Analysis of Minimum Distances in High-Dimensional Musical Spaces

Michael Casey, Christophe Rhodes and Malcolm Slaney

Abstract

We propose an automatic method for measuring music similarity using audio features so we can enhance the current generation of taxonomy-based music search engines and recommender systems . Efficiency is important in an Internet-connected world, where users have access to millions of tracks. Brute-force algorithms for searching through this content are not practical.

Many previous approaches to track similarity require pair-wise processing between all audio features in a database and therefore are generally not practical for large collections. Our features are time-ordered overlapping fixed-length subsequences of equal-temperament pitch-class profiles and log-frequency cepstral coefficients; the technique is analogous to the technique of shingling used for text retrieval. We use locality sensitive hashing to implement approximate matching for our high-dimensional audio shingles. This approach retrieves near neighbors within a specified distance of the query rather than retrieving only the *nearest* neighbors; the degree of approximation, ϵ, is a parameter. LSH achieves sub linear query time performance with respect to the number of tracks in a collection but requires an accurate threshold on retrieval distance for efficient performance. In this paper, we present a new method for estimating the optimal search radius for LSH retrieval tasks by modeling the between-shingle distance distributions for non-similar audio shingles. We derive an estimator for a minimum distance for two shingles to be considered drawn from different tracks. therefore, are considered to be drawn from similar tracks.

We evaluate our proposed methods on three contrasting music similarity tasks: retrieval of mis-attributed recordings (Apocrypha), retrieval of the same work by performed by different artists (Opus) and retrieval of edited and sampled versions of a query track by remix artists (Remixes). Our results achieve near-perfect performance in the first two tasks and 80% precision at 70% recall in the third task.

Index Terms

Music similarity, distance distributions, locality sensitive hashing, audio shingles, matched filter distance

I. Introduction

We are interested in efficient algorithms for finding similar musical pieces based on their acoustic signal. Systems for song retrieval are most easily characterized by the specificity of the query. There are four types of queries. The first three types of audio queries have the common property that they match sequences of audio features, therefore they use temporal features. While the fourth and least specific can be answered using statistical averages.

This research supported by Engineering and Physical Sciences Research Council Grants EPSRC GR/S84750/01 and EPSRC EP/E02274X/1.

1) Fingerprinting: Audio fingerprints are the basis of the most specific type of query. We wish to uniquely identify a recording in the presence of distorting communications channel. The song might be corrupted by loud background noise and very lossy compression. Fingerprinting is valuable to consumers because they would like more information about a song they are listening to.
2) Remixes:

   A remix is a version of a song that contains audio content from the original song, but it is remixed to give a different feel. It is usually by the same artist. Here, a song is similar to the query since a small part of the song closely matches a part of the query.
3) Cover songs:

   In contrast, cover-song retrieval seeks to find all versions of the same song title performed by different artists. Here we expect none of the query's audio to be contained within the retrieved songs; i.e. it is the *musical* content that is similar.
4) Genre: The least specific type of audio matching disregards temporal information altogether and attempts to match global spectral properties of songs using statistical models.

In fingerprinting we look for exact matches, but the second and third types of queries are approximate matches and thus we need to find near neighbors in some feature space. The proportion of snippets that closely match determines whether the song is a version of the query song. This work addresses the problem of finding near neighbors for acoustic signals in a high-dimensional space.

Potential applications include near duplicate elimination for improving the ranks of relevant search results, sampling source identification, high-level music structure extraction (identifying repeats), remixed song retrieval, cover version retrieval and linking of recommendation data between closely related songs. We wish to describe an implementation of such applications in today's million-song music databases.

We evaluate our proposed methods on three contrasting music similarity tasks: retrieval of mis-attributed recordings (Apocrypha), retrieval of the same work by performed by different artists (Opus) and retrieval of edited and sampled versions of a query track by remix artists (Remixes).

Our contributions are: 1) a matched-filter distance metric for sub-linear query time retrieval, 2) a new efficient algorithm for retrieving songs with related content, 3) analysis of the between-song distance distributions, and 4) experimental evidence that the song-similarity problem, as stated, is well-posed.

[CHECK] The structure of the paper is as follows: we review different approaches to audio similarity in Section II, we develop a model for retrieving tracks by audio shingles in Section III, we then given details of related-track recognition algorithms in Section III-C and give an analysis of the algorithms in Section IV. Finally, we present the results of various related-track retrieval experiments on 2000-track data sets in Section V and we conclude with the implications of our results for efficient audio matching applications in Section VI.

II. Related Work

There is a wide spectrum of previous work covering a range of music-similarity tasks: from very specific identification or fingerprinting work [21][34] to genre recognition [33][29]. Our work falls in the middle. We want to find songs that are similar, but not exactly like another song. Our tasks needs both musically-relevant features—we don't expect the very specific features used in fingerprinting to work—and a robust matching criteria because we expect that remixes will have segments rearranged and new material inserted.

All fingerprinting and musical similarity systems combine feature analysis with a matching stage. There is always a tradeoff between the complexity of the feature and the amount of effort needed to discriminate good from bad matches in the song database. Roughly speaking, each system transforms the audio into a musically-relevant feature vector, then looks for that vector in the music database. Both of these steps are hard: the wrong feature magnifies meaningless differences between the query and the database entries, and brute-force techniques are no longer viable due to the size of today's music libraries. For context, we will talk about each aspect in turn, as they are used in other music-similarity systems.

A. Fingerprinting

Querying for similar songs is different from the classic work that has been done on audio fingerprinting [26][23][7][34]. These systems assume that some portion of the audio is an exact match so conventional computer hashes can reduce the search space. We do not expect to see exact matches in remixes.

Our work also differs from traditional fingerprinting or matching work because we choose a different division of labor between the feature and the matching stages. Many systems use relatively short features, perhaps 20ms long, and then use a more complicated matching scheme to find a sequence of matches. Our approach, on the other hand, uses a long acoustic feature vector, 4 seconds, and a single, conceptually simple lookup algorithm to find the near neighbors. We implement the near-neighbor lookup using a randomized algorithm known as locality-sensitive hashing (LSH), which we believe is of growing importance as our databases grow in size. The long feature vector and the relatively simple lookup algorithm makes analysis of our performance feasible.

B. Features for Similarity

Features used in music-similarity tasks range from simple to specific. At the simplest level, features provide a relatively straightforward representation of the musical signal. Examples of such systems include Kurth [28] and our own work on the Opus and Remix tasks. Both these systems use a musical representation based on the 12 notes of an octave. With the exception of smoothing or downsampling of the signal, the resulting feature vector contains the essential aspects of the musical signal. One could reconstruct a musically expressive audio signal from these features alone.

The same is not true of the classic fingerprint systems. In fingerprinting, users want to find the name of a recording given a sample of the audio. The secret sauce that makes fingerprinting work is the use of a robust feature of the signal that is not harmed by difficult communications channels (i.e. a noisy bar or a cell phone). We call these representations cartoons. Typical features include the peaks of a spectral-temporal representation [34], and the relative energy levels in adjacent spectral bands [22], and the spectral-temporal wavelets with maximum energy [2]. These aspects of the signal are not (often) harmed by noisy environments and are good for finding exact matches.

More recent fingerprinting or identifications systems have learned their feature representation. Burges [1] used an optimization procedure to design acoustic features that are robust to nine low-level distortions: compression, nonlinear amplitude/bass distortion, various notch filters or frequency boosts, pitch distortion and companding. The OPCA projection vectors are learned by training the system with noisy and clean versions of the same signal, and from all these vectors they estimate suitable overall covariance and correlation matrices. Similarly, other researchers have learned features from a timbral representation known as MFCC [35] or optimized their features based on spectrograms as part of an overall identification system [35] [3]. The adaptive nature of these features makes it hard to analyze their performance.

C. Querying for Similarity

Given a feature set, there are many ways to find the database entries that are closest to the query. The brute force solution is easy to describe: look at each database entry, compute the pair-wise distance, and then return the pair with the minimum distance. In practice, query approaches differ based on whether the algorithm finds exact matches, or whether they allow approximations, and whether they are concerned with large databases and need efficient solutions.

Brute force techniques, searching every database entry, are possible for small databases. Mueller's audio matching system [28] is successful with this approach, as well as Burges' [1]. But this won't work in databases with millions of songs and billions of audio snippets.

Conventional hashing techniques work well if one is looking for an exact match. A conventional hash converts a set of data, whether it's a string or a vector, into an index into a table. If the desired entry is present in the table it is returned, otherwise a miss is recorded. This converts an O(N) problem, where N is the size of the database, into an O(1) problem in time, but O(N) in memory. This type of exact hatch is the basis of the classic fingerprinting work [22] [34] and it is even useful when the learned features are especially robust [25]. Ke extends their low-dimensional queries to include nearby variations based on small, deterministic changes to the query vector.

When using larger databases, and for systems that are interested in similarity as opposed to identification, a more sophisticated approach is necessary. Locality-senstive hashing (LSH) was originally designed to find nearby web pages [6], but has since been extended to music [38], image retrieval and other tasks [14]. At its core, LSH is a randomized algorithm that finds near neighbors in high-dimensional spaces [32]. In practice it is an implementation technique—it approximates the best answer in sub-linear time. It is a practical solution to the problems described in this work, and others [2]. Baluja [3] describes an improvement of LSH that replaces the random projections in LSH with "projections" that are learned from loose labels on the data.

Finally, a new approach by Weinstein [35] builds a finite-state automata to recognize a large collection of music.

They train up to 1024 musical "phones" by clustering, and then recognize sequences of these phones using the automata. A collection of 15 thousand songs, after combining related nodes, leads to a finite-state automata with 60 million edges.

Our work in this paper uses simple PCP and LFCC features, and uses LSH to implement the search. Our matching threshold is motivated by E^2LSH's query threshold. But we are essentially assuming brute-force search in this paper. The simple features and the brute-force query allow us to perform the statistical analysis we describe in Section IV.

D. Matching with Locality Sensitive Hashing

Our audio work is based on an important randomized algorithm known as locality-sensitive hashing (LSH) [6]. This is the basis of a popular way to detect duplicate web pages. It allows one to determine if a new web page discovered by a web crawl is already in the database.

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. A hash trades computational effort, searching through a list of keys, for memory. Given a new key, the hash value is used to index into a table. In a well-designed hash table, a value is retrieved with minimal effort, irregardless of the number of entries in the table. This allows the string-matching problem to be greatly sped up because it's rare that two strings will have the same hash. But this speedup is possible only for exact matches.

LSH solves the similarity problem using hashes that preserve the distance between nearby input points. LSH is an example of a randomized algorithm because it uses random projections, each of which reduces a high-dimensional problem to a 1-dimensional hash table, to find with arbitrarily high probability a near neighbor in the database. LSH was first applied to web pages, and then to object recognition and music similarity. In this paper we more formally evaluate the performance of this approach.

Our previous work showed that matched filters, and therefore Euclidean distance, using chromagram and cepstral features performs well for measuring the similarity of passages within songs [8][9]. 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.

E. Generic Queries

At the low end of the specificity scale, genre-recognition [33], global song similarity [29], artist recognition [16], musical key identification [30], and speaker identification [31] use much more general models such as global statistics. The probability density functions are approximated by Gaussian mixture models (GMM). 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.

III. Retrieving Similar Tracks

We seek to find different versions of a track at a given level of specificity. Specificity is defined by the features used and the distance threshold used to accept a song as relevant. For example, relevant tracks may be versions of a track that have been edited or filtered in some way, as in the Apocrypha retrieval task, or versions of a track by different artists as in the Opus retrieval and Remix retrieval tasks.

In the Apocrypha task the goal is to make the features robust to transformations such as filtering and a small amount of time compression/expansion. But the Opus task, for example, requires features that are robust to a greater degree of variation because we don't expect any part of the audio signal of the query track to be present in a relevant track. The Remix task falls somewhere between the two. We expect a small amount of the original audio to be present in the remix, but the surrounding audio context may have changed, new instruments for example, and the original audio may have been treated in some way such as time compression/expansion, pitch shifting or filtering.

A. Audio Features

We chose Log-Frequency Cepstral Coefficients (LFCC) for the high-specificity task (Apocrypha retrieval) and equal-temperament pitch-class profiles (PCP) for Opus retrieval and Remix retrieval; the mid-specificity tasks. Features are extracted from uncompressed audio sourced from the CHARM Mazurkas collection, [11], and the Yahoo! Music database. The audio sources are in 44.1kHz uncompressed PCM stereo format which we convert to mono PCM by extracting the left channel for the Apocrypha task and by mixing the channels for the other tasks. We window the audio using a 8192-point Hamming window with 4410-sample (100ms) hop and compute a 16384-point DFT using an FFT. We choose the FFT window size so that two frequency samples are available from the DFT for the lowest pitch class $C0 = 65.4Hz$. The band edges are chosen at the mid point (quartertone) between chromatic pitch classes with the low edge set to $63.5Hz$ and high edge of $7246.3Hz$, a quartertone above $A7$.

We assign the 8193 DFT magnitudes to pitch-class bands and share samples near the band edges proportionally between the bands, as shown in [4], and we ignore the remaining DFT coefficients. The pitch-class assignments are then folded into a single octave by summing over all octaves for each pitch class, and the logs of sums 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. We take the log of the values and transform the result to cepstral coefficients using a Discrete Cosine Transform (DCT) yielding 20 coefficients per 100ms. Note this feature is a slightly modified form of the widely used MFCC feature, the difference being in the concentration of low frequency components. We choose the features so we can reuse that the FFT parameters between the PCP and LFCC calculations.

B. Audio Shingles

By analogy to the work on finding duplicate web ages [6], audio shingles represent short (several seconds) segments of audio. We concatenate 30 feature vectors at a time into a single high-dimensional vector representing a sequence of audio features. Informed by previous studies [9][28] we use window of $3s$ with a hop of $0.1s$ yielding 10Hz $\times$ 20 dimensions $\times\ 3s = 600$ dimensions for LFCC and 10Hz $\times$ 12 dimensions $\times\ 3s = 360$ dimensions for PCP.

1) Silence Removal and Norming: We did 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 one quarter of the geometric mean of the shingle power in each song.

The remaining shingles are normalized to unit norm so that differences in power between pairs of shingles are not included in the distance metric. This provides invariance to differences in levels between different recordings of the same work, for example, or different treatments of the same track as in the Apocrypha task.

C. Shingle Similarity

Let $\{Q\}$ be the set of audio shingles from a query sound and $\{A^{(k)}\}$ be the shingles for track k (an entire song) drawn from the database. Let $q_i \in \{Q\}$ and $a_j^{(k)} \in \{A^{(k)}\}$ be shingles drawn from tracks $\{Q\}$ and $\{A^{(k)}\}$ respectively. We define the similarity between tracks using the Euclidean distance implemented by a dot product between the two unit-normed shingle sequences. This operation is a multi-dimensional matched filter, which we find by expanding the quadratic and noting that the two vectors are unit norm,

$$d(q_i, a_j^{(k)}) = 2 - 2\sum_{l=1}^{L} q_{il} a_{jl}^{(k)} \tag{1}$$

Then, we rank the similarity between the query track and a database track, k, by the the number of query shingles, $q_i \in \{Q\}$, that are within the distance threshold, x_{min}, of the track's shingles $a_j^{(k)} \in \{A^{(k)}\}$. We do this by first forming a binary indicator function, $I_i(q_i, A^{(k)}) \in [0,1]$:

$$I_i(q_i, A^{(k)}) = \begin{cases} 1 & \text{if } \exists j \text{ s.t. } d(q_i, a_j^{(k)}) \le x_{min}; \\ 0 & \text{otherwise}. \end{cases} \quad \text{for } i \in \{1 \dots |Q|\} \tag{2}$$

Finally, we define the rank for the track pair $(Q, A^{(k)})$ to be:

$$R(Q, A^{(k)}) = \sum_i I_i(q_i, A^{(k)}). \tag{3}$$

We use the indicator function of Equation 2 so that only one relevant shingle in each track is counted per query shingle. The maximum rank, therefore, is $|Q|$ and the minimum is 0. Those tracks from the database with the highest rank, with respect to the query track, are the most similar. The value x_{min} is critical. Values of x_{min} that are too small result in retrieval of too few points and the false negative rate for retrieving near shingle pairs is high. In the extreme, $x_{min} \approx 0$, no points are returned for all tracks, so the ranking is singular. For values of x_{min} that are too large, the false positive rate will be high; admitting pairs of shingles as being relevant where they are not. For example, in the extreme, $x_{min} \approx 2$, all tracks in the database return a rank equal to the size of the query track.

For application to automatic retrieval systems, we propose a method to estimate the optimal value of x_{min} given a sample of the data set so that the proportion of near shingles is higher for relevant tracks than non-relevant. In Section IV we present an analysis of the statistics of distance distributions to derive a set of equations to estimate x_{min} given a sample of non-relevant shingle distances. These methods of computing inter-track similarity have the following desirable properties:

1) We have an analytic method to derive a decision boundary for detecting relevant and non-relevant shingle pairs for a retrieval task. The method uses a sample of non-relevant pairs of shingles as training data.
2) We have a measure of the resemblance between two tracks that is robust to structural changes such as repeats, insertions and deletions. This is necessary for Opus and Remix retrieval where versions of a work exhibit global structural differences but preserve the local sequence information of the work.
3) There exists an efficient means to solve the above distance and ranking expressions. In naive implementations, we evaluate the distances between the query shingles and all database entries and the resulting time complexity is $\mathbf{O}(dN)$ with respect to the total number of shingles N in the database and shingle dimensionality d. Locality Sensitive Hashing retrieves only those database shingles that are within x_{min} of the query shingles, thus admitting evaluation of Equations 2 and 3 with sub-linear time complexity $\mathbf{O}(dN^{1/c^2+o(1)})$ for an approximation factor $c = 1 + \epsilon > 1$ [32].

D. Locality Sensitive Hashing

LSH is an algorithm that retrieves only those shingles that are within a distance x_{min} of the query without exhaustive pair-wise distance computation [15][19][13]. LSH finds these near neighbors in a sub-linear time—a brute force algorithm is linear in the number of shingles in the database. Those shingles whose distances are within x_{min} of each other fall in the same hash bucket with some predetermined probability $1 - \alpha$.

In an ordinary hash, the goal is to efficiently determine exact matches from a target to a database. Here, any approximation in the match that we require must be designed into the features. In LSH, by contrast, we control the degree of approximation in the hash function itself.

There is a trade-off in the algorithm between its speed and accuracy. For approximation factor $c = 1 + \epsilon$ the time complexity of LSH is $\mathbf{O}(N^{1/c^2+o(1)})$ [32] **[Check to make sure this is equation is there!]**. For approximation factor greater than 1, the complexity is therefore sub linear in the size of the database, N. For our applications, we don't mind if there is a small amount of error in the hashing of similar shingles because we expect the rank of similar shingles to be high for similar tracks. We can set c relatively high without affecting the overall accuracy of the rank ordering of music information retrieval tasks.

1) LSH Insertion: Here, we modify the steps in our retrieval algorithm, from Section III-C, to use LSH. First, instead of directly computing the Euclidean distance using Equation 1, we use LSH to put the database shingles $A^{(k)}$ into a series of hash tables, $H^j, j \in \{1 \dots L\}$. Each hash table consists of N slots corresponding to the size of the database. We use $L \approx 100$ hash tables because the hashing is approximate (consisting of random projections) and the probability of achieving a collision is relatively small even for near shingles. Similarity is determined by a query shingle colliding with a database shingle in *any* of the hash tables H^j.

For a given shingle, $\mathbf{a} \in R^{d \times 1}$, we obtain the hash slot, into which we record the track index from which the shingle was drawn, via a hash function $h_{\mathbf{V},\mathbf{b}}(\mathbf{a})$, with $\mathbf{V} \in R^{m \times d} \sim N(0,1)$ and $\mathbf{b} \sim U(0,w)$:

$$h_{\mathbf{V},\mathbf{b}}(\mathbf{a}) = \lfloor \frac{\mathbf{V}.\mathbf{a} + \mathbf{b}}{w} \rfloor \tag{4}$$

Each random vector in **V** projects the shingle to the real line which is quantized into equal-length segments, w. We scale the data so that the search radius 1 by dividing by x_{min}, the desired search radius, $\mathbf{a}^{\star} = \frac{\mathbf{a}}{\sqrt{x_{min}}}$.

Insertion into the hash table occurs by summing the elements of the vector returned by the hash function modulo a large prime number [13] and folding by the hash table size N:

$$\begin{aligned} g &= h_{\mathbf{V},\mathbf{b}}(\mathbf{a}) && (5) \\ t1 &= ((\sum_{i=1}^{k} r_i^{'} g_i) \mod 2^{32}-5) \mod N && (6) \\ t2 &= ((\sum_{i=1}^{k} r_i^{''} g_i) \mod 2^{32}-5), && (7) \end{aligned}$$

with $r_i^{'}$ and $r_i^{''} \sim U(0, 2^{32}-1)$. The first hash key $t1$ is the hash table slot into which we record the track index. The second hash value, $t2$, is a fingerprint for the hash location that disambiguates hash keys pointing to the same slot but due to folding modulo N in the first hash key. These hash keys implement the efficient table lookup procedure that makes LSH work.

2) LSH Retrieval: To efficiently retrieve tracks that are similar to a query track, we hash each query shingle, $\mathbf{q}$, using the hash functions of Equation 4 into the L hash tables H^j. We generate the hash keys using Equation 7 and inspect the hash table buckets. We increment the rank count for track indexes that were stored in hash buckets pointed to by the hash keys. This step implements Equation 2 via hashing. We then apply Equation 3 to rank all the tracks and sort the results into descending rank order.

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

IV. Modeling distances between shingles

The distance between two shingles is the Euclidean distance (in this M-dimensional space) between the two vectors. These vectors are long, perhaps hundreds of dimensions. Finding near neighbors in this kind of very high-dimensional space is hard.

Most importantly, the curse of dimensionality [24] means that most data points are nearly equidistance from each other. This is demonstrated in Figure 1, which shows a histogram of the distance between points placed at random in a series of high-dimensional spaces. As the dimensionality of the feature space grows, the distribution of distances becomes more sharply peaked. This suggests that finding near neighbors in a high-dimensional space might be ill posed [24]—all points are nearly identically distant from each other. We are only interested in the left-hand tail of this distribution, the pairs of points with the smallest distances between them, but still we need to be careful to not look at too many points.

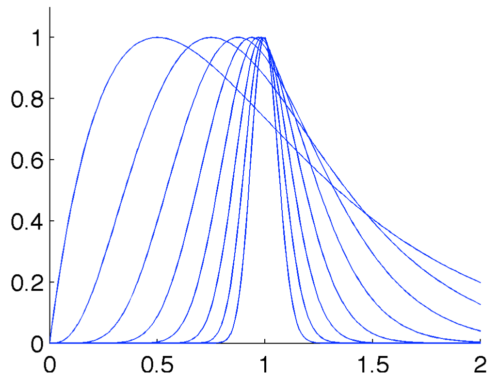

Fig. 1. A histogram of normalized distances for points placed uniformly at random in 4 (left-most curve), 8, 16, 32, 64, 128, 256, and 512 (inner curve) dimensional spaces. As the dimensionality of the space increases the distribution gets narrower, indicating that most points are equidistant from each other.

We wish to know if our data, from similar songs, is sufficiently non-random so that the near neighbor question is well posed. This analysis is also important because it will help us determine the optimum radius when using LSH to search for similar points. In this section we derive analytic approximations for five quantities related to our problem:

IV-A the pdf of uncorrelated shingle distances

IV-B an estimate of distribution parameters based on sampling distances to other shingles.

IV-C the pdf of the nearest neighbor in a chi-squared distribution

IV-D the pdf of the nearest neighbors for audio shingles

IV-F a decision rule for determining related shingles.

The pdf of uncorrelated shingle distances follows a chi-squared distribution (see Section IV-A). We derive an approximation of this distribution for points that are close to each other. We then derive a maximum-likelihood estimate of the distribution parameters given a set of sample data. Furthermore, we can derive an estimate of the pdf for the distance to the single point that is closest to the query. This is what we need to derive a threshold, a decision rule, that allows us to decide if a song's shingles represent a correlated or uncorrelated sample.

A. PDF of Shingle Distances

Our technique to related songs is based on short snippets of sound we call shingles. For this analysis, we start with the assumption that each element of the song's feature vector is from a Gaussian distribution. To calculate the distance between two shingles, we subtract two Gaussian variables, square the result, and sum over all dimensions. The magnitude of the difference between two i.i.d. Gaussian random vectors is also Gaussian and the probability density function (pdf) is given by the Chi-Squared, χ^2, distribution.

The shape of this distribution depends on the number of independent dimensions in a shingle. The pdf of distances will be narrow if each frame of data used to create the shingle is independent of the other (temporal) frames. The pdf of distances will be wide if some of the frames used to create the shingle are duplicates of others. This affects our calculation of the tails of the distribution, where the nearest neighbors are found, and motivates the discussion below about the degrees of freedom.

Consider two different extreme cases. First, assume the elements of the shingles are drawn from completely independent Gaussian processes with an overall population mean μ and variance σ^2. Then each of the M elements in each shingle is independent of all the others in that shingle, and of all the elements in all other shingles (assuming the shingles are taken with no overlap). By definition, the squared distance between the shingles are distributed as $\sigma^2\chi^2_M$, where χ^2_M is a random variable consisting of the sum of squares of M Gaussian random variables with zero mean and unit variance. At the other extreme, consider a process where all elements of the shingle are a single identical random variable. Then all of the M elements of each shingle are the same, and the squared distance are distributed as $M\sigma^2\chi^2_1$: one single squared quantity, multiplied by M. Note that both of these distributions have the same mean, but the χ^2_M distribution is much narrower.

In our case, we form the M-element feature vector from $M/12$ concatenated frames of PCP features of music. We expect the distribution of the squared distances to lie somewhere between the two extreme distributions above. Not only might some of the frames be duplicated because the acoustic signal might change slowly, there are likely to be less than 12 degrees of freedom in the chroma of one frame[1]. Successive frames are highly correlated, and so will add few extra degrees of freedom—certainly not as many as 12 per frame. Denoting the effective number of degrees of freedom as d, we hypothesize the squared distances between shingle vectors *drawn from unrelated songs* are a random variable of the form $\sigma^2 \frac{M}{d}\chi^2_d$.

The probability density function for a χ^2 distribution with d independent degrees of freedom is

$$f_\chi(x;d) = \frac{1}{(2^{d/2})\Gamma(d/2)} x^{(d/2)-1} e^{-x/2} \tag{8}$$

Therefore, the pdf of the *squared* distance x between two of these M-dimensional shingles, each with d effective degrees of freedom is

$$\begin{aligned} f_s &= d/(M*\sigma^2) f_\chi(d/(M*\sigma^2)x; d) && (9) \\ &= \left(\frac{d}{2\times M\times\sigma^2}\right)^{\frac{d}{2}} \frac{1}{\Gamma\left(\frac{d}{2}\right)} x^{\frac{d-2}{2}} e^{-\frac{d}{2\times M\times\sigma^2}x}, && (10) \end{aligned}$$

where σ^2 is the variance of the element-by-element distances and $0 < x < \infty$. Here d is a parameter expressing the internal structure of that space—how correlated different components of the individual feature vectors are with each other, and with the components of other feature vectors.

Near $x = 0$, that is, near the minimum possible distance between two feature vectors, this distribution is approxi-

[1]Perhaps the upper limit will be reached only in completely atonal music. In simple, harmonic music with a limited chord repertoire, the number of effective degrees of freedom is smaller than this.

mately a power law. By expanding the exponential in Equation 10 as a power series in x and taking the limit, keeping only the lowest-order term in x, we approximate the distance pdf with

$$f_s \approx \left(\frac{d}{2\times M\times\sigma^2}\right)^{\frac{d}{2}} \frac{1}{\Gamma\left(\frac{d}{2}\right)} x^{\frac{d-2}{2}}. \tag{11}$$

The cumulative distribution function $F(x)$ is the integral of Equation 11 with respect to x, which near $x = 0$ is

$$F(x \text{ for } ||x|| \ll 1) \approx \frac{2}{d}\left(\frac{d}{2\times M\times\sigma^2}\right)^{\frac{d}{2}} \frac{1}{\Gamma\left(\frac{d}{2}\right)} x^{\frac{d}{2}} \tag{12}$$

These two expressions give us an approximation for the pdf and the cdf of the shingle distances for the nearest neighbors, the left-tail of the distribution.

B. Fitting χ^2 distributions

In this section we use our above analysis of the distribution of shingle distances to fit parameters to observations drawn from two classes of shingle i) drawn from pairs of remixed songs and ii) drawn from pairs of unrelated songs.

As the χ^2 distribution is in the exponential family we can rewrite the shingle-distance pdf (10) to make it clear what the sufficient statistics are— that is, what numbers we can compute from samples to help us estimate parameters of the distribution. Thus we put everything in Equation 10 in terms of an exponential

$$f_s = \exp\left[\frac{d}{2}\log\left(\frac{d}{2M\sigma^2}\right) - \log\left(\Gamma\left(\frac{d}{2}\right)\right) + \frac{d-2}{2}\log(x) + \frac{-d}{2M\sigma^2}x\right] \tag{13}$$

which makes it obvious that the log likelihood, $\log\mathcal{L}$, for N data points x_i is

$$\exp\left[N\frac{d}{2}\log\left(\frac{d}{2M\sigma^2}\right) - N\log\left(\Gamma\left(\frac{d}{2}\right)\right) + \frac{d-2}{2}L + \frac{-d}{2M\sigma^2}S\right] \tag{14}$$

where $S = \sum_i x_i$ and $L = \sum_i \log(x_i)$ are sufficient statistics for this distribution—as the expression for the likelihood depends only on the $\sum$s, and not on how those $\sum$s are made up.

The maximum likelihood estimates for d and σ^2 are then given by solving for a stationary point. We now have

$$\frac{\partial\log\mathcal{L}}{\partial d} = -\frac{1}{2M\sigma^2}S + \frac{1}{2}C + \frac{N}{2}\log\frac{d}{2M\sigma^2} + \frac{N}{2} - \frac{N}{2}\Psi\left(\frac{d}{2}\right) \tag{15}$$

$$\frac{\partial\log\mathcal{L}}{\partial\sigma^2} = \frac{d}{2M\sigma^4}S - \frac{Nd}{2\sigma^2}. \tag{16}$$

[Ψ is the digamma function, which is $\frac{\mathrm{d}\log\Gamma(x)}{\mathrm{d}x}\ \frac{1}{\Gamma(x)}\frac{\mathrm{d}\Gamma(x)}{\mathrm{d}x}$. It is a continuous-domain analogue of $\sum_x \frac{1}{x}$]

By setting equation 16 to zero, we immediately get

$$\sigma^2 = \frac{1}{M}\frac{S}{N}; \tag{17}$$

substituting into 15 we get

$$\frac{\partial\log\mathcal{L}}{\partial d} = \frac{C}{N} + \log\left(\frac{d}{2}\right) - \log\frac{S}{N} - \Psi\left(\frac{d}{2}\right), \tag{18}$$

and solving for zero to find

$$\log\left(\frac{d}{2}\right) - \Psi\left(\frac{d}{2}\right) = \log\frac{S}{N} - \frac{C}{N}. \tag{19}$$

Then replacing $\log(x) - \Psi(x) = Y(x)$, and using Y^{-1} to represent the functional inverse of $Y(\cdot)$ we find the maximum likelihood estimate for the number of independent parameters in the data

$$d = 2Y^{-1}\left(\log\frac{S}{N} - \frac{C}{N}\right). \tag{20}$$

C. Order statistics

When doing these retrieval tasks we find a shingle's nearest neighbors and their corresponding distances. In the "null hypothesis" (unrelated) case, there are a set of N distances, one for each point in the database. We wish to find the pdf of these distance, first assuming that the query is unrelated (i.e. independent) to all the other points. This is the null hypothesis, and from this we can estimate a threshold for when two songs are related.

Following the derivation in Appendix 2 of Cox [18], $x_{\min}$ is the minimum value of a set of N values from the distribution $f(x)$ with cumulative distribution function $F(x)$. We wish to find the probability distribution function of $y_{\min}$. Note that for a value y to be the minimum value, all N values observed must be greater than or equal to that value, which occurs with probability $(1 - F(x))^N$.

Assume the probability density function has no weight below a lower limit, which we set without loss of generality to $x = 0$. Further assume that the cumulative distribution for a variable x near the distribution's finite lower limit fits a power law in x, so that $F(x) = ax^c$ for x close to 0. We are interested in the distribution of the minimum value $x_{\min}$ from a set of size N of i.i.d. values x_i drawn from this distribution.

Consider the following expressions for the cumulative probability of $x_{\min}$, the minimum for a given set of N points.

$$\begin{aligned} G(y) &= \Pr(x_{\min} < x) && (21)\\ &= 1 - \Pr(x_{\min} > x) && (22)\\ &= 1 - \Pr(x_i > x, \forall i) && (23)\\ &= 1 - (1 - F(x))^N && (24) \end{aligned}$$

In order to derive the distribution of the minimum, let us consider not $x_{\min}$ itself but

$$w = x_{\min}/b_N \tag{25}$$

where we will choose a value for the constant b_N later (as a function of N). The cumulative distribution function $G'(w)$ for the scaled minimum statistic is computed from the cumulative distribution function of the original data, $F(x)$, because for w to be the minimum value of x (scaled by b_N), all N values that we have drawn from x must be equal to or greater than that minimum value. This requires that

$$1 - G'(w_0) = P(w > w_0) = [1 - F(wb_N)]^N \tag{26}$$

With the assumption that the minimum value we are interested in lies in the power-law range (i.e. is sufficiently close to the actual minimum of the distribution for y), we replace F in Equation 12 with

$$F(x) = a(x)^c, \tag{27}$$

getting

$$1 - G'(w_0) = P(w > w_0) = (1 - a(w_0 b_N)^c)^N \tag{28}$$

Now, we exploit our freedom to choose b_N (or equivalently, the scale of w) to help us. A good choice is

$$b_N = 1/(aN)^{1/c}, \tag{29}$$

because this gives us

$$1 - G'(w_0) = (1 - (w_0)^c/N)^N \tag{30}$$

In the limit of large N, $(1 - z/N)^N$ tends to e^z, therefore $1 - G'(w_0) = exp(-(w_0)^c)$, or by rearranging and dropping the subscript, $G'(w) = 1 - exp(-(w^c))$.

This gives us the cumulative distribution function (CDF) as a function of w. The probability distribution function is the derivative of this CDF with respect to w, which is

$$g(w) = dG'(w)/dw = cw^{c-1}e^{-(w^c)}, \tag{31}$$

from which the mean and variance of the w distribution are easy to calculate—and then it's easy to get the mean and variance of the squared distance distribution x that we actually care about, by scaling by b_N.

The mean of this density is

$$\bar{w} = \int_0^\infty cw^c \exp(-w^c)\mathrm{d}w \tag{32}$$

and by converting to $u = w^c$ so $\mathrm{d}u = cw^{c-1}\mathrm{d}w$ we get

$$\bar{w} = \int_0^\infty cu^{\frac{1}{c}} \exp(-u)\mathrm{d}u = \Gamma\left(1 + \frac{1}{c}\right). \tag{33}$$

Similarly, the expectation of w^2 is $\Gamma\left(1 + \frac{2}{c}\right)$.

We expand these expressions for small x with the assumption that $1/c$ and $2/c$ in the above expressions will turn out to be small. Here c is half the effective number of degrees of freedom d, which we hope will be reasonably large for sequences of PCP vectors, but if we estimate a value of d that turns out to be small (say less than about 6 or so) then the following approximation is not valid. By a Taylor series

$$\Gamma(1 + x) = \Gamma(1) + \Gamma'(1)x + \frac{1}{2}\Gamma''(1)x^2 \tag{34}$$

$$= 1 - \gamma x + \frac{1}{2}\left(\frac{\pi^2}{6} + \gamma^2\right)x^2 \tag{35}$$

where γ is the Euler-Mascheroni constant, ~ 0.577.

The variance of w is equal to $\sigma_w^2 = \mathrm{E}[w^2] - \mathrm{E}[w]^2$, so, ignoring terms smaller than $\frac{1}{c^2}$, is simply $\sigma_w^2 = \frac{\pi^2}{6}\frac{1}{c^2}$.

D. Minimum distances for shingles

We compute the minimum value of a large set of values, of size N, by comparing a shingle to all shingles of an unrelated song. This will be distributed according to Equation 31. Using Equations 25, 29 and then rewriting Equation 12 using the factors in Equation 27 we conclude

$$w = \left[N\frac{2}{d}\left(\frac{d}{2 \times M \times \sigma^2}\right)^{\frac{d}{2}} \frac{1}{\Gamma\left(\frac{d}{2}\right)}\right]^{\frac{2}{d}} \times x_{\min}. \tag{36}$$

so the pdf of the minimum of unrelated shingle distances is

$$\frac{d}{2} w^{\frac{d-2}{2}} e^{-w^{d/2}} \tag{37}$$

which (as discussed above) has mean $\Gamma\left(1 + \frac{2}{d}\right) \approx 1$ and variance $\approx \frac{\pi^2}{6}\frac{4}{d^2}$.

We divide the mean and variance of the pdf in Equation 37 to find

$$\frac{\overline{x_{\min}}^2}{\sigma^2_{x_{\min}}} = \frac{d^2}{4}\frac{6}{\pi^2}, \tag{38}$$

which we rearrange to find

$$d^2 = 4\frac{\pi^2}{6}\frac{\overline{x_{\min}}^2}{\sigma^2_{x_{\min}}}. \tag{39}$$

This is an estimate for the number of independent dimensions, d, given the ratio between the mean and the variance of the minimum distances between a query and all unrelated shingles. This is true for large values of d.

E. Related song classifier

We derived the distribution in Equation 37 under the assumption that we generated the shingles we are comparing from two distinct, independent processes—that is, the songs from which the shingles are drawn are unrelated. Under this assumption (and all the modelling assumptions above) the distribution for the minimum holds. If an empirical distribution with significant differences is obtained, then we may be able to conclude that one or more of our modeling assumptions is violated.

In the context of related-song detection, the assumption that we expect to see violated is that of the independence of the two songs. If the two songs are related, for example through reuse of content in a remix, then the processes are likely coupled. The signature for this is probably a significantly lower mean value for the minimum.

Most of this modeling aside, the procedure is relatively clear from the point of view of solving a task: compute a baseline "unrelated" distribution for the minimum, approximated using the discussion above with measured mean and variance. Then the test for "related"ness between two songs is that the mean minimum distance over that song is outside whatever confidence interval we calculate for the mean minimum distance of unrelated songs.

F. Estimating the Optimal Search Radius

Our motivation for the analysis above was automatic optimization of the search radius, x_{min}, for retrieval using locality sensitive hashing algorithms. Here we show how we use the above analysis to achieve this. First, we rearrange Equation 36 to solve for x_{min} and solve for the factor w using the CDF form of the pdf given in Equation 37, giving:

$$cdf(w) = 1 - e^{-w^{d/2}}. \tag{40}$$

We then perform a reverse lookup on the CDF at the point, $cdf(w) = f$, that achieves the required proportion of retrieved non-relevant points (false positives). In our experiments we set $f = 0.01$ yielding an expected proportion of 1% false positives per track. This value is chosen so that the proportion is far below the expected proportion of retrieved shingles for similar tracks. The value of w is then obtained using:

$$\ln(w) = \frac{2}{d}\ln(-\ln(1-f)). \tag{41}$$

Finally, given d, σ^2, and w, we find x_{min} by rearranging Equation 36:

$$x_{min} = \frac{M\sigma^2 w}{N^{\frac{2}{d}}\frac{d}{2}(\frac{2}{d})^{\frac{2}{d}}(\frac{1}{\Gamma(\frac{d}{2})})^{\frac{2}{d}}}. \tag{42}$$

In our previous work, [8][9][10], we established that locality sensitive hashing algorithms solve music sequence similarity retrieval efficiently, and with the same precision as exact solutions, if the radius is set near the optimum value. The method given in this section provides a solution to the optimum radius given a distribution of distances for non-relevant data.

In the following section we present the results of three experiments designed to test the robustness and generality of the solution on a wide range of music retrieval tasks with different specificities, data sets and audio features.

V. RESULTS

We conducted three experiments to test our approach for music similarity, how well our models fit the data, and the accuracy of the optimum radius solution. The first experiment was Apocrypha identification, for which the task was retrieval of the original sources of 49 suspected falsely-attributed commercial recordings from a database of 2257 recordings consisting of 100 hours of audio. The second experiment was Opus Retrieval, for which the task was retrieval of the different performances of the work contained in the query performance, the opus, against a 2257-track database of many different works performed by many different artists. The final experiment was Remix Retrieval; here, the task was to retrieve remixed versions of 82 query tracks from a database of 2018 tracks consisting of 220 hours of audio. Remixes contain a small snippet of the original tracks audio, typically a vocal sample, and place it in an entirely new acoustic context so that there is only a small region of similarity between the original and remixed track. These properties make the remix task difficult for automated retrieval.

To solve these tasks with large databases, we need to solve them using a locality sensitive hashing algorithm which requires us to specify hash functions for a given retrieval radius, x_{min}. The LSH algorithm returns only those points that fall with the given radius of the query point. To find similar songs, we count the number of collisions between songs. A collision occurs when the distance between pairs of shingles in different tracks falls below the specified minimum distance, x_{min}. A maximum of one collision is recorded per query shingle for each track in the database. Thus the maximum collision count is the length of the query track.

The optimal radius, x_{min}, is both data and task dependent. We show this by using the same system on the same data to perform two different tasks in Experiments 1 and 2 at different specificities. Similarity for these tasks is entirely defined by the non-similar training data, so the methods generalize to any task for which a distance distribution of non-similar features can be constructed.

Each of the three tasks—Apocrypha, Opus and Remix retrieval—require sensitivity to different musical and acoustic. Table I lists the invariance properties required for each of the three tasks. As described below, we chose features for each task that are invariant to the listed properties.

TABLE I
INVARIANCE PROPERTIES FOR THE APOCRYPHA, OPUS AND REMIX TASKS

Invariance	Apocrypha	Opus	Remix
Performer		X	
Environment		X	X
Recording Equip.		X	
Instrumentation			X
Tempo		X	X
Dynamics		X	X
Volume	X	X	X
Pitch/Key		X	X
Harmony			X
Structure		X	X
Effects	X		X
Audio Encoding	X		X

A. Apocrypha

As reported in the Classical music press [11][12], there are 49 commercial recordings of the Chopin Mazurkas by concert pianist Eugen Indjic that were copied, modified and falsely attributed to pianist Joyce Hatto under the *Concert Artists* record label. Since each of the 49 recordings by Hatto is a modified recording of a different artists' performance, the goal of the first experiment was to robustly identify the mis-attributed recordings against the complete set of recordings by 125 artists of all 49 Mazurkas, giving a database of 2257 tracks.

The specificity of this task is similar to that of audio fingerprinting. We would like to establish whether two recordings are acoustically identical, but for some degree of signal transformation and distortion such as filtering or time compression/expansion.

1) Features: There are between 31 and 62 recordings of each of the Mazurkas in the database with a mean of 44 recordings for each Mazurka. For each performance, the chord sequences are identical. However, there are variations in the acoustic environment, expressive content such as tempo and dynamics, structure (which repeats are taken and how many times repeated), recording equipment, storage medium and playback device prior to digitization. Table I show a list of acoustic and musical invariance properties needed to perform the task.

From Table I we discern that, for the Apocrypha task, our chosen features must be sensitive to the specific details of a performance whilst being robust to possible effects processing such as filtering, time compress/expand and artificial reverberation. Given the specificity of the task, we chose Cepstral Coefficients as the features.

We extracted 20 Log-Frequency Cepstral Coefficients (LFCC) at 100ms intervals using a 372ms hamming window with left and right channels extracted separately. We retained only the left channels in this experiment. A 12th-octave filterbank was used and cepstral coefficients were extracted using a Discrete Cosine Transform on the set of 81 log-magnitude log-frequency bins. We formed audio shingles by concatenating $30\times$ 20-dimensional LFCC features, with a hop size of 1 frame (100ms), yielding 600-dimensional vectors. Shingles were removed from the database if the power was below a threshold of $0.25\times$the mean power of the track from which the shingle was drawn. The shingles were normalized to unit vectors, thus retrieval was invariant to the absolute power levels of the shingles.

2) Method: We used Equations 10, 41 and 40 to fit a sample distribution of non-relevant inter-track shingle distances and obtain an estimate of the minimum distance x_{min} for rejecting samples as non-similar. To eliminate any possible error due to approximation, all distances were computed by exhaustive evaluation using Equation 1. Tracks were ranked using Equation 3 which counts the number of query shingles that are close to any of the track shingles for each track using the indicator function of Equation 2.

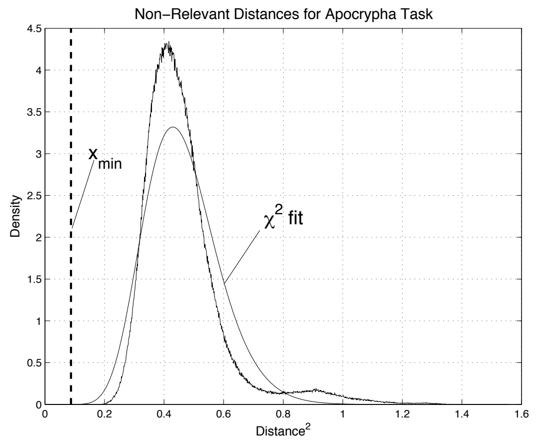

Fig. 2. Distribution of non-relevant distances for the Apocrypha task and their χ^2 fit. The estimated value of x_{min} is 0.095 for the LFCC shingles. This is used as the search radius for near neighbor retrieval algorithms.

Figure 2 shows the distribution of distances and the χ^2 fit. The estimate of the minimum search radius, x_{min}, was derived using Equations 41 and 42 using the estimated d and σ^2 parameters for the χ^2 distribution, estimated CDF factor z and a false alarm rate of 1%. Table II shows the values estimated from the non-relevant data set for each of the parameters.

TABLE II
ESTIMATED χ^2 PARAMETERS FOR APOCRYPHA TASK

Parameter	Value
x_{min}	0.095
d	34.3
σ^2	0.44
z	0.77

The estimate of the degrees of freedom value in Table II tells us that out of 360 dimensions in our shingles, 34 of them contribute to the variability in the distances. For stationary signals, over the 30-frame shingle window, the maximum value would be 20, one for each of the LFCC dimensions. A value higher than this informs us that the temporal concatenation of LFCC features is informative for this task.

3) Retrieval: We first constructed a suspect query list consisting of known misattributed recordings. The goal of the task was to rank the recordings and for the original song to be ranked at the top of the list.

The task is difficult because all of the remaining recordings for each work contain the same sequence of notes in the same key as well as many other consistencies due to established performance practice over the repertoire. Since, for the 49 Mazurkas, we know which recordings were copied to create the suspect queries, this set was used as a ground truth. If any of the remaining performances were sufficiently close to the ground-truth, a confusion would be manifest as a reduction in the rank of the retrieved ground-truth given the suspect query.

We counted all shingles in the database that fell on or below the minimum radius, x_{min} from the query shingles. This operation can be performed either by visiting all shingles in the database and computing the squared distance between them, or by constructing locality sensitive hash functions using the minimum radius estimate. The two methods produce near-identical results for optimal values of x_{min}, [9], but the LSH method is several orders of magnitude faster when the radius is set close to the optimal value.

For each query shingle, the distances to all shingles in each database track were measured and if any fell below the minimum distance, x_{min}, then the retrieved count was incremented for that track. This was repeated for all 2257 tracks in the database. The tracks were ranked by the number of matched query shingles, thus constituting a one-to-many mapping between query shingles and track shingles.

4) Apocrypha Results: The results of this first experiment were surprising. All 49 misattributed recordings were retrieved with rank 1 using the radius search method and the estimated x_{min}. In the first instance we retrieved the correct recording for each suspect recording from the remaining performances of the same work only. We then repeated the experiment using the entire collection of 2257 recordings as interference for each retrieval trial and the result remained perfect. We also tried repeating the experiment using a different method of ranking, the average distance of the 10 nearest returned shingles [9][10]; with this method of ranking the score decreased. Some of the target recordings were retrieved with lower rank than using our proposed retrieved-shingle count method.

Figure 3 shows the distribution of retrieved shingle counts for relevant and non-relevant tracks respectively. In this figure, we can see that the two distributions do not overlap, hence the perfect result is due to the high degree of separation between the number of retrieved shingles between the relevant and non-relevant classes. The comparison with the 10-nearest neighbor ranking method suggests that the task is not trivial and that our chosen method is robust for this task.

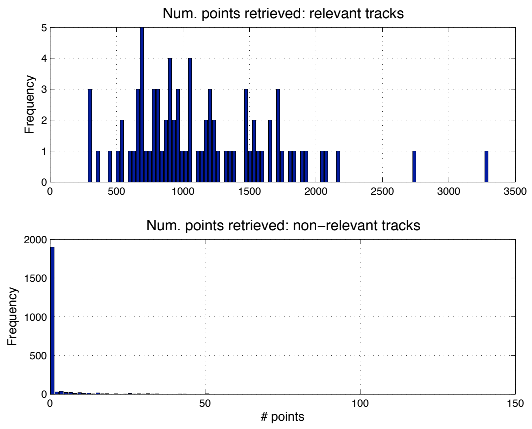

Fig. 3. Histogram of retrieved shingle counts for the Apocrypha task. The graphs show how many times the query shingles matched the database shingles for relevant and non-relevant tracks. The upper graph shows the counts for relevant data and the lower shows counts for non-relevant data. Performance in the task is related to the degree of separation between these graphs.

B. Experiment 2: Opus Retrieval

Our second experiment used the same audio data as experiment 1, but we made the task substantially harder by decreasing the specificity. The Opus retrieval task is defined as follows: given a query performance of one of the Chopin Mazurkas, retrieve all the performances of the same work given the entire 2257 track database. The task is difficult because performances are by different artists, with different expressive interpretations. Furthermore, each performance has different structuring due to choices over performing repeats and how many times to play them. The required invariance properties for this task are listed in Table I.

1) Features: To achieve the required invariance properties, we used 12 pitch-class profile features, one feature for each equal-temperament pitch class, constructed by assigning the energies in a $\frac{1}{12}$th-octave filterbank to the corresponding pitch-class bin. The features were computed over a 372ms window which was advanced by 100ms per frame and the resulting observation sequence was smoothed using a 10-point hamming window to reduce the degree of variation from non-harmony related signal properties. Finally, we constructed audio shingles by concatenating 30×100ms frames for each shingle, thus forming a sequence of 360-dimensional features.

2) Method: The method for retrieval was the same as Experiment 1. We formed a database of audio shingles consisting of all 2257 performances of the Chopin Mazurkas by 125 different artists. Given a single performance of each Mazurka, the goal was to retrieve the remaining performances of the same opus.

To fit the non-relevant distance distribution for the task, we collected a sample of 10s segments drawn from 40 non-related performances, i.e. different opuses performed by different artists. The squared distances of these samples were computed, rejecting intra-track distances, and we fit the distance distribution using the χ^2 estimation methods. Figure 4 shows the distribution of non-relevant shingle distances for the task as well as the χ^2 fit and x_{min} estimate.

Table III shows the parameters estimated from the non-relevant distance distribution for the Opus task. The false alarm rate was set to 1%, as in the previous task, so the only differences in the method were the features used and the set of non-relevant data used to create the fitted distribution.

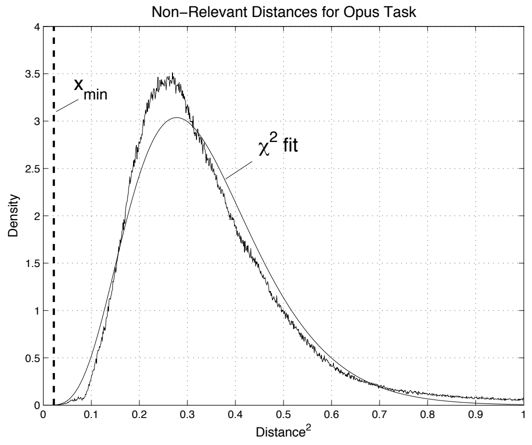

Fig. 4. Distribution of distances for the Opus task and corresponding χ^2 fit. The estimation for the minimum distance yielded $x_{min} = 0.01$ for the given background data using smoothed 12-chroma PCP shingles.

TABLE III
ESTIMATED χ^2 PARAMETERS FOR OPUS TASK

Parameter	Value
x_{min}	0.011
d	8.35
σ^2	0.32
z	0.33

The degrees of freedom estimate for the data in this task is substantially lower than for the Apocrypha task. This makes sense because we have designed the features to be more tolerant of differences between recordings, thus reducing the amount of variability between them.

3) Opus Results: The precision-recall graph for the Opus retrieval task are shown in Figure 5. Overall, the precision was very high for recall rates below 90%. For most of the 49 Mazurkas, there were 2-3 outliers in our database. On inspection, these were typically early recordings that were transferred from 78 RPM vinyl media and contained a high degree of surface noise and extreme wide-band filtering, additionally, the cut-off frequency for these tracks was typically much lower than the remaining tracks. These results suggest that a near-perfect score can be obtained for Opus retrieval if outlying recordings are first removed or pre-processed to make them compatible with the system.

Figure 6 shows the distribution of the number of retrieved shingles over all query tracks for relevant and non-relevant tracks. The precision-recall graph suggests that the distributions of counts do not collide on a per-query basis. In other words, when a query track produces a relatively high number of hits in non-relevant tracks, the relevant tracks get a proportionally higher number of hits thereby preserving the correct rank ordering. This makes the count method a robust classifier.

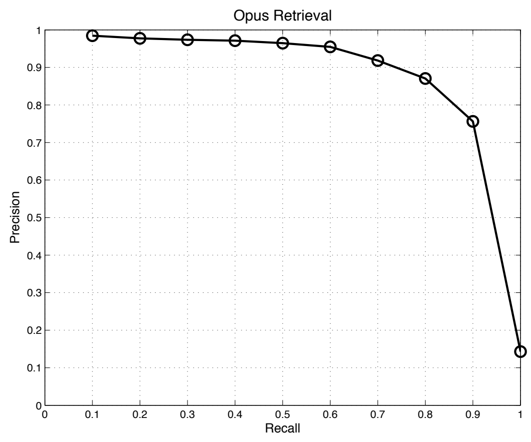

Fig. 5. Precision-recall evaluation plot for the Opus retrieval task.

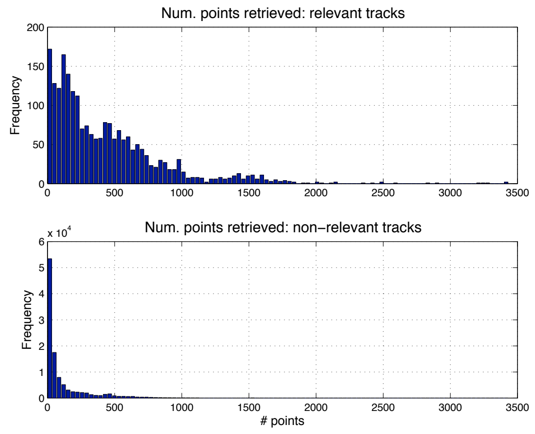

Fig. 6. Distribution of retrieved shingle counts for the Opus.

C. Experiment 3: Remix Retrieval

To test the method on a different data set, consisting of pop music and Jazz recordings, we devised a third experiment with specificities falling between Experiments 1 and 2. The goal in this task is to retrieve remixed versions of a query track from the database of 2018 recordings consisting of popular and jazz recordings.

1) Features: This task required a high degree of robustness to many different acoustic and musical properties. A summary of the required invariance properties for this task is given in Table I. To meet the invariance criteria we used 12-dimensional PCP features that were not smoothed because we wanted to retain sensitivity to any specific audio content from an original track that was used for a remix. Shingles were formed from the 12-d features making a series of 360-dimensional vectors. The vectors were normed and those shingles with power below $0.25\times$the mean power removed.

2) Method: A background of non-relevant distances was sampled from 40 un-related tracks, 10 seconds of shingles from each, forming a total of 400s of distance data. The distances between all the background vectors were calculated using convolution, which implements Euclidean distance in the case of unit vectors.

The minimum radius threshold was computed using Equation 41 and Equation 42. The estimated parameters are shown in Table IV for the distribution of non-relevant distances.

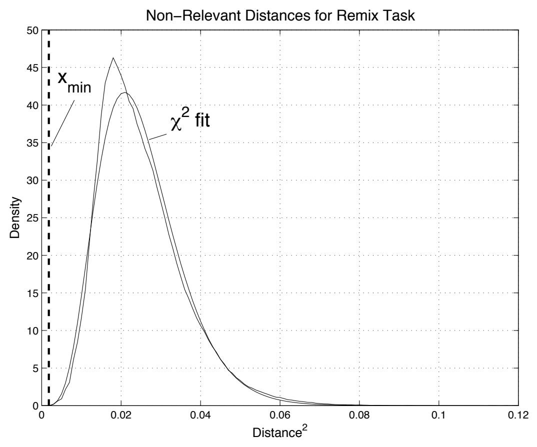

Fig. 7. Distribution of distances for the Remix task and χ^2 fit. The estimation for the minimum distance yielded $x_{min} = 0.004$ for a false alarm rate of 1%.

A set of queries was selected consisting of 82 tracks with 3-10 remixes each. The 82 tracks included all the remixes such that the base set was 20 tracks with remix versions expanding this set to 82. Each query shingle was compared against the shingles in each database track. If the query shingle resulted in a match, with distance below $x_{min} = 0.004$, then the count of minimum-value matches for the track was incremented. The counts were used to order the tracks with the track having the highest minimum-value counts placed in first rank.

TABLE IV
ESTIMATED χ^2 PARAMETERS FOR REMIX TASK

Parameter	Value
x_{min}	0.004
d	7.15
σ^2	0.024
z	2.08

3) Results: Figure 8 shows the precision/recall graph for the Remix retrieval task. The performance is satisfactory given the difficulty of the task, with precision at 50% for 100% recall and substantially higher for lower recall rates. The break-even point in precision-recall for this task was somewhere around 75% for the given features. Setting the search radius to a higher value did not improve the result and substantially slows down retrieval when using LSH.

Figure 9 shows the distribution of minimum-distance counts for the relevant and non-relevant tracks respectively. The top graph shows the distribution of the number of relevant shingles retrieved over all remixed tracks. The bottom graph shows the distribution of number of shingles retrieved for all non-relevant tracks. The distributions of counts overlap substantially more than the previous two experiments indicating that the task is more difficult.

4) Discussion: The results from our three experiments demonstrate that we were able to correctly order the tracks by relevance to the queries by rejecting the hypothesis that they were non-relevant. This method has two main advantages over other methods such as those based on averaging the retrieved distances per track. Firstly, hypothesis testing improves performance without requiring extensive empirical evaluation of parameters, such as k in the k-NN method. Secondly, and more importantly for our purposes, the minimum value, x_{min} is a distance which is interpreted

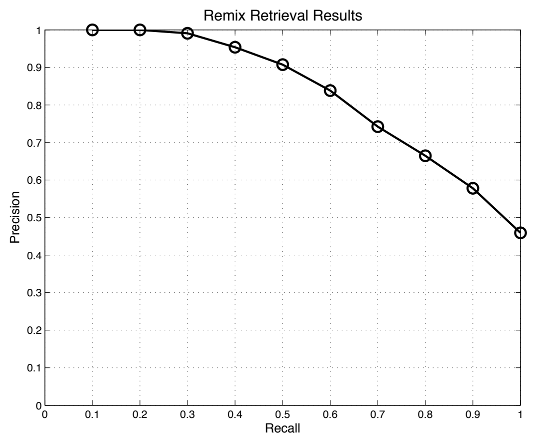

Fig. 8. Precision-recall evaluation of the Remix task.

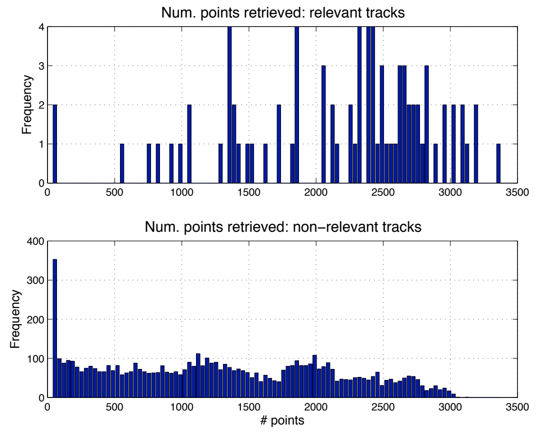

Fig. 9. Distribution of retrieved shingle counts for the Remix task.

as an absolute threshold on the distances of shingles to retrieve. This property means that it can be used to estimate the threshold for implementing our retrieval algorithm using Locality Sensitive Hashing, which we have shown in our previous work to perform with the same accuracy as the exact methods employed here, but more efficiently– approximately two orders of magnitude faster to retrieve relevant tracks from a 2000-track database.

Furthermore, the results indicate that our analysis of distance distributions is both correct and appropriate to the tasks we set out to solve. We believe that this analysis will lead to greater understanding of how retrieval systems perform for music tasks and may lead to new and improved algorithms for retrieval.

VI. Conclusions

[Malcolm needs to edit this yet...]

We have shown that 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 χ^2 distributions. With a suitable kernel space we showed that a threshold classifier can be used for robust audio matching for mid-specificity problems such as Apocrypha, Opus and 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 decrease in time complexity of between 10 and 100 for our data set.

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.

References

[1] Christophe J. C. Burges, John C. Platt, and Soumya Jana, "Distortion Discriminant Analysis for Audio Fingerprinting," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 3, pp. 165–174, 2003.

[2] Shumeet Baluja and Michele Covell, "Audio Fingerprinting: Combining Computer Vision & Data Stream Processing," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing,* vol.2, no., pp.II-213–II-216, 15–20 April 2007.

[3] Shumeet Baluja and Michele Covell, "Learning *Forgiving* Hash Functions: Algorithms and Large Scale Tests," International Joint Conference on Artificial Intelligence, 2007.

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

[5] Mark A. Bartsch and Gregory H. Wakefield, "Audio Thumbnailing of Popular Music Using Chroma-Based Representations," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 96–104, 2005.

[6] 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.

[7] 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.

[8] M. Casey and M. Slaney. The importance of sequences in music similarity. *in Proc. ICASSP*, 2006.

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

[10] M. Casey and M. Slaney, Fast Retrieval of Remixed Songs Using Locality Sensitive Hashing. In *Proc. ICASSP*, Honolulu:HI, May 2007.

[11] N. Cook, C. Sapp,Purely coincidental? Joyce Hatto and Chopin's Mazurkas, *http://www.charm.rhul.ac.uk/ content/ contact/ hatto_article.html*, Centre for the History and Analysis of Recorded Music (CHARM), Royal Holloway, University of London, 2007.

[12] Revenge of the Fraudster Pianist, *The Daily Mail*, UK, 24 February, 2007

[13] 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.

[14] T. Darrell and P. Indyk and G. Shakhnarovich (eds.). Nearest Neighbor Methods in Learning and Vision: Theory and Practice. MIT Press, 2005.

[15] 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

[16] 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.

[17] Michael Clausen and Frank Kurth, "A Unified Approach to Content-Based and Fault-Tolerant Music Recognition," *IEEE Transactions on Multimedia*, vol. 6, no. 5, pp. 717–731, 2004.

[18] David R. Cox and David V. Hinkley, "Theoretical Statistics", Chapman and Hall, 1974

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

[20] Matasaka Goto, "A Chorus Section Detection Method for Musical Audio Signals and Its Application to a Music Listening Station," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 5, pp. 1783–1794, 2006.

[21] 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.

[22] Jaap Haitsma, Ton Kalker, and Job Oostveen, “Robust audio hashing for content identification,” in *International Workshop on Content-Based Multimedia Indexing (CBMI’01)*, Brescia, Italy, September 2001.

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

[24] P. Indyk and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Proceedings of the Symposium on Theory of Computing,* 1998.

[25] Y. Ke, D. Hoiem, and R. Sukthankar, “Computer Vision for Music Identification,” *IEEE Conference on Computer Vision and Pattern Recognition,* Vol. 1, June, 2005, pp. 597–604.

[26] 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

[27] Fabian Mörchen, Alfred Ultsch, Michael Thies, and Ingo Löhken, “Modeling Timbre Distance With Temporal Statistics From Polyphonic Music,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 1, pp. 81–90, 2006.

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

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

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

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

[32] Malcolm Slaney and Michael Casey. Locality Sensitive Hashing: Finding a needle in a haystack. To be published in *IEEE Signal Processing Magazine*, May 2008.

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

[34] 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

[35] E. Weinstein and P. Moreno, P, “Music Identification with Weighted Finite-State Transducers,” *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol.2, no., pp.II-689–II-692, 15–20 April 2007.

[36] In-Kwon Yeo and Hyoung Joong Kim, “Modified Patchwork Algorithm: A Novel Audio Watermarking Scheme,” *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 4, pp. 381–386, 2003.

[37] Changsheng Xu, Namunu C. Maddage, and Xi Shao, “Automatic Music Classification and Summarization,” *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 5, pp. 441–450, 2005.

[38] Cheng Yang. Efficient Acoustic Index for Music Retrieval with Various Degrees of Similarity. *Proc. ACM Multimedia*, 2002.