Index-based Similarity Searching with Partial Sequence Comparison for Query-by-Content Audio Retrieval

Yi Yu, Masami Takata, and Kazuki Joe

Graduate School of Humanity and Science, Nara Women's University
Kitauoya nishi-machi, Nara 630-8506, Japan
{yuyi,takata,joe}@ics.nara-wu.ac.jp

Abstract. An efficient representation and organization of acoustic-based music data sequences can help to accelerate the retrieval procedure. In this paper we investigate suitable indexing structures applicable to audio content, depict musical objects representations from NoteSet to spectrum, and consider the design of hashing functions to organize the spectral features. Going on this premise, we present a fast and efficient index-based audio searching algorithm with Partial Sequence Comparison (PSC) for query-by-content music retrieval. Specifically, the features of the reference melodies are organized in terms of the hash function where the similar features collide with a high probability. With a query melody, only a few of the reference features in the database are indexed out. Fewer reference features remain after filtering. They are compared with the query by the proposed PSC method. Experimental results show that the proposed approach achieves high accuracy and the retrieval speed is more than ten times faster than the conventional Dynamic Programming (DP).

1 Introduction

Acoustic data contains the audio information and exhibits regular spectral characteristics, hence many spectral features, such as Mel-Frequency Cepstral Coefficients (MFCC) [1][2], Linear Predictive Coding (LPC) coefficients [3], Fourier Transform [5][6] and so on, have been proposed to extract salient information and diminish redundancy. By analyzing the music signal, the acoustic methods can be applied to any melody, especially the polyphonic ones. Also, it provides information for high-level Content-Based Music Information Retrieval (CBMIR) applications, e.g., musical genre classification [1], acoustic timbre similarity [2] and singer identification [3]. Interested readers may refer to [7] for a review of comparison and matching of content-based audio retrieval.

Query-by-content audio retrieval based on spectral similarity is usually difficult due to the high dimensionality of features, complex computation, and the large database size. These characteristics compound the problem of the time consumption for music information retrieval. One important thing in CBMIR

is how to make effective use of the features in the service of scalable matching and retrieval. To do this, many audio indexing techniques are emerging in the research field of CBMIR, for example, M-trees [4], hierarchical structure [5], R-trees [6]. As far as creation of an index-based spectral similarity retrieval system for audio content is concerned the main challenges are as follows: (1) How to characterize a corpus of acoustic objects with a corpus of relevant spectral features. (2) How to represent audio features so that they can be indexed. (3) How to locate the desired music segments with a given query in the format of acoustic sequences within the acceptable time.

One of the basic problems in audio retrieval is to design appropriate metrics and algorithms to reduce pairwise comparisons of feature sequences, consequently, provide a quick answer closest to query. Here we will start with an interesting direction in the study of acoustic sequences matching without comparing a query to each object from the database. We focus on the music performances of a captivating concert, where a popular 12-girl-band plays Chinese folk musical instruments mainly including lute, alto fiddle, dulcimer, wind instrument etc. Based on such an interesting database we study audio similarity index and retrieval, and scalable content-based searchability. Then we propose an index-based spectral similarity searching with Partial Sequence Comparison (PSC) based on Locality Sensitive Hashing(LSH) [8], to effectively compare the query sequence against the indexed partial sequences, which extends our earlier research [9] with much improvement in retrieval time and scalability. LSH is used to organize spectral feature sequences while PSC is used to match the acoustic sequences and obtain the answer closest to the query.

The rest of the paper is arranged as follows: Section 2 presents the approach for audio indexing, addresses feature selection, extraction, and organization, and details the retrieval procedure with LSH and PSC schemes. Section 3 lists the simulation environment and analyzes the experiment results. Finally Section 4 concludes the paper.

2 The Proposed Approach

Our work pays the most attention to providing fast and efficient content-based audio retrieval mechanisms that can be carried out by spectral similarity measurement depending on a suitable indexing structure. The goal is to hash the spectral features that are described in section 2.2, using several hash functions so as to make sure that, for each function, the collision probability of features that are close to each other is much high while the features far apart have a low collision probability. Our retrieval system consists of four main parts: selecting suitable audio feature, extracting feature, creating indexable acoustic structure and matching acoustic sequences.

The similarity measurement for audio content retrieval is executed according to the following procedure: given a set of n musical reference melodies, actually, which can be represented by the frames $R = \{r_{i,j} : r_{i,j} \in R_i,\ 1 \leq i \leq n, 1 \leq j \leq |R_i|\}$ where $r_{i,j}$ is the j^{th} spectral feature of the i^{th} reference piece R_i, are

located in a high-dimension Euclidean space (U, d) with a distance metric d. A query melody is also broken into a sequence of frames $q_1, q_2, ..., q_Q$. LSH is used to effectively pick up reference candidates resembling the query sequence. In terms of locality sensitive function $H(\cdot)$, each query frame q_m matches with a high probability some resemblances, $S_{i,m} = \{r_{i,j} : r_{i,j} \in H(q_m),\ d(q_m, r_{i,j}) \leq \delta\}$, stored in the audio buckets $H(q_m)$. Then the partial audio sequences of the i^{th} reference in the union $\bigcup_m S_{i,m}$ are reorganized and compared with the query by the proposed PSC scheme.

2.1 Feature Selection

Spectral analysis is a technique that has been verified to be useful in determining the qualitative and quantitative property of similarity. Short Time Fourier Transformation (STFT) is one of the basic representations of music signals. In most applications, only the amplitude of STFT is used. If the time scale is large, the spectrum amplitude can not reflect the time variant details. On the other hand, a short frame results in low frequency resolution.

When spectral similarity is adopted as the criterion for CBMIR, the spectral profile is the most important. Parametric spectrum usually has the same capability in describing the spectral structure with smaller dimensionality and less sensitivity to frequency resolution in comparison with STFT. Usually pitch (fundamental frequency) is extracted as the feature in the melody retrieval, which need to be supported by a nice transcription technique. However, the transcription of general polyphonic signal is quite difficult and is still under research though monophonic melodies can be transcribed with much accuracy.

We have witnessed the great success of applying MFCC in speech recognition. MFCC is usually calculated from STFT. According to Mel-frequency scaling filter banks, STFT is grouped into spectrum bins, which represent a rough spectral structure. From the spectrum bins MFCC is calculated. As a result, MFCC can represent the rough spectral structure, and take the place of STFT in the task of retrieving musical objects. Here we try to use MFCC as the feature to implement spectrum-based melody similarity retrieval.

2.2 Feature Extraction

Dissimilar to other kinds of audio, music has strong descriptive composition. Score always implies the music theme, which may appear anywhere in the music. The score is composed of notes. For the simplicity of description, we define NoteSet as the simultaneously initiated notes, either a single note or the combination of two or more notes. It is quite a reliable representation for music. No matter what a NoteSet plays, its sound usually lasts from tens to hundreds of milliseconds. For either human songs or instrument-generated music, the signal spectrum is relatively stable within a NoteSet while a transition happens between two adjacent NoteSet.

In other words, the adjacent frames in the same NoteSet are highly correlated and most of them are redundant in the sense of spectral similarity retrieval.

The frame merging is implemented by setting the Spectral Correlation (SC) threshold as suggested in our previous work [9]. Therefore, only the necessary spectral features are kept. The remaining frames can represent the salient feature of audio information. This is an important specification of our spectral similarity retrieval. The merged frames are regarded as searchable symbols in our system. So we can say that audio buckets store the compressed audio features not only existing as available points in high dimensional space but also capturing the salient property of audio information. As far as feature extraction is concerned this is also the significant difference in comparison with other methods [1]-[6].

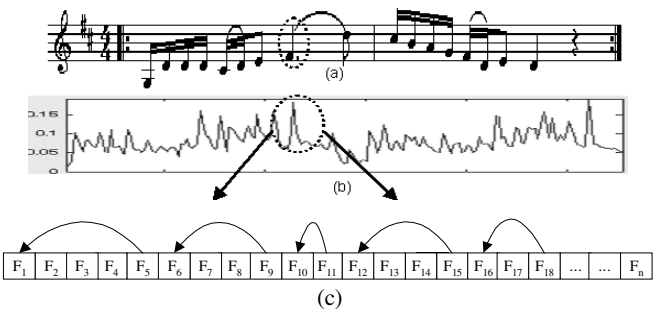

Fig. 1. Representation of a musical melody from NoteSet to spectral features.

Fig.1 is an example dealing with different aspects of a melody, from score to spectrum. The musical piece is taken from the Chinese folk, Alamuhan. Here a NoteSet contains one note. The analysis is similar when a NoteSet consists of multiple notes. Fig.1(a) presents the score of Alamuhan and Fig.1(b) is the corresponding energy profile calculated from the original waveform signal. Then SC between adjacent frames is calculated. Fig.1(c) shows the frames merging within NoteSets by setting a suitable SC threshold, for example, F_1,F_2,F_3,F_4 and F_5 can be merged into one frame.

2.3 Feature Organization and Audio Features Filtering with LSH

We believe that a nice design of feature sequence organization will be as valuable as feature extraction in information retrieval. This is because it can facilitate the access to the database and minimize the search time. When we try to filter audio sequences data by the aid of LSH, in the preprocessing stage these audio sequences firstly have to be divided into the small frames, for which MFCC is calculated and regarded as the feature in the high dimensional space. When LSH is adopted, a feature is directly quantified and its hash value for each hash table is calculated independently.

As was mentioned in [9], multiple hash tables increase the retrieval ratio of LSH. A group of LSH hash tables are simple and effective data storage struc-

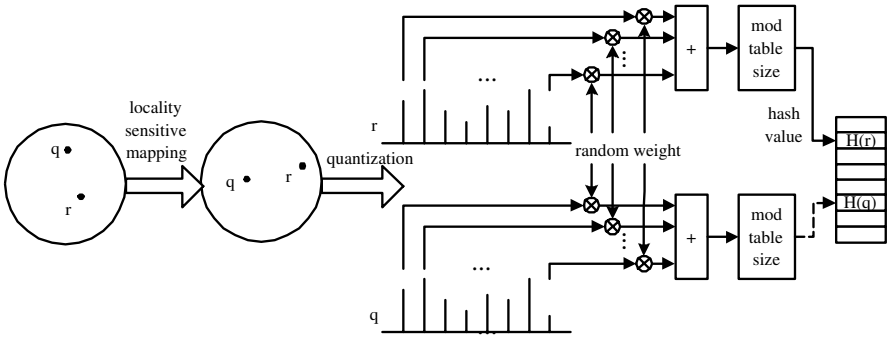

Fig. 2. Feature organization and indexing in a LSH table.

tures, which can be accessed randomly and implemented in parallel. Therefore we construct several hash tables, each containing all the features of the references.

Fig.2 shows a hash instance. Different locality sensitive mapping amplifies the difference between two features from different aspects. If two features are really similar, it is probable that they collide in at least one of the hash instances. The per-dimension quantization guarantees that two features with a short distance are quantified to the same integer sequence and generate the same hash value with a high probability. A feature (r) of the reference melody is stored in a bucket matching its hash value $(H(r))$. The random weight makes the features of reference melodies almost evenly distributed, which is expected. A query feature q indexes the potential similar features located at $H(q)$.

The k^{th} hash instance has its own hash function $H_k(\cdot)$. In the following, $H_k(\cdot)$ also means an audio bucket storing all the features with the same hash value. Its meaning is obvious from the context. The j^{th} spectral feature of the i^{th} reference melody, $r_{i,j}$, is stored in $H_k(r_{i,j})$, a bucket of the k^{th} hash table. Its melody number i and the corresponding time offset j are recorded together with the feature, for the purpose of providing facilities for reconstruction of the partial acoustic sequences after the filtering stage.

In the query stage, a sequence of query frames $q_1, q_2, ..., q_Q$ is used to find the closest reference melody. With a query frame q_m, the candidate reference frames in the bucket of the k^{th} hash table, $H_k(q_m)$, can be obtained. This bucket contains all the frames matching the same hash value. Though it is probable that the resemble frames lie in the bucket, many other non-similar frames also exist due to the limited hash table size. It is necessary to remove these non-similar frames so as to reduce the post computation. Therefore we define a distance function that can quantify the similarity degree, $d(X,Y) = \|X-Y\|_2 / \|X\|_2 \cdot \|Y\|_2$, the normalized Euclidean distance. Among the indexed frames in the candidate bucket, the ones with a distance to q_m greater than δ are filtered out and discarded by Eq. 1, namely, we would like to retain such frames that lie within the

ball centered at q_m with a radius δ.

$$S_{k,i,m} = \{r_{i,j} : r_{i,j} \in H_k(q_m),\ d(q_m, r_{i,j}) \leq \delta\} \tag{1}$$

The union $S_{k,i} = \underset{m}{\cup} S_{k,i,m}$ gives the candidate features of the i^{th} reference obtained from the k^{th} hash table for the whole query sequence.

2.4 Matching of Audio Sequences

To improve the retrieval ratio, several hash tables are used. The total candidates of the l^{th} reference melody obtained from all the hash tables are $S_l = \underset{k}{\cup} S_{k,l}$. Of the frames with duplicate timing offset, usually only a single copy is kept. With each candidate matching pair in S_l, the feature sequence of the l^{th} reference melody is $r_{l,j_1}, r_{l,j_2}, ..., r_{l,j_R}$, reorganized in the ascending of the time offset $j_1, j_2, ..., j_R$. Then the conventional Dynamic Programming (DP) [10]-[12] can be applied.

Assume that $D_l(m, j_n)$ is the minimum distance between the query and the l^{th} reference melody, beginning from the leftmost side $(1, j_1)$ of the DTW table to the current position (m, j_n). Equation Eq. 2 gives the recursive relation

$$D_l(m, j_n) = d(q_m, r_{l,j_n}) + \min \begin{cases} D_l(m-2, j_{n-1}) \\ D_l(m-1, j_{n-1}) \\ D_l(m-1, j_{n-2}) \end{cases} \tag{2}$$

which shows that the optimal path exists only when the query input is within half to twice the size of the reference. This constraint is very possible because most of the tempo variation is removed by frame merging. The remaining frames almost have the same timing

$$D_l = \min_{j_n} D_l(Q, j_n). \tag{3}$$

Then among the distances between the query and all the references, the following equation gives the desired melody

$$l = \arg \min_{l}\ D_l. \tag{4}$$

In the real system, usually several retrieval results are given for a single query, ranked in the order of D_l, with the best matching reference melody at the top.

However, the conventional DP may not be efficient since most of the matching pairs are missing after the LSH filtering, and the remaining matching pairs are sparsely distributed over the Dynamic Time Warping (DTW) table. We propose a Partial Sequence Comparison (PSC) scheme to compare the query against the partial reference sequences. This is different from the conventional DP: we directly utilize the distance calculated in the filtering stage by filling the distance into a DTW table. For each matching pair $< q_m, r_{l,j_n} > \in S_l$ obtained from k^{th}

hash table, the reverse of its distance is used as the weight in the matching procedure.

$$w_k(q_m, r_{l,j_n}) = \min\{\delta / d(q_m, r_{l,j_n}),\ w_{\max}\} \tag{5}$$

For a matching pair, its 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 $< q_m, r_{l,j_n} >$ does not exist, its weight is set to 0.

The sequence comparison is to select the path that maximizes the total weight. Despite the absence of most matching pairs, the matching path still contains as many points as possible. The maximum weight is found by the iteration in Eq. 6

$$W_l(m, j_n) = \sum_k w_k(q_m, r_{l,j_n}) + \max \begin{cases} W_l(m-2, j_{n-1}) \\ W_l(m-1, j_{n-1}) \\ W_l(m-1, j_{n-2}) \end{cases} \tag{6}$$

In this equation, the weight of the duplicates from different hash tables are summed since the recurrence of a single pair in different hash tables means that the pair is a suitable match with a high probability. For the reference melody, its weight is obtained by Eq. 7. Then by Eq. 8 the one with the maximum weight matches the query.

$$W_l = \max_{j_n} W_l(Q, j_n) \tag{7}$$

$$l = \arg \max_l\ W_l \tag{8}$$

3 Experiments and Results

The experiments have been carried out on the acoustic database with polyphonic melodies. Altogether there are 111 melody pieces in our database, all played by a popular 12-girl-band. These reference melodies have the same performance of music style, i.e., the melody pieces are similar to each other in music color. This database can be used to evaluate the effectiveness and the robustness of our approach. Each reference melody is segmented into 60-second-long melodic slip. Query melodic samples are segmented into 6-8 seconds long.

In our approach both the query music and reference melodies in the database are converted to single-channel 16-bit wave format and re-sampled to the rate of 22.05kHz. Each piece of melody is divided into overlapped frames, each containing 1024 samples with the adjacent frames having 50% overlapping. Each frame is weighted by a hamming window, which is further appended with 1024 zeros to fit the length of FFT. Then MFCC is calculated as follows: the bins of the amplitude spectrum are grouped and smoothed by the filter banks constructed according to the perceptually motivated Mel-frequency scaling; the log value of the resulting vector is further transformed by DCT in order to make the MFCC coefficients uncorrelated.

We use several hash instances, each having 128 entries. Two schemes of spectrum-based sequences comparison, the conventional DP and the proposed PSC, are evaluated. For the purpose of providing a thorough understanding of our music retrieval mechanism, in the experiment we examine our approach mainly from two aspects: computation time and retrieval ratio. Our retrieval mechanism involved two key parameters: the SC threshold ρ and the filter threshold δ. Calculation of the former refers to our previous work [9]. The latter is discussed in section 2.3. These two parameters directly affect the performance of the retrieval system.

3.1 Computation Time

During the evaluation of LSH-PSC we consider the computation time from three aspects: (1) Building the hash table. (2) Filtering the features with LSH. (3) Comparing feature sequences. The construction of LSH hash tables is usually time-consuming, however, this operation is done before the actual query takes place. The hash value of a query melody is calculated just before starting the searching procedure. For such a short query, this time is also negligible. The operation of LSH filtering is time-consuming in our retrieval scheme. The time linearly increases as the number of hash instances does and is not affected by the filtering threshold δ. Fortunately, frame merge can effectively decrease the total features of the references and the query, thus reducing both filtering time and sequences comparison time. That is, a good experimental value ρ can help to quicken the filtering features and sequence comparisons. The consumed time in the sequence comparisons is decided by the number of matching pairs, thus is strongly associated with the filtering threshold δ. Namely, a big filtering threshold leads to more matching pairs and a higher retrieval ratio, but at the cost of more computation.

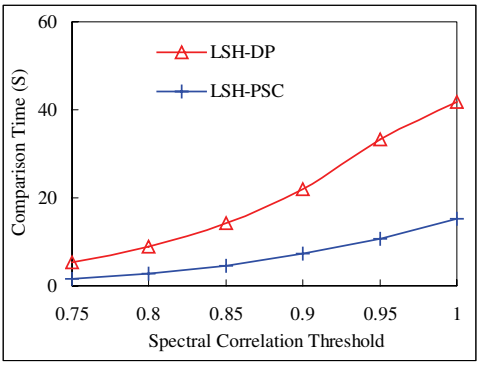

Fig. 3. Sequence comparison time in DP and PSC under different SC threshold ρ. ρ=1.0 means no frame merge (3 hash tables, LSH filtering threshold δ=0.5).

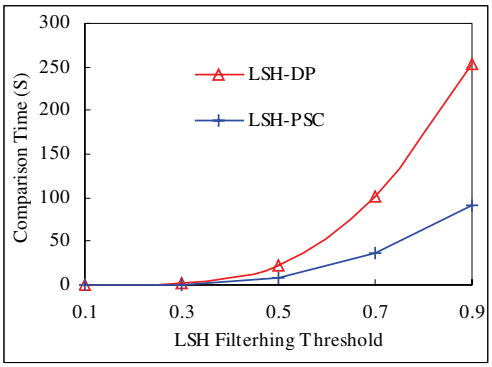

Fig. 4. Sequence comparison time in DP and PSC under different filtering threshold δ (3 hash tables, ρ=0.9).

The effectiveness of PSC against DP is verified by the total sequence comparison time of all the queries in this section. Fig. 3-4 show the total comparison time of 111 retrievals at different SC threshold ρ or different LSH filtering threshold δ. By merging adjacent frames, the comparison time in Fig.3 can be greatly reduced. In Fig.4 the sequence comparison time increases as the filtering threshold δ does. From both figures, when few reference frames are matched (either due to a heavy frame merge with a small ρ, or a serious LSH filtering with a small δ), the retrieval speed difference between LSH-DP and LSH-PSC is not obvious. As the matching pair increases (with a bigger ρ or δ) LSH-PSC outperforms LSH-DP with a much fast comparison speed since LSH-DP involves the calculation of pairwise feature distance among the remaining frames while in LSH-PSC, the feature distance is directly taken from the filtering stage.

Table 1. The total retrieval time consumed for 111 queries under different schemes. LSH filtering time is about 65.4s at ρ=0.9 and about 100s at ρ=1.0 (3 hash instances, LSH filtering threshold δ=0.5).

Scheme	LSH-DP (ρ=0.9)	LSH-DP (ρ=1.0)	LSH-PSC (ρ=0.9)	LSH-PSC (ρ=1.0)	DP (ρ=1.0)
Time(s)	87.6	139.2	72.8	118.7	1519.7

To show the effect of hashing, Table 1 lists the total retrieval time consumed for all the queries under the different schemes. The conventional DP (without hashing) takes 1519.7s, over 10 times than the time taken for LSH-DP (139.2s) and LSH-PSC (118.7s). The advantage of LSH-PSC over DP will be more evident with the explosion of the database. It is can be seen easily that combination of LSH and frame merge further reduces the total retrieval time by 51.6s in LSH-

DP, or 45.9s in LSH-PSC. In this table, the superiority of LSH-PSC over LSH-DP is not very obvious because the filtering takes too much time, about 65.4s at ρ =0.9 and about 100s at δ=1.0. We are building a larger database to evaluate the scalability of LSH-PSC in query-by-content audio retrieval.

3.2 Retrieval Ratio

LSH was initially proposed to retrieve from a database by single feature. To acquire a high retrieval ratio, many hash tables are required, which increases the filtering time. In our scheme, LSH is used for audio sequence comparison. Even though the retrieval ratio of a single frame is not very high, the following sequence comparison effectively removes the unsimilar melodies. Therefore, in our retrieval system, a few hash tables are sufficient to achieve a high retrieval ratio. Table 2 shows that the retrieval ratio is satisfactory with mere 3 hash tables. Table 3 shows the retrieval ratio under different filtering threshold δ. It is obvious that δ=0.5 is a suitable threshold since a smaller δ decreases retrieval ratio while a larger δ increases the computation cost. Under most of the cases the retrieval ratio of LSH-PSC is only a little less than that of LSH-DP.

Table 2. Top-4 retrieval ratio under different SC threshold ρ (3 hash tables, δ=0.5).

ρ	0.75	0.8	0.85	0.9	0.95	1.0
LSH-DP	0.928	0.955	0.982	0.982	0.991	0.991
LSH-PSC	0.856	0.892	0.937	0.946	0.964	0.982

Table 3. Top-4 retrieval ratio under different filtering threshold δ (3 hash tables, ρ=0.9).

δ	0.1	0.3	0.5	0.7	0.9
LSH-DP	0.5945	0.973	0.982	1.0	1.0
LSH-PSC	0.604	0.914	0.946	0.977	0.98

4 Conclusions

We have proposed an audio indexing approach for query-by-content music information retrieval and evaluated it, focusing on instrument-generated melody pieces that belong to the performance of the same music style (a popular band). More specifically, we studied the efficient organization of audio features by the hashing method and put forward the partial sequence comparison scheme. Compared with the conventional DP (without hash), the retrieval speed is accelerated

in three aspects: (1) Frame merge reduces the number of features of both the query and the reference melodies. (2) With LSH a single query feature only indexes few of the reference features and the full pairwise comparison is avoided. (3) Partial sequence comparison further decreases the comparison time by avoiding distance computation in the sequence comparison stage. The extensive evaluations confirm the effectiveness of the proposed algorithms. We also indicate that even with only a few hash tables and relatively low filtering threshold, the retrieval with a sequence still has a high successful ratio. We can see some possibilities to further improve the proposed approach, for example, realizing query-by-content audio retrieval in ubiquitous environments.

References

1. Tzanetakis,G.and Cook,P.:Musical Genre Classification of Audio Signals.IEEE Trans. Speech and Audio Processing, Vol.10, No.5, pp.293-302, 2002.
2. Flexer,A.and Pampalk,E., Widmer,G.:Noverty Detection Based on Spectral Similarity of Songs. Pro.ISMIR 2005.
3. Kim,Y.E. and Whitman,B.:Singer Identification in Popular Music Recordings Using Voice Coding Features. Proc.ISMIR, 2002.
4. Won,J.-Y., Lee,J-H., Ku,K., Part,J. and Kim,Y.-S.:A Content-Based Music Retrieval System Using Representative Melody Index from Music Databases. Computer Music Modeling and Retrieval, 2004.
5. Bertin,N. and Cheveigne,A.:Scalable Metadata and Quick Retrieval of Audio Signals.ISMIR 2005, pp.238-244, 2005.
6. Karydis,I., Nanopoulos,A., Papadopoulos,A.N. and Manolopoulos,Y.: Audio Indexing for Efficient Music Information Retrieval. The International MultiMedia Modeling Conference, pp.22-29, 2005.
7. Cano,P.,Batlle,E.,Kalker,T.and Haitsam,J.:A Review of Audio Fingerpring. The Journal of VLSI Signal Processing VOL.41.3,pp.271-284.
8. Indyk,P. and Motwani,R.:Approximate nearest neighbors: Towards removing the curse of dimensionality. Proceedings of the Symposium on Theory of Computing, pp.604-619, 1998.
9. Yu,Y., Watanabe,C. and Joe,K.:Towards a fast and Efficient Match Algorithm for Content-Based Music Retrieval on Acoustic Data. ISMIR 2005, pp.696-701 2005.
10. Tsai,W.H., Yu,H.M. and Wang,H.M.:A Query-by-Example Technique for Retrieving Cover versions of Popular Songs with Similar Melodies, ISMIR2005.
11. Jang,J.S.R. and Lee,H.R.:Hierarchical Filtering Method for Content-based Music Retrieval via Acoustic Input. Proceedings of the Ninth ACM International. Conference on Multimedia pp. 401-410 2001
12. Vintsyuk,T. K.:Speech discrimination by dynamic programming. Kibernetika, 4(2): 81-88.1968.