BEAT-STATION: A REAL-TIME RHYTHM ANNOTATION SOFTWARE

Marius Miron, Fabien Gouyon, Matthew E.P. Davies
INESC TEC
Porto, Portugal
mmiron,mdavies,fgouyon@inescporto.pt

André Holzapfel
Boğaziçi University
Istanbul, Turkey
andre@rhythmos.org

ABSTRACT

This paper describes an open-source software for real-time rhythm annotation. The software integrates several modules for graphical user interface, user management across a network, tap recording, audio playing, midi interfacing and threading. It is a powerful tool for conducting listening tests, but can also be used for beat annotation of music or in a game setup. The parameters of this software, including the real-time constraints, are not pre-defined in the code but can be easily changed in a settings file. Finally, the framework used allows for scalability, as it was developed in openFrameworks. We show the usefulness of the software by applying it in a cross-cultural beat tapping experiment during the ISMIR 2012 conference. An analysis of the collected real-time annotations indicates that listeners encounter difficulties in synchronizing to music in presence of unfamiliar rhythmic structures and instrumental timbres.

1. INTRODUCTION

When analyzing a piece of music, an important initial step is to obtain an understanding of its temporal structure; to know: where boundaries between melodic phrases are, where the downbeats are located, where an instrument begins to play a note? Annotating such aspects of musical structure is a time-consuming task, but human annotations are often needed for the evaluation of automatic analysis approaches, or for obtaining insight into human perception of musical structure.

Tanghe et al. [1] described the process of annotating note onsets. They noted the absence of suitable annotation tools for their purposes. In their conclusion they underlined the importance of an easy-to-use, flexible, dedicated system for music annotation. Visual feedback, multi-layer annotations, connectivity to external user interfaces, flexible input and output, were found to be the key features of such a system.

Moreover, during the time-consuming and possibly boring process of manual annotation, the subjects can be motivated by designing the application as a game, providing the subjects with some goal to achieve. With one single exception [2], focused on collecting tags for music pieces, we are not aware of annotation systems which have been specically designed as a game.

Regarding annotation tools that integrate into existing software, Gouyon et al. [3] presented a semi-automatic rhythm annotation tool. This open-source tool is integrated into WaveSurfer [1] . It comprises a beat tracking algorithm, which sets the time values for the beats, and an annotation tool, which allows the editing of these time values. The annotator can stratify the beats into several metrical levels, however without the ability to simultaneously edit beats at several levels in real-time. MUCOSA [4] is a music content semantic annotator also based on WaveSurfer. The environment stores metadata at three different levels using an annotation client and a collection tagger. Additionally, it supports sharing annotations between various research groups via an administrative web interface. Li et al. [5] introduced editable audio and music segmentation layers into the Audacity editor [2] . While it offers important visual cues to the annotator, it is mainly focused on phrase segmentations, and not on rhythm or note onset annotation.

With respect to stand-alone annotation tools, Sonic Visualizer [6] is a music analysis and annotation platform developed in C++. In a similar way to MUCOSA, it structures information into several editable layers. It lacks the annotation sharing and user management capabilities of MUCOSA, but is highly expandable due to the integration of VAMP plug-ins which can provide automatically generated guides for annotation, *e.g.* by using an onset detection algorithm first, and then modifying its output by hand.

Most existing annotation tools were not designed for experimental conditions or games. For these systems, important settings such as the time sampling rate for user inputs are not available for editing. Other features, such as user management [4], or MIDI interfacing are present in some systems, but absent from others [5].

In this paper, we present *beatStation*, a software focused on (but not restricted to) rhythm annotation. The software can be applied in a game setup and provides functionalities for beat perception experiments. Compared to the other annotation software presented above, beatStation has multiple modules, and can be easily expanded as it was implemented in openFrameworks [3] . The application is open-source, flexible and scalable. It uses several popular openFrameworks add-ons for user and file management across network, audio playing, graphical user interface, process

[1] http://sourceforge.net/projects/wavesurfer/
[2] http://audacity.sourceforge.net
[3] http://www.openframeworks.cc/

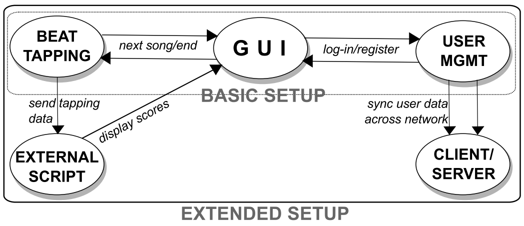


Figure 1: Software framework

threading, MIDI interfacing, and can be easily extended with more add-ons. Due to the flexibility of openFrameworks, it can be ported to any operating system platform, including mobile platforms.

Moreover, beatStation allows a user-friendly approach to conduct experiments by providing access to a range of parameters of the application from a simple settings file. Additionally, various stations can be simultaneously used in a client-server configuration, providing user management control. The software can be configured *e.g.* to annotate meter at several levels simultaneously in real-time, but also the annotation of melodic motif and phrase boundaries is possible. The data collected from the subjects is easily accessible and portable for analysis in the form of *xml*-files.

The beatStation was used during the ISMIR 2012 conference to gather sensorimotor synchronization [7] responses from subjects when asked to tap to Turkish and euro-genetic popular music. The experiment was designed as a game, in which attendants were motivated to tap the beat across a set of music stimuli to maximize their chances of winning a competition. This pilot experiment was a great success, and the software ran continuously for 5 days without incident. The results of the experiment provided us with some valuable insights into the ways listeners respond to musical styles they are not familiar with. These insights have already been used to design a dataset for a more thorough analysis of sensorimotor synchronization in Turkish music.

The remainder of the paper is structured as follows: In Section 2, we present the software framework with each module add-on. Section 3 presents details of our case study, and results are analyzed in Section 4. Finally, Section 5 concludes the paper.

2. SOFTWARE FRAMEWORK

As stated in [8] , openFrameworks introduced a framework which can be easily used by creative individuals, and also incorporates the strong assets of the C++ programming language. Since its introduction, a huge variety of applications has been developed and the community of developers has grown considerably. The framework grew as external code was added in the form of add-ons. Many of these add-ons can be combined in a single application, making it possible to connect various modules related to visuals, sensors and even algorithms for audio signal processing or sound synthesis.

The beatStation application has a basic setup, as depicted in Figure 1, including the core modules of beat tapping recording and storing, graphical user interface, and user management. Its extended setup includes external script calling and client-server communication.

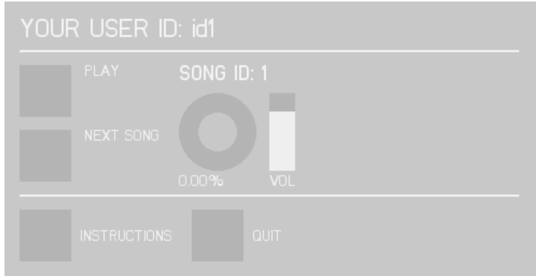


Figure 2: Screenshot of the *beatStation* interface

The add-ons used inside *beatStation* are as follows: *ofxUI* and *ofxTextSuite* for the graphical user interface, *ofxXMLSettings* for storing the data related to users and tapping, *ofxDirList* to load sound files or xml data files, *ofxTCPClient* and *ofxTCPServer* for tcp/ip communication. Additionally, the following pre-existing openFramework classes were used: *ofxMidi* for midi connectivity, *ofThread* for calling an external application to process the results, *ofSoundPlayer* for playing sounds.

2.1 Recording and storing real-time input

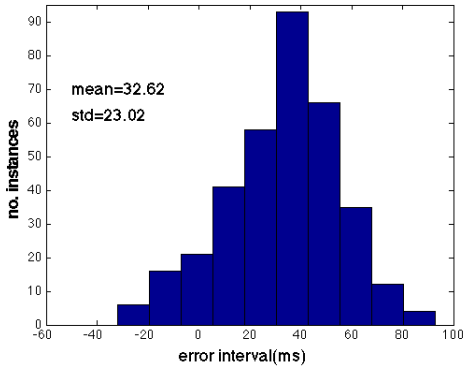
Our motivation was to create a software for real-time rhythm annotation, where the subject would be asked to tap to a musical stimulus. Hence, the tappings can capture various aspects such as beats, note onsets or downbeat structure – indeed any time-based information which can be meaningfully entered in real-time.

The subject is presented a set of songs, which is located in the `sounds` directory. Whether the songs are presented in a random or alphabetical order can be chosen by modifying the *randomFiles* variable in the `settings.xml` file. The number of times the subject can listen to a song can be set from the *noPlays* variable, in the same file. If a song is played more than once, then only the final set of annotations are retained. The subject's responses are saved in the `data` directory every time a user quits the interface or finishes annotating a song. Each user has an associated XML file where the tapping data is stored on the following structure:

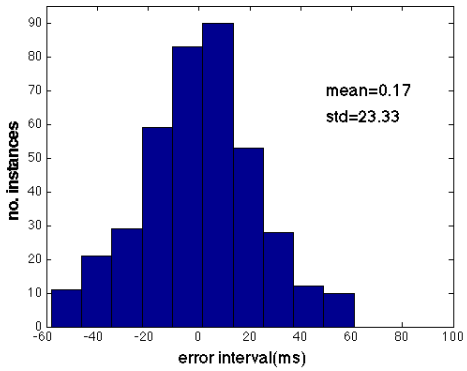
```
<tapping userID="14" currentSound="1">
    <songIndexes>2 </songIndexes>
    <song songID="2" tryNo="1">
        <fileName>file1.aiff</fileName>
        <transcription></transcription>
        <transcription2></transcription2>
    </song>
</tapping>
```

The unique ID of each user and the number of sounds tapped are stored at the top level. The lower levels stores the IDs of songs tapped, and for each one of those, the name of the song, and transcription values in milliseconds.

Annotation can be performed using the keyboard or a MIDI interface, *e.g.* drum pads for beat annotation. The settings related to MIDI are *midiPort*, *midiChannel*, *midiNote*, and *midiNote2*. When using the computer keyboard, the subjects can tap various metrical levels using the SPACE



(a) KEYBOARD



(b) MIDI INTERFACE

Figure 3: The histograms representing the delay between the onsets of the pulses in the signal and the time instances tapped on, using (a) the KEYBOARD, and (b) the MIDI INTERFACE

and TAB keys. The associated settings are *tapWithSpace*, and *tapWithTab*.

Regarding the real-time performance, the MIDI and the keyboard have associated event listener functions, which capture the events from the external interfaces. Checking for keyboard and MIDI interruptions with an event listener introduces a delay, which cannot be controlled using the application. This delay depends on the operating system, the MIDI drivers and the MIDI interface.

The unpredictable variability of the timing accuracy of general purpose computers was discussed in [9] . In this paper, we computed the tapping latency for our application when using a MIDI interface Roland Handsonic HPD-10 connected through a Digidesign USB sound card, and the keyboard, on a 2012 iMac computer. Five subjects were asked to tap along a sound example of 170 seconds duration, comprising equidistant pulses at a period of 500ms. Each subject tapped the sound example three times, first using the MIDI interface and then the keyboard.

In Figure 3, we depict a histogram of timing differences between the taps and the pulse locations in the signal. The delay using the keyboard is almost twice as large, but still lies within the tolerance demanded *e.g.* in beat tracking evaluation [10]. The standard deviation is similar for both interfaces and is in the range of the typically reported human variability [7] in sensorimotor synchronization.

2.2 Graphical User Interface

The graphical user interface (GUI) comprises several scenes for each stage of the application: user registration, user log-in, instructions, song annotation, and display of results. The user can move from one scene to another following a sequence of steps. First, the user has to register in the main GUI, where a basic description of the experiment is displayed. Then the application displays a set of instructions describing the task which is to be performed by the subject. After this, the tapping GUI is loaded. At this stage, the user can listen to songs and input their taps. A results page is displayed when the user finishes tapping all the songs or quits.

As depicted in Figure 2, the PLAY button is used to start playback of a song, after which the NEXT button can be used to move to the following sound example. All the buttons are disabled when a song is playing, except the QUIT button (provided the *canQuit* variable has been set in the settings file). The volume can be adjusted with the VOL slider. The instructions can be displayed with the INSTRUCTIONS button. The percentage played from a song can be seen in the rotary slider.

The software can be customized for various setups. The initial description can be edited in the `description.txt` file. In a similar way, the text for the instructions can be changed in the `instructions.txt` file.

The core of the GUI is the *ofxUI* add-on, which offers GUI scene management, widget layout, spacing, font loading, and several GUI widgets as buttons, input boxes, radio buttons etc. The absolute dimension of the GUI elements can be set by altering the value *itemDimGUI* in the settings file. Also, the application can be ran windowed or full screen by modifying the *fullscreen* variable in the same file. Users can be prevented to exit the application by setting a password in the field *passToExit*.

2.3 User Management

Before being presented with the audio stimuli, each subject is required to register in the application. For the prototype we only asked for a name, but more complex information could be gathered. Using the name, we generate a unique numeric ID and another ID based on the initials of the entered name. The latter can be used to log-in to the application and to resume the experiment at a future date, which is useful when the subject can perform the experiment in several parts. Additionally, we record the time and date of the registration. This data is stored in the `data/users.xml` file, with the following structure:

```
<users>
    <records>1</records>
    <maxID>0</maxID>
    <user>
        <ID>0</ID>
        <name>tt1</name>
        <fullname>tt</fullname>
        <date>2012-10-01</date>
    </user>
</users>
```

The number of records and the maximum ID in the XML is stored at the higher level, and the information concerning

each subject, ID, name, full name, registration date, on the lower level.

2.4 Client-Server Communication

Within our specific experiment setup, we wished to have several stations operating in parallel. Therefore we implemented a client-server architecture which allows different computers to communicate over a network. This architecture allows users to login into different machines (e.g. with different datasets) without the need to create a new user account each time.

Each client station sends a message to the server each time a user authenticates or creates an account. The server listens on a port for incoming messages from the client stations. If a message is received, the server checks if the user exists, and if it needs to, adds the user to the `users.xml`. Then, the server sends a message back to the client which tells the client if the user already exists in the database, or if it has just been added. Using this information from the server, the client allows the user to proceed using the application.

A station can either be determined as client or server by modifying the *isClient* variable in the settings file. If set to a client, the port to communicate is set using the *tcpPort* variable, along with the IP address of the server, *ipServer*. All annotations are stored on the local machine, regardless of whether the station is a client or server.

2.5 Calling External Scripts

In order to process annotation data on the fly, external scripts can be called depending on certain values in the settings file. These values are $launchScript$, which tells the application to launch a script or not, $scriptDirectory$, the relative path to the directory where the script resides, and $appToLaunchPath$, the full command line of the external program which calls the script. For instance, during our ISMIR experiment, we called an Octave [4] script to evaluate the recorded taps as follows:

```
<appToLaunchPath>
   /Applications/Octave.app/Contents
   /Resources/bin/octave -qf --quiet
   --eval "clear all; cd path;
   tapping2('xmlin','xmlout');exit;"
</appToLaunchPath> -
```

The $tapping2$ script reads the tapping data file, $xmlin$, and outputs a results file $xmlout$.

2.6 Designing for various setups

The beatStation was designed to function as a game during the ISMIR 2012 conference, having two stations in a client-server architecture. In this setup, the subjects were encouraged to annotate all songs, in order to achieve a better position in the high-score table. The high-score table was displayed on the log-in/registration page. We imposed a lower limit of five songs that someone has to tap in order to enter the high-score, a value that can be modified using the $minTaps$ variable. The version of the beatStation used at ISMIR 2012 can be downloaded from the Github repository [5].

In general experimental setups, there may be no requirement for a high-score table. For this reason, we disabled the script launching possibility in the final version, which automatically disables the high-score table. Subjects can not quit the application ($canQuit = 0$), and an additional page with a questionnaire can be launched in the application. The related code is commented in the source code, but can be adapted and activated. The more generic, non-ISMIR, version of the beatStation can be obtained from the Github repository [6].

The beatStation can be used for annotating the beats or any other events in music in real-time. As it is primarily designed for listening tests, it does not offer additional visual cues, and it can record high resolution annotations, on two (in our case: metrical) levels simultaneously (see Section 2.1). It doesn't allow editing of the annotations but the data can be easily exported to any other framework that allows editing (e.g. Sonic Visualiser). Functions to read the data into a Octave structure are provided with the software.

3. CASE STUDY

The goal of our experiments at ISMIR 2012 with two beatStations was, (apart from a real-world test of functionality), to collect data recording the sensorimotor synchronisation of listeners to music stimuli from two different music cultures; Turkish Makam music and euro-genetic popular music. By analysis of the recorded tapping sequences and their relation to annotated ground truth, we aim to address the question of whether high mutual agreement between tapping sequences (*i.e.* which arises when users tap the same way to the stimuli) is indicative of accurate tapping compared to the ground truth. By comparing these findings between the recorded taps and ground truth for stimuli from the two music cultures, we can obtain first indications into how difficult following the rhythm in Turkish music is for listeners who are unfamiliar with it, compared to generally more familiar euro-genetic popular music. We also investigate which musical properties caused problems in synchronizing with the stimuli.

When registering at a beatStation, a subject was asked to "listen to some short samples of music and to tap your perception of the most prominent pulse". Subjects were allowed to tap to a stimulus a second time, if the subject was not satisfied with their initial taps. The taps were recorded using the space bar of the keyboard, and high-quality headphones were used. The whole setup took place in the registration hall of the conference venue. While we are aware that this was not an ideal environment for conducting experiments of this kind, no subject reported the background ambient noise to be a problem. In order to motivate subjects to tap as many stimuli as possible, the beatStation was set up as a game, a kind of informal tapping competition. To that end, we used the script functionality (see Section 2.5) to compare the subjects taps with existing ground

[4] http://www.gnu.org/software/octave/

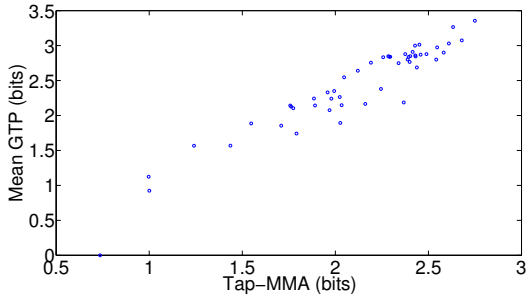
[5] http://github.com/nkundiushuti/beatStationISMIR/

[6] http://github.com/SMC-INESC/beatStation/

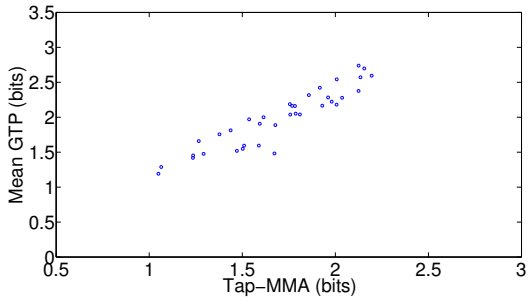
truth, and a high score table was generated with the scores of the "best" tappers.

In this paper, the Information Gain evaluation measure [10] was applied for the computation of all comparisons between beat annotations. In this evaluation measure, local timing deviations between beat annotations are summarized in a beat error histogram. The beat error histogram is characterized by a concentration of magnitudes in one or a few bins if annotations are strongly related, and by a flatter shape if the two annotations are unrelated. The deviation of this histogram from the uniform distribution, the so-called Information Gain, is measured using K-L divergence. This Information Gain measure has a range from 0 to 4.7 bits using the parameters described in [10], with 0 bits implying lack of any relation between two sequences, and higher values indicating a strong relation.

On the server beatStation, the dataset in [11] was used. This dataset, referred to as ISMIR2012, consists of 48 audio excerpts of 15s length each, which form part of the MillionSongSubset from the Million Song Dataset [12]. For the other beatStation, we selected 36 excerpts of 15s length, which we refer to as the MAKAM dataset. For all excerpts contained in the two datasets, ground truth annotations of the beat were performed by the authors of the paper. For excerpts in additive meters, *e.g.* 9/8, the pulsation at the metrical level of the 1/8 notes was annotated.



(a) ISMIR2012



(b) MAKAM

Figure 4: Scatter plots of the mean GTP over the Tap-MMA for, (a) the ISMIR2012, and (b) the MAKAM dataset

4. DATA ANALYSIS

Throughout the ISMIR2012 conference a total of 157 users registered at the beatStations. While this number reflects the high interest that the experiment attracted, many users only tapped a small number of files. While we didn't ask for explicit feedback from users, we suspect this may have been due to limited time available within the conference. To simplify our analysis, we retained only those users who tapped all files, which was done for each tapping station separately. By pure coincidence, we ended up with a set of 21 subjects for both the ISMIR 2012 and MAKAM dataset. While some enthusiastic subjects tapped to both datasets, the two sets of subjects were not identical.

We first compute the degree of mutual agreement between tapping sequences by comparing all pairs of annotations for a song using the Information Gain measure. We also computed the ground truth performance of each tapping sequence by comparing it with the ground truth annotation using the same measure. Then, we computed the Mean Mutual Agreement between all tapping sequences for a recording (Tap-MMA), and the mean Ground Truth Performance among all tappers for a specific song (mean GTP). Figure 4 shows a very high correlation between these two measurements, with the correlation coefficients being 0.951 for the ISMIR2012 and 0.930 for the MAKAM dataset. This shows that on both datasets mutual agreement in synchronization to the sound is strongly related to a high agreement with the annotated ground truth. While the correlations are high, both the Tap-MMA and the mean GTP of the taps are lower for the MAKAM data than for the ISMIR2012 data. On the MAKAM dataset we measure total means of 1.70 bits and 1.98 bits for Tap-MMA and mean GTP, respectively, on the ISMIR2012 dataset we obtain 2.43 and 2.11 bits. This is reflected by the scatter plot depicted in Figure 4a reaching further up to the right upper corner than the scatter plot for the MAKAM dataset depicted in Figure 4b.

Based on this analysis we infer that the MAKAM dataset represented a higher degree of difficulty for the sensorimotor synchronization than the ISMIR2012 dataset. This conclusion is supported by considering how often the subjects chose to tap a sample for a second time; For the MAKAM dataset 41% of the tapped annotations stem from a second attempt, while for the ISMIR2012 dataset the subjects only chose to tap a file again in 25% of the cases. This indicates that subjects were more confident that their spontaneous taps correlate with the musical meter for the familiar styles of euro-genetic popular music.

Finally, we use the outcome of the experiments to obtain conclusions about what traits influence synchronization for human listeners on the MAKAM dataset. First, those excerpts with $4/4$ time signatures were tapped more accurately compared to the ground truth, and wtih greated mutual agreement. Secondly, samples having either no percussive accompaniment or where the rhythmic accompaniment is played by Western drums cause problems for the listeners as well. While the former can be attributed to a larger *rubato* style in those performences, the latter reveals an interesting problem. In many recordings, Western drums, often in the form of electronic MIDI drums, are introduced to accompany rhythmic idioms that were originally not connected with them (such as the 9/8 *Aksak*). Even though the instrumental timbre of the drums introduce clear phenomenal cues for the beat, their acculturation seems to present problems to the listeners.

5. CONCLUSIONS

We presented a real-time annotation software, used for an experiment designed as a beat tapping game during the ISMIR 2012 conference. We introduced every layer of the used framework and its role in the overall architecture. Furthermore, we showed how basic features can be controlled from a simple settings file, in order to design various experimental setups. Our case study indicates that the software is stable having run across two machines for five continuous days during the ISMIR conference. We found that for our task the small latency in the system was not problematic and did not effect our ability to evaluate tap sequences. Through an objective comparison of input interfaces we confirmed that external MIDI hardware offers lower latency than using the SPACE bar on a standard computer keyboard.

The software can be used to capture in real time any kind of information that requires tapping: beats, onsets, rhythmic patters, structure segmentation. On the other hand, currently the data analysis is only offerred for comparing beats using Matlab/Octave. Moreover, the future development of the application can take various directions. For instance, an annotation software with off-line editing of annotations could incorporate visual cues and editing features, as well as additional interaction with the audio, such as adding pause and stop buttons. In an experimental setup for *e.g.* sensorimotor synchronization, the additional questionnaire page can be activated in the application and the parameters can be set according to the user's needs.

The results of our case study show that Turkish Makam music poses different challenges to listeners when asked to follow the beat of a piece. These challenges seem to be related to rhythmic structures as well as instrumental timbres. This motivates us to conduct a more formal comparative study of tapping behavior on additive and divisive rhythm which addresses the influence of cultural background of the listener and examines the various possible ways humans synchronize to these rhythms. Such a study is an important contribution to widening the focus of current research in MIR to include rhythms from other cultures, and to incorporate adequate cultural concepts of beat and rhythm into processing tools.

Finally, serialization of data and client-server communication should be adapted to gather tapping data from distributed devices onto a server. The basic game setup can be redesigned to allow users to compete against each other, or work in teams to achieve higher mutual agreement.

Acknowledgments

This work is partly supported by the European Research Council under the European Union's Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583), and by the ERDF – European Regional Development Fund through the (FCOM-01-0124-FEDER-014732) COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundacao para a Ciencia e a Tecnologia (Portuguese Foundation for Science and Technology) within project Shake-it, Grant PTDC/EAT-MMU/ 112255/2009. Part of this research was supported by the CASA project with reference PTDC/EIA-CCO/111050/2009 (FCT), and by the MAT project funded by ON.2. We would like to thank all participants of the test at ISMIR 2012.

6. REFERENCES

[1] K. Tanghe, M. Lesaffre, S. Degroeve, M. Leman, B. De Baets, and J. Martens, "Collecting ground truth annotations for drum detection in polyphonic music," in *Proc. ISMIR*, 2005, pp. 50–57.

[2] E. L. M. Law, L. V. Ahn, R. B. Dannenberg, and M. Crawford, "Tagatune: A game for music and sound annotation," in *Proc. ISMIR*, 2007, pp. 361–364.

[3] F. Gouyon, N. Wack, and S. Dixon, "An open source tool for semi-automatic rhythmic annotation," in *Proc. DAFx*, 2004, pp. 193–196.

[4] P. Herrera, Ò. Celma, J. Massaguer, P. Cano, E. Gómez, F. Gouyon, M. Koppenberger, D. Garcia, J. G. Mahedero, and N. Wack, "Mucosa a music content semantic annotator," in *Proc. ISMIR*, 2005, pp. 77–83.

[5] B. Li, J. A. Burgoyne, and I. Fujinaga, "Extending audacity for audio annotation," in *Proc. ISMIR*, 2006.

[6] C. Cannam, C. Landone, M. Sandler, and J. P. Bello, "The sonic visualiser: A visualisation platform for semantic descriptors from musical signals," in *Proc. ISMIR*, 2006.

[7] B. H. Repp, "Sensorimotor synchronization: a review of the tapping literature." *Psychonomic bulletin & review*, vol. 12, no. 6, pp. 969–992, Dec. 2005. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/16615317

[8] J. Noble, *Programming Interactivity: A Designer's Guide to Processing, Arduino, and Openframeworks*. O'Reilly Media, Incorporated, 2009.

[9] A. Wallace and G. Madison, "The timing accuracy of general purpose computers for experimentation and measurements in psychology and the life sciences," *The Open Psychology Journal*, 2010.

[10] M. E. P. Davies, N. Degara, and M. D. Plumbley, "Evaluation methods for musical audio beat tracking algorithms," Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06, 2009.

[11] J. R. Zapata, A. Holzapfel, M. E. P. Davies, J. Lobato Oliveira, and F. Gouyon, "Assigning a confidence threshold on automatic beat annotation in large datasets," in *Proc. ISMIR*, 2012, pp. 157–162.

[12] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc. ISMIR*, 2011, pp. 591–596.