Published at ICASSP-2007, Honolulu, Hawaii.

FAST RECOGNITION OF REMIXED MUSIC AUDIO

*Michael Casey**

Goldsmiths, University of London
Department of Computing
New Cross, London, UK

Malcolm Slaney

Yahoo! Research, Inc.
Sunnyvale
California, USA

ABSTRACT

We present an efficient algorithm for automatically detecting remixes of *pop* songs in large commercial collections. Remixes are closely related as commercial products but they are not closely related in their audio spectral content because of the nature of the remixing process. Therefore spectral modelling approaches to audio similarity fail to recognize them. We propose a new approach– that chops songs into small chunks called *audio shingles*– to recognize remixed songs. We model the distribution of pair-wise distances between shingles by two independent processes– one corresponding to remix content and the other correspoding to non-remix content in a database. A nearest neighbour algorithm groups songs if they share shingles drawn from the remix process. Our results show 1) log-chromagram shingles separate remixed from non-remixed content with 75%-75% precision-recall performance, cepstral coefficient features do not separate the two distributions adequately 2) increasing the observations from the remix distribution increases the separability. Efficient implementation follows from the separability of the distributions using locality sensitive hashing (LSH) which speeds up automatic grouping of remixes by between one to two orders of magnitude in a 2018-song test set.

Index Terms— Music, Statistics, LSH, Shingles, Databases

1. INTRODUCTION

Today's large commercial music collections contain millions of songs and their associated metadata. One of the major challanges for managing such collections is identifying and eliminating 'duplicate' entries, that is, catalogue items that are closely related from the users' perspective. In a commercial catalogue, songs may be considered closely related even if their spectral content is not similar for the most part. This is the case with remixes– a vocal sample is taken from a source recording and summed into a completely new musical work created by a *producer* or *DJ*. Even though the new song is almost entirely composed of different material from the source, the two songs are presented as different instances of the same title to the user. For example, the Madonna track *Nothing Fails* has 10 different versions including the remixes *Nevins Mix, Jackie's In Love In The Club Mix, Nevins Dub Mix, Tracy Young's Mix, Big Room Rock Mix, Classic House Mix* and a *Radio Remix*. User navigation of such catalogues can become weighed down with these duplicate entries thus impacting browsing efficiency, obscuring the users query and reducing sales.

The problem is closely related to that of near-duplicate text documents found on the World Wide Web. Search engine technology has been developed to eliminate near-duplicate or derivative items from internet searches using the technique of shingling [2]. In previous work we demonstrated the use of audio shingles on a derivative work (remix) retrieval task and we demonstrated that an efficient algorithm based on locality sensitive hashing (LSH) could efficiently identify them in a collection of 2000 songs, [4].

Our new contributions in this paper are 1) a remix recognition algorithm 2) a statistical analysis of the properties of remixes in audio feature space 3) experimental evidence that the remix problem, as stated, is well-posed 4) details of efficient implementation using locality sensitive hashing.

The structure of the paper is as follows: we review different approaches to audio similarity in Section 2, we develop a model for remix processes using distributions of Euclidean distances between audio shingles in Section 3, we then given details of remix recognition algorithms in Section 4 and we present the results of remix recognition experiments on a 2000 song data set in Section 5. We conclude with the implications of our results for efficient audio matching applications in Section 6.

2. PREVIOUS WORK

There is a wide spectrum of previous work covering a range of music-similarity tasks: from very specific fingerprinting work [9][17] to genre recognition [18][12]. This work falls in the middle. We want to find songs that are similar, but not exactly like another song. Our tasks needs both new features–we don't expect the very specific features used in fingerprinting to work–and a new matching criteria because we expect that remixes will have segments rearranged and new material inserted.

Our similarity definition means that our work is different from the work that has been done on audio fingerprinting [14][15][16][17]. With fingerprinting users want to find the name of a recording given a sample of the audio. The secret sauce that makes fingerprinting work is based on defining robust features of the signal that lend the song its distinctive character, and are not harmed by difficult communications channels (i.e. a noisy bar or a cell phone). These systems assume that some portion of the audio is an exact match—this is necessary so they can reduce the search space. We do not expect to see exact matches in remixes.

At the other end of the specificity scale, genre-recognition [18], global song similarity [12], artist recognition [5], musical key identification [11], and speaker identification [13] use much more general models such as probability densities of acoustic features approximated by Gaussian Mixture Models. These so-called bag-of-feature models ignore the temporal ordering inherent in the signal and, therefore, are not able to identify specific content within a musical work such as a given melody or section of a song.

*This research supported by Engineering and Physical Sciences Research Council Grant GR/1xxx/X.

2.1. Matching with Locality Sensitive Hashing

Our audio work is based on an important web algorithm known as shingles and a randomized algorithm known as locality-sensitive hashing (LSH) [2]. Shingles are a popular way to detect duplicate web pages and to look for copies of images. Shingles are one way to determine if a new web page discovered by a web crawl is already in the database. Text shingles use a feature vector consisting of word histograms to represent different portions of a document. Shingling's efficiency at solving the duplicate problem is due to an algorithm known as a locality-sensitive hash (LSH). In a normal hash, one set of bits (e.g. a string) is transformed into another. A normal hash is designed so that input strings that are close together are mapped to very different locations in the output space. This allows the string-matching problem to be greatly sped up because it's rare that two strings will have the same hash.

Our previous work we showed that matched filters, and therefore Euclidean distance, using chromagram and cepstral features performs well for measuring the similarity of passages within songs [3][4]. The current work applies these methods to a new problem, grouping of derived works and source works in a large commercial database using an efficient implementation based on LSH.

3. MODELING REMIXES

3.1. The Dataset

The data consist of 2018 songs chosen to be the complete catalogues of two artists (Miles Davis and Madonna) drawn from the Yahoo! Music Universal database. In this paper we focus on remixes in the set of Madonna songs, we use the Miles Davis songs as an extended database to test robustness against different artists' music data. The full catalogue of Madonna songs has a higher proportion of remix works than works with no remix versions.

3.2. Features

The remix songs share a small aspect of their content, usually a prominant vocal sample. The sample is specific, so we seek the specific audio content shared between songs. However, the difference in context can be significant– comprising different sets of instruments (bass, keyboards, drums, etc.) and different rhythmic / melodic components between remixed shingles. So we seek features that are robust to the change in context. To this end we choose a pitch-based feature (PCP) so that the specific pitch-sequence content used in a remix is represented.

3.3. Additive Noise / Silence Removal

We assume a volume process applied to a song with additive noise. We do not want silence or noise to be included in the matches between songs since these two processes are generic to all songs and would corrupt the recognition of similar content. We removed silence by thresholding audio segments by the geometric mean of the shingles' power in each song. Whilst this may seem an aggressive threshold, it is also reasonable to assume that remix content will be prominant aspects of the song.

3.4. Audio Features

We extracted Log-Frequency Cepstral Coefficients (LFCC) and pitch-class profiles (PCP) as follows: features were extracted from uncompressed audio sourced from the Yahoo! Music database. All files were 44.1kHz stereo, mixed to mono, 16384-point hamming windowed with 4410-sample hop and a 16384-pt DFT computed using the FFT. The window size was chosen such that two frequency samples were available from the DFT for the lowest pitch class $C0 = 65.4Hz$. The band edges were chosen at the mid point (quatertone) between chromatic pitch classes with the low edge set to $63.5Hz$ and high edge of $7246.3Hz$, a quatertone above $A7$.

The 8193 DFT magnitudes were assigned to pitch class bands with samples near the band edges shared proportionally between the bands [1]. The remaining DFT coefficients were disposed. The pitch-class assignments were then folded into a single octave by summing over all octaves for each pitch class, and the logs of the values were taken yielding the 12-dimensional pitch-class-profile (PCP) vector every 100ms.

The LFCC features used the same constant-Q transform as the PCP, then the log of the values was taken and the result transformed to cepstral coefficients using the DCT yielding 20 coefficients per 100ms. We note that this feature is a slightly modified form of the widely used MFCC feature.

3.5. Audio Shingles

Audio shingles concatenate feature vectors into high-dimensional vectors. Informed by previous studies [4][10] we used window of $4s$ with a hop of $0.1s$ yielding $10Hz \times 20d \times 4s = 800-$dimensions for LFCC and $10Hz \times 12d \times 4s = 480-$ dimensions for PCP.

4. RECOGNIZING REMIXES

We first propose remix recognition as a test between two songs; so to identify all remixes in a database, all pairs of songs must be tested. We later show how to recognize remixes without exhaustive selection from the database with full control over potential loss of accuracy.

Let $\mathbf{A}, \mathbf{B} \in \{\mathbf{S}\}$ denote two songs drawn randomly from a database and $x^i \in \mathbf{A}, y^j \in \mathbf{B}$ be shingles drawn from the songs. We then define the remix distance:

$$\delta_{remix}(\mathbf{A}, \mathbf{B}) = \sum_{N} \min_{i,j}^{N} \sum_{k} |x_k^i - y_k^j|^2,$$

this is the sum of the N minima of the pair-wise shingle distances between the songs. These are obtained by sorting the distances in acending order and summing the first N values. The complexity of this classifier, as stated, is $\mathbf{O}(n^2 + log(n))$ in the number of shingles in the database. However, the sort can be computed efficiently, as can the pair-wise distance computations.

To classify, define a scalar threshold, r_0, on the remix distance and perform the test $\delta_{remix}(\mathbf{A}, \mathbf{B}) < r_0$. If the remix distance falls below the threshold we label the song-pair as a remix.

4.1. Justification of the threshold approach

The parameter r_0 is critical in our method, and seems at first over simplistic. To justify its use we conducted an experiment chord root recognition on varied synthetic complex-tone audio data. The data consisted of 120 chords for 12 pitch-chroma class. We synthesized audio using Matlab by additive synthesis with randomly varying harmonics, formant frequencies, note octaves (7-octave range) and note intensities. Chord inversions were used so the root randomly varied in the voicing. Between two and six notes were used chosen from the major and minor triads for each pitch class. Classifiers were trained

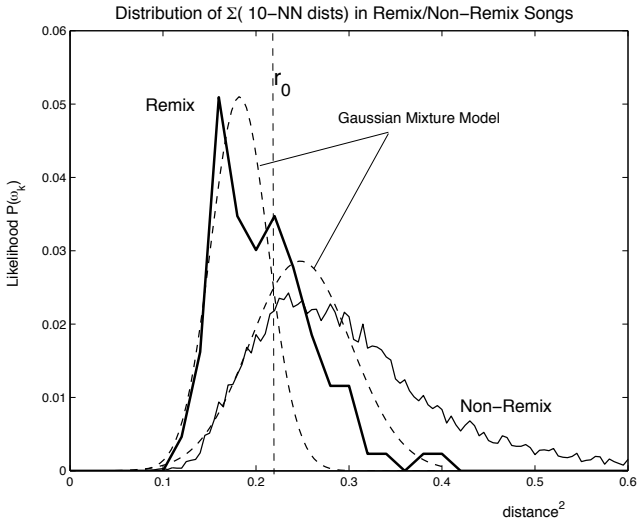

Fig. 1. Distribution of remix and non-remix between-song shingle distances and GMM fit to remix distribution (only) showing the choice of threshold as a likelihood ratio of two classes (True Positives and False Positives) in the target remix distribution. The False Positives are clearly drawn from the non-remix distribution.

on a subset of 70% of each class, with 30% held back for testing. All classifiers were trained and tested on the same random permutation per 10-fold cross-validation.

Classifiers were 1) 12-way support vector machine classifiers with a linear kernel (SVM-Lin) 2) polynomial kernel (SVM-Pol) and 3) k-NN classifier, with $k = 12$, consisting of the mean vector of the training data frames for each pitch class.

Table 1. Performance of kernel-feature-classifier combinations in a chord-root classification experiment

Classifier	linear PCP	log PCP	LFCC
SVM-Lin	75.2	99.9	23.74
SVM-Pol	98.2	99.8	45.81
k-NN	77.1	98.5	21.33

We infer from Table 1 the following: 1) PCP with logarithmic kernel increases seperability of similar harmony content in the presence of acoustic variation 2) the increase is due to the existence of threshold discriminant in the kernel space 3) the k-NN classifier performs as well as the SVM-Lin classifier using PCP with logarithmic kernel also reenforcing the existence of a discriminating threshold. This intermediate result motivates our choice of feature and choice of threshold classifier for identifying similar shingle content between remixes.

4.2. Estimation of the threshold

To estimate the threshold we chose $N = 10$ and fit the remix distribution of shingle distances with a mixture of two Gaussians. Figure 1 shows the distributions of δ_{remix} shown above over the Madonna songs divided by ground truth into remix and non-remix classes for $N = 10$. The figure shows a fit of the remix distribution with a mixture of two gaussians. The remix distribution can be seen to be composed of true positive (TP) remix shingles and false positive (FP) shingles that are actually drawn from the non-remix distribution. We choose the threshold r_0 using the likelihood ratio test for the two gaussians fit to the remix distribution. Assuming the data are independent and identically distributed (i.i.d.):

$$\frac{P(x_1, x_2, \ldots, x_N|\mu_1, \sigma_1)}{P(x_1, x_2, \ldots, x_N|\mu_2, \sigma_2)} = \frac{e^{-\frac{|\sum_k x_k - N\mu_1|^2}{\sigma_1^2}}}{e^{-\frac{|\sum_k x_k - N\mu_2|^2}{\sigma_2^2}}} >= \lambda_0$$

in the GMM example in Figure 1 $\lambda_0 = 1$. Note that the variance of order distributions go as $\frac{\sigma_k^2}{N}$. So increasing N reduces the variance and, therefore, also the degree of overlap in the two distributions. The reduced overlap results in increased classifier performance because we reduce the sampling error with more samples from each song pair. However, the proportion of remix shingles between remixed songs is small compared to the proportion of non-remixed shingles. This is because remix songs are often based around a small fragment, or a number of such fragments. If we choose N too large we exhaust the supply of available remixed fragments and we start to draw from the class of non-remixed shingles within the remixed song.

4.3. Locality Sensitive Hashing

Using the threshold r_0 as a search radius we used the E^2LSH algorithm to hash the shingles and retrieve only those below threshold [6][8][7]. Use of the LSH algorithm removes explicit computation of the between-shingle distances as it is implicitly estimated by LSH hash. Those shingles whose distances are within r_0 of eachother will fall in the same hash bucket with probability $1 - \epsilon$. The trade-off in the algorithm is between the speed and accuracy and is controlled by the $1 - \epsilon$ term.

We modify the steps in the remix recognition algorithm to use LSH in the following way. Instead of computing $\delta_{remix}(\mathbf{A}, \mathbf{B})$ we hash the shingles (x^i, y^j) by projecting against a fixed random basis $\mathbf{V}$ $N(0, 1)$. Each projection hashes the shingle to the real line which is chopped into equal-length segments. The size of the hash buckets is determined by the radius parameter r_0. Collisions are chained with probability $1 - \epsilon$ of shingles within a distance r_0 of eachother falling in the same bucket for each vector $v \in \mathbf{V}$. Because this hash table is so large, consisting of the entire range of distances within the dataset, a secondary hash table is constructed that chains the sparsely populated primary hashes.

To efficiently retrieve all the shingles within the database that fall within r_0 of eachother, we simply note which hash buckets contain collisions. The proportion of the database that caused collisions is very small compared to the size of the database, so we collect the indexes of those points and perform the Euclidean distance calculation.

This exact distance calculation is thresholded using r_0 and the points remaining are associated with their corresponding songs. Now, for each song, if there are close to $N = 10$ shingle indexes remaining we say that there is a remix in the database, and we simply locate the other songs by inspecting the indexes of the collided shingles.

The speedup factor for LSH over the pair-wise shingle distance computation is between a factor of 10 and 100 using the current implementation this algorithm.

5. RESULTS

The results of our remix retrieval experiment are shown in Figure 2. The dashed line on the figure shows performance for the exact algo-

rithm with distance threshold set to $r_0 = 0.2$. The remaining lines show the performance of the approximate (LSH) algorithm using thresholds $r_0 = 0.04, 0.1, 0.14, 0.16, 0.18, 0.2$ to show the affect on performance as we approach the optimal decision boundary. The LSH algorithm performs extremely accurately for a speedup factor of between one and two orders of magnitude when the threshold is set near the optimal. This highlights the importance of using a threshold estimation method as outlined above on some example data before choosing the LSH hash bucket sizes.

In our experiments performance was impacted with about 10% increased error when we introduce a large amount of material from another artist. In this case the interference is 1712 Miles Davis tracks that are highly unlikely to overlap with the TP remix shingle pairs for Madonna songs. This decrease in performance is caused by similar features, so we conclude that the log-PCP feature might be improved.

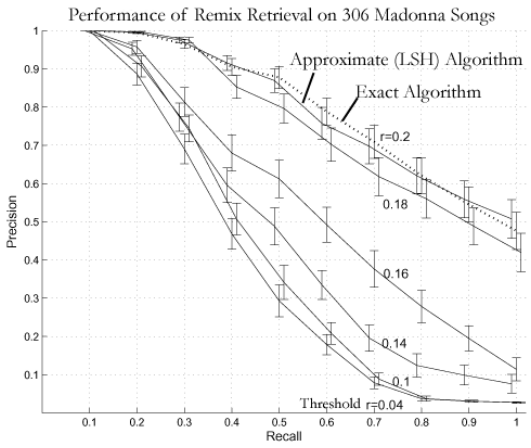

Fig. 2. Results of remix retrieval on 306 madonna songs using exact and approximate (LSH) algorithms. (Figure reproduced from [4]

6. CONCLUSIONS

We have shown that a a similarity measure between songs using only small parts of the song can be used effectively to identify songs that are related as remixes. We used the distributions of inter-song shingle distances and showed that separation of the two distributions can be achieved by choosing a suitable threshold on the distances and that this threshold could be estimated from examples using a mixture of Gaussians. With a suitable kernel space we showed that a threshold classifier can be used for robust audio matching for mid-specificity problems such as remix recognition.

We gave the details of two algorithms, one based on exact computation of between-song shingle distances and the other based on approximate evaluation using LSH. The results of experiments on a collection of 306 within-artist examples showed that the approximate algorithm performed very well with respect to the exact algorithm for a speedup factor of between 10 and 100.

In future work we hope to use more robust features to see if the degree of separation between distributions can be improved, this will lead to increased performance and a greater degree of generalisation of our results.

7. REFERENCES

[1] Mark A. Bartsch and Gregory H. Wakefield. To Catch a Chorus: Using Chroma-Based Representations for Audio Thumbnailing. in *Proc. WASPAA*, 2001.

[2] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. In Proceedings of WWW6 '97, pages 391–404, Elsevier Science, April 1997.

[3] M. Casey and M. Slaney. The improtance of sequences in music similarity. *in Proc. ICASSP*, 2006.
in Proc. ISMIR, 2006.

[4] M. Casey and M. Slaney. Song Intersection by Approximate Nearest Neighbor Search. *in Proc. ISMIR*, 2006.

[5] D. Ellis, B. Whitman, A. Berenzweig, S. Lawrence. The Quest for Ground Truth in Musical Artist Similarity. *Proc. ISMIR-02*, pp. 170–177, Paris, October 2002.

[6] M. Datar, P. Indyk, N. Immorlica and V. Mirrokni. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions, In *Proceedings of the Symposium on Computational Geometry*, 2004

[7] Locality-sensitive hashing using stable distributions, in *Nearest Neighbor Methods in Learning and Vision: Theory and Practice* , by T. Darrell and P. Indyk and G. Shakhnarovich (eds.), MIT Press, to appear.

[8] Aristides Gionis, Piotr Indyk and Rajeev Motwani. Similarity Search in High Dimensions via Hashing. *The VLDB Journal,* pp. 518–529, 1999.

[9] J. Herre, E. Allamanche, O. Hellmuth, T. Kastner. Robust identification/fingerprinting of audio signals using spectral flatness features. *Journal of the Acoustical Society of America*, Volume 111, Issue 5, pp. 2417–2417, 2002.

[10] Meinard Muller, Frank Kurth and Michael Clausen. Audio Matching via Chroma-Based Statistical Features. *In Proc. ISMIR*, London, Sept. 2005

[11] S. Pauws. Musical Key Extraction from Audio. *In Proc.ISMIR*, Barcelona, 2004.

[12] Elias Pampalk, Arthur Flexer, Gerhard Widmer. Improvements of Audio-Based Music Similarity and Genre Classificaton. *in Proc. ISMIR*, pp. 628-633, 2005.

[13] Douglas A. Reynolds. Speaker identification and verification using Gaussian mimxture speaker models. *Speech Commun.,* 17 (1–2):91–108, 1995.

[14] Matthew Miller, Manuel Rodriguez and Ingemar Cox. Audio Fingerprinting: Nearest Neighbour Search in High Dimensional Binary Spaces. Multimedia Signal Processing, 2002 IEEE Workshop on , 2002

[15] Jaap Haitsma, Ton Kalker. A Highly Robust Audio Fingerprinting System, Proc. *ISMIR*, Paris, 2002.

[16] P. Cano and E. Batlle and T. Kalker and J. Haitsma, A review of algorithms for audio fingerprinting. In International Workshop on Multimedia Signal Processing, US Virgin Islands, December 2002.

[17] Avery Li-Chun Wang, Julius O. Smith, III. System and methods for recognizing sound and music signals in high noise and distortion. United States Patent 6990453, 2006

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