International Journal of Semantic Computing
Vol. 3, No. 2 (2009) 209–234


World Scientific
www.worldscientific.com

# MULTI-VERSION MUSIC SEARCH USING ACOUSTIC FEATURE UNION AND EXACT SOFT MAPPING

YI YU* and KAZUKI JOE†

*Department of Advanced Information and Computer Sciences*
*Nara Women's University, Kitauoya nishi-machi*
*Nara 630-8506, Japan*
**yuyi@ics.nara-wu.ac.jp*
†*joe@ics.nara-wu.ac.jp*

VINCENT ORIA

*Department of Computer Science*
*New Jersey Institute of Technology*
*University Heights Newark, NJ 07102, USA*
*oria@cis.njit.edu*

FABIAN MOERCHEN

*Siemens Corporate Research, Integrated Data Systems*
*755 College Road East Princeton, NJ 08540, USA*
*fabian.moerchen@siemens.com*

J. STEPHEN DOWNIE

*Graduate School of Library and Information Science*
*University of Illinois at Urbana Champaign, USA*
*jdownie@uiuc.edu*

LEI CHEN

*Department of Computer Science*
*Hong Kong University of Science and Technology*
*HKSAR, China*
*leichen@cs.ust.hk*

Research on audio-based music retrieval has primarily concentrated on refining audio features to improve search quality. However, much less work has been done on improving the time efficiency of music audio searches. Representing music audio documents in an indexable format provides a mechanism for achieving efficiency. To address this issue, in this work Exact Locality Sensitive Mapping (ELSM) is suggested to join the concatenated feature sets and soft hash values. On this basis we propose audio-based music indexing techniques, ELSM and Soft Locality Sensitive Hash (SoftLSH) using an optimized Feature Union (FU) set of extracted audio features. Two contributions are made here. First, the principle of similarity-invariance is applied in summarizing audio feature sequences and utilized in training semantic audio representations based on regression. Second, soft hash values are pre-calculated to help locate the searching range more

accurately and improve collision probability among features similar to each other. Our algorithms are implemented in a demonstration system to show how to retrieve and evaluate multi-version audio documents. Experimental evaluation over a real "multi-version" audio dataset confirms the practicality of ELSM and SoftLSH with FU and proves that our algorithms are effective for both multi-version detection (online query, one-query vs. multi-object) and same content detection (batch queries, multi-queries vs. one-object).

*Keywords*: Query-by-audio; music information retrieval; musical audio sequence summarization; feature union; exact locality sensitive mapping/hashing.

## 1. Introduction

Query-by-audio Music Information Retrieval (MIR) usually involves three main applications:

1) *Query-by-example/humming/singing* [1, 6, 21]. Given a fragment of the query song via a recording or a microphone as input, find melodies similar to this query fragment.

2) *Near duplicate audio detection* [2] *or cover song detection* [16, 18]. Given a query file and a music collection, find all audio files whose similarity to the query is above a specified threshold. The found items and the query may be cover versions of the same song.

3) *Plagiarism analysis* [3]. Given a suspicious song, find from the collection all audio files that are partly identical to this song even though their entire melodies may be different.

With a continuous surge of music documents on the Internet, we are confronted by the task of supporting scalable audio-based music search. However, in addition to the difficulties in describing semantic similarity of audios through features, the feature descriptor of musical audio sequence itself is very high dimensional, which makes it difficult to quickly detect audio documents that closely resemble an input query as the database size increases. To solve this hard problem, some researchers refine music representations to improve the accuracy of musical semantic similarity, using pitch [5, 19], Mel-Frequency Cepstral Coefficient (MFCC) [7, 8], Chroma [16, 18, 27]; some other researchers use index-based musical audio structures to speed up music similarity searching, exploiting tree structure [1, 4, 12], hierarchical structure [10], Locality Sensitive Hashing(LSH) [6, 9, 24], Exact Euclidean LSH ($E^2$LSH) [22, 24] and active-search [28]). In fact, focusing on either aspect is not enough for scalable query-by-audio music search. For example, in [18], beat-synchronous Chroma features are successfully used in matching similar music sequences. An accurate audio-based pairwise matching is obtained. Unfortunately, pairwise comparisons among feature sequences costs much time. LSH-based indexing structure was used to directly map the features to the integer values [6, 9, 22, 24]. However, little investigation was performed to judge whether the used features were able to represent the music audio information and sufficiently recognize two acoustic sequences.

This work jointly considers music representation and feature organization and aims to provide a scalable music retrieval system. Specifically, a novel melody-based

summarization principle, Feature Union (FU), by using multivariable regression, is presented to determine a group of weights that define an optimal combination of several audio features. In addition, we study the relationship between FU summarization and hash values and propose two novel search schemes called Exact Locality Sensitive Mapping (ELSM) and Soft Locality Sensitive Hash (SoftLSH). A demonstration system is presented to display multi-version audio retrieval based on our approaches and some existing techniques. We validate our algorithms with real world multi-version queries over a large musical audio dataset. Two tasks are considered: multi-version detection which locates all relevant items with an audio query as input (one-query vs. multi-object) and the same content detection which locates one relevant item with different queries (multi-queries vs. one-object). Evaluations by these two tasks show that (1) the proposed FU can represent human perceptual similarity better than other competitive feature sets and (2) ELSM and SoftLSH algorithms have a better tradeoff between accuracy and search time compared with that of regular LSH and exhaustive KNN.

Paper organization: We provide an introduction to MIR research and related work in Sec. 2. Then we define the query-by-audio music search problem and present the main algorithms in Sec. 3. We first review typical spectral features that are often used in music retrieval and study the similarity-invariant summarization. The potential problem of LSH with discrete hash values is addressed and two LSH variants, ELSM and SoftLSH are proposed. Performance evaluation of the proposed algorithms is conducted over large datasets. The experiment setup and result analysis are addressed in detail in Sec. 4. Finally we conclude in Sec. 5 with a summary and suggestions for future work.

## 2. Background and Related Work

Audio-based MIR is the interdisciplinary science of retrieving music information and it involves many aspects and techniques. In this section we explain the MIR model, report related techniques and distinguish them from our work.

### 2.1. *Our MIR model*

Figure 1 shows our content-based MIR system model based on audio content searching. From the viewpoint of development and system architecture, from top to bottom it consists of three levels: GUI interacting with users (L1), the key search engine which is the main focus of this work (L2), and the database consists of both original audio tracks and their suitable representations (L3). GUI receives query input from users and passes it to the search engine, where the "matching" module compares the query audio against audio tracks in the database and returns the ranked results to the GUI for the graphical display. From the viewpoint of retrieval procedure, from left to right the MIR system consists of three stages:

1) *Audio representation* (S1). The original acoustic data is not suitable for representing songs. Therefore suitable features are extracted. Audio data is usually

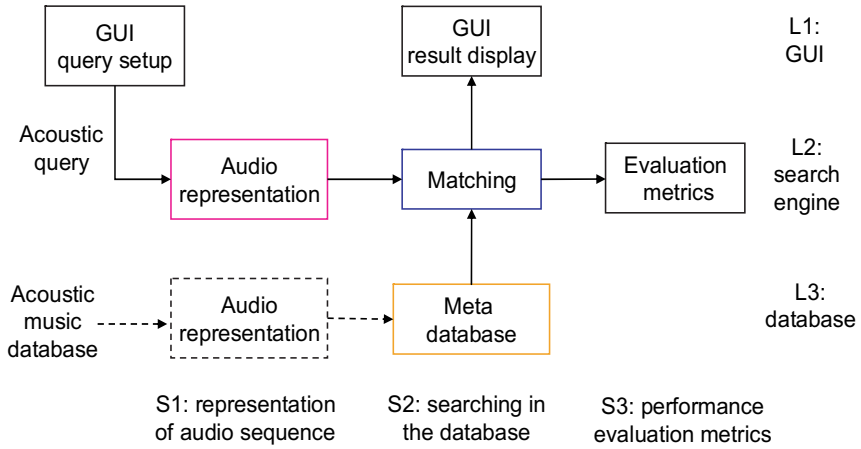


Fig. 1. Content-based music information retrieval model.

relatively long and is not stationary. Then it must be divided into frames, from each of which a feature vector is extracted and the whole song is represented as a feature sequence. Songs differ from each other with their specific scores, which are associated with spectrum. Therefore spectral features, such as Short Time Fourier Transform (STFT)[6], MFCC[7], pitch[5] and Chroma[16], are popular features. It is also possible to further compress the feature sequence, for example, by summarization.

2) *Matching* (S2). To determine whether two songs are similar or not, first a similarity criterion is chosen. There are several levels of similarity [6], from exactly the same copy to the same main melody with potential tempo variations. The focus of this work is the similarity in terms of the main melody. Since each song distinguishes itself with its score and spectrum, the main melody is embodied in the spectrum and the spectral similarity is adopted in this work. Under this criterion the spectral features of a query are compared against those of audio tracks in the database. This comparison can be divided into two steps. (a) The query is compared against one reference song. This possibly involves sequence comparison, which further depends on Dynamic Programming (DP) [29] to account for tempo variations. (b) The comparison is iterated over each reference song and a ranked list containing nearest neighbors of the query is generated. Both the sequence comparison in step (a) and the exhaustive search in step (b) are time-consuming. The semantic audio summarization and indexing-based searching methods can speed up the audio sequence comparisons and enable scalable retrieval, as is exactly the main work of this paper.

3) *Evaluation metrics* (S3). To evaluate the performance of a retrieval algorithm, some metrics are necessary. Typical metrics involve, but are not limited to, recall, precision, mean reciprocal rank and F-measure.

### 2.2. *Related techniques*

To efficiently accelerate or support a scalable audio-based music search, some works apply index-based techniques [1, 4, 6, 22, 24]. In [1] a composite feature tree (semantic features, such as timbre, rhythm, pitch, e.g.) was proposed to facilitate KNN search. Principle Component Analysis (PCA) is used to transform the extracted feature sequence into a new space sorted by the importance of acoustic features. In [4] the extracted features (Discrete Fourier Transform) are grouped by Minimum Bounding Rectangles (MBR) and indexed by a spatial access method. The false alarm resolution approach was also used to avoid costly operations.

The original concept of LSH [11] is a simple idea, which is an index-based data organization structure proposed to find all documents similar to a query in Euclidean space. Specifically, feature vectors are extracted from audio documents and regarded as being similar to one another if they are mapped to the same hash values. Yang used random sub-sets of spectral feature, STFT, to calculate hash values for the parallel LSH hash instances in [6]. With a query as input, the relevant features are matched from hash tables. To resolve bucket conflicts, a Hough transformation is performed on these matching pairs to detect the similarity between the query and each reference song by the linearity filtering.

Many variants of LSH have been proposed to effectively reduce the space requirement and shorten the response time, which also solves the Approximate Nearest Neighbors problem in a Euclidean space. $E^2$LSH [13] enhances LSH to make it more efficient for the retrieval with very high dimensional features. It performs locality sensitive dimension reduction to get the projection of the feature in different low-dimension sub-spaces. An application of $E^2$LSH in large-scale music retrieval is evaluated in [22]. Shingles are created by concatenating consecutive frames and used as high-dimensional features. Then $E^2$LSH is adopted to compare a query with references. In our previous work [24] LSH and $E^2$LSH are used to accelerate sequence comparison in music retrieval. Panigraphy [14] considered the distance $d(p, q)$ between the query $q$ and its nearest neighbor $p$ in the query stage of the LSH scheme. By selecting a random point $p'$ at a distance $d(p, q)$ from $q$ and checking the bucket that $p'$ is hashed to, the entropy-based LSH scheme ensures that all the buckets which contain $p$ with a high probability are probed. An improvement of this scheme by multi-probe was proposed in [9], where minor adjustment of the integer hash values are conducted to find the buckets that may contain the point $p$.

### 2.3. *Summary of LSH-based similarity retrieval*

In LSH-based approximation techniques, intrinsically, two points deserve attention:

(1) *A group of locality sensitive mapping functions are constructed for each hash instance.* The original features are converted into new features. However, the similarity is always kept (two original features with a short distance still have a short distance after locality sensitive mapping). A group of hash values is

assigned to each bucket to make the features in the same bucket as similar as possible (to improve precision and avoid false results).

(2) *All hash instances are parallel and independent.* By a single hash instance only part of the similar features can be found, resulting in a relatively low recall. Parallel and independent hash instances can guarantee that similar features collide in at least one of the hash instances with a high probability. The union of the results from all hash instances would improve the recall.

### 2.4. *Comparison with existing works*

We aim to develop a scalable audio-based MIR system that can run over the large real-world datasets, return retrieval results ranked by similarity within an acceptable time, and compare multiple retrieval algorithms by evaluation metrics.

To this end, we consider both the semantic summarization of audio documents and the hash-based approximate retrieval for the purpose of reducing retrieval time and improving search quality. In contrast with existing works, through a new principle of similarity-invariance, a concise audio feature, FU, is generated based on multivariable regression. Associated with the FU, variants of LSH (ELSM and Soft-LSH) are proposed. Different from the conventional LSH schemes, soft hash values are exploited to accurately locate the searching region and improve the search quality without requiring many hash instances. We care about possible difference between perceptually similar audio documents and map the features into a continuous hash space (soft mapping). The neighborhood determined by the query will intersect buckets that possibly contain similar documents (see Sec. 3.3). In this way the number of hash instances required in normal hashing schemes can be greatly reduced. In comparison with the exhaustive KNN and regular LSH schemes, our algorithms achieve much better search quality than that of LSH with the same number of hash instances and almost the same as that of an exhaustive KNN search but with much less search time. Our techniques are extended and summarized as a computer-assisted evaluation tool to help researchers to explore different retrieval schemes and evaluate them with different metrics (e.g. recall, precision).

## 3. The Proposed Approaches

In this section we reveal a principle of melody-based similarity-invariance, by which we can summarize long audio feature sequences and generate a single compact and semantic representation, FU. FU is a general concept of concatenating feature vectors. It can be implemented in different ways. We will discuss how to implement a special FU by concatenating frequently used spectral features. Instead of traditional hard (discrete) hash values we adopt soft (continuous) hash values and use exact locality sensitive mapping, which helps to locate the searching range more accurately. On this basis, two schemes associated with FU are proposed to solve the scalability of an audio-based music search. ELSM is similar to the first half of $E^2$LSH. However, in the proposed ELSM the sub-feature vector is not calculated

to obtain hash values. They are directly used as new features to perform exhaustive KNN searching. The ELSM feature is logically related to the number of hash instances, but is not mapped to the integer hash values. We also propose a SoftLSH algorithm as a variant of LSH. When the FUs of two audio tracks are similar to each other, their non-quantized hash values will also be similar to each other. Our SoftLSH scheme utilizes the non-quantified ELSM feature to accurately locate all the buckets that intersect the neighborhood determined by the query.

### 3.1. *An overview of the proposed schemes*

Our algorithms proceed in two main stages that can be briefly described as follows.

1) Given a collection of songs $R = \{r_{i,j} : r_{i,j} \in R_i,\ 1 \leq i \leq |R|, 1 \leq j \leq |R_i|\}$, ($r_{i,j}$ is the $j$th spectral feature of the $i$th song $R_i$), the feature sequence $\{r_{i,j}\}$ of the $i$th song $R_i$ is summarized to $V_i$, an $n$-dimension feature vector in the Euclidean space ($V_i \in \Re^n$). The summarized feature $V_i$ instead of the feature sequence $\{r_{i,j}\}$ is utilized in the retrieval stage (see Sec. 3.2). To further accelerate the retrieval speed, hash-based indexing is also adopted. Each hash function $h_k(\cdot)$ maps $V_i$ to a single value. Then a group of $N$ independent hash functions $h_1(\cdot),\ h_2(\cdot), \ldots, h_N(\cdot)$ generate a vector of hash values $H(V_i) = [h_1(V_i), h_2(V_i), \ldots, h_N(V_i)]^T$. Inside a hash instance each hash vector is assigned to a bucket and two summarized features with the same hash vector fall into the same bucket. $L$ parallel hash instances are constructed to support a high recall. By utilizing soft hash values, the number of hash instances can be greatly reduced meanwhile the search quality is retained (see Sec. 3.3).

2) Given a query song $Q$ (with the summarized feature $V_q$) and a similarity function $\rho(\cdot, \cdot)$, we would like to compute the similarity degree $\rho(V_q, V_i)$ between the query $Q$ and each of the songs $R_i$ in the database. They are similar if $\rho(V_q, V_i)$ is above a predefined similarity threshold. The similarity between two feature vectors can be computed by several similarity measures, such as Mahalanobis distance [30], Euclidean distance, etc. Using Mahalanobis distance requires that the covariance matrix be known in advance. However, as the dimension of features become very large (218 in this paper), the estimation of this covariance matrix may be inaccurate and degrades the performance instead. In this paper correlation among variables of the feature is not very strong (the remaining correlation can be removed with an extra preprocessing stage by using PCA, etc.). Therefore the distance between $V_q$ and $V_i$ is defined as $d(V_q, V_i) = \|V_q - V_i\|^2$. It also coincides with the fact that LSH is often discussed in the Euclidean space [11].

### 3.2. *Spectrum-based feature union*

Audio documents can be described by time-varying feature sequences. Computing the distance between audio documents (matching audio documents) is an important task in implementing query-by-audio music search. DP [16, 18] can be used in matching two audio feature sequences and is essentially an exhaustive search

Table 1. A list of popular audio features.

| Feature sequence representation |
|---|
| STFT [4][6] |
| MFCC [7][8] |
| Timbre, Rhythm and Pitch [1] |
| Chroma [9][18] |
| Pitch [5] [19] |

approach (which offers high accuracy). However it lacks scalability and results in a lower retrieval speed as the database gets larger. To speed up the matching of audio documents and enable a scalable search, compressed features are extracted from the audio data [1, 23]. Unfortunately, the semantic feature (high-level) used in [1, 23] was mainly proposed to summarize audio feature sequences for musical genre classification [17]. These semantic feature summarizations cannot effectively represent melody-based lower-level music information.

We list the most popular audio-based music search techniques in Table 1 in terms of audio sequences representation. Researchers have adopted different features according to their application biases. For example, Pitch [5] is extracted from the sung melody to generate a note sequence while semitone-based Chroma [18] is used to tolerate differences in instruments and general musical styles.

Different features represent different aspects of audio signals and were proposed for different purposes. Table 1 encourages us to generate a single description from these features to represent the characteristics of an audio sequence as comprehensively and effectively as possible. It is expected that with the single newly generated feature representing an audio sequence, the heavy computation of feature sequence comparison can be avoided and a query-by-audio music search can be applied in a large database. In the following we focus on concatenating frequently used spectral features and this special FU is called spectrum union (SU).

#### 3.2.1. *Similarity-invariance of summarization*

Two questions occur in the summarizing stage: (1) how to summarize the high dimensional features? (2) how to guarantee that a summarized feature reflects the melody information? As for the first question, there are several summarizing methods such as calculating the mean and standard deviation of all the features, PCA, and so on. As for the second question the summarizing procedure should exhibit the key characteristics of similarity-invariance, that is, similar melodies lead to similar summaries, non-similar melodies lead to non-similar summaries.

To satisfy the above summarization requirements, a basic melody-based summarization principle is considered as follows. $R_i = \{r_{i,l} : l = 1, 2, \ldots, |R_i|\}$ is the feature sequence of the $i$th reference song. It is summarized to $V_i$. The sequence similarity between $i$th song $R_i$ and $j$th song $R_j$ is $\varphi(R_i, R_j)$. The similarity between

the $i$th and $j$th feature summary is $\psi(V_i, V_j)$. Here $\varphi(.)$ and $\psi(.)$ are similarity functions. Similarity thresholds $\theta$ or $\theta'$ would state how close any two feature sequences or summarized features are. With a good summarization we could expect that

$$\varphi(R_i, R_j) > \theta \Leftrightarrow \psi(V_i, V_j) > \theta'. \tag{1}$$

In this sense the summarization is similarity-invariant.

### 3.2.2. *Regression model*

A single feature cannot summarize well a song and multiple features can be combined to represent a song. We focus on spectrum similarity between two audio sequences and consider the combination of frequently used spectral features such as MFCC, Mel-Magnitude spectrum, Chroma and Pitch. They are very typical audio features in melody-based MIR systems. As for feature extraction (the interested reader is referred to [5, 7, 18, 20] for a detailed description), each audio document is divided into frames and from each frame MFCC (13-dimension), Mel-Magnitude spectrum (40-dimension), Chroma (12-dimension) and Pitch (1-dimension) are extracted. Mean and standard deviation (std) of the sequence of MFCC, Mel-Magnitude spectrum and Chroma are calculated, respectively. A histogram of Pitch is also calculated. As shown in Table 2, the per-feature summarization leads to 7 feature vectors. These feature vectors play different roles in different aspects of music content representations. By assigning suitable weights to each feature vector and concatenating them together, the concatenated spectral feature, SU, can better represent an audio signal than an individual feature vector.

Training of these weights can be done by different methods. For the simplicity of associating the sequence distance with the SU distance, we decide to use multivariable regression to determine the weights. The goal of our approach is to apply linear regression models to investigate the correlation.

In the model we use $K = 7$ feature vectors listed in Table 2 to generate an SU. Consider a training database composed of $M$ pairs of songs $< R_{m1}, R_{m2} >$, $m = 1, 2, \ldots, M$, which contain both similar pairs and non-similar pairs. Let the feature vector of $mi$th song be $v_{mi,1}, v_{mi,2}, \ldots, v_{mi,K}$ $(i = 1, 2)$. With different weights $\alpha_k$ assigned to each feature vector, the total SU is

$$V_{mi} = [\alpha_1 v_{mi,1}, \alpha_2 v_{mi,2}, \ldots, \alpha_K v_{mi,K}]^T, \tag{2}$$

Table 2. Dimensions of features.

| Feature sets | Functions | Dimensions |
|---|---|---|
| MFCC | Mean and Std | $13 + 13$ |
| Mel-Magnitude | Mean and Std | $40 + 40$ |
| Chroma | Mean and Std | $12 + 12$ |
| Pitch | Histogram | 88 |
| Total | | 218 |

where $(\cdot)^T$ stands for transposition. The distance between two SUs $V_{m1}$ and $V_{m2}$ is

$$d(V_{m1}, V_{m2}) = d(\sum_{k=1}^{K} \alpha_k v_{m1,k}, \sum_{k=1}^{K} \alpha_k v_{m2,k}) = \sum_{k=1}^{K} \alpha_k^2 d(v_{m1,k}, v_{m2,k}). \quad (3)$$

To determine the weights in Eq. (2), we apply a multi-variable regression process. From $M$ pairs of songs we obtain the sequences of features and calculate $M$ sequence distances $d_{DP}(R_{m1}, R_{m2})$ via DP. The feature sequence may consist of MFCC, pitch and so on. Since Chroma is capable of representing distinct semitones of music octaves [18], it is adopted as the feature in the calculation of sequence distance.

We will choose the weights in Eq. (2) so that $d(V_{m1}, V_{m2})$, the SU distance, is as near to $d_{DP}(R_{m1}, R_{m2})$ (the sequence distance) as possible, that is, we hope the melody information is contained in the SU. After we determine the distance between the pairs of training data, we get an $M*K$ matrix $D_V$ and an $M$-dimension column vector $D_{DP}$. The $m$th row of $D_V$ has $K$ distance values, $d(v_{m1,k}, v_{m2,k}), k = 1, 2, \ldots, K$, calculated from individual feature vector and the $m$th element of $D_{DP}$ is the distance $d_{DP}(R_{m1}, R_{m2})$ between the two feature (Chroma) sequences. Let

$$A = [\alpha_1^2, \alpha_2^2, \ldots, \alpha_K^2]^T, \quad (4)$$

according to Eq. (3) $D_V$, $A$ and $D_{DP}$ satisfy the following equation

$$D_V \cdot A = D_{DP}. \quad (5)$$

Solving this equation we get

$$A = (D_V^T D_V)^{-1} D_V^T D_{DP}, \quad (6)$$

and obtain the weight $\alpha_k$. We are only interested in the absolute value of $\alpha_k$. The SU defined in Eq. (2) is a special implementation of FU.

### 3.3. *ELSM and SoftLSH*

Almost all hash schemes, including LSH, use hard (discrete) integer hash values. In LSH an SU $V_i$ is locality-sensitively mapped to $H(V_i)$, which is further quantized to integer hash value $\overline{H}(V_i) = round(H(V_i))$($round(x)$ is the nearest integer of $x$). Two SUs ($V_i$ and $V_j$) with a short distance ($d(V_i, V_j)$) have the same integer hash value ($\overline{H}(V_i) = \overline{H}(V_j)$) with a high probability. By assigning integer hash values to buckets, the songs located in the same bucket as the query can be found quickly.

However even if two similar SUs $V_i$ and $V_j$ have a short distance $d(V_i, V_j)$, it is not always guaranteed that they have the same hash values due to the mapping and quantization errors. When a vector composed of $N$ hash values instead of a single hash value is used to locate a bucket the precision can be improved and the effect of quantization error gets more obvious. To find a similar song from the database with a specific query, multiple parallel and independent hash instances are necessary, which in turn takes more time and requires more space. Our solution to the above problem is to exploit the continuous non-quantized hash values with two schemes, Exact Locality Sensitive Mapping (ELSM) and SoftLSH.

3.3.1. *Concept of soft mapping*

We assume the search-by-hash system has $L$ parallel hash instances and each hash instance has a group of $N$ locality sensitive mapping functions. In the $m$th hash instance the function group is $H_m = \{h_{m1}, h_{m2}, \ldots, h_{mN}\}$. Its $k$th function $h_{mk}(\cdot)$ maps an SU feature $V$ to a continuous non-quantized hash value $h_{mk}(V)$. After mapping, the hash vector in the $m$th hash instance corresponding to $V$ is $H_m(V) = \{h_{m1}(V), h_{m2}(V), \ldots, h_{mN}(V)\}$.

Consider the $k$th dimension of the hash vectors $H_m(V_i)$ and $H_m(V_j)$ corresponding to $V_i$ and $V_j$ respectively. By the first order approximation of Taylor series, the difference between $h_{mk}(V_i)$ and $h_{mk}(V_j)$ is

$$h_{mk}(V_i) - h_{mk}(V_j) \approx h'_{mk}(V_j)(V_i - V_j). \tag{7}$$

When $V_i$ and $V_j$ are similar to each other, they have a short distance $d(V_i, V_j)$. Then according to Eq. (7) $h_{mk}(V_i)$ and $h_{mk}(V_j)$ are close to each other and so is the vector $H_m(V_i)$ and $H_m(V_j)$.

At the quantization stage the hash space is divided into non-overlapping squares, where $H_m(V)$ is quantized to a set of $N$ integer hash values $\overline{H}_m(V)$, the center of the squares. Two SUs falling in the same square have the same integer hash values. But this quantization cannot well retain the distance between two SUs. Figure 2 shows an example where $N$ equals 2. $d(H_m(V_i), H_m(V_j))$ is less than the allowed error, but neither of the integer hash values of the two SUs is the same. $H_m(V_i)$ is quantized to $\overline{H}_m(V_i)$ =(2,3) while $H_m(V_j)$ is quantized to $\overline{H}_m(V_j)$ =(1,2). By careful observation we can learn that the quantization error usually happens when both $H_m(V_i)$ and $H_m(V_j)$ are near the edge of the squares. Even a little error near the edge will result in an error up to $N$ between two integer hash sets $\overline{H}_m(V_i)$ and $\overline{H}_m(V_j)$.

In Fig. 2 the SU feature $V_i$ is a neighbor of $V_j$ and the hash value $H_m(V_i)$ is located in the neighborhood $C(H_m(V_j), r)$, a ball centered at $H_m(V_j)$ with a radius $r$. But $H_m(V_i)$ and $H_m(V_j)$ are located in different squares and result in a big

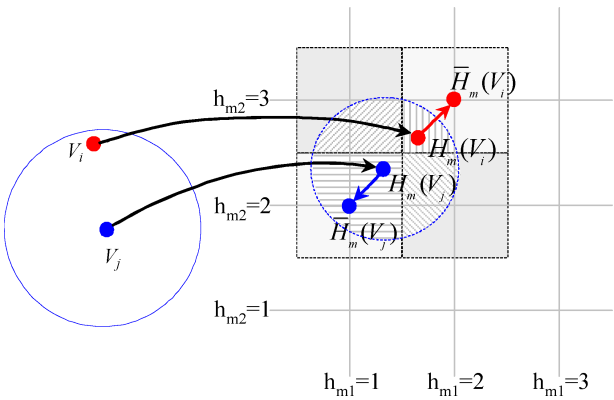


Fig. 2. Concept of soft mapping.

distance after quantization. Here each square corresponds to a bucket. It is obvious that $C(H_m(V_j), r)$ intersects several quantization squares simultaneously, including the square where $H_m(V_i)$ lies. Then with $V_j$ as query and $C(H_m(V_j), r)$ calculated in advance, the buckets that possibly hold $V_i$ can be easily found. From all the features in these buckets, the ones located in $C(H_m(V_j), r)$ are taken as the candidates. Then the KNN algorithm is applied to the candidates to find the features that are actually similar to $V_j$. Of course $V_i$ will be one of the nearest neighbors.

### 3.3.2. *Query with ELSM*

Our first solution to the quantization problem is to utilize the ELSM feature together with KNN instead of assigning SUs to buckets. Each song in the database is processed as in Fig. 3. Its SU feature $V$ is obtained by the regression discussed in Sec. 3.2.2.

The $m$th hash instance has its own sets of $N$ hash functions $h_{mk}(V) = (a_{mk} \cdot V + b_{mk})/w_{mk}$ $(1 \leq k \leq N)$, which is determined by $a_{mk}$ and $b_{mk}$, the random variables, and $w_{mk}$, the quantization interval. By $w_{mk}$, standard deviations of soft hash values in different hash instances are made almost equal and the distribution of hash vectors roughly spans a square in the Euclidean space. The hash set for the SU feature $V$ is $H_m(V) = [h_{m1}(V), h_{m2}(V), \ldots, h_{mN}(V)]$ in the $m$th hash instance. When there are $L$ parallel hash instances, the hash vectors generated from the SU feature $V$ for all hash instances are $H(V) = [H_1(V), H_2(V), \ldots, H_L(V)]$, which has $N \cdot L$ dimensions. Since the mapping function $h_{mk}(\cdot)$ is locality sensitive, the new hash vector $H(V)$ contains most of the information embedded in $V$.

$H(V)$ can serve as a feature (ELSM) and be used together with KNN. With the soft mapping value, it will not suffer quantization information loss. This scheme can also be regarded as an ideal hash, where each bucket only contains the ELSM features that are very similar to each other. When a query comes, its ELSM feature locates the bucket that contains all the similar songs (which are the same as exhaustive KNN). In such ideal cases the search accuracy is the same as where ELSM is utilized together with the exhaustive KNN. Usually $N \cdot L$ is much smaller than the

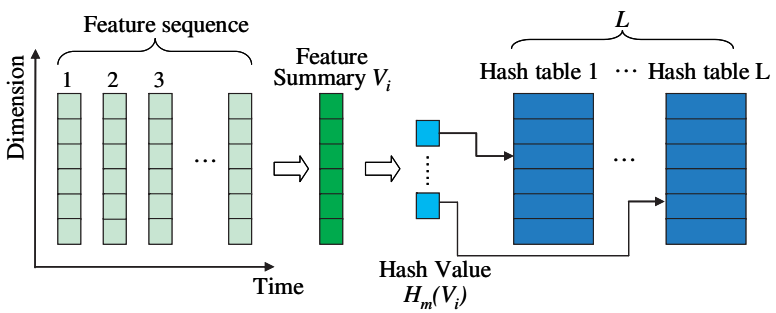


Fig. 3. Feature organization in the database.

dimension of SU. Though it cannot provide a response as fast as SoftLSH, it is still much faster than utilization of the SU feature directly. Meanwhile its search accuracy up bounds that of SoftLSH. With ELSM as the feature in the KNN search, we can verify the effectiveness of locality sensitive mapping.

#### 3.3.3. *Query with SoftLSH*

Our second solution to the quantization problem is to exploit the non-quantized hash values (SoftLSH) to locate in a hash instance all the buckets that possibly hold features similar to the query.

For the $i$th song with its SU feature $V_i$, in the $m$th hash instance its song number $i$ is stored in the bucket $\overline{H}_m(V_i)$. Its soft hash values corresponding to all hash instances, $H(V_i)$, are stored together in a separate buffer and utilized as the ELSM feature. The residual part $H_m(V_i)$-$\overline{H}_m(V_i)$ reflects the uncertainty. This part is usually neglected in all LSH indexing schemes. Fully exploiting this part facilitates the accurate locating of the buckets that possibly contain the similar features.

In times of retrieval the SU of the query, $V_q$, is calculated. Its ELSM feature, $H(V_q) = [H_1(V_q), H_2(V_q), \ldots, H_L(V_q)]$, is also calculated. In the $m$th hash instance the features similar to $V_q$ will be located in the buckets that intersect the neighborhood $C(H_m(V_q), r)$. Due to the quantization effect the buckets are squares. Any vertex of a bucket lying in the neighborhood will result in its intersection with the neighborhood. An example is shown in Fig. 2 where $N$=2 and $V_q = V_j$.

Buckets in a hash instance are centered at a vector of integer hash values. Their vertexes are the center plus or minus 0.5. $\overline{H}_m(V_q)$, the integer part of $H_m(V_q)$, indicates the bucket that most possibly contains similar features. Buckets near $\overline{H}_m(V_q)$ also possibly contain features similar to the query. Vertexes of these buckets, $\overline{H}_m(V_q) + (j_1 \pm 0.5, \ldots, j_N \pm 0.5)$, are examined. For the vector $(j_1, \ldots, j_N)$ where the vertexes falling in $C(H_m(V_q), r)$, $\overline{H}_m(V_q) + (j_1, \ldots, j_L)$ are the centers of the buckets that possibly contain the similar features. Features falling in these buckets are examined by KNN with the ELSM feature.

## 4. Experimental Evaluation

In Sec. 4.1 we introduce our experimental setup, including datasets, benchmark, tasks and evaluation metrics. In Sec. 4.2 we evaluate the regression scheme and find a group of optimal weights. In Sec. 4.3 and Sec. 4.4 we present the evaluation results of the searching schemes by two tasks and give the corresponding analysis, respectively.

### 4.1. *Experiment setup*

**Datasets.** Our music collection includes 5435 tracks that fall into five non-overlapping datasets, as summarized in Table 3. In this table Trains80 is collected

Table 3. Datasets description.

| | Train set | Evaluation set | | | |
|---|---|---|---|---|---|
| Dataset | Trains80 | Covers79 | ISMIR | RADIO | JPOP |
| #Tracks | 160 | 1072 | 1458 | 1431 | 1314 |
| Size | 0.211GB | 1.42GB | 1.92GB | 1.89GB | 1.73GB |

from our personal collections, and http://www.yyfc.com (a non-commercial amusement website where users can sing her/his favorite songs, make records online and share them with friends). It consists of 80 pairs of tracks. 40 pairs each contain two versions of the same song while each of the other 40 pairs contains different songs. These 160 tracks are used to train the weights of the regression model proposed in Sec. 3.2.2. Covers79 is also collected from http://www.yyfc.com and consists of 79 popular Chinese songs each represented in different versions (the same song sung by different people). Each song has 13.5 versions on average resulting in a total of 1072 audio tracks. Figure 4 shows the distribution of these tracks in detail, the histogram of the songs in terms of the number of their covers. For example, there are 12 songs each having 11 covers.

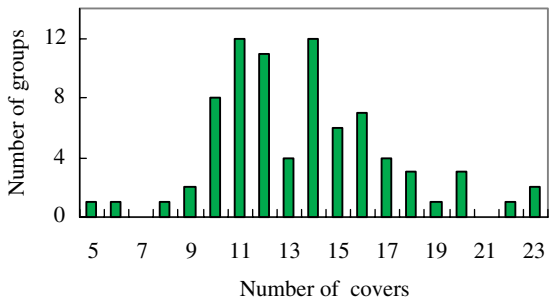


Fig. 4. Distribution of audio cover tracks in Covers79.

RADIO is collected from http://www.shoutcast.com and ISMIR is taken from http://ismir2004.ismir.net/genre_contest/index.htm. JPOP (Japanese popular songs) is from our personal collections. Covers79, ISMIR, RADIO and JPOP are used in the evaluation and altogether there are 5275 tracks and the last three datasets are used as background audio files of simulation. Each track is 30s long in mono-channel wave format, 16 bit per sample and the sampling rate is 22.05 KHz. The audio data is normalized and then divided into overlapped frames. Each frame contains 1024 samples and the adjacent frames have 50% overlap. Each frame is weighted by a hamming window and further appended with 1024 zeros. A 2048-point FFT is used to calculate the STFT from which the instantaneous frequencies are extracted and chroma is calculated. From the power spectrum pitch, MFCC and Mel-magnitude are calculated. Then the features in Table 2 and SU in Eq. (2) are calculated.

**Benchmark.** The ground truth is set up according to human perception. We have listened to all the tracks and manually labeled them so that retrieval results of our algorithms correspond to human perception to support practical application. Trains80 and Covers79 datasets are divided into groups according to their verses (the main theme represented by the song lyrics) to judge whether tracks belong to the same group or not (one group represents one song and different versions of one song are members of this group). The 30 s segments in these two datasets are extracted from verse sections of songs.

**Tasks.** By two tasks we demonstrate the performance of KNN, ELSM, LSH and SoftLSH and their potential applications in query-by-audio music searching. All schemes are based on the SU feature unless otherwise specified. KNN is an exhaustive search while LSH represents quantization into hash buckets. KNN achieves highest recall and precision (upper bound). LSH has the least retrieval time (low bound). We hope our algorithms would approach KNN in recall and precision while retaining almost the same retrieval time as LSH. Task 1 (Sec. 4.3) is mainly to solve the problem of cover songs detection or near duplicate detection of audio files similar to [2, 16, 18]. Task 2 (Sec. 4.4) is to solve the problem of query-by-example/humming/singing similar to [1, 6, 21, 22].

**Evaluation metrics.** We select recall, precision, average precision, F-measure and Mean Reciprocal Rank (MRR) as the evaluation metrics to compare four music searching schemes over our datasets. When evaluating SU and Task 1, corresponding to a single query there are multiple relevant items in the database. In such cases recall, precision and F-measure are effective metrics [25]. On the other hand, in Task 2, there is merely a single relevant item in the database corresponding to the query. As a result MRR(1) is the best choice [26]. To maintain notional consistency throughout experimental evaluation, an overview of evaluation metrics is given here. Given a query $q$ as musical audio input, $S_q$ is the set of its relevant items in the database. As a response to the query, the system outputs the retrieved set $K_q$ in a ranked list. Unless otherwise specified $|K_q|$ equals $|S_q|$ in the following. Recall, precision, average precision, F-measure and MRR(1) are defined by the following formulas respectively:

$$\text{recall} = \frac{|S_q \cap K_q|}{|S_q|}, \tag{8}$$

$$\text{precision} = \frac{|S_q \cap K_q|}{|K_q|}, \tag{9}$$

$$\text{average precision} = \frac{1}{|S_q \cap K_q|} \sum_{r=1}^{|S_q \cap K_q|} P_q(r), \tag{10}$$

$$\text{F-measure} = \frac{2 \cdot \text{recall} \cdot \text{precision}}{\text{precision} + \text{recall}}, \tag{11}$$

$$\mathrm{MRR}(1) = \frac{1}{N_q} \sum_{i=1}^{N_q} \frac{1}{\mathrm{rank}_i(1)}, \tag{12}$$

where $|S_q \cap K_q|$ is the number of relevant items found in the ranked output list, $P_q(r)$ is the precision achieved when the $r$th relevant document is retrieved, $N_q$ is the number of the total queries and $\mathrm{rank}_i(1)$ corresponds to the order of the first relevant item in the ranked list.

The experiments were run on a PC with an Intel Core2 CPU (2GHz) and 1GB DRAM under Microsoft Windows XP Professional OS.

### 4.2. *Optimal weights and SU superiority*

First we train the weights in the SU feature by using the Trains80 dataset as the ground truth. We selected 40 pairs of similar songs (each pair includes two versions of the same song) and 40 pairs of non-similar songs (each pair includes two different songs). Based on the idea of multivariable regression proposed in Sec. 3.2.2, we get a group of weights (normalized so that their squared sum is 1) for the seven feature sets, MFCC-mean (0.1900), MFCC-std (0.1489), Mel-magnitude-mean (0.3416), Mel-magnitudes-std (0.2729), Chroma-mean (0.5396), Chroma-std (0.3817), Pitch-histogram (0.5601). This SU is used in all following experiments.

Figure 5 compares the recall and average precision achieved by the individual features based on an exhaustive search of the Covers79 dataset. (The simulation setup is the same as that for Fig. 7 in Sec. 4.3, 1072 query tracks against the total evaluation set with 5275 tracks.) For a comparison, the result of concatenating feature vectors with equal weights is also given. It is only a little better than other competitive feature sets but much inferior to SU. On the other hand, with an optimal weight, SU has the best performance and can represent human perception more effectively. However, SU's recall is only 0.682. Note that a recall of 1 is hard to

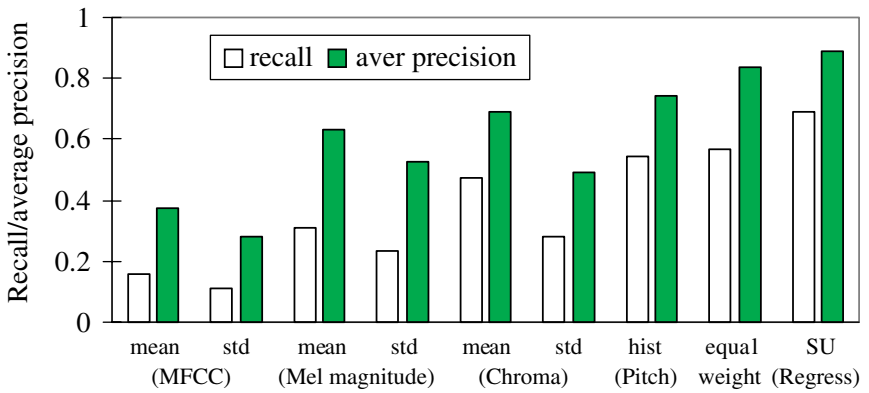


Fig. 5. Recall and average precision achieved by different features.

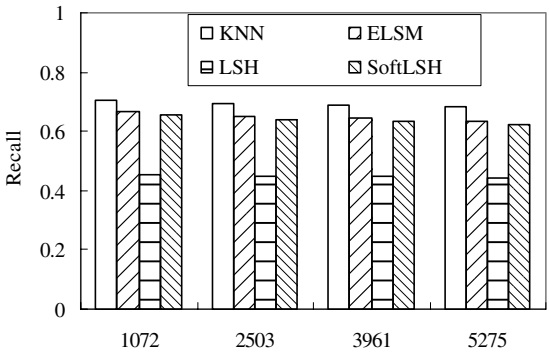


Fig. 6. Recall achieved at different database sizes.

achieve in a music audio search with compact audio feature representations even by the exhaustive search due to the following factor: some perceptually similar tracks have quite different features and result in quite large distances, which greatly affects the retrieval of these tracks.

It is not easy to find a good acoustic feature summarization since perceptually similar tracks (relevant tracks) sometimes have relative large distances in Euclidean space. To check whether our musical audio summarization can effectively represent music audio information in a complicated music background, we run SU summarization via four search schemes (KNN, ELSM, LSH, and SoftLSH) over datasets with different sizes. Figure 6 shows the effect of increasing datasets size. The four databases contain Cover79 (1072), Cover79 & RADIO ($1072 + 1431 = 2503$), Cover79 & RADIO & ISMIR ($1072 + 1431 + 1458 = 3961$), Cover79 & RADIO & ISMIR &JPOP ($1072 + 1431 + 1458 + 1314 = 5275$), respectively. It is obvious that the increase of datasets size has very little effect on the recall, which in turn confirms that the SU feature can effectively distinguish the (newly added) non-similar songs.

### 4.3. *Task 1 evaluation*

**Online query, one-query against multi-object.** Dataset Covers79 is embedded in the evaluation set with 5275 tracks (Covers79+ISMIR+RADIO+JPOP). The whole evaluation set has a broad range of music genres (classical, electronic, metal, rock, world, etc). With each track in the Covers79 used as a query we calculate the ranked tracks similar to the query. Each query $q$ chosen from Covers79 has its relevant set size $|S_q|$ (perceptually similar tracks), which is determined according to Fig. 4. The average size of a query's relevant set is 12.5 (on average each song in Covers79 has 13.5 covers. When one cover is used as a query, the remaining covers are in the database). The total number of relevant tracks can be calculated from Fig. 4 (a theoretical maximum is 14452). Recall, average precision and F-measure are used as evaluation measures in this task.

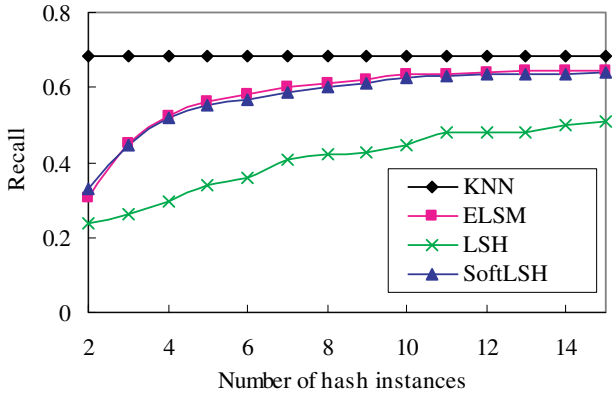


Fig. 7. Recall under different numbers of hash instances $|K_q| = |S_q|$.

Figure 7 is the recall of the four schemes in question. Given the same feature representation KNN is the golden standard and always performs best. LSH is always inferior to the newly proposed schemes. When there are very few hash instances, the ELSM feature has a low dimension and cannot well represent SU. As a result the performance of ELSM and SoftLSH is poor compared to KNN. As the number of hash instances increases from 2 to 6 the recall in both ELSM and SoftLSH increases correspondingly and the curves of ELSM and SoftLSH approach the KNN performance. The recall of SoftLSH is quite close to ELSM. This reflects that search in the neighborhood of the query's hash values has almost the same performance as an exhaustive search. The gap between KNN and ELSM/SoftLSH also decreases as more hash instances are used. The recall, however, does not increase linearly. The slope of recall approaches 0 and a further increase in hash instances results in diminishing returns. When the number of hash instances is greater than 10, the gap between ELSM/SoftLSH and KNN is almost constant, which means that the information loss due to utilizing a lower dimension feature cannot be salvaged by an increase in hash instances. When there are 10 hash instances, among the theoretically maximal 14452 relevant tracks KNN, ELSM, LSH and SoftLSH can identify 68.2%, 63.3%, 44.5% and 62.5% respectively.

Figure 8 shows the average retrieval time for each query. The exhaustive KNN always takes the longest time (0.0542s). Time consumption in the other three schemes gradually increases as the number of hash instances does. Average retrieval time of SoftLSH is about twice as much as LSH due to the search in multiple buckets that intersect the query's neighborhood. From Figs. 7–8 the tradeoff between accuracy and retrieval time indicates that 10 hash instances are a suitable choice. In such cases SoftLSH has a recall close to KNN and a retrieval time close to LSH. The additional time saved by LSH would result in a significant drop in accuracy. Therefore the number of hash instances is set to 10 in the following experiments of this task.

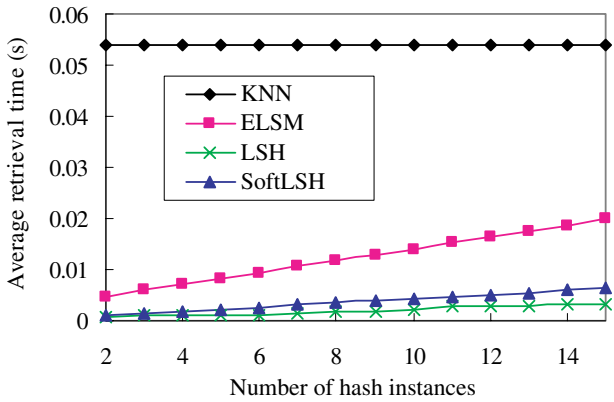


Fig. 8. Average retrieval time under different numbers of hash instances.

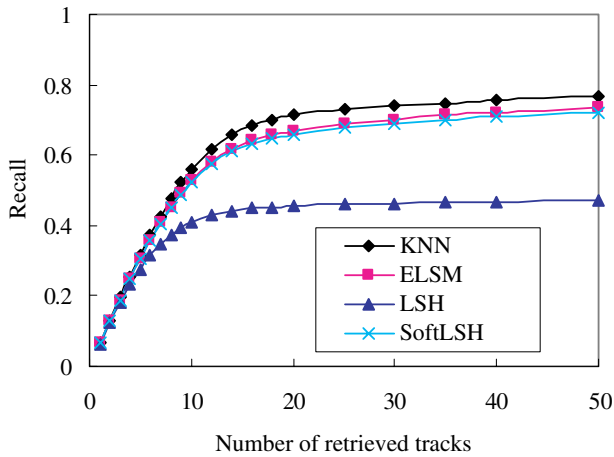


Fig. 9. Recall at different number of retrieved tracks.

Figures 9–11 demonstrate recall, precision and F-measure scores of the four schemes with respect to different numbers of retrieved tracks. It can be seen easily that LSH always performs worst. KNN performs slightly better than ELSM and SoftLSH at the cost of a much longer time to finish the search, as shown in Fig. 8. Here we address the fact that when the number of the retrieved tracks is less than that of the query's covers in the database ($|K_q| < |S_q|$), an increase of the retrieved tracks results in an almost linear increase of recall and a little decrease of precision. Therefore F-measure increases quickly. When the number of retrieved tracks gets greater than that of actual covers ($|K_q| > |S_q|$), the slopes of the recall curves in all schemes become steady in Fig. 9. Increasing the retrieved tracks always results in a decrease of precision in Fig. 10. In this experiment each query has an average number

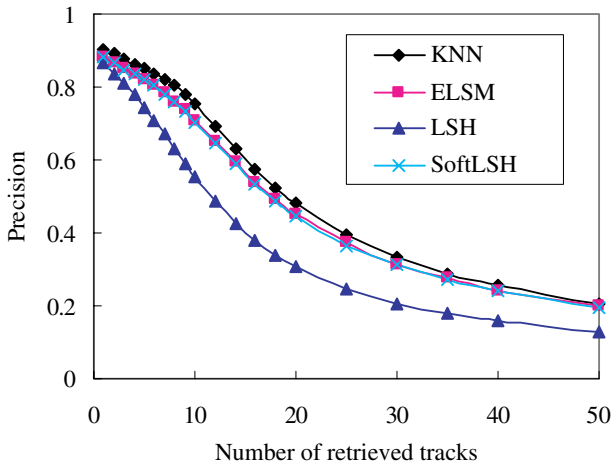


Fig. 10. Precision at different number of retrieved tracks.

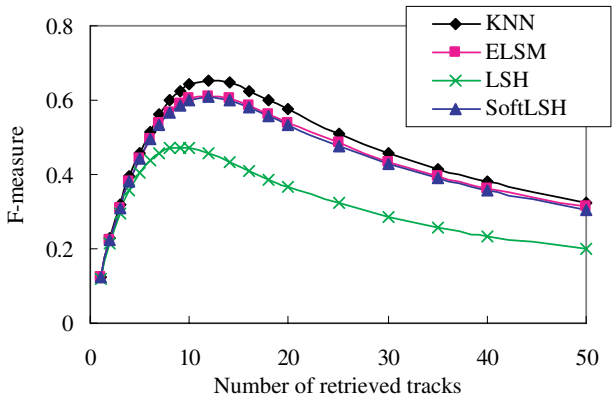


Fig. 11. F-measure at different numbers of retrieved tracks.

of 12.5 covers in the database. Coincidently in Fig. 11 the curves of KNN, ELSM and SoftLSH reach the maximal F-measure score when the number of retrieved tracks equals 12. This reflects that the SU feature is very effective in representing the similarity of tracks in each group. The tracks in the same group that genuinely have a short distance quickly appear in the returned list. Not-so-similar tracks have a relatively large distance and too long a return list only results in a very low precision and F-measure. It also confirms that SoftLSH is a good alternative to KNN.

Figure 12 is the precision-recall curve achieved by adjusting the number of system outputs. As expected at the same recall KNN always has the highest precision and LSH has the lowest. Some of the perceptually similar tracks have quite different

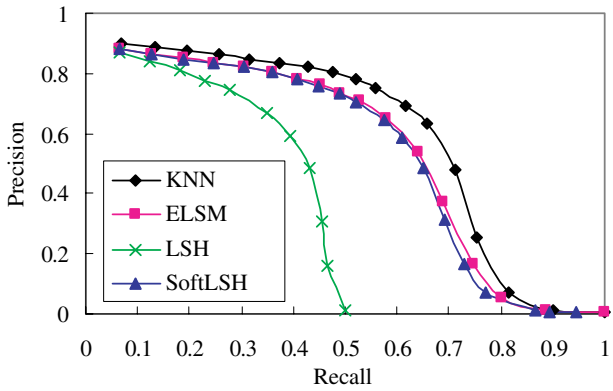


Fig. 12. Precision-recall curve (10 hash instances).

features and they can only be retrieved when KNN returns many tracks. Therefore the precision of KNN decreases quickly when recall is around 0.7. The performance of ELSM and SoftLSH approaches that of KNN, however, at the same precision they have an absolute loss of about 0.04 in recall compared with KNN due to utilizing a lower dimensional feature. Since the number of hash instances is fixed at 10 in the experiment, some of the tracks cannot be retrieved by the LSH scheme at all. Therefore the recall of LSH is upper bounded at 0.5 and a higher recall requires many more hash instances in LSH compared with SoftLSH.

### 4.4. *Task 2 evaluation*

**Batch queries, multi-query against one-object.** From each of the 79 songs in Covers79 one track is selected and altogether 79 tracks compose the ground truth. ISMIR, RADIO and JPOP are added as noise datasets (the whole database contains $79 + 1431 + 1458 + 1314 = 4282$ tracks). Remaining tracks in Covers79 ($1072 - 79 = 993$) are used as queries and the top 20 retrieved tracks are analyzed by default. Mean reciprocal rank (MRR(1)) and recall of the ground truth are calculated as the evaluation metrics.

Figure 13 shows the effect of different database sizes. The three databases respectively contain Cover79 (79), Cover79 & RADIO ($79 + 1431 = 1510$), Cover79 & RADIO & ISMIR ($79+1431+1458 = 2968$), Cover79 & RADIO & ISMIR & JPOP ($79 + 1431 + 1458 + 1314 = 4282$). It is obvious that the increase of database size has very little effect on the MRR, which confirms that the SU set can effectively distinguish the (newly added) non-similar tracks.

Figure 14 shows MRR(1) of the four schemes against different numbers of hash instances. When there are only two hash instances, ELSM and SoftLSH perform as poorly as LSH. ELSM and SoftLSH far outperform LSH when the number of hash instances is greater than 3. Their performances approach KNN when the number of hash instances is increased to 10. Table 4 shows the Top-1 and Top-20 recall

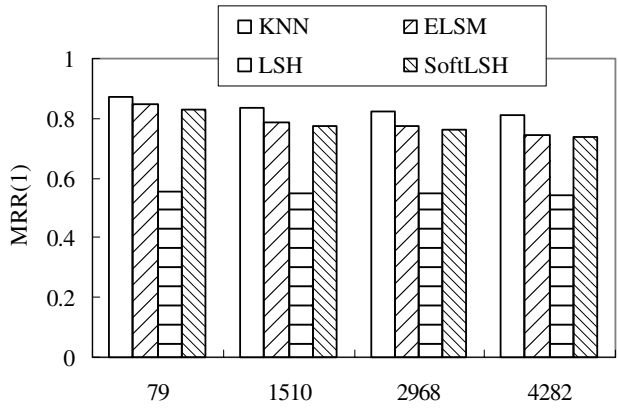


Fig. 13. MRR(1) at different database sizes.

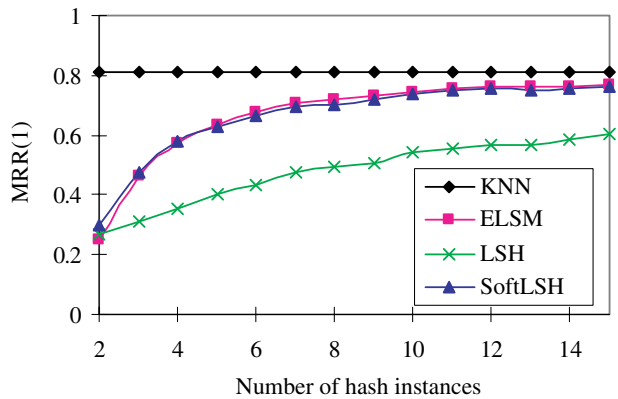


Fig. 14. MRR(1) at different number of hash instances.

Table 4. Top-1 and Top-20 recall.

| 10 Hash inst | KNN | ELSM | SoftLSH | LSH |
|---|---|---|---|---|
| Top-1 | 0.6354 | 0.5811 | 0.5771 | 0.4090 |
| Top-20 | 0.7543 | 0.7402 | 0.7271 | 0.4703 |
| Top20/Top1 | 1.1870 | 1.2738 | 1.2599 | 1.1474 |

achieved by the four schemes with 10 hash instances. We also list the ratio of recall achieved at Top-20 to that achieved at Top-1. This shows that ELSM and SoftLSH gain more benefit than KNN and LSH from the increase of the output, which means that in the two former schemes more non-similar songs appear in the first place while true covers appear behind. This is due to the following fact: KNN and LSH utilize the high-dimension SU feature while ELSM and SoftLSH use the lower-dimension ELSM feature. The dimension reduction loses some information, which affects the

distinguishing capability of ELSM and SoftLSH. However, this reduction is not so harmful when the dimension of ELSM is high enough, as reflected in Fig. 14.

### 4.5. *Screenshot of demonstration system*

Figure 15 illustrates our COSIN system for multi-version music searching, which is developed in C++. From the left side, the features and similarity searching methods can be selected. In this demo system, the query set consists of 79 groups with 1072 audio tracks (which is also embedded in the datasets). Any audio content in "Query List" can be freely selected as a query to retrieve over the datasets with 5275 audio tracks. Then the system gives an ordered list of the retrieval results in "Ranked List". The relevant audio tracks (which is decided by the number of group members) are reported automatically with the best similarity audio track as the first. Both queries and retrieval results can be played to confirm the correctness of searching. This system not only retrieves relevant audio tracks from the datasets, but also evaluates the performance of our approaches. In the current setting in Fig. 15 Feature "SFS" (Semantic Feature Summarization, i.e. SU), Similarity Searching "KNN" and the fourth member of the fourth group in "Query List" are set up respectively. In "Ranked List", the retrieval results are ordered and the relevant audio tracks are given in the list (amount "14" is also given) . The corresponding evaluation results are shown at the upper right side. We can easily see that with the fourth varsion of the song in group 4 as query input after searching over the datasets its best relevant audio track "Song04-Version15" appears in the first position.

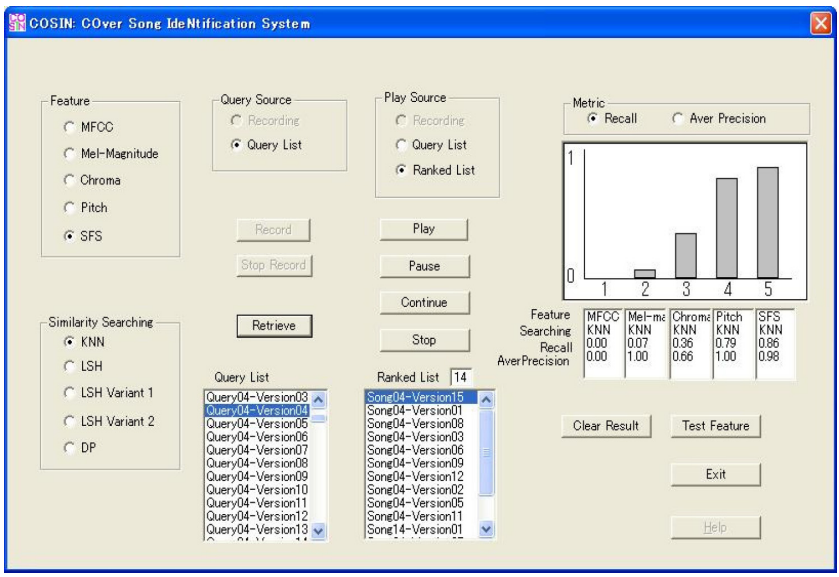


Fig. 15. Screenshot of COSIN DEMO.

## 5. Conclusion and Future Work

This work has focused on designing a scalable audio-based MIR system which enables real-time response to the audio-content query. To improve the scalability of audio-based MIR system we study both feature representations and indexing-based techniques and arrange our approaches in three steps. 1) The principle of similarity-invariance is applied to summarize audio feature sequences and utilized in training semantic audio representations based on supervised learning. The proposed FU (and its special implementation SU) better represents musical audio sequences. 2) Consider possible difference between perceptually similar audio documents and map the features into a continuous hash space (soft mapping). The neighborhood determined by the query will intersect buckets that possibly contain similar documents. 3) Extend our approaches to an MIR demonstration system that can implement multi-version audio tracks detection and retrieval over a large audio song collection.

We would like to consider the following different conditions to start our future work: (1) Music representation and approximate retrieval techniques are the key but independent aspects in achieving scalability of audio-based MIR. It is easy to make the proposed retrieval schemes applicable to other applications with a bit effort (especially video, bio-informatics, e.g.). We plan to refine our soft locality sensitive mapping and apply it to solve live video retrieval. (2) Information retrieval and search play an important role in MIR field, however, the human computer interaction aspects of MIR systems are often poorly designed. We further plan to explore an interactive MIR framework involves several aspects, such as, the relationship among system effectiveness, system efficiency, user's performance and satisfaction. (3) With the advent and popularity of commercial distribution of volumes of audio information, a great account of music content is shared and exchanged over P2P networks. Therefore it is necessary that distributed content-based search algorithms be developed for a P2P environment. This will also be a part of our future work.

## Acknowledgements

This work was supported by the Graduate Initiative Project of JSPS, a DoD grant from the KIMCOE Center of Excellence, and the Andrew W. Mellon Foundation.

## References

[1] B. Cui, J. Shen, G. Cong, H. Shen and C. Yu, Exploring composite acoustic features for efficient music similarity query, in *Proc. ACM MM'06*, 2006, pp. 634–642.

[2] M. Robine, P. Hanna, P. Ferraro and J. Allali, Adaptation of string matching algorithms for identification of near-duplicate music documents, in *Proc. SIGIR Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection*, 2007.

[3] J. F. Serrano and J. M. Inesta, Music motive extraction through hanson intervallic analysis, in *Proc. CIC'06*, 2006, pp. 154–160.

[4] I. Karydis, A. Nanopoulos, A. N. Papadopoulos and Y. Manolopoulos, Audio indexing for efficient music information retrieval, in *Proc. MMM'05*, 2005, pp. 22–29.

[5] W. H. Tsai, H. M. Yu and H. M. Wang, A query-by-example technique for retrieving cover versions of popular songs with similar melodies, in *Proc. ISMIR'05*, 2005, pp. 183–190.
[6] C. Yang, Efficient acoustic index for music retrieval with various degrees of similarity, in *Proc. ACM Multimedia*, 2002, pp. 584–591.
[7] T. Pohle, M. Schedl, P. Knees and G. Widmer, Automatically adapting the structure of audio similarity spaces, in *Proc. 1st Workshop on Learning the Semantics of Audio Signals (LSAS)*, 2006, pp. 66–75.
[8] N. C. Maddage, H. Li and M. S. Kankanhalli, Music structure based vector space retrieval, in *Proc. SIGIR'06*, 2006, pp. 67–74.
[9] Q. Lv, W. Josephson, Z. Wang, M. Charikar and K. Li, Multiprobe LSH: efficient indexing for high dimensional similarity search, in *Proc. of the Very Large Data Base (VLDB)*, 2007, pp. 950–961.
[10] N. Bertin and A. Cheveigne, Scalable metadata and quick retrieval of audio signals, in *Proc. ISMIR'05*, 2005, pp. 238–244.
[11] P. Indyk and R. Motwani, Approximate nearest neighbor: towards removing the curse of dimensionality, in *Proc. 30th Annual ACM Symposium on Theory of Computing*, 1998, pp. 604–613.
[12] J. Reiss, J. J. Aucouturier and M. Sandler, Efficient multi dimensional searching routines for music information retrieval, in *Proc. 2nd ISMIR*, 2001.
[13] http://web.mit.edu/andoni/www/LSH/index.html LSH Algorithm and Implementation ($E^2$LSH).
[14] R. Panigrahy, Entropy based nearest neighbor search in high dimensions, in *Proc. ACM-SIAM Symposium on Discrete Algorithms* (*SODA*), 2006.
[15] M. Lesaffre and M. Leman, Using fuzzy to handle semantic descriptions of music in a content-based retrieval system, in *Proc. LSAS'06*, 2006, pp. 43–52.
[16] J. P. Bello, Audio-based cover song retrieval using approximate chord sequences: testing shifts, gaps, swaps and beats, in *Proc. ISMIR'07*, 2007, pp. 239–244.
[17] G. Tzanetakis and P. Cook, Musical genre classification of audio signals, *IEEE Trans. on Speech and Audio Processing* **10**(5) (2002) 293–302.
[18] D. Ellis and G. Poliner, Identifying cover songs with chroma features and dynamic programming beat tracking, in *Proc. ICASSP'07*, 2007, pp. 1429–1432.
[19] R. Miotto and N. Orio, A methodology for the segmentation and identification of music works, in *Proc. ISMIR'07*, 2007, pp. 239–244.
[20] L. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition* (Prentice-Hall, 1993).
[21] www.midomi.com.
[22] M. Casey and M. Slaney, Song intersection by approximate nearest neighbor search, in *Proc. ISMIR'06*, 2006, pp. 144–149.
[23] F. Moerchen, I. Mierswa and A. Ultsch, Understandable models of music collection based on exhaustive feature generation with temporal statistics, in *Proc. KDD'06*, 2006, pp. 882–891.
[24] Y. Yu, K. Joe and J. S. Downie, Efficient query-by-content audio retrieval by locality sensitive hashing and partial sequence comparison, *IEICE Trans. on Information and System* **E91-D**(6) (2008) 1730–1739.
[25] http://en.wikipedia.org/wiki/Information_retrieval.
[26] http://en.wikipedia.org/wiki/Mean_reciprocal_rank.
[27] J. Serra, E. Gomez, P. Herrera and X. Serra, Chroma binary similarity and local alignment applied to cover song identification, *IEEE Trans. on Audio, Speech and Language Processing* **16**(6) (2008) 1138–1152.

[28] K. Kashino, G. Smith and H. Murase, Time-series active search for quick retrieval of audio and video, in *Proc. ICASSP'99*, 1999, pp. 2993–2996.
[29] T. K. Vintsyuk, Speech discrimination by dynamic programming, *Kibernetika* **4**(2) (1968) 81–88.
[30] R. De Maesschalck, D. Jouan-Rimbaud and D. L. Massart, The Mahalanobis distance, *Chemometrics and Intelligent Laboratory Systems* **50**(1) (2000) 1–18.