A MUSICAL AUDIO SEARCH METHOD BASED ON SELF-SIMILARITY FEATURES

Tomonori Izumitani and Kunio Kashino

NTT Communication Science Laboratories
3-1, Morinosato Wakamiya, Atsugi-shi, Kanagawa, Japan

ABSTRACT

We propose a method for musical audio search based on signal matching. A major problem in the signal matching approach to musical audio search has been key variation; if the key of a query signal is significantly different from the one in the stored database, the search will fail. To cope with this problem, our method newly employs self-similarity as the feature for signal matching. The self-similarity proposed here is similarity of the power spectrum defined between two time points within an audio signal. We show that the method increases the robustness of musical audio search with respect to key variation. In our experimentation, for example, the proposed method yields precision and recall rates of around 0.75 even when the pitches in queries and stored signals differ from each other by seven semitones, whereas a conventional signal matching method does not produce meaningful results in such a case.

1. INTRODUCTION

Musical audio search, or finding pieces of audio signal from a database, has become important with the rapid diffusion of portable music players and multi-function cell phones. Among various methods for musical audio search, the approach based on signal similarity is often referred to as audio fingerprinting. Most audio fingerprinting methods have been based on a set of local characteristics derived from sound waveforms [1], such as the short-time power spectrum and Mel frequency cepstral coefficients.

While the audio fingerprinting systems have been successfully used in various practical applications, we have been interested in extending the task to searching for not only the same recordings as a query but also the same titles, which may be played by other players, with other instruments, with different arrangements, in different tempi, or in other keys. Obviously, local characteristics of sound waves used by audio fingerprinting systems may be severely affected by these variations. As an initial study, this paper focuses on a method to cope with key variations.

An approach addressing the key variation problem is to use chroma-based features. For example, Nagano et al. proposed PBFV (polyphonic binary feature vector) [2] for key-invariant music retrieval. Goto also used a chroma-based feature to detect repetition within a musical piece in his chorus-section detecting system [3]. Müller et al. used chroma-based features to unify different timbres played by various players [4].

However, when we consider musical audio search for a large database, finding key-converted music remains a difficult task. For example, when using a chroma-based feature, the computational cost can be nonnegligible because twelve matching processes, one for each is a semitone in an octave, are required for pitch-invariant search. In addition, chroma-based features are derived from individual frequency components, and therefore, they can be affected by overtone characteristics of the notes, and sometimes even by tuning frequency.

The main idea proposed here is to use a pitch-invariant feature. In our method, a self-similarity vector, whose elements are defined by spectral similarities to different lags, is generated at each time point. Then, we introduce an unequally-sized mesh to extract features from the self-similarity vectors to deal with tempo variations.

Self-similarity in music has already been investigated by many researchers, especially in the context of musical structure analysis. For example, Foote used self-similarity of the spectral configuration as a visualization of music [5] as well as a means of audio segmentation tasks [6]. Self-similarity representation reflects the structure of the relationship between various time points in a comparatively long time range and it does not include features dependent on pitch. We apply this property to pitch-invariant music search tasks.

2. METHODS

2.1. Audio Search Framework

We consider the following musical audio search system. Each database entry is constructed from a musical audio signal and a query is a short piece of musical audio signal. The task is to search the database to find the same parts of music as a given query.

Figure 1 shows a basic idea of the search. Most existing audio fingerprinting system represent features as a sequence of vectors or symbols generated by frequency analysis. These features are suitable for finding the same recordings as given queries. When a musical piece is played in different keys, however, the features often change dramatically, and it is not

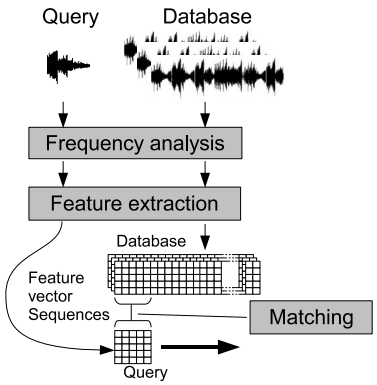


Fig. 1. A simple audio search system.

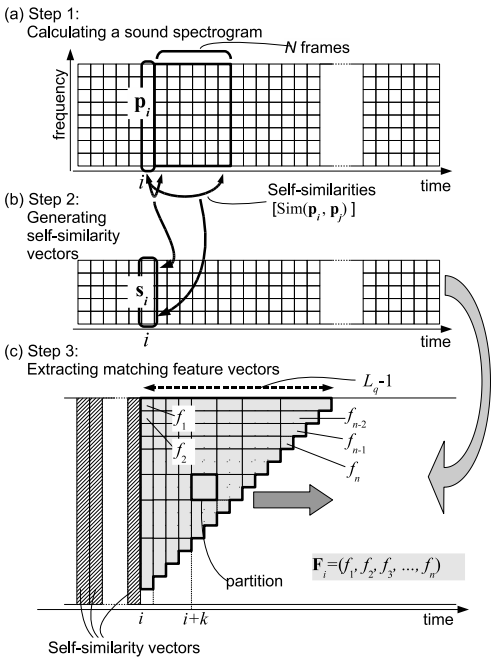


Fig. 2. Extracting features from a spectrogram of musical audio data.

straightforward to design a criterion for the search to allow for this difference.

2.2. Calculating self-similarity features

We assume that the relationship between two time points within a musical sound is invariant even if the music is played in another key. Thus, as shown in Fig. 2, we extract features, both from the stored and query signals, with the following three-step process.

In the first step, the spectral power vectors are calculated using a bandpass filter bank [Fig. 2(a)]. The filtered signals are divided into frames and the powers within each frame are averaged. A power spectrum at time point i, which means the i-th frame, is represented by a vector $\mathbf{p}_i$.

In the second step, self-similarity vectors $\mathbf{s}_i$ are calculated from spectral power vectors. Each of the vectors $\mathbf{s}_i$ is generated by calculating the similarity between $\mathbf{p}_i$ and each subsequent N frames $\mathbf{p}_j$ (j = i+1, i+2, ..., $i + N$) [Fig. 2(b)].

In this study, we adopt the radial basis function (RBF), which reflects the Euclidean distance between two vectors, for the similarity measure:

$$s_{ij} = \mathrm{Sim}(\mathbf{p}_i, \mathbf{p}_j) = \exp\left(-C|\mathbf{p}_i - \mathbf{p}_j|^2\right), \qquad (1)$$

where C is a constant.

Note that some elements of N-dimensional vector $\mathbf{s}_i$ are not obtained at the tail frames where the numbers of subsequent frames are less than N. Especially for a short query having L_q ($\leq N$) frames, only the first $L_q - k$ elements are available for k-th frame.

In the third step, matching feature vectors $\mathbf{F}_i$ are extracted. As in Fig. 2(c), we firstly apply a triangular window for calculating a matching feature. We then divide the triangular region into partitions using a mesh. The elements of self-similarity vectors (s..) within each partition are averaged, and the averaged value is used as a feature vector element. In the figure, f_1, f_2, ..., and f_n indicate feature values and these values constitute a feature vector $\mathbf{F}_i$. This averaging process reduces the number of dimensions of matching features.

We particularly use a mesh that contains unequally sized rectangular partitions; the upper left partitions are small and the lower or right ones are large. This is to deal with various tempi. When a query music is played in a different tempo from the corresponding music in the database, the time difference between the corresponding time frames become large as the time lags increase. We use larger averaging regions in lower or right hand side elements to absorb this effect.

The length of a side of the partition, $L_p(k)$, that includes k-th frame or the k-th element in the upper left, is defined as

$$L_p(k) = \left\lfloor \frac{k}{M} \right\rfloor + 1, \qquad (2)$$

where k denotes the k-th frame in the horizontal direction or the k-th element in the vertical direction. The value of k represents the distance from the upper-left position of the triangular region. M is a constant that controls the ratio of the size between the upper-left partition and the lower or right ones.

2.3. Matching process

The feature matching can be performed as in the exisiting audio fingerprinting systems. For example, hash functions or feature histograms may be used to accelerate the search [1]. For simplicity, here we describe a basic implementation based on the exhaustive matching with a one-by-one shift of a query feature vector (Fig. 1). The matching score of the system is based on the Euclidean distance between feature vectors from

query $\mathbf{F^q}$ and that from database $\mathbf{F^d}$. The Euclidean distances for all frames within the database are calculated and every frame that has a minimum distance value within a certain range around the frame (± 200 frames are used in this study) is selected as matched-frame candidates. To define a similarity measure, we represent the distance value at the i-th frame in the database as z_i and the set of candidate frames as D. Then we define the matching score S_i using the average z_m and standard deviation z_d of z_j, ($j \in D$) as follows:

$$S_i = \begin{cases} \max\left(-\frac{(z_i - z_m)}{z_d},\ 0\right), & i \in D, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

The system lists all frames in the database that yield $S_i > \Theta$, where Θ is a threshold parameter.

3. EXPERIMENTS

We evaluated robustness of the proposed method in a music signal retrieval task with the key and tempo variations.

The objective of the test was to evaluate the robustness in a systematic way. To this end, we used performances controlled by MIDI data. That is, we used 61 pieces of MIDI data in the RWC Music Database (RWC-MDB-C-2001) [7] to generate queries in various keys and tempi and a database.

First, we generated audio signals of the database by playing all 61 MIDI files using $\mathrm{Winamp}^{\mathrm{TM}}$ [1] and saving the files in the WAV format. Next, we randomly selected short segments, one from each of the pieces, and used them as a test set of queries. By modifying the MIDI files, we prepared 35 variations for each query, namely, seven key variations where the keys were shifted ± 0, ± 1, ± 4, ± 7 semitones from the original and five tempo variations where the interval between beats was shortened or extended to be 90%, 95%, 100%, 105%, and 110% of the original. The length of each query was 12 seconds.

Each audio signal was resampled at the sampling rate of 11,025 Hz and a spectrogram was generated by a filter bank consisting of 120 second-order-IIR bandpass filters. Central frequencies of the bandpass filters were set to have half semitone intervals as follows:

$$F_c(i) = 100\ (\mathrm{Hz}) \times 2^{\frac{(i-1)}{24}},\ \ i = 1, 2, ..., 120. \quad (4)$$

Powers of each signal were averaged at every time frame from 1,024 samples, represented in a log scale, and normalized to be $|\mathbf{p}_i| = 1$. Some parameter values were empirically chosen; the similarity parameter C was 30, and the mesh parameter M was 8.

The correct locations of each query within the database were determined manually for the evaluation. In total, 82 locations in the database were specified as the correct locations for 61 queries. For 18 queries, each query had more than one location within a musical piece. This is because some musical passages were repeated in the same musical pieces.

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

Table 1. Precision "Prec." and recall "Rec." rates when the matching score threshold Θ was fixed for all query variations. Queries are played (A) at the original tempo and (B) at a faster tempo (beat interval is 95 % of the original). Values in parenthesis denote baselines.

(A) The original tempo.

	-7	-4	-1	± 0	$+1$	$+4$	$+7$
Prec.	0.92	0.83	0.86	0.82	0.89	0.89	0.88
	(0.0)	(0.33)	(0.94)	(0.86)	(0.89)	(0.50)	(0.50)
Rec.	0.41	0.60	0.88	0.90	0.82	0.78	0.68
	(0.0)	(0.01)	(0.88)	(0.95)	(0.88)	(0.02)	(0.01)

(B) Played at a faster tempo (beat interval: 95 %).

	-7	-4	-1	± 0	$+1$	$+4$	$+7$
Prec.	0.84	0.86	0.89	0.86	0.89	0.92	0.88
	(0.0)	(0.0)	(0.90)	(0.87)	(0.89)	(1.00)	(1.00)
Rec.	0.26	0.38	0.76	0.82	0.72	0.55	0.45
	(0.0)	(0.0)	(0.57)	(0.83)	(0.59)	(0.02)	(0.01)

The performance was measured in terms of the precision and the recall rates. The precision is defined as the ratio of correctly identified locations to the total number of locations found by the system, and the recall as the ratio of correctly identified locations to the total number of correct locations.

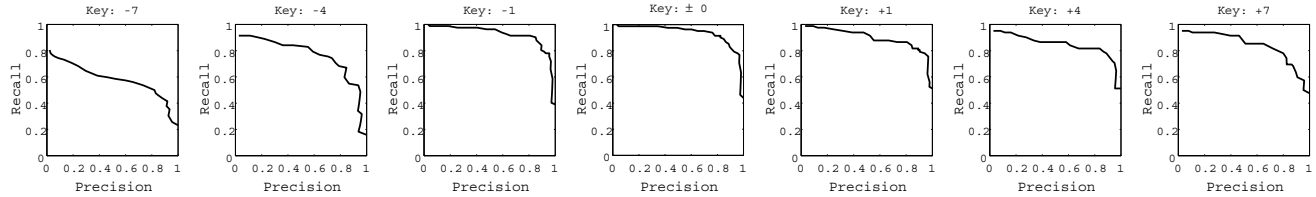
Figure 3 shows the precision-recall curves using queries played at (A) the original tempo, (B) a faster tempo with 95% beat intervals, and (C) with 90% beat intervals. These curves were obtained by changing the matching score threshold Θ. The results are from queries played in various keys, namely, the original pitch and pitches shifted -7, -4, -1, ± 0 (the original), $+1$, $+4$, and $+7$ semitones from the original.

When a query was played at the original tempo or with 95% beat intervals, the method efficiently worked. With the original tempo, precision and recall rates were simultaneously greater than 0.75 even in the case of the seven-semitones higher keys. When the tempo was altered to 90% beat interval, the accuracy decreased especially with large key alterations.

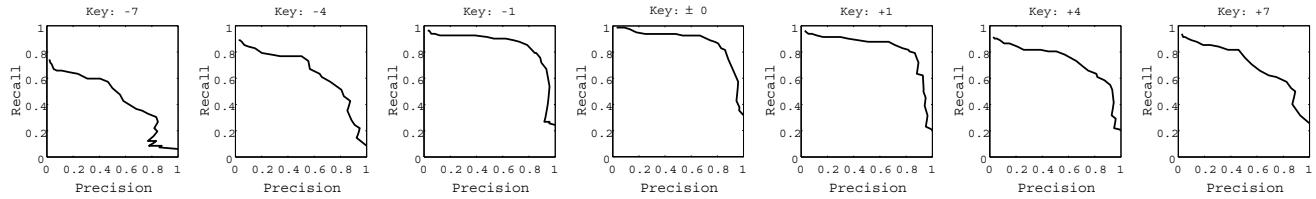
When queries were played at slower tempi, where beat intervals were 105% and 110%, the tendency seen in the precision-recall curves was almost the same as for faster tempi, 95% and 90%, respectively.

Table 1 shows precision and recall rates when the matching score threshold Θ was fixed for all query variations. Values in parenthesis denote baselines calculated using spectrum-based features; namely, $\mathbf{p}_i$ was used instead of $\mathbf{F}_i$. For the baseline test, the dimension of each $\mathbf{p}_i$ was adjusted to 20 by averaging the values within six neighboring frequency bins. Since there were 128 time frames in a query, the number of dimensions of the matching feature vector was 2,560 in the baseline method. The threshold value Θ was chosen so that the precision value averaged over the 3 conditions (key shifts -1, ± 0, $+1$; 100% beat interval) equals the recall value averaged over them; it was 2.9 for the proposed method and 3.8

(A) Played in the original tempo



(B) Played at a faster tempo (beat interval: 95%)



(C) Played at a faster tempo (beat interval: 90%)

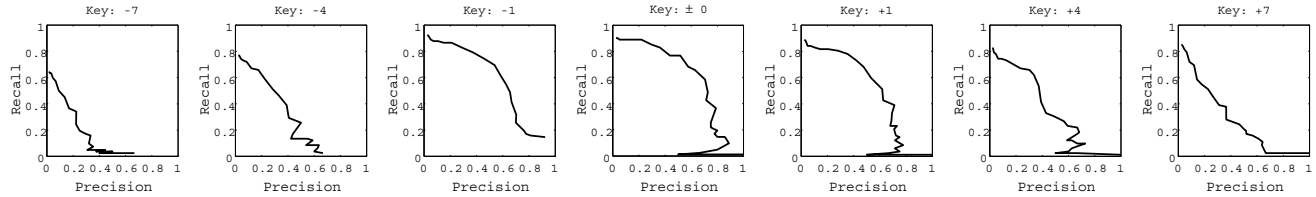


Fig. 3. Precision-recall curves using queries played in various keys and tempi. (A) played in the original tempo, (B) played faster (beat interval: 95%), (C) played faster (beat interval: 90%). Columns of the charts show results from queries with various keys, -7, -4, -1, ± 0(the original key), $+1$, $+4$, and $+7$ semitones shifted from the original, left to right in order.

for the baseline method.

We can see that the recall rates from the baseline method are very low when the queries are played in keys ± 4 and ± 7 semitones shifted from the original. This means that it is difficult for the baseline method to retrieve the desired signals in such cases. Thus, it is shown that the proposed method outperforms the baseline method in terms of robustness to key variations.

4. CONCLUSION

We have proposed a musical signal search method. The method employs self-similarity features to deal with audio signals played in keys significantly different from the ones in the database. Experimental results using systematically generated musical audio signals indicate that the proposed method achieves significantly high accuracy, around the break-even rate of 0.75, even when the pitches in queries and stored signals differ from each other by seven semitones. We anticipate that future work will include evaluation with acoustically played musical signals and extension to the other musical variations, such as tempo and arrangement.

5. REFERENCES

[1] K. Kashino, T. Kurozumi, and H. Murase, "A quick search method for audio and video signals based on histogram pruning," *IEEE Trans. Multimedia*, vol. 5, no. 3, pp. 348–357, 2003.

[2] H. Nagano, K. Kashino, and H. Murase, "Fast music retrieval using polyphonic binary feature vectors," in *Proc. of ICME*, pp. 101–104, 2002.

[3] M. Goto, "A chorus-section detecting method for musical audio signals," in *Proc. of ICASSP*, pp. 437–440, 2003.

[4] M. Müller, F. Kurth, and M. Clausen, "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness," in *Proc. of ISMIR*, pp. 11–18, 2005.

[5] J. Foote, "Visualizing music and audio using self-similarity," in *Proc. of ACM Multimedia*, pp. 77–80, 1999.

[6] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Proc. of ICME*, pp. 452–455, 2000.

[7] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical, and jazz music databases," in *Proc. of ISMIR*, pp. 287–288, 2002.