AUTORHYTHM: A MUSIC GAME WITH AUTOMATIC HIT-TIME GENERATION AND PERCUSSION IDENTIFICATION

Pei-Pei Chen[1], Tzu-Chun Yeh[2], Jyh-Shing Roger Jang[3], Wenshan Liou[4]

[1]Graduate Institute of Electrical Engineering, National Taiwan University
[2]Department of Computer Science, National Tsing Hua University
[3]Department of Computer Science and Information Engineering, National Taiwan University
[4]Smart Network System Institute, III, Taipei, Taiwan, R.O.C.

{peipei.chen, kenshin.yeh, jang}@mirlab.org, wsliou@iii.org.tw

ABSTRACT

This paper describes a music rhythm game called AutoRhythm, which can automatically generate the hit time for a rhythm game from a given piece of music, and identify user-defined percussions in real time when a user is playing the game. More specifically, AutoRhythm can automatically generate the hit time of the given music, either locally or via server-based computation, such that users can use the user-supplied music for the game directly. Moreover, to make the rhythm game more realistic, AutoRhythm allows users to interact with the game via any objects that can produce percussion sound, such as a pen or a chopstick hitting on the table. AutoRhythm can identify the percussions in real time while the music is playing. The identification is based on the power spectrum of each frame of the recording which combines percussions and playback music. Based on a test dataset of 12 recordings (with 2455 percussions of 4 types), our experiment indicates an F-measure of 96.79%, which is satisfactory for the purpose of the game. The flexibility of being able to use any user-supplied music for the game and to identify user-defined percussions from any objects available at hand makes the game innovative and unique of its kind.

Index Terms— Music game, Rhythm game, Onset detection, Percussion identification, Content generation

1. INTRODUCTION

Among games over mobile devices, music games are among the most popular ones, mainly because of the close and intense interactions between the user and the game. Recently, music rhythm games has attracted more and more users with their variety and richness in interactions. Thus, more and more research and business investments are focusing on the next generation of the music games, especially music rhythm games.

A typical flow of a music rhythm game is described next. First, a user chooses a music piece that he/she would like to play with, and then interact with the game system. Most of the music rhythm games will ask the user to press a specific button, or hit on a surface or a dummy instruments at the right timing to obtain scores. Users may compete with other players through LAN or internet. After finishing playing the game, a user may be awarded a score or rank which represents the skill level of the user. According to the score, the user can achieve a title or a bonus for the next round of game playing. This may create an atmosphere to encourage the user to play more games in order to achieve more titles.

However, there are still several issues that hinder the prevalence of the game. One of them is that a user cannot choose any song from their own music collection. Currently, all the music pieces in a given music rhythm game are fixed since the hit-time contents must be generated manually in advance. This paper introduces a novel music rhythm game system that can automatically generate the playable hit-time contents for any given music piece. A user can simply upload a song from his/her collection, and plays with his/her

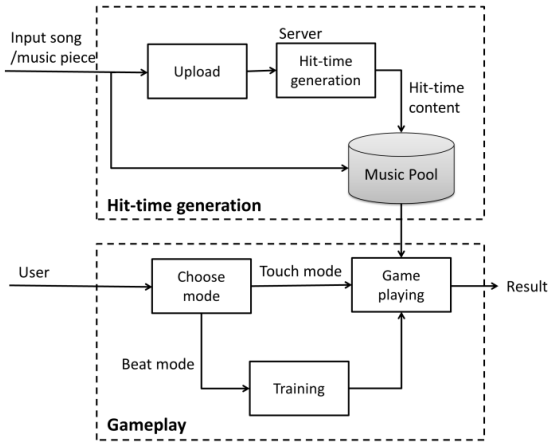

Fig. 1. System Overview.

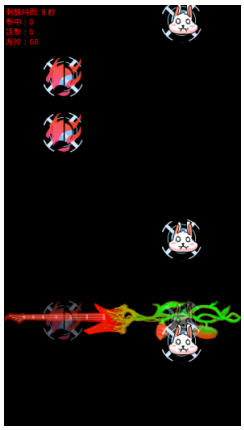

Fig. 2. Screenshot of AutoRhythm during gameplay.

favorite songs immediately after the hit-time contents are generated by the server automatically. Moreover, a user can even use any objects as percussion instruments rather than press buttons or tap screen, which provides more interaction and fun. The proposed system increases the playability of the rhythm game, and makes gaming experience more personalized.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to the whole system. Section 3 describes the features and the methods for percussion identification and automatic hit-time generation. Then, the experimental result is shown in section 4. Error analysis is discussed in section 5. Section 6 gives the conclusions and points out possible future direction of this work.

2. SYSTEM OVERVIEW

Fig.1 shows the main blocks of the proposed system. A user can upload any songs or music pieces to our server for hit-time generation. Once the hit time is available, a user can start playing the game with two different modes (to be detailed later). Fig. 2 is the screenshot of AutoRhythm, where there are two lines of icons falling down. The hit time is actually the time when an icon hit the horizontal bar.

To make the game more realistic, AutoRhythm provides 2 different modes for playing. The traditional playing mode is via tapping. That is, a user can tap the horizontal bar when an icon hits the bar in order to obtain scores. This paper proposes an innovative playing mode of tapping on real objects. That is, a user can choose any two kinds of sounds made with any readily available objects, such as a pencil tapping on the desk and a remote control tapping on the chair leg. These two kinds of sounds correspond to the left and right icons, respectively. That is, when any left icon falls onto the horizontal line, the user needs to make the first kind of sound to obtain score. Similarly, when any right icon falls onto the horizontal line, the user needs to make the second kind of sound. In the training stage, user must record these 2 kinds of sounds

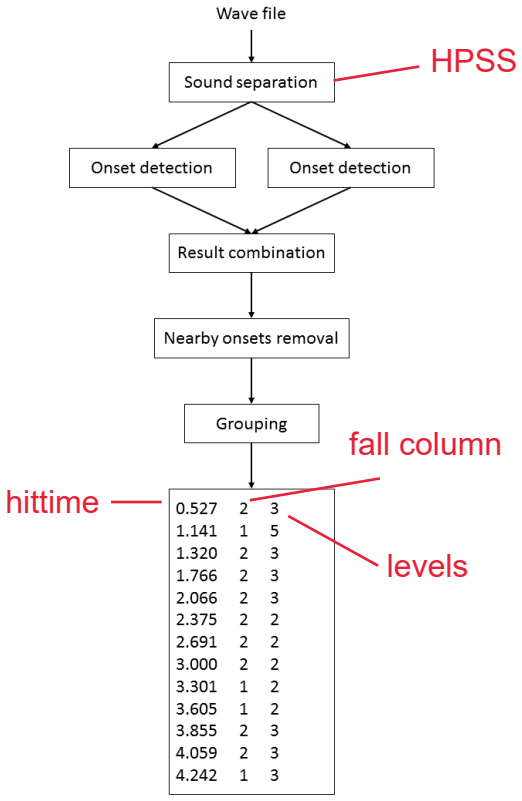

Fig. 3. Basic blocks of hit time generation.

several times for model training. The detailed description is in section 3.2.

3. FEATURES AND METHOD

3.1 Hit time generation

Fig. 3 illustrates the flowchart of hit-time generation of AutoRhythm. The hit times of a given music piece usually coexist with onsets of both harmonic components (such as string instruments or human voices) and percussive components (such as drums or cymbals). Thus, we need to separate the given audio music signals into harmonic and percussive sources by harmonic/percussive source separation (HPSS) [1]. After HPSS, spectral-flux-based onset detection is applied to both sources to find the onsets in both parts. Here, we proposed a weighted version of spectral flux to achieve better performance of detected onsets when compared with human labeled ones. The final hit times are obtained by combining the onsets from both sources, and eliminating hit times that are too close. Moreover, based on their density, the hit times are grouped into various sets corresponding to the levels of difficulty of the rhythm game.

The format of the output file is also shown in Fig. 3, where the first column is the hit times in second, the second

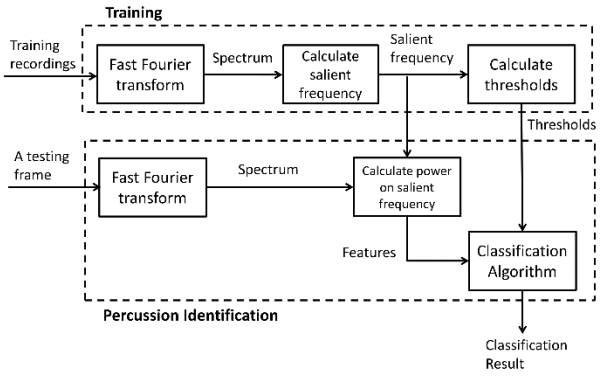

Fig. 4. Flowchart of percussion identification method.

column is the channel ID of the falling icons, and the last column is the levels of difficulty. The channel assignment is based on k-means clustering on MFCC (Mel-frequency Cepstral coefficients) at the hit times, such that different channels are likely to host onsets from different instruments.

3.2 Percussion identification

There have been some studies on percussion/drum sound identification in the literature. Herrera [2] compared lots of features and classification techniques, including attack-related descriptors, decay-related descriptors, relative energy descriptors, and MFCC as features, and K-nearest neighbors, Canonical discriminant analysis, and decision tree as classifiers. W. Andrew Schloss [3] used power of selected frequency bins and music structure to perform the transcription of percussive music. However, both studies can only classify monosyllabic/clear percussion sounds without background music/noisy.

Gouyon [4] identified percussive sounds in popular music, using features related to attack time, spectrum, and zero-crossing rate. The percussive sounds are classified into snare-like sounds and bass-drum-like sounds with agglomerative clustering. Yoshii [5] used an iterative adaptation algorithm to obtain the adapted template of power spectrum of percussive sounds in polyphonic signals. However, none of these studies identify percussion sounds in real time since they used either long-term features or features which are too complicated to calculate in real time.

To perform real-time percussion identification, the features must be short-term (frame-based) and the computation for identification must be as simple as possible to shorten the delay. Here we propose a simple but effective method which classifies a frame with only 2 spectrum-based features, as shown in Fig. 4. Details about the features and the classification algorithm are described in *3.2.1* and *3.2.2*.

3.2.1 Features

After observing the magnitude spectra of percussion sounds from different objects and surfaces, we found that different

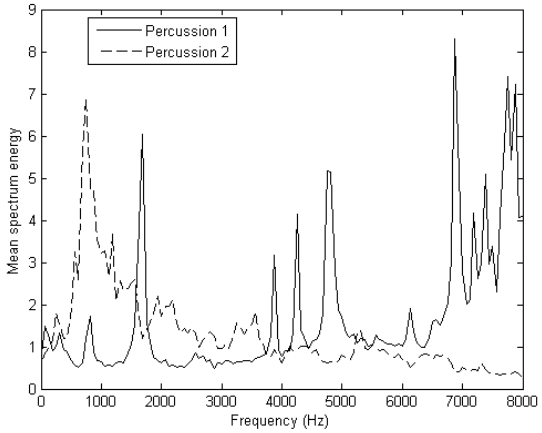

Fig. 5. Mean spectral energy of two different kinds of percussion sounds.

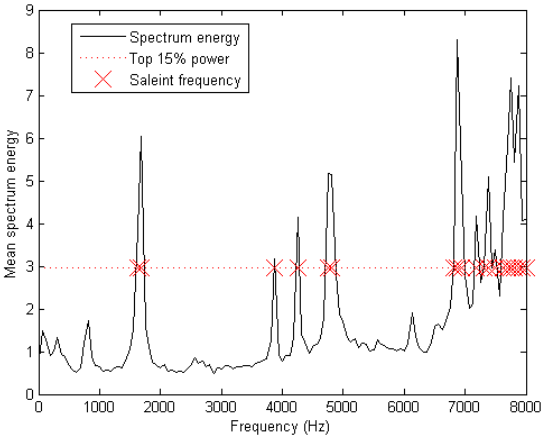

Fig. 6. Salient frequencies of percussion 1 in Fig. 5.

percussion sounds are likely to have different spectral patterns with distinctive peaks at different frequencies, as shown in Fig. 5. Based on the observation, we can define the salient frequencies as the frequency bins with top 15% of the magnitude spectrum of a given type of percussion, as shown in Fig. 6. The magnitudes on the salient frequencies are thus representative for a percussion sound, which are not easily interfered by the music being played.

Before users start to play the game, they are asked to record two types of percussion sounds to be used for the game. Each of the recording lasts for 5 seconds, and energy-based endpoint detection is applied to locate where the percussions are. After obtaining the frames with percussion sounds, the first three frames of each percussion sound are used to calculate the salient frequencies because the timbre of a percussion sound may change along time.

In this studies, we consider only two types of percussions to be used in the game. As a result, we have two sets of salient frequencies. We shall use the total energy within the salient frequencies as the frame-based feature for

Algorithm 1. Pseudo code for classifying a frame

if $SF_{(i,A)} > Th_A$ **and** $SF_{(i,B)} > Th_B$
  if $SF_{(i,A)} > SF_{(i,B)}$
    '$percussion_A$ at $frame_i$.'
  else
    '$percussion_B$ at $frame_i$.'
  end
else if $SF_{(i,A)} > Th_A$
  '$percussion_A$ at $frame_i$.'
else if $SF_{(i,B)} > Th_B$
  '$percussion_B$ at $frame_i$.'
else
  'No percussion at $frame_i$.'
End

classifying a frame into percussion_A, percussion_B, or none of the above.

3.2.2 Proposed Algorithm

To reduce the interference from the background music, the average of the magnitude spectrum is subtracted from the frame, such that the sum of magnitude spectrum is equal to zero in each frame.

We use several thresholds to identify whether there is a percussion sound or not. The thresholds are derived from the user-recording of two types of percussions at the beginning of the game. After obtaining the salient frequencies of each type of percussion, the average power magnitude on salient frequencies of each frame with percussion is calculated. Then, the average magnitudes on salient frequencies are summed up and multiplied by 0.4 (obtained empirically) to have the thresholds of two types of percussion, Th_A and Th_B, respectively

Let $\text{SF}_{(i,A)}$ and $\text{SF}_{(i,B)}$ denote the sum of magnitude on salient frequencies of frame_i of percussion_A and percussion_B, respectively. Then the pseudo code for classifying a frame is shown in Algorithm 1.

Since a percussion sound usually last for more than one frame, some post-processing is necessary to prevent a sound from being identified duplicately. There is a world record made by Takahashi Meijin during the 1980s, who could trigger 14 times per second in gameplay with a joystick. Therefore, it is not likely for people to make two consecutive percussion sounds within 71.4 ms (*1/14=0.0714*). In this study, each frame lasts for 16 ms (*256/16000=0.016*), so we set a rule that if $frame_i$ is identified as a percussion sound, then $frame_{i+1}$ to $frame_{i+3}$ are considered as none. Furthermore, we believe that the energy of a percussion sound is decreasing with time, so there is another rule that only when $SF_{(i,X)} > SF_{(i-1,X)}$, $frame_i$ would be identified as $percussion_X$.

4. EXPERIMENT

Table 1. The detail information of each type of percussion

	percussion instrument	Number of percussion sounds with *BGM-1*	Number of percussion sounds with *BGM-2*
$Percussion_A$	A smaller ceramic cup	325	304
$Percussion_B$	A larger ceramic cup	329	283
$Percussion_C$	A paper box	324	304
$Percussion_D$	A plastic box	307	279

4.1 Dataset

All the audio files in our dataset are mono wave files recorded by cell phones with a sample rate of 16000Hz and 16-bit resolution. Two types of background music and four types of percussions are collected in this dataset. One of the background music is a rock song with lots of percussions (*BGM-1*) and the other is a soft music with nearly no percussion instruments (*BGM-2*). There are totally 2455 percussion sounds in our test dataset, which were created by using a steel stick tapping on some hard surfaces. The detail information of each type of percussion sounds is shown in Table 1.

There are 16 files in total, 4 for training while 12 for test. Four of them are 5-second pure percussions of each type. These 4 files are regarded as the recordings by users at the beginning of the game for obtaining the salient frequency and thresholds. The other 12 files are 2-minute recordings which combine background music playing and 2 different types of percussion sounds. These files simulate the situation while gameplay and are used for test in our experiment. The whole dataset (for both training and test) is available at http://mirlab.org/dataset/public/AutoRhythmDataset.rar

4.2 Experimental Results

There are four situations which may occur after the classification of a frame – correct, insertion, deletion, and confusion. Assume that there are two types of percussion – $percussion_X$ and $percussion_Y$. $Correct_X$ represents the number of frames where $percussion_X$ is identified correctly. (A percussion sound is considered identified correctly when the deviation is within 2 frames or 32 milliseconds.) $Insertion_X$ represents the number of frames which is without any percussion sound but the classification result says that there is $percussion_X$ at the frames. $Deletion_X$ represents the number of frames where $percussion_X$ is but the classification result tells that there is only background music at the frames. $Confusion_{(X,Y)}$

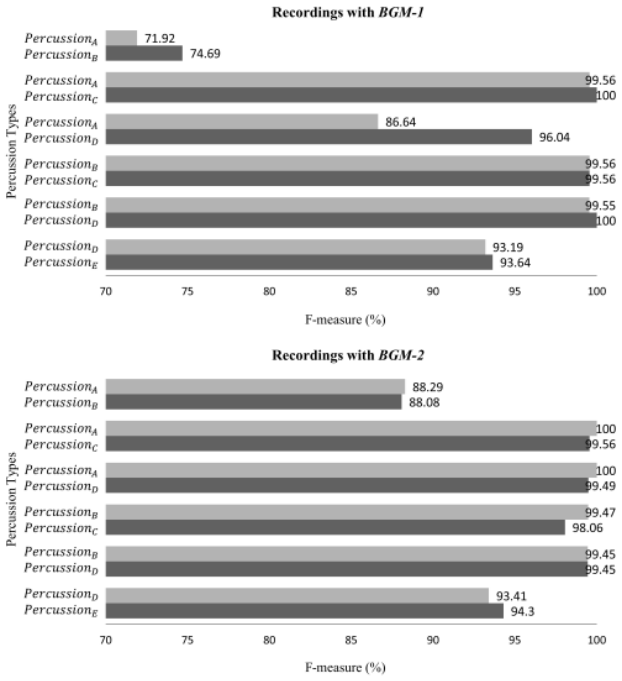

Fig. 7. F-measure with SVM as classifier.

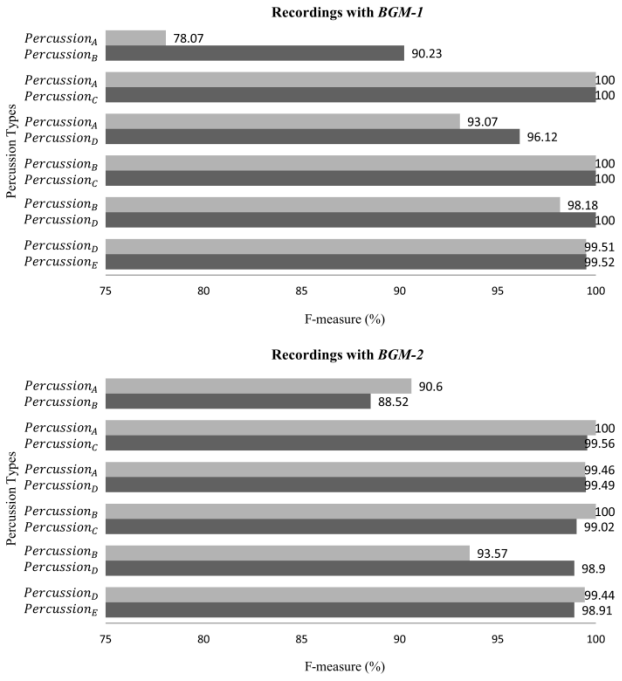

Fig. 8. F-measure with Algorithm 1 as classifier.

Table 2. The experimental result of recordings with *BGM-1*

$Percussion_X$ (Number of sounds)	$Correct_X$	$Insertion_X$	$Deletoin_X$
A (325)	318	55	1
B (329)	317	6	1
C (324)	323	0	0
D (307)	301	2	3

Table 3. The experimental result of recordings with *BGM-2*

$Percussion_X$ (Number of sounds)	$Correct_X$	$Insertion_X$	$Deletoin_X$
A (304)	300	2	3
B (283)	255	3	11
C (304)	303	3	0
D (279)	279	4	0

Table 4. Confusion table of four types of percussion

Percussion (Number of sounds)		Predicted			
		A	B	C	D
Actual	A (629)	618	7	0	0
	B (612)	24	572	0	0
	C (628)	0	0	626	2
	D (586)	3	0	0	580

represents the number of frames where the classification misidentifies $percussion_X$ as $percussion_Y$.

4.2.1 Support Vector Machine (SVM)

We applied SVM with the proposed features, where each frame is classified into three outputs: background-music-only and two types of percussion sounds. 100 frames are randomly selected from the background music to be our training data of the background-music-only frames. The training data for percussions is obtained from the percussion recordings recorded by the user at the beginning of the game. The pre-processing and post-processing proposed in this paper were also performed

F-measure is used to represent the overall performance. The F-measure of $percussion_X$ (F_x) is defined as

$$F_x = 2 \times \frac{precision_x \times recall_x}{precision_x + recall_x}$$

$$precision_x = true\ positive/test\ outcome\ positive = Correct_x/(Correct_x + Insertion_x + Confusion_{(Y,X)})$$

$$recall_x = true\ positive/condition\ positive = Correct_x/(Correct_x + Deletion_X + Confusion_{(X,Y)})$$

The F-measure result with SVM is shown in Fig. 7. The average F-measure of recordings with *BGM-1* is 92.86%, and that with *BGM-2* is 96.63%. The overall average F-measure is 94.75%.

4.2.2 Our classification method

Table 2 and Table 3 show the $Correct_X$, $Insertion_X$ and $Deletion_X$ with *BGM-1* and *BGM-2*, respectively Table 4 is the confusion table which shows $Confusion_{(X,Y)}$ and $Confusion_{(Y,X)}$. The F-measures of each recording is shown in Fig. 8. The average F-measure of recordings with

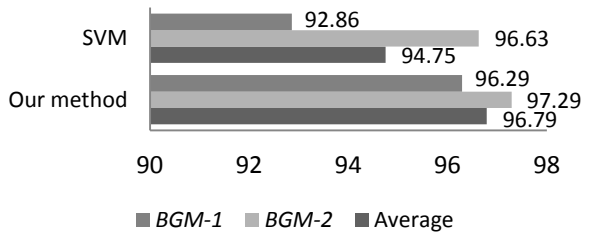

Fig. 9. Results comparison between our method and SVM.

BGM-1 is 96.29%, and that with *BGM-2* is 97.29%. The overall average F-measure is 96.79%.

5. DISCUSSION

5.1 Error Analysis

This section describes some preliminary error analysis of the percussion identification.

Even under the same input percussion instrument, percussion sounds still vary significantly. Players may unconsciously tap different part of the input instrument and use different strength every time, which leads to variations in the generated sounds. For example, tapping on the center or the edge of a box would produce different sounds, leading to inconsistent identification result.

In particular, when players apply less strength in tapping, the volume of the sounds would be lower, which makes deletions more likely to happen. Table 5 shows that the frames where deletion occur have lower volume than the frames identified correctly in average.

In Table 2, recordings with *BGM-1* have lots of insertion of $Percussion_A$. It's because the spectrum peaks of $Percussion_A$ is similar to those of some music instrument in *BGM-1*. Moreover, most of the confusions are between $Percussion_A$ and $Percussion_B$ since the salient frequencies of $Percussion_A$ and $Percussion_B$ have a significant amount in common.

5.2 Discussion over SVM results

Our method outperforms SVM, as shown in Fig. 9. One potential reason is that, SVM was constructed from the training data obtained from the background music which varies a lot from frame to frame. Hence, the training set of background music does not have stable and consistent features, leading to an SVM with less-desirable performance.

It should be noted that SVM's performance is heavily influenced by its parameters [6], and the optimum parameters vary for different percussion sounds. For example, the best parameters for classifying $Percussion_A$ and $Percussion_B$ with *BGM-1* are *gamma = 0.0001, cost = 1*. The best parameters to classify $Percussion_A$ and $Percussion_D$with *BGM-2* are *gamma = 0.0004, cost = 1*. Therefore the parameters which maximize the overall F-measure may not be the best parameters for every recording.

Table 5. Average volume of correct and deletion frames

	Average volume (decibel)	
	Correct frames	Deletion frames
$Percussion_A$	23.12	10.59
$Percussion_B$	24.39	5.97
$Percussion_C$	31.32	No deletion occurs
$Percussion_D$	30.19	19.65

6. CONCLUSIONS

In this paper, we have introduced an innovative music rhythm game "AutoRhythm" which provides a new way to interact with users. The innovation is two folds. First of all, AutoRhythm can locate the hit time of the user-supplied music to be used with the game. Secondly, users can tap along the rhythm of the music with any objects that can produce clear and distinct percussion sounds. Experimental results show that the proposed algorithm for percussion identification during gameplay can achieve an F-measure of 96.79%, which is quite feasible for the game. These two folds of innovation make the game quite unique and attractive of its kind.

7. REFERENCES

[1] N. Ono, K. Miyamoto, J. Le Roux, H. Kameoka and S. Sagayama, "Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram," in Proceedings of the European Signal Processing Conference, 2008.

[2] Perfecto Herrera, Alexandre Yeterian, and Fabien Gouyon, "Automatic Classification of Drum Sounds: A Comparison of Feature Selection Methods and Classification Techniques," International Conference on Music and Artificial Intelligence (ICMAI), 2002.

[3] W. Andrew Schloss, "On the Automatic Transcription of Percussive Music – From Acoustic Signal to High-level Analysis," STAN-M-27, Stanford, CA, CCRMA, Department of Music, Stanford University, 1985.

[4] Fabien Gouyon, Francois Pachet, Oliver Delerue, "On the Use of Zero-Crossing Rate for and Application of Classification of Percussive Sounds," the COST G-6 Conference on Digital Audio Effect, 2000.

[5] Kazuyoshi Yoshii, Masataka Goto, Hiroshi G. Okuno, "Automatic Drum Sound Description For Real-world Music Using Template Adaptation and Matching Methods," International Conference on Music Information Retrieval, 2004.

[6] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: a library for support vector machines," ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.