Score Following and Retrieval Based on Chroma and Octave Representation

Wei-Ta Chu and Meng-Luen Li,

Department of Computer Science and Information Engineering,
National Chung Cheng University,
Chiayi, Taiwan 621
wtchu@cs.ccu.edu.tw, badbadyuniko@hotmail.com

Abstract. With the studies of effective representation of music signals and music scores, i.e. chroma and octave features, this work conducts score following and score retrieval. To complement the shortage of chromagram representation, energy distributions in different octaves are used to describe tone height information. By transforming music signals and scores into sequences of feature vectors, score following is transformed as a sequence matching problem, and is solved by the dynamic time warping (DTW) algorithm. To conduct score retrieval, we modify the backtracking step of DTW to determine multiple partial matchings between the query and a score. Experimental results show the effectiveness of the proposed features and the feasibility of the modified DTW algorithm in score retrieval.

Keywords: Score following, score retrieval, chroma, octave, sequence matching.

1 Introduction

With the rapid popularity of digital music, people usually have huge music collections. People who love music usually tend to follow the corresponding music score when a music piece is played. From another viewpoint, musicians in concerts usually have to manually turn pages of sheet music. For a professional performer, it is a routine for page turning during live performance. However, tuning pages is uncomfortable for a beginner and may distract performance. To address these issues, finding temporal matching between a music signal and the corresponding score is necessary.

We develop a framework as shown in Figure 1 to conduct music-score matching so as to achieve score following. Chroma features and octave features are extracted from both music signals and music scores. Based on the same representation, an approximate sequence matching algorithm is used to match two sequences, and the determined matching represents temporal correspondence between two media. We therefore achieve score following, in which a score bar can be highlighted when its corresponding music segment is being played. With the module that evaluates similarity between a music signal and a music score, we extend this framework to conduct score retrieval, which is similar to query by humming.

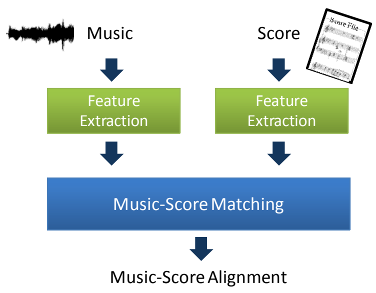


Figure 1. The framework of music-score alignment.

Contributions of this work mainly lie on the usage of novel features. We elaborately explore characteristics of music signals, and thus extract chroma and octave information. In experiments, we verify that the proposed features provide promising performance, as comparing with conventional energy-based and pitch-based features.

The rest of this paper is organized as follows. Section 2 gives brief literature survey. Section 3 describes the features used to describe music signals and scores, in which octave features are newly proposed. Details of score following and score retrieval are described in Section 4. Section 5 provides extensive evaluation results, following by conclusion of this work in Section 6.

2 Related Work

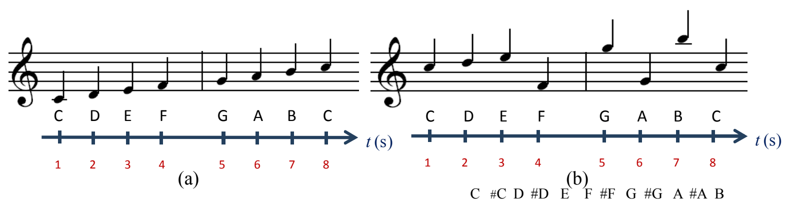
By definition, score following is to align part of a music score to the corresponding portion of a music signal. The approach in [1] uses a two-level hidden Markov model (HMM) to handle score following for polyphonic instruments, such as the piano and guitar. A real-time decoding scheme was designed to return the HMM state sequence corresponding to the alignment between music and score. A Viennese company Qidenus [2] develops a page-turning device, which can be controlled by foots and supports musicians to avoid inconvenient manual page-turning. In [3], this page-turner device is utilized to automate page-turning of sheet music. They first convert sheet music into MIDI files, and then extract audio features from them. The positions where the pages should be turned are labeled manually. From the lively performed music signal, audio features are also extracted so that feature sequences from music and score are aligned by the dynamic time warping algorithm. Recently, the importance of score following has been widely recognized, and more and more researchers are involved. Therefore, the annual evaluation campaign "Music Information Retrieval Exchange" [4] starts the task of score following from year 2006.

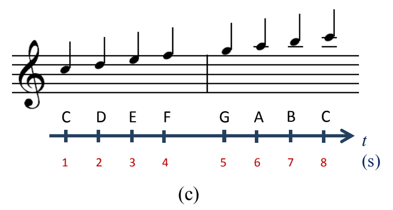
3 Feature Extraction

3.1 Chroma and Octave Features

Shepard advocated that human's pitch perception is like a helix, in which the vertical and angular dimensions represent *tone height* and *chroma*, respectively [5]. Music notes in an octave can be classified into twelve pitch families, i.e. chroma. In this helix, as the pitch increases (e.g. from C0 to C1), it looks like moving along the helix, passing through other pitch families chromatically, and finally going back to the same family (C) that is one octave higher than the initial point. It means that two different music notes could be grouped into the same pitch family if their corresponding frequencies have some relationship. For example, in Figure 2(a), the first and the eighth music notes are both mapped to the chroma C. With this idea, every audio frame is transformed into a 12-dimensional chromagram, in which each dimension represents the accumulated energy of a chroma, respectively.

Chromagram only conveys information of pitch families. Therefore, music sequences with the same evolution of chroma changes may have the same representation. In Figure 2, although music notes of these three segments are different, their chromagrams are the same. In this work, we propose an 8-dimensional octavegram to address this issue. An octave is composed of 12 semi-tones (i.e. C, C#, D, #D, etc.), and the frequency domain can be divided into eight octaves. Every audio frame can be transformed into an 8-dimensional octavegram, in which each dimension represents energy corresponding to one octave, respectively. With this representation, three segments in Figures 2(a), (b), and (c) can be discriminated.





	C	#C	D	#D	E	F	#F	G	#G	A	#A	B
frame 1	(**1**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0)
frame 2	(0,	0,	**1**,	0,	0,	0,	0,	0,	0,	0,	0,	0)
frame 3	(0,	0,	0,	0,	**1**,	0,	0,	0,	0,	0,	0,	0)
frame 4	(0,	0,	0,	0,	0,	**1**,	0,	0,	0,	0,	0,	0)
frame 5	(0,	0,	0,	0,	0,	0,	0,	**1**,	0,	0,	0,	0)
frame 6	(0,	0,	0,	0,	0,	0,	0,	0,	0,	**1**,	0,	0)
frame 7	(0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**1**)
frame 8	(**1**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0)

(d)

Figure 2. (a)(b)(c) three different music segments; (d) the chromagram corresponding to all (a), (b), and (c).

3.2 Feature Extraction from MIDI

To efficiently represent music scores, which are represented by MIDI format in this work, we parse bit streams of MIDI files to extract music scores, and then compute corresponding chroma features and octave features. A tuple consisting of two components is used to describe the information of music notes:

$$(N, duration), \tag{1}$$

where $N = \{n_1, n_2, ..., n_K\}$ is a set of music notes that strike simultaneously, and n_i denotes the MIDI node id.

Assume that a MIDI file has M tracks, and the ith track has j_i tuples. This MIDI file is then represented as M sequences $(\boldsymbol{S}_1, \boldsymbol{S}_2, ..., \boldsymbol{S}_M)$, in which $\boldsymbol{S}_i = \{(N_{i1}, t_{i1}), ..., (N_{ij_i}, t_{ij_i})\}$. Figure 3 shows a MIDI track consisting of six tuples. The first tuple, for example, consists of four music notes (48, 52, 55, 60) with duration 0.4 seconds.

To calculate chroma value of a music note, an MIDI node is converted into a chroma by

$$c = \begin{cases} (n_k \% 12) + 1, & \text{for } n_k \geq 0, \\ -1, & \text{for } n_k < 0. \end{cases} \tag{2}$$

We divide the MIDI-based score into non-overlapping 0.2-second segments, called score frames. For each score frame, we count the number of chroma occurring in this segment, and describe this segment as a 12-dim chroma vector. A music score is therefore transformed into a sequence of chroma vectors.

Figure 4 illustrates the process for converting Figure 3 into chroma vectors. The first tuple $(\{48, 52, 55, 60\}, 0.4)$ is first converted into chroma representation $(\{\mathrm{C}, \mathrm{E}, \mathrm{G}, \mathrm{C}\}, 0.4)$. At the first score frame (0~0.2 second), the chroma C occurs twice since both notes 48 and 60 map to C. The normalized value of the kth chroma bin of the chroma vector $\tilde{\boldsymbol{p}}_f^c$ at the score frame f can then be calculated as

$$\tilde{\boldsymbol{p}}_{f,k}^c = \frac{\boldsymbol{p}_{f,k}^c}{\sum_{j=1}^{12} \boldsymbol{p}_{f,j}^c}. \tag{3}$$

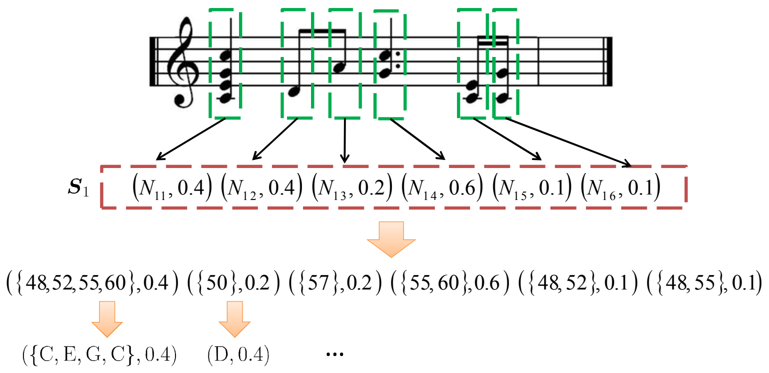
To calculate octave features from MIDI, an MIDI node is converted into an octave number by

$$o = \begin{cases} \lfloor n_k / 12 \rfloor + 1, & \text{for } n_k \geq 0, \\ -1, & \text{for } n_k = -1. \end{cases} \tag{4}$$

With the same idea of calculating chroma features, we divide the MIDI-based score into non-overlapping 0.2-second score frame. For each score frame, we count the number of octaves in which some music notes strike, and describe this segment as an 8-dim octave vector, since there are totally eight octaves in the frequency domain. A music score is therefore transformed into a sequence of octave vectors.

The normalized value of the kth octave bin of the octave vector $\tilde{\boldsymbol{p}}_f^o$ at the score frame f can then be calculated as

$$\tilde{\boldsymbol{p}}_{f,k}^o = \frac{\boldsymbol{p}_{f,k}^o}{\sum_{j=1}^{8} \boldsymbol{p}_{f,j}^o}. \tag{5}$$



S_1
$(N_{11}, 0.4)$ $(N_{12}, 0.4)$ $(N_{13}, 0.2)$ $(N_{14}, 0.6)$ $(N_{15}, 0.1)$ $(N_{16}, 0.1)$
$(\{48,52,55,60\},0.4)$ $(\{50\},0.2)$ $(\{57\},0.2)$ $(\{55,60\},0.6)$ $(\{48,52\},0.1)$ $(\{48,55\},0.1)$
$(\{C,E,G,C\},0.4)$ $(D,0.4)$ ...

Figure 3. An example of track sequence.

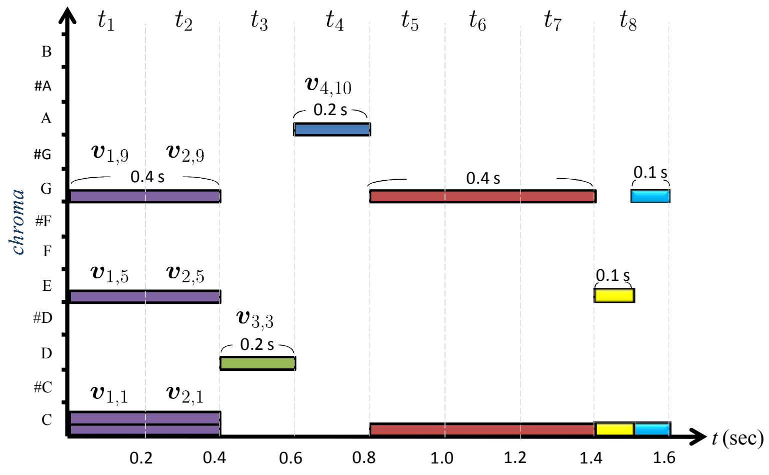


Figure 4. An example for calculating energies in different bins.

3.3 Feature Extraction from Audio

To efficiently describe audio signals, audio data are first transformed into the frequency domain, and then energy of each frequency bin is calculated. In experiments of this work, we collect music pieces from [6] because this website contains music performed by different instruments and the corresponding music scores stored in MIDI format. Each music piece is interpreted by a player rather than computer-synthesized, and there may be noise in music. Therefore, we propose a pre-emphasis method to strengthen energy distribution in each frame.

First, the frequency domain is divided into eight regions, and each region contains twelve frequency bins. Let $E_{f,k}$ be the energy of the kth frequency bin at frame f, and $E_{f,r}$ be the accumulated energy of the rth region, where $r \in [1,8]$. The average energy of the rth region at frame f is calculated as $AvgE_{f,r} = E_{f,r}/12$. With the previous definitions, the pre-emphasis process is formulated as

$$\hat{E}_{f,k} = \begin{cases} w_e \times E_{f,k}, & \text{for } E_{f,k\in S_r} \geq AvgE_{f,r}, \\ E_{f,k}, & \text{otherwise,} \end{cases} \tag{6}$$

where $w_e \geq 1$ indicates the *energy weight*, and S_r is a set of frequency bins that corresponds to the rth region.

After pre-emphasis, we calculate the corresponding chromagram. The accumulated energy $\tilde{E}_{f,h}$ of each pitch family h is calculated as

$$\tilde{E}_{f,h} = \textstyle\sum_{k\in S_h} \hat{E}_{f,k}, \tag{7}$$

where S_h is a set of frequency bins that corresponds to the chroma family h. The normalized value of the kth chroma bin of the chroma vector $\tilde{\boldsymbol{q}}_f^c$ at the music frame f can then be calculated as

$$\tilde{\boldsymbol{q}}_{f,k}^c = \frac{\tilde{E}_{f,k}}{\sum_{j=1}^{12} \tilde{E}_{f,j}}. \tag{8}$$

Examples of chromagram can be seen in Figure 2(d).

To generate an 8-dim octave vector, we calculate energy of each frequency bin, and the accumulated energy $\tilde{E}_{f,g}$ of each element of an octave vector could be calculated as

$$\tilde{E}_{f,g} = \textstyle\sum_{k\in S_g} \hat{E}_{f,k}, \tag{9}$$

where S_g is a set of frequency bins that corresponds to the octave family g. The normalized value of the kth octave bin of the octave vector $\tilde{\boldsymbol{q}}_f^o$ at the music frame f can then be calculated as

$$\tilde{\boldsymbol{q}}_{f,k}^o = \frac{\tilde{E}_{f,k}}{\sum_{j=1}^{8} \tilde{E}_{f,j}}. \tag{10}$$

4 Score Following and Score Retrieval

4.1 Music-Score Matching

We have transformed both music signals and music scores into sequences of feature vectors, in the representation of chroma and octave. The problem of music-score matching is thus transformed into a sequence matching problem. Assume that there are m music frames and n score frames in these two sequences. Let $\boldsymbol{Q}^c = (\boldsymbol{q}_1^c, \boldsymbol{q}_2^c, ..., \boldsymbol{q}_m^c)$ and $\boldsymbol{Q}^o = (\boldsymbol{q}_1^o, \boldsymbol{q}_2^o, ..., \boldsymbol{q}_m^o)$ respectively denote the sequences of chroma vectors and octave vectors of a music signal Q. Let $\boldsymbol{P}^c = (\boldsymbol{p}_1^c, \boldsymbol{p}_2^c, ..., \boldsymbol{p}_n^c)$ and $\boldsymbol{P}^o = (\boldsymbol{p}_1^o, \boldsymbol{p}_2^o, ..., \boldsymbol{p}_n^o)$ respectively denote the sequences of chroma vectors and octave vectors of the score corresponding to the music signal Q. Note that we simplify the notations $\tilde{\boldsymbol{q}}_i^c$, $\tilde{\boldsymbol{q}}_i^o$, $\tilde{\boldsymbol{p}}_i^c$, $\tilde{\boldsymbol{p}}_i^o$ as $\boldsymbol{q}_i^c$, $\boldsymbol{q}_i^o$, $\boldsymbol{p}_i^c$, $\boldsymbol{p}_i^o$ to denote normalized vectors. We then respectively construct a cost matrix based on chroma features and octave features. The (i,j)-th entry of the chroma-based cost matrix $M_{i,j}^c$ is calculated as

$$M_{i,j}^c = d(\boldsymbol{q}_i^c, \boldsymbol{p}_j^c) = \sqrt{\textstyle\sum_{\ell=1}^{12} (\boldsymbol{q}_{i,\ell}^c - \boldsymbol{p}_{j,\ell}^c)^2}. \tag{11}$$

Similarly, the (i, j)-th entry of the octave-based cost matrix $M^o_{i,j}$ is calculated as

$$M^o_{i,j} = d(\boldsymbol{q}^o_i, \boldsymbol{p}^o_j) = \sqrt{\sum_{\ell=1}^{8} (\boldsymbol{q}^o_{i,\ell} - \boldsymbol{p}^o_{j,\ell})^2}. \tag{12}$$

We combine two cost matrices by

$$M^b_{i,j} = w_c M^c_{i,j} + (1 - w_c) M^o_{i,j}, \tag{13}$$

where w_c indicates the weight controlling the relative impact of chroma and octave information.

Based on the combined cost matrix, finding the optimal matching between a music signal and its corresponding music score can be solved by finding the alignment with the lowest cost between two feature sequences. This problem is well known to be solved by a dynamic time warping algorithm. Figure 5 shows two examples of alignment results, in which brighter pixels mean smaller costs. The left side shows that the music signal is played strictly according to the score, and the optimal alignment lies on the main diagonal of this cost matrix. The right side shows a matching with slight variations to the main diagonal, because the music signal is played in varied speeds and may be disturbed by noises.

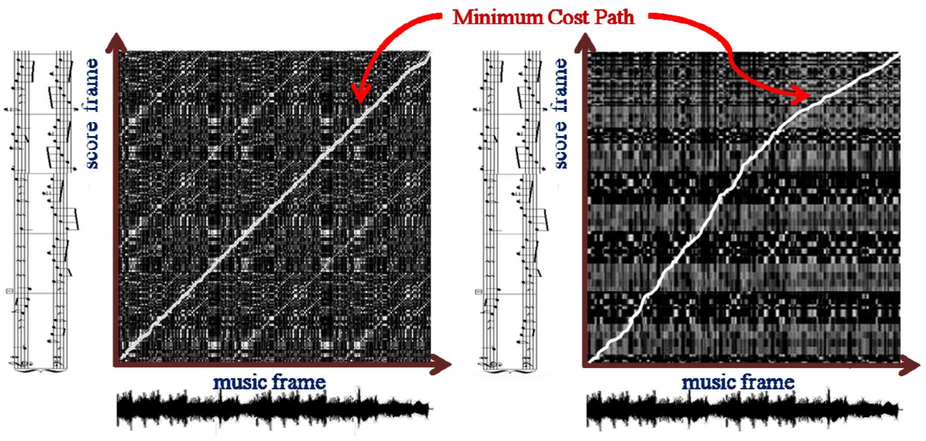


Figure 5. Two alignment results obtained by the dynamic time warping algorithm.

4.2 Score Retrieval

We conduct score retrieval by the same way as music-score matching. Four steps are included in our score retrieval system.

- Feature extraction and cost matrix calculation

The query music piece and score files are converted into sequences of feature vectors. We then calculate a cost matrix between the query music piece and every score in the database, respectively.

- Finding optimal alignment

The dynamic time warping algorithm is used to find an alignment with the minimal cost, while a different implementation is used in the backtracking step. In the classical DTW algorithm, the start point of backtracking is limited at (m, n), where m and n denote the lengths of two sequences, respectively. This setting is not suitable for score retrieval since the query music piece often simply corresponds to part of a score. The globally optimal alignment between a music score and a short query piece may not give the best solution. Moreover, there may be several similar portions in a score, i.e. the query piece may correspond to multiple segments of a score. Therefore, we do not limit the number of the best alignment to be found.

Instead of always backtracking from the (m,n)-th entry of the cost matrix, we sequentially conduct backtracking from (m,n), $(m,n-1)$, ..., $(m,m+1)$, assuming that the query piece is always shorter than a music signal in the database $(m < n)$. An optimal alignment (a path) can be determined from each backtracking case, and we can calculate the average cost of each path based on the entries where the path goes through. Figure 6 shows examples of alignment results, in which P_k, $k = m+1, m+2, ..., n$, denotes a path corresponding to an alignment result of a backtracking case. To eliminate matching noises, a path is filtered out when its average cost is larger than a predefined cost threshold ϵ. The path P_{174} in Figure 6 is discarded, for example.

- Path clustering and filtering

After filtering, several neighboring paths may perceptually correspond to the same alignment, i.e. they nearly indicate the same correspondence between the score and the query. To avoid redundancy, we cluster paths that are obtained by backtracking from neighboring start points. For example, in Figure 6, the remaining paths (say $P_{100}, P_{101}, P_{212}, P_{213}, P_{214}$, and P_{215}) are grouped into two clusters, i.e. $\{P_{100}, P_{101}\}$ and $\{P_{212}, P_{213}, P_{214}, P_{215}\}$. From each cluster, we only choose the path of the smallest average cost to be the representative of a cluster. Recall that each representative path indicates a correspondence between the query and a score. Given a query piece, alignments (representative paths) between it and each score are first found. All paths are then ranked by their average costs, and are returned as the ranked retrieval results.

5 Experiments

Each piece of music used in this work is polyphonic. The standard MIDI file (.mid) is used to store music scores. Three datasets are used in this work. Dataset1 consists of 67 music files and 67 corresponding scores (MIDI files), which are collected from [6]. Each music piece is recorded from a real performance, which proceeds according to a corresponding score. Music pieces have durations ranging from 27.1 to 191.5 seconds. The sampling rate is 44.1kHz and each sample is represented by 16 bits. The amounts of score bars of scores range from 9 to 92. To evaluate performance of music-score matching, we listen to all music files and manually identify how a segment of a music signal corresponds to a score bar in the corresponding music score. There are also 67 music files and 67 corresponding score files (MIDI files) in Dataset2. The score files are same as that in Dataset1, while the music files in Dataset2 are converted from MIDI files by a MIDI synthesizer. Dataset3 is generated for evaluating the score retrieval system. It consists of 1491 music queries, 67 corresponding score files

(MIDI files), and 133 irrelevant score files. Each query piece is generated by randomly selecting a segment from a music signal in Dataset1.

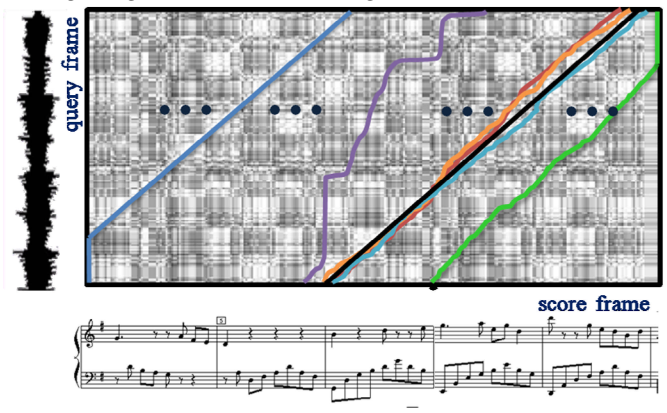


Figure 6. Various alignment results between the query piece and a music score.

5.1 Performance of Music-Score Matching

For two sequences Q and P, the point (i,j) in the alignment path denotes that the music frame $\boldsymbol{q}_i$ is similar to the score frame $\boldsymbol{p}_j$. In the backtracking path, a point is claimed as a correct matching if it is located in the right score bar. This evaluation scheme tolerates slight skews of alignment results, which are hardly perceived by humans. Accuracy of music-score matching is therefore defined as:

$$\text{accuracy} = \frac{\text{number of correct points}}{\text{number of points in the path}} \times 100\%. \tag{14}$$

Based on Dataset1 and Dataset2, we conduct music-score matching with different cost weights w_c, which controls the importance of chroma vectors, and with different emphasis weights w_e, which controls the degree of pre-emphasis. The value $w_c = 1$ means only the chroma representations is used. The value $w_e = 1$ means pre-emphasis is not applied.

Figure 7 shows accuracy values for Dataset1 and Dataset2, under different weighting settings. We find that an appropriate combination of w_c and w_e provides better music-score matching, while the best settings for two datasets are different. However, adding the pre-emphasis process is not suitable for all the cases. Comparing the results of $w_e = 1$ with that of $w_e = 10$, we observe that the pre-emphasis process improves performance in Dataset1 but not in Dataset2. It means that the pre-emphasis process provides more impacts on music in real performance.

To further verify the superiority of our proposed features, we compare performance of music-score matching based on four features: 1) pitch histogram [7]; 2) energy increasing feature [8]; 3) chroma [5]; 4) chroma+octave (ours). Figure 8 shows that our approach presents the highest accuracy. Although satisfactory accuracy can be achieved (more than 85%) based on energy increasing features, the high-dimensional representation increases computation cost.

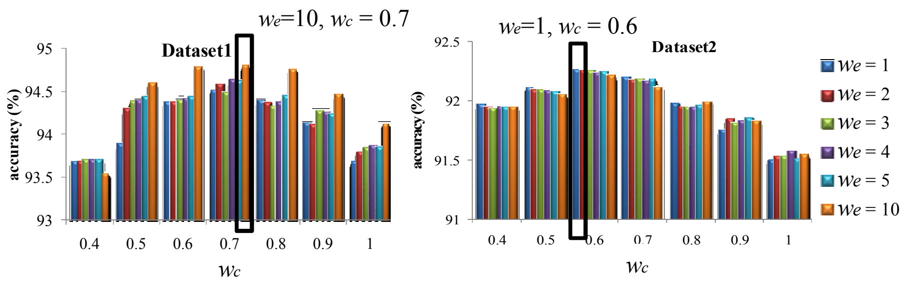


Figure 7. Accuracies of music-score matching for Dataset1 and Dataset2.

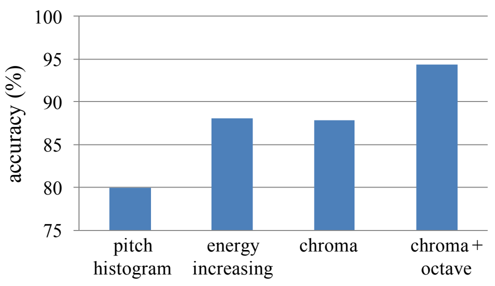
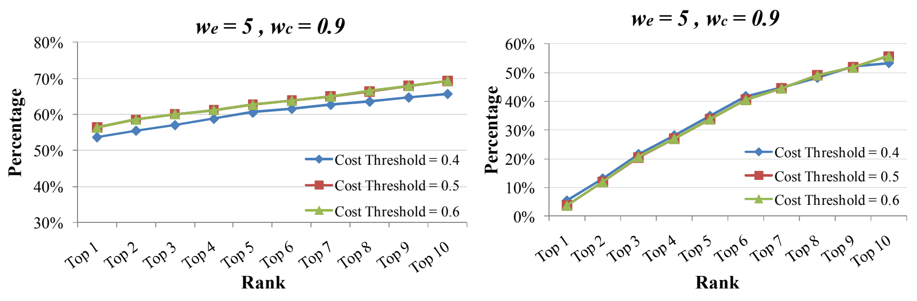


Figure 8. Performance of music-score matching based on different features.



(a) score-based retrieval (b) portion-based retrieval

Figure 9. Performance of score-based retrieval and portion-based retrieval.

5.2 Performance of Score Retrieval

Two kinds of retrieval are evaluated: 1) *Score-Based Retrieval* and 2) *Portion-Based Retrieval*. In score-based retrieval, since a score file may contain several similar portions, retrieving a score file that really contains a portion similar to the query is regarded as a correct result. In portion-based retrieval, not only the corresponding score should be retrieved, but also the corresponding segments in this score should be correctly indicated.

Dataset3 is adopted in this experiment. For two kinds of retrievals, the results are respectively presented as ranked lists. Figure 9 shows the percentage of queries that retrieve the correct results from the top k returned results, with the best parameter setting ($w_e = 5, w_c = 0.9$). From these results, we found that chroma features play a more important role ($w_c = 0.9$) in music-score matching, no matter in score-based retrieval or portion-based retrieval. Another observation is that the cost threshold ϵ used for filtering out noisy paths doesn't matter a lot, which shows the reliability of the modified DTW algorithm.

6 Conclusion

We have presented score following and score retrieval based on a newly proposed feature and a modified DTW algorithm. In addition to utilize chroma features that are more appropriate to describe music content, we further propose an octave-based feature with a pre-emphasis process to describe energy distributions in different octaves. Score following is transformed into the problem of finding optimal correspondence between two feature sequences, and thus a conventional DTW algorithm is adopted. We modify the backtracking steps of DTW, and extend the idea to conduct score retrieval. Experimental results show superiority of the proposed feature and feasibility of the proposed pre-emphasis method. In the future, more elaborate features would be investigated to improve matching performance.

Acknowledgement
The work was partially supported by the National Science Council of Taiwan, Republic of China under research contract NSC 99-2221-E-194-036 and NSC 98-2221-E-194-056.

References

1. Schwarz, D., Orio, N., and Schnell, N.: Robust Polyphonic MIDI Score Following with Hidden Markov Models. In: Proceedings of the International Computer Music Conference, pp. 129--132 (2004)
2. Qidenus Technologies: http://www.qidenus.com
3. Cont, A., Schwarz, D., Schnell, N., and Raphael, C.: Evaluation of Real-Time Audio-to-Score Alignment. In: Proceedings of the International Society for Music Information Retrieval, pp. 315--316 (2007)
4. The Music Information Retrieval Exchange (MIREX): http://www.music-ir.org/mirex/wiki/MIREX_HOME
5. Shepard, R.N.: Circularity in Judgements of Relative Pitch. Journal of the Acoustical Society of America, 36, pp. 2346--2353 (1964)
6. Free-Scores.com: http://www.free-scores.com/
7. Tzanetakis, G., Ermolinskyi, A., and Cook, P.: Pitch Histograms in Audio and Symbolic Music Information Retrieval. In: Proceedings of the International Conference of Music Information Retrieval, pp. 31--38 (2002)
8. Arzt, A.: Score Following with Dynamic Time Warping: An Automatic Page-Turner. Master's Thesis, University of Technology, Vienna, 2008.