AN INTERACTIVE RHYTHM TRAINING TOOL FOR USULS OF TURKISH MAKAM MUSIC

Burak Uyar
Bahçeşehir University
burak.uyar@stu.bahcesehir.edu.tr

Barış Bozkurt
Koç University
barisbozkurt0@gmail.com

ABSTRACT

The education of the Turkish makam music practice is carried out by face-to-face sessions with the masters. Our aim in this work is to present a rhythm training software that helps users imitate recording of a master and receive visual feedback about their performances compared to the recording of the master. The system automatically detects onsets in the recordings and then performs comparison of the onset locations with a novel approach defined here. We present a demonstration of the software and preliminary test results on basic rhythmic patterns of Turkish makam music.

1. INTRODUCTION

In Turkish makam music culture, education process is mainly based on practicing the melodic and rhythmic patterns. Being an oral tradition, the fundamentals of this music culture is generally learned from a master in face-to-face sessions. This education model is called *meşk* in Turkish makam music tradition. During *meşk*, the master explains and demonstrates the pattern and then the student practices until the master approves the student's proficiency. This study targets the presentation of a software tool developed to help students practice rhythmic patterns at home.

In this study, we specifically focus on practicing the rhythmic patterns in makam music. The rhythmic patterns are called *usuls* in Turkish makam music. A sufficient description for *usuls* is as stated in Bozkurt et al. [2014], "An usul is a rhythmic pattern of a certain length that contains a sequence of strokes with varying accents." There are a few studies and commercial applications related to *usul* training. One of them is Mus2Okur, which includes a large amount of *usuls* with auditory material and symbolic representations. Another application is Usul-Velvele-Editor, which also provides auditory and visual data for *usuls*. These applications include comprehensive collections but lack interactivity with the user.

Various interactive tutoring systems have been developed previously for other music cultures. An interactive music training system is described by Percival [2008]. In the rhythm training part of the system, the user's performance recording is analyzed and an accuracy score is provided. One other interactive system is developed in the *i-maestro* [1] project. One of the key outcomes of that system is denoted as creating tools for instrumental training in different environments. This system includes both auditory and visual analysis of a user's performance. Wang and Lai [2011] have developed a mobile digital game based tool for rhythm learning. Ferguson [2006] described an interactive real-time system for instrumental practice and tuition. Existence of such tools encourage us to create similar systems for *usul* training.

[1] http://www.i-maestro.org/

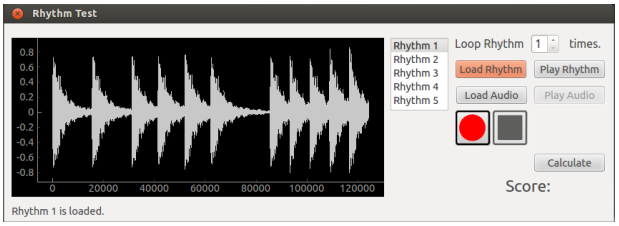

Figure 1: Software interface with a reference rhythm loaded.

In our study we consider the points mentioned above to create a tool for learning and practicing *usuls*. By taking the advantages of the MIR methodologies, it is possible to develop a complementary tool to support the *meşk* process. In case of difficulty in accessing to masters of Turkish makam music, this tool can also be used for self-tutoring. The following parts of the paper include the description of the system with its constraints, data, modules and methods.

2. SYSTEM DESCRIPTION

The main purpose of the program is to provide a tool for practicing *usuls* in an interactive manner. In this system, users can select *usuls* from a list and record their performances after listening the *usul* patterns. After recording, users can use the tool to compare their rhythm performance with the *usul* template. In line with this purpose, the software has an interface to provide simple tools to the user as seen in Figure 1. In the following parts, the constraints of the system, the data used for the current version, modules of the software and the methods are described in detail. Additionally, some features of the system are compared with a system designed for a similar task.

2.1 Constraints

The system is designed in consideration with some environmental and hardware-related constraints. First, the noise in the room should be at a reasonable level so that the onset detection algorithm can provide reliable results. Secondly, the user should be able to interact with the computer by using the speakers and the microphone.

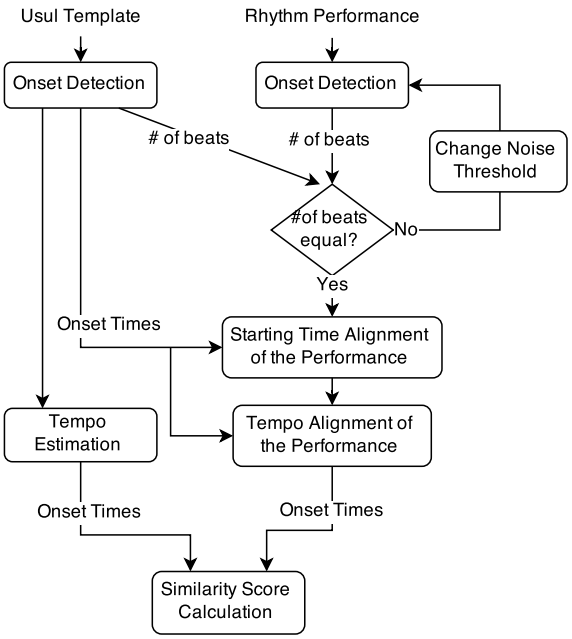

Figure 2: Workflow diagram of the tool.

2.2 Data

The system includes an audio file collection of *usul* patterns with symbolic representations as well as recordings from masters. These *usuls* are obtained from the Turkish makam music corpus, Uyar et al. [2014], considering their prevalence. Then, the most common *usuls* with different time signatures have been selected for preliminary tests. After creating a list of *usuls*, their audio files and symbolic representations have been prepared. The audio files are in Wav format and the symbolic data is prepared as MusicXML [2] scores. While generating the images from MusicXML files, Lilypond [3] is utilized because of its ease of use and compatibility with MusicXML files.

2.3 Modules and Methods

Mainly, there are two work branches in the system. One is for processing the reference rhythm and the other one is for analyzing the performance recording. The information retrieved from the reference rhythm is used for scoring the performance. The general view of the workflow is as seen in Figure 2. For audio signal processing and analysis, the Essentia [4] library is used. The tool is implemented in Python [5] programming language and the current version of the software runs on Ubuntu OS [6]. The system consists of several modules. These are the audio recorder/player, the audio signal visualizer and the audio processor modules.

The audio recorder/player module is implemented by using the PyAudio [7] library. This module enables the users to load a certain *usul* to practice, record a performance or load a file from the file system to calculate a score for it by taking the selected *usul* as the reference. Users can listen to the selected *usul* pattern, their own performances or the file loaded from the file system.

The audio signal visualizer part of the system is implemented by using the library provided by PyQtGraph [8]. By using the plotting functions of this library, the waveform of the reference *usul* is displayed on the interface. After the scoring of the performance or the file loaded from the file system, the onsets are displayed by vertical lines on the reference waveform.

The audio processing module is practically the core of the software. In this module there are functions for detecting onsets in an audio file, creating a binary vector as the onset signal, aligning the onsets detected from two audio files in terms of tempo and starting time, tempo estimation and score calculation processes.

For the onset detection, the *OnsetDetection* [9] function is used as the base with the parameter of *method="flux"*. This parameter sets the function to use the *Spectral Flux* detection method. The default values are used for the frame parameters as *frameSize=1024* and *hopSize=512*, as suggested in the documentation. In this study, the onsets detected by the *OnsetDetection* algorithm are selected by an automatic noise threshold setting. This additional feature makes us to be able to use the system in different environments with different noise levels. To do this, the algorithm changes the *noisethreshold* parameter at every iteration and runs until the number of onsets obtained from the performance recording is the same as the number of onsets in the reference recording.

Since the performance usually does not start as soon as the recording starts, it is necessary to align the starting times of the onsets of the performance and the reference. To accomplish this, a function is implemented to move the detected onset times of the performance providing that the first onset times of the reference and the performance are the same. In this procedure, the first onset time of the performance is subtracted from all onset times of the performance. Then, the first onset time of the reference recording is added to all onset times of the performance. For the tempo alignment, the tempo of the performance is assumed to be constant in order to achieve a consistent rhythmic pattern. With this idea in mind, the tempo alignment algorithm stretches or squeezes the onsets detected from the performance recording in a linear manner. The algorithm compares the durations of the reference and the performance recordings. Then it resizes the performance onset times by a constant ratio of D_R/D_P, where D_R and D_P denote the duration of the reference and the duration of the performance, respectively. The tempo estimation function [10] is used as it is provided in the Essentia library.

For the score calculation, Percival [2008] uses Mean-

[2] http://www.musicxml.com/
[3] http://www.lilypond.org/
[4] http://essentia.upf.edu
[5] https://www.python.org/
[6] http://www.ubuntu.com/
[7] https://github.com/bastibe/PyAudio
[8] http://pyqtgraph.org/
[9] http://essentia.upf.edu/documentation/reference/std_OnsetDetection.html
[10] http://essentia.upf.edu/documentation/reference/std_RhythmExtractor2013.html

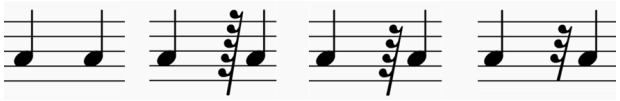

Figure 3: The beats for the perceptional test.

Squared-Error (MSE) to measure the correctness of a performance according to a reference rhythm. After the onsets of the performance are detected and aligned with the reference, the error for each onset is calculated in terms of frames. These errors are used to calculate the MSE. After the MSE is calculated, it is scaled with a constant value which is set after experiments, and the final score is provided by $(1 - MSE)$. However, in our study the criteria were decided to have constraints dependent on musical perception. For this purpose, a simple experiment was executed to understand how the short-timed notes are perceived at different tempo, ranging from 80 beats per minute (bpm) to 120. 5 subjects who are moderately experienced musicians were asked to differentiate the beats in Figure 3 by listening. Our initial listening tests show that the beat without rest and the beat with 128th rest were almost indistinguishable, the beat with 64th rest was recognizable around 80 bpm and the beat with 32nd rest was always recognized. The current scoring algorithm is based on this data. However, more detailed listening tests will be conducted in order to improve the reliability of the scoring algorithm. By considering the results of the initial listening tests, the scores for the different time intervals in terms of note lengths for individual onsets are set as in Table 1. In the scoring procedure, each onset of the performance recording is checked with the corresponding onset of the reference. The time difference between these onsets is calculated. Smaller time differences indicate more accurate performance timings. For example, if the time difference for an individual onset is less than the duration of a 128th note, it is scored as 100% correct. If it is between the duration of a 128th note and a 64th note, the onset is scored as 90% correct. After scoring the individual onsets, the average of the scores of all onsets of the performance except the first and the last ones is calculated to provide the overall score of the performance. An example is provided in Figure 6.

By using the functions described above, the procedure for calculating the score of a performance is as follows. First, an *usul* is selected from the rhythms list. Right after an *usul* is selected, the onsets and the tempo of the selection are calculated in the back-end and the waveform is displayed on the interface. Then, the user listens the *usul* and records a performance. After recording, user can listen the performance or directly click the *Calculate* button. When the button is clicked, the onset detection algorithm runs on the performance recording. The function takes the number of onsets of the reference as a parameter N_R and repeats the analysis by changing the *noisethreshold* until the number of detected onsets is equal to N_R. If the algorithm cannot find onsets providing this condition, it warns

Interval	$\leq$128th	64th	32nd	16th	8th	$\geq$4th
Score	1	0.9	0.8	0.3	0.2	0.1

Table 1: Scores for individual onsets according to intervals indicating the difference from corresponding onset of the reference.

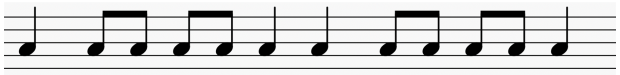

Figure 4: The beat pattern used for comparing our system's output with the Percival's.

the user by a text message and prompts to start again. After finding the onsets, the onset times of the performance and the reference are aligned to start at the same time. Then, the tempo of the performance is aligned with the reference. Once these operations are done, the score is calculated and presented to the user as the correctness score of the performance.

2.4 Review of the System

The most similar system provided for measuring the similarity between two rhythm patterns is the one provided in Percival [2008]. Therefore, we will compare our methodology with this one and discuss the pros and cons after summarizing his methodology briefly. In his method, the onsets are detected by calculating the root mean square (RMS) for each frame and comparing the RMS with a preset threshold. The frames with RMS higher than the threshold are considered as onsets. Once the onsets are detected and the start times of the performance is aligned with the reference, the MSE is calculated to provide the score of the performance. The first difference between this system and our system is the onset detection method. While the threshold for detecting the onset frames is constant in Percival's method, using the *spectral flux density* in our method enables users to use the system in rooms with different stable noise levels. In Percival's system, rhythm patterns in different tempos are treated by the same timing parameters. In the system we describe, the scores of the onsets are defined in terms of musical note lengths, which are dependent on the tempo. This provides measuring the correctness of a performance according to the perceptional criteria. Another difference is that in our methodology, the tempo of the user's performance does not affect scoring. This is because we mainly focus on learning the rhythm pattern, not playing it at the same tempo as the reference recording. To provide a basic comparison, the reference beat pattern (Figure 4) and the performance used to test Percival's method are regenerated. The data can be reached at GitHub [11] . For this data, Percival's method results a final grade of 68% (Figure 5), where our method provides an accuracy score of 85% (Figure 6).

[11] `https://github.com/burakuyar/rhythm_tool_test_dataset`

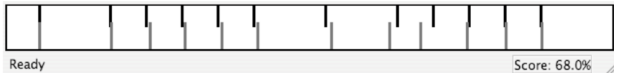

Figure 5: Output of Percival's system for the test data.

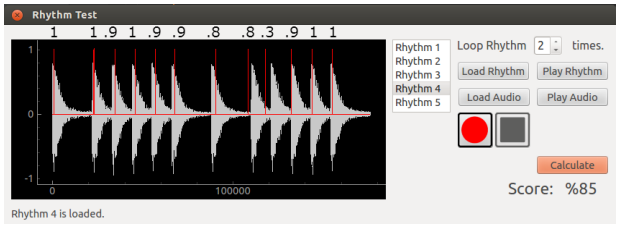

Figure 6: Output of our system for the test data.

Currently, the implementation of the system is completed. However, it is tested for limited number of scenarios. According to the experiments conducted until now, it is observed that the system can distinguish very good and very bad performances. As the next step, 10 different *usuls* will be selected and the performances of these *usuls* will be recorded from 5 different users. These performances will be scored by music experts and will be ordered considering these scores. Then, our algorithm will be optimized in order to provide the same ordering.

3. CONCLUSION AND FUTURE EXPECTATIONS

In this paper, an interactive tool for learning and practicing rhythmic structures, *usuls*, of Turkish makam music is described. In parallel to the methodology, a collection of symbolic beat scores and audio recordings describing the basic *usuls* are prepared. The tool presented analyzes two different recordings of a rhythmic pattern and presents an accuracy score with respect to the preset conditions. The current method for score calculation will be improved by conducting experiments and comparing the outputs with the outputs of the method provided by Percival [2008]. We hope that this study would stimulate research on novel interfaces and computational methodologies for the analysis and education of Turkish makam music.

4. ACKNOWLEDGMENTS

This work is supported by the European Research Council under the European Union's Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583).

5. REFERENCES

B. Bozkurt, R. Ayangil, and A. Holzapfel. Computational analysis of turkish makam music: Review of state-of-the-art and challenges. *Journal of New Music Research*, 43(1): 3–23, 2014.

Sam Ferguson. Learning musical instrument skills through interactive sonification. In *Proceedings of the 2006 conference on New interfaces for musical expression*, pages 384–389. IRCAMCentre Pompidou, 2006.

Mus2Okur. Multimedia encyclopedia of turkish music. URL `http://www.musiki.org/Mus2okur_en.aspx`.

G.K. Percival. Computer-assisted musical instrument tutoring with targeted exercises, university of victoria. 2008.

Usul-Velvele-Editor. Usul-velvele editor. URL `http://notist.org/usul_velvele_editoru.html`.

B. Uyar, H.S. Atlı, S. Şentürk, B. Bozkurt, and X. Serra. A corpus for computational research of turkish makam music. In *1st International Digital Libraries for Musicology Workshop, London*, pages 57–65, 2014.

Ching-Yu Wang and Ah-Fur Lai. Development of a mobile rhythm learning system based on digital game-based learning companion. In *Edutainment Technologies. Educational Games and Virtual Reality/Augmented Reality Applications*, pages 92–100. Springer, 2011.