music information retrieval algorithms where the evaluation tasks are defined by the research community under the coordination of International Music Information Retrieval Systems Evaluation Laboratory at the University of Illinois at Urbana-Champaign [20]. The evaluation task was kept relatively straightforward: providing a temporal segmentation of an entire piece and grouping of segments to parts. The evaluation data was provided from the OMRAS2 metadata project [49], and it consisted of 297 songs, mostly by The Beatles (179 songs), and the remaining songs were from four other performers making the data rather homogenous. It should also be noted that a large part of the data was publicly available before the evaluation and may have been used in the development of some of the methods. The five submissions from four teams represent slightly different approaches: one searches diagonal stripes from SDM in a greedy manner [50] ($F = 60.0\%$), one aims at maximizing a fitness function from a combined approach [62] ($F = 53.0\%$), and one uses agglomerative hierarchical clustering on smaller segments [66] ($F = 53.3\%$). The details of the two other submissions ($F = 57.7\%$ and $F = 58.2\%$) were not published. Despite the differring approaches, there were no significant performance differences between the methods and depending on the evaluation metric the ranking order changed considerably (with the Rand index metric the ranking is almost reversed).

7. CONCLUSIONS

This paper has given an overview of the music structure analysis problem, and the methods proposed for solving it. The methods have been divided into three categories: novelty-based approaches, homogeneity-based approaches, and repetition-based approaches. The comparison of different methods has been problematic because of the differring goals, but an effort at this was made in MIREX2009. The results of the evaluations suggest that none of the approaches is clearly superior at this time, and that there is still room for considerable improvements.

Perhaps one of the largest problems in music structure analysis is not directly technical, but more conceptual: the ground truth for this task should be better defined. The need for this is indicated by the fact that the annotations made by two persons disagree to a certain degree [62]. Defining the ground truth better requires interdisciplinary work between engineers and musicologists. The current results suggest that the structure description should not only be on a single level, but include also the information of hierarchical recurrences—similar to human perception. Another major task consists in collecting and annotating a representative data set, which is free for use in research projects worldwide. Also, contrary to many earlier

[5] http://www.music-ir.org/mirex/2009/index.php/Structural_Segmentation

data sets, it would be beneficial to have multiple parties involved to ensure data diversity and agreement on the target data. Having more accurate ground truth and a representative data set, the evaluation metrics can be defined more rigorously too: none of the current metrics corresponds to the perceived performance very accurately.

To date, the research has mostly been focusing on Western popular music, in which the sectional form is relatively prominent. It would be both challenging and interesting to broaden the target data set to include classical and non-Western music. Some of the principles employed by the current methods have been applied for these types of music too, but there is still a large need for research to cope with the complexity and diversity of general music data. As has been discussed in this paper, it is not enough to only use a single musical aspect in the analysis—also humans typically utilize multiple cues simultaneously. Related to this, more perceptually (and musically) motivated features should be investigated, as well as the distance measures used to compare frame-level features. Methods combining several musically motivated information sources have shown promising results, and such trends should be pursued further.

Acknowledgement. The first author was supported by the Academy of Finland, (application number 129657, Finnish Programme for Centres of Excellence in Research 2006–2011). The second author is supported by the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University.

8. REFERENCES

[1] S. Abdallah, K. Nolad, M. Sandler, M. Casey, and C. Rhodes. Theory and evaluation of a Bayesian music structure extractor. In *Proc. of 6th International Conference on Music Information Retrieval*, pages 420–425, London, England, UK, Sept. 2005.

[2] S. Abdallah, M. Sandler, C. Rhodes, and M. Casey. Using duration models to reduce fragmentation in audio segmentation. *Machine Learning*, 65(2–3):485–515, Dec. 2006.

[3] J.-J. Aucouturier and F. Pachet. Improving timbre similarity: How high's the sky. *Journal of Negative Results in Speech and Audio Sciences*, 1, 2004.

[4] J.-J. Aucouturier, F. Pachet, and M. Sandler. "The way it sounds": Timbre models for analysis and retrieval of music signals. *IEEE Transactions on Multimedia*, 7(6):1028–1035, Dec. 2005.

[5] J.-J. Aucouturier and M. Sandler. Segmentation of musical signals using hidden Markov models. In *Proc. of 110th Audio Engineering Society Convention*, Amsterdam, The Netherlands, May 2001.

[6] J.-J. Aucouturier and M. Sandler. Finding repeating patterns in acoustic musical signals: Applications for audio thumbnailing. In *Proc. of Audio Engineering Society 22nd International Conference on Virtual, Synthetic and Entertainment Audio*, pages 412–421, Espoo, Finland, 2002.

[7] L. Barrington, A. B. Chan, and G. Lanckriet. Modeling music as a dynamic texture. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):602–612, Mar. 2010.

[8] M. A. Bartsch and G. H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, Feb. 2005.

[9] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, Sept. 2005.

[10] M. J. Bruderer, M. McKinney, and A. Kohlrausch. Structural boundary perception in popular music. In *Proc. of 7th International Conference on Music Information Retrieval*, pages 198–201, Victoria, B.C., Canada, Oct. 2006.

[11] M. Casey and M. Slaney. The importance of sequences in musical similarity. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 5–8, Toulouse, France, May 2006.

[12] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing. On tempo tracking: Tempogram representation and kalman filtering. *Journal of New Music Research*, 28(4):259–273, 2001.

[13] W. Chai. *Automated Analysis of Musical Structure*. PhD thesis, Massachusetts Institute of Technology, Boston, Mass., USA, Sept. 2005.

[14] H.-T. Cheng, Y.-H. Yang, Y.-C. Lin, and H. H. Chen. Multimodal structure segmentation and analysis of music using audio and textual information. In *Proc. of IEEE International Symposium on Circuits and Systems*, pages 1677–1680, Taipei, Taiwan, May 2009.

[15] M. Cooper and J. Foote. Summarizing popular music via structural similarity analysis. In *Proc. of 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 127–130, New Platz, N.Y., USA, Oct. 2003.

[16] R. B. Dannenberg and M. Goto. Music structure analysis from acoustic signals. In D. Havelock, S. Kuwano, and M. Vorländer, editors, *Handbook of Signal Processing in Acoustics*, volume 1, pages 305–331. Springer, New York, N.Y., USA, 2008.

[17] R. B. Dannenberg and N. Hu. Pattern discovery techniques for music audio. In *Proc. of 3rd International Conference on Music Information Retrieval*, pages 63–70, Paris, France, Oct. 2002.

[18] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Readings in Speech Recognition*, pages 65–74, 1990.

[19] S. Dixon, E. Pampalk, and G. Widmer. Classification of dance music by periodicity patterns. In *Proc. of 4th International Conference on Music Information Retrieval*, pages 159–165, Baltimore, Md., USA, Oct. 2003.

[20] J. S. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[21] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle. Recurrence plots of dynamical systems. *Europhysics Letters*, 4(9):973–977, Nov. 1987.

[22] D. P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[23] A. Eronen. Chorus detection with combined use of MFCC and chroma features and image processing filters. In *Proc. of 10th International Conference on Digital Audio Effects*, pages 229–236, Bordeaux, France, Sept. 2007.

[24] J. Foote. Visualizing music and audio using self-similarity. In *Proc. of ACM Multimedia*, pages 77–80, Orlando, Fla., USA, 1999.

[25] J. Foote. Automatic audio segmentation using a measure of audio novelty. In *Proc. of IEEE International Conference on Multimedia and Expo*, pages 452–455, New York, N.Y., USA, Aug. 2000.

[26] J. Foote and S. Uchihashi. The beat spectrum: A new approach to rhythm analysis. In *Proc. of IEEE International Conference on Multimedia and Expo*, pages 881–884, Tokyo, Japan, Aug. 2001.

[27] S. Gao, N. C. Maddage, and C.-H. Lee. A hidden Markov model based approach to music segmentation and identification. In *Proc. of 4th Pacific Rim Conference on Multimedia*, pages 1576–1580, Singapore, Dec. 2003.

[28] J. Goldberger, S. Gordon, and H. Greenspan. An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures. In *Proc. of Ninth IEEE International Conference on Computer Vision*, pages 487–493, 2003.

[29] E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, UPF Barcelona, 2006.

[30] M. M. Goodwin and J. Laroche. A dynamic programming approach to audio segmentation and music / speech discrimination. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 309–312, Montreal, Que., Canada, May 2004.

[31] M. Goto. A chorus-section detecting method for musical audio signals. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 437–440, Hong Kong, 2003.

[32] P. Grosche and M. Müller. A mid-level representation for capturing dominant tempo and pulse information in music recordings. In *Proc. of 10th International Conference on Music Information Retrieval*, pages 189–194, Kobe, Japan, Oct. 2009.

[33] P. Grosche, M. Müller, and F. Kurth. Cyclic tempogram a mid-level tempo representation for music signals. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Dallas, Texas, USA, Mar. 2010.

[34] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, Dec. 1985.

[35] International Organization for Standardization. *ISO/IEC 15938-4:2002 Information technology – Multimedia content description interface – Part 4: Audio*. Geneva, Switzerland, 2002.

[36] T. Jehan. *Creating Music by Listening*. PhD thesis, Massachusetts

Institute of Technology, Boston, Mass., USA, Sept. 2005.

[37] K. Jensen. A causal rhythm grouping. In *Computer Music Modeling and Retrieval*, volume 3310 of *Lecture Notes in Computer Science*, pages 83–95. Springer Berlin / Heidelberg, 2004.

[38] K. Jensen. Multiple scale music segmentation using rhythm, timbre, and harmony. *EURASIP Journal on Advances in Signal Processing*, 2007. Article ID 73205, 11 pages.

[39] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, Cambridge, Mass., USA, 1999.

[40] M. Levy, K. Noland, and M. Sandler. A comparison of timbral and harmonic music segmentation algorithms. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1433–1436, Honolulu, Hawaii, USA, Apr. 2007.

[41] M. Levy and M. Sandler. Structural segmentation of musical audio by constrained clustering. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):318–326, Feb. 2008.

[42] M. Levy, M. Sandler, and M. Casey. Extraction of high-level musical structure from audio data and its application to thumbnail generation. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 13–16, Toulouse, France, May 2006.

[43] B. Logan and S. Chu. Music summarization using key phrases. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 749–752, Istanbul, Turkey, June 2000.

[44] L. Lu, M. Wang, and H.-J. Zhang. Repeating pattern discovery and structure analysis from acoustic music data. In *Proc. of Workshop on Multimedia Information Retrieval*, pages 275–282, New York, N.Y., USA, Oct. 2004.

[45] H. Lukashevich. Towards quantitative measures of evaluating song segmentation. In *Proc. of 9th International Conference on Music Information Retrieval*, pages 375–380, Philadelphia, Pa., USA, Sept. 2008.

[46] N. C. Maddage. Automatic structure detection for popular music. *IEEE Multimedia*, 13(1):65–77, Jan. 2006.

[47] N. C. Maddage, C. Xu, M. S. Kankanhalli, and X. Shao. Content-based music structure analysis with applications to music semantics understanding. In *Proc. of ACM Multimedia*, pages 112–119, New York, N.Y., USA, Oct. 2004.

[48] M. Marolt. A mid-level melody-based representation for calculating audio similarity. In *Proc. of 7th International Conference on Music Information Retrieval*, pages 280–285, Victoria, B.C., Canada, Oct. 2006.

[49] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler. OMRAS2 metadata project 2009. In *Proc. of 10th International Conference on Music Information Retrieval*, Kobe, Japan, Oct. 2009. Extended abstract.

[50] M. Mauch, K. Noland, and S. Dixon. Using musical structure to enhance automatic chord transcription. In *Proc. of 10th International Conference on Music Information Retrieval*, pages 231–236, Kobe, Japan, Oct. 2009.

[51] R. Middleton. Form. In B. Horner and T. Swiss, editors, *Key terms in popular music and culture*, pages 141–155. Wiley-Blackwell, Sept. 1999.

[52] M. Müller. *Information Retrieval for Music and Motion*. Springer, 2007.

[53] M. Müller and M. Clausen. Transposition-invariant self-similarity matrices. In *Proc. of 8th International Conference on Music Information Retrieval*, pages 47–50, Vienna, Austria, Sept. 2007.

[54] M. Müller, S. Ewert, and S. Kreuzer. Making chroma features more robust to timbre changes. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1877–1880, Taipei, Taiwan, Apr. 2009.

[55] M. Müller and F. Kurth. Enhancing similarity matrices for music audio analysis. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 437–440, Toulouse, France, May 2006.

[56] M. Müller and F. Kurth. Towards structural analysis of audio recordings in the presence of musical variations. *EURASIP Journal on Advances in Signal Processing*, 2007.

[57] B. S. Ong. *Structural analysis and segmentation of musical signals*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2006.

[58] B. S. Ong, E. Gómez, and S. Streich. Automatic extraction of musical structure using pitch class distribution features. In *Proc. of 1st Workshop on Learning the Semantics of Audio Signals*, Athens, Greece, Dec. 2006.

[59] J. Paulus and A. Klapuri. Music structure analysis by finding repeated parts. In *Proc. of 1st ACM Audio and Music Computing Multimedia Workshop*, pages 59–68, Santa Barbara, Calif., USA, Oct. 2006.

[60] J. Paulus and A. Klapuri. Acoustic features for music piece structure analysis. In *Proc. of 11th International Conference on Digital Audio Effects*, pages 309–312, Espoo, Finland, Sept. 2008.

[61] J. Paulus and A. Klapuri. Labelling the structural parts of a music piece with Markov models. In S. Ystad, R. Kronland-Martinet, and K. Jensen, editors, *Computer Music Modeling and Retrieval: Genesis of Meaning in Sound and Music - 5th International Symposium, CMMR 2008 Copenhagen, Denmark, May 19-23, 2008, Revised Papers*, volume 5493 of *Lecture Notes in Computer Science*, pages 166–176. Springer Berlin / Heidelberg, 2009.

[62] J. Paulus and A. Klapuri. Music structure analysis using a probabilistic fitness measure and a greedy search algorithm. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1159–1170, Aug. 2009.

[63] G. Peeters. Deriving musical structure from signal analysis for music audio summary generation: "sequence" and "state" approach. In *Computer Music Modeling and Retrieval*, volume 2771 of *Lecture Notes in Computer Science*, pages 143–166. Springer Berlin / Heidelberg, 2004.

[64] G. Peeters. Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach. In *Proc. of 8th International Conference on Music Information Retrieval*, pages 35–40, Vienna, Austria, Sept. 2007.

[65] G. Peeters. Template-based estimation of time-varying tempo. *EURASIP Journal on Advances in Signal Processing*, 2007(1):158–158, 2007.

[66] G. Peeters. MIREX 2009 "Music structure segmentation" task: Ircamsummary submission. In *Proc. of Fifth Annual Music Information Retrieval Evaluation eXchange*, Kobe, Japan, Oct. 2009. Extended abstract.

[67] G. Peeters and E. Deruty. Is music structure annotation multi-dimensional? A proposal for robust local music annotation. In *Proc. of 3rd Workshop on Learning the Semantics of Audio Signals*, pages 75–90, Graz, Austria, Dec. 2009.

[68] G. Peeters, A. La Burthe, and X. Rodet. Toward automatic music audio summary generation from signal analysis. In *Proc. of 3rd International Conference on Music Information Retrieval*, pages 94–100, Paris, France, Oct. 2002.

[69] E. Peiszer. Automatic audio segmentation: Segment boundary and structure detection in popular music. Master's thesis, Vienna University of Technology, Vienna, Austria, Aug. 2007.

[70] C. Rhodes and M. Casey. Algorithms for determining and labelling approximate hierarchical self-similarity. In *Proc. of 8th International Conference on Music Information Retrieval*, pages 41–46, Vienna, Austria, Sept. 2007.

[71] M. P. Ryynänen and A. P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[72] Y. Shiu, H. Jeong, and C.-C. J. Kuo. Musical structure analysis using similarity matrix and dynamic programming. In *Proc. of SPIE Vol. 6015 - Multimedia Systems and Applications VIII*, pages 398–409, 2005.

[73] Y. Shiu, H. Jeong, and C.-C. J. Kuo. Similar segment detection for music structure analysis via Viterbi algorithm. In *Proc. of IEEE International Conference on Multimedia and Expo*, pages 789–792, Toronto, Ont., Canada, July 2006.

[74] H. Terasawa, M. Slaney, and J. Berger. The thirteen colors of timbre. In *Proc. of 2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 323–326, New Platz, N.Y., USA, Oct. 2005.

[75] D. Turnbull, G. Lanckriet, E. Pampalk, and M. Goto. A supervised approach for detecting boundaries in music using difference features and boosting. In *Proc. of 8th International Conference on Music Information Retrieval*, pages 51–54, Vienna, Austria, Sept. 2007.

[76] G. Tzanetakis and P. Cook. Multifeature audio segmentation for browsing and annotation. In *Proc. of 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 103–106, New Platz, N.Y., USA, Oct. 1999.

[77] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proc. of Seventh IEEE International Conference on Computer Vision*, pages 975–982, Kerkyra, Greece, Sept. 1999.

[78] J. Wellhausen and M. Höynck. Audio thumbnailing using MPEG-7 low-level audio descriptors. In *Proc. of The SPIE Internet Multimedia Management Systems IV*, volume 5242, pages 65–73, Nov. 2003.

[79] C. Xu, X. Shao, N. C. Maddage, M. S. Kankanhalli, and T. Qi. Automatically summarize musical audio using adaptive clustering. In *Proc. of IEEE International Conference on Multimedia and Expo*, pages 2063–2066, Taipei, Taiwan, June 2004.

[80] R. Zhou, M. Mattavelli, and G. Zoia. Music onset detection based on resonator time frequency image. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1685–1695, 2008.

music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data

Michael Scott Cuthbert **Christopher Ariza**

Music and Theater Arts
Massachusetts Institute of Technology
{cuthbert, ariza}@mit.edu

ABSTRACT

music21 is an object-oriented toolkit for analyzing, searching, and transforming music in symbolic (score-based) forms. The modular approach of the project allows musicians and researchers to write simple scripts rapidly and reuse them in other projects. The toolkit aims to provide powerful software tools integrated with sophisticated musical knowledge to both musicians with little programming experience (especially musicologists) and to programmers with only modest music theory skills.

This paper introduces the music21 system, demonstrating how to use it and the types of problems it is well-suited toward advancing. We include numerous examples of its power and flexibility, including demonstrations of graphing data and generating annotated musical scores.

1. INTRODUCTION: WHY MUSIC21?

Computers have transformed so many aspects of musicology—from writing and editing papers, to studying manuscripts with digital files, to creating databases of composers' letters, to typesetting editions—that it is incredible that most analytical tasks that music historians perform remain largely untouched by technology. The study of the rich troves of musical data in scores, sketches, intabulations, lead-sheets, and other sources of symbolic music data is still done almost exclusively by hand. Even when databases and spreadsheets are employed, they are usually created for a single purpose. Such specialized approaches cannot easily be reused.

Computer scientists often assume that, compared to working with scanned images of scores or sound files, manipulating symbolic data should be a cinch. Most of the information from a score can easily be converted to text-based or numeric formats that general-purpose statistical or information-retrieval tools can manipulate. In practice the complexities of music notation and theory result in these tools rarely being sufficient.

For instance, a researcher might want to compare how closely the music of two composers adheres to a particular scale (say, the major scale). What begins as a straightforward statistical problem requiring little musical knowledge—simply encode which notes are in the scale of the piece's key and which are not—can quickly grow beyond the capabilities of general statistics packages. Suppose that after some initial work, our researcher decides that notes on stronger beats should be weighed more heavily than those on weaker beats. Now she must either add the information about beats by hand to each note or write a new algorithm that labels the beats. Beat labeling is another task that initially seems easy but rapidly becomes extremely troublesome for several reasons. Are grace-notes accented or unaccented? Only a musically-trained ear that also knows the norms of an era can tell. Incompletely-filled measures, such as pickup measures and mid-bar repeats, present problems for algorithms. As the researcher's corpus expands, the time spent on meta-research expands with it. What began as a straightforward project becomes a set of tedious separate labors: transforming data from multiple formats into one, moving transposing instruments into sounding pitch, editorial accidentals in early music, or finding ways of visualizing troublesome moments for debugging.

Researchers in other fields can call upon general-purpose toolkits to deal with time-consuming yet largely solved problems. For instance, a scientist working with a large body of text has easy access to open-source libraries for removing punctuation, converting among text-encoding formats, correcting spelling, identifying parts of speech, sentence diagramming, automatic translation, and of course rendering text in a variety of media. Libraries and programs to help with the musical equivalents of each of these tasks do exist, but few exchange data with each other in standardized formats. Even fewer are designed in modern, high-level programming languages. As a result of these difficulties, computational solutions to musicological problems are rarely employed even when they would save time, expand the scope of projects, or quickly find important exceptions to overly broad pronouncements.

The music21 project (http://web.mit.edu/music21) expands the audience for computational musicology by creating a new toolkit built from the ground up with intuitive simplicity and object-oriented design throughout. (The "21" in the title comes from the designation for MIT's classes in Music, Course 21M.) The advantages of object-oriented design have led to its wide adoption in many realms of software engineering. These design principles have been employed in music synthesis and generation systems over the past 25 years [2, 9, 10] but have not been thoroughly integrated into packages for the analysis of music as symbolic data. Humdrum, the most widely adopted software package [6], its contemporary ports [7, 11], and publications using these packages show the great promise of computational approaches to music theory and musicology. Yet Humdrum can be difficult to use: both programmers and non-programmers alike may find its reliance on a chain of shell-scripts, rather than object-oriented libraries, limiting and not intuitive.

Nicholas Cook has called upon programmers to create for musicologists "a modular approach involving

an unlimited number of individual software tools" [3]. A framework built with intuitive, reusable, and expandable objects satisfies Cook's call without sacrificing power for more complex tasks.

As a new, open-source, cross-platform toolkit written in Python, `music21` provides such a modular approach, melding object-oriented music representation and analysis with a concise and simple programming interface. Simplicity of use, ease of expansion, and access to existing data are critical to the design of `music21`. The toolkit imports Humdrum/Kern, MusicXML [4], and user-defined formats (with MIDI and MuseData forthcoming). Because it is written in Python, `music21` can tap into many other libraries, integrating internet resources (such as geomapping with Google Maps), visualization software, and sophisticated database searches with musical analysis.

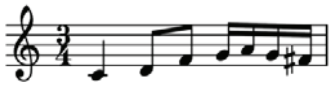
This brief paper gives an overview of the `music21` toolkit. Through examples of musicological applications that the system enables, the main distinguishing features are illustrated: simplicity of use and expansion.

2. SCRIPTING AND OBJECTS

`Music21` is built in Python, a well-established programming language packaged with Macintosh and Unix computers and freely downloadable for Windows users. The toolkit adds a set of related libraries, providing sophisticated musical knowledge to Python. As shown in Figure 1, after adding the system with "`from music21 import *`", straightforward tasks such as displaying or playing a short melody, getting a twelve-tone matrix, or converting from Humdrum's Kern format to MusicXML can each be accomplished with a single line of code.

Display a simple melody in musical notation:

```
tinyNotation.TinyNotationStream(
      "c4 d8 f g16 a g f#", "3/4").show()
```



Print the twelve-tone matrix for a tone row (in this case the opening of Schoenberg's Fourth String Quartet):

```
print(serial.rowToMatrix(
          [2,1,9,10,5,3,4,0,8,7,6,11]) )
```

or since most of the 2nd-Viennese school rows are already available as objects, you could instead type:

```
print(serial.RowSchoenbergOp37().matrix() )
```

*Convert a file from Humdrum's **kern data format to MusicXML for editing in Finale or Sibelius:*

```
parse('/users/documents/composition.krn').
    write('xml')
```

Figure 1. Three simple examples of one-line `music21` scripts.

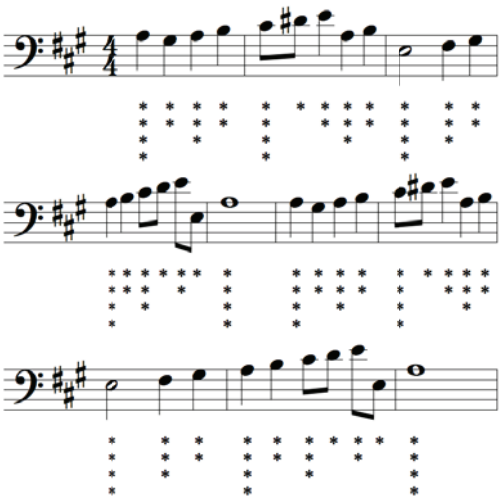
Though single-line tasks are simpler to accomplish in `music21` than in existing packages, the full power of the new toolkit comes from bringing together and extending high-level objects. The framework includes many objects, including Pitches, Chords, Durations, TimeSignatures, Intervals, Instruments, and standard Ornaments. Through method calls, objects can perform their own analyses and transformations. For instance, Chords can find their own roots, create closed-position versions of themselves, compute their Forte prime forms, and so on. Researchers can extend objects for their own needs, such as altering the pitch of open Violin strings to study *scordatura,* specializing (subclassing) the Note class into MensuralNote for studying Renaissance Music, or grouping Measures into Hypermeters. The object-oriented design of `music21` simplifies writing these extensions.

3. STREAMS: POWERFUL, NESTABLE, CONTAINERS OF TIMED ELEMENTS

At the core of `music21` is a novel grouping of musical information into Streams: nestable containers that allow researchers to quickly find simultaneous events, follow a voice, or represent instruments playing in different tempi and meters. Elements within Streams are accessed with methods such as `getElementById()`, an approach similarly to the Document Object Model (DOM) of retrieving elements from within XML and HTML documents. Like nearly every `music21` object, Streams can immediately be visually displayed in Lilypond or with programs that import MusicXML (such as Finale and Sibelius). Through the Stream model, a program can find notes or chords satisfying criteria that change from section to section of a piece, such as all notes that are the seventh-degree of the current key (as identified either manually or with an included key-detection algorithm) and then retrieve information such as the last-defined clef, dynamic, or metrical accent level at that point.

Many tools to visualize, process, and annotate Streams come with the `music21` framework. These tools include graphing modules, analytical tools, and convenience routines for metrical analysis [8], phrase extraction, and identification of non-harmonic tones. Figure 2 demonstrates the use of metrical analysis, derived from nested hierarchical subdivisions of the time signature [1], to annotate two Bach chorales in different meters.

```
from music21.analysis import metrical

# load a Bach Chorale from the music21 corpus of supplied pieces
bwv30_6 = corpus.parseWork('bach/bwv30.6.xml')

# get just the bass part using DOM-like method calls
bass = bwv30_6.getElementById('Bass')

# get measures 1 through 10
excerpt = bass.getMeasureRange(1,10)

# apply a Lerdahl/Jackendoff-style metrical analysis to the piece.
metrical.labelBeatDepth(excerpt)

# display measure 0 (pickup) to measure 6 in the default viewer
# (here Finale Reader 2009)
excerpt.show()
```



```
# perform the same process on a different chorale in 3/4 time
bwv11_6 = corpus.parseWork('bach/bwv11.6.xml')
alto = bwv11_6.getElementById('Alto')
excerpt = alto.getMeasureRange(13,20)
metrical.labelBeatDepth(excerpt)
excerpt.show()
```

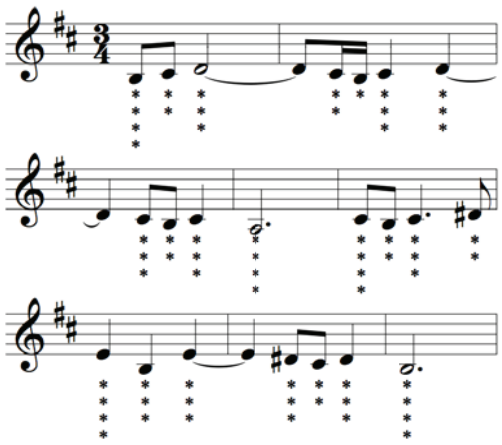


Figure 2. Analytical tools, such as this metrical accent labeler, are included with `music21` and work with most Streams (including Scores and Parts). Here, excerpts of two Bach chorales, each in a different meter, are labeled with dots corresponding to their metric strengths.

4. FURTHER FEATURES

In addition to providing sophisticated resources in a modern programming language, the `music21` package takes advantage of some of the best contemporary approaches to software distribution, licensing, development, and documentation. These approaches assure both the longevity of the system across multiple platforms as well as the ability of the system to grow and incorporate the work of contributors.

4.1 An Integrated and Virtual Corpus of Music for Researchers

`Music21` comes with a corpus package, a large collection of freely-distributable music for analysis and testing, including a complete collection of the Bach Chorales, numerous Beethoven String Quartets, and examples of Renaissance polyphony. The virtual corpus extends the corpus package even further. Similar to a collection of URL bookmarks to music resources, additional repertories, available online, can be automatically downloaded when first requested and then made available to the researcher for future use. The corpus includes both Kern and MusicXML files. Future system expansions will not only grow the tools for analysis, but also the breadth and depth of the corpus of works.

4.2 Permissive License and Full Documentation

`Music21` is a toolkit: a collection of tools that work together in a wide range of contexts. The promise of a toolkit is only achieved if users can expand and integrate software components in their own work. Thus, `music21` is released under the Lesser General Public License (LGPL), allowing its use within both free and commercial software. So that implementation and documentation stay synchronized, the toolkit also features high-quality, indexed, and searchable documentation of all modules and classes, automatically created from the source code and test routines. The `music21` site (http://web.mit.edu/music21) hosts up-to-date information, documentation and release links. Code browsing, feature requests, and bug reports are housed at Google Code.

5. EXAMPLES

Better than an explanation of high-level features, a few specific examples illustrate the toolkit's promise. These examples are chosen for both their novel and practical utility.

5.1 Finding Chords within Melodic Fragments

The script in Figure 3 searches the entire second violin part of a MusicXML score of Beethoven's *Große Fuge*, op. 133, to find measures that melodically express dominant seventh chords in consecutive notes. It then displays the chord in closed position, the surrounding measure, and the Forte prime form. (Running the same query across barlines would add just a few lines of code).

```
op133 = corpus.parseWork(
              'beethoven/opus133.xml')
violin2 = op133.getElementById('2nd Violin')

#  an empty container for later display
display = stream.Stream()

for thisMeasure in violin2.measures:

  # get a list of consecutive notes, skipping unisons, octaves,
  # and rests (and putting nothing in their places)
  notes = thisMeasure.findConsecutiveNotes(
  skipUnisons = True, skipOctaves = True,
  skipRests = True, noNone = True)

  pitches = stream.Stream(notes).pitches
```

```
  for i in range(len(pitches) - 3):
    # makes every set of 4 notes into a whole-note chord
    testChord = chord.Chord(pitches[i:i+4])
    testChord.duration.type = "whole"

    if testChord.isDominantSeventh():
      # A dominant-seventh chord was found in this measure.

      # We label the chord with the measure number
      # and the first note of the measure with the Forte Prime form
      testChord.lyric = "m. " + str(
                thisMeasure.measureNumber)
      primeForm = chord.Chord(
          thisMeasure.pitches).primeFormString
      firstNote = thisMeasure.notes[0]
      firstNote.lyric = primeForm

      # Then we append the chord in closed position followed
      # by the measure containing the chord.

      chordMeasure = stream.Measure()
      chordMeasure.append(
         testChord.closedPosition())
      display.append(chordMeasure)
      display.append(thisMeasure)
display.show()
```

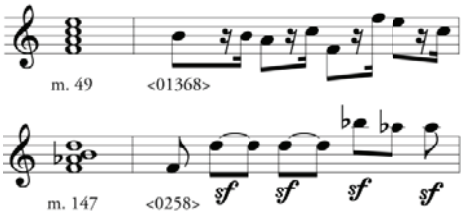
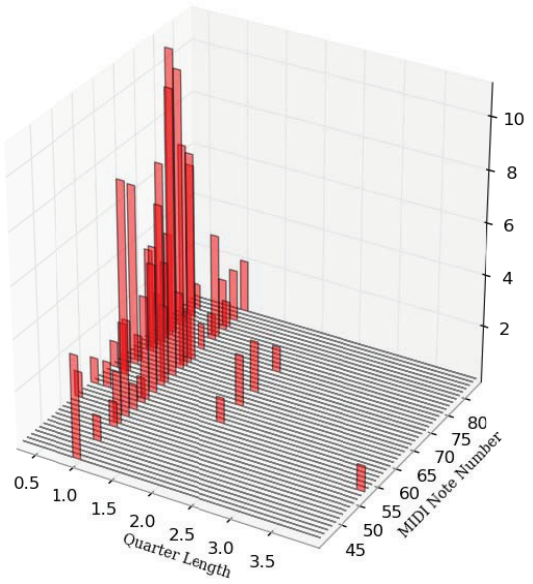


Figure 3. The results of a search for chords expressed melodically.

5.2 Distributions of Notes by Pitch and Duration

Figure 4 demonstrates the ability of `music21` graphs to help visualize trends that are otherwise difficult to discern. These graphs plot three features: pitch, duration of notes, and how frequently these pitches and durations are used. From two small excerpts of pieces in 3/4 by Mozart (a minuet, in red) and by Chopin (a mazurka, in blue), it can be seen that pitches in the Mozart example follow a type of bell-curve distribution, with few high notes, few low notes, and many notes toward the middle of the registral space. Chopin's usage jumps throughout the piano. The differences in pitch usage suggest that this line of inquiry is worth pursuing further, but no connection between duration and pitch appears. `Music21`'s easy-to-use graphing methods help researchers find the best visualization tool for their data, easily switching among diverse formats.

```
from music21.musicxml import testFiles as xml
from music21.humdrum import testFiles as kern

# display 3D graphs of count, pitch, and duration
mozartStream = music21.parse(
    xml.mozartTrioK581Excerpt)
notes = mozartStream.flat.stripTies()
g = graph.Plot3DBarsPitchSpaceQuarterLength(
    notes, colors=['r'])
g.process()
```




```
# perform the same process on a different work
chopinStream = music21.parse(kern.mazurka6)
notes = chopinStream.flat.stripTies()
g = graph.Plot3DBarsPitchSpaceQuarterLength(
    notes, colors=['b'])
g.process()
```

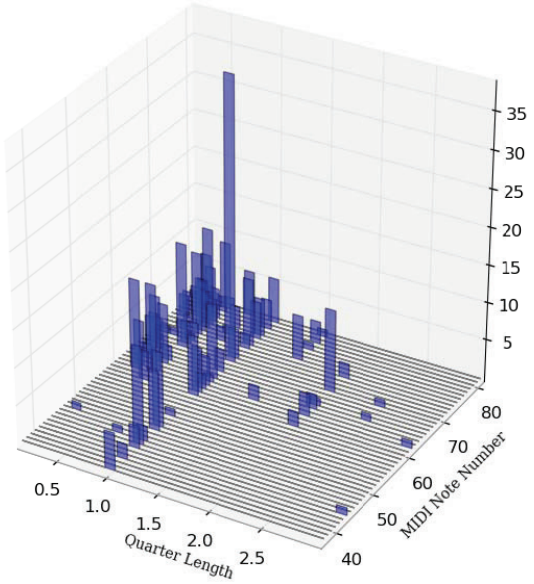


Figure 4. Differences in pitch distribution between Mozart and Chopin.

The Mozart and Chopin examples, while showing distinctive individual usage, show little correlation between pitch and duration. Many other pieces do. An extreme example is Messiaen's "Mode de valeurs et d'intensités" from *Quatre études de rythme*, perhaps the first work of total serialism. A perfect correlation between pitch and duration, as found in the middle voice (isolated for clarity), is plotted in Figure 5. An aspect of

the composition that is difficult to observe in the score but easy to see in this graph is the cubic shape ($-x^3$) made through the choice of pitches and rhythms. This shape is not at all explained by the serial method of the piece. Also easily seen is that, although Messiaen uses lower notes less often, there is not a perfect correlation between pitch and frequency of use (e.g., 21 B-flats vs. 22 A-flats).

```
messiaen = converter.parse(
               'd:/valeurs_part2.xml')
notes = messiaen.flat.stripTies()
g = graph.PlotScatterWeightedPitch\
    SpaceQuarterLength(notes, xLog = False,
    title='Messiaen, Mode de Valeurs,
    middle voice')
g.process()
```

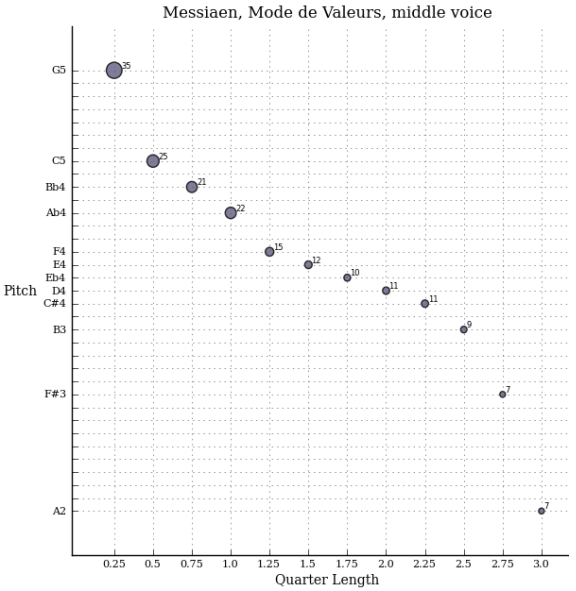


Figure 5. A graph of pitch to duration relationships in Messiaen, "Mode de valeurs," showing the correlation between the two note attributes.

5.3 Pitch Class Density Over Time

In Figure 6, pitch class usage, over the duration of the composition in the cello part of a MusicXML score of Beethoven's *Große Fuge*, is graphed. Even though the temporal resolution of this graph is coarse, it is clear that the part gets less chromatic towards the end of the work. (We have manually highlighted the tonic and dominant in this example.)

```
beethovenScore = corpus.parseWork('opus133.xml')
celloPart = \
      beethovenScore.getElementById('Cello')

# given a "flat" view of the stream, with nested information
# removed and all information at the same hierarchical level,
# combine tied notes into single notes with summed durations
notes = celloPart.flat.stripTies()

g = graph.PlotScatterPitchClassOffset(notes,
    title='Beethoven Opus 133, Cello')
g.process()
```

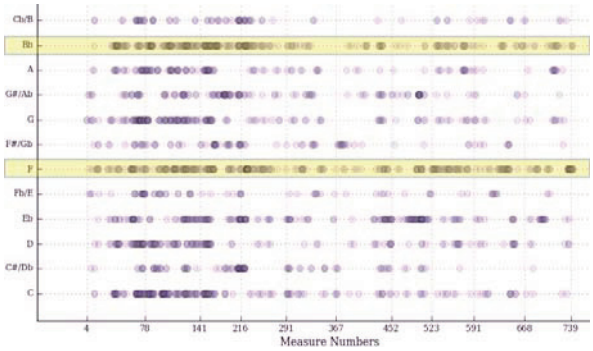


Figure 6. A graph of pitch class usage over time in Beethoven's *Große Fuge*.

5.4 Testing Nicolaus de Capua's *Regulae* of *Musica Ficta*

This example shows a sophisticated, musicological application of `music21`. Among his other writings, the early-fifteenth century music theorist Nicolaus of Capua gave a set of *regulae*, or rules, for creating *musica ficta* [5]. *Musica ficta*, simply put, was a way of avoiding tritones and other undesirable intervals and create more conclusive cadences through the use of unwritten accidentals that performers would know to sing. Unlike the rules of most other theorists of his time, Nicolaus's four rules rely solely on the melodic intervals of one voice. Herlinger's study of Nicolaus's rules suggested that they could be extremely successful at eliminating harmonic problems while at the same time being easy enough for any musician to master. However, as is conventional in musicology, this study was performed by hand on a few excerpts of music by a single composer, Antonio Zachara da Teramo. Using `music21` we have been able to automatically apply Nicolaus's rules to a much wider set of encoded music, the complete incipits and cadences of all Trecento ballate (about 10,000 measures worth of music) and then automatically evaluate the quality of harmonic changes implied by these rules. Figure 7 shows an excerpt of the code for a single rule, that a descending major second ("M-2") immediately followed by an ascending major second ("M2") should be transformed into two half-steps by raising the middle note:

```
# n1, n2, and n3 are three consecutive notes
# i1 is the interval between n1 and n2
# i2 is the interval between n2 and n3

i1 = generateInterval(n1,n2)
i2 = generateInterval(n2,n3)

# we test if the two intervals are the ones fitting the rule
if i1.directedName == "M-2" and \
   i2.directedName == "M2":

  # since the intervals match , we add an editorial accidental
  n2.editorial.ficta = \
         Accidental("sharp")
```

```
  # we also color the affected notes so that if we display the music
  # the notes stick out. Different colors indicate different rules
  n1.editorial.color = "blue"
  n2.editorial.color = "forestGreen"
  n3.editorial.color = "blue"
```

Figure 7. Applying ficta accidentals with `music21`.

The results of applying one or all the rules to an individual cadence or piece can be seen immediately. Figure 8 shows the rules applied to one piece where they create two "closest-approaches" to perfect consonances (major sixth to octave and minor third to unison). These are the outcomes one expects from a good set of *regulae* for *musica ficta*.

```
  # get a particular worksheet of an Excel spreadsheet
ballataObj = cadencebook.BallataSheet()
  # create an object for row 267
pieceObj = ballataObj.makeWork(267)
  # run the four rules (as described above)
applyCapua(pieceObj)
  # display the first cadence of the piece (second snippet) by
  # running it through Lilypond and generating a PNG file
pieceObj.snippets[1].lily.showPNG()
```

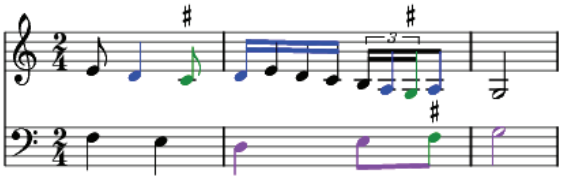


Figure 8. Music21 code for automatically adding *musica ficta* to Francesco (Landini), *De[h], pon' quest'amor,* first cadence.

In other pieces, Nicolaus's rules have an injurious effect, as Figure 9 shows. With the toolkit, we were able to run the rules on the entire database of Trecento ballatas and determine that Nicolaus's rules cannot be used indiscriminately. Far too many cases appeared where the proposed *ficta* hurt the harmony. One of the main advantages of the `music21` framework is making such observations on large collections of musical data possible.

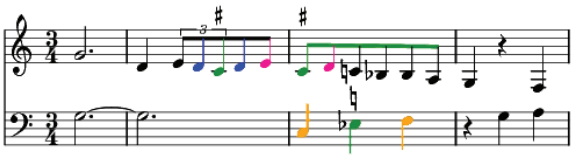


Figure 9. Francesco, *D'amor mi biasmo,* incipit after automatically applying *ficta* accidentals.

6. FUTURE WORK

The first alpha releases of `music21` introduce fundamental objects and containers and, as shown above, offer powerful tools for symbolic music processing.

The next stage of development will add native support for additional input and output formats, including MIDI. Further, libraries of additional processing, analysis, visualization routines, as well as new and expanded object models (such as non-Western scales), will be added to the system. We are presently focusing on creating simple solutions for common-practice music theory tasks via short `music21` scripts, and within a year hope to be able to solve almost every common music theory problem encountered by first-year conservatory students.

7. ACKNOWLEDGEMENTS

The authors thank the Seaver Institute for their generous funding of `music21`. Additional thanks are also extended to three anonymous reviewers for their helpful comments.

8. REFERENCES

[1] Ariza, C. and M. Cuthbert. 2010. "Modeling Beats, Accents, Beams, and Time Signatures Hierarchically with music21 Meter Objects." In *Proceedings of the International Computer Music Conference.* San Francisco: International Computer Music Association.

[2] Buxton, W. and W. Reeves, R. Baecker, L. Mezei. 1978. "The Use of Hierarchy and Instance in a Data Structure for Computer Music." *Computer Music Journal* 2 (4): 10-20.

[3] Cook, N. 2004. "Computational and Comparative Musicology." In *Empirical Musicology: Aims, Methods, Prospects*. N. Cook and E. Clarke, eds. New York: Oxford University Press. 103-126.

[4] Good, M. 2001. "An Internet-Friendly Format for Sheet Music." In *Proceedings of XML 2001*.

[5] Herlinger, J. 2004. "Nicolaus de Capua, Antonio Zacara da Teramo, and *musica ficta."* In *Antonio Zacara da Teramo e il suo tempo.* F. Zimei, ed. Lucca: LIM. 67–89.

[6] Huron, D. 1997. "Humdrum and Kern: Selective Feature Encoding." In *Beyond MIDI: the Handbook of Musical Codes*. E. Selfridge-Field, ed. Cambrdige: MIT Press. 375-401.

[7] Knopke, I. 2008. "The PerlHumdrum and PerlLilypond Toolkits for Symbolic Music Information Retrieval." *ISMIR 2008* 147-152.

[8] Lerdahl, F. and R. Jackendoff. 1983. *A Generative Theory of Tonal Music*. Cambridge: MIT Press.

[9] Pope, S. T. 1987. "A Smalltalk-80-based Music Toolkit." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 166-173.

[10] Pope, S. T. 1989. "Machine Tongues XI: Object-Oriented Software Design." *Computer Music Journal* 13 (2): 9-22.

[11] Sapp, C. S. 2008. "Museinfo: Musical Information Programming in C++." Internet: http://museinfo.sapp.org.

AN AUDIO PROCESSING LIBRARY FOR MIR APPLICATION DEVELOPMENT IN FLASH

Jeffrey Scott[†], Raymond Migneco[†], Brandon Morton[†], Christian M. Hahn[‡]
Paul Diefenbach[‡], Youngmoo E. Kim[†]
Electrical and Computer Engineering, Drexel University[†]
Media Arts and Design, Drexel University[‡]
{jjscott, rmigneco, bmorton, cmhahn, pjdief, ykim }@drexel.edu

ABSTRACT

In recent years, the Adobe *Flash* platform has risen as a credible and universal platform for rapid development and deployment of interactive web-based applications. It is also the accepted standard for delivery of streaming media, and many web applications related to music information retrieval, such as Pandora, Last.fm and Musicovery, are built using Flash. The limitations of Flash, however, have made it difficult for music-IR researchers and developers to utilize complex sound and music signal processing within their web applications. Furthermore, the real-time audio processing and synchronization required for some music-IR-related activities demands significant computational power and specialized audio algorithms, far beyond what is possible to implement using Flash scripting. By taking advantage of features recently added to the platform, including dynamic audio control and C cross-compilation for near-native performance, we have developed the *Audio-processing Library for Flash* (ALF), providing developers with a library of common audio processing routines and affording Flash developers a degree of sound interaction previously unavailable through web-based platforms. We present several music-IR-driven applications that incorporate ALF to demonstrate its utility.

1. INTRODUCTION

The use of web applications is now commonplace due to the widespread availability of broadband connections, improved client processing power, and the capabilities afforded by Adobe Flash. Flash has become the dominant platform for the development of web-based interactive media applications by providing tools for easily implementing rich graphics, animation and user interface controls as well as cross-platform deployment. Despite its popularity, however, Flash's support for sound and music processing has historically been limited. ActionScript, Flash's native development language, was never intended to accommodate computationally intensive algorithms, such as the signal processing required for real-time audio feature extraction and analysis.

Recognizing the potential for developing audio- and music-centric applications on the web, we have developed the *Audio processing Library for Flash* (ALF), which addresses the audio limitations of the Flash platform. ALF is based on Flash version 10 and capitalizes on the the recently introduced Adobe *Alchemy* framework, which allows existing algorithms written in C/C++ to be compiled into byte code optimized for the ActionScript Virtual Machine for significantly improved performance [1, 2]. By utilizing the dynamic audio capabilities recently added to Flash 10 and the computational benefits of Alchemy, ALF provides Flash developers with a library of common audio processing routines that can be incorporated into applications, such as spectral feature extraction and analysis, filtering and reverberation.

By including real-time audio processing capabilities to Flash, ALF provides web applications with an additional degree of sound interaction that has previously only been available on native PC platforms. For example, ALF is capable of supporting music-based games in Flash requiring responses from the player precisely timed to music. Although ALF can be used to enhance the audio of any Flash application, our goal is to enable a new form of web apps that can be driven by user-supplied audio. This potentially allows a user to choose a wide range of customized musical inputs, such as selections from their personal collection or completely user-generated music content (song remixes and mashups, which are becoming increasingly commonplace). As we will demonstrate, ALF facilitates the development of games that are dynamically driven by the acoustic features of songs from a user's music library, thus creating unique game play experiences depending on the provided audio content.

2. RELATED WORK

There are many software packages available that provide libraries for feature extraction and audio synthesis that exist as open source projects for research and development. While many provide similar functionality, each library was developed to address particular implementation issues, such as cross-platform support, computational ef-

ficiency and ease of implementation. In this section, we provide a brief description of some existing libraries.

Marsyas (Music Analysis, Retrieval and Synthesis for Audio Signals) is an audio processing and MIR framework built in C++ with a GUI based on Qt4 [3]. The project includes a wide variety of functions for analysis and synthesis as well as audio features and classification algorithms. Being one of the first such projects, the scope of Marsyas is significant and it has been used in many research projects as well as commercial endeavors.

jAudio was developed to be an easy to use Java-based system for feature extraction. The cross-platform nature of Java and GUI tools were the motivating factors for the choice of development language. The creators attempted to make the system as easily extensible as possible, avoid redundant computation, and ensure the algorithms were separate from other functionality to increase ease of portability [4].

M2K is a project under the International Music Information Retrieval System Evaluation Laboratory which is based off of the Data to Knowledge (D2K) machine learning and data mining environment [5]. D2K is employs a visual programming environment in which users connect modules together to prototype algorithms. The M2K project has taken this framework and built in an array of MIR tools for rapid development and testing of MIR systems.

The *MIRToolbox* is an audio feature extraction library built in MATLAB that emphasizes a modular, parameterizable framework [6]. The project offers a wide range of low-level and high-level features as well as tools for statistical analysis, segmentation and clustering.

CLAM is an analysis/synthesis system written in C++ designed to be entirely object-oriented to allow for significant re-usability of code and functionality [7]. It provides audio and MIDI input/output, supports XML and provides tools for data visualization.

FEAPI is a platform-independent programming application interface for low-level feature extraction written in C [8]. In contrast to the previously described systems, FEAPI allows developers to create their own applications using C/C++ without being required to use the interfaces designed to work with the above libraries.

3. IMPLEMENTATION

The driving force behind the development of ALF was to provide developers with an efficient, cross-platform and open source MIR and audio synthesis library. By choosing Flash as the development platform, we target developers seeking to rapidly develop and deploy web-based and/or cross-platform desktop applications. As we will discuss, the multi-layered and open source architecture of ALF also permits ease of development for programmers with various expertise and does not require prior knowledge or experience in audio programming.

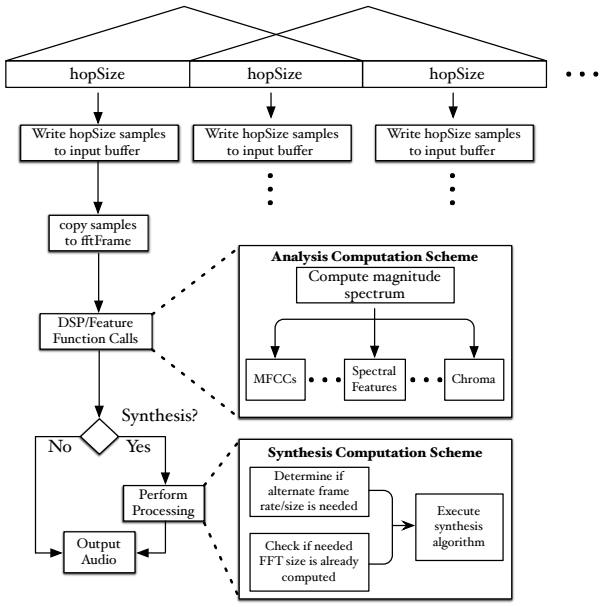


Figure 1. Frame-based computation and processing flow in ALF.

3.1 Architecture

The dynamic audio functionality in the current version of Flash is somewhat asynchronous, allowing sound to be processed outside of the main application thread. Thus the DSP routines can execute independently of the Flash script rather than having to wait for C/C++ functions to finish executing, allowing front-end UI and other operations to continue if they are not dependent on data computed using ALF functions.

There are several layers of abstraction in ALF providing a flexible framework with various levels of control depending on the needs of the developer. The heavy computation is executed by the C/C++ functions which are compiled using the Adobe supplied Alchemy compiler for use with ActionScript (AS). We provide a basic AS wrapper to properly handle the shared memory management between C/C++ and ActionScript for those wishing to have basic access to the C functionality. The top layer streamlines audio input/output and provides simple calls to perform feature extraction and analysis-synthesis tasks. The entire project is open source so that a developer may customize the architecture to meet application-specific needs. ALF is fully documented and is currently available via a subversion repository online [1] .

To ensure tight synchrony between the video and audio output in Flash, the processing flow was developed according to the diagram shown in Figure 1. The frame size is set by the the video frame rate since ALF is designed with graphical oriented applications in mind, thus the time-frequency resolution of the system is also determined by this parameter. Whenever possible, a single FFT is used in computing the features returned to the user, however, certain algorithms require transforms of sizes other than the

[1] http://music.ece.drexel.edu/ALF

Table 1. ALF Functions

	Function Name	Description
Analysis	Spectrum	Computes the magnitude spectrum using the FFT algorithm
	Harmonics	Finds the harmonics of the frequency spectrum
	MFCC	Calculates the Mel-Frequency Cepstral Coefficients
	LPC	Performs Linear Prediction and returns the coefficients and gain
	Bandwidth	The frequency range present in the signal
	Centroid	The center of gravity of the frequency spectrum
	Flux	The change in energy from the previous frame
	Intensity	Calculates the energy of the spectrum
	Rolloff	The frequency below which %85 of the spectral energy lies
	Autocorrelation	Computes the autocorrelation via the FFT
	Chroma	An representation of the spectral energy present in the 12 individual semitones
	Beat Tracking	Returns whether a beat occurs on each frame (based on bandwise autocorrelation)
Synthesis	Filter	Filters the audio signal - FIR and IIR implementation
	Reverb	Applies reverb by using a room impulse response (RIR) as an FIR filter
	Phase Vocoder	Alters the tempo and/or pitch of the audio

default size. A shared buffer system is also used so that we can perform operations at variable frame rates and overlap lengths without having to read in the data again using different frame sizes.

3.2 Performance

As previously mentioned, the computationally intensive routines in ALF are implemented in the Alchemy-optimized C code to avoid the limitations of ActionScript. While slightly slower than native C code, the Alchemy-optimized code provides significant performance gains over identical algorithms implemented with ActionScript. In a related paper, we performed a benchmark analysis of the FFT algorithm using the ActionScript *as3mathlib* implementation versus our Alchemy-compiled C implementation as well as Java's JTransforms. Averaging the computation speed over 10,000 iterations, we showed our implementation to be nearly 30 times faster than the ActionScript version [1]. The results of this performance comparison are outlined in Table 2. These computational gains open up myriad possibilities for developing interactive music-IR driven applications in the Flash framework.

Table 2. Comparison of FFT Computation Time for ActionScript and Alchemy-compiled C code in milliseconds.

Target Platform	FFT Size 8192	4096	2048	1024	512	256
ActionScript	45.157	20.818	9.276	4.460	2.041	0.925
Java	20.703	9.393	4.345	1.956	0.901	0.385
Alchemy-C	1.371	0.628	0.297	0.139	0.067	0.034

3.3 ALF Functions

The functions available in ALF are categorized as either "analysis" or "synthesis" and are outlined in Table 1. The analysis functions include several spectral processing routines and features, such as partial extraction and MFCCs, that are useful in many MIR tasks [9]. Synthesis functions are also available so that the developer can dynamically modify the audio output stream to achieve a desired effect. In a related paper, we discuss the implementation and algorithms used for the reverb and filter functions [2]. The remainder of this section will briefly discuss the implementation of two additional synthesis functions added to ALF: phase vocoding and beat tracking.

The most important consideration in developing the beat tracking algorithm was the stipulation that it run in real-time. Our beat tracking algorithm is based off of that proposed by Klapuri but uses an autocorrelation as opposed to a bank of comb filters for computational efficiency [10]. We first compute the power spectrum and separate it into six octave-based sub-bands. The energy envelope in each sub-band is calculated and the bandwise autocorrelation of these vectors is computed. Summing the resulting six autocorrelations and finding the highest peak after the zeroth lag yields an estimate of the tempo.

The phase vocoder is based on a popular, FFT-based implementation in which overlapping frames (specified by ALF's frame rate) are analyzed and re-synthesized using overlap-add in order to perform pitch and/or time-scale modification in real-time [11]. Each frame is processed by a FFT, which is used to determine the phase offset for each frequency bin and thus the estimated, true bin frequency. Pitch modification is achieved by multiplying the bin frequencies by a pitch shift factor, which shifts the audio's pitch in the desired manner after performing the IFFT. Time stretching is achieved by first applying the appropriate pitch-shift factor, performing an IFFT and and re-sampling the audio frame in the time domain to achieve the desired speed.

4. DEVELOPING WITH ALF

Many of the applications developed with ALF thus far have followed the same basic program structure, which is detailed in Figure 3. Input audio is analyzed on a per frame basis and feature values are returned in real-time for the

developer to incorporate into their application. Any additional processing required for synthesis functions is executed in a separate processing chain, which eliminates any computational overhead when synthesis functions are not required.

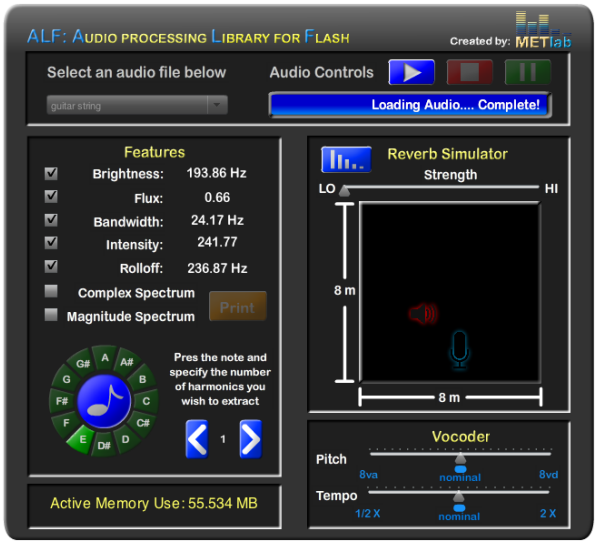


Figure 2. Application demonstrating ALF functionality.

The flexible nature of the architecture shown in Figure 3 combined with the low learning curve of Flash allows developers to rapidly create audio and music-based applications to serve a variety of target audiences and purposes. Possible applications include:

- Music-centric games requiring real-time feature extraction to drive the game environment
- Music exploration interfaces that group user libraries into categories (emotional, genre, etc.) based on extraction and comparison of audio features
- Educational activities for enhancing K-12 curricula in natural science and/or mathematics [12]

Currently, we have several applications developed using ALF for the purpose of audio-based experimentation, analysis/synthesis and music-driven games for entertainment, which we will discuss in the subsequent sections.

4.1 ALF Workbench

Figure 2 demonstrates the ALF Workbench, which allows developers to interactively experiment with different audio files and some of the functions available in ALF. The left panel of the interface showcases the spectral features, which are updated during audio playback and can be exported in a CSV file when the file completes. A pitch wheel is also shown, which allows the user to determine the chromatic notes present in the spectrum of tonal audio. The right panel of the work bench features the room reverb and phase vocoding functions. The reverb interface allows the user to manipulate the positions of the source and listener in a virtual room to simulate immersive environments.

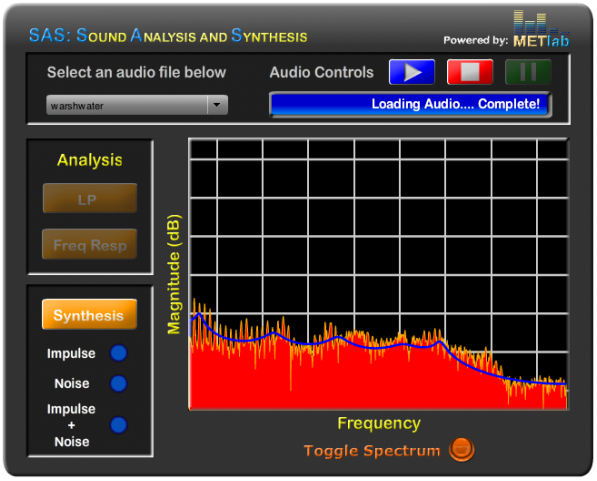


Figure 4. Sound analysis-synthesis app showing linear predictive analysis and magnitude spectrum of speech.

4.2 Beat-Sync-Mash-Coder

Recently, so-called artist "mashups", blending two or more songs in a creative way, have emerged as a popular form of expression for musicians and hobbyists. To this end, the *Beat-Sync-Mash-Coder* is a tool developed for semi-automated, real-time creation of beat-synchronous mashups [13]. This application utilizes the beat-tracking and phase vocoding functions available in ALF along with an intuitive, Flash-based GUI to help automate the task of synchronizing various clips without the complexities incurred with traditional digital audio workstations. The Beat-Sync-Mash-Coder is capable of sustaining real-time phase vocoding on 5-9 audio tracks, depending on the available hardware, thus allowing the user to create dynamic, intricate and musically coherent soundscapes.

4.3 Sound Analysis and Synthesis Application

The application depicted in Figure 4 uses ALF's analysis and synthesis capabilities to perform linear-predictive analysis on speech signals in order to re-synthesize it using different excitation signals. Linear prediction coefficients are extracted at each frame using the Levinson-Durbin recursion to obtain a time-varying model of the vocal tract [14]. The user can then simulate the effect of various excitation sources by using ALF's filtering function to sample the vocal tract with impulsive, noisy or mixed-spectra signals.

4.4 Applications For Music-Driven Gameplay

We present two novel music-driven games which resulted from a collaboration between departments at our university. Both games harness MIR functionality in ALF to create unique and immersive gameplay experiences.

4.4.1 Pulse

Pulse is a musically reactive, side-scrolling platform game that utilizes a player's personal music collection to drive the gameplay. Unlike other music games, which rely on

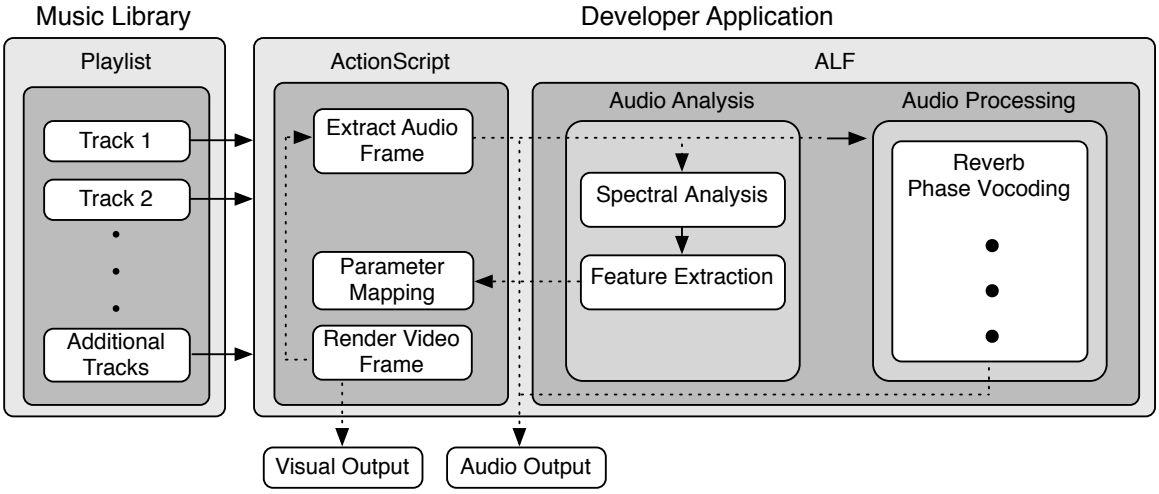


Figure 3. Typical implementation of an application using ALF.

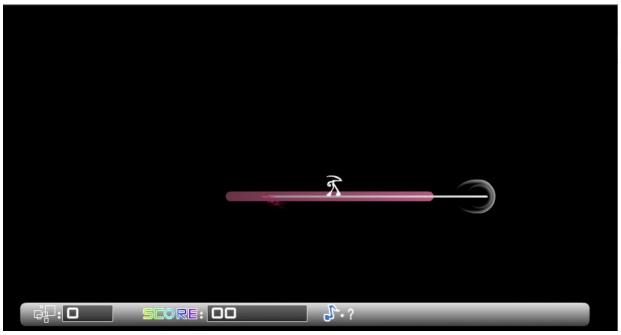
off-line audio analysis to determine the gaming environment, Pulse utilizes ALF functionality to update the game environment in real-time, mapping the quantitative features extracted from the audio to changes in the game's environment variables. This concept increases the replay value of Pulse, since the gamer's experience is limited only by the number of tracks in their music library.

By employing ALF's frame-based processing structure, ALF maps features extracted from the user-selected audio to environment parameters so they are updated in sync with the user-specified frame rate. To permit ample rendering time for the graphics, a "frame-look-ahead" parameter is specified which delays audio playback while features are accumulated from ALF functions. Game environment variables that react to changes in the game's audio include the background scenery, enemies and obstacles of the Pulse character as well as the slope of platform supporting the character. The effect of the audio on the gameplay is evident in Figure 5 where (a) shows the game screen when there is no audio playing and (b) is typical realization of the parameter mapping to game output.

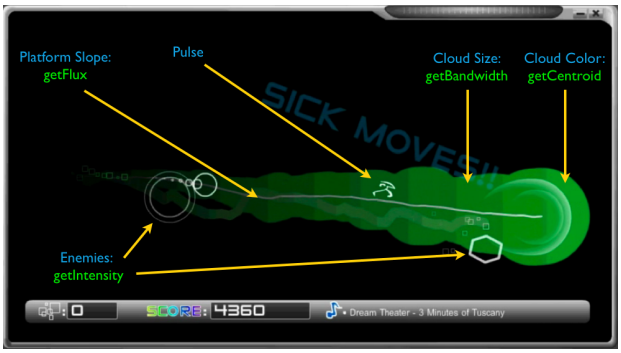
4.4.2 *Surge*

The concept behind *Surge* is to facilitate exploration of one's own music library though an interactive, DJ-style beat matching game. This expands the concept of audio feature-based gaming environments to include tempo analysis and modification of the game's music. Whereas gameplay in Pulse depends on audio features to dictate the environment, Surge uses game environment parameters to alter the audio in real-time.

The Surge game environment, shown in Figure 6, consists of planets that represent songs the player has provided from their music library. Each song is analyzed with ALF's beat tracker function so that the planet is associated with a song tempo. The game audio depends on which planet the player is on and their proximity to nearby planets. As the player nears a new planet, they will hear the music associated with the new planet. In order to move from planet-to-planet, the player (by moving their character) must adjust the rotation of their current planet (altering tempo and beats of the song) to match that of the target planet. That is, the music tempo is adjusted using ALF's phase vocoder according to the planet's rotation, which is dependent upon the player's actions in the game environment.



(a)



(b)

Figure 5. Pulse game environment during static (a) and dynamic (b) moments in the game's music.

5. FUTURE WORK

There are several features we would still like to add to ALF including spectral contrast features and other less com-

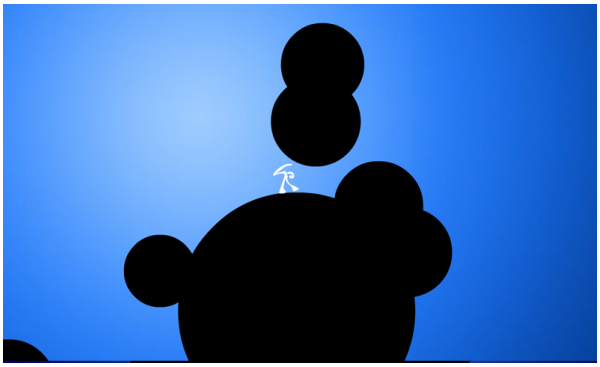


Figure 6. *Surge* game environment.

monly used statistical spectrum descriptors. The most significant component that would augment the usefulness of ALF for the music-IR community would be classification. There are many open source classification libraries available to perform common classification methods such as GMM, SVM, and naive Bayes classification that can be integrated into the current framework.

We will continue to emphasize the real-time capabilities of ALF and optimize the algorithms and architecture to ensure additional algorithms operate in real-time. The newest version of the Flash Player (10.1) will allow byte level access to the audio input creating potential for even further user interaction via real-time analysis and processing of voice/music input to a microphone or other audio device connected to a computer.

6. CONCLUSIONS

The Audio processing Library for Flash affords music-IR researchers the opportunity to generate rich, interactive, real-time music-IR driven applications. The various levels of complexity and control as well as the capability to execute analysis and synthesis simultaneously provide a means to generate unique programs that integrate content based retrieval of audio features. We have demonstrated the versatility and usefulness of ALF through the variety of applications described in this paper. As interest in music driven applications intensifies, it is our goal to enable the community of developers and researchers in music-IR and related fields to generate interactive web-based media.

7. REFERENCES

[1] T. M. Doll, R. Migneco, J. J. Scott, and Y. Kim, "An audio DSP toolkit for rapid application development in flash," in *IEEE International Workshop on Multimedia Signal Processing*, 2009.

[2] R. Migneco, T. Doll, J. Scott, C. Hahn, P. Diefenbach, and Y. Kim, "An audio processing library for game development in Flash," in *Proc. of the IEEE Games Innovations Conference (ICE-GIC 2009)*, Aug. 2009, pp. 201 –209.

[3] G. Tzanetakis and K. Lemstrom, "Marsyas-0.2: A case study in implementing music information retrieval systems," in *Intelligent Music Information Systems: Tools and Methodologies*. Information Science Reference, 2008, pp. 31–49.

[4] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle, "jAudio: A feature extraction library," in *Proc. of the 6th International Conference on Music Information Retrieval*. London, U.K.: ISMIR, 2005.

[5] J. S. Downie, A. F. Ehmann, and X. Hu, "Music-to-knowledge (M2K): a prototyping and evaluation environment for music digital library research," in *Proc. of the 5th ACM/IEEE-CS Joint Conf. on Digital Libraries*. New York, NY, USA: ACM, 2005, pp. 376–376.

[6] O. Lartillot, P. Toiviainen, and T. Eerola, *A Matlab Toolbox for Music Information Retrieval.*, ser. Studies in Classification, Data Analysis, and Knowledge Organization. Springer, 2007, pp. 261–268.

[7] X. Amatriain, M. De Boer, and E. Robledo, "CLAM: An OO framework for developing audio and music applications," in *Proc. of the 17th Annual Conference on Object-Oriented Programming, Systems, Languages and Applications*, 2002.

[8] A. Lerch, G. Eisenberg, and K. Tanghe, "FEAPI: A low level feature extraction plugin api," in *In Proc. of 8th Int. Conference on Digital Audio Effects (DaFX '05)*, 2005.

[9] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[10] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," in *IEEE Transactions Speech and Audio Processing*, 2004, pp. 342–355.

[11] M. Dolson, "The phase vocoder: A tutorial," in *Computer Music Journal*, vol. 10, no. 4. MIT Press, 1986, pp. 14–27.

[12] T. M. Doll, R. V. Migneco, and Y. E. Kim, "Online activities for music informatioin and acoustics education and psychoacoustics data collection," in *Proc. of the International Conference on Music Information Retrieval*. Philadelphia, PA: ISMIR, 2008.

[13] G. Griffin, Y. E. Kim, and D. Turnbull, "Beat-sync-mash-coder: A web application for real-time creation of beat-synchronous music mashups," in *Proc. of the IEEE Conf. on Acoustics, Speech, and Signal Processing*, 2010.

[14] T. F. Quatieri, *Discrete-Time Speech Signal Processing*, A. V. Oppenheim, Ed. Prentice Hall Signal Processing Series, 2002.

WHAT MAKES BEAT TRACKING DIFFICULT? A CASE STUDY ON CHOPIN MAZURKAS

Peter Grosche
Saarland University
and MPI Informatik
pgrosche@mpi-inf.mpg.de

Meinard Müller
Saarland University
and MPI Informatik
meinard@mpi-inf.mpg.de

Craig Stuart Sapp
Standford University
CCRMA / CCARH
craig@ccrma.stanford.edu

ABSTRACT

The automated extraction of tempo and beat information from music recordings is a challenging task. Especially in the case of expressive performances, current beat tracking approaches still have significant problems to accurately capture local tempo deviations and beat positions. In this paper, we introduce a novel evaluation framework for detecting critical passages in a piece of music that are prone to tracking errors. Our idea is to look for consistencies in the beat tracking results over multiple performances of the same underlying piece. As another contribution, we further classify the critical passages by specifying musical properties of certain beats that frequently evoke tracking errors. Finally, considering three conceptually different beat tracking procedures, we conduct a case study on the basis of a challenging test set that consists of a variety of piano performances of Chopin Mazurkas. Our experimental results not only make the limitations of state-of-the-art beat trackers explicit but also deepens the understanding of the underlying music material.

1. INTRODUCTION

When listening to a piece of music, most humans are able to tap to the musical beat without difficulty. In recent years, various different algorithmic solutions for automatically extracting beat position from audio recordings have been proposed. However, transferring this cognitive process into an automated system that reliably works for the large variety of musical styles is still not possible. Modern pop and rock music with a strong beat and steady tempo can be handled by many methods well, but extracting the beat locations from highly expressive performances of, *e.g.*, romantic piano music, is a challenging task.

To better understand the shortcomings of recent beat tracking methods, significant efforts have been made to compare and investigate the performance of different strategies on common datasets [6, 10, 13]. However, most approaches were limited to comparing the different methods by specifying evaluation measures that refer to an entire recording or even an entire collection of recordings. Such globally oriented evaluations do not provide any information on the critical passages within a piece where the tracking errors occur. Thus, no conclusions can be drawn from these experiments about possible *musical reasons* that lie behind the beat tracking errors. A first analysis of *musical properties* influencing the beat tracking quality was conducted by Dixon [6], who proposed quantitative measures for the rhythmic complexity and for variations in tempo and timings. However, no larger evaluations were carried out to show a correlation between these theoretical measures and the actual beat tracking quality.

In this paper, we continue this strand of research by analyzing the tracking results obtained by different beat tracking procedures. As one main idea of this paper, we introduce a novel evaluation framework that exploits the existence of different performances available for a given piece of music. For example, in our case study we revert to a collection of recordings for the Chopin Mazurkas containing in average over 50 performances for each piece. Based on a local, beat-wise histogram, we simultaneously determine consistencies of beat tracking errors over many performances. The underlying assumption is, that tracking errors consistently occurring in many performances of a piece are likely caused by musical properties of the piece, rather than physical properties of a specific performance. As a further contribution, we classify the beats of the critical passages by introducing various types of beats such as non-event beats, ornamented beats, weak bass beats, or constant harmony beats. Each such beat class stands for a musical performance-independent property that frequently evokes beat tracking errors. In our experiments, we evaluated three conceptually different beat tracking procedures on a corpus consisting of 300 audio recordings corresponding to five different Mazurkas. For each recording, the tracking results were compared with manually annotated ground-truth beat positions. Our local evaluation framework and detailed analysis explicitly indicates various limitations of current state-of-the-art beat trackers, thus laying the basis for future improvements and research directions.

This paper is organized as follows: In Sect. 2, we formalize and discuss the beat tracking problem. In Sect. 3, we describe the underlying music material and specify various beat classes. After summarizing the three beat tracking strategies (Sect. 4) and introducing the evaluation measure (Sect. 5) used in our case study, we report on the experimental results in Sect. 6. Finally, we conclude in

ID	Composer	Piece	#(Meas.)	#(Beats)	#(Perf.)
M17-4	Chopin	Op. 17, No. 4	132	396	62
M24-2	Chopin	Op. 24, No. 2	120	360	64
M30-2	Chopin	Op. 30, No. 2	65	193	34
M63-3	Chopin	Op. 63, No. 3	77	229	88
M68-3	Chopin	Op. 68, No. 3	61	181	50

Table 1: The five Chopin Mazurkas and their identifiers used in our study. The last three columns indicate the number of measures, beats, and performances available for the respective piece.

Sect. 7 with a discussion of future research directions. Further related work is discussed in the respective sections.

2. PROBLEM SPECIFICATION

For a given piece of music, let N denote the number of *musical beats*. Enumerating all beats, we identify the set of musical beats with the set $\mathcal{B} = [1:N] := \{1, 2, \ldots, N\}$. Given a performance of the piece in the form of an audio recording, the musical beats correspond to specific physical time positions within the audio file. Let $\pi : \mathcal{B} \to \mathbb{R}$ be the mapping that assigns each musical beat $b \in \mathcal{B}$ to the time position $\pi(b)$ of its occurrence in the performance. In the following, a time position $\pi(b)$ is referred to as *physical beat* or simply as *beat* of the performance. Then, the task of *beat tracking* is to recover the set $\{\pi(b) \,|\, b \in \mathcal{B}\}$ of all beats from a given audio recording.

Note that this specification of the beat tracking problem is somewhat simplistic, as we only consider physical beats that are defined by onset events. More generally, a beat is a perceptual phenomenon and perceptual beat times do not necessarily coincide with physical beat times [7]. Furthermore, the perception of beats varies between listeners.

For determining physical beat times, we now discuss some of the problems, one has to deal with in practice. Typically, a beat goes along with a note onset revealed by an increase of the signal's energy or a change in the spectral content. However, in particular for non-percussive music, one often has soft note onsets, which lead to blurred note transitions rather than sharp note onset positions. In such cases, there are no precise timings of note events within the audio recording, and the assignment of exact physical beat positions becomes problematic. This issue is aggravated in the presence of tempo changes and expressive tempo nuances (*e.g.*, ritardando and accelerando).

Besides such physical reasons, there may also be a number of musical reasons for beat tracking becoming a challenging task. For example, there may be beats with no note event going along with them. Here, a human may still perceive a steady beat, but the automatic specification of physical beat positions is quite problematic, in particular in passages of varying tempo where interpolation is not straightforward. Furthermore, auxiliary note onsets can cause difficulty or ambiguity in defining a specific physical beat time. In music such as the Chopin Mazurkas, the main melody is often embellished by ornamented notes such as trills, grace notes, or arpeggios. Also, for the sake of expressiveness, the notes of a chord need not be played at the same time, but slightly displaced in time. This renders a precise definition of a physical beat position impossible.

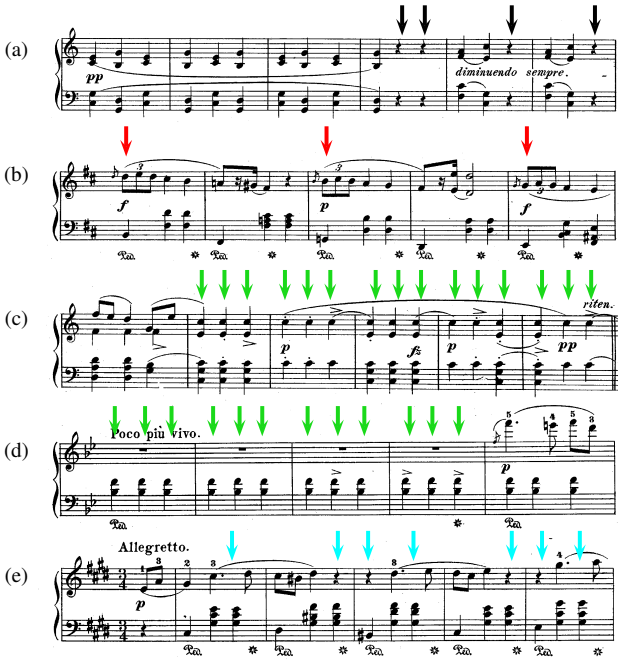


Figure 1: Scores of example passages for the different beat classes introduced in Sect. 3. **(a)** Non-event beats ($\mathcal{B}_1$) in M24-2, **(b)** Ornamented beats ($\mathcal{B}_3$) in M30-2, **(c)** Constant harmony beats ($\mathcal{B}_5$) in M24-2, **(d)** Constant harmony beats ($\mathcal{B}_5$) in M68-3, and **(e)** Weak bass beats ($\mathcal{B}_4$) in M63-3.

3. DATA AND ANNOTATIONS

The Mazurka Project [1] has collected over 2700 recorded performances for 49 Mazurkas by Frédéric Chopin, ranging from the early stages of music recording (Grünfeld 1902) until today [15]. In our case study, we use 298 recordings corresponding to five of the 49 Mazurkas, see Table 1. For each of theses recordings the beat positions were annotated manually [15]. These annotations are used as ground truth in our experiments. Furthermore, Humdrum and MIDI files of the underlying musical scores for each performance are provided, representing the pieces in an uninterpreted symbolic format.

In addition to the physical beat annotations of the performances, we created musical annotations by grouping the musical beats $\mathcal{B}$ in five different beat classes $\mathcal{B}_1$ to $\mathcal{B}_5$. Each of these classes represents a musical property that typically constitutes a problem for determining the beat positions. The colors refer to Fig. 4 and Fig. 5.

- **Non-event beats $\mathcal{B}_1$ (black)**: Beats that do not coincide with any note events, see Fig. 1(a).
- **Boundary beats $\mathcal{B}_2$ (blue)**: Beats of the first measure and last measure of the piece.
- **Ornamented beats $\mathcal{B}_3$ (red)**: Beats that coincide with ornaments such as trills, grace notes, or arpeggios, see Fig. 1(b).
- **Weak bass beats $\mathcal{B}_4$ (cyan)**: Beats where only the left hand is played, see Fig. 1(e).
- **Constant harmony beats $\mathcal{B}_5$ (green)**: Beats that correspond to consecutive repetitions of the same chord, see Fig. 1(c-d).

Furthermore, let $\mathcal{B}_* := \cup_{k=1}^{5} \mathcal{B}_k$ denote the union of the five beat classes. Table 2 details for each Mazurka the number of beats assigned to the respective beat classes.

ID	$\|\mathcal{B}\|$	$\|\mathcal{B}_1\|$	$\|\mathcal{B}_2\|$	$\|\mathcal{B}_3\|$	$\|\mathcal{B}_4\|$	$\|\mathcal{B}_5\|$	$\|\mathcal{B}_*\|$
M17-4	396	9	8	51	88	0	154
M24-2	360	10	8	22	4	12	55
M30-2	193	2	8	13	65	0	82
M63-3	229	1	7	9	36	0	47
M68-3	181	17	7	0	14	12	37

Table 2: The number of musical beats in each of the different beat classes defined in Sect. 3. Each beat may be a member of more than one class.

Note that the beat classes need not be disjoint, *i.e.*, each beat may be assigned to more than one class. In Sect. 6, we discuss the beat classes and their implications on the beat tracking results in more detail.

4. BEAT TRACKING STRATEGIES

Beat tracking algorithms working on audio recordings typically proceed in three steps: In the first step, note onset candidates are extracted from the signal. More precisely, a *novelty curve* is computed that captures changes of the signal's energy, pitch or spectral content [3, 5, 8, 12]. The peaks of this curve indicate likely note onset candidates. Fig. 2(c) shows a novelty curve for an excerpt of M17-4 (identifier explained in Table 1). Using a peak picking strategy [3] note onsets can be extracted from this curve. In the second step, the local tempo of the piece is estimated. Therefore, the onset candidates are analyzed with respect to locally periodic or reoccurring patterns [5, 12, 14]. The underlying assumption is that the tempo of the piece does not change within the analysis window. The choice of the window size constitutes a trade-off between the robustness of the tempo estimates and the capability to capture tempo changes. In the third step, the sequence of beat positions is determined that best explains the locally periodic structure of the piece, in terms of frequency (tempo) and phase (timing) [5, 12], see Fig. 2(d).

In our experiments we use three different beat trackers. First, we directly use the onset candidates extracted from a novelty curve capturing spectral differences [11] as indicated by Fig. 2(c). In this method, referred to as ONSET in the following sections, each detected note onset is considered as a beat position. Second, as a representative of the beat tracking algorithms that transform the novelty curve into the frequency (tempo) or periodicity domain [5, 12, 14], we employ the predominant local periodicity estimation [11], referred to as PLP in the following. We use a window size of three seconds and initialize the tempo estimation with the mean of the annotated tempo. More precisely, we define the global tempo range for each performance covering one octave around the mean tempo, *e.g.*, for a mean tempo of 120 BPM, tempo estimates in the range $[90 : 180]$ are valid. This prevents tempo doubling or halving errors and robustly allows for investigating beat tracking errors, rather than tempo estimation errors. The third beat tracking method (SYNC) we use in our experiments employs the MIDI file available for each piece. This MIDI file can be regarded as additional knowledge, including the pitch, onset time and duration of each note. Using suitable synchronization techniques [9] on the basis of coarse harmonic and very precise onset information,

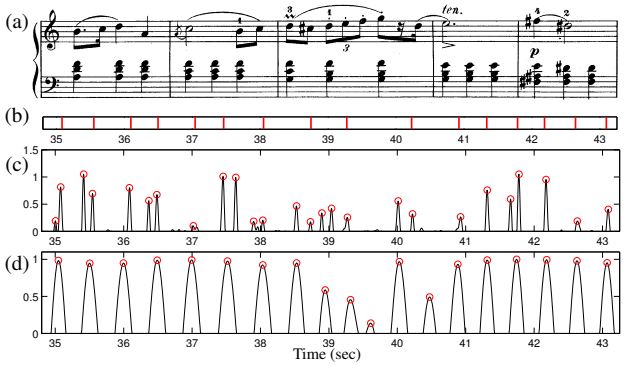


Figure 2: Representations for an excerpt of M17-4. **(a)** Score representation of beats 60 to 74. **(b)** Annotated ground truth beats for the performance pid50534-05 by Horowitz (1985), see [1]. **(c)** Novelty curve (note onset candidates indicated by circles). **(d)** PLP curve (beat candidates indicated by circles).

we identify for each musical event of the piece (given by the MIDI file) the corresponding physical position within a performance. This coordination of MIDI events to the audio is then used to determine the beat positions in a performance and simplifies the beat tracking task to an alignment problem, where the number of beats and the sequence of note events is given as prior knowledge.

5. EVALUATION MEASURES

Many evaluation measures have been proposed to quantify the performance of beat tracking systems [4] by comparing the beat positions determined by a beat tracking algorithm and annotated ground truth beats. These measures can be divided into two groups. Firstly, measures that analyze each beat position separately and secondly, measures that take the tempo and metrical levels into account [5, 12, 13]. While the latter gives a better estimate of how well a *sequence* of retrieved beats correlates with the manual annotation, it does not give any insight into the beat tracking performance at a specific beat of the piece.

In this paper, we evaluate the beat tracking quality on the beat-level of a piece and combine the results of all performances available for this piece. This allows for detecting beats that are prone to errors in many performances. For a given performance, let $\Pi := \{\pi(b) \,|\, b \in \mathcal{B}\}$ be the set of manually determined physical beats, which are used as ground truth. Furthermore, let $\Phi \subset \mathbb{R}$ be the set of beat candidates obtained from a beat tracking procedure. Given a tolerance parameter $\tau > 0$, we define the τ-neighborhood $I_\tau(p) \subset \mathbb{R}$ of a beat $p \in \Pi$ to be the interval of length 2τ centered at p, see Fig. 3. We say that a beat p has been *identified* if there is a beat candidate $q \in \Phi$ in the τ-neighborhood of p, *i.e.*, $q \in \Phi \cap I_\tau(p)$. Let $\Pi_{\mathrm{id}} \subset \Pi$ be the set of all identified beats. Furthermore, we say that a beat candidate $q \in \Phi$ is *correct* if q lies in the τ-neighborhood $I_\tau(p)$ of some beat $p \in \Pi$ and there is no other beat candidate lying in $I_\tau(p)$ that is closer to p than q. Let $\Phi_{\mathrm{co}} \subset \Phi$ be the set of all correct beat candidates. We then define the precision $\mathrm{P} = \mathrm{P}_\tau$, the recall $\mathrm{R} = \mathrm{R}_\tau$, and F-measure $\mathrm{F} = \mathrm{F}_\tau$ as [4]

$$\mathrm{P} = \frac{|\Phi_{\mathrm{co}}|}{|\Phi|}, \quad \mathrm{R} = \frac{|\Pi_{\mathrm{id}}|}{|\Pi|}, \quad \mathrm{F} = \frac{2 \cdot \mathrm{P} \cdot \mathrm{R}}{\mathrm{P} + \mathrm{R}}. \qquad (1)$$

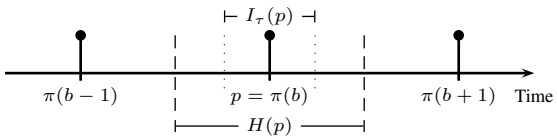


Figure 3: Illustration of the τ-neighborhood $I_\tau(p)$ and the half-beat neighborhood $H(p)$ of a beat $p = \pi(b)$, $b \in \mathcal{B}$.

Table 3 shows the results of various beat tracking procedures on the Mazurka data. As it turns out, the F-measure is a relatively soft evaluation measure that only moderately punishes additional, non-correct beat candidates. As a consequence, the simple onset-based beat tracker seems to outperform most other beat trackers. As for the Mazurka data, many note onsets coincide with beats, the onset detection leads to a high recall, while having only a moderate deduction in the precision.

We now introduce a novel evaluation measure that punishes non-correct beat candidates, which are often musically meaningless, more heavily. To this end, we define a *half-beat neighborhood* $H(p)$ of a beat $p = \pi(b) \in \Pi$ to be the interval ranging from $\frac{\pi(b-1)-\pi(b)}{2}$ (or $\pi(b)$ for $b = 1$) to $\frac{\pi(b+1)-\pi(b)}{2}$ (or $\pi(b)$ for $b = N$), see Fig. 3. Then, we say that a beat $b \in \mathcal{B}$ has been *strongly identified* if there is a beat candidate $q \in \Phi$ with $q \in \Phi \cap I_\tau(p)$ and if $H(p) \cap \Phi = \{q\}$ for $p = \pi(b)$. In other words, q is the only beat candidate in the half-beat neighborhood of p. Let $\Pi_{\text{stid}} \subset \Pi$ be the set of all strongly identified beats, then we define the *beat accuracy* $A = A_\tau$ to be

$$A = \frac{|\Pi_{\text{stid}}|}{|\Pi|}. \quad (2)$$

6. EXPERIMENTS

We now discuss the experimental results obtained using our evaluation framework and explain the relations between the beat tracking results and the beat classes introduced in Sect. 3.

We start with discussing Table 3. Here, the results of the different beat tracking approaches for all performances of the five Mazurkas are summarized, together with some results from the MIREX 2009 beat tracking task [2]. All beat trackers used in our evaluation yield better results for the Mazurkas than all trackers used in the MIREX evaluation. As noted before, the F-measure only moderately punishes additional beats. In consequence, ONSET ($\mathrm{F} = 0.754$) seems to outperform all other methods, except SYNC ($\mathrm{F} = 0.890$). In contrast, the introduced beat accuracy A punishes false positives more heavily, leading to $A = 0.535$ for ONSET, which is significantly lower than for PLP ($A = 0.729$) and SYNC ($A = 0.890$). For SYNC, the evaluation metrics P, R, F, and A are equivalent because the number of detected beats is always correct. Furthermore, SYNC is able to considerably outperform the other strategies. This is not surprising, as it is equipped with additional knowledge in the form of the MIDI file.

There are some obvious differences in the beat tracking results of the individual Mazurkas caused by the musical reasons explained in [6]. First of all, all methods deliver

	SYNC	ONSET				PLP			
ID	P/R/F/A	P	R	F	A	P	R	F	A
M17-4	0.837	0.552	0.958	0.697	0.479	0.615	0.743	0.672	0.639
M24-2	0.931	0.758	0.956	0.845	0.703	0.798	0.940	0.862	0.854
M30-2	0.900	0.692	0.975	0.809	0.623	0.726	0.900	0.803	0.788
M63-3	0.890	0.560	0.975	0.706	0.414	0.597	0.744	0.661	0.631
M68-3	0.875	0.671	0.885	0.758	0.507	0.634	0.755	0.689	0.674
Mean:	**0.890**	0.634	0.952	**0.754**	0.535	0.665	0.806	**0.728**	0.729

	MIREX				Our Methods		
Method	DRP3	GP2	OGM2	TL	SYNC	ONSET	PLP
F	0.678	0.547	0.321	0.449	**0.890**	**0.754**	**0.728**

Table 3: Comparison of the beat tracking performance of the three strategies used in this paper and the MIREX 2009 results (see [2] for an explanation) based on the evaluation metrics Precision P, Recall R, F-measure F and the beat accuracy A.

the best result for M24-2. This piece is rather simple, with many quarter notes in the dominant melody line. M17-4 is the most challenging for all three trackers because of a frequent use of ornaments and trills and many beat positions that are not reflected in the dominating melody line. For the ONSET tracker, M63-3 constitutes a challenge ($A = 0.414$), although this piece can be handled well by the SYNC tracker. Here, a large number of notes that do not fall on beat positions provoke many false positives. This also leads to a low accuracy of PLP ($A = 0.631$).

Going beyond this evaluation on a piece-level, Fig. 4 and Fig. 5 illustrate the beat-level beat tracking results of our evaluation framework for the SYNC and PLP strategy, respectively. Here, for each beat $b \in \mathcal{B}$ of a piece, the bar encodes for how many of the performances of this piece the beat was not *strongly identified* (see Sect. 5). High bars indicate beats that are incorrectly identified in many performances, low bars indicate beats that are identified in most performances without problems. As a consequence, this representation allows for investigating the musical properties leading to beat errors. More precisely, beats that are consistently wrong over a large number of performances of the same piece are likely to be caused by musical properties of the piece, rather than physical properties of a specific performance. For example, for both tracking strategies (SYNC and PLP) and all five pieces, the first and last beats are incorrectly identified in almost all performances, as shown by the blue bars ($\mathcal{B}_2$). This is caused by boundary problems and adaption times of the algorithms.

Furthermore, there is a number of significant high bars within all pieces. The SYNC strategy for M68-3 (see Fig. 4) exhibits a number of isolated black bars. These non-event beats do not fall on any note-event ($\mathcal{B}_1$). As stated in Sect. 2, especially when dealing with expressive music, simple interpolation techniques do not work to infer these beat positions automatically. The same beat positions are problematic in the PLP strategy, see Fig. 5. For M30-2 (Fig. 4) most of the high bars within the piece are assigned to $\mathcal{B}_3$ (red). These beats, which coincide with ornaments such as trills, grace notes, or arpeggios are physically not well defined and hard to determine. For the Mazurkas, chords are often played on-beat by the left hand. However, for notes of lower pitch, onset detection is problematic, especially when played softly. As a consequence, beats that only coincide with a bass note or chord, but without any note being played in the main melody, are a frequent source

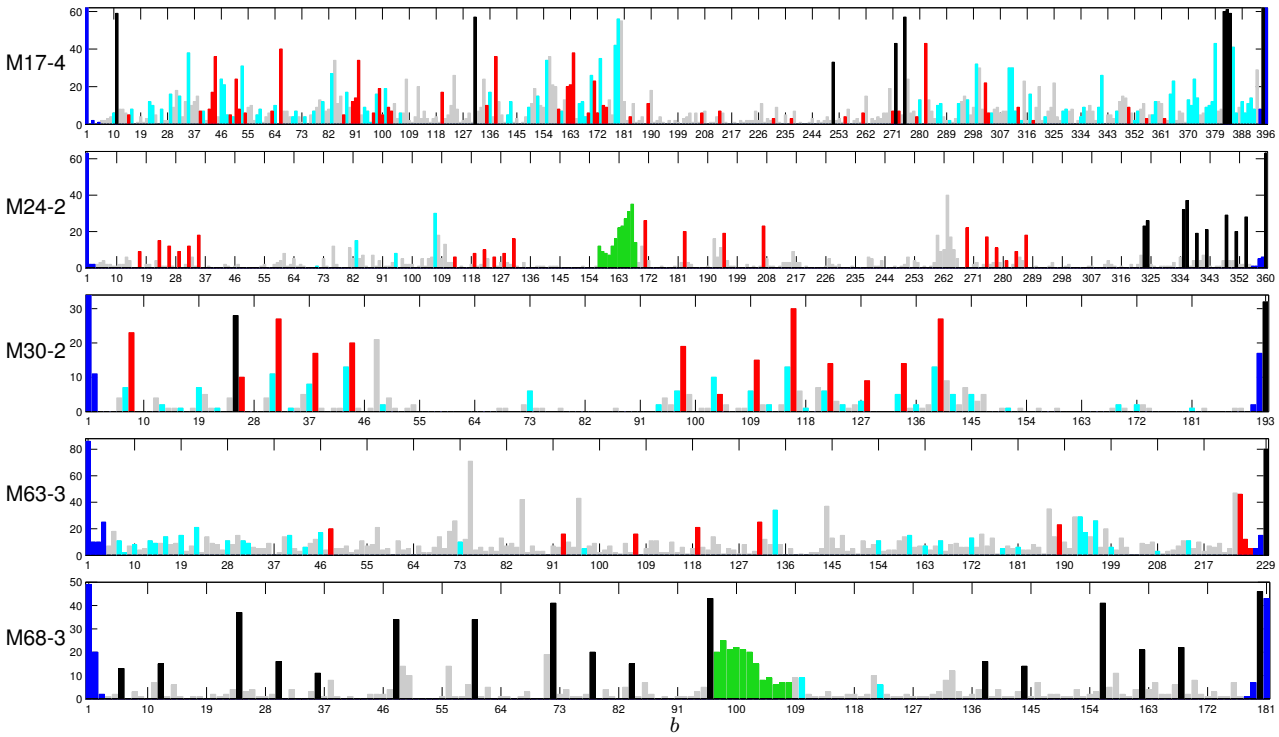


Figure 4: The beat error histogram for the synchronization based beat tracking (SYNC) shows for how many performances of each of the five Mazurkas a beat b is not identified. The different colors of the bars encode the beat class $\mathcal{B}$ a beat is assigned to, see Sect. 3.

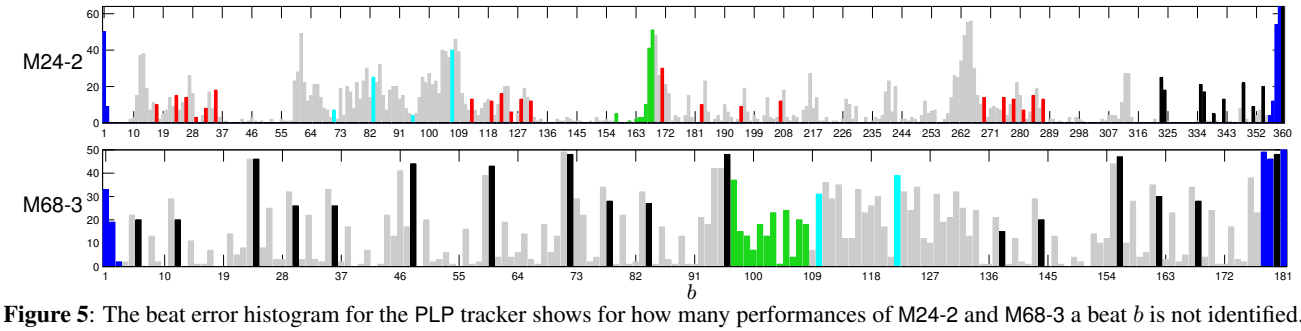


Figure 5: The beat error histogram for the PLP tracker shows for how many performances of M24-2 and M68-3 a beat b is not identified. The different colors of the bars encode the beat class $\mathcal{B}$ a beat is assigned to, see Sect. 3.

for errors. This is reflected by the cyan bars ($\mathcal{B}_3$) frequently occurring in M17-4 (Fig. 4). Finally, $\mathcal{B}_5$ (green) contains beats falling on consecutive repetitions of the same chord. This constitutes a challenge for the onset detection, especially when played softly. Both M24-2 and M68-3 exhibit a region of green bars that are incorrectly tracked by the SYNC (Fig. 4) and PLP (Fig. 5) trackers.

As mentioned in Sect. 4, PLP can not handle tempo changes well. As a consequence, many of the beat errors for PLP that are not assigned to any beat class (*e.g.*, M24-2 in Fig. 5, $b = [260 : 264]$) are caused by sudden tempo changes appearing in many of the performances. However, these are considered a performance-dependent property, rather than a piece-dependent musical property and are not classified in a beat class.

Table 4 summarizes the effect of each beat class on the piece-level results. Here, the mean beat accuracy is reported for each of the five Mazurkas, when excluding the beats of a certain class. For example, M30-2 contains many beats of $\mathcal{B}_3$. Excluding these ornamented beats from the evaluation, the overall beat accuracy increases from $A = 0.900$ to $A = 0.931$ for SYNC (Table 4 (left)) and from 0.788 to 0.814 for PLP (Table 4 (right)). The challenge of M68-3 however, are non-event beats ($\mathcal{B}_1$). Leaving out these beats, the accuracy increases from 0.875 to 0.910 for SYNC and from 0.674 to 0.705 for PLP.

Aside from musical properties of a piece causing beat errors, physical properties of certain performances make beat tracking difficult. In the following, we exemplarily compare the beat tracking results of the performances of M63-3. Fig. 6 shows the beat accuracy A for all 88 performances available for this piece. In case of the SYNC tracker, the beat accuracy for most of the performances is in the range of $0.8 - 0.9$, with only few exceptions that deviate significantly (Fig. 6(a)). In particular, Michalowski's 1933 performance with index 39 (pid9083-16, see [1]) shows a low accuracy of only $A = 0.589$ due to a poor condition of the original recording which contains a low signal-to-noise ratio and many clicks. The low accuracy ($A = 0.716$) of performance 1 (Csalog 1996, pid1263b-12) is caused by a high amount of reverberation, which makes a precise determination of the beat positions hard. The poor result of performance 81 (Zak 1951, pid918713-20) is caused by a detuning of the piano. Compensating

ID	$\mathcal{B}$	$\mathcal{B}\backslash\mathcal{B}_1$	$\mathcal{B}\backslash\mathcal{B}_2$	$\mathcal{B}\backslash\mathcal{B}_3$	$\mathcal{B}\backslash\mathcal{B}_4$	$\mathcal{B}\backslash\mathcal{B}_5$	$\mathcal{B}\backslash\mathcal{B}_*$
M17-4	0.837	0.852	0.842	0.843	0.854	0.837	0.898
M24-2	0.931	0.940	0.936	0.941	0.933	0.939	0.968
M30-2	0.900	0.900	0.903	0.931	0.905	0.900	0.959
M63-3	0.890	0.890	0.898	0.895	0.895	0.890	0.911
M68-3	0.875	0.910	0.889	0.875	0.875	0.887	0.948
Mean:	0.890	0.898	0.894	0.897	0.894	0.892	0.925

ID	$\mathcal{B}$	$\mathcal{B}\backslash\mathcal{B}_1$	$\mathcal{B}\backslash\mathcal{B}_2$	$\mathcal{B}\backslash\mathcal{B}_3$	$\mathcal{B}\backslash\mathcal{B}_4$	$\mathcal{B}\backslash\mathcal{B}_5$	$\mathcal{B}\backslash\mathcal{B}_*$
M17-4	0.639	0.650	0.641	0.671	0.593	0.639	0.649
M24-2	0.854	0.857	0.862	0.857	0.856	0.854	0.873
M30-2	0.788	0.788	0.794	0.814	0.772	0.788	0.822
M63-3	0.631	0.631	0.638	0.639	0.647	0.631	0.668
M68-3	0.674	0.705	0.689	0.674	0.678	0.674	0.733
Mean:	0.729	0.735	0.734	0.739	0.723	0.729	0.751

Table 4: Beat accuracy A results comparing the different beat classes for SYNC **(left)** and PLP **(right)**: For all beats $\mathcal{B}$, excluding non-event beats $\mathcal{B}_1$, boundary beats $\mathcal{B}_2$, ornamented beats $\mathcal{B}_3$, weak bass beats $\mathcal{B}_4$, constant harmony beats $\mathcal{B}_5$, and the union $\mathcal{B}_*$.

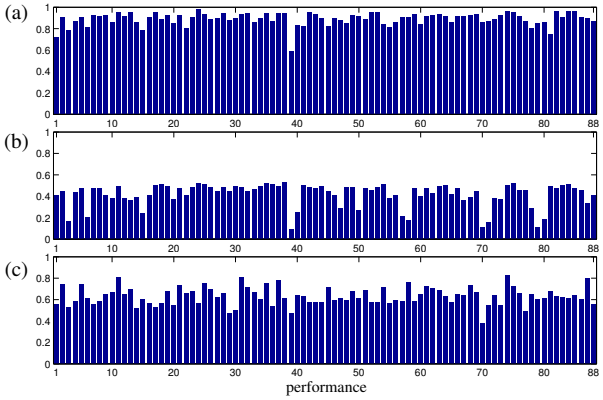


Figure 6: Beat accuracy A for the beat tracker SYNC **(a)**, ONSET **(b)**, and PLP **(c)** of all 88 performances of M63-3.

for this tuning effect, the synchronization results and thus, the beat accuracy improves from $A = 0.767$ to $A = 0.906$. As it turns out, ONSET tends to be even more sensitive to bad recording conditions. Again, performance 39 shows an extremely low accuracy ($A = 0.087$), however, there are more recordings with a very low accuracy (70, 71, 79, 80, 57, and 58). Further inspection shows that all of these recordings contain noise, especially clicks and crackling, which proves devastating for onset detectors and leads to a high number of false positives. Although onset detection is problematic for low quality recordings, the PLP approach shows a different behavior. Here, the periodicity enhancement of the novelty curve [11] provides a cleaning effect and is able to eliminate many spurious peaks caused by recording artifacts and leads to a higher beat accuracy. However, other performances suffer from a low accuracy (performances 29, 30, and 77). As it turns out, these examples exhibit extreme local tempo changes that can not be captured well by the PLP approach, which relies on a constant tempo within the analysis window. On the other hand, some performances show a noticeably higher accuracy (2, 5, 11, 31, 74, and 87). All oft these recordings are played in a rather constant tempo.

7. FUTURE DIRECTIONS

Our experiments indicate that our approach of considering multiple performances simultaneously for a given piece of music for the beat tracking task yields a better understanding not only of the algorithms' behavior but also of the underlying music material. The understanding and consideration of the physical and musical properties that make beat tracking difficult is of essential importance for improving the performance of beat tracking approaches. Exploiting the knowledge of the musical properties leading to beat errors one can design suited audio features. For example, in the case of the Mazurkas, a separation of bass and melody line can enhance the quality of the novelty curve and alleviate the negative effect of the ornamented beats or weak bass beats.

Acknowledgment. The first two authors are supported by the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University. The raw evaluation was generated by the third author at AHRC Centre for the History and Analysis of Recorded Music (CHARM), Royal Hollway, University of London.

8. REFERENCES

[1] The Mazurka Project. http://www.mazurka.org.uk, 2010.

[2] MIREX 2009. Audio beat tracking results. http://www.music-ir.org/mirex/2009/index.php/Audio_Beat_Tracking_Results, 2009.

[3] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. E. P. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Trans. on Speech and Audio Processing*, 13(5):1035–1047, 2005.

[4] M. E. P. Davies, N. Degara, and M. D. Plumbley. Evaluation methods for musical audio beat tracking algorithms. Technical Report C4DM-TR-09-06, Queen Mary University, Centre for Digital Music, 2009.

[5] M. E. P. Davies and M. D. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Trans. on Audio, Speech and Language Processing*, 15(3):1009–1020, 2007.

[6] S. Dixon. An empirical comparison of tempo trackers. In *Proc. of Brazilian Symposium on Computer Music*, pages 832–840, 2001.

[7] S. Dixon and W. Goebl. Pinpointing the beat: Tapping to expressive performances. In *Proc. of International Conference on Music Perception and Cognition*, pages 617–620, Sydney, Australia, 2002.

[8] A. Earis. An algorithm to extract expressive timing and dynamics from piano recordings. *Musicae Scientiae*, 11(2), 2007.

[9] S. Ewert, M. Müller, and P. Grosche. High resolution audio synchronization using chroma onset features. In *Proc. of IEEE ICASSP*, Taipei, Taiwan, 2009.

[10] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Trans. on Speech and Audio Processing*, 14, 2006.

[11] P. Grosche and M. Müller. A mid-level representation for capturing dominant tempo and pulse information in music recordings. In *Proc. of ISMIR*, pages 189–194, Kobe, Japan, 2009.

[12] A. P. Klapuri, A. J. Eronen, and J. Astola. Analysis of the meter of acoustic musical signals. *IEEE Trans. on Audio, Speech and Language Processing*, 14(1):342–355, 2006.

[13] M. F. McKinney, D. Moelants, M. E. P. Davies, and A. Klapuri. Evaluation of audio beat tracking and music tempo extraction algorithms. *Journal of New Music Research*, 36(1):1–16, 2007.

[14] G. Peeters. Template-based estimation of time-varying tempo. *EURASIP Journal on Advances in Signal Processing*, 2007.

[15] C. S. Sapp. Hybrid numeric/rank similarity metrics. In *Proc. of ISMIR*, pages 501–506, Philadelphia, USA, 2008.

UPBEAT AND QUIRKY, WITH A BIT OF A BUILD: INTERPRETIVE REPERTOIRES IN CREATIVE MUSIC SEARCH

Charlie Inskip
Dept of Info Science,
City University London
c.inskip@city.ac.uk

Andy MacFarlane
Dept of Info Science,
City University London
andym@soi.city.ac.uk

Pauline Rafferty
Dept of Info Studies,
University of Aberystwyth
pmr@aber.ac.uk

ABSTRACT

Pre-existing commercial music is widely used to accompany moving images in films, TV commercials and computer games. This process is known as music synchronisation. Professionals are employed by rights holders and film makers to perform creative music searches on large catalogues to find appropriate pieces of music for synchronisation. This paper discusses a Discourse Analysis of thirty interview texts related to the process. Coded examples are presented and discussed. Four interpretive repertoires are identified: the Musical Repertoire, the Soundtrack Repertoire, the Business Repertoire and the Cultural Repertoire. These ways of talking about music are adopted by all of the community regardless of their interest as Music Owner or Music User.

Music is shown to have multi-variate and sometimes conflicting meanings within this community which are dynamic and negotiated. This is related to a theoretical feedback model of communication and meaning making which proposes that Owners and Users employ their own and shared ways of talking and thinking about music and its context to determine musical meaning. The value to the music information retrieval community is to inform system design from a user information needs perspective.

1. INTRODUCTION

The record and music publishing industries and artists and writers benefit financially from secondary exploitation of their copyrights when they are used in films, TV shows, advertising and computer games. This process is known as music synchronisation, or 'sync'. The professional music Users employ specialists to search large catalogues for pre-existing commercial music in conjunction with the Owners' in-house specialists. Often these creative music searches are based on an ever-changing written query, or 'brief', which is sometimes accompanied by a moving visual clip or still images. [1]

The major Owners have attempted to disintermediate this process somewhat by developing and maintaining web-based applications which search their catalogues. These mainly use controlled vocabularies to explore databases of textual metadata linked to the relevant audio files. As would be expected, the metadata fields used in these applications include bibliographic information such as Artist, Title, Year and Chart position. Additionally they recognize the need for the Users to search for unknown items, and include more descriptive domain-based fields such as Mood, Genre, Tempo and Subject. Cataloguing is done by hand [2].

This paper presents a Discourse Analysis of thirty face-to-face interviews with professionals involved in sync in the UK. These semi-structured interviews have taken place over a period of two years as part of a wider investigation into the communication processes and information needs of this group of under-researched creative music searchers. The aim of the paper is to present an analysis of these texts which identifies the various interpretive repertoires used by this community of specialist users. A range of ways of talking about music is discussed, derived from a Discourse Analytic approach. The repertoires are adopted throughout the community and no repertoire is exclusive to one type of stakeholder. The varying discourses represent different ways of constructing reality and reveal important factors which may contribute to the design of music information retrieval systems for the purpose of music synchronisation.

Publications discussing qualitative research of user information needs traditionally bemoan the fact that there is little work in this area. However awareness of user needs and behaviour keeps users on the ISMIR radar, even though they are not usually the focus of reported research. Generally focus is on tagging and certain aspects of evaluation, such as ground truth and playlist evaluation. However in [3] the word 'user' does not appear in any top ten lists for ISMIR paper titles over the ten years of the conference, nor, indeed, in the top 20 bi-grams from titles and abstracts. Nevertheless, applying 'music information need' or 'user behaviour' as a query to the ISMIR Cloud Browser [4] does generate a range of relevant work focusing on user information needs such as [5,6,7]. This paper is situated within the user information needs paradigm and reflects the call at ISMIR 2009 [8] for the community to meet a number of challenges, the first identified being "*ISMIR needs to more actively encourage the participation of potential users of music-IR systems.*" [8]

The next section introduces and describes the methodology. This is followed by a summary of the findings and some examples of the coding and analytic process. In the final section the implications of the use of these repertoires are discussed, applying them to a theoretical model,

and suggestions are made as to how this work may be relevant to the music information retrieval community.

2. METHODOLOGY

In Discourse Analysis (DA), language is seen to construct reality, rather than simply reflect and describe it [9]. There are numerous methodologies under the DA umbrella, which vary widely in the amount of detail in which they look at the texts being considered [10]. Texts may be any written or spoken form of interaction, including interviews and other documents which are related to the subject in question. The linguistic approach identifies pauses and hesitations and detailed lexicographic units, while the social psychology approach, used here, seeks to identify attitudes, beliefs and attributions [9]. Interpretive repertoires are described as *"a lexicon or register of terms and metaphors drawn upon to characterize and evaluate actions and events"* [9:138]. Although there is no 'recipe' [11] for identifying interpretive repertoires [12] there is a developing DA literature in the library and information studies and human computer interaction domains [13-18].

Since October 2007, 23 professionals directly involved in searching for music to accompany moving images have taken part in semi-structured interviews. Seven people were observed while making relevance judgments, three of whom had previously been interviewed [19]. The sample was derived using snowball sampling [20], where each participant in the research recommended a small number of people to approach for the next interview. This method allows access to previously hidden communities and distances the sample from the researcher's preconceived ideas of who may be relevant. All participants were provided with an explanatory statement detailing and contextualizing the research project and gave informed consent. Interviews and observations lasted up to one hour, were recorded digitally and transcribed using MS Word. The transcriptions were then imported into NVivo software [21] and coded manually by the corresponding author, ensuring consistency.

The objective of the analysis was to identify interpretive repertoires within the interview and observation texts, highlighting the ways in which this community of varied-interest stakeholders talk about music. Interpretive repertoires are drawn from and used by a wide community of interest. One viewpoint of DA is that no one participant will be consistent in their talk, and the researcher is likely to find consistencies and variability not only between texts, which may be expected, but also within them. These consistencies and contradictions are drawn from a variety of repertoires which represent different ways of thinking about something [11,12], in this case, music. All of the participants are talking about searching for music in large collections and using music with moving images. However some of them are rights holders and their intermediaries (Owners) while others are music supervisors and film makers (Users). Each group draws from the other's repertoires in their music talk. Analyzing these repertoires in detail should identify more than one way of talking about music, informing work on meaning making in creative music search.

For the purposes of analysis there were two iterations of coding. On the first pass examples of 'talk about music' were identified. These were marked up using the coding facility in NVivo. This enables the researcher to tag highlighted text elements with bespoke codes and then extract, sort and analyse data tagged under specific codes in order to spot patterns, word and tag frequencies etc.. All the sections of text coded as 'talk about music' were then examined to determine how music was being described. Previous work had identified two broad groups of facets used in sync search engines [2] and user sync queries [22]: Bibliographic (content-based) and Descriptive (contextual). These facets were used as a starting point for the coding. There seemed to be more of a focus on Bibliographic data (eg Artist, Title) in the Owners' search engines while the Users' queries were more based on Descriptive language (eg Mood, Novelty).

3. IDENTIFIED REPERTOIRES

The language within each 'talk about music' section was carefully considered. This close reading of the transcriptions brought to light ways of talking about music that did not fit into either Bibliographic or Descriptive talk. It was found that a total of four types of language were consistently employed. These were identified by contradictions within or between texts or signalled by regularly-arising metaphors or phrases. Contradictions can be resolved by acknowledging a participant is switching repertoire and acknowledging the existence of more than one point of view. It is widely agreed in DA that this is a strong indication of interpretive repertoires. The words and phrases were divided into categories based on their themes, and coded within the interview texts (Table 1). Each theme, or repertoire, positions music differently in a users' world view. These are presented below as four interpretive repertoires, which have been named the Musical Repertoire, the Business Repertoire, the Soundtrack Repertoire and the Culture Repertoire.

3.1 The Musical Repertoire

In this repertoire, music is *an asset which is created and has identifiable characteristics*. The repertoire is identified by the appearance of bibliographic musical keywords (Table 1) , such as 'artist', 'title', 'instrumental', 'lyrics'. These familiar facets are commonly used to identify a piece of music. However, they relate more to how the Owners identify the music in their catalogues than how the musical elements are matched to a visual. Referring to an analysis of the Owners' bespoke search engines [2] these facets identify a recording or a composition and

help to isolate it within a large catalogue of recordings or compositions. The record companies and music publishers responsible for curating commercial music catalogues and exploiting recordings and compositions use these 'traditional' musical library facets when organizing their materials.

3.2 The Business Repertoire

In the Business Repertoire, music is *a large collection of recordings which are marketable, contractual and negotiable and have monetary value to the Owner*. There are a number of facets relating to music talk that are not immediately obviously musical, but they are important in exploitation terms nonetheless. These criteria are more concerned with business issues relating to signing, exploiting, and licensing music and include such keywords as "*license"* and "*clearance"*". They also employ the words used to sell the music to consumers, such as "*brand new"*, and "*cool*". The size of a catalogue is very important in this repertoire.

There are frequent co-locations of physical metaphors when the Business Repertoire is used: "*work with it", "at the coalface", "splattering", "wall-to-wall", "throw music up against it", "dig it out", "churn up a ton of songs", "trawl through a catalogue"*. These physical metaphors indicate the way of thinking that music is a physical capital resource for the Owners and Users alike, and using it as such adds value to their commercial activities.

3.3 The Soundtrack Repertoire

Here, music is *a mood enhancing ingredient inextricably linked to User's message being conveyed by moving image to viewer / listener*. This repertoire differs significantly from the Musical Repertoire. In the Soundtrack Repertoire, music is '*upbeat and quirky, with a bit of a build*' as opposed to '*uptempo and leftfield, with a crescendo*'. It is *'recessive and background'* rather than '*acoustic with sparse instrumentation*'. This repertoire reflects the way in which the music functions when it is synchronized with the music, and the goal of the film maker in this process. It predominates in user queries [22] but also appears in interviews across the stakeholder spectrum.

3.4 The Cultural Repertoire

Finally, music is represented as being *a subjective appealing distraction which is personal and emotive*. The piece of film has a final audience, which also includes the participants in this process in their recreational lives consuming the media they are involved in creating. As recreational consumers themselves they often bring less 'professional' music talk to these discussions, indicating they are enthusiastic fans of the cultures of music and film:

These purely subjective evaluations of media content appear throughout the texts and are an important way of communicating the meaning and value of a piece of music, film, or the combination of the two. It is marked by a frequent trope: '*when it works, it works*', '*you just know*', or '*it's gut instinct*'. This phrase arises throughout the interviews in response to the question '*what makes a great sync?*'

These repertoires are summarised in Table 1 (below) alongside examples of nouns, phrases and adjectives which help to identify the repertoire in the data:

Repertoire	Keywords
Musical Repertoire: Music is an asset which is created, and has identifiable characteristics.	*Artist, song title, writer, year, album title, chart position, genre, keyword, tempo, lyrics, mood, subject, vocal mix / instrumental*
Business Repertoire: Music is a large collection of recordings which are marketable, contractual and negotiable and have monetary value to the Owner.	*Brand new, cool, big catalogue, comprehensive, demographic, one stop, originating territory, physical*
Soundtrack Repertoire: Music is a mood enhancing ingredient inextricably linked to User's message being conveyed by moving image to viewer / listener.	*Effervescent, uplifting, recessive, theme, build, quirky, unexpected, familiar, theme, background, match the music to the picture*
Cultural Repertoire: Music is a subjective appealing distraction which is personal and emotive	*Like it, opinion, brilliant, great, hate it, it just works, gut feeling, instinct*

Table 1 Talk about music - interpretive repertoires

4. REPERTOIRE ANALYSIS

4.1 Extract 1

An example of coded text can be seen in Appendix 8.1. It can be seen from this extract that the participant is using a range of approaches in her music talk. She is a synchronisation manager in a music publishing company (Owner) and her role is to secure syncs for the music in the catalogue she represents. Her answer to the question:

> "*How do you then match those to the briefs that you are sent and how do you promote them to to your potential clients?"*

incorporates all four repertoires, which in the extract are tagged as <MR> (Musical Repertoire),
 (Business Repertoire, <SR> (Soundtrack Repertoire) and <CR> (Cultural Repertoire). (The colour coding used in NVivo has been translated in this paper into XML-type codes for ease of explanation and reproduction). In the BR firstly she identifies her business resource, the physical "*dedicated music server*", which contains a database of her collection, which is "*quick*" and efficient (*"the most optimum way"*) and refers to the physical acts of making cds and putting mp3s on an ftp site.

She switches to SR, using the film makers' special language of "*briefs*", "*visuals*", "*matching the music to picture*" and "*marry it up*". Although it is not specifically her role to match the music to the moving image it is frequently described by participants as their preferred way of determining relevance. Incorporating this SR act in her discourse indicates an understanding of "*the other side"*,

their way of thinking and working. Indeed she has work experience in the film world and is therefore in a position to adopt repertoires representing different interests.

The CR is clearly identifiable through the use of the subjective opinion-oriented comments of "*I think…*" ("*…are going to work / fit / appropriate*"). This repertoire presents the idea that the 'fit' between music and film is very subjective, and allows the User to make the final decision. Forcing a piece of music on a User ("*this is the one for you*") arises throughout the interviews as a bad approach, whereas a subtle negotiation approach or "*letting the user decide / discover"* is preferred. The CR allows this deference without devaluing the knowledge and expertise of speaker and puts them in a safe position if the final choice is not successful or popular, distancing them from unpopular decisions.

The participant's use of MR in this section discusses the key elements of the musical content of specific "songs", including lyrics ("*words*"), genres ("*rock*", "*pop*") and instrumentation ("*acoustic instrumentals*"). Unsurprisingly these facets appear throughout the texts and are used widely by the participants. Technical musical terms, however, such as melody, harmony, key, or rhythm are rarely mentioned. The MR is more focused on higher level bibliographic metadata than technical musical content. This widespread use of layman's musical language enables easy communication between all parties and stakeholders regardless of their musical expertise. It consists of easily identified facets which are used to organize rights holders' collections rather than more technical film or musical terms used in the SR, or the marketing-based language of the BR.

4.2 Extract 2

Here (Appendix 8.2) a different participant (019SYN) discusses "*What makes a great sync*". He draws from the CR and BR in his answer, switching quickly from one to the other. Although he appears to believe that a "*great sync*" is one that "*works perfectly with that film*" he fully acknowledges that there are other factors which come into play from the BR, including "*cost*", "*politics*", "*the PR and the story*". Again, combining these repertoires justifies and explains self-contradiction and acknowledges the wide variety of factors that impact on the choice of music in this process. Although he initially aligns himself with the CR, presenting the BR as an unpleasant but necessary fact of life, he reinforces his professional standing by acknowledging the importance of market-based factors to successful synchronisation.

5. DISCUSSION

5.1 Meaning-making

These repertoires combine dynamically to determine musical meaning within this community. Music for synchronisation is not purely an abstract art form. It has commercial value, and can be bought and sold, negotiated and cleared; it has physicality, weight and volume; it is an identifiable unique item in a large collection or an amorphous mass of a collection itself; it is defined by the factors around its creation, the artist, the date, or it is defined by its effect on the mood or even purchasing activity of the listener / viewer; it is personal and subjective or it is a perfect match.

Although there is often some emphasis on one or another of the repertoires, each of the participants acknowledges this range of meanings in their music talk. These repertoires can be used to identify their Codes (ways of looking at music) and Competences (ways of looking at the world) [23]. Indeed, Owner Codes mainly draw from MR, User Codes from SR while Owner Competences relate more closely to BR and User Competences to CR (see Fig 1, below).

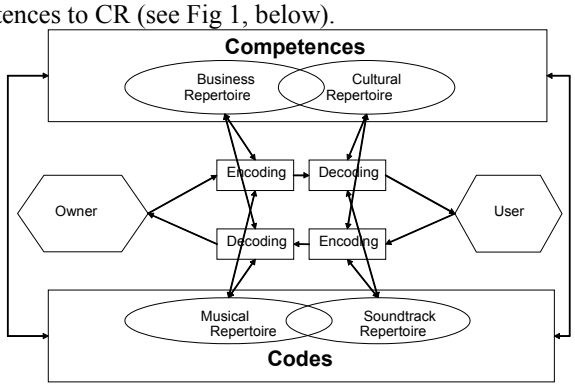


Figure 1 Repertoires as Codes and Competences (adapted from [23])

The model in Figure 1 is adapted from [24], suggesting that the meaning making process in music synchronisation is a dynamic feedback loop between the Owner and the User. The Owners and Users draw from their own and shared Codes and Competences in determining and communicating musical meaning. The results of the DA reported here reinforce the Codes and Competences aspect of the model. The intention is to investigate the Encoding / Decoding process in future analyses.

5.2 Music Information Retrieval

The value of this work to the wider discipline of Music Information Retrieval is twofold. Firstly, the rich and detailed insights into the Repertoires employed within this community of users offered by the analysis indicate a wide variety of ways of thinking about music. In terms of tool and, ultimately, system design, recognizing that music is a multi-variate concept with conflicting features (it is abstract and concrete, it is objective and subjective and it can be used as part of a multi-media construct while standing alone) is key to successfully meeting user information needs. For example, if these ideas were incorporated in the design of a system to find music for sync then the music would not only be described using bibliographic metadata (MR) but would incorporate facets from all of the repertoires. It would allow a user to search databases for a selection of thirty second sections of tracks which are popular with a specific target audience (BR), which have not been used in advertising (SR), have a build (SR), no vocal (or a vocal with a specific lyric which is relevant to the commercial's message) (MR), specific instruments

and feels (MR), price ranges and ease of approval (BR), and is of a style which is preferred by the stakeholders (CR). Much of the BR information can be found in the royalties and business affairs services in Owners systems and attempts are being made by some corporations to incorporate this into their search applications. Automated content-based tools such as 'crescendo detectors' or 'timbre identifiers' would be of use for SR and MR, while autotagging and playlist-building reflect CR. A holistic approach can only benefit industry and the research community.

Secondly, the dynamic element of this process reminds us that meaning is not static but relates both to content and to ever-changing context. This constant flux means that any research is purely a snapshot of ways of thinking and talking about music. As the digital information society develops and music becomes all-pervasive, users and systems become more sophisticated. As the music industry's relationship with music is forced by this development to change then the Codes and Competences made apparent by this analysis are equally likely to develop and change.

6. CONCLUSION

There are appearances through the texts of four repertoires. Music appears to have many forms, which are all considered by all of the participants. Although at first glance it may appear that one group of people (the Owners) thinks one way while another (the Users) think another, this is not the case. Indeed their views are often similar. The ways of thinking about music in this community are more complex. There is certainly some value in analyzing the texts for their surface content - indeed this is an useful way to determine key themes and for the researcher to get an initial understanding of the dynamics of a multi-stakeholder information communications process [1]. However, although it is time-consuming, applying DA to these texts has revealed patterns that were not already clear, given this analysis deeper insight into meaning making within this community and allowed some testing of the theoretical model [24].

7. ACKNOWLEDGEMENTS

Many thanks to all the anonymous participants in this research for being so free with their valuable time and insight. Charlie Inskip gratefully acknowledges financial support from AHRC for this PhD research. The anonymous reviewers comments were also invaluable.

8. APPENDICES

8.1 Extract 1

In this interview extract (001SYN) it can be seen how the participant, who works for a rights holder, uses a range of repertoires to make a decision on the relevant piece of music. Each repertoire example is marked in <>:

Question: How do you then match those to the briefs that you are sent and how do you promote them to to your potential clients?

*Answer:
I have all our music on a dedicated music server</BR><SR> so I will get a brief in and quite often I'll actually get the visual in as well so if I have the visual up on screen</SR>
 I'll bring up my music database </BR><SR>[the visual?]. The visual of the ad, for instance, they'll send me the visual of the ad, so I've got the 60 second or the 30 second ad in front of me which really helps, because it's very different reading a brief and actually seeing how they shoot it. So I'll see it </SR>
then I'll bring up my music database and the</BR><MR> songs</MR> that <CR>I think work</CR>
I'll pick up</BR> and <MR>I'll play the sections of the song</MR> that <CR>I think are going to fit</CR>. <SR>I'll match the music to the picture. I'll marry it up and see if it works or not.</SR>
 That's the most optimum way of doing it </BR><SR>if you get the actual visual in. if I get the script then I'll look at the script, </SR><MR>I'll see if sometimes they'll have a keyword search sometimes they want words say sunshine in it, so I'll look at
all our songs</BR> you know which songs have the word sunshine in </MR>and then <SR>match see </SR>
if pitch those </BR><CR>see if those work</CR>. <MR>Or there'll be a genre, what kind of style, you know they'll say 'no rock, no pop, we just want purely acoustic instrumentals' anything like that, so I'll go through the all the instrumentals that I have in that genre and listen to those</MR>
 and pitch </BR><CR>what I think's appropriate.</CR>
Nowadays I have to say, I used to make up cds and send them out but because of the fast turnaround I email mp3s, or I put them onto an ftp site and I say 'here* [indistinct] *here's* [indistinct] *package you know download these,</BR><CR> these are the songs that I think are going to work for you</CR>.
 And that's how I get them out there. Because it's much quicker now to do that, much.</BR>*

8.2 Extract 2

This example, features a freelance creative music searcher employed by ad agencies:

Question: ok. Last one. What makes a great sync?

*Answer: Good question, what makes a great sync?
I think the most important thing for me is not to compromise.
<CR> It has to be the best piece of music for that film.</CR>
.And away from all the other factors around it, ie cost, politics, all those things that come into it,<CR> it has to have that feeling</CR> that no matter where this piece of music has come from, no matter how much it costs, no matter who owns it, and who's getting the money,</BR> <CR>it is the right piece for this film. That's the essence, I feel.</CR>
Beyond that, I think, other things on top of the sync, beyond the sync, can make it a great thing, I mean the PR and the story. If it's a band that have been launched off the back of an amazing spot I think that can*

also be really exciting, but that's just an added extra. </BR><CR>I think it's just how that piece of music works perfectly with that film. .. yes.</CR>

9. REFERENCES

[1] C. Inskip, A. MacFarlane & P. Rafferty: "Music, Movies and Meaning", *Proceedings of 9th International Society for Music Information Retrieval Conference*, Vienna, Austria, 2008.

[2] C. Inskip, A. MacFarlane & P. Rafferty: "Organising Music for Movies", *Proceedings of International Society for Knowledge Organization (UK) Content Architecture conference*, London, UK, 22-23 Jun 2009

[3] J. Lee, M. Cameron Jones, J.S.Downie: "An analysis of ISMIR Proceedings: Patterns of Authorship, Topic and Citation", *Proceedings of 10th International Society for Music Information Retrieval Conference*, Kobe, Japan, 2009.

[4] M.Grachten, M.Schedl, T.Pohle, G.Widmer: "The ISMIR Cloud: A Decade of ISMIR conferences at your fingertips", *Proceedings of 10th International Society for Music Information Retrieval Conference*, Kobe, Japan, 2009.

[5] D.Bainbridge, S.J.Cunningham, J.S.Downie: "How People Describe Their Music Information Needs: A Grounded Theory Analysis Of Music Queries", *Proceedings of 4th International Society for Music Information Retrieval Conference,* Baltimore, 2003

[6] S.J.Cunningham & D.Nichols: "Exploring social music behaviour: an investigation into music search at parties", *Proceedings of 10th International Society for Music Information Retrieval Conference*, Kobe, Japan, 2009.

[7] J.Lee, J.S.Downie, M.Cameron Jones: "Preliminary Analyses of Information Features Provided by Users for Identifying Music", *Proceedings of 8th International Society for Music Information Retrieval Conference*, Vienna, Austria, 2007.

[8] J.S.Downie, D.Byrd, T.Crawford: "Ten Years of ISMIR: Reflections on Challenges and Opportunities", *Proceedings of 10th International Society for Music Information Retrieval Conference*, Kobe, Japan, 2009.

[9] J. Potter & M. Wetherell: *Discourse and Social Psychology,* Sage Publications, London, 1987

[10] B.Paltridge: *Discourse Analysis*, Continuum, London, 2006

[11] C.Antaki, M.Billig, D.Edwards & J.Potter: "Discourse analysis means doing analysis", *Discourse Analysis Online,* 2002 [available at <http://extra.shu.ac.uk/daol/articles/v1/n1/a1/antaki20020 02-paper.html> last accessed March 2, 2010]

[12] P.J.McKenzie: "Interpretive Repertoires" chapter 36 in eds K. Fisher, S. Erdelez & L.McKenzie: *Theories of Information Behaviour*", ASIST, Information Today, Medford, 2005

[13] S. Talja: *Music, Culture, and the Library*, Scarecrow Press, Maryland, 2001

[14] J. Carlisle: "Digital Music and Generation Y: discourse analysis of the online music information behaviour talk of five young Australians", *Information Research*, Vol 12, No 4, 2007

[15] B. Frohmann: "Discourse Analysis as a Research Method in Library and Information Science", *Library and Information Science Research,* Vol 16, pt 2, pp 119-138, 1994

[16] J.M.Budd & D.Raber: "Discourse Analysis: Method and Application in the study of information*", Information Processing and Management*, Vol 32, No 2, pp 217-226, 1996

[17] D.Stowell, A.Robertson, N.Bryan-Kinns & M.D.Plumbley: "Evaluation of live human-computer music-making: quantitative and qualitative approaches", *International Journal of Human Computer Studies*, Vol 67, No 11, pp 960-975, 2009

[18] C. Inskip, A. MacFarlane & P. Rafferty: "Creative Professional Users' Musical Relevance Criteria*", Journal of Information Science* In Press, 2010

[19] M.Patton: *Qualitative evaluation and research methods.* Sage Publications, Newbury Park, California, 1990

[20] NVivo software: http://www.qsrinternational.com/ QSR International, 2010

[21] C. Inskip, A. MacFarlane & P. Rafferty (2009) "Towards the Disintermediation of Creative Music Search", *Proceedings of ECDL Workshop on Exploring Musical Information Spac*es, Corfu, Greece, 1-2 Oct 2009

[22] P.Tagg: *Introductory notes to the Semiotics of Music, version 3,* 1999 [internet] (Accessed [07 Dec 2006]), http://www.tagg.org/xpdfs/semiotug.pdf

[23] C. Inskip, A. MacFarlane & P. Rafferty (2008) "Meaning, communication, music: towards a revised communication model", *Journal of Documentation* Vol 64, No 5, pp 687-706.

Papers of *f*(MIR)
the 2nd workshop on
the future of MIR

http://www.columbia.edu/tb2332/fmir2010

A ROADMAP TOWARDS VERSATILE MIR

Emmanuel Vincent
INRIA

Stanisław A. Raczyński, Nobutaka Ono, Shigeki Sagayama
The University of Tokyo

ABSTRACT

Most MIR systems are specifically designed for one application and one cultural context and suffer from the semantic gap between the data and the application. Advances in the theory of Bayesian language and information processing enable the vision of a *versatile*, *meaningful* and *accurate* MIR system integrating all levels of information. We propose a roadmap to collectively achieve this vision.

1. INTRODUCTION

MIR has the vocation of covering all music and all music-related applications, *e.g.* transcription, structuration, alignment, tagging, personalization, composition and interaction. Yet, most systems to date are designed for one class of applications and one cultural context, namely Western popular music, which limits their reusability and their meaningfulness in the sense of [12]. In addition, most systems rely on general pattern recognition techniques applied onto a bag of low-level features, which bounds their accuracy to some glass ceiling. A system integrating all levels of information would make it possible to address virtually any application on any data in a versatile, meaningful and accurate fashion. For instance, it would enable much higher-level interaction, *e.g.* changing the music genre of a recording without affecting some of its other features. While many share the vision of this complete system [1], no fully satisfying approach has yet been proposed to achieve it.

One integration approach adopted *e.g.* by the NEMA [1] project or by [5] is to queue several audio and symbolic feature extraction modules, so as to compute higher-level "features of features". This bottom-up approach greatly improves accuracy but will eventually reach a glass ceiling too due to the propagation of errors from one processing stage to the next. Also, it is not fully versatile since each application requires the implementation of a specific workflow and the inputs and outputs of a module cannot be swapped otherwise than by developing a new module. Top-down approaches based on *probabilistic graphical models* address these issues by estimating the hidden features best accounting for the observed features [10]. All applications then amount to inferring and possibly manipulating some of the hidden features, without changing the model nor the general decoding algorithm. For instance, small graphical models such as Hidden Markov Models (HMMs) integrating harmony and pitch are routinely used to infer the most probable chord sequence given a set of MIDI notes [8] or conversely to generate the most probable melody given a chord sequence [2] using the general Viterbi algorithm.

It is common belief that the formalism of graphical models has the potential to yield a versatile and accurate MIR system by integrating more and more features into a *hierarchical model*. Yet, this formalism alone does not suffice to achieve this vision, as challenging issues pertaining to the definition of the model structure, the parameterization of conditional distributions, the design of unsupervised learning algorithms and the collection of development data have often been overlooked. In this paper, we explicitly state and clarify these issues and derive a feasible roadmap.

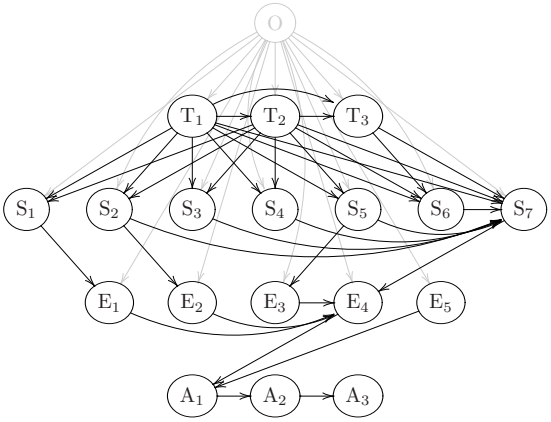
2. COMPLETE MODEL STRUCTURE

The first task is to collectively define a taxonomy of music features and their statistical dependencies that virtually covers all existing and future music. This involves the following steps: building an exhaustive list of music features and their definition, identifying features amenable to the same theoretical treatment, *e.g.* genre and mood, and organizing these features into a dependency network. This is not straightforward, since the current MPEG-7 standard [2] mostly addresses low-level or application-specific features and agreed-upon definitions of features such as musical structure or rhythm remain to found. Also, the dependency network is not unique and a sparse network is preferred.

We propose a draft model of a music piece as a *dynamic Bayesian network* in Figure 1. While it may be incomplete, we believe that it provides a useful basis for community discussion. In this graph, each node represents a sequence of uni- or multidimensional features considered as a vector random variable. Statistical dependencies are indicated by arrows such that the conditional distribution of a variable given its ancestors depends on its parents only. "Vertical" hierarchical dependencies are explicitly displayed, while "horizontal" temporal dependencies within and between nodes are implicitly accounted for. We adopt a *generative modeling* point of view [10] where lower-level features depend on higher-level features. The *joint distribution* of all variables then factors as the product of the distribution of each variable given its parents [10].

[1] http://nema.lis.uiuc.edu/

[2] http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm



Overall features

O *Tags*: set of tags in $\mathcal{V}_{\text{tags}}$ covering all or part of the piece, *e.g.* genre, mood, composer, performer, place and user preference

Temporal organization features

T_1 *Structure*: set of possibly overlapping/nested sections, each defined by its quantized duration in bars and by a section symbol in $\mathcal{V}_{\text{sect}}$

T_2 *Meter*: sequence of bars, each defined by its reference beat and time signature in $\mathcal{V}_{\text{meter}}$ and by the associated metrical accentuation level of each beat and beat subdivision

T_3 *Rhythm*: sequence of *events* associated to one or more simultaneous note onsets, each defined by its quantized duration in beats and by the associated number of onsets [11]

Symbolic features

S_1 *Notated tempo*: beat-synchronous sequence of tempo and tempo variation symbols in $\mathcal{V}_{\text{tempo}}$

S_2 *Notated loudness*: beat-synchronous sequence of loudness and loudness variation symbols in $\mathcal{V}_{\text{loud}}$

S_3 *Key/mode*: beat-synchronous sequence of key/mode symbols in $\mathcal{V}_{\text{key}}$

S_4 *Harmony*: beat-synchronous sequence of chord symbols in $\mathcal{V}_{\text{chord}}$

S_5 *Instrumentation*: beat-synchronous sequence of sets of active voices, each defined by a voice symbol in $\mathcal{V}_{\text{inst}}$ (including instruments, orchestra sections, singer identities, and sample IDs, with various playing or singing styles)

S_6 *Lyrics*: event-synchronous sequence(s) of syllables in $\mathcal{V}_{\text{syll}}$

S_7 *Quantized notes*: set of notes (including pitched/drum notes, voices or samples), each defined by quantized onset and duration in beats, its articulation symbol in $\mathcal{V}_{\text{artic}}$, its loudness and loudness variation symbol in $\mathcal{V}_{\text{loud}}$, its quantized pitch and pitch variation symbol in $\mathcal{V}_{\text{pitch}}$, its voice symbol in $\mathcal{V}_{\text{inst}}$ and its syllable in $\mathcal{V}_{\text{syll}}$

Expressive performance features

E_1 *Expressive tempo*: beat-synchronous sequence of actual tempo values in bpm

E_2 *Expressive loudness*: beat-synchronous sequence of actual global loudness values in sones

E_3 *Instrumental timbre*: beat-synchronous sequence of vectors of parameters modeling the timbre space of each voice

E_4 *Expressive notes*: set of notes, each defined by its actual onset time and duration in s, its loudness curve in sones, its pitch curve in Hz, and its trajectory in the timbre space

E_5 *Rendering*: time-synchronous sequence of vectors of parameters characterizing the recording setup (*e.g.* reverberation time, mic spacing) or the software mixing effects and the spatial position and spatial width of each voice

Acoustic features

A_1 *Tracks*: rendered acoustic signal of each voice

A_2 *Mix*: overall acoustic signal

A_3 *Classical low-level features*: MFCCs, chroma, etc

Figure 1. Draft model of a music piece. Dependencies upon overall features are shown in light gray for legibility.

A wide application range is ensured by allowing the feature *value sets* $\mathcal{V}_{\text{sect}}$, $\mathcal{V}_{\text{meter}}$, $\mathcal{V}_{\text{tempo}}$, $\mathcal{V}_{\text{loud}}$, $\mathcal{V}_{\text{chord}}$, $\mathcal{V}_{\text{inst}}$, $\mathcal{V}_{\text{artic}}$, $\mathcal{V}_{\text{pitch}}$ to be either fixed or adaptive and to contain a $\emptyset$ symbol denoting the lack of structure, meter and so on. The variables T_1, T_3, S_4 and S_7 also implicitly depend on a set of structural, rhythmic, harmonic, melodic and bass *patterns* denoted $\mathcal{P}_{\text{sect}}$, $\mathcal{P}_{\text{rhythm}}$, $\mathcal{P}_{\text{chord}}$, $\mathcal{P}_{\text{melo}}$ and $\mathcal{P}_{\text{bass}}$. More generally, all model parameters can themselves be regarded as variables [10].

Any variable may be either fully observed, partially observed or hidden, leading to a huge range of scenarios. For instance, automatic accompaniment consists of inferring A_2 given part of A_1, while symbolic genre classification consists of inferring part of O given S_7. Playlist generation or cover detection may also be addressed by comparing all features O, T_*, S_* and E_* inferred from A_2 in each piece of a database according to some criterion.

3. SCALABLE CONDITIONAL DISTRIBUTIONS

Once the variables have been defined, the next step consists of designing conditional distributions between these variables. While recent studies in the field of audio source separation have already led to complete family of acoustic models $P(\mathrm{A}_1|\mathrm{E}_4, \mathrm{E}_5)$ [9], many other dependencies have either been studied in a deterministic setting or not investigated yet. More crucially, current probabilistic symbolic models, *e.g.* [4, 8, 11], only account for short-term dependencies between two or three variables taking few possible values and rely on hand tying of probabilities based on Western musicology. This design method does not scale with the long-term high-dimensional dependencies found in Figure 1. For instance, the virtually infinite set of overall features O affects most other features and the probability of quantized notes $P(\mathrm{S}_7|\mathrm{O}, \mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3, \mathrm{S}_2, \mathrm{S}_3, \mathrm{S}_4, \mathrm{S}_5, \mathrm{S}_6)$ depends on as many as 9 other features. *Scalable* methods must hence be found to parameterize each conditional distribution so as to avoid overfitting.

A promising approach consists of modeling the conditional distribution of a variable given its parents by *interpolation* of its conditional distributions given each parent individually. This approach is widely used in the language processing community [3] but does not account for possible interactions between parents. This issue may be tackled by reparameterizing the space of parent variables in terms of a smaller number of factors using *e.g. Latent Semantic Indexing* (LSI) techniques [7] developed for text retrieval and collaborative filtering. We believe that the extension of these approaches to all symbolic music data will lead to a similar breakthrough as in the above domains.

4. UNSUPERVISED LEARNING ALGORITHMS

The design of conditional distributions is closely related to that of learning and decoding algorithms. Indeed, due to the above dimensionality issues and to the variety of music and individual music experiences, most distributions cannot be fixed a priori but must be learned from possibly user-specific training data or from the test data. Similarly, the

feature value sets $\mathcal{V}_*$ and the patterns $\mathcal{P}_*$ must be learned to identify *e.g.* the most relevant set of chord symbols and patterns for a given song, in line with human listening that picks up regularities based on prior exposure without actually naming them [12]. These learning tasks are always *unsupervised* since training data annotated with all features of Figure 1 will most probably never exist.

The estimation of some hidden variables consists of marginalizing *i.e.* integrating the likelihood over the values of the other hidden variables [10]. This can be achieved using the modular sum-product and max-product *junction tree algorithms* [10] that generalize the classical Baum-Welch and Viterbi algorithms for HMMs. The considered objective is often the maximization of the posterior distribution of the inferred variables given the data. Although this Maximum A Posteriori (MAP) objective may be used for unsupervised learning of the model parameters (*i.e.* conditional probabilities) [8], it cannot infer the model order (*i.e.* the dimension, the value set and the parents of each feature). Unsupervised pruning of feature dependencies would also considerably accelerate the speed of the junction tree algorithm, that is exponential in the number of dependencies, and make it possible to match the available computational power in an optimal fashion. Suitable *model selection* criteria and algorithms are hence of utmost importance.

Automatic Relevance Determination (ARD) and several other popular model selection techniques employ prior distributions over the model parameters favoring small model orders [7]. The alternative *variational Bayesian inference* technique [10] selects the model with highest marginal probability by integrating the posterior distribution of all hidden variables. This technique also provides an approximation of the posterior distribution of all hidden variables as a by-product. This increases the meaningfulness and interpretability of the results compared to the estimation of the MAP variable values only, at the cost of higher computational complexity. Again, we believe that advances will eventually be achieved by combining these approaches.

The choice of algorithms will guide that of feature formats. Efficient *graph formats* exist for the representation of posterior distributions of symbolic feature sequences [6], but they must yet be extended to *e.g.* polyphonic note sequences. More generally, the high dimensionality of all features will necessitate *compressed feature formats*.

5. MULTI-FEATURE ANNOTATED DATABASE

Finally, although the development of a database annotated with all features of Figure 1 appears infeasible, some multi-feature annotated data will nevertheless be needed to initialize and evaluate the unsupervised learning process. This implies strong community coordination to push current annotation efforts towards the same data and bridge the cultural gap between experts of different music styles. Also, this advocates for the evaluation of *conditional feature estimation* tasks within MIREX, where some other features would be known, as opposed to the current full-fledged estimation tasks, where only audio or MIDI are given.

6. SUMMARY AND IMPLICATIONS

We provided a feasible roadmap towards a complete MIR system, emphasizing challenges such as scalable parameterization and unsupervised model selection. As recommended in [1], this implies that most efforts in the MIR community now focus on symbolic data. Yet, other strong implications also arise regarding the need for a coordinated effort and the definition of MIREX tasks and data.

7. ACKNOWLEDGMENT

This work is supported by INRIA under the Associate Team Program VERSAMUS (http://versamus.inria.fr/).

8. REFERENCES

[1] J. S. Downie, D. Byrd, and T. Crawford. Ten years of ISMIR: Reflections on challenges and opportunities. In *Proc. ISMIR*, pages 13–18, 2009.

[2] S. Fukayama, K. Nakatsuma, et al. Orpheus: Automatic composition system considering prosody of japanese lyrics. In *Proc. ICEC*, pages 309–310, 2009.

[3] D. Klakow. Log-linear interpolation of language models. In *Proc. ICSLP*, pages 1695–1699, 1998.

[4] K. Lee. A system for automatic chord transcription using genre-specific hidden Markov models. In *Proc. AMR*, pages 134–146, 2007.

[5] A. Mesaros, T. Virtanen, and A. Klapuri. Singer identification in polyphonic music using vocal separation and pattern recognition methods. In *Proc. ISMIR*, pages 375–378, 2007.

[6] S. Ortmanns, H. Ney, and X. Aubert. A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech and Language*, 11(1):43–72, 1997.

[7] M. K. Petersen, M. Mørup, and L. K. Hansen. Sparse but emotional decomposition of lyrics. In *Proc. LSAS*, pages 31–43, 2009.

[8] C. Raphael and J. Stoddard. Harmonic analysis with probabilistic graphical models. *Computer Music Journal*, 28(3):45–52, 2004.

[9] E. Vincent, M. G. Jafari, et al. Probabilistic modeling paradigms for audio source separation. In *Machine Audition : Priciples, Algorithms and Systems*. IGI, 2010.

[10] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[11] N. Whiteley, A. T. Cemgil, and S. J. Godsill. Bayesian modelling of temporal structure in musical audio. In *Proc. ISMIR*, pages 29–34, 2006.

[12] F. Wiering. Meaningful music retrieval. In *Proc. f(MIR)*, 2009.

PREDICTING DEVELOPMENT OF RESEARCH IN MUSIC BASED ON PARALLELS WITH NATURAL LANGUAGE PROCESSING

Jacek Wołkowicz
Dalhousie University
Faculty of Computer Science
jacek@cs.dal.ca

Vlado Kešelj
Dalhousie University
Faculty of Computer Science
vlado@cs.dal.ca

ABSTRACT

The hypothesis of the paper is that the domain of Natural Languages Processing (NLP) resembles current research in music so one could benefit from this by employing NLP techniques to music. In this paper the similarity between both domains is described. The levels of NLP are listed with pointers to respective tasks within the research of computational music. A brief introduction to history of NLP enables locating music research in this history. Possible directions of research in music, assuming its affinity to NLP, are introduced. Current research in generational and statistical music modeling is compared to similar NLP theories. The paper is concluded with guidelines for music research and information retrieval.

1. INTRODUCTION

Along with the information revolution triggered by the introduction of computers, new opportunities have emerged for music artists and researchers. When some began executing computational problems on large mainframe computers others used them to generate early computer music. At this point of time one started to think how computers might be used to process and analyze music matter. Music, similarly to human speech, accompanied human evolution from its beginning, so deep understanding of music can allow better understanding of better human cognition. However, music data is in most cases still treated as unstructured binary data left on the same shelf with images, movies, computer programs; opposite to textual data, which are easy to process, search, index, driven by a large number of available computer aided techniques provided by natural language processing, information retrieval or text data mining like classification, analysis, generation, summarization, indexing, searching, translation and much more. However, music can be treated as a natural language and could be processed in the similar way as text. Although there are substantial differences between written text and music, they have many features in common.

NLP level	Music research areas
phonetics	Waveform analysis, audio signals
phonology	Sound events identification
morphology	Score symbols, symbolization
syntax	N-grams, shallow reduction and parsing
semantics	Harmonics, phrase level, parsing
pragmatics	Phrases, voice leading
discourse	Interpretations, context of a piece

Table 1. NLP Levels with respective music research tasks.

2. MUSIC AS A NATURAL LANGUAGE

By definition, a natural language is any language which arises in an unpremeditated fashion as the result of the innate facility for language possessed by the human intellect. Not everybody agrees that music fits this definition, but music researchers, who know the rules of music, are usually more prone to agree with it. Music, as well as text, has the symbolic representation that has its origins dated back in ancient times. Music and language are the only old human creative activities where symbolic representation is commonly used. Others, like painting, sculpture, dance did not have such common symbolic notation. Music notation cannot be directly ported into computers like text is, but this is only a representation issue that could be easily overcome. For instance, the argument that text can easily be split into words - the basic features for Natural Language Processing (NLP) and Information Retrieval (IR), which is not the case for music, can be countered if one mentions that there are natural languages that do not use anything to separate words, like Thai.

3. MUSIC RESEARCH PARALLEL TO NLP

In order to treat music as a natural language, one has to show that music processing works on the same classes of problems as NLP does. One distinguishes certain levels of a text processing, listed in the Table 1. NLP tries to convey the research through all those levels, from recording (a voice, speech) to understanding (the meaning of a speech).

These levels also exist for music. Similarly to a natural language, music can be recorded and presented primarily as a waveform. On the 'phonetics' level one tries to investigate the structure of a sound, separate and distinguish

between notes or instruments. However, music is much more complex in this area and sound recognition tasks are still facing basic problems.

The second very important similarity results from the fact both domains use symbolic notations. Music score also consists of characters which are called notes. Similarly to NLP's morphology and syntax — music has hidden, grammar-like structure, hidden rules. Part of it is the harmony. It determines how to put words (notes) together, how to build well-formed phrases using them. It also manages the musical meaning of a piece of which the basic exemplification is a progression of chords and notes. In the case of notes and their dependencies — we may talk about the syntax of the music while in the case of chords or harmonic progressions — about the semantics of the certain phrase or given the phrasing — the pragmatics of the excerpt. This is very similar in its form to one of the main areas of NLP, which is grammatical analysis. The highest level of NLP (discourse) is also common in music in a form of ideas, desires or aspirations (romantic music) of a composer as well as pictures and actions behind it (program music). Dukas's "The Sorcerer's Apprentice" or Smetana's "die Moldau" are the very good example of such music.

4. HISTORY OF NLP AND MUSIC RESEARCH

The history of NLP reaches beginnings of the history of computation since it is believed that the ability of computers to process natural language as easily as we do will signal creation of real intelligent machines. It comes from the assumption that the use of language is an inherent part of human cognitive abilities. Based on that a test, known as the Turing test, has been proposed to determine if a machine is intelligent. The main objective of this test is that a truly intelligent machine could carry on a discussion with a human so that the latter could not recognize if he is talking to a human or to a machine. This definition of intelligence triggered the research in NLP with the ultimate goal to create such a conversational program. If it is a feasible goal one has to actually and physically implement the way people think, reason and formulate thoughts.

At this point we can see a resemblance between human speech and music. Assuming music is a natural language, the Turing test would consist in generating musical pieces so that an expert could not recognize if an author is a machine or not. Why not a layman? Because it would be equivalent to a Turing test where an interlocutor does not know the language of the talk. We would call it a *Soft Turing Test* which, unlike for regular human natural languages, makes more sense for music.

4.1 Introduction of Grammars

There has been some pioneering work in both areas, NLP and Music Research. Some natural language analysis in a form of linguistics theories were made before the introduction of computing machines. As an early, pre-computer music research one can point work of Heinrich Schenker with his Reduction theory in the beginnings of 20^{th} century.

These things changed rapidly for NLP just after computers were invented. Interface to computers is textual and this is a natural format of computer files. Moreover, there was a very strong need for developing automatic machine translation. The beginning of this - the Noam Chomsky's theory of context free grammars for natural texts - dates back to 1956. Representing and processing music was not the top priority of that times. As a similar work in the field of music one can point the book "A Generative Theory of Tonal Music" by Lerdahl and Jackendoff published in 1983. Both of this approaches deal with the respective areas in the same way - by introducing a formal grammar that may generate instances in the given areas.

4.2 The period of 'Look ma, no hands'

The early NLP researchers were very optimistic. The research was driven by the goal of developing automatic machine translation. Various systems were created but, although they worked perfectly on several, very limited examples, they were failing in the real-world applications. The research came to the point where nothing more could be achieved, and yet they haven't created any robust system that will work on real data. This created a crisis in the whole field. It looked that despite their complicated system, it is not possible to mimic human cognition in the area of natural languages.

It is likely the time where the music research has just come. It is not that crucial as for natural languages, since one can still try to trick unexperienced listeners and thus pass the *Soft Turing Test* with the system that does not demonstrate the full understanding of the matter it deals with. Another sign that the field of music research might be in this kind of situation is the introduction of this kind of tracks, where one asks about the future of the field(e.g. fMIR).

4.3 Present NLP

The old approach - to create a model that will solve all our problems in the area of cognition of human speech — seemed to be wrong. Current research of NLP lies on one hand on creating more precise generative models that includes probability or other aspects of other features of languages (for example using attribute value matrices — AVMs). On the other hand, stochastic NLP has gone towards classical Data Mining (DM) where with the use of shallow parsing with mining on textual data gives much better results than simple DM. NLP techniques are also being injected to Information Retrieval increasing performance of this systems. Finally, the great improvement has been done in machine translation, the first goal of NLP researchers, where Google Translate or Wave's Rosy are the examples. This is the direction which current music research may follow.

5. RELATED WORK

Current research in music concentrates around Music Information Retrieval, both for the signal and symbolic music representations. In most cases it deals on basic issues how computers should deal with music data in general. The level of music interpretation does not go into semantics, probably because it is vague what the meaning in music is. However, one should notice that current text Information Retrieval benefits from the semantic layer of text (text classification, ontologies and relations between terms, dependencies between documents, linguistic layer of text).

We would like to emphasize the work of Lerdahl and Jackendoff [5], who first describe a generative approach that one can use toward the music. They describe it in a computational linguistics manner, using preference rules approach, mentioning that it could be possible to implement their rules in a working system. For the implementation of their system we had to wait for a long time, since they did several elisions of some tough to define, important basic notions, understandable by humans, but hard to implement on a machine. A recent try, ATTA [3], deals with all the implementation issues by introducing several important limitations to the system, which does not go beyond syntactic level, leaving behind harmony issues.

Another preference rules generative approach, that introduces very important component of modern NLP - probability, is described by Temperley [7]. Probabilities and corpus based statistics is an inherent part of all modern NLP theories hence probabilities can model the meaning of text by inferring dependencies within it [6]. This work reminds of the idea of probabilistic grammars introduced earlier for text and proposed for music by Bod [1].

Statistical analysis is a very important component of NLP models and it has played (Cope [2]) and will play a major role in music research. In many cases, solutions to many problems that gave good results for texts, could give comparable results in music area. As an example, the n-gram method of authorship attribution developed for natural language texts [4] gave good results for composer recognition of musical pieces [8].

6. FUTURE OF MUSIC RESEARCH

If the hypothesis that NLP and current computational music research are in a sense the same but operate in two similar but not identical fields, both fields could benefit from this legacy. For instance, applications that span large number of levels of NLP (e.g. try to draw some high level conclusions based on low level music representations) would work better, if they focus on a few levels only. As we have pointed out the layers of NLP, some of them are not that well covered for the music matter. Lots have been done in the areas of music 'phonetics', 'phonology' and 'morphology'. We notice some recent work in the area of 'semantics' but there is no models in higher, much more interesting but complicated levels: 'semantics' 'pragmatics' and 'discourse'. Those areas define the meaning of the data we deal with, the understanding of undergoing structure and the flow of composers ideas within a piece. In general, one can stack different applications given the structure of NLP i.e. the output of a model that operates on syntactic level could be an input of a model operating on semantic level.

A few tasks that are relevant for music research and are well developed within NLP are sentiment analysis, genre classification, automatic summarization or idiom extraction. Other approach would be to enhance MIR with some semantic aspects of music matter - music ontologies with an application of shallow parsing (or alternatively, local reductions) to reach the level of current state-of-the-art of textual Information Retrieval. However, it is still not clear how to represent meaning of music in computational tasks but in this case statistical approach and data mining techniques may be relevant tool to describe this phenomenon.

7. CONCLUSIONS

The domain of music research resembles research in NLP. Both fields operate on the similar types of data that share common features. Both domains deal with data that are easily perceivable by humans but pose a lot of problems to make them fully understandable by computers. Research in both areas uses similar techniques and should be able to take from each other in the areas that are more developed in one of them. Music researchers could share their insight in tasks like voices separation or boundaries detection while benefit from NLP's statistical methods, automatic approaches to semantics or aiding information retrieval and data mining with natural language understanding. It is not necessary that all those inherited techniques and approaches will work but definitely, it is worth trying.

8. REFERENCES

[1] Rens Bod. A unified model of structural organization in language and music. *JAIR*, 17(1):289–308, 2002.

[2] David Cope. Computer modeling of musical intelligence in emi. *Comp. Music Journ.*, 16(2):69–83, 1992.

[3] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Implementing a generative theory of tonal music. *Journal of New Music Research*, 35:249–277, 2006.

[4] Vlado Keselj, Fuchun Peng, Nick Cercone, and Calvin Thomas. N-gram-based author profiles for authorship attribution. In *Proc. of the PACLING03 Conf.*, 2003.

[5] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, 1983.

[6] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, June 1999.

[7] David Temperley. *Music and Probability*. The MIT Press, 2007.

[8] Jacek Wołkowicz, Zbigniew Kulka, and Vlado Keselj. N-gram-based approach to composer recognition. *Archives of Acoustics*, 33(2008)(1):43–55, Jan 2008.

Author Index

ISBN 978-90-393-53813