Proceedings of the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music, July 23-28, 2012, Thessaloniki, Greece
Cambouropoulos E., Tsougras C., Mavromatis P., Pastiadis K. (Editors)

Tap-It: An iOS App for Sensori-Motor Synchronization Experiments

Hyung-Suk Kim,[*1] Blair Kaneshiro,[*2] Jonathan Berger[*3]
[*]*Center for Computer Research in Music and Acoustics, Stanford University, Stanford, CA, U.S.A.*
[1]hskim08@stanford.edu, [2]blairbo@ccrma.stanford.edu, [3]brg@ccrma.stanford.edu

ABSTRACT

This paper describes Tap-It, an iOS application for sensori-motor synchronization (SMS) experiments. Tap-It plays an audio file while simultaneously collecting time-locked tapped responses to the audio. The main features of Tap-It compared to desktop-based SMS apparatuses are mobility, high-precision timing, a touchscreen interface, and online distribution. Tap-It records both the time stamp of the tap time from the touchscreen, as well as the sound of the tapping, recorded from the microphone of the device. We provide an overview of the use of the application, from setting up an experiment to collecting and analyzing the output data. We analyze the latencies of both types of output data and assess the errors of each. We also discuss implications of the application for mobile devices. The application is available free of charge through the Apple App Store, and the source code is also readily available.

I. BACKGROUND

Sensorimotor synchronization (SMS) research, especially research involving tapping tasks, has been a main topic in the study of rhythm perception, and there is an abundance of literature in this area (Clarke, 1998; Repp, 2005). An important component of SMS research is the apparatus used for obtaining measurements. Various apparatuses and interfaces have been implemented for SMS experiments over the years, from horizontal-drum kymographs (Stevens, 1886), to telegraph keys with a teleprinter machine (Klemmer, 1957; Wing, 1973), to a microswitch with rolling ball (Keele, 1985). The introduction of the personal computer reduced the need for customized hardware and enabled use of various input devices, including the keyboard, mouse, and various MIDI controllers. Software packages such as FTAP reduced the overhead of building customized software. (Finney, 2001)

Despite these advances in measurement devices, most tapping research must be conducted in laboratory settings due to equipment constraints. In addition, tapped responses still suffer from tactile feedback (e.g. key or mouse click) and noise (e.g. microphone), as well as latency and jitter in the time stamps. Effects of these factors tend not to be quantified or reported in publications. Variability of response collection methods across research groups adversely affects comparability of data and repeatability of experiments.

With the goal of addressing these limitations, we developed Tap-It – a free, open-source iOS mobile application that plays audio files while simultaneously recording time-locked tap responses using both the touchscreen and the microphone of the mobile device. We believe this application will be of interest and relevance to the SMS community as a tool for tap-based empirical research, and that it can expand the range of possible experimental paradigms. This paper will focus on Tap-It rather than the evaluation of existing apparatuses for collecting tapped responses.

II. DEVELOPMENT

Our goal was to address two principal challenges of existing tools – timing inaccuracies resulting from interface mechanics, and the immobility of the collection device. With this in mind we focused on two goals, high-precision timing and mobility. For the former we implemented a low latency method that ensured synchrony between tap-time and the audio stimulus with a touchscreen interface free of mechanical noise. To the latter, we added the goal of offering an application that would be not only mobile, but also freely available and easily accessible to the research community at large.

A. Mobility

As mentioned in the previous section, response collection devices historically have not been mobile. Using a mobile device opens the possibility of tapping experiments while in motion, e.g. walking or dancing. It also enables experiments outside of the controlled lab environment.

In recent years, mobile devices have come to provide enough computational processing power to be used for music synthesis and interaction (Wang, 2008). They introduce a tangible, interactive experience compared to that of the traditional desktop configuration. Current features of mobile devices (e.g. multi-touch, accelerometer, and gyroscope) enable novel methods of interaction and allow for collection of new types of data. Mobility and connectivity of mobile devices also enable new methods of transmitting, storing, and retrieving data.

In terms of the type of mobile device for application development, we chose iOS devices due to low audio latency, abundance of devices (e.g., iPhones and iPod Touches), and the ease of distribution of the application via the Apple App Store.

B. High-Precision Timing

In order to achieve high-precision timing, Tap-It was programmed using the MoMu Toolkit (Bryan, 2010) and STK:MoMu release for audio playback. Tap-It utilizes the audio playback timer, not the device (CPU) clock, for time stamps to ensure strict synchronization of tap times with the audio playback. Performance analysis is presented in section IV (Data Analysis).

C. Flat Tapping Surface

Collecting tapped responses using mechanical devices such as the computer keyboard or mouse introduces mechanical and tactile latencies, which may adversely affect the behaviour of the experiment participant. The touchscreen of the iOS device, being flat, removes these latencies and feedback.

D. Distribution and Broad-Range Data Collection

iOS devices have come to be ubiquitous in many places. Thus, the ease and accessibility of downloading and installing

applications to such devices facilitates experiments that can now be run remotely and over a broad geographical range.

III. SPECIFICATIONS

We briefly describe the procedure for downloading the application, and then setting up and executing an experiment using Tap-It. Following this, we describe the format of the response data. An in-depth description of the following sections can be found on the Tap-It webpage.[1]

A. Terminology

We will be using the following terms. By *task*, we mean a tapping task with one full audio file playback. A *session* is a series of tasks, usually by one subject. An *experiment* is a series of sessions using the same set of audio files with the same setup file (this point is explained in greater detail in the following subsection). An *experimenter* is the person setting up the experiment. A *subject* is the person who performs the tasks and whose tapped responses are being collected.

B. Description of Use

Tap-It can be downloaded and installed free of charge to an iOS device via Apple App Store.[2] Figure 1 shows the informational screen of the application.

Figure 1. Information screen of Tap-It in the Apple App Store.

1) Setting up an experiment. In order to set up an experiment, the following files are required:

- Audio files for testing (.wav format)
- Task description file (trackList.xml)

Currently, audio files must be in .wav format to be loaded by the experimenter. The tracklist.xml file contains the names of all .wav files to be presented in the experiment. This document is simple to edit and we provide a template as part of the online documentation.[3] Figure 2 shows the template trackList.xml file, which is initialized with a list to play .wav files titled "theFirstFile.wav" and "theSecondFile.wav".

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xml>
    <filename>theFirstFile.wav</filename>
    <filename>theSecondFile.wav</filename>
</xml>
```

Figure 2. Example trackList.xml file. More files can be added by adding additional <filename> </filename> lines and enclosing the file name between the two tags.

Experiment files are uploaded to the mobile device using the iTunes Documents browser. The application will deliver an error message if the files are not configured correctly or cannot be loaded. By default, audio files will be presented in the order they are listed in the .xml file. If randomization is desired, the application also includes a setting for randomizing track order, which can be accessed in the Tap-It section of Settings menu in the iOS device.

2) Running an experiment. Once the experiment files have been successfully uploaded to the device, the application is ready to be used for data collection. The interface is shown in Figure 3.

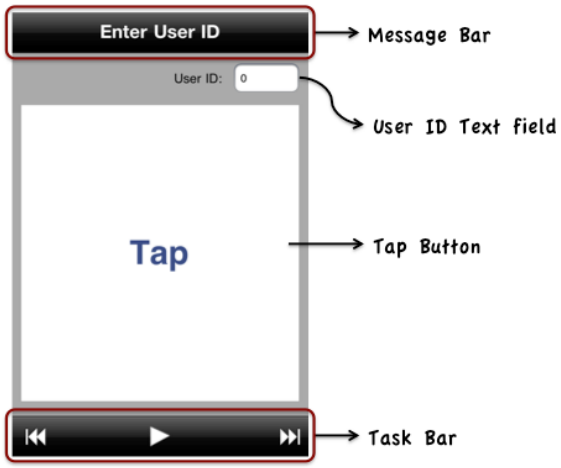

Figure 3. Main interface of Tap-It. The user may tap anywhere in the main white box. Touching the "Enter User ID" brings up a pop-up window for entering the ID; the task bar at the bottom is used to play audio files, and to move backward or forward through the track list.

The experimenter initiates an experiment session by entering a numeric user ID and pressing the "Done" button that appears in the pop-up window. Following this, the application will cycle through the audio files listed in the xml playlist. For each audio file, the subject presses the play button in the bottom task bar to begin audio playback and start recording the taps. At the end of each audio track, the application will automatically load the next audio file, waiting for the subject to press play to continue.

[1] URL: https://ccrma.stanford.edu/~hskim08/TapIt/

[2] URL: http://itunes.apple.com/mu/app/tap-it/id456418861?mt=8 or search for "Soundscape Studio".

[3] https://ccrma.stanford.edu/~hskim08/TapIt/user.html

C. Data Description

Tap-It creates two files for every audio file that is played through the device:

- [userID]_[track#].txt
- [userID]_[track#].wav

The .txt file contains the time stamps created by tapping the screen, with the time stamps in seconds since the beginning of the audio playback. The .wav file is an audio file of the microphone recording. There are pros and cons to each format.

The .txt file is easy to read into any software or script, and is unlikely to contain false inputs. However, due to the limitations of the device, there is a time latency of approximately 15ms from a tap on the screen to the actual time stamp (see next section).

To compensate for the latency, Tap-It also saves the microphone recording. This recording is strictly synchronized to the playback audio, and the taps will be exact. However, due to the fact that this is a direct microphone input, the signal is prone to ambient noise, and the microphone sensitivity varies from device to device. If this is the preferred data format, subjects should wear headphones so that the sound of the audio file being played out of the device does not overwhelm the sound of the taps. In addition, silent taps may not be easy to pick up through the built-in microphone.

In the current implementation of Tap-It, the .txt file is created only when a task ends, i.e. the current audio finishes playing. Therefore, the task must be completed in order for the time stamp file to be created. In contrast, the tap audio recordings are saved as the task is being performed.

In addition to the files created for each audio file that is played, the following file is created for each user ID:

- [userID]_play_order.txt

This .txt file contains the actual play order for the experiment session, and is particularly useful when the randomization setting is enabled.

All data files are saved in the "data" folder within the Tap-It Documents, visible in the iTunes Apps tab. If multiple sessions are conducted under the same user ID, existing versions of the output files will be overwritten by the most recent version.

IV. DATA ANALYSIS

In this section we provide an overview of methods to analyse the data files and use these methods to assess the timing precision of Tap-It. Our analyses are performed using MATLAB.

A. Reading Data Files

As shown in Figure 4, the time stamps in the .txt files are six-digit decimal numbers separated with line breaks. This file format is easy to load into software programs. Extracting the time stamps from the output .wav files, however, is not so trivial, as the tap times must be inferred by finding peaks in the audio. Example MATLAB files for loading the output data files from experiment sessions and analysing the results are provided in the Data Analysis section of the online Tap-It documentation.[4] Figures 5 and 6 were generated using those files.

```
0.835918
1.114558
1.578957
2.043356
2.693515
3.157914
3.715193
```

Figure 4. Example of time stamps in .txt files. Time stamps are recorded in seconds.

B. Latency Analysis

We define latency as the time from the actual tap of the device to the time it is observed and recorded. Latency of a tap apparatus can be caused at any point of the response collection process. For the time stamps in the .txt file, possible sources of latency are, the time the device takes to recognize the tap event from the touchscreen, the time it takes for the event to be notified to the application, and the time for the app to process the event. For the tap microphone recording, the main source of latency is a constant audio buffer swapping.

Another artefact closely related to latency is jitter. Jitter is caused by the variation of latency. Since any latency for audio will be related to buffer swapping, there is a negligible amount of jitter (< 1ms) for the microphone recording. However, the latency of the time stamps varies.

We have already mentioned that the touchscreen, compared to a keyboard or mouse button, has no moving parts, and thus prevents tactile feedback for the user while tapping.

In Tap-It, the audio input and output are handled on the same device, and in the same callback function. This reduces timing jitter in the difference between the microphone signal and the playback audio. Thus we view the microphone signal as a more accurate representation of the actual tap event, and treat it as the ground truth of the tap time, keeping in mind the limitation of the efficacy of the peak-finding algorithm.

We assessed performance of Tap-It timing using tap data collected with both an iPhone 3GS and an iPod Touch 4G. Tap onset times extracted from the microphone input were compared to the time stamps in the .txt file. Latency was measured by comparing the peaks in the microphone recording (.wav) and the time stamps (.txt) generated from the same task (Figure 5). The tap-to-time stamp latency was observed to be approximately 15 ms, with a standard deviation of approximately 5 ms, depending upon the load and/or state of the device (Figure 6) Differences between devices were negligible.

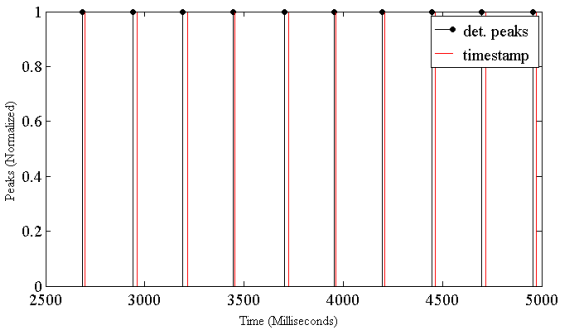

Figure 5. Plot of peaks detected from the microphone input and the time stamp data. Detected peaks from the .wav file are shown in black, and the time stamps from the .txt file are shown in red.

[4] https://ccrma.stanford.edu/~hskim08/TapIt/data.html

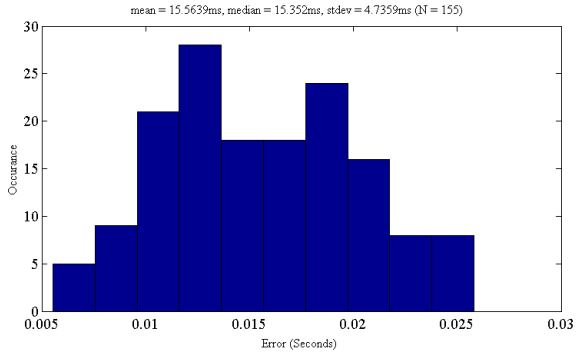

Figure 6. Histogram of error between the time stamps and the detected peaks from the audio data. The mean of the errors is 15.56ms with a standard deviation of 4.74ms. (N = 155)

V. IMPLICATIONS

Tap-It is ready for use in experiments. We now identify the advantages and limitations of the application in its current state.

A. Advantages

The mobility of the apparatus makes possible a wide range of experiment settings for studying sensorimotor responses, on two levels. On the first level, experiments can now be conducted outside of the laboratory setting, and tapped responses can be collected while the participant is engaged in other activities such as walking or dancing.

On a broader level, the fact that the application can be installed on any number of iOS devices can enables advantages ranging from faster collection of data in the same location, to the ability for data to be collected across great geographical distances, and for these data to be easily comparable. In addition, performing data collection using this application implies increased reproducibility of results. This increased scope of distribution and standardization also has implications toward crowdsourcing.

B. Limitations

We acknowledge limitations to the application in its current state. File management currently does not leverage the networked capability of iOS devices, and must be performed manually by physically connecting the device to a computer. Experiment files must be uploaded to, and downloaded from, the device using iTunes (i.e. they cannot currently be downloaded directly to the device or uploaded from the device to a server). Data analysis must additionally be performed offline. At this time, audio files must be in .wav format, and the application is developed only for iOS devices.

As mentioned previously, participants must wear headphones if the experimenter wishes to utilize the microphone recording of the tap times. Additionally, in this case the participant must tap with enough force that the sound of the tap is loud enough to be captured by the microphone. This may not be an easy or ideal activity for all tapping tasks.

Finally, the features we chose to implement in the application prevented development of other features. The flat touchscreen precludes any tactile feedback from tapping. Audio feedback (such as a mouse click) is also eliminated, aside from the sound of the tap on the touchscreen.

Improved file management, support for additional audio file formats, and optional visual and audio feedback may be implemented in a future version of the application.

VI. CONCLUSION

We identified a need, in SMS research, for a data-collection apparatus that embodied features of mobility, high-precision timing, flat tapping surface, and easy distribution for broad-range data collection. We implemented all of these functionalities in Tap-It, a freely available application designed for use in iOS devices. The application is available through the Apple App Store and is ready for use in tapping experiment.

We have discussed the specifications of the application as well as its use in setting up experiments, running experiments, and analysing the collected data. Timing performance issues of latency and jitter have been assessed. We have identified current advantages and limitations of using the device.

REFERENCES

Bryan, N. J., Herrera, J., Oh, J., & Wang, G. (2010). MoMu: A mobile music toolkit. In K. Beilharz, A. Johnston, S. Ferguson, & A. Y.-C. Chen (Eds.), *Proceedings of the 2010 Conference on New Interfaces for Musical Expression (NIME 2010)*. Sydney: University of Technology Sydney.

Clarke, E. F. (1998). Rhythm and timing in music. In D. Deutsch (Ed.), *The Psychology of Music* (2nd ed.). San Diego: Academic Press.

Finney, S. A. (2001). FTAP: A Linux-based program for tapping and music experiments. *Behavior Research Methods, Instruments, and Computers, 33*, 65-72.

Keele, S. W., Pokorny, R. A., Corcos, D. M., & Ivry, R. (1985). Do perception and motor production share common timing mechanisms: A correctional analysis. *Acta Psychologica, 60*, 173-191.

Klemmer, E. T. (1957). Rhythmic disturbances in a simple visual-motor task. *American Journal of Psychology, 70*, 56-63.

Repp, B. H. (2005). Sensorimotor synchronization: A review of the tapping literature. *Psychonomic Bulletin & Review, 12*, 969-992.

Stevens, L. T. (1886). On the time-sense. *Mind, 11*, 393-404.

Wang, G., Essl, G., & Penttinen, H. (2008). Do mobile phones dream of electric orchestras? In *Proceedings of the International Computer Music Conference (ICMC)*. Belfast.

Wing, A. M., & Kristofferson, A. B. (1973). Response delays and the timing of discrete motor responses. *Perception & Psychophysics, 14*, 5-12.