Using Exact Locality Sensitive Mapping to Group and Detect Audio-Based Cover Songs

Yi Yu[+], J. Stephen Downie*, Fabian Moerchen**, Lei Chen[++], Kazuki Joe[+]
[+]*Department of Information and Computer Sciences, Nara Women's University Kitauoya nishi-machi, Nara, 630-8506, Japan. {yuyi, joe}@ics.nara-wu.ac.jp*
[*]*Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign 501 E. Daniel St. Champaign, IL, 61820, USA. jdownie@uiuc.edu*
***Siemens Corporate Research 755 College Road East Princeton NJ08540 USA. fabian.moerchen@siemens.com*
[++]*Department of Computer Science, Hong Kong University of Science and Technology, HKSAR, China. leichen@cs.ust.hk*

Abstract

Cover song detection is becoming a very hot research topic when plentiful personal music recordings or performance are released on the Internet. A nice cover song recognizer helps us group and detect cover songs to improve the searching experience. The traditional detection is to match two musical audio sequences by exhaustive pairwise comparisons. Different from the existing work, our aim is to generate a group of concatenated feature sets based on regression modeling and arrange them by indexing-based approximate techniques to avoid complicated audio sequence comparisons. We mainly focus on using Exact Locality Sensitive Mapping (ELSM) to join the concatenated feature sets and soft hash values. Similarity-invariance among audio sequence comparison is applied to define an optimal combination of several audio features. Soft hash values are pre-calculated to help locate searching range more accurately. Furthermore, we implement our algorithms in analyzing the real audio cover songs and grouping and detecting a batch of relevant cover songs embedded in large audio datasets.

1. Introduction

Powerful Internet provides us an open space to store and share with the world our music digital recordings or performance. An increasing number of people are joining the online-based music social communities via the Internet to upload their audio recordings or performance. Cover song is a new rendition of a previously recorded song in popular music (see http://en.wikipedia.org/wiki/Cover_version). With the personal recordings and performance increasing on the music social website, unknown cover song detection is becoming extremely important. Through the audio sequence comparison, an accurate audio-based pairwise matching is usually obtained. However, feature extractor of musical audio sequence is very high dimensional, which makes it time-consuming to quickly detect relevant audio tracks. Solving this hard problem involves two main aspects: (i) refining music representation to improve the accuracy of musical semantic similarity (pitch [5][22], Mel-Frequency Cepstral Coefficient (MFCC) [7], Chroma [19][21]) and (ii) organizing music documents in the way that helps speed up music similarity searching (trees structure [1][4][13] hierarchical structure[10], Locality Sensitive Hashing (LSH) [6][9]). Merely strengthening one point is not enough. For example, in [21] beat-synchronous Chroma features are successful for matching similar music audio sequences. Unfortunately, one pairwise comparison of feature sequences costs about 0.3 second. This means it would take about 30s if the database length is 100.

We pay attention to the above addressed aspects to solve audio-based cover song detection and retrieval: accurately locate the searching range of music audio tracks with the concatenated feature sets via exact locality sensitive mapping. A novel melody-based summarization principle is presented to determine a group of weights that define an optimal combination of several audio features, Features Union (FU), by using multivariable regression. In addition, we study the relationship between FU summarization and hash values

978-0-7695-3454-1/08 $25.00 © 2008 IEEE
DOI 10.1109/ISM.2008.18

and propose two retrieval schemes called Exact Locality Sensitive Mapping (ELSM) and SoftLSH. We confirm the practicality of our algorithms with real world multi-cover-version queries over the large musical audio datasets. Interesting examples of cover songs can be found and listened at http://www.e.ics.nara-wu.ac.jp /~yuyi/AudioExamples.htm.

2. Related work

To efficiently accelerate audio-based content detection, some researchers applied index-based techniques [1][4][6]. In [1] a composite feature tree (semantic features, such as timbre, rhythm, pitch, e.g.) was proposed to facilitate KNN search. The weight of each individual feature is determined by multivariable regression. 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 compared with an R*-tree. Though the number of features can be reduced, sometimes the summarized (grouped) features may not sufficiently discriminate two different signals.

LSH [11] is an index-based data organization structure proposed to find all similar pairs of a query point in the Euclidean space. It has gained great success in different applications as a well-known solution to determine whether any pair of documents are similar or not (web page [12], audio [6][9], image [15], video [14], etc.). Feature vectors are extracted from document and regarded as similar to one another if they are mapped to the same hash value. Yang used random sub-set of the spectral features (Short Time Fourier Transform) to calculate hash values for the parallel LSH hash instances in [6]. With a query as input, its features match reference features from hash tables. Then Hough transformation is performed on these matching pairs to detect the similarity between the query and each reference song by the linearity filtering. In [9] MFCC is extracted as the feature from single speech word. Based on basic LSH idea they proposed multi-probe LSH, which can probe multiple buckets that are probable to contain the content similar to query.

LSH scheme is described as follow. If two features (V_q, V_i) are very similar they will have a small distance $\|V_q - V_i\|$, hash to the same value and fall into the same bucket with a high probability. If they are quite different they will collide with a small probability. A function family $H = \{h: S \to U\}$, each h mapping one point from domain S to U, is called locality sensitive, if for any features V_q and V_i, the probability

$$Prob(t) = P_{rH}\left[h(V_q) = h(V_i) : \|V_q - V_i\| = t\right] \quad (1)$$

is a strictly decreasing function of t. That is, the collision probability of features V_q and V_i is diminishing as their distance increases. The family H is further called (R, cR, p_1, p_2) ($c > 1$, $p_2 < p_1$) sensitive if for any $V_q, V_i \in S$,

$$\begin{aligned} &if \parallel V_q - V_i \parallel < R, \;\; P_{rH}[h(V_q) = h(V_i)] \geq p_1 \\ &if \parallel V_q - V_i \parallel > cR, \; P_{rH}[h(V_q) = h(V_i)] \leq p_2 \end{aligned} \quad (2)$$

A good family of hash functions will try to amplify the gap between p_1 and p_2.

There is a significant shortcoming in existing works [1][4][6][9]. No analysis was done to investigate whether the used features can completely or maximally represent the original melody characteristics and have the capability in distinguishing two audio feature sequences. Also, there was no evaluation of the retrieval quality according to a natural benchmark (human perception). In this work we study the correlation between audio feature sets and soft hash/mapping values by using exhaustive musical audio sequence comparisons to predict desired but unseen music songs, and give some simple principles to evaluate melody-based music retrieval. In contrast to the existing works, our work has two advantages: (i). Similarity-invariance among audio feature sequence comparisons is applied in training semantic audio representations based on supervised learning. The proposed Features Union (FU) better represents musical audio sequences. (ii). We consider possible difference between perceptually similar audio documents and map the feature into continuous hash space (soft mapping). The neighborhood determined by the query will intersect buckets that possibly contain the similar documents. In comparison with the exhaustive KNN and previously used LSH retrieval schemes our algorithms achieve almost the same retrieval quality as KNN but with much less retrieval time.

3. Algorithms

In this section our algorithms include two main parts: spectrum-based audio semantic summarization and soft-hashing-based information retrieval. We reveal a principle of spectral-based similarity-invariance, by which we can summarize long audio feature sequences and generate a compact and semantic single feature FU. Instead of traditional hard hash values we assume a group of soft hash values and use exact locality sensitive mapping, which help to locate searching range more accurately. Associated with FU two retrieval schemes are proposed to solve the problem of cover song detection.

3.1. Spectrum-based features union

Audio documents can be described as time-series feature sequences. Directly computing the distance between audio feature sequences (matching audio documents) is an important task in implementing query-by-content audio information retrieval. Dynamic Programming (DP) [5][19] can be used in matching two audio feature sequences and is an essentially exhaustive searching approach (which offers high accuracy). But it lacks scalability and results in a lower retrieval speed as the database gets larger. To quicken the audio feature sequence comparison and obtain the scalable content-based retrieval, semantic features are extracted from the audio structures. The semantic features (high-level) used in [1] are mainly proposed for musical genre classification [20] and can not effectively represent melody-based lower-level music information. In the following section we propose a new semantic feature summarization suitable for melody-based music information.

A single feature can not well summarize a song. Multiple features can be combined to represent a song. These features play different roles in the query stage and must be weighed by different weights. Existing retrieval schemes have selected different audio features. We choose several competitive audio features and introduce a scheme based on multivariable regression to determine the weight for each feature. The goal of our approach is to apply linear and non-parametric regression models to investigate the correlation. In the model we use K (K=7) groups of features (218-dimension): Mean and Std of MFCC (13+13) [7], Mean and Std of Mel-Magnitudes (40+40) [8], Mean and Std of Chroma (12+12) [21], Pitch Histogram (88) [5].

Let the groups of features of mi^{th} song be $v_{mi,1}, v_{mi,2}, ..., v_{mi,K}$ (i=1,2). With different weight α_k assigned to each feature group the total summary vector is

$$V_{mi} = [\alpha_1 v_{mi,1}, \alpha_2 v_{mi,2}, ..., \alpha_K v_{mi,K}]^T \quad (3)$$

The Euclidean distance between two features V_{m1} and V_{m2} is

$$d(V_{m1}, V_{m2}) = d(\sum_{k=1}^{K} \alpha_k v_{m1,k}, \sum_{k=1}^{K} \alpha_k v_{m2,k}) = \sum_{k=1}^{K} \alpha_k^2 d(v_{m1,k}, v_{m2,k}) \quad (4)$$

To determine the weight in Eq.(3), we apply multivariable regression process. Consider a training database composed of M pairs of songs $< R_{m1}, R_{m2} >, m = 1, 2, ..., M$, which contain both similar pairs and non-similar pairs. From these pairs we obtain the sequences of Chroma similar to [21] and then calculate M sequence distances $d_{DP}(R_{m1}, R_{m2})$ via DP.

We will choose the weight in Eq.(3) so that the distance $d(V_{m1}, V_{m2})$, calculated by the summary, is as near to the sequence distance $d_{DP}(R_{m1}, R_{m2})$ as possible, i.e., we hope the melody information is contained in the summary. After we determine the distance between the pairs of training data, we get a M*K matrix D_V and a M-dimension column vector D_{DP}. The m^{th} row of D_V has K distance values calculated from independent features $d(v_{m1,k}, v_{m2,k}), k = 1, 2, ..., K$ and the m^{th} element of D_{DP} is the normalized distance between the two feature sequences $d_{DP}(R_{m1}, R_{m2}) \cdot K / (| R_{m1} | \cdot | R_{m2} |)$. Let $A = [\alpha_1^2, \alpha_2^2, \cdots, \alpha_K^2]^T$. According to Eq.(4) D_V, A and D_{DP} satisfies the equation $D_V \cdot A = D_{DP}$ Then $A = (D_V^T D_V)^{-1} D_V^T D_{DP}$ and we obtain the weight α_k. We are only interested in the absolute value of α_k. The feature set defined in Eq.(3) is call features union (FU).

3.2. Exact locality sensitive hashing/mapping

Almost all the hash schemes, including LSH, use hard (discrete) integer hash values. In LSH a FU 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 FUs (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 FUs 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 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.

3.2.1 SoftLSH

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 $H_m = \{h_{m1}, h_{m2}, ..., h_{mN}\}$. Its k^{th} function $h_{mk}(\cdot)$ maps an FU 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 $H_m(V) = \{h_{m1}(V), h_{m2}(V), ..., h_{mN}(V)\}$.

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

$$h_{mk}(V_i) - h_{mk}(V_j) \approx h'_{mk}(V_j)(V_i - V_j) \quad (5)$$

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.(5) $h_{mk}(V_i)$ 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 FUs falling in the same square have the same integer hash values. But this quantization can not well retain the distance between two FUs. Figure 1 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 FUs 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 set $\overline{H}_m(V_i)$ and $\overline{H}_m(V_j)$.

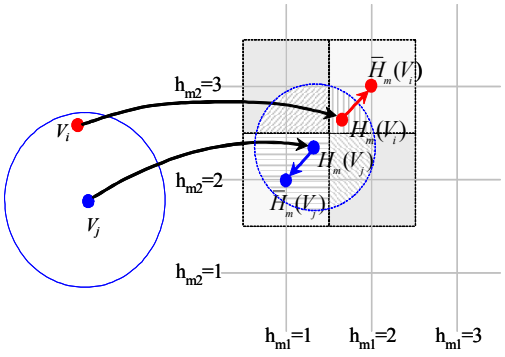

Figure 1 Concept of Soft LSH.

In Figure 1 the FU 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 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.

3.2.2 Query with ELSM

Our first solution to the quantization problem is to utilize the ELSM feature together with KNN instead of assigning FUs to buckets.

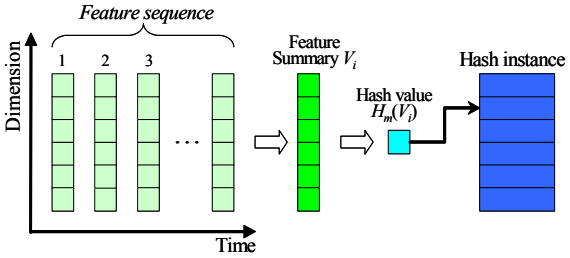

Figure 2 Feature organization in the database.

Each song in the database is processed as in Figure 2. Its FU feature V is obtained by the regression model. The m^{th} hash instance has its own sets of N hash functions $h_{mk}(V) = (a_{mk} \cdot V + b_{mk}) / w_{mk}$ ($1 \le k \le N$), 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 summarized semantic FU feature V is $H_m(V) = [h_{m1}(V), h_{m2}(V), ..., h_{mN}(V)]$ in the m^{th} hash instance. When there are L parallel hash instances, the hash vectors generated from the FU feature V for all hash instances are $H(V) = [H_1(V), H_2(V), ..., H_L(V)]$, which has $N*L$ 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 $N*L$ is much smaller than the dimension of FU. Though it can not provide a response as fast as SoftLSH, it is still much faster than utilization of the FU 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.

3.2.3 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 FU feature V_i, in the m^{th} hash instance its sequence 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 FU of the query, V_q, is calculated. In this way its ELSM feature, $H(V_q) = [H_1(V_q), H_2(V_q), ..., H_L(V_q)]$, 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.

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. The buckets near $\overline{H}_m(V_q)$ also possibly contain features similar to the query. Vertexes of these buckets, $\overline{H}_m(V_q) + (j_1 \pm 0.5, ..., j_N \pm 0.5)$, are examined. For the vector $(j_1, ..., j_N)$ where the vertexes falling in $C(H_m(V_q), r)$, $\overline{H}_m(V_q) + (j_1, ..., j_L)$ 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.

3.2.4 Summary

The original concept of LSH [11] was introduced in section 2. In E^{2}LSH [16] a high-dimensional feature vector is first projected to sub-feature space by a group of locality sensitive functions. Then hash values are calculated from the sub-features. ELSM is similar to the first half of E^{2}LSH. However, in 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. However, the ELSM feature is not mapped to the integer hash values. We also propose a SoftLSH scheme as a variation of LSH. It quantizes the ELSM feature into integer hash values and utilizes the ELSM feature to accurately locate the searching region.

All the LSH members solve Approximate Nearest Neighbors problem in a Euclidean space. E^{2}LSH [16] enhances LSH to make it more efficient for the retrieval with the very high dimensional feature. It performs locality sensitive dimension reduction to get the projection of the feature in different low-dimension sub-spaces. With multiple hash tables in parallel, the retrieval accuracy can be guaranteed meanwhile the retrieval speed is accelerated. Panigraphy [17] 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. According to Eq.(5) when the feature summary of two tracks are similar to each other, their non-quantized hash values will also be similar to each other. Instead of probing, our SoftLSH scheme utilizes the ELSM feature to accurately locate all the buckets that intersect the neighborhood determined by the query.

4. Experimental setup

Our music collection includes 5275 tracks that fall into five non-overlapping datasets. Trains80 is collected from www.yyfc.com (a non-commercial amusement website where users can sing her/his favorite songs, make records online, and share them with friends) and our personal collections. 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 regression model proposed in section 3.1. Covers79 is also collected from www.yyfc.com and consists of 79 popular Chinese songs each represented in different versions (sung by different people with similar background music). Each song has 13.5 versions on the average resulting in a total of 1072 audio tracks.

RADIO is from www.shoutcast.com and ISMIR is collected from http://ismir2004.ismir.net/genre_contest. 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, 16bit per sample and the sampling rate is 22.05KHz. The audio data is nor-

malized and then divided into overlapped frames. Each frame contains 1024 samples and the adjacent frames have 50% overlap. Each frame is weighed by a hamming window and further appended with 1024 zeros to fit the length of FFT (2048 point). From FFT result the instantaneous frequencies are extracted and Chroma is calculated. From the amplitude spectrum pitch, MFCC and Mel-magnitude are calculated. Then the summary is calculated from all frames.

The ground truth is set up according to human perception. We have listened to all the songs and manually labeled them so that retrieval results of our algorithms correspond to human perception to support practical application. Trains80 and Covers79 datasets were divided into groups according to their verse (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 in this group). The 30s segments in these two datasets are extracted from verse sections of songs.

5. Evaluation

In this section we present the performance evaluation of the searching schemes, KNN, ELSM, LSH and SoftLSH, and give the corresponding analysis and demonstrate their potential applications in query-by-content musical audio retrieval. All schemes are based on the FU feature. 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 (lower bound). We hope our algorithms would approach KNN in the performance while retaining almost the same retrieval time as LSH. Our task is mainly to solve the problem of cover songs detection or near duplicate detection of audio files similar to [2][5][19][21]. Our methods can be extended to solve the query-by-example audio-based retrieval problems.

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 as the query in turn we would calculate the ranked tracks similar to the query. Each query q chosen from Covers79 has its relevant set S_q (perceptually similar songs), which is determined according to the number of audio cover tracks in each group. The average size of query's relevant set is 12.5 (on the average each song in Coves79 has 13.5 covers. When one cover is used as query, the rest covers are in the database). The total number of relevant items can be calculated from each group (a theoretical maximum is 14452). The retrieval system also outputs the retrieved set K_q. To evaluate performance of the algorithms, in our experiment recall and precision are respectively defined as $|S_q \cap K_q| / |S_q|$ and $|S_q \cap K_q| / |K_q|$, and also F-measure is defined as $2 \cdot recall \cdot precision / (precision + recall)$.

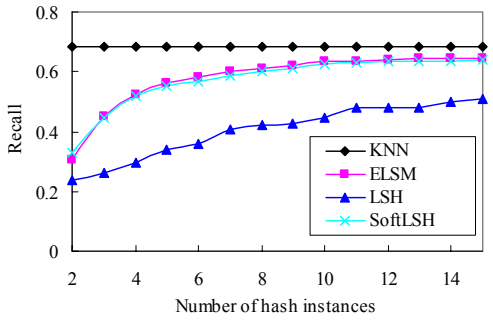

Figure 3 Recall under different number of hash instances.

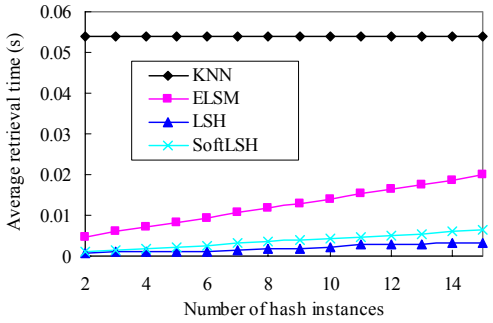

Figure 4 Average retrieval time under different number of hash instances.

Figure 3 is the recall of the four schemes in question. KNN is the golden standard given this feature representation and always performs best. LSH is always inferior to the new proposed schemes. When there are very few hash instances, ELSM feature has a low dimension and can not well represent FU. 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 are approaching 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 further increase of 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 can not be salvaged by the increase of hash instances. When there are 10 hash instances, 0.682*14452 can be identified with KNN, 0.633*14452 are identified with ELSM, 0.445*14452 are identified with LSH and 0.625*14452 are identified with SoftLSH.

Figure 4 shows the average retrieval time for each query. The exhaustive KNN always takes the longest time (0.542s). Time consumption in the other three schemes gradually increases as the number of hash instances does. Average retrieval time of SoftLSH is about double as much as LSH due to the search in multiple buckets that intersect the query's neighborhood. From Figure 3 and Figure 4 the tradeoff among accuracy and time indicates that 10 hash instances are a suitable choice. In such cases SoftLSH has a recall close to KNN with a much less retrieval time. The additional time saved by LSH would result in a significant drop of accuracy. Therefore the number of hash table is set to 10 in the following experiments.

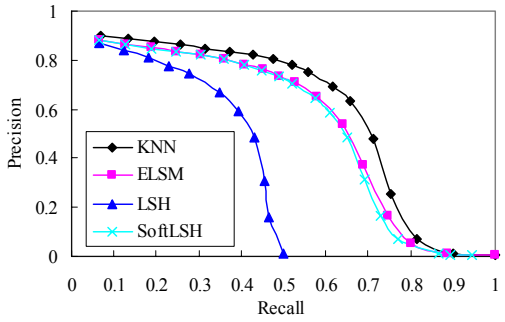

Figure 5 Precision-recall curve (10 hash instances).

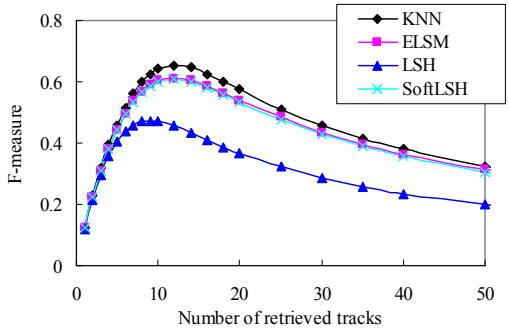

Figure 6 *F*-measure at different number of retrieved tracks.

Figure 5 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 precision. Some of the perceptually similar tracks have quite different 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. ELSM and SoftLSH have a performance approaching that of KNN. But at the same precision they have a loss of about 4% in recall compared with KNN due to utilizing a lower dimensional feature. The number of hash instances is fixed at 10 in the experiment. Some of the tracks can not be retrieved by the LSH scheme at all. Its recall is upper bounded at 0.5 and a higher recall requires much more hash instances in LSH compared with SoftLSH.

Figure 6 demonstrates *F*-measure scores of the four schemes with respect to different number of retrieved tracks. It can be seen easily that the 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 Figure 4. Here we would address that when the number of the retrieved tracks is less than that of the query's covers in the database, 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 larger than the actual tracks, the slopes of the recall curves in all schemes become steady while increasing the retrieved tracks always results in a decrease of precision. In this experiment each query has an average number of 12.5 covers in the database. Coincidently in Figure 6 the curves of KNN, ELSM and SoftLSH reach the maximal F-measure score when the number of returned songs equal 12. This reflects that the FU feature is very effective in representing the similarity of tracks in each group. The tracks belonging to the same group that really have a short distance quickly appear in the returned list. Not-so-similar tracks have a relatively large distance and too many retrieved tracks only result in a very low precision and F-measure. It also confirms the SoftLSH is a good alternative to KNN.

6. Conclusion

Both the representation and organization of audio files play important roles in audio content detection. In this paper we have considered both the semantic summarization of audio documents and the hash-based approximate retrieval for the purpose of reducing retrieval time and improving retrieval quality. By a new principle of similarity-invariance, a concise audio feature representation (FU) is generated based on multivariable regression. Associated with the FU, variants of LSH (ELSM and SoftLSH) are proposed. Different from the conventional LSH schemes, soft hash values are exploited to accurately locate the searching region

and improve the retrieval quality without requiring many hash instances. It is easy to make the proposed retrieval schemes applicable to other applications with a bit effort (especially video, bio-informatics, e,g.).

We experimentally show the efficacy of our algorithms via evaluation on 'multi-versions' music covers datasets, adopting human perception as a quality measure. As expected our results demonstrate that (i) the FU feature is a good summary of audio sequence (ii) SoftLSH achieves a better balance between retrieval time and accuracy than conventional LSH and KNN. This work remains the room to be improved. In the future we will study semantic features that better represent melody information and other training models that best combine feature groups.

Acknowledgment

We thank Initiative Project of Nara Women's University for supporting the first author to visit IMIRSEL, where this work was partly discussed in summer, 2007. The second author was supported by the Andrew W. Mellon and national Science Foundation (NSF) under Nos. IIS-0340597 IIS-0327371.

References

[1] B. Cui, J. Shen, G. Cong, H. Shen, C. Yu. Exploring Composite Acoustic Features for Efficient Music Similarity Query, ACM MM'06, pp.634-642, 2006.

[2] M. Robine, P. Hanna, P. Ferraro and J. Allali. Adaptation of String Matching Algorithms for Identification of Near-Duplicate Music Documents, SIGIR Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection, 2007.

[3] J. F. Serrano J. M. Inesta. Music Motive Extraction through Hanson Intervallic Analysis. CIC'06, pp.154-160, 2006.

[4] I. Karydis, A. Nanopoulos, A. N. Papadopoulos and Y. Manolopoulos, "Audio Indexing for Efficient Music Information Retrieval", MMM'05, pp.22-29, 2005.

[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", pp.183-190, ISMIR2005.

[6] C.Yang, "Efficient Acoustic Index for Music Retrieval with Various Degrees of Similarity", ACM Multimedia, pp.584-591, 2002.

[7] T. Pohle, M. Schedl, P. Knees, and G. Widmer. Automatically Adapting the Structure of Audio Similarity Spaces. Proceedings of the 1st Workshop on Learning the Semantics of Audio Signals (LSAS), pp.66-75, 2006.

[8] L.Rabiner and B.-H. Juang. Fundamentals of Speech Recognition. Prentice-Hall, 1993.

[9] Q. Lv , W. Josephson, Z. Wang, M. Charikar, K. Li, "MultiProbe LSH: Efficient Indexing for High Dimensional Similarity Search", In Proc. of the Very Large Data Base (VLDB), pp.950-961, 2007.

[10] N. Bertin, A. Cheveigne, "Scalable Metadata and Quick Retrieval of Audio Signals", ISMIR 2005, pp.238-244, 2005.

[11] P. Indyk and R. Motwani. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. In Proc. of the 30th Annual ACM Symposium on Theory of Computing, pp.604–613, 1998.

[12] M. Henzinger. Finding Near-Duplicate Web Pages: a Large-Scale Evaluation of Algorithms. In Proc. of the 29th conference on research and development in IR, 2006.

[13] J. Reiss, J. J. Aucouturier and M. Sandler, "Efficient multi dimensional searching routines for music information retrieval", 2nd ISMIR, 2001.

[14] S. Hu, "Efficient Video Retrieval by Locality Sensitive Hashing", ICASSP 2005, pp.449-452, 2005.

[15] P. Indyk and N. Thaper, "Fast color image retrieval via embeddings," Workshop on Statistical and Computational Theories of Vision (ICCV), 2003.

[16] LSH Algorithm and Implementation (E2LSH) http://web.mit.edu/andoni/www/LSH/index.html.

[17] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In Proc. of ACM-SIAM Symposium on Discrete Algorithms(SODA), 2006.

[18] M. Lesaffre and M. Leman, "Using Fuzzy to Handle Semantic Descriptions of Music in a Content-based Retrieval System.", In Proc. LSAS06, pp.43-5, 2006.

[19] J. P. Bello, "Audio-based Cover Song Retrieval Using Approximate Chord Sequences: Testing Shifts, Gaps, Swaps and Beats", pp.239-244, ISMIR2007.

[20] G. Tzanetakis and P. Cook, "Musical Genre Classification of Audio Signals", IEEE Transactions on Speech and Audio Processing, Vol.10, No.5, pp. 293-302, 2002.

[21] D. Ellis and G. Poliner. Identifying cover songs with chroma features and dynamic programming beat tracking. In Proceedings of ICASSP-07, Volume: 4, pp.1429-1432, 2007.

[22] R. Miotto and N. Orio. "A Methodology for the Segmentation and Identification of Music Works.", pp.239-244, ISMIR 2007.