IEICE TRANS. INF. & SYST., VOL.E91–D, NO.6 JUNE 2008

PAPER

Efficient Query-by-Content Audio Retrieval by Locality Sensitive Hashing and Partial Sequence Comparison*

Yi YU[†a)], *Student Member*, Kazuki JOE[†], *and* J. Stephen DOWNIE[††], *Nonmembers*

SUMMARY This paper investigates suitable indexing techniques to enable efficient content-based audio retrieval in large acoustic databases. To make an index-based retrieval mechanism applicable to audio content, we investigate the design of Locality Sensitive Hashing (LSH) and the partial sequence comparison. We propose a fast and efficient audio retrieval framework of query-by-content and develop an audio retrieval system. Based on this framework, four different audio retrieval schemes, LSH-Dynamic Programming (DP), LSH-Sparse DP (SDP), Exact Euclidian LSH (E^2LSH)-DP, E^2LSH-SDP, are introduced and evaluated in order to better understand the performance of audio retrieval algorithms. The experimental results indicate that compared with the traditional DP and the other three compititive schemes, E^2LSH-SDP exhibits the best tradeoff in terms of the response time, retrieval accuracy and computation cost.

key words: *indexing, locality-sensitive hashing, content-based audio retrieval, dynamic programming*

1. Introduction

Content-based audio retrieval is not only a very promising research topic but also one of the main problems in multimedia information processing. Handling audio sequence data is usually time-consuming due to the high dimensionality of the features, which makes it inconvenient to utilize the potential content-based information retrieval techniques on the Internet or personal media devices. To access a huge mass of audio information efficiently, it is necessary to explore the audio information, facilitate the management of audio data and serve multimedia applications. Consequently various indexing structures have been reported in the study of audio retrieval. These include, for example, hierarchical structure [1], R-trees [2], M-trees [3], KD-trees [4], active-search [18], Locality Sensitive Hashing (LSH) [5], [16], [17].

As far as the creation of query-by-content audio retrieval mechanism via indexing techniques is concerned the main challenges are as follows: (1) How to characterize a corpus of acoustic objects with a corpus of relevant features. (2) How to organize audio features by indices. (3) How to locate the desired music segments with a given acoustic query sequence within the acceptable time.

There are several levels of closeness with increasing logic level in the similarity match between two songs [5]. We aim to search in an acoustic database the songs with the same main melody as the query and accelerate the retrieval by the indexing technique. This paper extends our previous work [16], [17], focusing on the studies of the algorithms to evaluate its scalable performance. Mainly we show how to design the audio indexing structures and how to realize partial sequences comparison so as to support scalable query-by-content audio retrieval. The retrieval procedure can be divided into two stages. Firstly features of the reference songs are organized in the database where "indexing" means assigning hash values to features so that features are stored in the buckets according to their hash values. In times of query "retrieving" means (i) finding in the database the features that may be similar to the features of the query by the hash values and (ii) generating a ranked list of reference songs in the decreasing order of their similarity to the query. We depend on mapping features to hash values by heuristics and reducing pairwise comparisons (pairwise comparison means the distance calculation between a pair of features) by designing hashing structure.

This work begins with a novel approach to making effective comparisons of the massive acoustic sequences by designing appropriate metrics and algorithms to avoid all pairwise comparisons of feature sequences. We care about indexing structure on acoustic sequences and reorganization of feature sequences. We present a novel framework to perform audio indexing and retrieval and provide scalable content-based searchability. The following retrieval scenario is considered: given a corpus of N musical reference songs find the similar songs with a query input of a much shorter length. First each audio data is divided into frames and spectral features are extracted. The reference songs can be represented by $R = \{r_{i,j} : r_{i,j} \in R_i,\ 1 \le i \le N, 1 \le j \le |R_i|\}$, where $r_{i,j}$ is the j^{th} spectral feature of the i^{th} reference song R_i. Then the spectral features of the query, $q_1, q_2, \ldots, q_Q$, are used to filter the resemblances by Locality Sensitive Hashing (LSH) or Exact Euclidean LSH (E^2LSH). The resembled features of the i^{th} reference songs are reorganized into partial sequences and compared with the query by either Dynamic Programming (DP) or the proposed Sparse DP (SDP).

The rest of the paper is organized as follows: Sect. 2 provides the background of the LSH/E^2LSH and related works. Section 3 presents the framework of audio index-

Manuscript received November 26, 2007.
Manuscript revised February 11, 2008.

[†]The authors are with the Graduate School of Humanity and Science, Nara Women's University, Nara-shi, 630–8506 Japan.
[††]The author is with the Graduate School of Library and Information Science, University of Illinois Urbana Champaign 501 E. Daniel St. Champaign, IL, 61820, USA.

*This work was partly discussed and done when Yi visited IMIRSEL Summer, 2007.

a) E-mail: yuyi@ics.nara-wu.ac.jp

DOI: 10.1093/ietisy/e91–d.6.1730

ing and describes content-based retrieval schemes in detail. Section 4 lists the simulation environment and analyzes the experiment results. Finally Sect. 5 concludes the paper.

2. Background and Related Work

2.1 Basics of LSH/E^{2}LSH

LSH is an index-based data organization structure proposed to improve the scalability for retrieval over a large database, which essentially is a spatial access method—construct some locality sensitive hashing functions to perform indexing in parallel in Euclidean space [9], [10]. LSH plays an important part in various applications, e.g., database access and indexing [11], data compression and mining, multimedia information retrieval [5], [12]. In particular the features of the objects are represented as the points-form in the high dimensional space and a distance metric is adopted to judge whether two multimedia objects are similar (such as audio [5], video [12], image [13]).

E^{2}LSH [14] is to solve Approximate Nearest Neighbors problem in a high-dimensional Euclidean space. It 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. If two features (q, r) are very similar they will have a small distance $\|q - r\|$, 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 $H = \{h : S \rightarrow U\}$, each h mapping one point from domain S to U, is called locality sensitive, if for any features q and r, the probability

$$Prob(d) = P_{rH}\left[h(q) = h(r) : \|q - r\| = d\right] \tag{1}$$

is a strictly decreasing function of d. That is, the collision probability of features q and r 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 $q, r \in S$,

$$\begin{aligned} &\text{if } \|q - r\| < R,\ P_{rH}[h(q) = h(r)] \geq p_1 \\ &\text{if } \|q - r\| > cR,\ P_{rH}[h(q) = h(r)] \leq p_2 \end{aligned} \tag{2}$$

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

Consider a distribution D over $\Re$ and any i.i.d. random variables $x_1, x_2, \ldots, x_l, x$ with distribution D. Let $X = (x_1, x_2, \ldots, x_l)^T$. If there exists P so that for any l-dimension real vector $\bar{v}_k = (v_1^k, v_2^k, \ldots, v_l^k)^T$, $f_{\bar{v}_k}(X) = \sum_{i=1}^{l} v_i^k x_i$ has the same distribution as $\left(\sum_{i=1}^{l} |v_i^k|^P\right)^{1/P} x$, the distribution D is called P-stable. $\bar{v}_k$ satisfying $\left(\sum_{i=1}^{l} |v_i^k|^P\right)^{1/P} = 1$ makes $f_{\bar{v}_k}(X)$ follow the same distribution D as x.

In E^{2}LSH the locality sensitive dimension reduction can be applied on a vector whose each dimension follows the same P-stable distribution. Each $f_{\bar{v}_k}(\cdot)$ with the parameter $\bar{v}_k$ $(1 \leq k \leq m)$ generates a single output. Then the $l * m$ matrix $V = (\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_m)$ leads to an m-dimension vector $f_V(X) = (f_{\bar{v}_1}(X), f_{\bar{v}_2}(X), \ldots, f_{\bar{v}_m}(X))^T$, each dimension of which also follows the distribution D. When each dimension of q and r follows a P-stable distribution, each dimension of $f_V(q)$ and $f_V(r)$ also follows the same distribution. Then q and r can be replaced by $f_V(q)$ and $f_V(r)$ respectively in Eq. (1-2).

2.2 Acceleration of Sequence Comparison

A quick search engine is of great importance to retrieval systems. Many researchers [6]–[8] have studied DP and also applied DP or optimized DP in content-based music information retrieval to match the query input against the songs in the database. Usually DP requires the calculation of distance between all pairs of features so as to fill in the DTW table and then performs the sequence comparison.

Instead of exhaustive search, reduction of the number of comparisons can effectively decrease the response time. In our previous work [15] spectral-similarity based feature merge was proposed to remove the spectral redundancy. In [2] the extracted features are grouped by Minimum Bounding Rectangles (MBR) and compared with an R*-tree. In [18] the time-series active search scheme is adopted. Features in a window are summarized by a histogram. Though the number of features can be reduced in [2], [18], sometimes the summarized (grouped) features may not sufficiently discriminate two different signals.

Some researchers also applied LSH in the field of audio sequences comparison. Yang used random sub-set of the spectral features to calculate hash values for the parallel LSH hash instances in [5]. 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.

2.3 Our Work

Similar to [5] our retrieval scheme adopts the spectral feature. We would like to avoid exhaustive comparison as suggested by [2], [5], [18]. Compared with [2] and [18] where the feature summary is used to reduce the number of total features, we filter features in the database that are similar to the query by LSH/E^{2}LSH without losing the discriminating capability. In comparison with [5] where LSH is also adopted to organize features, we utilize both LSH and E^{2}LSH and report retrieval mechanisms of index-based audio sequences, data organization structure of features as well as recreation and comparison of the feature sequences. In our system DP takes the query sequence and the partial reference sequence (partial sequence has a shorter length compared with the original one) as input and compares them over a DTW table with a smaller size. The proposed SDP uses the full size DTW so that the calculated distance in

the filtering stage by LSH/E^{2}LSH can be directly filled in, avoiding recalculation of the distances. The extensive comparison among the presented four schemes (LSH-DP, LSH-SDP, E^{2}LSH-DP, E^{2}LSH-SDP) shows that the optimal combination—E^{2}LSH-SDP—outperforms the others.

3. Design of Index-Based Audio Content Retrieval

We focus on providing fast and efficient content-based retrieval mechanisms of searching audio sequences data over a large audio sequences database by utilizing a suitable indexing structure. Our retrieval system allows a user to take a fragment of the query song as input, performs content-based audio retrieval and returns songs similar to this query fragment.

3.1 The Basic Framework

The indexing framework for music information retrieval consists of two main parts: (i) building indexing data structure of feature sequences and (ii) matching the recovered feature sequences. For the first object E^{2}LSH/LSH establishes an effective data structure for acoustic-based music information. For the second object SDP/DP matches the recovered feature sequences and obtains the song closest to the query. Therefore, we propose four different retrieval schemes, LSH-DP, LSH-SDP, E^{2}LSH-DP and E^{2}LSH-SDP. Based on this general framework we want to evaluate the four schemes to solve the problem of scalable audio content retrieval and select the best tradeoff according to the response time, retrieval accuracy and computation cost. The details are discussed in the experiment parts.

We begin by introducing the basic query-by-content framework of acoustic-based music information retrieval. This framework is shown in Fig. 1 and its main procedure is summarized as follows: For each frame, its high dimensional spectral feature, Short Time Fourier Transform (STFT), is calculated. The spectral features of all the reference songs in form of searchable symbols are stored in the hash tables according to their respective hash values. E^{2}LSH/LSH provides an effective organization of the audio features, gives a principle to store the feature sequences, and facilitates an efficient sequences reconstruction. With each feature in the query sequence, the candidate features are searched in the hash table. The non-similar features are filtered out. Then the remaining features are reorganized into new sequences and the SDP/DP algorithm is employed to perform an accurate similarity comparison between the query and the partial reference sequences.

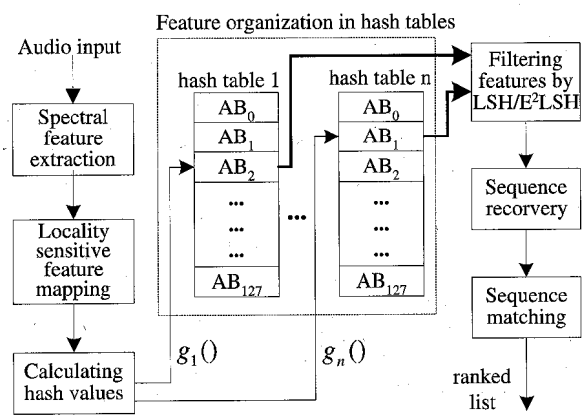

Fig. 1 An index-based audio retrieval framework. It mostly consists of locality sensitive feature mapping, parallel hash tables, filtering features, sequence recovery and matching.

The Central Limit Theorem states that if the sum of many independent and identically-distributed random variables has a finite variance, it will be approximately normally distributed (i.e., following a Gaussian distribution). In the calculation of STFT, each bin is a weighed sum of (e.g. $N = 2048$) consecutive random audio samples and can be approximated by a Gaussian distribution. Any linear combination of Gaussian distribution is still Gaussian. In such cases P equals 2 in the calculation of E^{2}LSH.

3.2 Calculating Hash Values

First we consider a single hash instance, which logically consists of all the features of the audio sequences in the database. Each of the audio sequences is divided to frames. For each frame STFT is calculated as the spectral feature and regarded as a searchable symbol in Euclidean space. This feature goes through a linear transformation—locality sensitive mapping. When LSH is used, this mapping generates a new feature with the same size. While in E^{2}LSH, the mapping also reduces the feature dimension: a high-dimensional feature X is projected to a low-dimension sub-feature $f_V(X)$ with the P-stable random matrix V. The sub-feature is of dimension m. Then the sub-feature is per-dimensionally quantized to an integer vector, followed by hash value calculation. This quantization ensures the locality sensitivity in the hash. An equivalent hash function $g(\cdot)$ involves the effect of $f_V(\cdot)$ and the hash function $H(\cdot)$. In the following, $g(\cdot)$ also means an audio bucket storing all the features with the same hash value. Its meaning is obvious from the context.

Figure 2 gives an example of the hash instance with two original features q_0 and r_0. The new features after locality sensitive mapping are q and r. Then per-dimension quantization is performed. This results in integer vectors. With the random weight, their hash values, $g(r_0)$ and $g(q_0)$

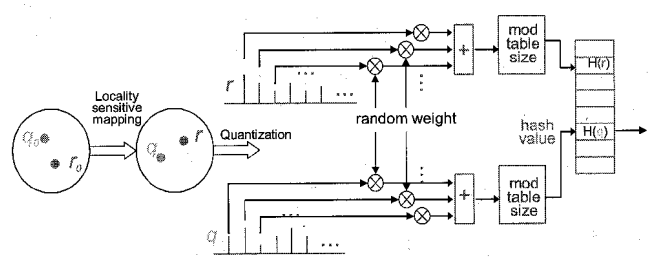

Fig. 2 Hash calculation.

are calculated. Due to the random weight the hash values of reference features are pseudo-random and belong to a wide range. By the following *mod* operation on the hash table size they are projected to a predefined integer range (e.g., 0-127) and associated with the buckets. In such cases, the hash table size needs not be a prime and a power of two (e.g. 128) makes it convenient for the *mod* operation.

If q and r have a short Euclidian distance, then with a high probability they are quantized to the same integer sequences and generate the same hash value. Since r is stored in the bucket $H(r)$, it is expected that r_0 can be found by q_0 when $g(q_0) = g(r_0)$. Sometimes even two vectors with a close distance may be quantized to two different integer vectors because the values of a specific dimension belong to adjacent quantization space. As a result the two close vectors have different hash values. More hash instances are necessary to resolve this problem and increase the retrieval accuracy of LSH/E^{2}LSH, as was stated in [14]. Therefore we construct several hash tables, each logically containing all the features of the references. Hash instances are distinguished by their own locality sensitive mapping. The k^{th} hash instance has an equivalent hash function $g_k(\cdot)$, a combination of $f_{V_k}(\cdot)$ and $H_k(\cdot)$. Then it is probable that the quantized vectors of two similar features are the same in at least one of the hash instances.

3.3 Feature Organization in the Hash Tables

The j^{th} spectral feature of the i^{th} reference song, $r_{i,j}$, is stored in $g_k(r_{i,j}), k = 1, 2, \ldots$. Its song number i and the corresponding time offset j are stored together with the feature, for the purpose of providing facilities for reconstruction of the partial acoustic sequences after the filtering stage.

3.4 Filtering Features by LSH/E^{2}LSH

In the query stage a sequence of query features $q_1, q_2, \ldots, q_Q$ is used to find the closest reference song. With a query feature q_m, the candidate reference features in the bucket of the k^{th} hash table, $g_k(q_m)$, can be obtained. This bucket contains all the features matching to the same hash value. Though it is probable that the resemble features lie in the bucket, other non-similar features also exist due to the limited hash table size. It is necessary to remove these non-similar features so as to reduce the post computation. We define a distance function that can represent the similarity degree, $d(X, Y) = \|X - Y\|_2 / \sqrt{\|X\|_2 \cdot \|Y\|_2}$, the normalized Euclidean distance. From the bucket $g_k(q_m)$, we get the match pairs $<q_m, r_{i,j}, d>$. If the distance d between the feature $r_{i,j}$ and q_m is greater than δ, the feature $r_{i,j}$ is filtered out and discarded by Eq. (3). Namely, we would like to retain such features that lie within the ball centered at q_m with a radius δ.

$$S_{k,i,m} = \{r_{i,j} : r_{i,j} \in g_k(q_m), d(f_{V_k}(q_m), f_{V_k}(r_{i,j})) \leq \delta\} \quad (3)$$

The union $S_{k,i} = \bigcup_m S_{k,i,m}$ gives the candidate features of the i^{th} reference obtained from the k^{th} hash table for the whole query sequence. And the total candidates of the i^{th} reference are obtained from all the hash tables as $S_i = \bigcup_k S_{k,i}$. Since in E^{2}LSH $f_{V_k}(X)$ has a much smaller size than X, the filtering stage of E^{2}LSH can be greatly accelerated in contrast to LSH, as is verified by the simulation.

Figure 3 gives an example of single feature matching with two hash instances, where q is similar to f_1, f_2, f_3. In the first hash instance, q, f_1, f_3 and f_5 have the same hash value. With q, f_1 and f_3 can be found and f_2 is missing. The non-similar f_5 can be filtered out if the distance $\|f_5 - q\|$ is greater than the threshold δ defined in Eq. (3). In the second hash instance, f_1, f_2 and f_4 have the same hash value as q and f_3 is missing. f_4 can be filtered out if the distance $\|f_4 - q\|$ is greater than δ. In the best case where both f_4 and f_5 are filtered out, the union of results from two hash instances gives the similar set $\{f_1, f_2, f_3\}$ as desired. Due to the approximation property of LSH/E^{2}LSH, the non-similar features f_4 and f_5 may also have a distance to q less than δ and remain in the similar set. Then the following sequence comparison is necessary to prevent the non-similar songs from appearing in the final ranked list. In Fig. 4 a query with 5 features is used to find the target reference song with the help of two hash tables. With each of the query features, the reference features in the two buckets of the two hash tables are obtained and the non-similar features are filtered out with the best effort. With 5 query features similar reference features in the 10 buckets are obtained and finally 14 features remain.

3.5 Sequence Matching

Few features of the songs that are non-similar to the query remain after filtering. Among the features of the song that

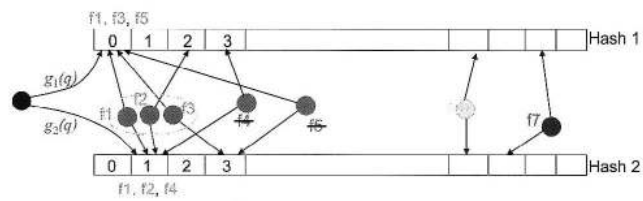

Fig. 3 Matching with a single feature.

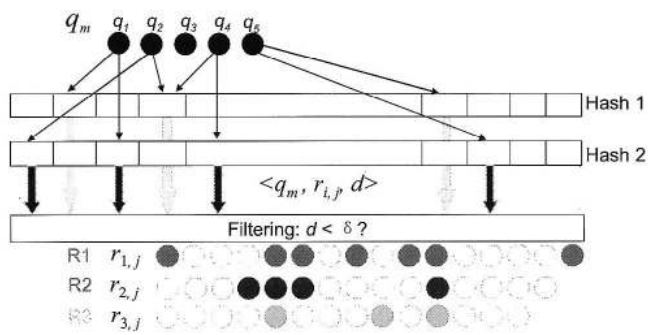

Fig. 4 Matching with a sequence of features.

actually match the query only part of features remain. In addition the remaining features that belong to the same song may not be in the same time order as those of the query. The reordering of the remaining features of the i^{th} reference song is to reorganize the features of the partial reference sequence in the ascending time offset $j_1, j_2, \cdots, j_R$, where $j_1, j_2, \cdots$ are non-continuous integers indicating the frame number of the partial reference sequence. Then the query $q_1, q_2, \ldots, q_Q$ is matched against each partial reference sequence to find the desired song with the DP method similar to [7] or the proposed SDP scheme.

3.5.1 Sequence Comparison with DP

Assume that $D_i(m, j_n)$ is the minimum distance between the query and the i^{th} reference song, beginning from the leftmost side $(1, j_1)$ of the DTW table to the current position (m, j_n). Equation (4) gives the recursive relation

$$D_i(m, j_n) = d(q_m, r_{i,j_n}) + \min \begin{cases} D_i(m-2, j_{n-1}) \\ D_i(m-1, j_{n-1}) \\ D_i(m-1, j_{n-2}) \end{cases} \quad (4)$$

The matching features from different hash tables belong to different sub-space and have different distance to the query. Then the Euclidean distance $d(q_m, r_{i,j_n})$ calculated from the original features q_m and r_{i,j_n} is used in Eq. (4). A single match pair $<q_m, r_{i,j_n}, d>$ is enough to provide the match information and the duplicate match pairs are neglected. Then the distance between the query and the i^{th} reference is

$$D_i = \min_{j_n} D_i(Q, j_n) \quad (5)$$

Among all the reference songs, the following equation gives the one that best matches the query.

$$i_{DP} = \arg \min_i D_i \quad (6)$$

3.5.2 Partial Sequence Comparison with SDP

A much simple comparison way is to directly utilize the distance calculated in the filtering stage by filling the distance in a DTW table. However, as shown in Fig. 5, most of the points in the DTW table are gone after filtering. Therefore, the conventional DP does not work well. We perform the sequence comparison in a different way. For each matched pair $<q_m, r_{i,j_n}>$ from k^{th} hash table, the reverse of its distance is used as the weight in the matching procedure, as shown in Eq. (7). δ is the filtering threshold used in Eq. (3). Therefore the weight is no less than 1. With $w_{\max}$, an occasional perfect match of a single feature has less effect on the sequence comparison. On the other hand, if the pair $<q_m, r_{i,j_n}>$ does not exist, its weight is set to 0.

$$w_k(q_m, r_{i,j_n}) = \begin{cases} \min\{\delta / d(f_{V_k}(q_m), f_{V_k}(r_{i,j_n})), w_{\max}\} \\ 0, \quad <q_m, r_{i,j_n}> \text{ does not exist} \end{cases} \quad (7)$$

The recurrence of a single match pair in different hash tables means that the pair is a suitable match with a high probability. Then instead of a single weight in Eq. (7), the weights of the duplicate match pairs (with the same suffix) from different hash tables can be accumulated by $\sum_k w_k(q_m, r_{i,j_n})$ and in Fig. 5 the match pair becomes $<q_m, r_{i,j_n}, weight>$, where *weight* equals $\sum_k w_k(q_m, r_{i,j_n})$.

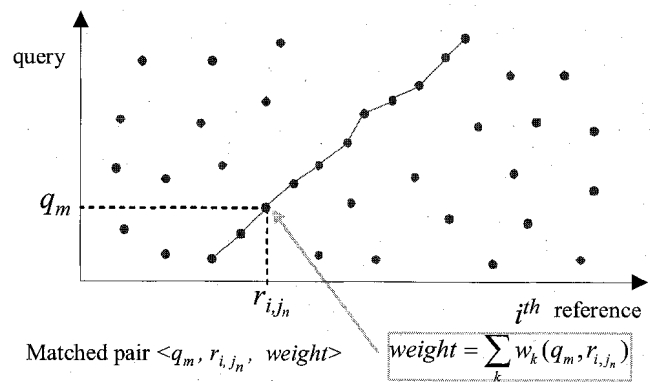

Fig. 5 Sequence comparison with SDP. Since most of the pairs are removed by LSH/E^{2}LSH and the filtering stage, the remaining points are sparse in the DTW table.

The sequence comparison is to select the path that maximizes the total weight. Despite the absence of most matching pairs in Fig. 5, the matching path still contains as many points as possible. The maximum weight is found by the iteration in Eq. (8)

$$W_i(m, j_n) = \sum_k w_k(q_m, r_{i,j_n}) + \max \begin{cases} W_i(m-2, j_{n-1}) \\ W_i(m-1, j_{n-1}) \\ W_i(m-1, j_{n-2}) \end{cases} \quad (8)$$

For the i^{th} reference song, its weight W_i is obtained by Eq. (9). Then by Eq. (10) the one with the maximum weight matches the query.

$$W_i = \max_{j_n} W_i(Q, j_n) \quad (9)$$

$$i_{SDP} = \arg \max_i W_i \quad (10)$$

4. Experiments Results and Analysis

The experiments have been carried out on the acoustic database with both monophonic and polyphonic melodies. Altogether we collected 506 songs from the public web sites. These songs belong to three sub-sets. (i) 122 western songs falling into 4 genres: dance, classical, jazz and country. (ii) $67 + 44 \times 2 = 155$ Chinese folks from Twelve Girls Band. (iii) 229 popular songs from two female Chinese singers. Among the 506 songs, 44 songs in the second subset have two versions with possible tempo variations. The rest $506 - 44 \times 2 = 418$ songs have single version. These 418 single-version songs and 44 out of the two-version songs form the database and each of the 462 references consists of a 60-second-long melodic slip. We also prepared 462 queries, each being 6-8 seconds long. 418 queries corresponding to the single-version songs are randomly segmented from the reference songs and have different frame

alignment (and different features). The rest 44 quires corresponding to the two-version songs are extracted from the version different from the one in the database. Noise is also added to each query song and the signal to noise ratio is 20 dB.

The songs are in single-channel wave format, 16 bit per sample, and the sampling rate is 22.05 KHz. The direct current component is removed and the music is normalized with the maximum value equaling 1. Each audio data is divided into overlapped frames. Each frame contains 1024 samples and the adjacent frames have 50% overlapping. Each frame is weighed by a hamming window and further appended with 1024 zeros to fit the length of FFT (2048 point). Like [5], the spectrum from 100 Hz to 2000 Hz is used as the feature. Accordingly, each feature has the size 177. We use several hash instances, each having 128 entries. In E^{2}LSH the dimension of the sub-feature is chosen to be $m = 8$ by some initial experiment result.

In our retrieval system each song in the database is accompanied with its feature sequence. When the system boots the hash tables are constructed in the memory. Therefore, utilization of LSH/E^{2}LSH has no extra disk storage requirement though it does require some memory to hold the hash tables.

Figure 6 gives a query-by-content music information retrieval demo system developed under the matlab environment. The query has two sources, either the recording via a microphone or those in the query list (on the left side). From the right side the spectral feature, matching method and LSH scheme can be selected. The ranked retrieval list gives the top-4 results, with the best matching song as the first. Both the query and retrieval result can be played to confirm the correctness of the retrieval.

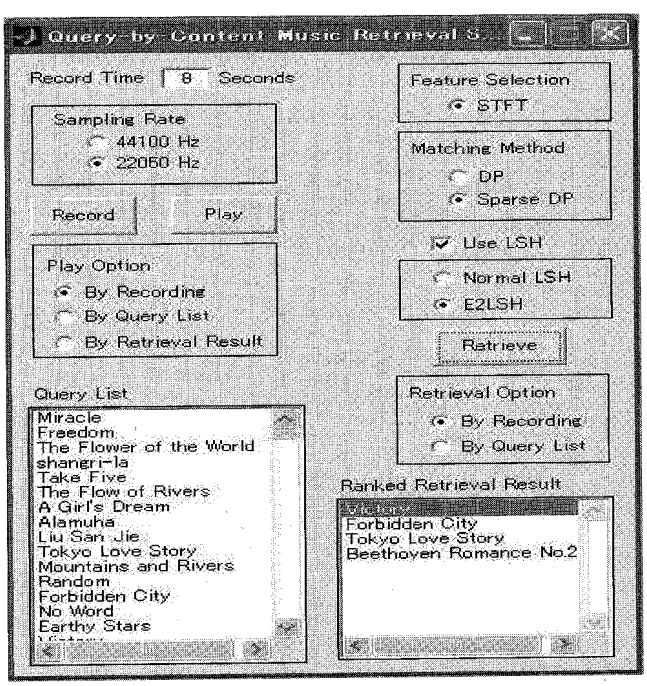

Fig. 6 Demo system.

In the current setting on the figure "Sparse DP" of "Matching Method", "Use LSH" and "E^{2}LSH", "By Recording" of "Retrieval Option" are set respectively. The "Ranked Retrieval Result" part shows 4 songs that best match the recorded 8s-long piece of the query song called "victory", with the desired song ("victory") appearing in the top.

4.1 Evaluation Metrics

For the purpose of providing a thorough understanding of our music retrieval mechanism, four different schemes are examined and compared on the basis of the general framework. In the experiment, we mainly consider three metrics that can evaluate every scheme roundly.

1) Matched percentage. Since LSH/E^{2}LSH is used to avoid pairwise comparison, the first question always touch on how much it can reduce the computation. Figure 7 shows 4 parameters.

- N_{tm}: the number of total features in a reference song.
- N_{dm}: the number of directly matched features in a reference song with LSH/E^{2}LSH before filtering.
- N_{mm}: the number of features of the matched part in the desired reference song. The desired song is known in the preparation of the query. By applying normal DP (without hash) to compare the query with its desired song we can learn the start and end point of the query in the reference song, and then N_{mm}.
- N_{rm}: the number of remaining features of matched part in the desired reference song after the filtering stage in LSH/E^{2}LSH.

Further let N_{fm} represent the number of remaining ones of the N_{dm} features after filtering. From these parameters we define three ratios N_{dm}/N_{tm}, N_{fm}/N_{tm} and N_{rm}/N_{mm} as Roughly Matched Percentage (RMP), Filtered Matched Percentage (FMP) and Valid Match Percentage (VMP) respectively. RMP reflects how much pairwise comparison can be reduced, FMP is associated with the potential reduction of computation in the sequence comparison stage and VMP affects the retrieval accuracy. With a good design of the hash functions one will expect a low RMP/FMP and a high VMP.

2) Computation time. We will compare the filtering

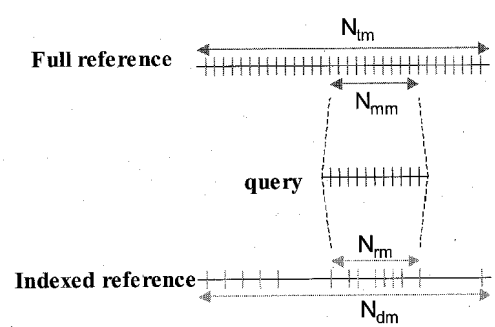

Fig. 7 Matched part of the desired reference.

time between LSH and E^{2}LSH, and the sequence comparison time between DP and SDP. The retrieval time of a query is the sum of its hash value calculation time, its filtering time and its sequence comparison time.

3) Retrieval accuracy. In our experiment each of the queries is used to retrieve the desired song from the database. The retrieval accuracy is defined as the ratio of the number of correctly retrieved songs to that of the total queries, as shown below:

$$\frac{\#\text{ queries with desired songs in ranked list}}{\#\text{queries}} \times 100\% \quad (11)$$

4.2 Matched Percentage

In the conventional CBMR system pairwise comparison is done by the DP procedure. In our scheme LSH/E^{2}LSH reduces most of the pairwise comparison. Its effectiveness in computation cost reduction is reflected in RMP/FMP. Figure 8 shows RMP under different number of hash tables. The estimation of RMP is simply (number of hash tables) / (number of buckets in each hash table). The curves of LSH, E^{2}LSH and the estimation match very well. This reflects that the hash values of all the reference features almost evenly distribute over all the buckets, which is the desired case.

LSH/E^{2}LSH reduces most of the pairwise comparison. However, the features with the same hash value are not necessarily similar to the query feature. The filtering is done just after indexing the hash tables to keep only the near neighbors of the query. The filtering threshold δ in Eq. (3) plays an important role. It determines how many features will remain, which in turn affects the computation and the retrieval accuracy.

Table 1 shows the normalized FMP, the ratio of FMP to RMP achieved at three hash instances in Fig. 8. The normalized FMP actually equals N_{fm}/N_{dm}, the ratio of the number of remaining features after filtering to the number of features found by LSH/E^{2}LSH before filtering. Due to the limited bucket entries, many non-similar features fall in the same bucket as the query features. Table 1 indicates that by setting the filtering threshold most of the non-similar features can be filtered out and a low FMP can be achieved in both LSH and E^{2}LSH. FMP increases as δ does, i.e., more features remain after filtering with a loose threshold and the subsequent sequence comparison takes longer time. Table 2 shows VMP under different filtering threshold for LSH (Table 2 (a)) and E^{2}LSH (Table 2 (b)). A bigger δ leads to a higher VMP. But it also results in a larger RMP/VMP which is associated with heavy computation. By selecting a suitable δ for the filtering stage, a low RMP/FMP can be achieved while VMP is maintained at a certain level. It is shown later that even a moderate VMP can achieve a high retrieval accuracy. Hereafter unless otherwise specified δ_{LSH} is set to 0.03 and δ_{E^2LSH} is set to 0.0075.

Figure 9 reveals that the increase of hash tables only results in a little gain in VMP. Therefore in the following three hash tables are the default setting.

4.3 Computation Time

LSH/E^{2}LSH hash table construction is usually time-consuming. Fortunately this is done before the actual query takes place. The hash value of the query is calculated just before retrieval. For a short query this time is almost negligible.

The filtering stage in LSH/E^{2}LSH is quite different since the distance is directly calculated with the high dimensional feature in LSH while in E^{2}LSH the distance is calculated with the sub-features of a low dimension. Filtering takes much time, especially when the feature has a big size in the LSH scheme. With E^{2}LSH the reduction of

Table 1 Normalized FMP under different filtering threshold (3 hash tables).

(a)	δ_{LSH}	0.01	0.02	0.03	0.04	0.05
	LSH	0.0185%	0.121%	0.590%	3.56%	12.03%
(b)	δ_{E^2LSH}	0.0025	0.005	0.0075	0.01	0.0125
	E^{2}LSH	0.0224%	0.145%	0.606%	1.93%	4.78%

Table 2 VMP under different filtering threshold (3 hash tables).

(a)	δ_{LSH}	0.01	0.02	0.03	0.04	0.05
	VMP_{LSH}	0.113	0.255	0.400	0.537	0.669
(b)	δ_{E^2LSH}	0.0025	0.005	0.0075	0.01	0.0125
	VMP_{E^2LSH}	0.123	0.240	0.363	0.472	0.573

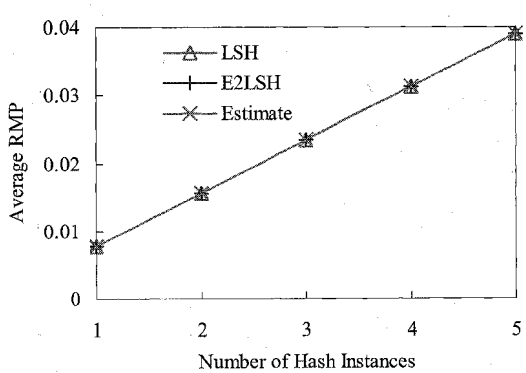

Fig. 8 RMP under different number of hash tables.

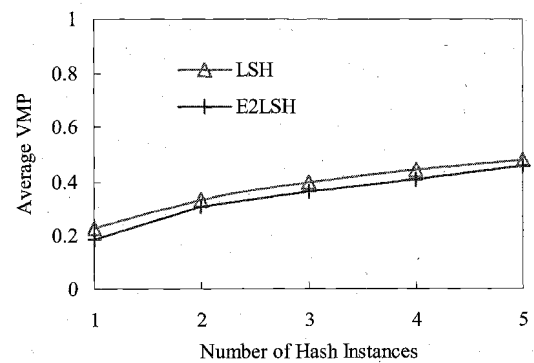

Fig. 9 VMP under different number of hash tables (δ_{LSH} = 0.03, δ_{E^2LSH} = 0.0075).

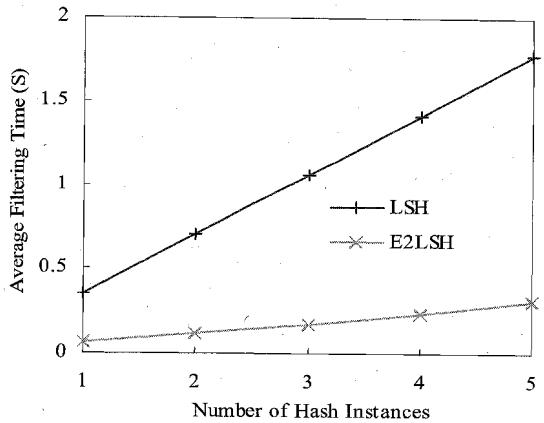

Fig. 10 Filtering time in LSH and E^2LSH.

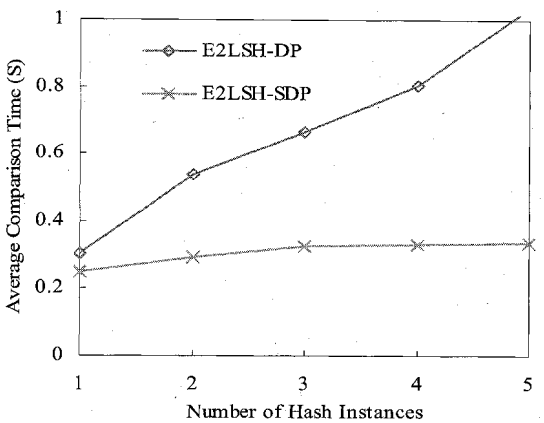

Fig. 11 Sequence comparison time in DP and SDP under different number of hash tables ($\delta_{E^2LSH} = 0.0075$).

feature dimension greatly decreases the filtering time. Figure 10 shows that with more hash tables, the filtering time in both LSH and E^2LSH increases, but at different rate, which indicates that E^2LSH is more suitable for retrieval with high-dimensional features.

Two sequence comparison methods are used in our schemes. Both the number of hash tables and the filtering threshold affect the comparison. Figure 11 shows the sequence comparison time with respect to the number of hash instances and Fig. 12 shows the similar results under different filtering threshold δ. When few reference features are matched, DP and SDP almost have the same retrieval speed. This occurs when there are few hash instances in Fig. 11 or the filtering threshold is low in Fig. 12. As the number of remaining features increases, SDP has very obvious superiority over DP since it avoids the calculation of feature distance and its sequence comparison time approaches a steady value in Figs. 11-12, which guarantees the worst-case retrieval time. This can be explained as follows.

In DP the query is compared with the partial reference sequence over a DTW table of the size (length of query) × (length of the partial reference). The DTW table becomes smaller in case few reference features remain. However DP involves the calculation of pairwise distance among the remaining features. In contrast, in SDP, the feature distance is taken from the filtering stage. But in the filtering stage SDP does not know the final length of the partial reference. Then it constructs a DTW table of full size (length of query) × (length of the reference) so that the weight calculated in Eq. (7) can be directly filled in. As the number of hash instances increases in Fig. 11 or the filtering threshold increases in Fig. 12, the number of VMP also increases, however in a non-linear way. The biggest gain of VMP is obtained by going from single hash instance to two hash instances, or the filtering threshold going from 0.0025 to 0.005. Further increase of hash instance or filtering threshold yields diminishing returns in terms of VMP and causes little increase in the computation cost in Eq. (7). Meanwhile the best path search over a full-size DTW takes a fixed time in Eq. (8). Then in a total the sequence comparison time of SDP approaches a steady value. Therefore SDP is preferred when there are more hash instances or the filtering threshold is large.

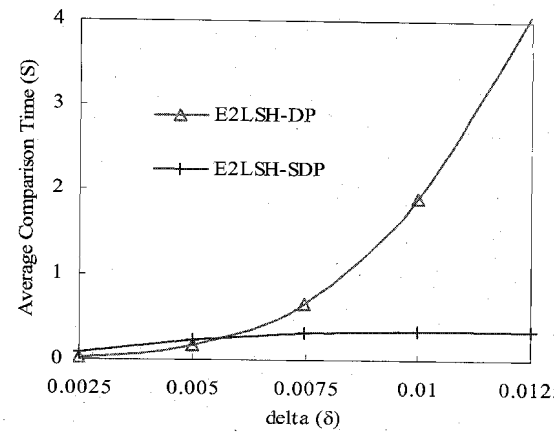

Fig. 12 Sequence comparison time in DP and SDP under different δ (3 hash tables).

Table 3 Average retrieval time under different schemes.

Scheme	LSH –DP	LSH –SDP	E^2LSH –DP	E^2LSH –SDP	DP
Time(s)	1.56	1.28	0.84	0.51	21.7

To show the effect of hashing, Table 3 lists the average retrieval time consumed for each query under the different schemes. The conventional DP (without hashing) takes 21.7 s. In comparison, E^2LSH-SDP reduces the time to 0.51 s, accelerating the retrieval speed by 42.7 times.

4.4 Retrieval Accuracy

LSH/E^2LSH was initially proposed to retrieve from a database by single feature. To acquire a high retrieval accuracy, many hash tables are required, which increases the filtering time. In our scheme, LSH/E^2LSH is used for audio sequence comparison. Even though the retrieval accuracy of a single feature is not very high, the following sequence comparison effectively removes the unsimilar se-

Table 4 Top-4 retrieval accuracy with respect to the number of hash instances (δ_{LSH} = 0.03, δ_{E^2LSH} = 0.0075).

# hash table	1	2	3	4	5
LSH-DP	0.89	0.90	0.92	0.93	0.95
LSH-SDP	0.90	0.91	0.91	0.94	0.95
E^{2}LSH-DP	0.91	0.92	0.92	0.94	0.95
E^{2}LSH-SDP	0.91	0.92	0.93	0.93	0.94

Table 5 Top-4 retrieval accuracy of LSH/E^{2}LSH (3 hash tables).

(a)	δ_{LSH}	0.01	0.02	0.03	0.04	0.05
	LSH-DP	0.83	0.88	0.92	0.91	0.93
	LSH-SDP	0.86	0.89	0.91	0.92	0.94
(b)	δ_{E^2LSH}	0.0025	0.005	0.0075	0.01	0.0125
	E^{2}LSH-DP	0.86	0.89	0.92	0.93	0.93
	E^{2}LSH-SDP	0.88	0.91	0.93	0.93	0.94

Table 6 Top-t retrieval accuracy with respect to t (δ_{LSH} = 0.03, δ_{E^2LSH} = 0.0075, 3 hash tables).

Top-t	1	2	3	4
LSH-DP	0.71	0.84	0.89	0.92
LSH-SDP	0.68	0.82	0.88	0.91
E^{2}LSH-DP	0.69	0.82	0.88	0.92
E^{2}LSH-SDP	0.69	0.83	0.89	0.93

quences. Therefore, in our retrieval system, a few hash tables are sufficient to achieve high retrieval accuracy. Table 4 shows that the retrieval accuracy is satisfactory with mere 3 hash tables.

Table 5 shows the retrieval accuracy under different filtering threshold δ. It is obvious that δ_{LSH} = 0.03 and δ_{E^2LSH} = 0.0075 are suitable thresholds since lower δ decreases retrieval accuracy while larger δ increases the computation cost. Table 6 shows the Top-t retrieval accuracy with δ_{LSH} = 0.03, δ_{E^2LSH} = 0.0075, and 3 hash tables. t is the length of the ranked list. When the desired song appears in the ranked list, the retrieval is regarded as successful. According to Table 6, the four schemes almost have the same retrieval accuracy and the retrieval accuracy increases as t does. The Top-4 retrieval accuracy is relatively satisfactory.

5. Conclusions

We have established an index-based query-by-content framework for music information retrieval, proposed and evaluated four different retrieval schemes. In our system the retrieval speed is accelerated in three ways: LSH reduces the pairwise comparison; E^{2}LSH further reduces the filtering time by applying locality sensitive dimension reduction; SDP decreases the comparison time by avoiding pairwise distance computation in the sequence comparison stage.

From the extensive simulation results it is obvious that E^{2}LSH-SDP is the optimal choice. We also show that even with only a few hash tables and relatively low filtering threshold, the retrieval with a sequence still has high accuracy.

References

[1] N. Bertin and A. de Cheveigne, "Scalable metadata and quick retrieval of audio signals," ISMIR 2005, pp.238–244, 2005.
[2] I. Karydis, A. Nanopoulos, A.N. Papadopoulos, and Y. Manolopoulos, "Audio indexing for efficient music information retrieval," MMM'05, pp.22–29, 2005.
[3] J.-Y. Won, J-H. Lee, K. Ku, J. Part, and Y.-S. Kim, "A content-based music retrieval system using representative melody index from music databases," Computer Music Modeling and Retrieval: Second International Symposium, CMMR 2004.
[4] J. Reiss, J.-J. Aucouturier, and M. Sandler, "Efficient multidimensional searching routines for music information retrieval," 2nd ISMIR, 2001.
[5] C. Yang, "Efficient acoustic index for music retrieval with various degrees of similarity," ACM Multimedia, pp.584–591, 2002.
[6] 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," ISMIR2005.
[7] J.S.R. Jang and H.R. Lee, "Hierarchical filtering method for content-based music retrieval via acoustic input," Proc. Ninth ACM International. Conference on Multimedia pp.401–410, 2001.
[8] R.B. Dannenberg and N. Hu, "Understanding search performance in query-by-humming systems," Proc. ISMIR 2004, pp.236–241, 2004.
[9] M. Datar and N. Immorlica, "Locality-sensitive hashing scheme baesd on P-stable dostributions," Proc. Symposium on Computational Geometry, pp.253–262, 2004.
[10] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," Proc. 30th ACM STOC, 1998.
[11] J. Buhker, "Efficient large-scale sequence comparison by locality seneitive hashing," Bioinformatics vol.17, no.5, pp.419–428, 2001.
[12] S. Hu, "Efficient video retrieval by locality sensitive hashing," ICASSP 2005, pp.449–452, 2005.
[13] P. Indyk and N. Thaper, "Fast color image retrieval via embeddings," Workshop on Statistical and Computational Theories of Vision (at ICCV), 2003.
[14] LSH Algorithm and Implementation (E^{2}LSH) http://web.mit.edu/andoni/www/LSH/index.html
[15] Y. Yu, C. Watanabe, and K. Joe, "Towards a fast and efficient match algorithm for content-based music retrieval on acoustic data," ISMIR 2005, pp.696–701, 2005.
[16] Y. Yu, M. Takata, and K. Joe, "Index-based similarity searching with partial sequence comparison for query-by-content audio retrieval," Proc. LSAS06, 1st Workshop on Learning Semantics of Audio Signals, pp.76–86, Athens, Greece, Dec. 2006.
[17] Y. Yu, M. Takata, and K. Joe, "Similarity searching techniques in content-based audio retrieval via hashing," Proc. MMM07, 13th International Multimedia Modeling Conference, Singapore, Jan. 2007. Springer LNCS4351, vol.1, pp.397–407, 2007.
[18] K. Kashino, G. Smith, and H. Murase, "Time-series active search for quick retrieval of audio and video," Proc. ICASSP, vol.VI, pp.2993–2996, 1999.

Yi Yu is a Ph.D. student at Graduate School of Humanity and Science, Nara Women's University. She received Bachelor's degree of Engineering from University of Science and Technology of China in 2001 and Master's degree of Science from Nara Women's University in 2006 respectively. Her research interests include Multimedia Database Organization, Index-based Query-by-Content Audio Information Retrieval and Audio Content Representation and Audio Annotation.

Kazuki Joe received the B.S. in Mathematics from Osaka University in 1984, and M.S. and Ph.D. in Information Science from Nara Institute of Science and Technology in 1995 and 1996, respectively. He is currently a Professor at Nara Women s University. From 1984 to 1986, he was a Software Engineer of Japan DEC. From 1986 to 1990, he was a Researcher of ATR Auditory and Visual Perception Research Lab. From 1991 to 1993, he was a Senior Researcher of Kubota corporation. From 1996 to 1997, he was an Assistant Professor at Nara Institute of Science and Technology. From 1997 to 1999, he was an Associate Professor at Wakayama University. His research interests include parallel computer architectures, analytic modeling for parallel computers, parallelizing compilers, neural networks, image processing and multimedia.

J. Stephen Downie is an Associate Professor at the Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign (UIUC). He earned his MLIS (1993) and Ph.D. (1999) in Library and Information Science from University of Western Ontario. Professor Downie is Director of the International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL). He is Principal Investigator on the Human Use of Music Information Retrieval Systems (HUMIRS) and the Music-to-Knowledge (M2K) music data-mining projects. He has been very active in the establishment of the Music Information Retrieval and Music Digital Library communities through his ongoing work with the ISMIR series of MIR conferences as a member of the ISMIR steering committee. His research interests include design and evaluation of IR systems, multimedia and music information retrieval, digital library design and Web-based music analysis technologies. Dr. Downie's research is supported by the Andrew W. Mellon and the National Science Foundation (NSF) under Nos.IIS-0340597 IIS-0327371.