BACKGROUND MUSIC IDENTIFICATION THROUGH CONTENT FILTERING AND MIN-HASH MATCHING

Chih-Yi Chiu[*], *Dimitrios Bountouridis*[†], *Ju-Chiang Wang*[†], *and Hsin-Min Wang*[†]
[*] Department of Computer Science and Information Engineering, National Chiayi University, Taiwan
[†] Institute of Information Science, Academia Sinica, Taiwan

ABSTRACT

A novel framework for background music identification is proposed in this paper. Given a piece of audio signals that mixes background music with speech/noise, we identify the music part with source music data. Conventional methods that take the whole audio signals for identification are inappropriate in terms of efficiency and accuracy. In our framework, the audio content is filtered through speech center cancellation and noise removal to extract clear music segments. To identify these music segments, we use a compact feature representation and efficient similarity measurement based on the min-hash theory. The results of experiments on the RWC music database show a promising direction.

Index Terms— fingerprint identification, copy detection, content-based retrieval, min-hash

1. INTRODUCTION

Fast network connections and advanced capturing techniques have multiplied, legally or otherwise, the dissemination of multimedia content worldwide. Content owners seek for efficient and effective solutions to manage the source and its copies. In digital music domain, some industrial systems like Snocap, Music2Share, and Shazam provide mature techniques in music sharing and search applications; they are suitable to identify the "clear" music without much speech/noise interfered. However, background music that is usually quality-degraded and mixed with lots of noise elements (e.g., asides, dialogs, and environmental sounds) makes the identification much more challenging.

In conventional background music identification methods, the audio signal is first decomposed into numerous small components [1-4]. Each of which is then compared with the source music data, and the result is aggregated with each other to determine the identity of the background music. This approach has two main disadvantages. First, all the decomposed components are compared with the source music data; since many of them are speech/noise, such comparisons are unnecessary. Second, as a small component conveys insufficient information, it is apt to retrieve many false positives, which might interfere with the aggregation.

To alleviate the above problems, we propose a novel framework for background music identification. Two practical steps are employed to analyze the audio signal. First, based on our observation in TV broadcasting and video, we leverage the stereo format capability to cancel the speech that is panned to the center. Second, for noise and speech that cannot be cancelled, they are located through GMM-based classification and removed to obtain clear music snippets for identification. Thus, we can reduce unnecessary comparisons and false positives.

To achieve efficient and accurate identification, we present a min-hash-based feature representation and similarity measurement. Each music frame is represented by a one-dimensional min-hash value, and the similarity between two music segments is computed by their hash collision rate. Fig. 1 shows an overview of the proposed framework. We validate the framework through several experiments performed on the RWC music database [5].

The remainder of this paper is organized as follows. Sections 2 and 3 describe the content filtering and min-hash indexing, respectively. Sections 4 and 5 summarize the experiment results and conclusions, respectively.

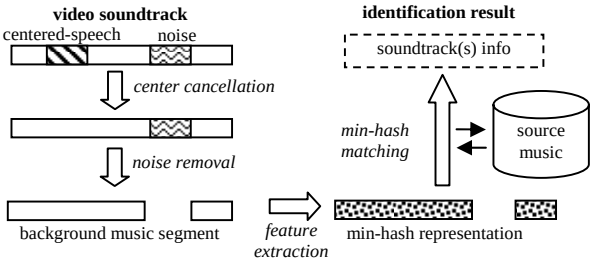


Figure 1. An overview of the proposed framework

2. CONTENT FILTERING

2.1. Speech Center Cancellation

Some knowledge derived from practical sound-editing or post-production used in the TV/movie industry can be employed. For example, in a movie or TV program, the sound editor usually tries to achieve a perceptually clear speech signal because it contains the information that viewers would probably want to focus on. Generally, this is accomplished by leveraging the stereo format capabilities, i.e., the

original stereo music track is panned to the left and/or right, while speech is panned to the center [6]. In other words, the left and right channels of the video-derived audio would contain the same speech information but probably different music signals. Let L and R be the left and right tracks, M_L and M_R be the left and right music signals, and S be the speech signal. A typical scenario would be as follows:

$$L = S + M_L, \quad (1)$$
$$R = S + M_R. \quad (2)$$

Subtracting Eq. (1) from Eq. (2) yields

$$L - R = M_L - M_R. \quad (3)$$

Eq. (3) shows that speech is cancelled and a music signal $M_L - M_R$ is obtained. Suppose that the source stereo music MS is panned left and right by g% and k%, respectively, then M_L and M_R would be as follows:

$$M_L = g\,MS_L + (1-k)\,MS_R, \quad (4)$$
$$M_R = (1-g)\,MS_L + k\,MS_R. \quad (5)$$

Therefore, Eq. (3) can be further derived as:

$$L - R = M_L - M_R = (2g-1)\,MS_L - (2k-1)\,MS_R. \quad (6)$$

If $g = k$, then Eq. (6) is simplified as:

$$M_L - M_R = (2g-1)(MS_L - MS_R). \quad (7)$$

The result in Eq. (7) is a mono track containing the source MS information with lower amplitude and inversed phase; its normalized real-value-based FFT is identical to the source mono music, i.e., $0.5(MS_L + MS_R)$. Thus, it manifests that speech center cancellation would yield the same FFT-based feature for the source mono music and the center-cancelled music, as long as the original soundtrack is evenly panned to the left and right channels. This assumption actually agrees with the most usual case in professional produced TV/movie videos. In practice, we also apply speech center cancellation to our source music data to make the feature representation consistent.

2.2. Noise Removal

Ideally, after speech center cancellation, one should have a signal containing either silence or clear music. In fact, movies/TV programs sometimes contain different noise elements, such as environment sounds or speech panned at non-center positions. Thus, we analyze the mono audio signal derived from the previous subsection to remove the noise segments and reserve the clear music segments for identification. For this task, we employ a GMM-based classification approach that uses two models (24 mixtures each): M-GMM fitted to a training set of pure music and N-GMM fitted to a training set of noise (including silence, speech, effects, environment sounds, etc.). MFCC features are extracted every 10ms with a window size of 20ms for training. Both models range from 50 to 5kHz.

Given an audio signal, we first apply a spectral change detector proposed by Duxbury *et al.* [7] to locate the boundaries of homogenous audio segments. Afterwards, we classify each audio segment as "pure music" or "noise" based on the M-GMM and N-GMM models. The consecutive segments that are classified as music are concatenated into a larger segment.

3. MIN-HASH MATCHING

After the background music segments are extracted, each segment will be matched with the soundtracks in the source dataset. We propose a compact feature representation and efficient similarity measurement based on the min-hash theory for background music identification.

Let $Q = \{q_i \mid i = 1, 2, \ldots, n_Q\}$ be a background music segment (denoted as a query) with n_Q frames, where q_i is the i-th query frame; and let $S = \{s_j \mid j = 1, 2, \ldots, n_S\}$ be a source soundtrack with n_S frames, where s_j is the j-th source frame, and usually $n_Q << n_S$. Our goal is to quickly and accurately find a sequence in S for the given Q if Q is derived from the sequence by copying, editing, and/or transformation.

The audio signal is first divided into overlapping frames of 370ms with a hop size of 11.6ms. Then, the Sign of Energy Difference (SED) [8] is employed to represent each audio frame. SED is a 32 dimensional binary vector, which models FFT energy differences in the neighborhood bands. We denote $F_{x,y}$ as the y-th element ($y = \{1, 2, ..., 32\}$) of the SED feature for frame x.

A sliding window is used to scan S. Let $W = \{s_k, s_{k+1}, ..., s_{k+n_Q-1}\}$ be a windowed sequence with n_Q frames, where $k \in \{1, n_S-n_q+1\}$. A conventional method is to use the *Jaccard coefficient* to compute the similarity between Q and W, which is defined as:

$$J(Q,W) = \frac{\sum_{i=1}^{n_Q} |Y_{q_i} \cap Y_{s_{k+i-1}}|}{\sum_{i=1}^{n_Q} |Y_{q_i} \cup Y_{s_{k+i-1}}|}, \quad (8)$$

where $|\cdot|$ denotes the cardinality; $Y_{q_i} = \{y \mid F_{q_i,y} = 1\}$ is the non-zero index set of q_i's SED feature, and Y_{s_j} is the non-zero index set of s_j's SED feature. The computation cost of Eq. (8) is about $O(|F_{x,y}| \cdot n_Q)$.

The above computation cost can be reduced by utilizing the approximation of the Jaccard coefficient, e.g., *min-hash* Baluja and Covell [9] used random permutations of the indices and recorded in which position the first "1" occurred as the min-hash. However, for the purpose of efficiency, following [10], we simply associate a frame feature with hash values and take the minimum as the min-hash. The probability of two feature vectors having the same min-hash value is proportional to their Jaccard coefficient. As each frame is calculated by a one dimensional min-hash, the computation cost for matching is reduced substantially.

To generate hash values of a SED feature, we apply the hash method in [11]. In the method, r random vectors, each with the same dimension as the SED feature vector, are generated. For each frame, its SED feature vector is projected to r random vectors to generate r hash values. Using the median of these hash values as a threshold, we generate a bit "1" if a hash value is greater than the threshold; otherwise, we generate a bit "0". Subsequently, we have an r-dimensional binary vector for the frame. Here r = 10.

Let $Z_{x,z}$ be the z-th element of frame x's binary hash vector. We take the index of the first non-zero hash value of the hash vector as the min-hash:

$$H_x = \min\left(z \mid Z_{x,z} = 1\right). \tag{9}$$

Then, the min-hash similarity can be estimated by:

$$M(Q,W) = \frac{\sum_{i=1}^{n_Q} |H_{q_i} \cap H_{t_{k+i-1}}|}{\min(n_Q, \sum_{i=1}^{n_Q} |Y_{q_i} \cup Y_{t_{k+i-1}}|)}. \tag{10}$$

Our preliminary study shows that n_Q was usually smaller than $\sum_{i=1}^{n_Q} |Y_{q_i} \cup Y_{s_{k+i-1}}|$. Therefore, we can replace the denominator by n_Q. As a result, the computation cost is O(n_Q).

4. EXPERIMENTS

We evaluated the proposed framework on the classical music part of the RWC music database [5]. Fifty music files with a total of 386 minutes playing time served as the source data. These files were converted to a uniform 22kHz, 16-bit format in stereo. Three experiments were conducted against various music degradations and transformations. For convenience, the query files were artificially generated by simulating practical TV/movie video production to mix music with speech and noise; this offers us an easy way to investigate the performance in terms of various accuracy metrics.

4.1. Accuracy of Music Sequence Matching

The first experiment evaluated min-hash matching. Seven test queries ranging from 10 seconds to 30 seconds were excerpted from the source dataset. The following degradations and transformations that are widely used in music/video production are applied to these test queries:

- all pass filtering using the system function $H(z) = (0.81z^2\text{-}1.64z+1)/(z^2\text{-}1.64z+0.81)$;
- amplitude compression with the following ratios: 4.47:1 for $|A| \geq$ -28.6 dB, 0.86:1 for -46.4 dB$<|A|<$ -28.6 dB, and 1:1.61 for $|A| \leq$ -46.4 dB;
- equalization with the setting of [8];
- band-pass filtering using a second order Butterworth filter with cut-off frequencies at 100Hz and 6000Hz;
- noise addition with uniform white noise with a maximum magnitude of 512 quantization steps;
- MPEG1-Layer3 using the Lame algorithm at 64/96/128 kbps, and the Fraunhaufer algorithm at 96kbps;
- AAC at 96kbps;
- Microsoft ADPCM;
- MPEG1-Layer2; and
- Windows Media Audio at 28.8/48/96 kbps.

As a result, we obtained a total of 105 (7×15) music queries to be used to search in the source dataset.

The *recall* and *precision* rates are used as the accuracy metrics. Unlike other identification work where the source soundtrack id with the highest similarity score, called *top-1 precision*, is returned to users, our matching method further checks whether the exact location in the source soundtrack is returned. We believe it would be helpful for users if the desired content can be located explicitly. A search result is deemed correct if the detected window has any overlap with the region from which the query was extracted. While this is a very wide tolerance, the average overlap ratio can achieve 94% in our study.

Table 1 lists the recall and precision rates derived by using the Jaccard coefficient in Eq. (8) and the min-hash similarity in Eq. (10) with different thresholds θ. Surprisingly, the min-hash similarity outperforms the Jaccard coefficient in most cases. However, as will be shown in the later experiment, short query sequences usually yield drastic variations of the min-hash similarity and thus degrade the precision rate. In contrast, the Jaccard coefficient is less affected by short query sequences. We also consider the top-1 precision. Both the Jaccard coefficient and the min-hash similarity achieved the perfect 100% rate.

Table 1. Recall (R) and precision (P) rates of the Jaccard coefficient and the min-hash similarity

		$\theta = 0.5$	$\theta = 0.55$	$\theta = 0.6$	$\theta = 0.65$	$\theta = 0.7$
Jaccard	**R**	1.0000	1.0000	0.8381	0.7333	0.6667
	P	0.8167	0.8750	1.0000	1.0000	1.0000
min-hash	**R**	1.0000	1.0000	0.9810	0.8381	0.7619
	P	0.7745	0.9833	1.0000	1.0000	1.0000

4.2. Accuracy of Noise Removal

The same music queries used in the previous subsection were applied. Each query was mixed with effects or other artifacts so that it consisted of 67% music part and 33% noise part. Degradations and transformations were applied to obtain a new query set for this task. We employed a query-based metric to measure the accuracy. Suppose that the groundtruth contains s clear music segments, each with M_s frames. For the t query segments, each of which contains N_t clear music frames and $\tilde{N}_t$ noise frames. The accuracy is defined as:

$$\frac{\sum_t (N_t - \tilde{N}_t)}{\sum_s M_s}. \tag{11}$$

The accuracy of a random guess for a query that considers the whole audio clip as "music" is about 50% ((67-33)/67*100%). The average performance of the proposed method is 77.78% and 80.63% with/without degradations and transformations, respectively.

4.3. Accuracy of the Proposed Method

The final experiment evaluated the whole framework. For simplicity, the query files were only converted to the MP3 and WMA families, which are the most representative formats in PC and Web applications. Three configurations are set to deal with these query files:

- speech center cancellation but no noise removal (abbreviated as "CC");
- speech center cancellation and noise removal ("CC+NR");
- speech center cancellation and manual noise removal, ("MANUAL").

In addition, we implemented two matching schemes. The first scheme used the min-hash similarity only. The second is a *coarse-to-fine* scheme, which used the min-hash similarity for coarse matching first, and then the candidates were further checked by the Jaccard coefficient for fine matching.

The performance in terms of the recall and precision rates at θ = 0.55 and the top-1 precision rate are listed in Table 2. It is obvious that if we use the complete video soundtrack clip to search the source soundtracks, i.e., "CC," the accuracy is poor because a large portion of a query clip contains noise. "CC+NR" and "MANUAL" yield noticeable improvements in the recall rates thank to the removal of noise. At the same time, however, their precision rates drop significantly by using the min-hash matching scheme only. This is because after noise removal, a query clip is usually partitioned into several short sequences. Since a short sequence contains less information, the min-hash similarity becomes very sensitive, and thus many false positives will be retrieved. In contrast, the Jaccard coefficient is less affected. Fig. 2 shows an example of using a short query sequence to match a source soundtrack, where the X-axis and Y-axis indicate the frame index and the similarity score, respectively. With the coarse-to-fine matching scheme, we can improve the precision rate effectively.

5. CONCLUSIONS

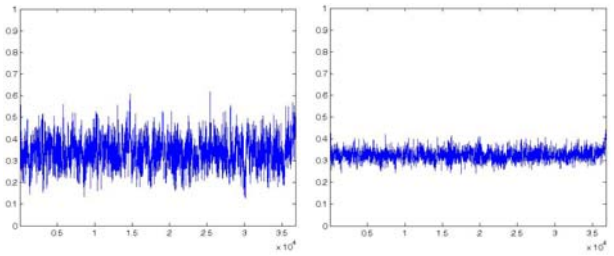
We have proposed a novel framework for background music identification. The framework is realized through effective content filtering and efficient min-hash matching. The experiments on the RWC database with various degradations and transformations manifests a promising result. We will continue exploring its real-applications in TV broadcasting and video blog.

6. ACKNOWLEDGEMENTS

This work was supported in part by the National Science Council of Taiwan under Grants: NSC 98-2218-E-415-003 and NSC 98-2631-001-013.

Table 2. Recall (R) and precision (P) rates of different query generation and sequence matching methods

		CC	*CC+NR*	*MANUAL*
Min-hash	**R (θ = 0.55)**	0.6000	0.9200	0.9561
	P (θ = 0.55)	0.7778	0.1034	0.0706
	P (top-1)	1.0000	0.9867	0.9474
min-hash + Jaccard	**R (θ = 0.55)**	0.6000	0.9067	0.9473
	P (θ = 0.55)	0.7778	0.8608	0.8182



(a) **(b)**

Figure 2. (a) The min-hash similarity; **(b)** the Jaccard coefficient

7. REFERENCES

[1] M. Abe and M. Nishiguchi, "Self-optimized spectral correlation method for background music identification," *ICME*, 2002.

[2] H. Nagano, K. Kashino and H. Murase, "A Fast Search Algorithm for Background Music Signals Based on the Search for Numerous Small Signal Components," *ICASSP*, 2003.

[3] Y. Suga, N. Kosugi, and M. Morimoto, "Real-time background music monitoring based on content-based retrieval," *ACM International Conference on Multimedia*, 2004.

[4] K. Kashino, A. Kimura, H. Nagano, and T. Kurozumi, "Robust Search Methods for Music Signals Based on Simple Representation," *ICASSP*, 2007.

[5] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Popular, Classical, and Jazz Music Databases," *ISMIR*, 2002.

[6] R. Thom, "A few notes on the music in the final mix," May 2009. http://www.filmsound.org/randythom/finalmix.htm.

[7] C. Duxbury, J. P. Bello, M. Davies, and M. Sandler: "Complex domain onset detection for musical signals," *International Conference on Digital Audio Effects*, 2003.

[8] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," *ISMIR*, 2002.

[9] S. Baluja and M. Covell, "Audio fingerprinting: combing computer vision & data stream processing," *ICASSP*, 2007.

[10] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang, "Finding interesting associations without support pruning," *ICDE*, 2000.

[11] R. Radhakrishnan, C. Bauer, C. Cheng, and K. Terry, "Audio signature extraction based on projections of spectrograms," *ICME*, 2007.