TOWARDS A FAST AND EFFICIENT MATCH ALGORITHM FOR CONTENT-BASED MUSIC RETRIEVAL ON ACOUSTIC DATA

Yi YU, Chiemi WATANABE, Kazuki JOE
Graduate school of Humanity and Science
Nara Women's University
Kitauoya nishi-machi, Nara 630-8506, Japan
{yuyi, chiemi, joe}@ics.nara-wu.ac.jp

ABSTRACT

In this paper we present a fast and efficient match algorithm, which consists of two key techniques: Spectral Correlation Based Feature Merge(SCBFM) and Two-Step Retrieval(TSR). SCBFM can remove the redundant information. In consequence, the resulting feature sequence has a smaller size, requiring less storage and computation. In addition, most of the tempo variation is removed; thus a much simpler sequence match method can be adopted. Also, TSR relies on the characteristics of Mel-Frequency Cepstral Coefficient(MFCC), where the precise match in the second step depends on the first step to filter out most of the dissimilar references with only the low order MFCC feature. As a result, the whole retrieval speed can be further improved. The experimental evaluation verifies that SCBFM-TSR yields more meaningful results in comparatively short time. The experiment results are analyzed with a theoretical approach that seeks to find the relation between Spectral Correlation(SC) threshold and storage, computation.

Keywords: Content based music retrieval, spectral correlation, dynamic programming, feature merge, pre-filtering.

1 INTRODUCTION

Acoustic data contains all the information of the music. Content Based Music Retrieval (CBMR) in acoustic form is generally the most natural but difficult due to the high dimensionality of the features, complex computation, and large database size. Therefore there is a great need to simplify acoustic-based music retrieval and provide a real-time response when the involved music is exploded in volume.

CBMR usually consists of two main steps: feature extraction and feature sequence match. Firstly, the audio segments are divided into overlapped frames, each generating a feature vector. Secondly, the feature sequence of the query is compared with those in the database. The computation cost for feature comparison increases as the number of music in the database gets large. To reduce the huge computation with almost no efficient indexing algorithms, many researchers have tried two ways: improving the CBMR by reducing the dimensionality of the features [7, 8, 9, 10] and optimizing the algorithms on matching two music segments sequences [1, 2, 3, 4, 5, 6, 12, 14].

In our algorithm, we calculate the cepstrum coefficients for each frame by short time spectrum analysis with regard to a particular music signal. We set a Spectral Correlation(SC) threshold to indicate the inter-frame spectral redundancy degree. If the SCs of multiple continuous frames are above the threshold, these frames are merged as a single frame, thus reducing the total number of frames. In this way, most of the time variation is removed, and a much simpler match method can be adopted for the remaining frames. We notice that the different order MFCC represents different spectral information and the low order cepstrum coefficients usually reflect the spectral structure. On these basis, we utilize the low order MFCC to filter all the music in the database; then among the survivors, we use all the cepstrum coefficients, both the low order and high order, to retrieve the best target. The pre-filtering method further improves the retrieval speed.

The rest of the paper is organized as follows. A review of related work is provided in Section 2. A fast and effective algorithm, SCBFM-TSR, is described in Section 3. The experiment results and evaluations are given in Section 4. Finally Section 5 concludes the paper.

2 RELATED WORK

As a general nonlinear alignment method, Dynamic Programming (DP) is exploited in the speech recognition to account for tempo variations in speech pronunciation [11, 13]. Many researchers [1, 2, 6, 12, 14] have studied DP and also applied DP or optimized DP in CBMR to match the query input against the reference melodies in the database. In these works [1, 5, 6, 12] usually query is determined from acoustic signals, and is matched against the database composed of pitch sequences. In other words, the sequence comparison relies on similarity measurement between acoustic input and symbolic database.

Among music retrieval research implemented on acoustic input and acoustic database, the ”energy profile” is adopted as the feature in [2], and the spectrum-based minimum-distance is used to improve the accuracy; both of the features sequence are compared by DP. Yang [14] also adopted the short time spectrum as the feature, except that only the signal of the local maximum is selected to calculate the feature. A variation of DP methods is used in feature comparison and the result is further refined with the linear filtering. More recently Haitsma, J., and T. Kalker [3] constructed a cryptography hash function to classify pre-defined fingerprints of acoustic data in a database. A two-stage search algorithm is built on only performing full fingerprint comparisons at candidate positions pre-selected by a sub-fingerprint search. H.Harb [4] reported a query by example music retrieval system based on the local (1s) and global (10-20s) acoustic similarities. The symmetric KL(KullBack Leibler) distance between the spectral features (MFCC) of two short audio slices is calculated as the local similarity.

Different from the existing methods, we remove the spectral redundancy by merging the adjacent similar frames. Based on the MFCC properties, we use the low order MFCC to pre-filter the reference melodies; this is to further improve the retrieval speed while maintaining the retrieval ratio.

3 SCBFM-TSR

In the following, we present a fast and efficient algorithm, SCBFM-TSR, which shows: (1) the redundant spectral information can be diminished, further, tempo variation can be tolerated. (2) keeping only the few significant spectral features can provide a concise description for a musical sound. (3) SCBFM-TSR is much faster than the straightforward application of DP between the two raw musical sounds while holding almost the same retrieval ratio.

3.1 Feature extraction

Mel-Frequency Cepstral Coefficient (MFCC) is adopted as the spectral feature. It is extracted from each frame, and is calculated from the short time spectrum. The bins of the magnitude 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. Both the query music and those in the database are in single-channel 16-bit wave format and re-sampled to the rate of 22.05kHz. The music is divided into overlapped frames. Each frame of music contains 1024 samples and the adjacent frames have 50% overlapping. Each frame is weighted by a hamming window, which is further appended with 1024 zeros to fit the length of FFT. Then the spectrum of each frame, $S_i = \{s_{i,1}, s_{i,2}, ..., s_{i,K}\}, i = 1, 2, ...I$, is calculated, where K stands for the maximum frequency and I is the number of frames of the music slice in question. The SC ρ is calculated for each S_i, and the L-order MFCC feature $M_i = \{m_{i,1}, m_{i,2}, ..., m_{i,L}\}, i = 1, 2, ...I$ is obtained.

3.2 Feature merge

The SC of each frame, S_i, is calculated, and the MFCC features, M_i, are merged as following:

(1) Remove the direct current components

$$s_{i,k} = s_{i,k} - \bar{s}_i,\ \bar{s}_i = \frac{1}{K}\sum_{k=1}^{K} s_{i,k} \qquad (1)$$

(2) Calculate the SC between S_i and S_j ($< \cdot, \cdot >$ stands for the inner product between two vectors.)

$$\rho_{i,j} = \frac{< S_i, S_j >}{\sqrt{< S_i, S_i >< S_j, S_j >}} \qquad (2)$$

```
Set = []
i=1;
while $i \le I$
  append $M_i$ to Set
  j=i+1;
  while $j \le I$  &&  $\rho_{i,j} > \rho_{th}$
    j++;
  end
  i=j;
end
```

Figure 1: Frame merge procedure

(3) Frame merge.

In Fig.1, out of the continuous frames with an SC bigger than a certain threshold, for example $\rho_{th} = 0.7$, only the first frame is kept. Fig.2 gives an example of frame merge. For the query music, frame 1 and 2 are spectral similar, so the two frames are merged to one frame s1. Frame 4 and 5 are spectral similar to frame 3, merged to s3. By merging the neighboring spectral similar frames, the storage and computation are decreased, and most of the time variation is removed.

3.3 Feature sequence match

The other advantage of frame merge is to mitigate the tempo variation problem. In Fig.2, the query music and

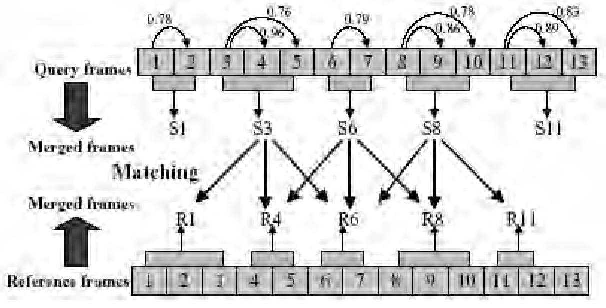

Figure 2: Feature merging and matching

reference music have different timing where the same notes (for example, 3rd, 4th and 5th frames in the query and 4th and 5th frames in the reference) have different length. However, by merging frames, the redundant information is removed. If the two sequences have the same score, it is reasonable that the merged sequences almost have the same timing, that is, the tempo feature of the merged query is nearly the same as that in the merged reference music. Therefore a much simpler match method can be adopted for the remaining frames: each frame is compared with multiple neighboring frames so as to consider the remaining time variation effect. The minimum of all the frames in the query music is added together to get the distance between the query music and the reference music.

```
query feature sequence: $M_{i_1}, M_{i_2}, ..., M_{i_Q}$
$r^{th}$ reference feature sequence: $M^r_{j_1}, M^r_{j_2}, ..., M^r_{j_R}$
for $m = j_1, j_2, ..., j_R$
  $D^r_m = 0$
  for $l = i_1, i_2, ..., i_Q$
    $d^r_l = \min_{n \in [-N,N]} |M_l - M^r_{m+l+n}|$;
    $D^r_m = D^r_m + d^r_l$
  end
end;
$D^r = \min_m D^r_m$
```

Figure 3: Sequences match procedure

Figure 3 gives the procedure of the feature sequence comparison between the query and one of the reference melodies. By merging correlated frames, the remaining query feature becomes $M_{i_1}, M_{i_2}, ..., M_{i_Q}$ while the r^{th} reference feature becomes $M^r_{j_1}, M^r_{j_2}, ..., M^r_{j_R}$. Then the query is matched against the r^{th} reference at different time shifts. For a specific time shift m, each frame in the query is compared with 2N+1 neighboring frames in the reference music so as to consider the remaining time variation effect. The minimums d^r_l of all the query frames are added together to get D^r_m, the distance between the query music and the reference music at time shift m. The distance between the query and the r^{th} reference, D^r, is the minimum among D^r_m with different m. Then for all the reference melodies, the following equation gives the desired music.

$$r = \arg\min_r D^r \tag{3}$$

In the real system, for a certain query, usually several retrieval results are given, ranked in the decreasing order of D^r, with the best matching reference melody at the top.

3.4 Acceleration of retrieval speed

For the purpose of obtaining a faster response as the number of music melodies increases, it is necessary to speed up the whole retrieval procedure.

MFCC is the DCT result of Mel-scaled spectrum. The low order MFCC is the low frequency components of the DCT, and reflects the basic spectral profile, which roughly stands for the music score; the high order MFCC is the high frequency component of the DCT, and reflects more details of the energy distribution in the different Mel bands. Based on the property of MFCC, the retrieval can be further accelerated with two-step retrieval by a pre-filtering method.

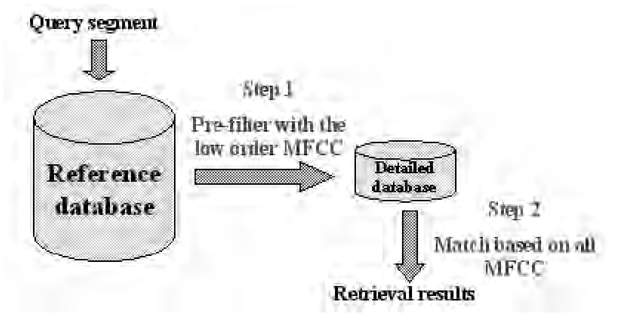

Figure 4: Two-step retrieval based on MFCC

Figure 4 explains the basic idea of the two-step retrieval. In the first step, only the low order MFCC of the query is used to roughly choose some candidates, removing most of the unlikely references in the music database. Only a small percentage of reference music survives as the candidates, and constructs a new database. In the second step, the desired music is searched with all MFCC coefficients in the small database. In consequence, the whole retrieval speed can be improved efficiently.

In our method, we used the following parameters:

ρ_{th}: SC threshold

L: MFCC order

L_1: number of low order MFCC used for pre-filtering in the first step in Fig.4

S: survive rate of the retrieval, the ratio between query output and all the references in the database

S_1: survive rate of the pre-filtering

S_2: survive rate of the second step retrieval

In the above, $S = S_1 \cdot S_2$, that is, the number of final retrieval results is the same for both the normal retrieval and the two-step retrieval with pre-filtering. For the former, the retrieval ratio is $R(\rho_{th}, L, S)$, a function of ρ_{th}, L and S; for the latter, the retrieval ratio is the product of the retrieval ratio in the pre-filtering stage and the retrieval ratio in the second stage: $R(\rho_{th}, L_1, S_1) \cdot R(\rho_{th}, L, S_2)$, which further depends on L_1 and S_1.

Though pre-filtering can improve the retrieval speed, it may lower the retrieval ratio, because the desired music may get lost in the pre-filtering stage. Thus it is required that $R(\rho_{th}, L_1, S_1)$ must be high enough so that the total retrieval ratio is almost not affected.

3.5 Computation analysis

Figure 5 shows the match procedure of DP (Fig.5(a)) and SCBFM-TSR (Fig.5(b)) respectively. When two sequences are matched, DP always acquires the optimal path, an alignment between query and reference found by recursively building a Dynamic Time Warping (DTW) table. However, it consumes enormous computation time. Its matching line is curved due to tempo variation. In SCBFM-TSR, both the query and references have a

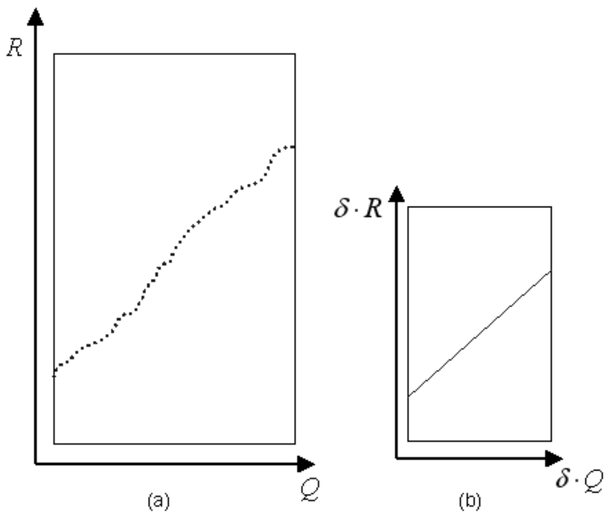

Figure 5: Match procedure

smaller length by merging similar frames, and the matching line tends to be straight because most of the time variation is removed. That is, the remaining query and reference melodies almost have the same tempo.

In order to analyze the computation, we take the following assumption: (1) The average number of frames of all the references is R, and that of query is Q. (2) In SCBFM-TSR, by merging similar adjacent frames, only few of the frames remain, which depends on the SC threshold ρ_{th}; the percentage of the remaining frames is $\delta(\rho_{th})$.

The order of the computation for DP, C_{DP}, is given in Eq.4. In SCBFM-TSR, the number of remaining frames for the references and query is $\delta(\rho_{th}) \cdot R$ and $\delta(\rho_{th}) \cdot Q$ respectively. The corresponding computation, C_{FM}, is given in Eq.5. With the two-step retrieval, in the pre-filtering stage, L_1 cepstrum coefficients are used, and S_1 percentage of references survive; in the second stage, the surviving references are searched with L cepstrum coefficients to get the best target. The total computation of the two stages, C_{PF}, is given in Eq.6.

$$C_{DP} = R \cdot Q \cdot L \tag{4}$$

$$C_{FM} = [\delta(\rho_{th}) \cdot R] \cdot [\delta(\rho_{th}) \cdot Q] \cdot L \tag{5}$$

$$\begin{aligned} C_{PF} &= [\delta(\rho_{th}) \cdot R] \cdot [\delta(\rho_{th}) \cdot Q] \cdot L_1 \\ &\quad + [\delta(\rho_{th}) \cdot R \cdot S_1] \cdot [\delta(\rho_{th}) \cdot Q] \cdot L \\ &= [\delta(\rho_{th}) \cdot R] \cdot [\delta(\rho_{th}) \cdot Q] \cdot [L_1/L + S_1] \end{aligned} \tag{6}$$

By pre-filtering, the total computation in Eq.6 is decreased by a factor $F_{PF} = L_1/L + S_1$ compared with Eq.5. Reducing either L_1 or S_1 can decease the computation. However, L_1 and S_1 have different effects on the retrieval ratio.

4 EXPERIMENTS AND RESULTS

Our music database consists of 166 melody pieces and is generated from Chinese folks (44 pieces from 12-Chinese-girl band) and western instruments sound (122 pieces performed by different instruments), including most of overlapped timbre, that is, all pieces consist of captivating polyphonic. These pieces are collected from CD-recordings and the Internet but the reference melodies in the database and music query samples are different versions so that they are played with different tempo. Each melody in the database is segmented into 60 seconds long melodic slip. Each query melodic slip contains part of the music and is about 8 seconds long. In the simulation, generally $L = 8$, $L_1 = 4$, $S_1 = 20\%$, and $\rho_{th} = 0.7$ except especially pointed out.

In the following, first the various SCs of different music segments and the corresponding frame merge are shown. Next the effect of ρ_{th}, L_1, and S_1 on storage, computation and retrieval ratio are discussed in order to demonstrate the feasibility of the SCBFM-TSR method and that the pre-filtering can speed up retrieval meanwhile the retrieval ratio is almost not affected.

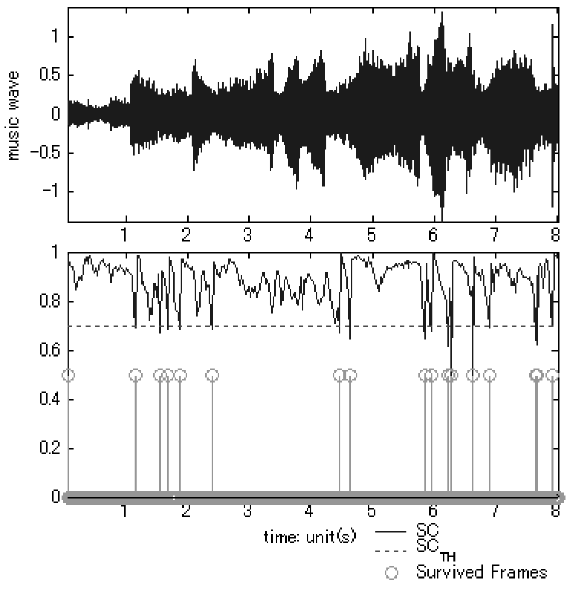

Figure 6: Spectral correlation and frame merge, $\rho_{th} = 0.7$ (No Word melody from 12-girl band)

4.1 Spectral correlation and frame merge

When the note duration is longer than the frame length, the adjacent frames may have very similar spectral structure, and are strongly correlated. Figure 6-7 show the SC (ρ) of the frames for two extreme cases. In Fig.6, the music is relatively simple, and most of the adjacent frames have a bigger SC than the SC threshold; thus they are merged. Out of the total 344 frames, only 4.9%, 17 frames are kept, which means that a single note contains about $344/17 = 20$ overlapped frames. In the experiment, the frame step length is 23.2ms, so the average note length is 460ms, corresponding to the slow rhythm. In Fig.7, the music spectral is much more complex, and the spectral features changes frequently. However, still more than half frames are merged and only 47.8% frames are kept. It is obvious that much of the spectral redundancy can be removed by setting up the SC threshold.

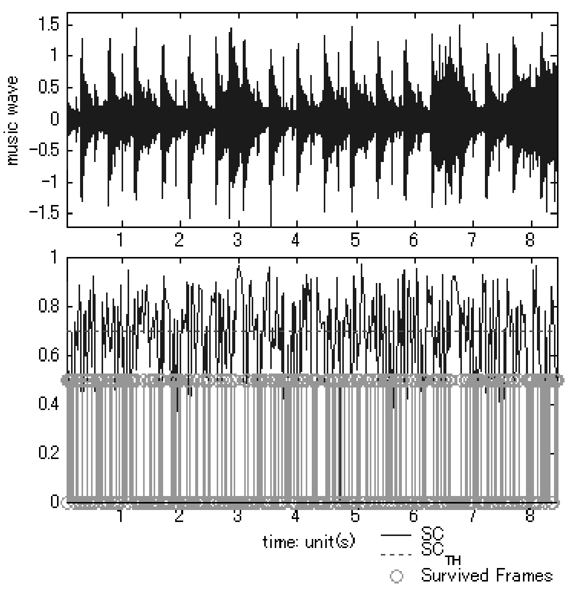

Figure 7: Spectral correlation and frame merge, $\rho_{th} = 0.7$ (Freedom melody from 12-girl band)

4.2 Storage

Figure 8 shows the feature storage with respect to SC threshold ρ_{th}. The storage is normalized with the conventional one for $\rho_{th} = 1.0$. The normalized storage actually equals $\delta(\rho_{th})$ defined in Section 3.5. When $\rho_{th} = 0.4$, $\delta(\rho_{th})$ is less than 10%. $\delta(\rho_{th})$ monotonically increases as ρ_{th} does. When ρ_{th} reaches 1.0, no frames are merged and $\delta(\rho_{th})$ equals 1. From the experiment, we approximate the relation between the storage and the SC threshold ρ_{th} as below:

$$\delta(\rho_{th}) = 0.0001e^{9\rho_{th}} + 0.1085 \quad (0 < \rho_{th} < 1) \quad (7)$$

Though the adjacent frames may be highly correlated, the music is composed of different notes, which has little correlation. So as ρ_{th} becomes smaller, $\delta(\rho_{th})$ can not reach 0. This is reflected in Fig.8 that as ρ_{th} is below 0.8, the slope of the storage decreases quickly; it is also reflected in Eq.7 by the constant item.

The pre-filtering method improves the retrieval speed and affects the retrieval ratio, however, it does not change the storage. So for both the retrieval with and without pre-filtering, the storage is the same.

4.3 Computation

Figure 9 shows the computation with respect to ρ_{th}. The curves corresponding to $\rho_{th} = [0.4, 0.8]$ are amplified in the middle to display the result more clearly. The upper curve stands for the computation with only frame merge, stated in Eq.5; and the bottom curve is the computation with both frame merge and pre-filtering, stated in Eq.6. The computations are normalized by the one given in Eq.4. When calculating Eq.5 and Eq.6, $\delta(\rho_{th})$ is the experiment result described in Fig.8. Merging frames contribute to the storage reduction of the references to the percentage $\delta(\rho_{th})$. With the same operation on the query

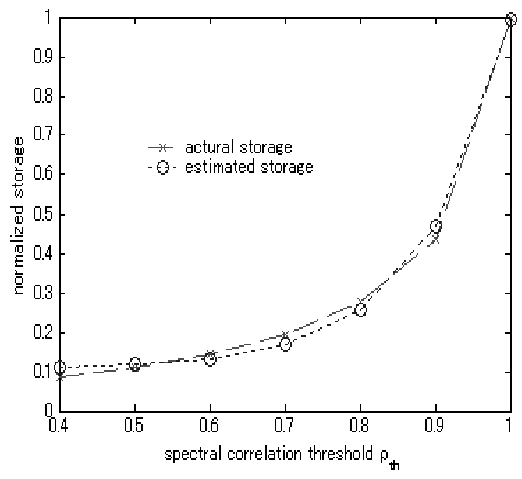

Figure 8: Feature storage (Normalized by the storage for $\rho_{th} = 1.0$)

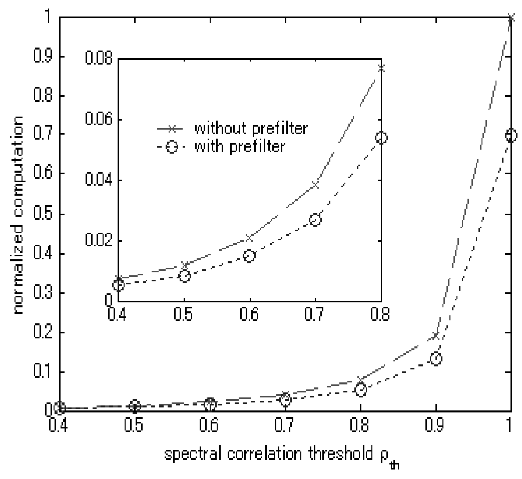

Figure 9: Average computation (Normalized by the computation without pre-filtering for $\rho_{th} = 1.0$)

segment, the computation in both Eq.5 and Eq.6 is proportional to $\delta(\rho_{th})^2$, so the computation reduction is very efficient. At $\rho_{th} = 0.7$, $\delta(\rho_{th}) = 0.195$, and the computation is reduced to 0.038, nearly 1/25 of the original computation.

The computations are the monotonically increasing function of ρ_{th}, and the one with pre-filtering is smaller for all ρ_{th}. For the pre-filtering case, $L_1 = 4$, $L = 8$, $S_1 = 20\%$, the corresponding factor $F_{PF} = 0.7$. Other combinations of L_1, L and S_1 with the same F_{PF} have the identical computation.

4.4 Retrieval ratio

Figure 10 gives the retrieval ratio for both top-4 retrieval and top-1 retrieval. For most of the cases, the retrieval ratio with pre-filtering is almost the same as that without pre-filtering for both top-4 and top-1 retrieval. When ρ_{th} is smaller than 0.7, the retrieval ratio increases as ρ_{th} does. The retrieval ratio is almost unchanged when ρ_{th} is within

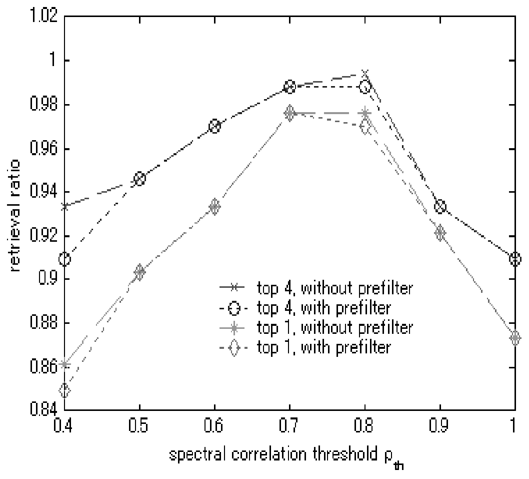

Figure 10: Retrieval ratio of top-4 retrieval and top-1 retrieval with respect to different SC threshold ρ_{th}, $S_1 = 0.2, L_1 = 4$)

$[0.7, 0.8]$. As ρ_{th} increases further, the retrieval ratio for both top-1 and top-4 retrieval decreases. This is due to the fact that as ρ_{th} gets very big, most of the feature frames are kept, and the simple feature match method adopted in our proposal can not deal well with the remaining time variation. Under all cases, top-1 retrieval result is worse than top-4 result, however, it still gives good result when ρ_{th} is within [0.7, 0.8].

From Fig.8-10, both the storage and the computation increase as ρ_{th} does. The retrieval ratio reaches its maximum when ρ_{th} lies within [0.7, 0.8]. It is obvious that there is a tradeoff between the storage, computation, and retrieval ratio. When the SC threshold ρ_{th} is set to 0.7, we can achieve almost the highest retrieval ratio with little storage and computation.

5 CONCLUSION

We have proposed a fast and efficient algorithm, SCBFM-TSR, to accelerate the content-based music retrieval. The SCBFM method has shown three main traits: (1) it removes the spectral redundancy and in turn reduces the feature storage of the reference music in the database; (2) both the query and the reference melodies have a short feature sequence, which improves the retrieval speed; (3) most of the tempo variation is removed, thus a simple feature sequence match method can be used. In addition, relying on the characteristic of MFCC, the TSR method further speeds up the whole retrieval.

Experimental results have shown that SCBFM-TSR approach can detect the desired melodies while tolerating tempo variations. Also, with a mathematical methodology, the relation between storage, computation and SC threshold is derived.

Future work may include the study of the higher-level content-based music information retrieval such as acoustic instrument timbre similarity, musical genre classification, and singer identification and so on. We are currently collecting more music melodies and building the acoustic-based distributed retrieval system.

References

[1] R. B. Dannenberg and N. Hu. Understanding search performance in query-by-humming systems. In *Proc. ISMIR 2004*, pages 236–241, 2004.

[2] J. Foote. Arthur: Retrieving orchestral music by long-term structure. In *Proc. ISMIR 2000*, 2000.

[3] J. Haitsma and T. Kalker. A highly robost audio fingerprinting system. In *Proc. 3rd International Conference on Music Information Retrieval*, pages 107–115, 2002.

[4] H. Harb and L. Chen. A query by example music retrieval algorithm. In *Proc. 4th European Worshop on Image Analysis for Multimedia interactive Services*, pages 122–128, 2003.

[5] H. Hashiguchi, T. Nishimura, J. Takita, J. Xin Zhang, and R. Oka. Music signal spotting retrieval by a humming query. In *Proc. 5th World Multi-Conference on Systemics, Cybernetics and Informatics*, volume 7, pages 280–284, 2001.

[6] J. S. R. Jang, H. R. Lee, J. C. Chen, and C. Y. Lin. Research and development of an mir engine with multi-modal interface for real-world applications in east asia. *Journal of the American Society for Information Science and Technology*, 55(12):1067–1076, 2004.

[7] K. V. R. Kanth, D. Agrawal, and A. Singh. Dimensionality reduction for similarity searching in dynamic databases. In *Proc. ACM SIGMOD international conference on Management of data*, pages 166–176, 1998.

[8] K. Kashino, G. Smith, and H. Murase. Time-series active search for quick retrieval of audio andvideo. In *Proc. ICASSP 1999*, volume 6, pages 2993–2996, 1999.

[9] A. Kimura, K. Kashino, T. Kurozumi, and H. Murase. A quick search method for multimedia signal using feature compression based on piecewise linear maps. In *Proc. ICASSP 2002*, volume 4, pages 3656–3659, 2002.

[10] A. Kimura, K. Kashino, T. Kurozumi, and H. Murase. Dynamic segmentation based feature dimension reduction for quick audio/video searching. In *Proc. IEEE International Conference on Multimedia and Expo*, volume 2, pages 389–392, 2003.

[11] J. R J. G. Proakis and J. H. L. Hansen. *Discrete-time processing of speech signals*. Macmillan Pub. Co., 1993.

[12] J. M. Song, S. Y. Bae, and K. Yoon. Mid-level music melody representation of polyphonic audio for query-by-humming system. In *Proc. ISMIR 2002*, pages 133–139, 2002.

[13] T. K. Vintsyuk. Speech discrimination by dynamic programming. *Kibernetika*, 4(2):81–88, 1968.

[14] C. Yang. Music database retrieval based on spectral similarity. In *Proc. ISMIR 2001*, 2001.