A METHOD FOR EXTRACTING A MUSICAL UNIT TO PHRASE MUSIC DATA IN THE COMPRESSED DOMAIN OF TWINVQ AUDIO COMPRESSION

Motohiro Nakanishi, Michihiro Kobayakawa, Mamoru Hoshi, Tadashi Ohmori

Graduate School of Informantion Systems, University of Electro-Communications.
kobayakawa@computer.org

ABSTRACT

A method for phrasing music data into meaningful musical pieces (*e.g.*, bar and phrase) is an important function to analyze music data. To realize this function, we propose a method for extracting a unit of music data (musical unit) *in the compressed domain of TwinVQ audio compression (MPEG-4 audio)*.

Our key idea is to extract a musical unit from a sequence of autocorrelation coefficients computed *in the encoding step of TwinVQ audio compression*. We call the sequence of the autocorrelation coefficients the "autocorrelation sequence $\boldsymbol{r}$". We use the k-th autocorrelation sequence $\boldsymbol{r}_k$ ($k = 1, 2, \ldots, 20$) of music data for extracting a musical unit of music data.

First, we calculate the j_k-th autocorrelation coefficient $a_k^{j_k}$ of the k-th autocorrelation sequence $\boldsymbol{r}_k$ ($j_k = 38, 39, \ldots, 208$; $k = 1, 2, \ldots, 20$).

Second, for detecting the peak in the sequence ($a_k^{38}, a_k^{39}, \ldots, a_k^{208}$), the Laplacian filter is applied to the sequence. We then obtain the order p_k for which the maximum differential coefficient is attained.

Finally, we compute the musical unit using p_k.

To evaluate the performance of extracting the musical unit by our method, we collected 64 music data and obtained autocorrelation sequences by applying the TwinVQ encoder to each data. We then applied our extraction algorithm to each autocorrelation sequence.

The experimental results reveal a very good performance in the extraction of a musical unit for phrasing music data.

1. INTRODUCTION

Music distribution systems on the Internet have recently gained popularity. By using these systems, we can easily collect (buy) and listen to a considerable amount of compressed music data. To effectively retrieve music data from music databases, music retrieval systems need to be capable of handling various types of queries. Among the conventional music database systems, keyword-based music retrieval systems are widely used. In addition, we require content-based music retrieval systems for handling various types of queries. In a content-based music retrieval system, various types of musical contents are expressed by various music features. It is becoming increasingly important to develop a method for extracting a music feature in a content-based music retrieval system.

We consider the methods for extracting music features in the compressed domain of TwinVQ audio compression. We have already shown that the i-th autocorrelation coefficient with bit rate B_1 computed in the encoding step of TwinVQ audio compression approximates the j-th autocorrelation coefficient with bit rate B_2, where $i = \left\lfloor \frac{B_1}{B_2} j \right\rfloor$ [1], and have proposed a method for extracting a period of beat from the sequence of autocorrelation coefficients [2].

In this paper, to analyze the music structure of music data in the compressed domain of TwinVQ audio compression, we propose a method for extracting a musical unit to phrase music data into the meaningful musical pieces (*e.g.*, bar, half-bar, double-bar, phrase and passage).

Our key idea is to extract a musical unit from the autocorrelation sequence computed in the encoding step of TwinVQ audio compression (MPEG-4 audio).

This paper is organized as follows. Section 2 briefly describes some related works on both music information retrieval and music retrieval. Section 3 presents the autocorrelation coefficients computed in the encoding step of TwinVQ. Section 4 describes our method for extracting the length of bar of a piece of music from the sequences of autocorrelation coefficients. Section 5 describes the experiment for evaluating the performance of extracting the length of bar, the results of which are discussed in Section 6. Section 7 summarizes this paper.

2. RELATED WORKS

This section briefly shows some previous works on music information retrieval and music retrieval in the compressed domain of MPEG-1 audio (Layer 1, Layer 2 and Layer 3) standard.

For encoded music data, Liu *et al.* proposed a method for retrieving an MP3 music object by using hummed queries [3]. They used the (F_s, F_t)-subband energies computed from the encoded data as an index feature vector. From the hummed queries, they obtained the (F_s, F_t)-subband energies in the encoding step as a query feature vector. Further, they proposed a method for classifying the encoded data to identify a singer. They used the energy of a frame for classification [4].

Wang *et al.* proposed a method for extracting a beat of a piece of music from the MP3 bitstreams. They used the subband energy computed from the bitstream [5].

Pye proposed a method for managing music data [6]. They classified the MP3 music data into genres and labeled them artist name using cepstrum.

Though several methods have been proposed for music information retrieval and music retrieval in the compressed domain of MPEG-1, the fields of music information retrieval and music retrieval in the compressed domain are still in their infancy, and such methods are gaining increasing importance.

0-7803-9332-5/05/$20.00 ©2005 IEEE

3. AUTOCORRELATION COEFFICIENTS COMPUTED IN THE ENCODING STEP OF TWINVQ AUDIO COMPRESSION

This section briefly describes the autocorrelation coefficients computed in the encoding step of TwinVQ audio compression. The details of TwinVQ audio compression are referred to [7, 8, 9] or at the URL http://www.twinvq.org/. A sequence of original signals of a piece of music is divided into frames. The MDCT (Modified Discrete Cosine Transform) is then applied to the signals of each frame. For the MDCT coefficients of each frame, five analyses, namely, LPC analysis, Pitch analysis, Barkscale envelope analysis, Power analysis and Weighted VQ analysis, are carried out in this order. The autocorrelation coefficients are computed in the LPC analysis.

The autocorrelation coefficients are calculated from the power spectrums of the MDCT coefficients $(f_0, f_1, \ldots, f_{N-1})$. The power spectrums are given by $(F_0, F_1, \ldots, F_{N-1}) \equiv ((f_0)^2, (f_1)^2, \ldots, (f_{N-1})^2)$.

The k-th autocorrelation coefficient R_k is defined as follows:

$$\begin{aligned} R_k &= \sum_{i=0}^{N-1} F_i \cos(k\theta) \qquad\qquad (0 \le \theta < 2\pi) \\ &= \sum_{i=0}^{N-1} F_i \cos\left(k \cdot \frac{2\pi i}{N}\right). \end{aligned}$$

The k-th autocorrelation coefficient R_k is normalized by R_0 as $r_k = R_k/R_0$.

Let S be a sampling rate and let B be a bit rate for compressing signals. To compress signals of a piece of music with bit rate B, the encoder computes the MDCT coefficients $(f_0, f_1, \ldots, f_{N-1})$ and uses up to $(M^B - 1)$-th MDCT coefficients, where

$$M^B = \left\lfloor \min\left(\frac{B}{S}, 1\right) \times N \right\rfloor; \tag{1}$$

$\lfloor x \rfloor$ is the greatest integer less than or equal to x.

Then, the encoder generates a set of MDCT coefficients $(f_0^B, f_1^B, \ldots, f_{N-1}^B)$ given by

$$f_a^B = \begin{cases} f_m & (a = \lfloor (N/M^B)m \rfloor,\ m = 0, \ldots, M^B - 1) \\ 0 & (otherwise). \end{cases} \tag{2}$$

The k-th autocorrelation coefficient with bit rate B R_k^B is defined as follows:

$$\begin{aligned} R_k^B &= \sum_{a=0}^{N-1} F_a^B \cos(k\theta) \qquad\qquad (0 \le \theta < 2\pi) \\ &= \sum_{a=0}^{N-1} F_a^B \cos\left(k \cdot \frac{2\pi a}{N}\right), \end{aligned}$$

where $F_a^B = (f_a^B)^2$.

The k-th autocorrelation coefficient with bit rate B R_k^B is normalized by R_0^B as $r_k^B = R_k^B/R_0^B$.

We use the k-th autocorrelation coefficient with bit rate B r_k^B for extracting a musical unit of music data. Hereafter, the k-th autocorrelation coefficient with bit rate B r_k^B is simply denoted by r_k.

Procedure extract_musical_unit($\boldsymbol{R}$, S, f, l)

input $\boldsymbol{R}$: set of the autocorrelation sequences $\boldsymbol{r}_k$ $(k = 1, 2, \ldots, 20)$
S: sampling rate
f: number of samples of a frame

output l: musical unit

$f_t = \frac{f}{S}$: length of a frame

$T_{min} = 50 \quad (bpm)$: minimum tempo

$T_{max} = 200 \quad (bpm)$: maximum tempo

$l_{min} = 0.9 \quad (= 3 \cdot \frac{60}{T_{max}})$: minimum length of bar

$l_{max} = 4.8 \quad (= 4 \cdot \frac{60}{T_{min}})$: maximum length of bar

$s_{min} = \lfloor 0.9 \cdot f_t \rfloor$, $s_{max} = \lceil 4.8 \cdot f_t \rceil$

// Step 1 //

for $k \leftarrow 1$ **to** 20 **do**

Calculate the j_k-th autocorrelation coefficients of the k-th autocorrelation sequence $\boldsymbol{r}_k$ $(j_k = s_{min}, \ldots, s_{max})$.

// Step 2 //

for $k \leftarrow 1$ **to** 20 **do**

Apply the Laplacian filter to the sequence $(a_k^{s_{min}}, \ldots, a_k^{s_{max}})$ and obtain the differential coefficients $d_k^{j_k}$.

Find $d_k^{p_k}$ for which $\max_{j_k=s_{min}}^{s_{max}} d_k^{j_k}$ is attained.

// Step 3 //

Obtain the order p_i for which $\max_{k=1}^{20} d_k^{p_k}$ is attained.

Calculate the musical unit $l = p_i \cdot f_t$.

end procedure

Fig. 1. Procedure of extracting a musical unit.

4. METHOD FOR EXTRACTION

This section presents an algorithm for extracting a musical unit of music data from the sequence of the k-th autocorrelation coefficients.

Suppose that the music data consists of N frames. We obtain the k-th autocorrelation coefficients $(k = 1, 2, \ldots, 20)$ for each frame. The k-th autocorrelation coefficient of the n-th frame is denoted by $r_{k,n}$. Now, we have the k-th autocorrelation sequence $\boldsymbol{r}_k \equiv \{r_{k,1}, \ldots, r_{k,N}\}$ $(k = 1, 2, \ldots, 20)$, we define the j_k-th autocorrelation coefficient $a_k^{j_k}$ of the k-th autocorrelation sequence $\boldsymbol{r}_k$ as

$$a_k^{j_k} = \frac{\sum_{i=1}^{N-j_k} (r_{k,i} - \bar{r}_{k,1})(r_{k,i+j_k} - \bar{r}_{k,j_k})}{(N - j_k - 1)\sigma_{k,1}\sigma_{k,j_k}},$$

where

$$\bar{r}_{k,j_k} = \frac{1}{N - j_k} \sum_{i=j_k}^{N-j_k} r_{k,i},$$

$$\sigma_{k,j_k} = \sqrt{\frac{1}{N-j_k}\sum_{i=j_k}^{N}(r_{k,i}-\bar{r}_{k,j_k})^2}.$$

We can determine the minimum and maximum values of j_k by the following two assumptions based on common musical knowledge: 1) musical time of almost all pieces of music is triple or quadruple time, 2) the tempo of almost all pieces of music is between $T_{min} = 50(bpm)$ and $T_{max} = 200(bpm)$.

From the above assumptions, we can define the common range for the length of bar $\boldsymbol{l} = [l_{min}, l_{max}]$: $[0.9(= \frac{60(sec)}{200(bpm)} \times 3), 4.8(= \frac{60(sec)}{50(bpm)} \times 4)]$.

Thus, we obtain $s_{min} = \lfloor 0.9 \cdot f_t \rfloor$ and $s_{max} = \lceil 4.8 \cdot f_t \rceil$ as the minimum and maximum values of j_k, respectively, where $\lceil x \rceil$ is the least integer greater than or equal to x.

For extracting a musical unit of music data, we execute the following steps: **Step 1** $\sim$ **Step 3**. The procedure for extraction is shown in Fig. 1.

Step 1 [Calculate the autocorrelation coefficients]: For the k-th autocorrelation sequence ($k = 1, \ldots, 20$), calculate the j_k-th autocorrelation coefficients $a_k^{j_k}$ ($j_k = s_{min}, \ldots, s_{max}$).

Step 2 [Detect the peaks]: Apply the Laplacian filter to the sequence $(a_k^{s_{min}}, \ldots, a_k^{s_{max}})$ and obtain the differential coefficients $d_k^{j_k}$ ($k = 1, \ldots, 20$); then, find the p_k-th differential coefficients $d_k^{p_k}$ for which $\max_{j_k=s_{min}}^{s_{max}} d_k^{j_k}$ is attained.

Step 3 [Determine the musical unit]: From the peaks found in **Step 2**, obtain the order p_i for which $\max_{k=1}^{20} d_k^{p_k}$ is attained; then, calculate the musical unit $l = p_i \cdot f_t$.

5. EXPERIMENT

We collected 64 pieces of music (Japanese POP and Rock) and tested our method on each piece of music. To obtain the autocorrelation sequences for each piece of music , we ripped music CD tracks to the sequences of original signals (with a sampling rate of $44.1\ kHz/ch$) and then applied the TwinVQ encoder (MPEG-4 audio Reference Software, Verification Model ver 1.2) with a bit rate of $44\ kbps$ to each sequence of original signals. We then obtained the set of autocorrelation sequences for each piece of music.

We obtained the musical unit by applying our algorithm to the set of autocorrelation sequences for each piece of music . We call the extracted musical unit the "musical unit by our method" and denote it by "O."

To evaluate the performance of extractions by our method, we obtained two length of bar by the following two methods:

1. We obtained the length of bar from the sequence of original signals by using the wave viewer. We call it the "length of bar by hand" and denote it by "H."
2. We obtained the length of bar from the metronomic identification in a score. We call it the "length of bar by score" and denote it by "S."

Table 1 summarizes the results. In Table 1, R_{oh} and R_{os} are the ratios $\frac{O}{H}$ and $\frac{O}{S}$, respectively. The ratios R_{oh} and R_{os} show the performance of extraction. Hereafter, we use the ratio R_{oh} to evaluate the performance of extraction by our method because there is little difference between the length of bar by hand, H, and that by score, S.

When the ratio R_{oa} is 1.0, 0.5 or 2.0, we say that our method has succeeded in extracting the length of bar, length of half-bar or length of double-bar as the musical unit, respectively.

6. DISCUSSION

In this section, we evaluate the performance of extractions of a music unit by our method. The evaluation is not straightforward because listeners may phrase a piece of music into bar, double-bar, or passage. Thus, not only the length of bar, but also the length of half-bar or length of double-bar are used as a fundamental unit for the music structure analysis. That is, when the ratio R_{oh} is $[0.485, 0.515]$, $[0.97, 1.03]$, or $[1.94, 2.06]$ (with an accuracy of ± 3 percent), we say that our method has succeeded in extraction of length of half-bar, length of bar or length of double-bar, respectively. We call such an extraction the "successful extraction."

We count the number of successful extractions (called the "number of successes") and compute the success rate defined by $\frac{\#\text{successes}}{64}$. The success rate is an indicator of the global performance of our method.

Table 1 reveals 61 successful extractions. The success rate obtained is $0.953(= \frac{61}{64})$, *i. e.,* 95.3 percent of the extracted units by our method are meaningful musical lengths. Out of the 61 successful extractions, we obtained length of bar of 44, length of half-bar of 3, and length of double-bar of 14.

Out of the 3 unsuccessful extractions, we obtained 0.248 as the ratio R_{oh} of music data of music ID 6, *i. e.,* length of quarter-bar. Further, we obtained 1.938 as the ratio R_{oh} of music data of music ID 23, *i. e.,* length of double-bar with an accuracy of ± 4 percent.

Thus, our method for extracting a musical unit of music data shows a very good performance in extracting a musical unit.

7. CONCLUSION

We proposed a method for extracting a musical unit of music data in the compressed domain of TwinVQ audio compression (MPEG-4 audio) .

Our key idea was to extract a musical unit of music data from the sequence of the autocorrelation coefficients computed *in the encoding step of TwinVQ audio compression*.

To extract a musical unit, we calculated the autocorrelation coefficients of autocorrelation sequences, detected the peak in the autocorrelation coefficients, and then determined the musical unit of music data.

We tested our method on 64 music data and evaluated performance of the extractions. The experimental results show a very good performance in extracting a musical unit for phrasing music data into musically meaningful pieces such as half-bar, bar, and double-bar.

Our contribution to the content-analysis of a piece of music is to realize a method for extracting a musical unit to phrase music data into meaningful musical pieces in the compressed domain of TwinVQ audio compression.

Our method has many applications such as music structure analysis, music retrieval by short pieces of music and refrain detection.

Table 1. Results of the extractions.

music ID	length of bar (sec)			R_{oh}	R_{os}	music ID	length of bar (sec)			R_{oh}	R_{os}
	H	S	O				H	S	O		
1	1.738	1.740	1.742	1.002	1.001	33	1.684	1.792	1.695	1.007	0.946
2	2.500	2.500	2.508	1.003	1.003	34	1.608	1.580	1.579	0.982	0.999
3	2.055	2.052	2.043	0.994	0.996	35	2.470	2.448	2.461	0.996	1.005
4	2.336	2.332	2.322	0.994	0.996	36	1.964	1.952	3.878	1.974	1.987
5	2.400	2.376	4.760	1.983	2.003	37	2.222	2.224	2.229	1.003	1.002
6	4.796	4.000	1.184	0.248	0.296	38	1.892	1.920	1.858	0.982	0.968
7	2.180	2.164	2.160	0.991	0.998	39	1.792	1.776	1.788	0.998	1.007
8	1.908	1.920	3.808	1.996	1.983	40	2.132	2.332	4.249	1.993	1.822
9	1.756	1.740	1.742	0.992	1.001	41	2.032	2.032	4.064	2.000	2.000
10	2.824	2.792	2.832	1.003	1.014	42	1.972	1.968	1.974	1.001	1.003
11	2.108	2.088	4.180	1.983	2.002	43	1.320	1.320	1.324	1.003	1.003
12	1.968	1.968	1.974	1.003	1.003	44	1.892	1.920	0.952	0.503	0.495
13	2.444	2.424	2.461	1.007	1.015	45	1.328	1.320	3.297	2.482	2.498
14	2.408	2.144	4.714	1.958	2.199	46	1.848	1.848	1.858	1.005	1.005
15	2.172	2.164	2.160	0.994	0.998	47	2.312	2.308	4.621	1.999	2.002
16	2.428	2.424	1.207	0.497	0.498	48	1.380	1.372	1.370	0.993	0.999
17	2.696	2.696	2.694	0.999	0.999	49	2.072	2.068	2.090	1.009	1.011
18	1.948	1.968	3.901	2.003	1.982	50	1.596	1.480	3.204	2.008	2.165
19	2.476	2.476	2.461	0.994	0.994	51	2.065	2.068	2.067	1.001	0.999
20	1.952	1.968	3.901	1.998	1.982	52	2.552	2.552	2.554	1.001	1.001
21	2.264	2.264	2.276	1.005	1.005	53	2.136	2.144	2.136	1.000	0.996
22	3.015	3.000	2.996	0.995	0.999	54	2.464	2.448	2.438	0.989	0.996
23	2.120	2.052	4.110	1.939	2.003	55	2.523	2.528	2.531	1.003	1.001
24	1.852	1.820	3.646	1.968	2.003	56	2.564	2.580	1.300	0.507	0.504
25	1.995	2.016	1.997	1.001	0.991	57	3.243	3.244	3.251	1.002	1.002
26	2.320	2.284	4.528	1.952	1.982	58	2.582	2.580	2.577	0.998	0.999
27	3.080	3.076	1.486	0.483	0.483	59	2.608	2.608	2.601	0.997	0.997
28	1.281	1.304	1.277	0.997	0.979	60	2.529	2.528	2.531	1.001	1.001
29	1.809	1.804	1.811	1.001	1.004	61	3.629	3.380	3.646	1.005	1.079
30	2.394	2.376	2.392	0.999	1.007	62	3.817	3.808	3.808	0.998	1.000
31	1.740	1.820	1.742	1.001	0.957	63	1.902	1.894	1.904	1.002	1.004
32	2.434	2.492	2.438	1.002	0.978	64	1.459	1.417	1.439	0.986	1.015

8. REFERENCES

[1] Kensuke Onishi, Michihiro Kobayakawa, Mamoru Hoshi, and Tadashi Ohmori, "A Feature Independent of Bit Rate for TwinVQ Audio Retrieval," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME2001)*, 2001, pp. 409–416.

[2] Michihiro Kobayakawa, Takashi Okunaru, Kensuke Onishi, and Mamoru Hoshi, "A New Method for Extracting a Period of Beat of Music in Compressed domain of TwinVQ Audio Compression," in *Proceedings of the Fourth IEEE Pacific-Rim Conference on Multimedia (PCM2003)*, 2003, 3A1.2.

[3] Chih-Chin Liu and Po-Hun Tsai, "A Singer Identification Technique for Content-based Classification of MP3 Music Objects," in *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM2001)*, 2001, pp. 438–445.

[4] Chih-Chin Liu and Po-Hun Tsai, "Content-based Retrieval of MP3 Music Objects," in *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM2002)*, 2002, pp. 506–511.

[5] Ye Wang and Miika Vilermo, "A Compressed Domain Beat Detector Using MP3 Audio Bitstreams," in *Proceedings of the Ninth ACM International Conference on Multimedia (MM2001)*, 2001, pp. 194–202.

[6] David Pye, "Content-based Methods for Management of Digital Music," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2000)*, 2000, pp. 2437–2440.

[7] ISO/IEC JTC 1/SC 29/WG11 N2203, "Working Draft of ISO/IEC CD 144963," May 1998.

[8] N Iwakami, T Moriya, and S Miki, "High-quality audio-coding at Less Than 64/kbit/s Using Transform Domain Weighted Interleave Vector Quantization," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP1995)*, 1995, pp. 3095–3098.

[9] T Moriya, N Iwakami, K Ikeda, and S Miki, "Extension and Complexity of TwinVQ Audio Coder," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP1996)*, 1996, pp. 1029–1032.