Song Intersection by Approximate Nearest Neighbour Search

Michael Casey
Goldsmiths College
University of London
m.casey@gold.ac.uk

Malcolm Slaney
Yahoo! Research Inc.
Sunnyvale, CA
malcolm@ieee.org

Abstract

We present new methods for computing inter-song similarities using intersections between multiple audio pieces. The intersection contains portions that are similar, when one song is a derivative work of the other for example, in two different musical recordings. To scale our search to large song databases we have developed an algorithm based on locality-sensitive hashing (LSH) of sequences audio features called audio shingles. LSH provides an efficient means to identify approximate nearest neighbors in a high-dimensional feature space. We combine these nearest neighbor estimates, each a match from a very large database of audio to a small portion of the query song, to form a measure of the approximate similarity. We demonstrate the utility of our methods on a derivative works retrieval experiment using both exact and approximate (LSH) methods. The results show that LSH is at least an order of magnitude faster than the exact nearest neighbour method and that performance is not impacted by the approximate method.

Keywords: Music similarity, audio shingling, nearest neighbours, high dimensions

1. Introduction

This paper explores a means to compute the intersection between multiple audio pieces. We want to find the portions of a piece that are similar, perhaps because one is a derivative of the other, in two different musical recordings.

We are interested in approximate methods, where the approximation can be as good as necessary, because we now have access to million-song databases. Exact algorithms, based on brute-force audio similarity measures are prohibitively expensive. The key to our work is a new type of algorithm called locality-sensitive hashing (LSH). LSH provides a very efficient means to identify (approximate) nearest- neighbors in a high-dimensional feature space. We combine these nearest neighbor estimates, each a match from a very large database of audio to a small portion of the query song, to form a measure of the approximate similarity of two songs.

In our application two songs are similar if one portion is approximately contained in another song.

There are two practical needs driving this work. First, user's often have a playlist and want to move it to a new system. We want to be able to offer the user a close match if we don't have the exact song title. Second, and perhaps more importantly, commercial success in these days of large music catalogs is based on finding the music that people want to listen to. This is driven by a recommendation system, which depends on user's rating data. A recommendation system will perform much better if we can propagate a user's rating to other recordings of the same song. The problem is analogous to near-duplicate elimination in text document [3] and image archives [9] and has many interesting analogues in the audio domain.

1.1. Audio Similarity

It is difficult to define similarity and even more difficult to score results. For the purposes of this work, we say two songs are similar if one is a derivative of another. Derivative works do not simply contain "samples" of the signal of an original work, but instead use part of a vocal track and remix it with new percussion and bass tracks. Furthermore, only a small part of the source work is used for the derivative work, so any method used to identify derivative works must be able to identify a small amount of material in a completely new context; this is called partial containment. Hence identification of derivative works requires determining partial containment of approximately matching audio. For purposes of evaluation, our ground truth is identified as songs with overalpping title stems which is discussed in Section 3.4. Table 1 illustrates the related titles for Madonna's *Nothing Fails*.

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 characteristics of the signal that lend the song its distinctive character, and are not harmed by difficult communications channels (i.e. a noisy bar and a cell phone). Often 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 a exact match anywhere and we are interested in ranking the songs that are similar to each other.

Table 1. Derivative works of the Madonna title *Nothing Fails* in a commercial database.

Duration	Title
4m49s	Nothing Fails
3m55s	Nothing Fails (Nevins Mix)
7m27s	Nothing Fails (Jackie's In Love In The Club Mix)
7m48s	Nothing Fails (Nevins Global Dub)
7m32s	Nothing Fails (Tracy Young's Underground Mix)
6m49s	Nothing Fails (Nevins Big Room Rock Mix)
8m28s	Nothing Fails (Peter Rauhofer's Classic House Mix)
3m48s	Nothing Fails (Radio Edit)
4m0s	Nothing Fails (Radio Remix)

1.2. Locality Sensitive Hashing

Our audio work is based on an important new web algorithm known as shingles and a randomized algorithm known as locality-sensitive hashing (LSH) [3] [20]. 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.

LSH, instead, does exactly the opposite; two patterns that are close together are hashed to locations that are close together. Each hash produces an approximate result since there is always a chance that two nearby points will end up in two different hash buckets. Thus, we gain arbitrarily-high precision by performing multiple LSH mappings, each from a different random direction, and noting which database frames appear multiple times in the same hash bucket as our query. Each hash can be as simple as a random projection of the original high-dimensional data onto a subspace of the original dimensions.

1.3. Contributions

This paper discusses our approach to song-similarity using approximate matches. Our earlier work [19] showed that matched filters, and thus Euclidean distance in feature space, are an effective way to measure song similarity. Another work [20] introduced the idea of audio shingles and described how we can use them to effectively search a large database of songs. This work explores the idea of approximate matching by finding nearest-neighbor matches. Each nearest-neighbor match is weak evidence that the two songs share a common musical motif or passage. By combining these simple and fast distance measures, we can effectively compute the intersection and similarity between nearby songs.

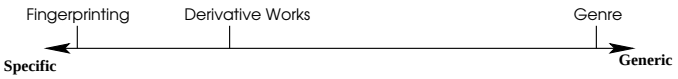

Figure 1. Specificity of derivative works identification. The most specific queries are on the left of the figure and the most generic on the right. Derivative works identification, as described in this work, falls in between.

2. Previous Work

To date, a range of feature-based techniques have been proposed for describing and finding musical matches from a collection of audio. Figure 1 shows the range of options. Fingerprinting [8] finds the most salient portions of the musical signal and uses detailed models of the signal to look for exact matches. At the other end of the specificity scale, genre-recognition [18], global song similarity [11], artist recognition [4], musical key identification [12], 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.

Our application requires algorithms that are robust to differences in the lyrics, instrumentation, tempo, rhythm, chord voicing and so forth, so we explore features that are invariant to various combinations of these [1][2].

Inherent in our problem is the need to measure distances in a perceptually relevant fashion and quickly find similar matches without an exhaustive search through the entire database. Existing Gaussian Mixture Model methods for computing audio similarity do not scale to large databases of millions of songs due to the computation required in pair-wise comparison of models using a suitable distance function such as Earth Movers Distance (EMD) [10]. Likewise, high-dimensional feature representations are susceptible to the curse of dimensionality that leads to inefficient (linear time) search algorithms. We will fail in large databases if we need to look at every signal to decide which are closest.

Recent work shows that audio features are efficiently retrieved using locality-specific hashes (LSH) which have sub-linear time complexity in the size of the database. This is a key requirement for audio retrieval systems to scale to searching in catalogues consisting of many millions of entries. These methods have already found applicability in image-retrieval problems [3]. LSH solves approximate nearest neighbour retrieval in high dimensions by eliminating the curse of dimensionality [7][5][6].

The features used to describe the signal are critical. LSH is only appropriate when the signal can be represented by a point in a fixed-dimensional metric space with a simple norm (such as L2). For example, methods that compare sequences of different lengths, such as dynamic time warping, are not easy to implement using LSH. Other models fail this metric because the distance measure is not simple. This includes Gaussian mixture models, hidden Markov models, and distance metrics based on dynamic time warping. Earlier work [19] shows that LSH is theoretically able to solve the audio sequence search problem accurately, and in sublinear time, when the similarity measure is a convolution of sequences of audio features which provides an L2 norm.

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 [19]. 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. Song Intersection

We now describe the steps for retrieving songs from a database with content that partially intersects with a query song.

3.1. Feature Extraction

Uncompressed 44.1kHz PCM audio signals are first segmented into length 372ms frames overlapped with a hop size of 100ms. A hop size of 100ms was chosen to trade off temporal acuity against time and space complexity for the search. Previous work indicates that, even at the signal level, the spectrum is sufficiently correlated in time that small shifts in frame alignment lead to small changes in feature values [14].

We derive two features using constant-Q spectrum transform. Log-frequency cepstral coefficients (LFCC) are extracted using a 16th-octave filterbank and chromagram features are extracted with a 12th-octave filterbank. In both cases the filterbank extended from 62.5Hz to 8kHz. The filterbank was normalized such that the sum of the logarithmic band powers equalled the total power.

To extract the LFCC coefficients we used a discrete cosine transform (DCT) retaining the first 20 coefficients. To extract CHROM features we summed the energy in logarithmic bands at octave multiples of 12 reference pitch classes corresponding to the set $\{C, C\#, D, ..., A\#, B\}$.

3.2. Audio Shingles

We create a shingle by concatenating 30 frames of 12-dimensional chromagram features into a single 360 dimensional vector. Much like the original work on shingles [3], we advance a pointer by one frame time, 100ms, and then calculate a new shingle. Unlike text shingles, which are word histograms, our shingles are time-varying vectors. To make the shingles invariant to energy level we normalized the shingle vectors to unit length.

We use the vector dot product to compute the similarity between a pair of shingles. This can be computed efficiently for audio shingles using convolution which is proportional to the L2 (Euclidean) distance between them [19].

3.3. Similarity Measurement

For this paper, we use a new version of LSH based on p-stable distributions [5]. With a p-stable distribution, vector sums of random variables from a p-stable distribution still have the original probability distribution. We form a number of dot products between the database entries and random variables from the p-stable distribution. Each of these dot products forms a projection onto the real axis, and helps us estimate the true distance.

We can then divide up the real axis into buckets and form a hash that is locality specific points that are close together in the input space will be close together after projection onto the real axis.

Our similarity measurement is performed in two stages. We first search for the N audio shingles in our database that are closest to each query song. Given these nearest-neighbor matches, found using brute force or LSH, we look at the top-N shingle matches for a pair of songs and compute the similarity by averaging these smallest-N distance scores to find the similarity between the two songs. Thus a short fragment that is contained in another song will cause the similarity measure to be small and indicate a close match.

Our use of LSH is different from its use when finding nearest- neighbor matches. Normally, the points found by LSH are checked with an exact distance calculation to ensure that they are true nearest neighbors, and not the result of a hash conflict. In our case, we skip this filter. We are only interested in the average distance, so we use all the close points returned by LSH to form our estimate. In essence we are using LSH to estimate the matched filter between two shingles.

In addition, our data is more randomly distributed than in normal uses of LSH. Often the nearest matches when finding text duplicates are truly close to the query, perhaps differing in a few discrete directions. In our case, we see that the data is randomly distributed in our 1200 dimensional space. We expect a Gaussian noise models the distance between an audio shingle and it's closest neighbor.

Figure 2 shows a plot of the inter-point distances between chromagram shingles and random Gaussian-distributed vectors. The distance histograms, after scaling, are nearly identical. This equidistance behavior, and the exponential growth of the distance histogram means that it is hard to pick the right radius for the nearest-neighbor calculation for this application of LSH.

3.4. Data Set

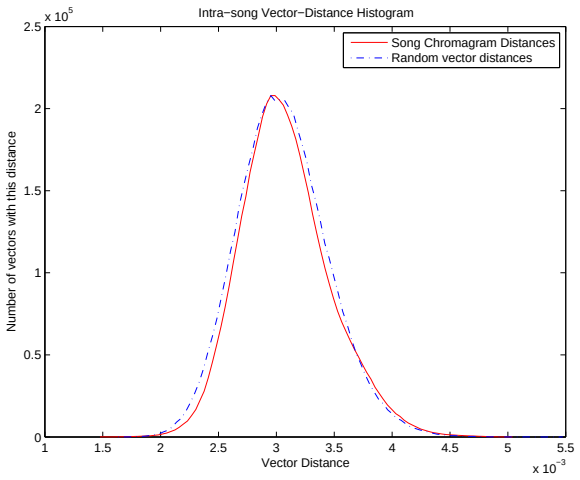

Figure 2. Intra-song Vector Distance Histograms. Comparing the distance between 100-frame chromagram shingles and (solid line) and two Gaussian random vectors (dashed line.)

Table 2. Distribution of derivative works in a 2018 song subset of the database.

Artist	Tracks	Stems	Sources	Derivatives
Madonna	306	142	82	164
Miles Davis	1712	540	348	1172

We performed our experiments on the complete recordings of two artists, Madonna and Miles Davis. These two artists were chosen because they both have extensive back catalogs and their music is available electronically. Each recording has a unique 20-digit unique identifier (UID) that is used to locate metadata such as artist, title, album and song length. We obtained exact copies of each commercially distributed recording in a *lossless* format from the Yahoo YMU warehouse (80GBytes of data) and performed our feature extraction directly on the 44.1kHz PCM representation. Our experiment catalogue consists of 306 separate Madonna recordings and 1712 separate Miles Davis recordings. The total duration of audio was 222 hours 26 minutes and 14 seconds.

On inspecting the catalogue, it is immediately apparent that many recordings share all, or part, of their title strings. To stem the titles, we first removed any puncuation, such as quotation marks, and truncated each title up to the first parenthesis if present, else no truncation occured. Any leading or trailing whitespace after these transformations was also removed. For example, all of the titles in the Table 1 were transformed by the stemming to the string *"Nothing Fails."*

Once the titles in the database were stemmed, we gathered statistics on title use within each artists' collection of songs, which are summarized in Table 2.

There were 306 different Madonna recordings in the database with 142 had unique title stems, 82 of which had derivative versions (58%), giving a total of 164 derivative works. Similarly, there were 1712 different Miles Davis recordings, with 540 unique title stems, of these 348 had derivative versions (64%) giving a total of 1172 derivative works.

We used 20 Madonna songs with deriative works as our test set. From the set of songs with the same title stem, a "source" song was selected as being the historically earliest version of the song in the database. The number of relevant matches for the set of 20 such source queries (not including the queries themselves) is 76 songs of the 2018.

4. Results

In this section we describe the details and evaluation of retrieving derivative works by nearest neighbor audio shingles. The similarity measure is a measure of the degree of intersection between the songs in the database. In our experiments, reported here, silence was first removed using an absolute threshold and then low-energy shingles were removed if they were below the mean energy for the song.

4.1. LSH Experiment

In the first experiment we extracted 30-frame shingles of 12-dimensional CHROM features with a hop size of one frame $(0.1s)$. This yielded a 360 dimension vector every $0.1s$. For each song in the detabase we found 10 nearest neighbors for pairs of query and database song shingles. The average of the 10 nearest distances for each song was taken to be the measure of intersection between the query song and the database song. Sorting the distances yielded a ranked list of database songs for the given query song. This operation was performed for all of 20 query songs.

We used textual title stem matches to identify ground truth derivative works, see Table 1. We recorded true positives and false positives at each level of recall standardized into 10th-percentiles. Confidence intervals were estimated using the standard deviation of the precisions at each 10th-percentile interval and dividing by the square root of the number of query songs.

Figure 3 shows the results of retrieval of song intersections using the LSH algorithm varying the search radius for nearest neighbors. The dotted line shows the result for exact nearest neighbour retrieval. The remaining lines show the performance of LSH retrieval using raidii $0.04 \leq r \leq 0.2$. At 70% recall the algorithm achieves 70% precision for $r = 0.2$, dropping to 51% precision for 100% recall. We note the the LSH approximation did not introduce any significant error in the derivative works retrieval task for a radius of $r = 0.2$, but for lower radii the precision decreased significantly when compared with the exact algorithm's performance. This illustrates the need to choose the correct search radius for the task.

4.2. Feature Variation

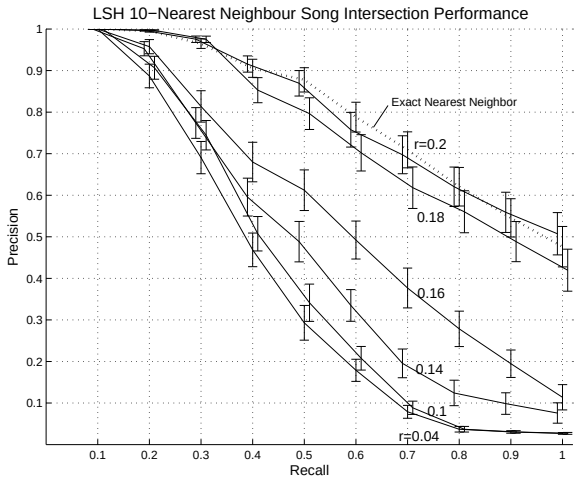

Figure 3. Performance of LSH nearest neighbor retrieval for estimating song intersections (derivative works). Audio shingles were 30 frames and the search radius varied between $r = 0.04$ and $r = 0.2$. The dotted line shows the exact nearest neighbor result for search radius $r = 0.2$.

In the next experiment we varied the features to test which feature combination performed best in our task. We also increased the shingle size to 100 frames, thus yielding 1200 dimensional vectors for the CHROM features and 2000 dimensional vectors for LFCC. For comparison to the Chromagram extraction method of Bartsch [1] we tried a variation on CHROM features with a cutoff frequency of 2kHz instead of the 8kHz cutoff used for the rest.

We also tried a joint feature space consisting of both CHROM and LFCC features. Here, the song similarity measure is a weighted average of the chromagram and lfcc features. Results are shown for 0.9*CHROM + 0.1*LFCC; an empirically determined mixture of the distances.

Figure 4 shows the results for the feature variation experiment. The worst performing features were CHROM, with 2kHz cutoff, and LFCC both returning a precision of 65% at 70% recall. For the CHROM features, with 8kHz cutoff, the performance is much better and almost identical to that shown in Figure 3. From this we conclude that increasing the shingle size from $3s$ to $10s$ had no significant impact on the results. We also note that CHROM features performed significantly better than LFCC but significantly worse than the joint CHROM+LFCC feature space. The improvement might be accounted for by the false negative rate being reduced but not the false positive rate using the joint feature space. CHROM and LFCC encode qualitatively different aspects of the songs– CHROM features encode the harmony and pitch content, and LFCC features encode the timbral content. However, we were surprised that the joint features performed better and we are investigating the reason.

To see how retrieval performance scaled, we compared performance using the 306-song subset with the 2018-song

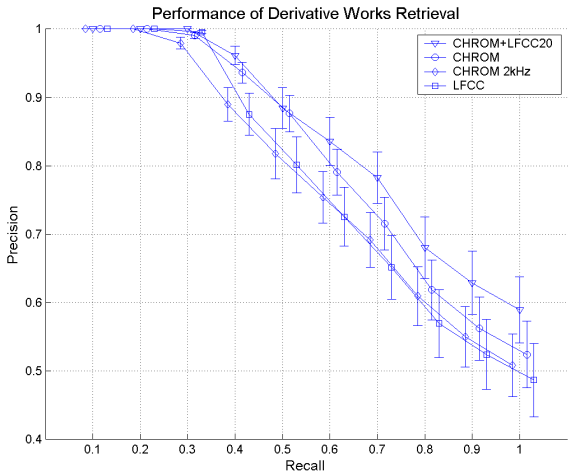

Figure 4. Performance of exact audio shingle retrieval for different features and a feature combination. Here the audio shingles are $10s$ in length.

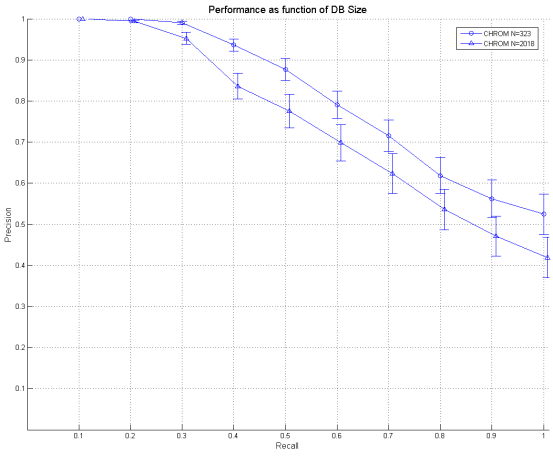

Figure 5. Comparative performance for database of 306 songs and 2018 songs.

database, Figure 5. There was a 10% drop in precision for the larger database at recall rates greater than 40%. The precision was 63% at a recall of 70% for the larger data set.

4.3. Time complexity of Exact vs. LSH Algorithms

The time complexity of the exact approach is $|Q| \times |\bar{S}| \times d \times w \times \mathbf{O}(\mathbf{N})$ in the number of songs in the database, N, the number of query shingles, $|Q|$, the average number of shingles per song,$|\bar{S}|$, the feature dimensionality, d, and the length of the shingles, w. For the twenty queries matched against a 306-song database using chromagram features, this results in approximately $306 \times 20 \times 3000 \times 3000 \times 12 \times 30 = 19.8 \times 10^{12}$ multiply-accumulate operations. Computation for the exact algorithm approximately 7 hours using a 3GHz PPC processor. For the 2018-song database, computation time increased to approx. 150 hours for the exact algorithm.

The LSH algorithm's performance depends on the size of the hash buckets and the degree of approximation used in the nearest neighbour search. For our chosen parameters, the LSH program completed the task in approximately 1 hour for the 306-song dataset. However, more than half of the time was spent self-tuning the parameters and building the hash tables, both of these are operations that only need to be performed once for each radius. We observed that the retrieval part of the execution cycle took less than 30 minutes, therefore running at least 14 times faster than exact nearest neighbour retrieval.

5. Conclusions

We introduced audio shingles for measuring musical similarity. We employed them as a means for identifying musical works that approximately match, or intersect, over a part of their content. We described the features used and the similarity methods employed as well as two algorithms for implementing the similarity-based retrieval using nearest neighbour search.

The exact method gives good results, but it takes a long time to compute the answer, scaling linearly in the size of the database. The approximate algorithm based on LSH is greater than an order of magnitude faster and yields accurate results on our chosen task.

Our conclusion is that hashing for low-level audio features is accurate and speeds up complex retrieval tasks significantly.

6. Acknowledgements

The authors gratefully acknowledge Alex Andoni and Piotr Indyk at MIT for providing the LSH code (E2LSH).

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] J. P. Bello, and J. A. Pickens. A Robust Mid-level Representation for Harmonic Content in Music Signals. *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR-05)*, London, UK. September 2005.

[3] 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.

[4] 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.

[5] 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

[6] 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.

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

[8] 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.

[9] Yan Ke, Rahul Sukthankar, Larry Huston. An efficient near-duplicate and sub-image retrieval system. *ACM Multimedia*, 2004: 869–876.

[10] B. Logan and S. Chu. Music Summarization Using Key Phrases. *In Proc.IEEE ICASSP*, Turkey, 2000.

[11] E. Pampalk, A. Flexer and G. Widmer. Improvements of Audio Based Music Similarity and Genre Classification. *Proc. of ISMIR*, London, Sept. 2005.

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

[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.

[19] Michael Casey and Malcolm Slaney. The Importance of Sequences for Music Similarity. *IEEE ICASSP processing.*, Toulouse, May 2006.

[20] Michael Casey and Malcolm Slaney. Audio Shingling for Measuring Musical Similarity. Submitted to *ACM Multimedia*, April, 2006.