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

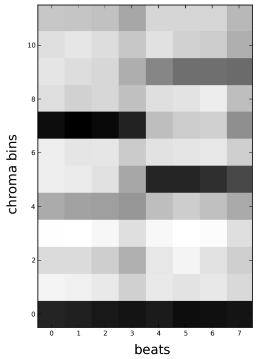


Figure 1: A typical codeword from a codebook of size 200 (code 7 in Figure 4), corresponding to a tonic-subdominant chord progression. The patch is composed of 2 bars and the pattern length was set to 8 beats.

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 < ℓ ≤ 1
  for nIters do
    for p ∈ {P_n} do
      c ← min_{c∈C_k} dist(p, c)
      c ← c + (p − c) ∗ ℓ
    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/

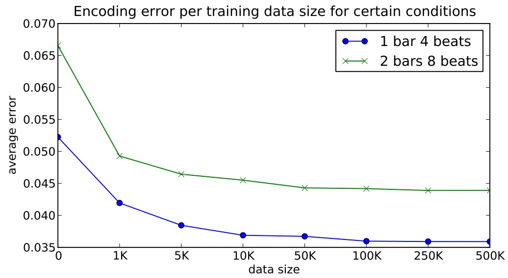


Figure 2: Distortion for a codebook of size 100 encoding one bar at a time with by 4 columns. Therefore, each codeword has $12 \times 4 = 48$ elements. Distortion is measured on the test set. Training data sizes range from 0 (just initialization) to $500,000$. Patterns were selected at random from the 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:

$$\text{dist}(p_1, p_2) = \sum_{i,j} \frac{(p_1(i,j) - p_2(i,j))^2}{\text{size}(p_1)} \tag{1}$$

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

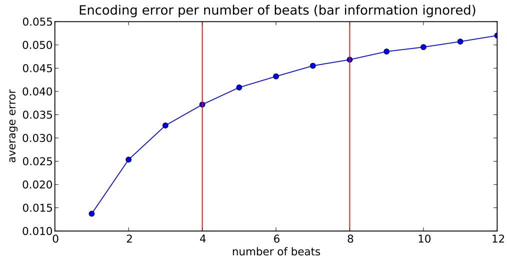


Figure 3: Encoding patterns of different sizes with a fixed size codebook of 100 patterns. The size of the pattern is defined by the number of beats. Downbeat (bar alignment) information was not used for this experiment.

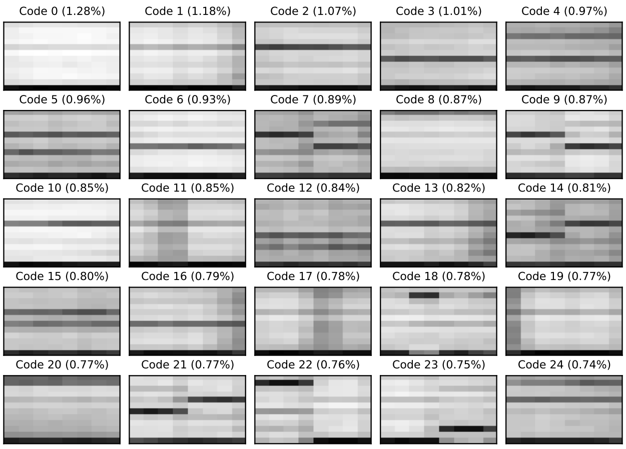


Figure 4: The 25 codes that are most commonly used for the *artist20* test set. Codes are from the 200-entry codebook trained on 2 bar, 12×8 patches. The proportion of patches accounted for by each pattern is shown in parentheses.

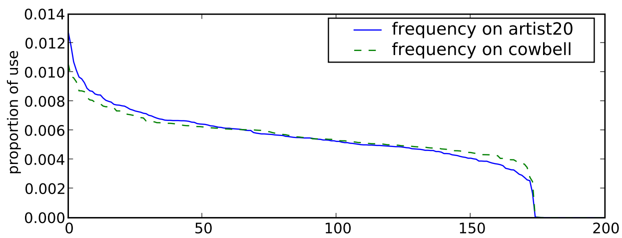


Figure 5: Usage proportions for all 200 codewords on the *artist20* test set (which comprises 71,832 patterns). Also shown are the usage proportions for the training set ("cowbell"), which are similar. Note that even though all codewords are initialized from samples, some are used only once in the training 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/`

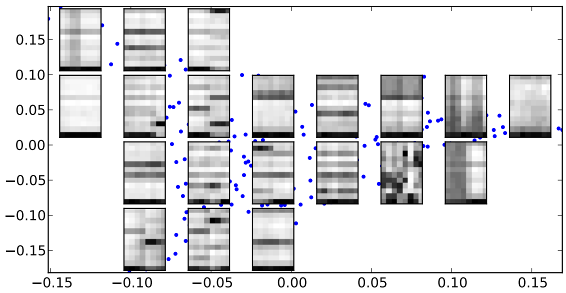


Figure 6: LLE visualization of the codebook. Shown patterns are randomly selected from each neighborhood.

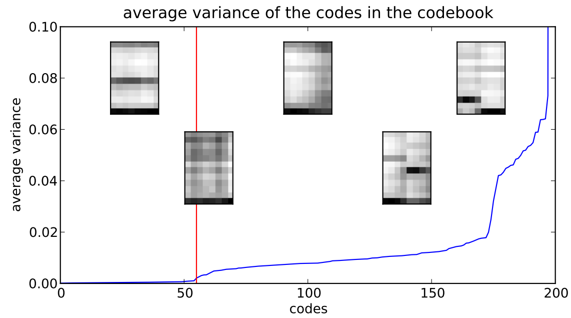


Figure 7: Average variance of codewords along the time dimension. The vertical axis cuts at the 53rd pattern, roughly the number of codewords consisting entirely of sustained chords. Representative patterns are shown in each range.

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-

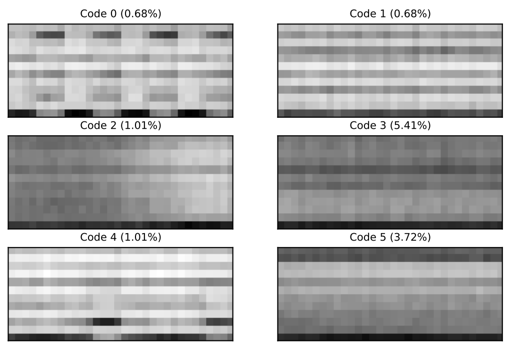


Figure 8: Sample of longer codewords spanning 8 bars. Codewords were randomly selected from a 100-entry codebook. Percentages of use are shown in parentheses. Most patterns consist of sustained notes or chords, but code 0 shows one-bar alternations between two chords, and code 4 contains two cycles of a $1 \rightarrow 1 \rightarrow 1 \rightarrow 2$ progression.

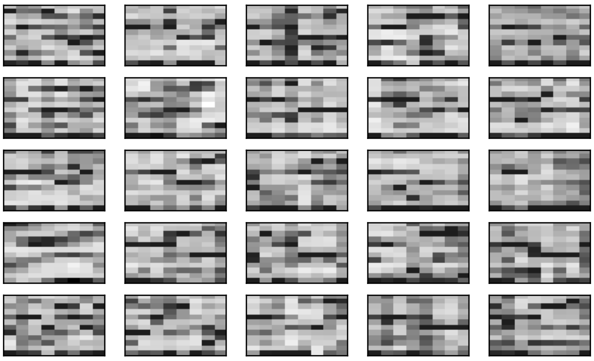


Figure 9: Cluster around centroid presented in Figure 1. Taken from the *artist20* dataset, the cluster size is actually 639. Shown samples were randomly selected. This gives an intuition of the variance in a given cluster.

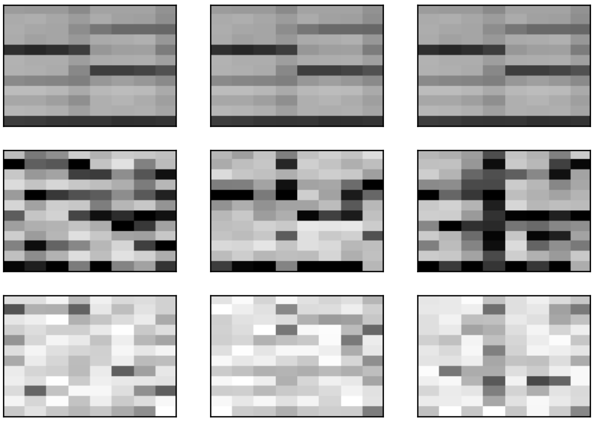


Figure 10: First three patterns of figure 9 (2nd line) presented with the centroid from Figure 1 (1st line) and the absolute difference between both (3rd line).

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 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.

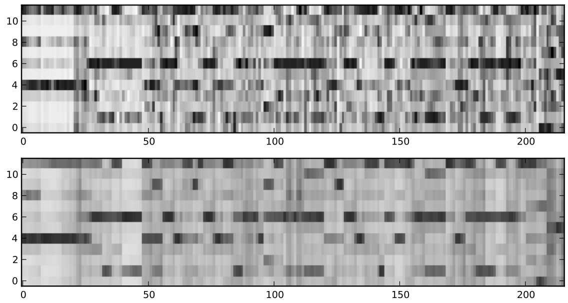


Figure 11: Good Day Sunshine by *The Beatles*. Original song and encoding with a 200 entry codebook of 2 bar patterns.

Offset	% of times chosen
0	**62.6**
1	16.5
2	9.4
3	11.5

Table 2: Bar alignment experiment: offsets relative to groundtruth 4-beat bar boundaries that gave minimum distortion encodings from the bar-aligned codebook.

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

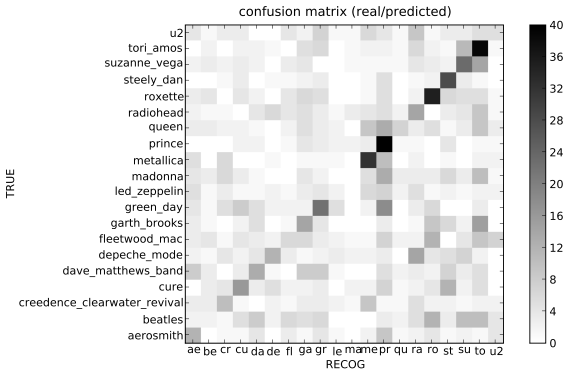
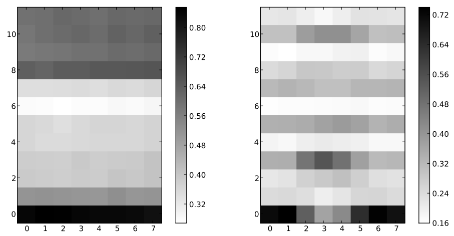


Figure 12: Confusion matrix for the artist recognition task.



(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].

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.

[3] See *Papers* section at `www.columbia.edu/~tb2332/`