A ROBUST MUSICAL AUDIO SEARCH METHOD BASED ON DIAGONAL DYNAMIC PROGRAMMING MATCHING OF SELF-SIMILARITY MATRICES

Tomonori Izumitani
NTT Communication Science Laboratories
izumi@eye.brl.ntt.co.jp

Kunio Kashino
NTT Communication Science Laboratories
kunio@eye.brl.ntt.co.jp

ABSTRACT

We propose a new musical audio search method based on audio signal matching that can cope with key and tempo variations. The method employs the self-similarity matrix of an audio signal to represent a key-invariant structure of musical audio. And, we use dynamic programming (DP) matching of self-similarity matrices to deal with time variations. However, conventional DP-based sequence matching methods cannot be directly applied for self-similarity matrices because they cannot treat gaps independently of other time frames. We resolve this problem by introducing "matched element indices," which reflect the history of matching, to a DP-based sequence matching method. We performed experiments using musical audio signals. The results indicate that the proposed method improves the detection accuracy in comparison to that that obtained by two conventional methods, namely, DP matching with chroma-based vector rotations and a simple matching of self-similarity feature vectors.

1 INTRODUCTION

Musical audio search, that is, identifying the title and corresponding locations in an audio database from a short musical audio excerpt of a query, has become important with the rapid expansion of digital content utilization.

A technology known as audio fingerprinting has been widely used to search for identical audio signals to queries. Most audio fingerprinting systems can be applied to audio signals with noises or other signal distortions. However, it is also demanded to cope with other kinds of variations caused by different players, keys, tempi, arrangements, and so on.

We focus on a musical audio search task where audio signals include key and tempo variations. For this task, many approaches have been reported. An early work using the chroma-based features was reported by Nagano et al., who proposed a feature representation named the polyphonic binary feature vector (PBFV) [1]. Their method deals with key variations by rotating query vector elements. They used beat detection and dynamic programming for tempo variations. Müller et al. also used chroma-based features for audio matching with different timbres played by various players [2].

For similar purposes, cover song detection methods have been proposed. Typically, they detect the same title as the query in the database by comparing music tracks. Ellis et al. proposed a method based on chroma-based features and a beat tracking technique [3]. Tsai et al. adopted melody extraction and a dynamic programming (DP) based matching method, known as dynamic time warping (DTW) [4]. For the audio cover song identification task of MIREX[1] , several methods have been proposed and evaluated using the same test data.

In the above mentioned methods, chroma-based features with element rotation and key adjustment are often used to detect transposed musical pieces. The former needs twelve matching processes and the latter need schemes for accurate key adjustment, for example, when key changes are included in a piece. For tempo variations, DP-based matching or time adjustment by beat tracking is commonly used.

In contrast, here we deal with key variation in terms of features invariant to it. We use a self-similarity matrix that has elements defined by spectral (or other) similarities between two different time points in an audio signal.

The self-similarity matrix has been used to represents the global structure of a musical piece. Foote used it for visualization [5] and audio segmentation [6]. Marolt used it to detect repeated parts within a musical piece for a musical audio retrieval system [7].

A self-similarity matrix also represents local structures, such as frame-wise relationships. And, it is insensitive to key variations because the relationship between two time points within a musical audio signal tends to be kept even when the music is played in a different key. In this work, we constructed a musical audio search system using this property.

In our earlier work, self-similarity was applied for musical audio search with key variations using simple vector matching [8]. However, the detection accuracy degrades when tempo variations are significant.

As a solution to this problem, we utilize a DP-based matching scheme. However, conventional DP methods for sequential data are not applicable because columns and rows of self-similarity matrices, which include gaps, have to be matched simultaneously in each step of the DP calculation.

[1] http://www.music-ir.org/mirex/2007/index.php/Main_Page

We propose a method for resolving this problem. The key idea is the introduction of "matched element indices," which reflect the history of the matching process. This reduces the problem to one that conventional DP-based sequence matching schemes can handle.

2 METHODS

2.1 Audio search framework based on self-similarity matrix

The proposed method searches the database for the same parts as a query given as a short piece of musical audio signal based on self-similarity matrices.

Self-similarity matrix $\mathbf{A} = (a_{ij})$ represents audio similarity of two time points within an audio signal. Element a_{ij} is a similarity value between i-th and j-th frame. In this study, we define a_{ij} using spectral power of i-th and j-th frame and the radial basis function (RBF) as follows:

$$a_{ij} = \exp\left(-C|\mathbf{p}_i - \mathbf{p}_j|^2\right), \quad (1)$$

where C is a constant. Vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ indicate spectral powers of i-th and j-th frame, respectively.

RBF is related to the Euclidean distance between two vectors and its value ranges from 0 to 1. There are some other similarity measures, such as a cosine measure, for the self-similarity calculation. The reason we adopt the RBF is that it can control the distribution of similarity values by constant C. In the experiments, we used $C = 30$, which yields an average similarity value of around 0.5.

A schematic view of the proposed audio search method is shown in Fig. 1, where (A) and (B) show self-similarity matrices of the query and database musical audio signals, respectively. Elements represented by white pixels show similar frame pairs within an audio signal.

In Fig. 1, the query is played seven semitones higher and 20 % faster than the original music in the database. We can see that general patterns of corresponding regions of the query and the database are very similar, which shows that the self-similarity measure affected very little by key variations. We utilize this nature for our musical audio search system.

The region corresponding to the query in the database is larger than the query self-similarity matrix because the query is played faster. To find such variously sized patterns, we propose a method based on DP.

2.2 Self-similarity matching based on DP

DP-based methods that consider pixel-wise deletion and insertion have been developed for two-dimensional image matching [9]. However, in our approach, these methods are not suitable for matching of self-similarity matrices because we do not treat elements of a self-similarity matrix in the same row or column independently. For example, when the i-th frame is deleted, the i-th row and column of the self-similarity matrix disappear together. Therefore, it is only

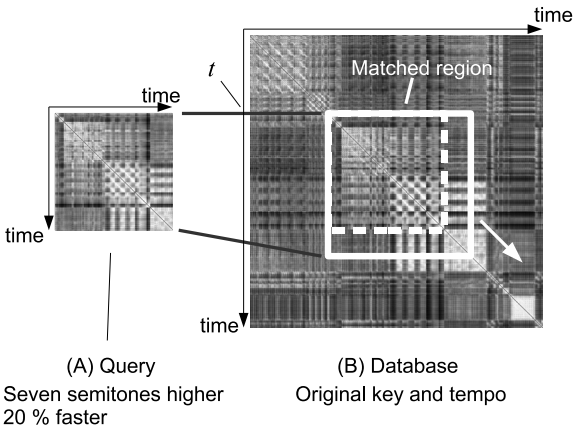

Figure 1. A schematic view of self-similarity matching. (A) Self-similarity matrix of the query. The query is played seven semitones higher and 20 % faster than the original. (B) Self-similarity matrix of the database with the original key and speed. The square region within the thick white line is matched with the query. The part outlined by the dashed line is the same size as the query self-similarity matrix.

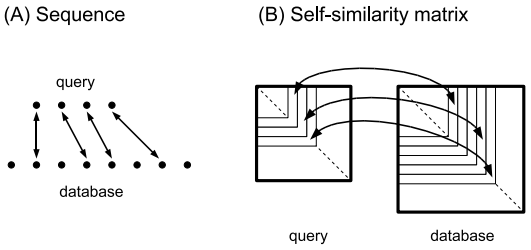

Figure 2. Alignment of (A) sequences and (B) self-similarity matrices.

necessary to search the diagonal direction of database self-similarity matrices because the objective is to find locations in the time direction. For this purpose, we can utilize an alignment framework for a one-dimensional sequence.

DP has been commonly used for matching of two vector or symbol sequences with different lengths. It has been applied for not only audio signals but also for sequences of symbols, such as amino or nucleic acids [10]. This method aligns their elements, including gaps, and yields the alignment score.

The alignment of (A) sequences and (B) self-similarity matrices is illustrated in Fig. 2. When query and database sequences are aligned, each query element corresponds to an element in the database. When self-similarity matrices are aligned, inverted L regions along the diagonal on the matrices are matched, instead of single elements.

The framework of the proposed method is shown in Fig. 3(A). Self-similarity matrices of a database and a query are denoted by $\mathbf{S} = (s_{ij})$ and $\mathbf{R} = (r_{ij})$, where s_{ij} and r_{ij} are

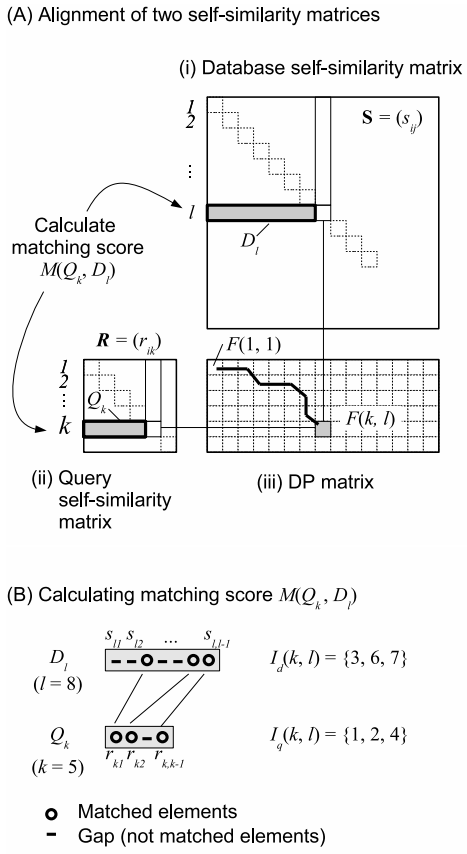

Figure 3. The framework of the proposed method. (A) An illustration of the alignment procedure for two self-similarity matrices. (B) The calculation of matching score $M(Q_k, D_l)$ in eq. (2).

(i, j) elements of $\mathbf{S}$ and $\mathbf{R}$, respectively. Diagonal elements of $\mathbf{S}$ and $\mathbf{R}$ are not used because, having a constant value of 1, they are not informative.

As well as conventional sequence alignment methods, we introduce a $m \times n$ DP matrix $F(k, l)$ where m and n denote the lengths of the self-similarity matrices of the query and database, respectively. $F(k, l)$ means the score of an alignment of the initial k query frames and l database frames.

$F(k, l)$ is recursively calculated from adjacent elements $F(k-1, l-1)$, $F(k-1, l)$, and $F(k, l-1)$ in the following way:

$$F(k,l) = \max \begin{cases} F(k-1,l-1) + M(Q_k, D_l), & \dots(a) \\ F(k-1,l) - g, & \dots(b) \\ F(k,l-1) - g, & \dots(c) \end{cases} \tag{2}$$

where Q_k and D_l denote features at the k-th frame of the query and l-th frame of the database, respectively, $M(Q_k, D_l)$ is the matching score between Q_k and D_l, and g is a penalty for a gap.

The thick line on the DP matrix in Fig. 3(A)(iii) shows a history of the selections in eq. (2) from $(1, 1)$ to (k, l). The vertical part of the line shows a gap in the query and the horizontal parts shows gaps in the database.

When we search aligned subsequences in a query and a database, we can use local alignment technique, where $F(k, 0) = 0$ and $F(0, l) = 0$ are set as initial conditions. This means that initial gaps in both the query and the database are omitted from the alignment.

In this study, we assume that the query is a short excerpt of the database and that the whole query is used for an alignment. Moreover, we do not use the 0-th row and column of $F(k, l)$ because self-similarities are not defined for the first frame of either the query or database. Then, $F(k, 1) = -(k-1)\ g$ and $F(1, l) = 0$ are used for the initial conditions.

Finally, by backtracking the path on the DP matrix from the peaks of the final (m-th) row of $F(k, l)$, we can find the locations corresponding to the query on the database.

2.3 Calculating matching score from self-similarity matrices

For alignment of symbol or vector sequences, $M(Q_k, D_l)$ in eq. (2) is defined using similarity measures of vectors or symbol substitution matrices.

On the other hand, for alignment of self-similarity matrices, the proposed method matches inverted L regions on self-similarity matrices. We consider only the left lower region of self-similarity matrices, i.e., the gray regions within the thick lines in Fig. 3(A), since they are symmetrical. Consequently, Q_k and D_l in eq. (2) can be represented as vectors:

$$\begin{aligned} Q_k &= (r_{k1}, ..., r_{k,k-1}), \\ D_l &= (s_{l1}, ..., s_{l,l-1}). \end{aligned} \tag{3}$$

It is not appropriate to use all elements of Q_k and D_l to calculate $M(Q_k, D_l)$, since, generally, vectors Q_k and D_l have different lengths because they may include elements corresponding to gaps.

To obtain the globally optimum matching score, an exhaustive search for the optimum combination of non-gap elements in Q_k and D_l is needed in each eq. (2)(a) calculation. To reduce the computational cost, we use non-gap elements that are determined by past DP calculations. Namely, we chose elements where the case (a) was selected in past eq. (2) calculations. Rightward diagonal parts of the DP path in Fig. 3(A)(iii) show these elements.

By this approximation, the order of eq. (2) calculations comes down to $O(m^2 n)$ in the worst case, considering the order of $M(Q_k, D_l)$ calculation. In case of conventional vector sequence alignment, by comparison, the order is $O(dmn)$, where d is the vector dimension.

Fig. 3(B) shows the correspondence of elements of Q_k and D_l when the DP path from $(1, 1)$ to (k, l) is that shown

in Fig. 3(A)(iii). The s_{l3} , s_{l6}, and s_{l7} correspond to r_{k1}, r_{k2}, and r_{k4}, respectively. We call these elements "matched elements."

Matched elements of Q_k and D_l are different at every position of (k, l) in the DP matrix even if the k or l is the same. Then, we introduce "matched element indices" $I_q(k, l)$ and $I_d(k, l)$, which indicate sets of indices at each position of (k, l) for the query and database respectively. In Fig. 3(B), $I_q(k, l) = \{1, 2, 4\}$ and $I_d(k, l) = \{3, 6, 7\}$.

Using $I_q(k, l)$ and $I_d(k, l)$, the same sized vectors $\mathbf{q}_k(l)$ and $\mathbf{d}_l(k)$ can be generated at each position of (k, l) on a DP matrix as follows:

$$\begin{aligned} \mathbf{q}_k(l) &= (r_{kj}) : j \in I_q(k, l), \\ \mathbf{d}_l(k) &= (s_{lj}) : j \in I_d(k, l). \end{aligned} \tag{4}$$

Using the above vectors, we define the matching score in eq. (2) as

$$M(Q_k, D_l) = M'(\mathbf{q}, \mathbf{d}) = 0.5 - \frac{1}{a} \sum_{i=1}^{a} |q_i - d_i|, \tag{5}$$

where $\mathbf{q}$ and $\mathbf{d}$ are simplified descriptions of $\mathbf{q}_k(l)$ and $\mathbf{d}_l(k)$, q_i and d_i are their elements, and a denotes the number of dimensions of $\mathbf{q}$ and $\mathbf{d}$. $M'(\mathbf{q}, \mathbf{d})$ ranges from -0.5 to 0.5.

Matched element indices $I_q(k, l)$ and $I_d(k, l)$ can be determined recursively during the DP score calculations in eq. (2) in the following way.

1. Make temporary indices $I'_q(k, l)$ and $I'_d(k, l)$ from $I_q(k-1, l-1)$ and $I_q(k-1, l-1)$ for the calculation of $M'(\mathbf{q}_k(l), \mathbf{d}_l(k))$.

$$\begin{aligned} I'_q(k, l) &= \{j,\ k-1 \mid j \in I_q(k-1, l-1)\}, \\ I'_d(k, l) &= \{j,\ l-1 \mid j \in I_d(k-1, l-1)\}. \end{aligned}$$

2. If case (a) is chosen in eq. (2), $I_q(k, l) = I'_q(k, l)$ and $I_d(k, l) = I'_d(k, l)$.

3. If case (b) is chosen, $I_q(k, l) = I_q(k-1, l)$ and $I_d(k, l) = I_d(k-1, l)$.

4. If case (c) is chosen, $I_q(k, l) = I_q(k, l-1)$ and $I_d(k, l) = I_d(k, l-1)$.

As initial conditions, $I_q(k, 1) = \emptyset$ and $I_d(1, l) = \emptyset$ are used.

2.4 Reducing memory by splitting databases

To make a self-similarity matrix, memory space of $O(n^2)$ is needed. When we use large databases generated from long musical audio signals, we can reduce memory space by splitting a large self-similarity matrix into multiple submatrices with comparatively small size along the diagonal (Fig. 4). This causes few problems because the left lower and right upper parts of the matrix rarely match with a query matrix. To avoid degradation around the boundaries of the submatrices, we allocate the submatrices so that they overlap each other. In experiments, we used 4,000 frames (200 sec.) for the submatrix size with 2,000 (100 sec.) overlapping frames.

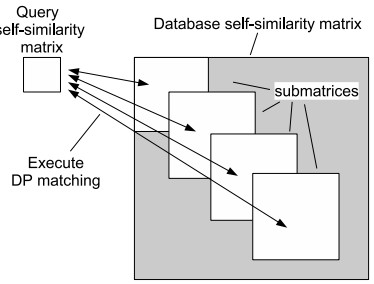

Figure 4. Splitting a self-similarity matrix into overlapping submatrices.

3 EXPERIMENTS

We evaluated the proposed method in a musical audio signal retrieval task with key and tempo variations.

To evaluate the method in a systematic way, we used performances controlled by MIDI data. That is, we used ten pieces of MIDI data in the RWC Music Database (RWC-MDB-C-2001) [11] to generate queries in various keys and tempi and a database.

For the database, we generated audio signals by playing ten MIDI files using $\mathrm{Winamp}^{\mathrm{TM}}$ [2]. For queries, we randomly extracted ten twelve-second-long excerpts.

For the query, we generated three variations of key and tempi: the original key and tempo, the original key with 20 % faster tempo, and the key seven semitones higher than the original with 20 % faster tempo. These variations were made by modifying the original MIDI files.

Each audio signal was sampled at 11,025 Hz and a spectrogram was generated by 144 second-order-IIR bandpass filters with a frame rate of 20 Hz. Central frequencies of these filters were set between 55 and 3,520 Hz with half semitone intervals. Spectral powers were represented in a log scale and normalized to be $|\mathbf{p}_i| = 1$.

The ground truth was determined from the original MIDI files by comparing the constitution of twelve chromatic notes at each frame between the query and database. In total, 15 locations in the database were extracted as the ground truth for ten queries. For five queries, two parts were found in the database and two of them were played seven semitones higher than the original parts.

Accuracies were measured by the percentage of correct identification in top N_i candidate locations, where N_i is the number of correct locations in the database corresponding to the i-th query.

We also examined two existing methods for accuracy comparisons. One is a variant of Nagano et al.'s method [1]

[2] http://www.winamp.com

based on alignment of chroma vector sequences and that also employs the same DP calculation as the proposed method [eq. (2)]. A chroma vector represents the constitution of twelve chromatic note at each frame. Then, we can deal with key variation by applying DP alignment calculations twelve times, rotating the elements of the vector one by one. We call this method "chroma+DP" in this study.

As was done by Nagano et al., we represented a chroma vector as a binary vector. We used a threshold $\mu + \sigma$, where μ and σ are the average and the standard deviation of the original chroma vector elements at each frame. For simplicity, we did not apply harmonics elimination or other conversions, which is the same as for the proposed method.

To calculate matching scores in eq. (2) for each frame, we used a measure based on the Dice's coefficient and defined as

$$2n_{qd}/(n_q + n_d) - 0.5, \quad (6)$$

where n_q and n_d are the numbers of elements of chroma vectors that exceed the threshold for the query and database, respectively. The n_{qd} denotes the number of elements that exceed the threshold for both the query and database. It is equivalent to the similarity measure used in [1] except that our measure is made by subtracting 0.5.

The other previous method we examined is the one we have proposed [8], which also uses self-similarity of audio signals. The method simply matches two vector sequences without considering gaps. Instead, it weights vectors according to time distance from start points. We call this method "self-similarity vector matching."

We tested three variations of gap penalty, namely, $g = 0.1$, 0.3, and 0.5, for the proposed method and for chroma+DP. Neither method was sensitive to g, and g=0.1 for the proposed method and g=0.5 for chroma+DP yielded the lowest error rate in total.

Fig. 5 shows the search accuracy for each method using the best g values as described above. The proposed method yield high accuracy for all three key and tempo patterns. Noteworthy is that it outperforms the other methods when both key and tempo variations are large.

The accuracy for original key and the original tempo are a little lower than in other cases using the proposed method. This is because the proposed method using queries with the original keys and tempi failed to identify the two positions in the database played in the key seven semitones higher.

Totally, the results indicate that the self-similarity features are robust to key variations and that the alignment method based on DP works effectively for the self-similarity matrix matching.

4 CONCLUSION

We have proposed a musical audio signal search method based on DP matching of self-similarity matrices. By introducing indices that represents the history of the DP path, the problem is reduced to the one that a sequence alignment scheme can be applied. The experimental results show that a self-similarity matrix can represent a key-invariant structure of musical audio signal and that the DP framework works very effectively for dealing with tempo variations. We will evaluate the method using larger data sets, including sets containing acoustically played musical signals and improve it to deal with other musical variations, such as variations of instruments and arrangements.

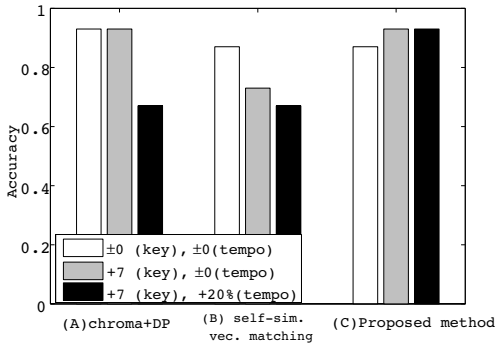

Figure 5. Search accuracy of (A) chroma + DP, (B) self-similarity vector matching, and (C) the proposed method. The "+7 (key)" and "20 % (tempo)" mean the queries are played in a key seven semitones higher at a 20 % faster tempo, respectively.

5 REFERENCES

[1] Nagano, H. et al. "Fast Music Retrieval Using Polyphonic Binary Feature Vectors," *Proc. of ICME*, Vol. 1, 101–104, 2002.

[2] Müller, M. et al. "Audio Matching via Chroma-Based Statistical Features," *Proc. of ISMIR*, 2005.

[3] Ellis, D. P. W. and Poliner, G. E. "Identifying 'Cover Songs' with Chroma Features and Dynamic Programming Beat Tracking," *Proc. of ICASSP*, IV-1429–IV-1432, 2007.

[4] Tsai, W.-H. et al. "A Query-by-Example Technique for Retrieving Cover Versions of Popular Songs with Similar Melodies," *Proc. of ISMIR*, 2005.

[5] Foote, J. "Visualizing Music and Audio using Self-Similarity," *Proc. of ACM Multimedia*, 77–80, 1999.

[6] Foote, J. "Automatic Audio Segmentation using a Measure of Audio Novelty," *Proc. of ICME*, Vol. 1, 452–455, 2000.

[7] Marolt, M. "A Mid-level Melody-based Representation for Calculating Audio Similarity," *Proc of ISMIR*, 2006.

[8] Izumitani, T. and Kashino, K. "A Musical Audio Search Method Based on Self-Similarity Features," *Porc. of ICME*, 68–71, 2007.

[9] Uchida, S. and Sakoe, H. "A Monotonic and Continuous Two-Dimensional Warping Based on Dynamic Programming," *Proc. of ICASSP*, Vol. 1, 521–524, 1998.

[10] Durbin, R. et al. "Biological Sequence Analysis," *Cambridge University Press*, 1998.

[11] Goto, M. et al. "RWC Music Database: Popular, Classical, and Jazz Music Databases," *Proc. of ISMIR*, 287–288, 2002.