Vol. 48 No. 3 IPSJ Journal Mar. 2007

Regular Paper

Drumix: An Audio Player with Real-time Drum-part Rearrangement Functions for Active Music Listening

Kazuyoshi Yoshii,† Masataka Goto,†† Kazunori Komatani,† Tetsuya Ogata† and Hiroshi G. Okuno†

This paper presents a highly functional audio player, called *Drumix*, that allows a listener to control the volume, timbre, and rhythmic patterns (drum patterns) of bass and snare drums within *existing audio recordings* in real time. A demand for *active music listening* has recently emerged. If the drum parts of popular songs could be manipulated, listeners could have new musical experiences by freely changing their impressions of the pieces (e.g., making the drum performance more energetic) instead of passively listening to them. To achieve this, Drumix provides three functions for rearranging drum parts, i.e., a *volume control function* that enables users to cut or boost the volume of each drum with a drum-specific volume slider, a *timbre change function* that allows them to replace the original timbre of each drum with another selected from a drop-down list, and a *drum-pattern editing function* that enables them to edit repetitive patterns of drum onsets on a graphical representation of their scores. Special musical skills are not required to use these functions. Subjective experiments revealed that Drumix could add a new dimension to the way listeners experience music.

1. Introduction

The development of computer-aided music-playback interfaces has recently been facilitated because of the great demand for *active music listening* that has emerged[1)]. For example, customers who use listening stations in music stores for quick trials often want to jump directly to the chorus of popular songs. To satisfy this need, Goto[2)] proposed a new listening station, called SmartMusicKIOSK, which provides a "jump to chorus" button and a "jump to next section" button[☆]. This is an active listening environment in which listeners can have new listening experiences by easily skipping sections that hold no interest. In another scenario, music listeners who access numerous songs with flat-rate services (e.g., Last.fm[3)] and Pandora[4)]) often want to listen to one "something else" after another according to their preferences. To meet this demand, M. and T. Goto[5)] proposed a music-playback interface, called Musicream, which makes playlists highly customizable; listeners who come across unexpected songs can quickly and easily arrange playlists that suit their momentary feelings (e.g., by arranging playback order, deleting non-favorites, and inserting new songs).

These interfaces are achieved by automatically analyzing musical contents in polyphonic audio signals such as commercial CD recordings. SmartMusicKIOSK is based on a signal-processing method of detecting the chorus, called RefraiD[6)], which captures the musical structures of these complex audio signals. The acoustic features are extracted from audio signals as musical content with Musicream. To add new songs to a playlist, the content similarity is calculated between several songs in the playlist and other songs in the database. Pachet and Delerue[7)] proposed an active listening environment, called MidiSpace, in which listeners can change the spatialization and localization of individual instruments. However, MidiSpace is implemented with MIDI sound modules because the instrument parts of polyphonic audio signals are quite difficult to separate.

The demand for active music listening remains unsatisfied because of the difficulty of dealing with audio signals in real-world music. For example, when focusing on a melody line (e.g., a sung melody), the volume of accompanying instruments such as drums should be attenuated. Because the characteristics of drum performances (e.g., volume, timbre, and repetitive patterns) are important factors that in-

† Graduate School of Informatics, Kyoto University
†† National Institute of Advanced Industrial Science and Technology (AIST)

☆ For example, using this button on a song structure like "intro ⇒ (verse A ⇒ verse B ⇒ chorus) × 2 ⇒ chorus" would enable a listener to jump to the beginnings of verse and chorus sections.

fluence the moods of popular songs, listeners may be interested in the creative experience of freely rearranging the drum parts to suit their moods. Unfortunately, ordinary graphic equalizers cannot be used for such purposes because they cannot control the frequency components of *specific instruments*; they can only be used to control those of *specific frequency bands*.

To meet these demands, we propose a highly functional audio player, called *Drumix*, which supports the real-time rearrangement of drum parts within existing audio recordings by providing the following three functions:

- *Volume control* allows users to intuitively cut or boost the volume of bass and snare drums with drum-specific volume sliders.
- *Timbre change* enables users to replace the original timbre of each drum with another timbre selected from a drop-down list.
- *Drum-pattern editing* helps users to freely edit the repetitive patterns of each drum on a score-like interface.

By using these functions, users can enjoy music in a much more active manner.

To implement Drumix, it is necessary to *automatically analyze musical contents* of polyphonic audio signals, i.e., (1) estimate the frequency components of bass and snare drum sounds used in a target musical piece, (2) detect the onset times of each drum sound, and (3) estimate the times of bar lines. To do so, we used computational methods, i.e., our drum sound recognition method, called *AdaMast* 8), and the beat-tracking method proposed by Goto 9). The former comprises sequential template-adaptation and template-matching stages using drum-sound spectrograms as templates. These two stages can accomplish (1) and (2), respectively. The latter accomplishes (3).

The rest of this paper is organized as follows. Section 2 presents the user interfaces for the three functions. Section 3 explains the internal architectures of these functions, and Section 4 describes two methods of analyzing musical contents. Section 5 reports subjective experiments, and Section 6 discusses the significance of this work. Finally, Section 7 summarizes the key points.

2. User Interfaces for Real-time Drum-part Rearrangement

Drumix not only has basic functions for playback control that ordinary audio players are equipped with (e.g., play, pause, and stop) but also three novel ones for drum-part rearrangement: *volume control*, *timbre change*, *and drum-pattern editing*, which operate cooperatively instead of independently. Because these functions can be used even if an audio file is being played back, users can enjoy music while rearranging drum parts in real time. In other words, every time various operations are done for drum-part rearrangement, users can immediately compare the expected effects with the actual effects (e.g., a change in the song's mood) of these operations.

Drumix enables end-users to appreciate music in a new environment where they play dual roles; they behave not merely as listeners but also as production-related people. In a typical process of music production, *arrangers* select appropriate timbres for drums and compose drum patterns that suit the melody, chords, and total arrangement. *Mixing engineers* then integrate the multiple audio tracks of individual instrument parts including the melody and drum tracks into a single stereo track with the appropriate mixing balance. This process is called mix-down. Using Drumix, users can easily arrange and mix-down drum parts.

2.1 Volume Control Function

With this function, users can freely adjust the mixing balance of bass and snare drum sounds in audio recordings as if they were mixing engineers. The function enables them to intuitively cut or boost the volume of bass and snare drums with drum-specific volume sliders. The volume of each drum can be boosted by moving the corresponding slider from the left (muted) to the right (maximum volume) and vice versa, as shown in **Fig. 1**.

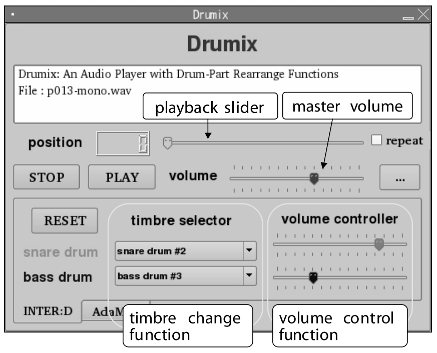


Fig. 1 Screenshot of main window: basic functions of playback control in the upper window and volume-control and timbre-change functions in the lower window.

2.2 Timbre Change Function

During playback, users can change the timbres of bass or snare drums as if they were arrangers. As shown in Fig. 1, the original timbre of each drum can be replaced with an arbitrary one chosen from various kinds of timbres listed in the corresponding drop-down list. Several timbres, which are actually PCM audio files of solo tones, are registered in the list beforehand, and users can add new ones to the list.

2.3 Drum-pattern Editing Function

With this function, users can compose drum patterns, which is one of the arranger's most important tasks. In this paper, a set of onset times for bass and snare drums in a measure[☆] is called a drum pattern. The requirements for using this function are as follows:

- *Requirement 1:* The time signature of the target musical piece is 4/4. That is, the time length of a measure is equal to that of four quarter notes.
- *Requirement 2:* If the time signature of the original piece is 2/4, the time signature can be considered to be 4/4 by concatenating two successive measures.
- *Requirement 3:* The tempo is roughly constant and is between 61 M.M.[☆☆] and 185 M.M.

These requirements are not critical in practice because most popular songs will satisfy them.

2.3.1 Drum-pattern Editor

The onset times for the drums in a drum pattern can be freely located by using a drum-pattern editor as shown in **Fig. 2**. This editor provides a graphical representation of drum scores in which the onset times and volumes of the drums are directly visualized. Each of the four large rectangular areas in Fig. 2 corresponds to a measure. The lower and upper sides of each measure area represent the respective scores of bass and snare drums, and the vertical and horizontal axes represent the respective volume and time.

Each vertical bar in the measure areas corresponds to an onset (note), and its height represents the volume. The vertical line, which indicates the playback position, progresses through the four measure areas (upper-left area ⇒ upper-right area ⇒ lower-left area ⇒ lower-right area ⇒ upper-left area ⇒ ⋯). In the initial state, the original drum pattern is displayed in each measure area, which is updated every time the playback position enters that area.

☆ A "measure" is often called a "bar," which is the duration between successive bar lines.

☆☆ Mälzel's Metronome: the number of quarter notes per minute.

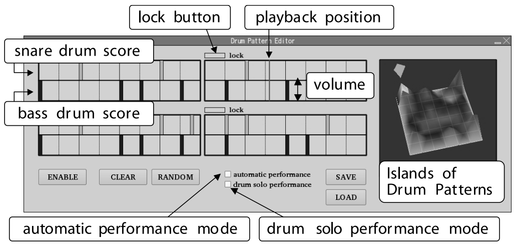


Fig. 2 Screenshot of drum-pattern editor.

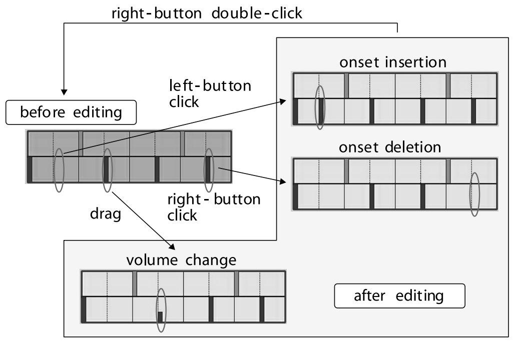


Fig. 3 Manipulation of drum scores with mouse.

2.3.2 Original-pattern-based Editing

Users can compose new drum patterns on the basis of the originals displayed in the initial state. To do this, the following mouse operations have been provided (see also **Fig. 3**):

- *Arrangement of onset times*
  Clicking on the left (right) button of the mouse at an arbitrary time position in one of the four measure areas inserts (deletes) an onset at that position.
- *Change of onset volumes*
  Dragging the mouse up and down at the top of an arbitrary onset bar changes the volume of that onset.
- *Restoration of original drum patterns*
  Double-clicking on the right button of the mouse in an arbitrary measure area resets a modified drum pattern to the original in that area.

Once a user modifies the original drum pattern displayed in the measure area, the modified pattern is not updated by subsequent original patterns appearing in that area. The audio file that is played back is one in which the original drum patterns at intervals of three measures are replaced with modified drum patterns.

2.3.3 Template-pattern-based Editing

A major problem in using this function is that

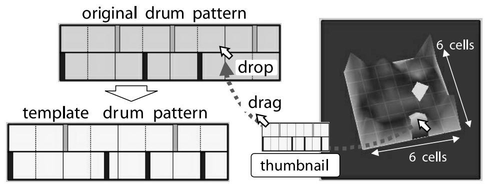


Fig. 4 Referring to template drum pattern: dragging the mouse from the Islands of Drum Patterns.

users who have little musical knowledge have difficulty in editing drum patterns. This is because some expertise in musical composition or performance is required to compose new drum patterns without references or from scratch.

To solve this problem, we developed a novel interface on which drum patterns collected from other musical pieces in advance can be referred to as templates. Users can edit drum patterns merely by searching for their favorite drum patterns in these templates. If some of the onset location details in a template do not suit a user, he or she can easily modify them.

The 3D representation at the right of Fig. 2 is called "Islands of Drum Patterns," which represents the distribution of template drum patterns. Each of them is categorized into one of 36 (6×6) square cells according to the similarity of onset locations. The height of a cell corresponds to the number of drum patterns allocated to that cell; the more drum patterns that are allocated, the higher the cell is.

Users can place a template on an arbitrary measure area by dragging the mouse from a specified cell to that area, as shown in **Fig. 4**. During dragging, a thumbnail of the template is attached to the mouse pointer. The dragged template is randomly selected from similar drum patterns allocated to the same cell. Users can promptly learn what patterns are clustered in each cell by dragging the mouse from that cell several times. This enables users to encounter many drum patterns that might suit their preferences because similar patterns with fine differences in their details can be easily generated.

2.4 Two Playback Modes

Two playback modes have been provided to support different ways of listening to music. Both modes can be enabled or disabled by using the corresponding checkbox shown in Fig. 2.

- *Automatic performance mode*
  Users can play back music with a drum pattern randomly selected from the template drum patterns every time the playback position enters a new measure. This mode frees users from the labor of editing drum patterns. If a favorite drum pattern appears in a measure area, they can fix it by clicking on a "lock" button.
- *Drum solo performance mode*
  Users can play back only the drum tracks by muting the sounds of other musical instruments. This mode helps them to concentrate on rearranging the drum parts (e.g., carefully selecting drum timbres).

3. Internal Architectures

This section explains the internal architectures of the drum-part rearrangement functions and improvements to sound quality. The procedures described in this section are performed *in real time* under the assumption that the following three musical contents have already been obtained *in advance* from the target polyphonic audio signal, as described in Section 4:

- Spectrograms of isolated tones of bass and snare drums.
- Onset times for each drum sound.
- Bar-line times.

The bar-line times are only required for the drum-pattern editing function.

Here, let P_X ($X = BD$ or SD) denote a drum-sound spectrogram where X represents a bass or snare drum used in the target piece. $P_X(t, f)$ ($1 \leq t \leq T$ [frames], $1 \leq f \leq F$ [bins]) represents a local component in the time-frequency domain, where T and F are fixed values.

3.1 Volume Control Function

This function amplifies or attenuates frequency components of drum X without affecting those of other musical instruments. The algorithm is as follows (also see **Fig. 5**):

(1) Starting from each onset time, a spectrogram segment P_i ($i = 1, \cdots, N_D$), which has the same time length as spectrogram P_X, is extracted from the audio signal of the target piece, where N_D is the number of detected onsets of drum X.

(2) Spectrogram P_X is added to each spectrogram segment P_i after being weighted by an arbitrary amplitude change ratio r (cutting: $-1 \leq r < 0$, boosting: $0 < r$):
$$P'_i(t, f) = P_i(t, f) + r \cdot P_X(t, f), \quad (1)$$
where P'_i is a processed spectrogram segment with a phase that is the same as that of the original segment.

(3) A processed spectrogram of the target piece in which the volume of drum X is changed is obtained by replacing P_i with

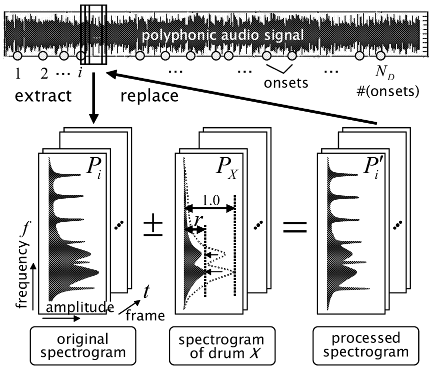


Fig. 5 Cutting/boosting volume of drum sounds.

P'_i.

(4) A processed audio signal of the target piece is obtained by applying overlap-add synthesis to the processed spectrogram of that piece.

To obtain spectrograms of input audio signals sampled at 44.1 [kHz], we used Short-time Fourier Transform (STFT) with a Hanning window (4,096 points) with a shifting interval of 441 points. T and F were empirically set to 10 [frames] and 2,048 [bins].

3.2 Timbre Change Function

This function changes the timbre of drum X by muting its sounds that are included in a polyphonic audio signal and then adding another drum sound (let Y be that drum sound) to the signal. The algorithm is as follows:

(1) A processed audio signal of the target piece in which the sounds of drum X are muted is obtained by using the volume control function with the amplitude change ratio r set to -1.0.

(2) The audio signal of drum Y (a solo tone) is added to the audio signal of that piece at each onset time of drum X.

3.3 Drum-pattern Editing Function

This function replaces the original drum pattern with one composed by users or selected from the Islands of Drum Patterns. We first explain the method used to create the Islands of Drum Patterns. This method should be used in advance. We then describe the architecture for this function.

3.3.1 Drum-pattern Clustering

To create the Islands of Drum Patterns, we use the method proposed by Pampalk, et al.[10], which visualizes the distribution of musical pieces as "Islands of Music." Using a self-organizing map (SOM), this method is firstly used to locate musical pieces on a 2-dimensional plane according to the similarity of acoustic features. It is then used to paint a map of the musical pieces so that locations where many pieces are clustered look like islands.

To apply Pampalk's method, we represent a drum pattern as a 96-dimensional vector by splitting a measure into 48 temporal segments☆; we concatenate two 48-dimensional vectors, which correspond to bass and snare drum scores. Each element in these vectors is either 0 or 1, depending on the absence or presence of an onset in the corresponding temporal segment. The algorithm for creating the Islands of Drum Patterns is as follows:

(1) Various drum patterns, which are used as templates, are collected from MIDI files of popular songs.

(2) These drum patterns, which are represented as 96-dimensional vectors, are input to an SOM having 36 (6×6) cells.

(3) The map is visualized in a 3-dimensional space in which cells where many drum patterns are clustered look like islands.

3.3.2 Drum-pattern Replacement

Recall our assumption that bar-line times were estimated. Therefore, drum-sound onset sequences can be split into drum patterns, which can be displayed in the drum-pattern editor. In addition, composed drum patterns can be synchronized with appropriate time durations. The algorithm for replacing the original drum pattern with a composed one is as follows:

(1) A processed audio signal of the target piece in which bass and snare drum sounds are muted is obtained by using the volume control function.

(2) Each drum sound that has a favorite timbre selected from the drop-down list is added to the new audio signal according to the composed drum score.

3.4 Improved Sound Quality

We propose two methods for generating higher-quality audio signals because there is much room for improving the sound quality.

3.4.1 Dynamic Amplitude Adjustment

The spectrogram P_X actually includes small local frequency components of other sounds (i.e., P_X cannot be the same as the precise spectrogram of an isolated tone) because it is estimated from a complex polyphonic spectrogram. Therefore, if we only use P_X to rearrange drum

☆ This quantization is commonly used in the field of desktop music (DTM).

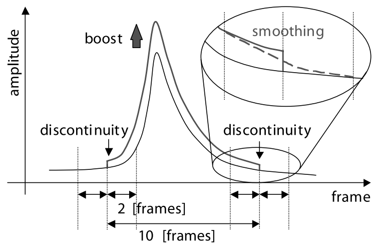


Fig. 6 Spectral smoothing along the time axis before and after the processed spectrogram segment.

parts, the frequency components that were not derived from the target drum sound will also be unnecessarily adjusted.

To solve this problem, we dynamically the adjust amplitude change ratio r to minimize unnecessary adjustments to frequency components. We decrease r when adjusting the small local frequency components in P_X. That is, we replace r in Eq. (1) with

$$R(t,f) = \frac{P_X(t,f)}{P_M} \cdot r, \tag{2}$$

where P_M is the maximum local amplitude included in P_X.

3.4.2 Temporal Smoothing

When the spectrogram P_X is added to or subtracted from an original spectrogram segment, temporal continuity deteriorates before and after the processed spectrogram segment, as depicted in **Fig. 6**. This temporal discontinuity may have a negative influence on the listener's musical perception.

To cope with this, we conduct spectral smoothing in the neighboring frames of each discontinuous frame. Frequency components for each frequency band in a range of 4 [frames] (i.e., 2 [frames] before and after a discontinuous frame) are smoothed using Savitzky-Golay's smoothing method[11].

4. Music Content Analysis

This section describes the computational methods that automatically analyze the musical contents of polyphonic audio signals. This analysis should be done *in advance* for the drum-part rearrangement functions to obtain the three musical contents mentioned at the beginning of Section 3. To obtain the first two contents, we use our method of drum sound recognition[8]. To obtain the third, we use the beat-tracking method proposed by Goto[9].

4.1 Drum Sound Recognition

We first describe the method that estimates the spectrograms of isolated bass and snare drum sounds and detects the onset times of each drum sound.

4.1.1 Problems

There are two problems in estimating the spectrograms and detecting the onset times:

- *Problem 1* (*individual differences*):
The spectrograms of actual drum sounds used in an input audio signal cannot be prepared beforehand because the sound characteristics depend on that piece.
- *Problem 2* (*overlapping sounds*):
It is quite difficult to detect the onset times of the drum sounds when other sounds overlap them.

4.1.2 Approach

To solve these problems, we use a drum sound recognition method, called AdaMast[8], which comprises template-adaptation and template-matching stages using the spectrogram of each drum sound as a template. The adaptation stage yields the actual spectrogram of each drum sound used in an input audio signal. The matching stage detects the onset times of that drum sound even if other sounds simultaneously occur. These stages are sequential and can be independently performed to detect the onsets of bass and snare drum sounds.

- *Solution 1* (*template adaptation*): First, we should prepare spectrograms of isolated bass and snare drum sounds. These initial spectrograms are called *seed templates*, and they are different from the actual drum-sound spectrograms in an input audio signal. Note that this is not a problem because each seed template is iteratively adapted to spectrograms extracted from highly likely onsets of the corresponding drum sound. These onsets are selected from many onset candidates found as possible onsets of various instruments.
- *Solution 2* (*template matching*): This stage uses the spectral distance measure proposed by Goto and Muraoka[12], which is robust against overlapping sounds. With this distance measure, an adapted spectrogram is compared with a spectrogram extracted from each onset candidate to identify whether it is the actual onset of the corresponding drum.

4.2 Bar-line Time Estimation

Next, we explain the beat-tracking method that estimates the beat times including the bar-line times.

4.2.1 Problems

There are two major problems with tracking the beat times:

- *Problem 1:* The beat-time candidates should be detected from polyphonic audio signals by focusing on appropriate musical contents.
- *Problem 2:* There is ambiguity in selecting correct combinations of beat times from the beat-time candidates.

4.2.2 Approach

These two problems can be solved by using Goto's beat-tracking method [9]:

- *Solution 1* (*drum sound detection*): The method tries to detect drum-sound onsets and chord changes as important cues that indicate beat-time candidates. In this study, we use our method of drum sound recognition to detect drum-sound onsets.
- *Solution 2* (*multi-agent architecture*): The architecture is based on multiple agents that have different interpretations of locations for beat times. These agents always predict coming beat times simultaneously, and an agent that has the most confident interpretation outputs the beat times. This enables robust beat tracking.

4.2.3 Advantages of Our Approach

We discuss the advantages of AdaMast and Goto's methods in terms of analysis accuracy and calculation time.

- *Good performance:* AdaMast can robustly recognize drum sounds in polyphonic audio signals. In fact, AdaMast won first prize in the Audio Drum Detection Contest [13], where several methods were compared for their accuracies of onset detection. Goto reported that his beat-tracking method worked accurately for about 90% of the tested popular songs. This is sufficiently high for practical use.
- *Practical computational cost:* Both methods are fast enough to achieve a stress-free musical appreciation environment. This was an important factor in our decision to use AdaMast. It takes less time to complete a content analysis on a standard personal computer in 2005 than to play a musical piece (i.e., it is faster than a real-time performance).

5. Subjective Evaluation

To evaluate Drumix, we conducted subjective experiments with 24 subjects who were university students.

Table 1 Six popular songs used in the experiments.

song	onset detection accuracy bass drum	onset detection accuracy snare drum	tempo (M.M.)
No.13	83.6 %	75.0 %	103
No.21	80.6 %	82.8 %	98
No.7	94.2 %	76.9 %	122
No.90	79.0 %	90.8 %	127
No.35	75.5 %	70.5 %	170
No.37	70.1 %	76.4 %	184

Table 2 12 cases of each comparative experiment.

	S1 →	S2		S2 →	S1
(1)	No.13	No.21	(7)	No.13	No.21
(2)	No.21	No.13	(8)	No.13	No.21
(3)	No.7	No.90	(9)	No.90	No.7
(4)	No.90	No.7	(10)	No.7	No.90
(5)	No.35	No.37	(11)	No.37	No.35
(6)	No.37	No.35	(12)	No.35	No.37

Note: For convenience, the two systems to be compared are called S1 and S2.

5.1 Experimental Conditions

Table 1 lists the six popular songs we used. The songs were extracted from the popular music database *RWC-MDB-P-2001* [14]. comprising 100 copyright-cleared popular songs. They contained the sounds of various instruments as might be expected in commercial CD recordings. The original data were stereo recordings sampled at 44.1 [kHz] with 16 [bits]. We converted them into monaural recordings.

The six songs were categorized into three pairs: (No.13, No.21), (No.7, No.90), and (No.35, No.37). The two songs of each pair had similar tempi and moods. Their drum-sound spectrograms were correctly estimated with AdaMast. The onset detection accuracies (harmonic means of recall rates and precision rates) achieved with AdaMast are listed in Table 1. The bar-line times were accurately estimated using Goto's beat-tracking method.

We conducted two comparative experiments. In each experiment, two systems to be compared were presented to 24 subjects (two systems in the first experiment were different from those in the second one). Note that we considered the systems' order of presentation and song combination. In total, 12 cases in **Table 2** were evaluated (two subjects were allocated to each case).

5.2 Evaluation of Capability for Customizing Music

To evaluate the usefulness of Drumix in customizing music, we conducted an experiment

Table 3 Comparative evaluation of Drumix and graphic equalizer: 24 subjects were asked to answer each question on a seven-point scale (−3 representing the "graphic equalizer" and 3 representing "Drumix").

question	histogram for scores							average	standard
	−3	−2	−1	0	1	2	3	score	deviation
Which system helped you to enjoy the music?	0	1	6	1	3	9	4	1.0	1.6
Which system gave an unexpected experience?	0	3	2	1	4	12	2	1.1	1.6
Which system let you customize music according to your preference?	0	1	5	8	5	4	1	0.38	1.2

Table 4 Comparative evaluation of Islands of Drum Patterns: 24 subjects were asked to answer each question on a seven-point scale (−3 representing "absence of Islands of Drum Patterns" and 3 representing "presence of them").

question	song	histogram for scores							average	standard
	numbers	−3	−2	−1	0	1	2	3	score	deviation
Which system helped you to compose favorite drum patterns?	No.13, No.21 No.7, No.90	0	3	0	0	7	3	3	1.0	1.7
	No.35, No.37	0	0	3	1	2	2	0	0.38	1.3

comparing Drumix and the ordinary graphic equalizer that is attached to Microsoft Windows Media Player 10. We told 24 subjects to freely customize musical pieces instead of passively listening to them. All subjects freely used both two systems. Drumix was given only the volume-control and timbre-change functions. Each system was used for 2 [min], which excluded the time to give brief instructions.

After using both systems, each subject was asked to answer three comparative questions listed in **Table 3**. Each question was answered on a seven-point (−3-to-3) scale where −3 corresponds to the graphic equalizer and 3 to Drumix, respectively. 0 represents the equivalence of both systems. For every comparison, Drumix outperformed the graphic equalizer, which is currently one of the most powerful tools for customizing music. In addition, some subjects answered that they felt as if they had successfully rearranged MIDI files in spite of their lack of musical skills. This proves that Drumix is capable of assisting users to enjoy musical composition, and provided them with unexpected exciting experiences. However, Drumix was not superior to the graphic equalizer in terms of the quality of generated sounds. Quality improvements should be addressed in future work.

5.3 Evaluation of Islands of Drum Patterns

To evaluate the usefulness of the Islands of Drum Patterns, we conducted a second experiment in which 24 subjects edited drum patterns using two systems: the drum-pattern editor with the Islands of Drum Patterns and the editor without them. Each system was used for 3 [min], which excluded the time to give brief instructions.

In advance, we had expected that the Islands of Drum Patterns would be less helpful for fast-tempo songs than for normal-tempo ones. This is because there is little room for greatly modifying the original drum patterns of fast-tempo songs. These drum patterns tend to have only four onsets: two bass-drum onsets on the first and third beats and two snare-drum onsets on the second and fourth beats.

After using both systems, each subject was asked to answer the question shown in **Table 4**. The question was answered on a seven-point scale where −3 (3) represents the absence (presence) of the Islands of Drum Patterns. The results proved against our expectation the Islands of Drum Patterns were useful in all the songs. The results of the F-test and Student's t-test revealed that there was no difference in usefulness between fast-tempo songs and normal-tempo ones at a significance level of 5%.

5.4 Evaluation of Drum-part Rearrangement Functions

To evaluate the three functions of Drumix in terms of usability, interest, and necessity, each subject was asked to answer nine questions on a seven-point scale where −3 corresponds to "no" and 3 to "yes" after they had freely used Drumix. **Table 5** lists the questions and his-

Table 5 Evaluation of three functions for drum-part rearrangement: 24 subjects were asked to answer each question on a seven-point scale (−3 representing "no" and 3 representing "yes").

function	question	histogram for scores							average score	standard deviation
		−3	−2	−1	0	1	2	3		
volume control function	Is this easy-to-use?	0	0	0	0	2	6	16	2.6	0.65
	Is this interesting?	0	0	2	8	4	6	4	1.1	1.3
	Is this necessary?	0	0	0	3	7	7	7	1.8	1.0
timbre change function	Is this easy-to-use?	0	0	0	1	3	7	13	2.3	0.87
	Is this interesting?	0	0	1	2	6	8	7	1.8	1.1
	Is this necessary?	0	0	0	1	6	11	6	1.9	0.83
drum-pattern editing function	Is this easy-to-use?	1	2	6	2	5	4	4	0.50	1.8
	Is this interesting?	0	0	0	2	2	11	9	2.1	0.90
	Is this necessary?	0	0	2	3	4	7	8	1.7	1.3

tograms for the corresponding answers.

The usabilities of the volume-control and timbre-change functions were excellent because these functions are based on familiar interfaces such as sliders and drop-down lists. Although the average usability score of the drum-pattern editing function was not good, we should take into account that the subjects used Drumix for only a few minutes; the period was too short for them to become familiar with the new interface.

The subjects' interest in the three functions was sufficiently high. Their interest increased in order of the volume-control function, the timbre-change function, and the drum-pattern-editing function. The latter gave the subjects more freedom to rearrange drum parts. This resulted in a more exciting experience of musical composition.

The necessity for all functions was quite high. An interesting fact is that these necessities remained almost the same despite the differing interest. This indicates that the three functions worked well cooperatively as we expected.

5.5 Shortcomings of Drumix

When the 24 subjects were asked whether they wanted to continue using Drumix after the experiments, 20 (83%) answered "yes." This indicates that Drumix would be a useful tool for satisfying the desires of active music listeners. Here, we would like to assess the negative answers. The subjects who did not want to continue to use Drumix tended to have the following opinions:

- *Opinion 1:* Because the existing audio recordings have already been "completed," there is no room for manipulating them.
- *Opinion 2:* It was not interesting to rearrange drum parts.

Two subjects who are skilled at playing musical instruments were of the first opinion. They seemed to be interested in masterpieces, which could serve good references for improving their own playing and composing skills. The fulfillment of such an interest would require the generated audio signals to have higher quality. Therefore, we plan to improve the drum sound detection performance. Note that only two subjects with no musical skills were of the second opinion.

6. Discussion

This section discusses the significance of Drumix in terms of "reuse" of conventional user interfaces and existing music designs.

6.1 Reuse of User Interfaces

We designed Drumix on the basis of familiar user interfaces, i.e., horizontal sliders for the volume-control function, drop-down lists for the timbre-change function, and a MIDI-sequencer display of scores for the drum-pattern editing function. In general, these familiar interfaces themselves cannot provide users with unexpected interactive experiences.

Despite this, users actually had such experiences when they used Drumix. The novelty was in Drumix's ability to overcome common limitations, such as:

- It is normally impossible to manipulate only the drum part in a commercial CD recording.
- It is normally impossible to rearrange drum parts without special musical skills.

These are key points in which our work is different from other studies whose main objective was to design a new user interface. Even if we reuse existing familiar interfaces, novel interactive experiences can be achieved by leveraging state-of-the-art signal processing and pattern recognition techniques.

6.2 Reuse of Music Designs

With the Islands of Drum Patterns, existing drum patterns can be reused to suggest new compositions. These existing drum patterns were extracted from MIDI files, and musical contexts such as the moods and tempi of the original songs were discarded. Therefore, some subjects said that it was better that they could freely exchange drum patterns between two arbitrary audio recordings. This extension can be easily achieved using the methods described in this paper.

The reuse of music designs of existing musical pieces (final products) has been addressed in some studies. To let computers expressively perform music, Toyoda, et al. [15] constructed a database of performance deviations (differences between scores and performances), which were collected from actual expressive performances. By using this database, Katayose and Okudaira [16] proposed a user interface for conducting music by gesture, in which users can feel as if they conducted excellent performances. Katayose, et al. [17] developed a user interface that assists amateurs to perform the appropriate mix-down, which requires expertise, in a process of music production. For example, when a user wants to compose a metal-rock song, the user can reuse mix-down parameters (e.g., volume balance of instrument tracks) of an existing metal-rock song composed by a professional musician.

7. Conclusion

We described a highly functional audio player called Drumix that can rearrange drum parts in existing audio recordings in real time. Drumix provides users with easy-to-use interfaces, i.e., horizontal sliders for controlling the volumes of bass and snare drum sounds, drop-down lists for selecting the timbres of these drum sounds, and a MIDI-sequencer display of scores for editing drum patterns. These three functions are achieved by automatically detecting the onset times for each drum sound while estimating its power spectrogram. To implement the drum-pattern editing function, the bar-line times are also automatically estimated by using the detected drum-sound onsets. The results of subjective experiments using popular songs demonstrated that Drumix was usable and capable for customizing music. We believe that users will enjoy the experiences of simultaneously listening to and composing music.

A main contribution of this work is the proposal of a new active-music-listening concept based on decomposition and re-synthesis of musical instrument parts within existing audio signals. To achieve this, various methods for musical instrument recognition should be integrated. We believe this is an indispensable study using many promising techniques in the field of music information processing for practical use. We plan to use sound source separation techniques to rearrange various instrument parts.

Demonstration video clips are available at
http://winnie.kuis.kyoto-u.ac.jp/
members/yoshii/drumix/

Acknowledgments This research was partially supported by a Grant-in-Aid for Scientific Research from the JSPS Research Fellowship and by the COE program of MEXT.

References

1) Goto, M.: Linking Music Information Processing with the Real World: Music Information Processing for End Users, *Technical Committee Meeting on Musical Acoustics* (*in Japanese*), MA2005-39, Acoustical Society of Japan, Vol.24, No.4, pp.97–100 (2005).
2) Goto, M.: SmartMusicKIOSK: Music Listening Station with Chorus-search Function, *UIST*, pp.31–40 (2002).
3) Last.fm: http://www.last.fm/
4) Pandora: http://pandora.com/
5) Goto, M. and Goto, T.: Musicream: New Music Playback Interface for Streaming, Sticking, Sorting, and Recalling Musical Pieces, *ISMIR*, pp.404–411 (2005).
6) Goto, M.: A Chorus-Section Detecting Method for Musical Audio Signals, *ICASSP*, pp.437–440 (2003).
7) Pachet, F. and Delerue, O.: Annotations for Real Time Music Spatialization, *Int. Workshop on Knowledge Representation for Interactive Multimedia Systems* (1998).
8) Yoshii, K., Goto, M. and Okuno, H.G.: Automatic Drum Sound Description for Real-World Music Using Template Adaptation and Matching Methods, *ISMIR*, pp.184–191 (2004).
9) Goto, M.: An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds, *Journal of New Music Research*, Vol.30, No.2, pp.159–171 (2001).
10) Pampalk, E., Dixon, S. and Widmer, G.: Exploring Music Collections by Browsing Different Views, *ISMIR*, pp.201–208 (2003).
11) Savitzky, A. and Golay, M.: Smoothing

and Differentiation of Data by Simplified Least Squares Procedures, *Analytical Chemistry*, Vol.36, No.8, pp.1627–1639 (1964).

12) Goto, M. and Muraoka, Y.: A Sound Source Separation System for Percussion Instruments, *IEICE Trans.D-II* (*in Japanese*), Vol.J77-D-II, No.5, pp.901–911 (1994).

13) Yoshii, K., Goto, M. and Okuno, H.G.: AdaMast: A Drum Sound Recognizer based on Adaptation and Matching of Spectrogram Templates, *1st Annual Music Information Retrieval Evaluation eXchange* (*MIREX*) (2005).

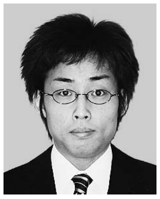
14) Goto, M., Hashiguchi, H., Nishimura, T. and Oka, R.: RWC Music Database: Popular, Classical, and Jazz Music Databases, *ISMIR*, pp.287–288 (2002).

15) Toyoda, K., Noike, K. and Katayose, H.: Utility System for Constructing Database of Performance Deviations, *ISMIR*, pp.373–380 (2002).

16) Katayose, H. and Okudaira, K.: Using an Expressive Performance Template in Music Conducting Interface, *NIME*, pp.124–129 (2004).

17) Katayose, H., Yatsui, A. and Goto, M.: A Mix-Down Assistant Interface with Reuse of Examples, *AXMEDIS*, pp.9–16 (2005).

(Received June 21, 2006)
(Accepted September 14, 2006)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.3, pp.134–144.)



Kazuyoshi Yoshii received the B.E. and M.S. degrees from Kyoto University, Japan, in 2003 and 2005, respectively. He is currently a Ph.D. candidate of the Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University. He is supported by the JSPS Research Fellowships for Young Scientists (DC1). He has received several awards including the FIT Paper Award (FIT 2004), the Best Interactive Presentation Award (Interaction 2006), and the IPSJ Yamashita SIG Research Award. His research interests include music scene analysis and music recommendation.

Masataka Goto received the Doctor of Engineering degree from Waseda University, Japan, in 1998. He is currently a Senior Research Scientist of the National Institute of Advanced Industrial Science and Technology (AIST). He serves concurrently as an Associate Professor of the Graduate School of Systems and Information Engineering, University of Tsukuba. He has received 19 awards, including the IPSJ Best Paper Award, IPSJ Yamashita SIG Research Awards, and Interaction 2003 Best Paper Award.

Kazunori Komatani received the B.E., M.S. and Ph.D. degrees from Kyoto University, Japan, in 1998, 2000, and 2002, respectively. He is currently an Assistant Professor of the Graduate School of Informatics, Kyoto University, Japan. His research interests include spoken dialogue systems.

Tetsuya Ogata received the B.E., M.S. and Ph.D. degrees from Waseda University, Japan, in 1993, 1995, and 2000, respectively. He is currently an Associate Professor of the Graduate School of Informatics, Kyoto University, Japan. He serves concurrently as a Visiting Associate Professor of the Humanoid Robotics Institute of Waseda University. His research interests include multi-modal active sensing and robot imitation.

Hiroshi G. Okuno received the B.A. and Ph.D. degrees from the University of Tokyo, Japan, in 1972 and 1996, respectively. He is currently a Professor of the Graduate School of Informatics, Kyoto University, Japan. He received various awards including the Best Paper Awards of JSAI. His research interests include computational auditory scene analysis, music scene analysis and robot audition.