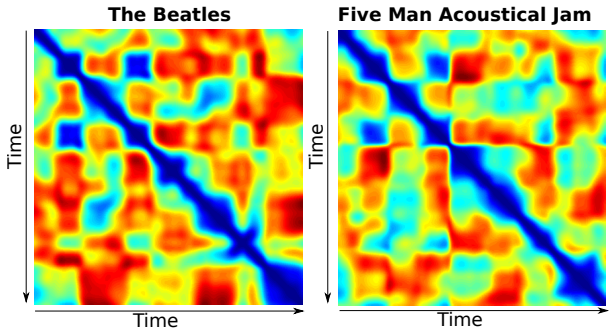
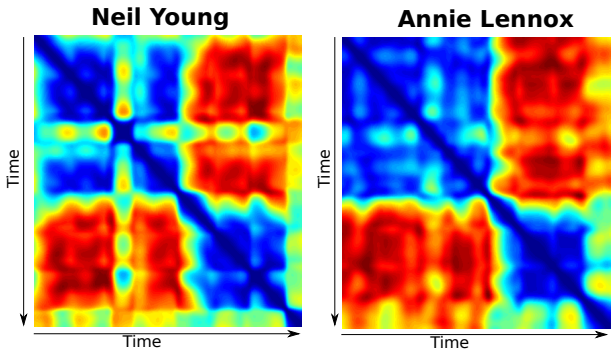
MIREX 2015: COVER SONGS VIA SEQUENCES OF LOCAL MFCC SELF-SIMILARITY MATRICES

Christopher J. Tralie
Duke University Department of
Electrical and Computer Engineering
chris.tralie@gmail.com

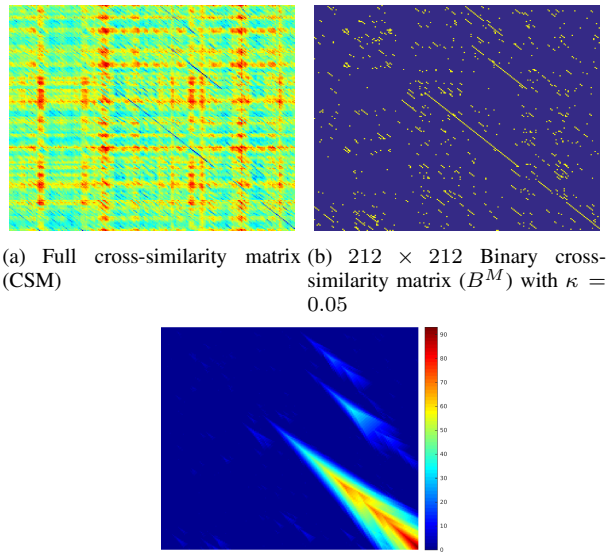


(a) A block of 4 beats with 400 windows sliding in the song "We Can Work It Out" by The Beatles with a cover by Five Man Acoustical Jam



(b) A block of 4 beats with 400 windows sliding in the song "Don't Let It Bring You Down" by Neil Young with a cover by Annie Lennox.

Figure 2. Two examples of MFCC SSM blocks which were matched between a song and its cover in the covers80 dataset. Hot colors indicate windows in the block are far from each other, and cool colors indicate that they are close.



(a) Full cross-similarity matrix (CSM)

(b) 212×212 Binary cross-similarity matrix (B^M) with $\kappa = 0.05$

(c) Smith Waterman with local constraints: Score 93.1

Figure 3. Cross-similarity matrix and Smith Waterman on MFCC-based SSMs for a true cover song pair of "We Can Work It Out" by The Beatles and Five Man Acoustical Jam.

1. INTRODUCTION

The purpose of this MIREX submission is to test an algorithm recently developed in [4] for cover song identification. While most previous approaches to cover songs have focused on chromagram representations or representations of note sequences, we found an approach that examines sequences local MFCC changes. This approach has been shown to work suprisingly well on the Covers 80 dataset [3] (44/80), considering absolute pitch information is significantly obscured in this representation. This shows that in spite of conventional wisdom, there are other "invariants to cover" beyond simply the notes. We like to think of our MFCC self-similarity matrices as summarizing "relative acoustic flow" or "high level riff information," which is preserved between different versions of the same song with different instruments, vocal balance, singers, etc.

More significant algorithmic details can be found in [4]. This document will mainly describe some implementation details, links to code used, and further experiments that were not covered in [4].

2. SYSTEM OVERVIEW

Figure 1 shows a block diagram of our system. We summarize small blocks of audio (on the order of 10 beats) by $d \times d$ self-similarity matrices (SSMs) of time-ordered MFCC features in those blocks. Then, treating each SSM as a d^2 dimensional vector equipped with the L^2 norm, we match sequences of these SSMs using the Smith Waterman algorithm. We like to think of these sequences of SSMs as "relative timbral shape sequences," because they describe the "shape" of a point cloud of MFCC features in a block of audio which is invariant to rotation and translation in the feature space. In more detail, the key steps of our algorithm (Figure 1) are as follows:

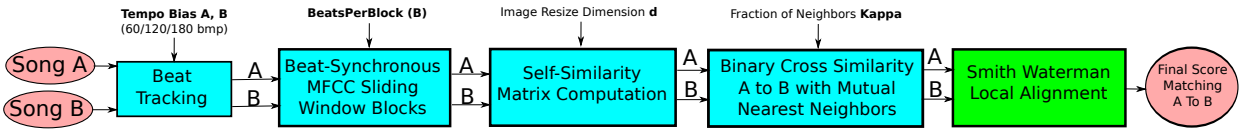


Figure 1. A block diagram of our system

1. Extract beat onset locations using code from [2]. This is used to control for tempo changes between cover songs

2. For every block of B contiguous beats in the song, extract MFCC features in each block using code from [1]. We use an unusually long MFCC window size which is the average length of a beat in the song, and we use a hop size which is $1/10^{\text{th}}$ of a beat (so roughly $10B$ MFCC windows per block). Note that in our original experiments in [4] we took a hop size of $1/200^{\text{th}}$ of a beat, but we found in further experiments (Section 3) that many fewer windows were adequate for this application. We like to think of the long window size as a sort of anti-aliasing before creating a lower resolution MFCC self-similarity matrix, but more theoretical analysis would be needed to fully back that claim.

3. Point-center and normalize the MFCC windows in each block, and compute a self-similarity matrix of the MFCC features in that block (we also tried simply discarding the zeroeth MFCC coefficient to control for loudness, as suggested by reviewers, but we found point centering and normalizing to be superior). Since the tempo is not completely stable from beat to beat, there may be a slightly different number of windows for each beat block. To make comparisons between different blocks, we simply resize the SSMs all to the same dimension d. Figure 2 shows some higher resolution examples of MFCC self-similarity matrices taken from [4], which show how these images are similar in blocks from very different realizations of the same song.

4. Given all of the SSMs for all beat blocks from two songs A and B, create a cross-similarity matrix, where each element (i, j) stores the L^2 distance between the SSM corresponding to the i^{th} beat block from song A and the j^{th} beat block from song B.

   Because each every pair of self-similarity matrices needs to be compared for every pair of cover songs, this step can be very computationally intensive. To speed it up, we exploited fast matrix multiplication of BLAS in Matlab as follows (assuming all SSMs from song A are stored as rows in the matrix $D1$ and likewise for song B and $D2$):

```
CSM = bsxfun(@plus,...
dot(Ds1,Ds1,2),dot(Ds2, Ds2,2)')...
-2*(Ds1*Ds2');
```

   We observed a speedup of several orders of magnitude over Matlab's built-in pdist2 for this application with a large number (d^2) of columns.

5. For each cross similarity matrix, we use the Smith-Waterman algorithm to score the longest diagonal, which indicates the longest sequence of beat blocks which are in common between two songs. We found a binary thresholding before applying Smith Waterman made our algorithm more robust, and our code accepts a parameter κ which controls the fraction of nearest neighbors considered for each point in song A to neighbors in song B. Figure 3 shows the cross-similarity matrix (CSM), binary cross-similarity matrix, and Smith Waterman algorithm on a cover song pair from the Covers 80 dataset.

The Smith Waterman score is computed on the cross-similarity matrices between all pairs of cover songs in a database and all cover songs in a query set, which leads to final ranking of each song in a database to each query song. A more formal, detailed description of this algorithm can be found in [4].

3. NEW COVERS 80 EXPERIMENTS

Here we show some results in addition to the ones presented in [4]. In order for our algorithm to scale to a larger dataset, we experimented with lower resolution self-similarity images (50×50 instead of 200×200) created from much coarser MFCC information (10 windows per beat instead of 200), and we found this had little effect on the results. The table below shows the results for various choices of κ and beats per block B, all for an SSM dimension d. The highest number of correct songs identified is 44/80, though the results are fairly stable for different parameter choices. For the MIREX benchmark, we choose $\kappa = 0.15$ and $B = 16$, as this gives good results with low median/mean rank and a smaller number of beats per block than other parameter choices which gave this result, meaning computation of SSMs is slightly faster.

4. CODE

Please visit `github.com/ctralie/PublicationsCode` under *ISMIR2015_CoverSongsShape* for all code related to this project, including an interactive GUI for examining cross-similarity matrices between two songs and the self-similarity matrices compared at each pixel

Kappa = 0.05	B = 6	B = 8	B = 10	B = 12
d = 50	24 (8.5 / 19.575)	29 (6 / 17.3)	30 (5.5 / 15.9875)	34 (3 / 13.9125)
Kappa = 0.1				
d = 50	27 (11 / 21.0625)	32 (6 / 17.95)	39 (2 / 11.7875)	40 (1.5 / 13.0125)
Kappa = 0.15				
d = 50	27 (7 / 17.4875)	34 (2 / 14.1)	39 (2 / 14.15)	42 (1 / 12.4625)
Kappa = 0.2				
d = 50	26 (6 / 17.8375)	29 (5.5 / 16.6625)	34 (2.5 / 15.875)	38 (2.5 / 15.225)
Kappa = 0.05	B = 14	B = 16	B = 18	
d = 50	39 (2 / 12.45)	42 (1 / 13.525)	42 (1 / 12.1125)	
Kappa = 0.1				
d = 50	39 (2 / 12.1875)	43 (1 / 11.4625)	43 (1 / 12.5625)	
Kappa = 0.15				
d = 50	42 (1 / 13.5125)	44 (1 / 12.5)	40 (1.5 / 12.9)	
Kappa = 0.2				
d = 50	42 (1 / 14.8625)	41 (1 / 13.9125)	40 (1.5 / 13.2375)	
Kappa = 0.05	B = 20	B = 22	B = 24	
d = 50	41 (1 / 11.6875)	41 (1 / 11.3)	44 (1 / 11.5625)	
Kappa = 0.1				
d = 50	42 (1 / 13.2)	42 (1 / 14.0875)	41 (1 / 13.6375)	
Kappa = 0.15				
d = 50	42 (1 / 13.35)	43 (1 / 12.4)	44 (1 / 13.1625)	
Kappa = 0.2				
d = 50	41 (1 / 13.4625)	40 (1.5 / 13.85)	40 (1.5 / 13.775)	

Table 1. New results on the Covers80 experiment, varying κ and B for $d = 50$. Number correct is shown, along with (medan rank / mean rank) in parentheses.

5. ACKNOWLEDGEMENTS

Chris Tralie was supported under NSF-DMS 1045133 and an NSF Graduate Fellowship.

6. REFERENCES

[1] Dan Ellis. Plp and rasta (and mfcc, and inversion) in matlab using melfcc. m and invmelfcc. m, 2006.

[2] Daniel PW Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[3] Daniel PW Ellis. The "covers80" cover song data set. *URL: http://labrosa. ee. columbia. edu/projects/coversongs/covers80*, 2007.

[4] Christopher J Tralie and Paul Bendich. Cover song identification with timbral shape sequences. *arXiv preprint arXiv:1507.05143*, 2015.