LARGE-SCALE COVER SONG RECOGNITION USING THE 2D FOURIER TRANSFORM MAGNITUDE

Thierry Bertin-Mahieux
Columbia University
LabROSA, EE Dept.
tb2332@columbia.edu

Daniel P.W. Ellis
Columbia University
LabROSA, EE Dept.
dpwe@ee.columbia.edu

ABSTRACT

Large-scale cover song recognition involves calculating item-to-item similarities that can accommodate differences in timing and tempo, rendering simple Euclidean measures unsuitable. Expensive solutions such as dynamic time warping do not scale to million of instances, making them inappropriate for commercial-scale applications. In this work, we transform a beat-synchronous chroma matrix with a 2D Fourier transform and show that the resulting representation has properties that fit the cover song recognition task. We can also apply PCA to efficiently scale comparisons. We report the best results to date on the largest available dataset of around 18,000 cover songs amid one million tracks, giving a mean average precision of 3.0%.

1. INTRODUCTION

Music videos are abundant on the web, and tracing the dissemination of a given work is a challenging task. Audio fingerprinting [26] can be used to find an exact copy of a given music track, but the same technology will not work for finding novel versions of the original work, i.e. "cover songs". Since cover songs are typically recorded by musicians other than those who originally commercialized the track, one motivation for identifying such "covers" is to ensure the correct handling of songwriting royalties. For instance, on YouTube, the copyright holder of the musical work can have the covers of her work removed, or she can receive part of the advertising revenue from the video [1]. Figure 1 shows how easy it is to find thousands of such covers online from the metadata alone, but many more are not identified as covers. Another reason to study cover song recognition is that finding and understanding transformations of a musical piece that retain its essential identity can help us to develop intelligent audio algorithms that recognize common patterns among musical excerpts.

Until recently, cover recognition was studied on a small scale (a few hundred tracks) due in part to the scarcity of generally-available databases. Most current algorithms are based on comparisons between chroma patterns, or a related feature, using some form of dynamic time-warping (DTW) [23]. Chromas are derived from the spectrogram and provide a coarse approximation of the musical score (see Section 2), which make them suitable for our task. Recently, the release of the Million Song Dataset [1] (MSD), which contains metadata and audio features (including chromas) for one million songs, has spurred the investigation of large-scale music information retrieval techniques. Linked to this dataset, the SecondHandSongs dataset (SHS) identifies approximately eighteen thousand cover songs. The task of finding these cover songs within one million tracks makes it much closer to a commercial-scale application.

Very large datasets constrain the amount of computation that can be devoted to individual comparisons, making DTW an increasingly infeasible choice. To work with millions of examples, we would ideally reduce each comparison to a simple operation in a low-dimensional space. Such a space may be defined via hash codes extracted from the signal in the spirit of fingerprinting algorithms [2]. Hash codes can be efficiently indexed, and finding a song that contains particular hash codes is extremely fast. Another option is to project the entire song into a small fixed dimension space in which nearest neighbors are our candidate covers. Nearest neighbor methods are easy to parallelize

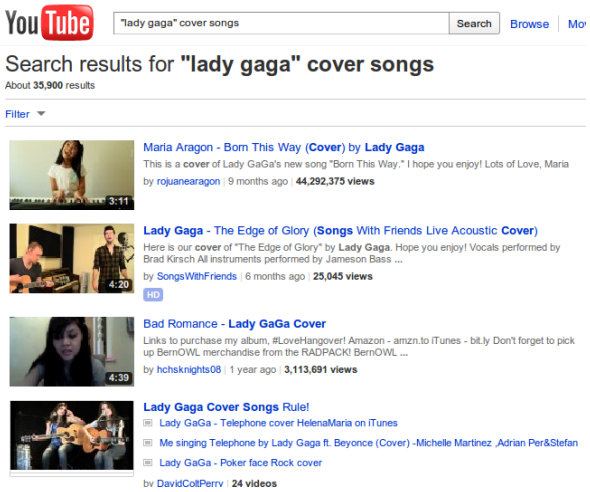

Figure 1. Search result for self-identified cover songs of Lady Gaga on YouTube on November 22nd, 2011. This simple query produces about 35, 900 results.

[1] http://www.youtube.com/t/faq

and scale, and working with a fixed-dimensional representation (instead of a variable-length audio signal) is a great convenience.

Disappointed by the results presented in [2], we focus on the second option. Beat-synchronous chroma representations form a relatively compact description that retains information relevant to covers, and may be cropped to a constant size. Unfortunately, direct comparison of chroma patterns using common metrics is poorly behaved [3]. Summarizing a song by extracting a few chroma patches and comparing them with Euclidean distance gives unusable results. In order to obtain an efficient nearest-neighbor algorithm, we need a representation for the chroma patches with the following properties:

- representation vectors can be compared using a simple metric, e.g. Euclidean distance;
- representation vectors are compact, i.e. low-dimensional;
- representation must be robust to semitone rotations (musical transpositions);
- representation should be robust to different pattern offsets (time alignments).

The last condition would allow us to match patches without having to identify a universal time alignment which is very difficult in a large set. Our candidate representation is the two-dimensional Fourier transform magnitude (2DFTM), a simplified version of the representation used in [18] and discussed in Subsection 3.2. The Fourier transform separates patterns into different levels of detail, which is useful for compacting energy (as in image compression schemes such as JPEG) and for matching in Euclidean space. Discarding the phase component provides invariance both to transposition (rotation) in the pitch axes and skew (misalignment) on the beat (time) axis. Thus, taking the 2DFTM of a chroma patch, we obtain a transformation of chroma features that makes Euclidean distance quite useful, even after dimensionality reduction through PCA. Our method encodes each track as a 50-dimensional vector and provides a large improvement over the hash code-based method [2]. On the SHS, this approach gives the best result reported so far.

The rest of this work is organized as follows: In Section 2, we present the chroma feature and its variants, its metric issues, and the task of cover song recognition. In Section 3, we describe how we transform the chroma matrices and look at the resulting invariance properties. Section 4 details our experiments on large-scale cover song recognition, and we conclude in Section 5.

2. PREVIOUS WORK

2.1 Chroma feature and distance

Chroma features were introduced as pitch-class profiles (PCP) [9]. Many variants have been derived, including HPCP [10] and CENS [21]; an overview can be found in [15]. There is even evidence that chromas can be learned from a simple similarity task [12].

Unfortunately, chroma matrices (or chroma patches, our term for chroma matrices with a fixed number of time samples) are high-dimensional features that are difficult to compare with usual metrics [3]. Previous work has experimented with Euclidean distance [3, 23], cosine distance [23], Kullback-Leibler divergence [3,22] and Itakura-Saito divergence [22]. None of the results were fully satisfying for the task of cover song recognition.

2.2 Cover song recognition

Cover song recognition has been widely studied in recent years, including a specific task within MIREX since 2007 [6]. An early system is Ellis and Poliner [8] and a good overview is in Serrà's thesis [23]. A significant amount of work has been done with classical music [16, 19–21] but popular music can present a richer range of variation in style and instrumentation.

Most cover song works were tested on a few hundred or thousand songs, a size not comparable to commercial collections (Spotify [2] claims more than 15M tracks). However, some of the work was made to scale and could be extended to larger sets. Kurth and Müller [17] use a codebook of CENS features to encode songs, thus creating an index that can be searched efficiently. Casey and Slaney [5] use locally-sensitive hashing (LSH) to quickly compare chroma patches, or *shingles*. Yu et al. [27] also use LSH to compare different statistics about a song. Kim and Narayanan [16] look at chroma changes over time and suggest using these changes as hash codes. Finally, our previous method [2] uses hash codes inspired by the fingerprint system of [26], i.e., identifying peaks in the chromagram and encode their relative positions. This was the first result reported on the SHS.

The idea of using the magnitude of the two-dimensional Fourier transform has been explored in [14, 18]. As with the methods above, these were tested on a few hundreds or thousands examples. The differences with the method used in this work are highlighted in Subsection 3.4

3. CHROMA FEATURE AND 2DFTM

Our feature representation is the magnitude of the two-dimensional Fourier transform of beat-aligned chroma patches. Below, we explain how they are computed and discuss their properties.

3.1 Chroma

We use the chroma features computed by the Echo Nest API [7] as described in [13]. They are available as part of the Million Song Dataset [1] and were used in [2].

A chromagram is similar in spirit to a constant-Q spectrogram except that pitch content is folded into a single octave of 12 discrete bins, each corresponding to a particular semitone (e.g., one key of the octave on a piano). For each song in the MSD, the Echo Nest analyzer gives a chroma vector (length 12) for every music event (called

[2] http://www.spotify.com

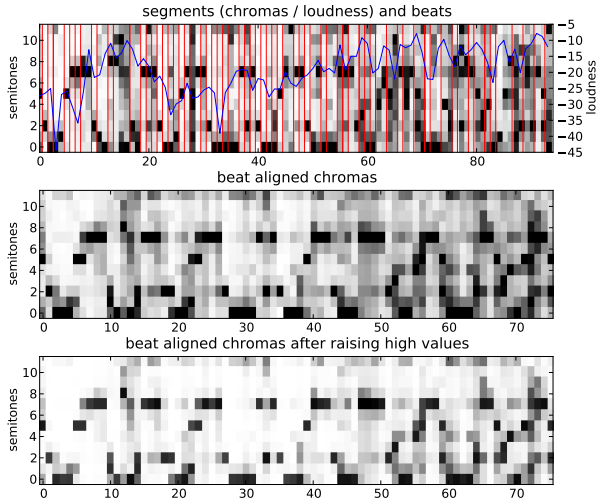

Figure 2. Beat-aligned chroma features.

"segment"), and a segmentation of the song into beats. Beats may span or subdivide segments. Averaging the per-segment chroma over beat times results in a beat-synchronous chroma feature representation similar to that used in [8]. Echo Nest chroma vectors are normalized to have the largest value in each column equal to 1.

Empirically, we found improved results from raising the highest values relative to the lowest ones by a power-law expansion. We believe this accentuates the main patterns in the signal. Figure 2 illustrates the stages in converting segment-aligned chroma features and their loudness to create beat-aligned chroma features for our task.

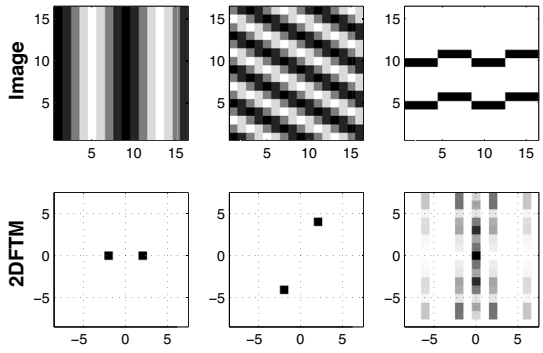

Figure 3. Examples of images and the corresponding 2DFTMs. FT images have the low-frequency values in the center.

3.2 Magnitude of the 2D Fourier Transform

Taking the two-dimensional Fourier transform is a common task in digital image processing where it is useful for denoising and compression, among other things [11]. As illustrated in the examples of Figure 3, a single point in the 2D Fourier transform corresponds to a sinusoidal grid of a particular period and orientation in the image (transform) domain; more complex images are built up out of multiple sinusoid grids.

We skip the definition of the 2D Fourier transform and its magnitude since it is widely known. Note that for visualization purposes, the bins are shifted so that the center of the axis system is in the middle of the image [3] . In that representation, the transformed image is symmetrical about the origin. Figure 4 gives an example of the transform applied to the chroma patch from Figure 2.

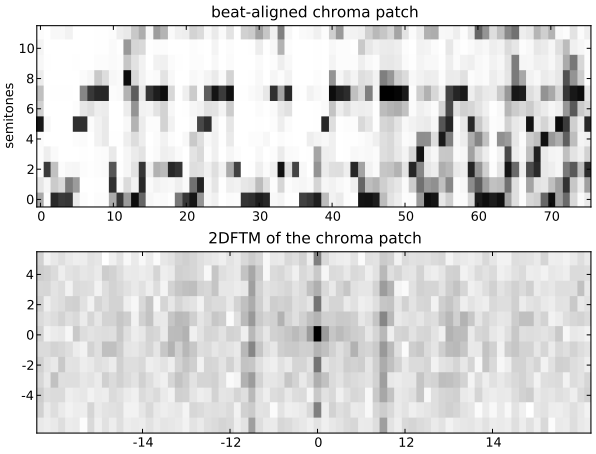

Figure 4. 2DFTM over a beat-synchronous chroma patch. Bins have been shifted so the maximum energy is in the center and magnitude values have been raised to the power 0.5 for visualization purposes. In the 2DFTM, the darker columns at -9 and $+9$ can be explained by the repetitive pattern in time in the chroma patch (bottom semitone) whose period is approximately 9 beats.

3.3 Rotation invariance

Analyzing nearest neighbors or clusters of chroma patches helps understand the properties of a given representation. Quantizing the chroma vector can be a useful step in an algorithm [20], but it can also be a goal on its own with the hope of seeing meaningful music patterns emerge [4]. The use of the 2DFTM introduces an interesting invariance, not only in key as in [4], but also in time, since small time shifts within a large patch correspond primarily to phase shifts in the Fourier components, with only slight changes in magnitude arising from edge effects. Figure 5 shows, for a randomly-chosen chroma patch at the top-left of the figure, the nearest neighbors obtained from the 2DFTM representation. For visualization purposes, we used 16-beat patches for this experiment. The result is noisy, but we see a clear square wave pattern that is repeated with different onsets in the first three neighbors.

3.4 Comparison with Similar Previous Approaches

Our method resembles those of [14, 18], but some steps are simplified in view of the size of the collection at hand. In [14], instead of beat-aligned chromagrams, the authors use two-dimensional autocorrelation, then, for each semitone, take 17 samples spaced logarithmically in time (to normalize tempo) building a 12x17 feature matrix for each song. Autocorrelation is the Fourier transform of the Fourier Transform magnitude we use.

[3] *fftshift* in MATLAB, *scipy.fftpack.fftshift* in Python

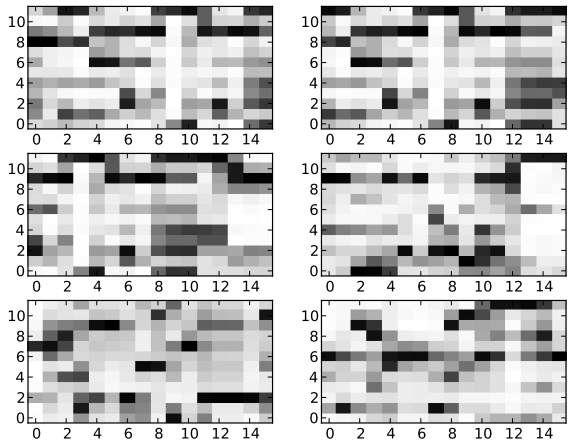

Figure 5. Neighbors of the upper left patch using 2DFTM representation (including raising values to power 1.96). The square signal shape, shifted in time, is visible in the first two neighbors (upper right, middle left) and partially in the middle right neighbor.

Compared to [18], we do not extract melodic lines, and we represent each track with an aggregate patch rather than storing many overlapping patches. We use PCA as the domain for comparison to improve generalization. Marolt describes time-shift invariance of the 2DFTM as an issue, but for us it is the main attraction. We found 75 beat patches to be most useful compared to the optimum at 32 in [18], perhaps because of PCA. A main emphasis of his work is the use of LSH, but reducing each track to a single pattern makes such indexing unneccessary, even for our much larger database.

4. COVER SONG RECOGNITION

Our goal is to find cover songs among mostly western pop music. The experiments are conducted on the SecondHandSongs dataset (SHS). The SHS was created by matching the MSD with the online database [4] that aggregates user-submitted cover songs. The SHS training set contains $12,960$ cover songs from $4,128$ cliques (musical works), and the testing set has $5,236$ tracks in 726 cliques. Note that we use the term "cover" in a wide sense. For instance, two songs derived from the same original work are considered covers of one another.

4.1 Method

We will represent each song by a vector of fixed-length, defining a point in Euclidean space. The closer two feature vectors lie in this space, the more likely the songs are covers. The steps to compute this feature vector for a song are summarized as follows:

1. obtain chroma from the MSD
2. resample onto beat grid
3. apply power-law expansion
4. extract FFT patches and keep magnitude
5. take the median
6. apply PCA

The first step is done using the code provided with the MSD. One can also call The Echo Nest API [7] if starting from audio (see Subsection 4.4). Beat estimation is also provided in the MSD, and the second step is mostly a linear scaling. The third step, in which we enhance the contrast between strong and weak chroma bins by raising the values to an exponent, was determined to be useful in our experiments. In the fourth step, we take the 2DFTM as shown in Figure 4 for every 75-beat long chroma patch. In the fifth step we keep the median, within each bin, across all the transformed patches. Finally, in the sixth step, we use PCA to reduce the dimensionality. PCA is done without normalization, the data is simply centered to zero and rotated. In our experiments PCA was computed on 250K songs that were not identified as covers (i.e., in neither the SHS train or test sets).

Many of the parameters above were chosen empirically. We do not present the list of parameters we tried for lack of space and the little interest they carry. Simply note that the following parameters were chosen based on their ability to identify 500 train covers (see Subsection 4.3, first experiment):

- patches of size 75 beats, we tried number of beats ranging from 8 to 100;
- median as opposed to average;
- raising to the power 1.96, we tried values between 0.25 and 3.0.

The number of PCs we keep after PCA is also a parameter, but choosing the best one is more difficult. The number of PCs is a trade-off between accuracy and feature vector size (and hence speed). We believe 50 is the "best" trade-off, but we report results for other numbers of PCs. Still regarding PCA, since we use chroma patches of 75 beats, we have $12 \times 75 = 900$ principal components. Note that half of the components (450) are redundant due to the symmetry in the 2DFTM, and have a variance of zero associated to them.

4.2 Reference methods

We compare our algorithm to other methods, at different scales depending on the speed of the algorithm. We start with our previous work [2], the only reported result on the SHS to our knowledge. Also taken from that work, we report again a comparison with the method from [8].

We also test a DTW-based method based on [24] using code from S. Ravuri [5]. This is more of a sanity check than a full comparison; the authors in [24] used up to 36 semitones instead of the 12 we possess, we did not re-estimate the DTW parameters, etc. It is likely that the full system from [24] outperforms our method, the problem being the execution time which is prohibitive on a large scale. In our implementation, each DTW comparison takes on the order of 10 ms. One query on the MSD would therefore take

[4] www.secondhandsongs.com

[5] http://infiniteseriousness.weebly.com/cover-song-detection.html

about 2.7 hours. Thus we do this comparison only on our 500 binary queries.

Finally, we compare with pitch histograms, a feature that was suggested for music identification in [25]. The pitch histogram of a song is the sum of the energy in the 12 semitones normalized to one. In our case, we compute it from beat-aligned chroma features. This feature is not powerful enough to perform cover recognition on its own, but it gives an idea of how much more information our method can encode in a $\sim$ 10-dimensional feature.

4.3 Experiments

Method	accuracy
random	50.0%
pitch hist.	73.6%
correlation	76.6%
DTW	80.0%
jcodes 1 [2]	79.8%
jcodes 2 [2]	77.4%
2DFTM (full)	82.0%
2DFTM (200 PC)	82.2%
2DFTM (50 PC)	**82.2%**
2DFTM (10 PC)	79.6%
2DFTM (1 PC)	66.2%

Table 1. Results on 500 binary tasks. PC is the number of principal components we retain after PCA. Empirically, 50 PC appear to be the best trade-off between accuracy and size.

We follow the methodology of [2] where the parameters are first tuned on a subset of 500 binary tasks created within the SHS training set (we use the same 500 queries). The goal is: Given a query song A and two songs B and C, find which of B or C is a cover of A. The result is the percentage of trials where the algorithm succeeds. We then present the results testing on the training set, mostly as a sanity check. Finally, we report result on the SHS test set using the full MSD.

In [2], the main reported result was the average rank of a known cover given a query. For instance, on 1M songs, picking at random would give 500K. We again report this measure to permit comparison, but for practical purposes this number may be misleading since it is dominated by the most difficult covers, of which there will always be a number, and hides differences in performance near the top of the ranking. We now prefer to report results in terms of mean average precision (meanAP), which puts emphasis on results that rank high, i.e., the covers that are in the top k results. present the recall curves for a few algorithms (see Figure 6).

As we see in Table 3, the method based on our 2DFTM representation provides a significant improvement over the method in [2] for both measures. In particular, based on meanAP, many more covers are ranked in the first few hundred results, which makes it much more valuable in a commercial system. Note that a second, slower step could be applied to the top k results, k being 1K or 10k, similar to the progressive refinement in [27]. A good candidate for this second step would be the full system of [24].

Method	average rank	mean AP
pitch hist.	4,032.9	0.01851
2DFTM (full)	3,096.7	0.08912
2DFTM (200 PC)	3,005.1	**0.09475**
2DFTM (50 PC)	**2,939.8**	0.07759
2DFTM (10 PC)	3,229.3	0.02649
2DFTM (1 PC)	4,499.1	0.00186

Table 2. Results on the training set (12,960 songs). For average rank, lower is better. For meanAP, higher is better.

Method	average rank	mean AP
random	500,000	$\sim$ 0.00001
pitch hist.	268,063	0.00162
jcodes 2	308,370	0.00213
2DFTM (200 PC)	180,304	**0.02954**
2DFTM (50 PC)	**173,117**	0.01999
2DFTM (10 PC)	190,023	0.00294
2DFTM (1 PC)	289,853	0.00003

Table 3. Results on 1M songs. For average rank, lower is better. For meanAP, higher is better.

Using the system with 50 principal components, Figure 6 shows us that more than half of the covers are ranked in the top 100k and more than a quarter of them are in the top 10k. For 112 queries, the first song returned was a known cover. We ignore here the songs that might have ranked second or third after other known covers. This includes the pairs *(Hayley Westenra, Kate Bush)* performing "Wuthering Heights" and *(The Boomtown Rats, G4)* performing "I don't like Mondays", matches considered *easy* in [2].

In terms of speed, with a 50-dimensional vector per song, ranking all one million for all 1,726 test covers in Python took 1h 46min on a machine with plenty of RAM. This is around 3-4 seconds per query without any optimization or parallelization. Compared to the 2.7 hours of the DTW method, that is a $\sim$ 2,500x speedup.

4.4 Out of collection data

Using audio as the query input with the SHS is a challenge, beat-synchronous features relies on a consistent beat tracker. Fortunately, The Echo Nest provides an open API which will convert any uploaded audio into the format provided in the Million Song Dataset. We experimented with this using cover songs found on YouTube. For instance, the song "Summertime" by *Ella Fitzgerald and Louis Armstrong* [6] was correctly associated with a cover version by

[6] `http://www.youtube.com/watch?v=MIDOEsQL7lA`

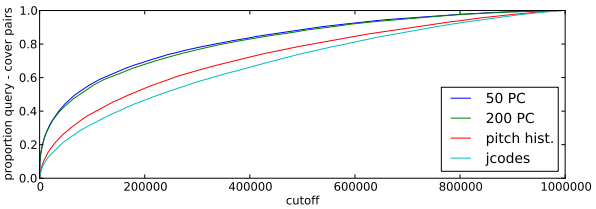

Figure 6. Recall using (*query, cover*) pairs on the full million songs for different systems. Legend is in order from best (upper left corner) to worst.

Joshua Redman (first match). The *Ella Fitzgerald and Louis Armstrong* version present in the SHS was found at rank 9. The fact that this was not the first match might be explained by the lower audio quality on YouTube, or it could be a different version.

5. CONCLUSION

The 2DFTM allows us to pose the search for cover songs as estimating an Euclidean distance. We show that this representation exhibits some nice properties and improves the state-of-the-art on large-scale cover song recognition. Furthermore, obtaining good results using patches of 75 beats suggests an easy way to include more time dependency in the harmonic representation. Future work will look into the usefulness of this representation for other tasks, such as music tagging and segmentation. Finally, all our code is available online [7].

6. ACKNOWLEDGMENTS

The authors would like to thank The Echo Nest and SecondHandSongs for making the MSD and the SHS available respectively. Thanks to S. Ravuri for his DTW code. TBM is a NSERC PGD scholar. This work has been supported by a gift from Google and by the NSF under grant IIS-0713334.

7. REFERENCES

[1] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of ISMIR*, 2011.

[2] T. Bertin-Mahieux and D.P.W. Ellis. Large-scale cover song recognition using hashed chroma landmarks. In *Proceedings of WASPAA*, New Platz, NY, 2011.

[3] T. Bertin-Mahieux, G. Grindlay, R. Weiss, and D. Ellis. Evaluating music sequence models through missing data. In *Proceedings of ICASSP*, 2011.

[4] T. Bertin-Mahieux, R. Weiss, and D. Ellis. Clustering beat-chroma patterns in a large music database. In *Proceedings of ISMIR*, 2010.

[5] M. Casey and M. Slaney. Fast recognition of remixed music audio. In *Proceedings of ICASSP*. IEEE Signal Processing Society, 2007.

[6] 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.

[7] The Echo Nest Analyze, API, `http://developer.echonest.com`.

[8] D. Ellis and G. Poliner. Identifying cover songs with chroma features and dynamic programming beat tracking. In *Proceedings of ICASSP*. IEEE Signal Processing Society, 2007.

[9] T. Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In *Proceedings of the International Computer Music Conference (ICMC)*, 1999.

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

[7] `http://www.columbia.edu/~tb2332/proj/coversongs.html`

[11] R.C. González and R.E. Woods. *Digital image processing*. Pearson/Prentice Hall, 2008.

[12] Ö. İzmirli and R.B. Dannenberg. Understanding features and distance functions for music sequence alignment. In *Proceedings of ISMIR*, 2010.

[13] T. Jehan. *Creating music by listening*. PhD thesis, Massachusetts Institute of Technology, 2005.

[14] J.H. Jensen, M.G. Christensen, and S.H. Jensen. A chroma-based tempo-insensitive distance measure for cover song identification using the 2d autocorrelation function. MIREX cover song identification contest, 2008.

[15] N. Jiang, P. Grosche, V. Konz, and M. Müller. Analyzing chroma feature types for automated chord recognition. In *Proceedings of the 42nd AES Conference*, 2011.

[16] S. Kim and S. Narayanan. Dynamic chroma feature vectors with applications to cover song identification. In *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, pages 984–987. IEEE, 2008.

[17] F. Kurth and M. Müller. Efficient index-based audio matching. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):382–395, 2008.

[18] M. Marolt. A mid-level representation for melody-based retrieval in audio collections. *Multimedia, IEEE Transactions on*, 10(8):1617–1625, 2008.

[19] R. Miotto, N. Montecchio, and N. Orio. Content-based cover song identification in music digital libraries. In *Digital Libraries*, pages 195–204. Springer, 2010.

[20] R. Miotto and N. Orio. A music identification system based on chroma indexing and statistical modeling. In *Proceedings of ISMIR*, 2008.

[21] M. Müller, F. Kurth, and M. Clausen. Audio matching via chroma-based statistical features. In *Proceedings of ISMIR*, London, UK, 2005.

[22] L. Oudre, Y. Grenier, and C. Févotte. Chord recognition by fitting rescaled chroma vectors to chord templates. *IEEE Transactions on Audio, Speech and Language Processing*, 19(7):2222 – 2233, Sep. 2011.

[23] J. Serrà. *Identification of versions of the same musical composition by processing audio descriptions*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2011.

[24] 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, and Language Processing*, 16(6):1138–1151, 2008.

[25] G. Tzanetakis, A. Ermolinskyi, and P. Cook. Pitch histograms in audio and symbolic music information retrieval. *Journal of New Music Research*, 32(2):143–152, 2003.

[26] A. Wang. An industrial strength audio search algorithm. In *Proceedings of ISMIR*, 2003.

[27] Y. Yu, M. Crucianu, V. Oria, and L. Chen. Local summarization and multi-level LSH for retrieving multi-variant audio tracks. In *Proceedings of the seventeen ACM international conference on Multimedia*, pages 341–350. ACM, 2009.