Data Conversion Tool For Tobii Pro Glasses 2 Live Data Files

- From .json to .txt

Andreas Wulff-Jensen
Dept. Architecture, Design and Media technology
Aalborg University Copenhagen
Copenhagen Denmark
awj@create.aau.dk

***Abstract*— The data gathered through the Tobii Pro Glasses 2 is saved in a .json file called livedata.json. This format is convenient for the Tobii analysis software, but for any other analysis software packages it can be rather troublesome as the software packages do not know how to interpret the .json[1] file. This paper will describe a small Java program which is capable of converting the .json file to a comma separated .txt file, which is ready to be used in either Ogama[2], Matlab[3] or another eye-tracking analysis software.**

Keywords— tobii pro glasses 2; Json: data conversion; java; eye-tracking: program;

I. INTRODUCTION

Tobii pro glasses 2 is a mobile flexible eye-tracking device, which is not bound to the screen based medium. This feature makes it appropriate for eye-tracking experiments towards any other kinds of stimuli in space [1]**.** The eye-tracker set-up procedure is fast and controlled with a few commands through the Tobii Pro Glasses Controller [2]**.** Through the controller, you can record your subjects, but nothing more. Afterwards or in the meantime you can get access to the data through the Tobii Pro Glasses 2 SDK [3] or the Tobii Pro Lab [4].

The Tobii Pro Glasses 2 SDK provides good overview of how to get access to the data in the glasses and an overview of the data structure embedded on the memory cards of the glasses. Moreover, if the user does not have the time or resources to research the API to create a custom made analysis or data acquisition software it may provoke more frustration than help. The Tobii Pro Lab is not open source but based on the documentation it seems like an efficient tool. With the tool users of the Tobii Glasses will quickly get access and analyze the data, but without it, the SDK is the only option. This observation laid the foundation to create the data converting program based on the SDK which will be described more throughout, through the rest of this paper. It provides the data for offline analysis for the program the user desires to work with.

The following sections will provide you with info about (II) how the .json structure looks like. (III) how the program is created. (IV) showcase how to use the program. (V) Conclusion of the paper.


```
{"ts":306066791,"s":0,"gp":[0.5504,0.0171],"l":106182}
{"ts":306066791,"s":0,"gp3":[-93.95,672.86,1419.67]}
{"ts":306086789,"s":0,"pc":[-34.38,-29.80,-34.94],"eye":"right"}
{"ts":306086789,"s":0,"pd":5.09,"eye":"right"}
{"ts":306086789,"s":0,"gd":[-0.0751,0.4771,0.8756],"eye":"right"
}
{"ts":306010700,"s":0,"ac":[-0.275,-9.857,1.687]}
{"ts":306023717,"s":0,"gy":[-8.050,-4.310,3.078]}
```


Fig. 1 shows how the data looks inside the livedata.json file

[1] More about .json go to this website http://www.json.org/
[2] Open source Eye-tracking analysis software
http://www.ogama.net/
[3] Software for statistical analysis
https://www.mathworks.com/products/matlab.html

II. .JSON STRUCTURE

Before converting the data it is important to know how the data inside the .json file looks like and how to interpret it. Based on Fig. 1 a sample of the data the Tobii Glasses provides the user with can be seen. Initially this can look foreign, but by breaking it down row by row the structure and the logic of the file starts revealing itself.

In the beginning of each row "ts" is written. This is a time stamp and can be used to synchronize the different data streams together [3, p. 28]. However, it is rather unhelpful outside the Tobii realm as other devices and data streams do not use the same kind of time coding.

The next tag is "s" which is for status, if it at zero nothing is wrong with the data, but if any other number something might be wrong [3, p. 25].

From here the different lines differs significantly and contains data of different kinds of data. "gp" being gaze position in camera view space. Meaning the position is based on what the camera in front sees. The data ranges from 0.0 – 1.0. where [0.0, 0.0] is in the upper left corner and [1.0, 1.0] is in the lower right corner [3, p. 27]. "gp3" is a little different gaze position. It is in 3D where 0.0.0 is at the camera of the glasses and the measurement is in mm [3, p. 28]. "pc" is pupil center. This center is in 3D and it is relative to the camera in the middle of the glasses. It denotes whether it is referring to left or right eye by "eye":"right"/"eye":"left". The measurement of this data is in mm [3, p. 26]. "pd" stands for pupil dilation, as the previous measure this denotes which eye it is referring to as well as being measured in mm [3, p. 26]."gd" is gaze direction is a unit vector, which has the pupil center as the center point, this measurement distinguishes between the two eyes in the same way as the previous measurements [3, p. 26].

The last two measures are "ac" and "gy" these stands for accelerometer and gyroscope. The unit for the accelerometer is meter per second squared and the unit for the gyroscope is degrees per second [3, pp. 27–28].

With the basic understanding of the .json file we can progress to create the program which can convert this file to a comma separated file, which can easily be imported in any data analysis or statistics software.

III. SOFTWARE

The software is programmed in Java and contains of two different parts, an User Interface (UI) frontend part and a .json interpreter part.
The UI is based on the native Java GUI library javax.swing which is a simple framework for constructing panels, buttons, text boxes, texts, etc.[5]. This has been used to communicate with the user. In this program they tell what the user can expect from the output files, which file the user should tell the program to read, where to input many seconds the tracking session has lasted and where the user wants the output file to be placed. Last, a JButton is created, if it is pressed the conversion procedure begins.

This procedure starts by loading in the first non-investigated json object from the .json file, through the gson Jsonreader [6]. A json object is one of the sections encapsulated in curly brackets found in fig. 1. In each of these objects the program looks for "s" to get the status of the device, thus whether it can use the data or not. Next it asks for one of the acronyms for the different data types, when it is found it, it adds the found value to the corresponding arraylist. In case the data can come from either left or right eye the program also reads that information before adding the value to an arraylist. When it has finished with one json object the program goes back to the beginning to load the next object. Since the number of objects is uncertain for the program, as it only reads one object at a time the program runs this opertation 1000000 times to ensure that it has been through all the objects in the dataset. In case the program reached the end of the document it stops reading.

When all the data has been stored, the arraylist with all the time stamps are sorted and all duplicates are deleted from it. The length of the new time array will be used together with the user input about the length of the recordings in seconds to create new timestamps which starts from 0.0 seconds and end with the amount of seconds the recording as lasted. The new time stamps will be calculated the following way new time = time stamp index in arraylist /(length of the time arraylist / seconds the recording has lasted).

In case the user do not know the length of the recording the program will use the default update rate from the device which is 50hz for the eye-tracking, 100hz for the accelerometer and 95hz for the gyroscope [1]**.** by this update rate the equation is new time = time stamp index / hz.

So far, the program has loaded all the data, it only needs to output it into .txt files. Three of such are created one for the eye-tracking, one for the accelerometer and one for the gyroscope, because they have different sample rates. The lengths of the time arrays are then used for the maximum index of how many times the other data arrays needs to be visited to extract and write the data to the .txt file. This results in comma separated documents, where the following structure is always prevailing.
For the eye-tracking file: Time, Pupil Diameter, Gaze Position, Gaze position in 3D, Gaze Direction: three coordinates per eye, pupil center: three coordinates per eye.
For the accelerometer file: Time, three axis of accelerometer.
For the Gyroscope file: Time, three axis of Gyroscope.

The software is created the following section will go through how to work with it.

IV. PROGRAM USAGE

The first thing to do before opening the program is to locate the livedata.json file and a fullstream.mp4 file. The latter contains the video recording and thereby the exact amount of time the recording has lasted.

On the memory card of the Tobii the files can be found through the following path. Root folder → projects → XXX(project name) → recordings → XXX(recording name) → segments → #(segment number) → livedata.json or fullstream.mp4.

The file path of the livedata.json file is written to the program as shown in Fig. 2 . To the right from that text field is a slot ready for the length of the recording in seconds. It is recommended to open the video file fullstream.mp4 in a video editing software to get the exact number of seconds and additional amount of frames. Thereby achieving higher temporal precision with both seconds and milliseconds noted. If you do not have a program which can see the number of frames in the file you are better off by leaving this text field blank, which then tells the program to use the default sampling rate. The lowest text field in the program is used to allocate where on your computer you want the output .txt files to be. The investigated program can be found through the following GitHub page (https://github.com/anwul4/Tobii_JsonToTxtConversionTool.git)

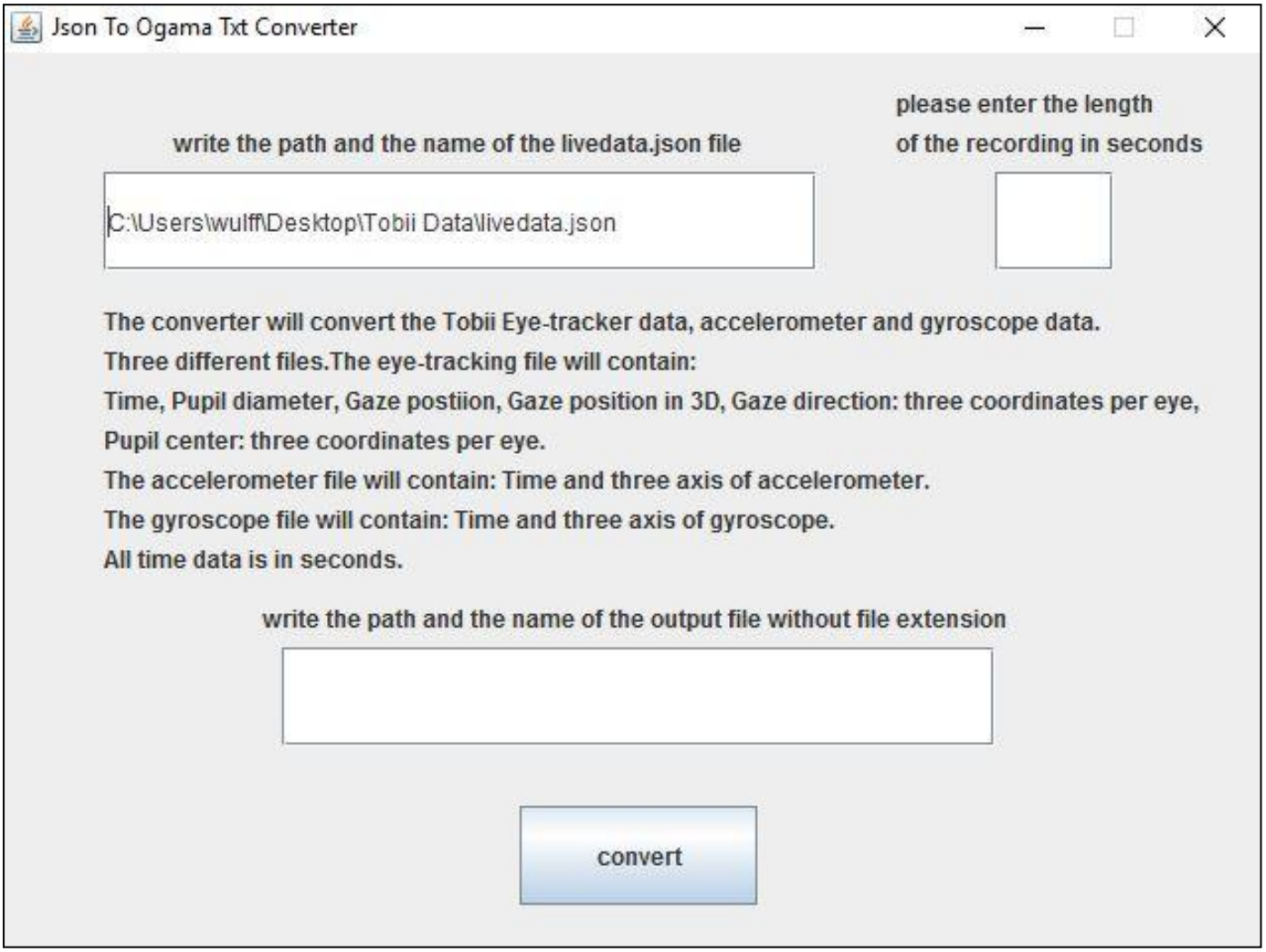

Fig. 2 shows how the programs look and what the software will output to the user.

V. Conclusion

In this paper we have visited the creation of a program which can convert the tobii eye-tracker livedata.json file to a sensible comma separated .txt file, which any analysis software can read and interpret. Through Java and the tobii SDK a simple UI and backend converting program was created, which is ready to be used by any interested party. The software and source code can be found on the following github page (https://github.com/anwul4/Tobii_JsonToTxtConversionTool.git)

References

[1] TobiiAB, 'Tobii Pro Glasses 2 Product Description', Stockholm, 2015.

[2] TobiiAB, 'Tobii Pro Glasses Controller'. TobiiAB, Stockholm, p. 1, 2016.

[3] TobiiAB, 'Tobi Pro Glasses 2 API Developer's Guide v.1.12.2'. TobiiAB, Stockholm, p. 40, 2016.

[4] TobiiAB, 'Tobii Pro Lab'. Stockholm, p. 1, 2016.

[5] Oracle, 'Package javax.swing', *Oracle*, 2016. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html. [Accessed: 01-Feb-2017].

[6] Google, 'gson', *gitHub*, 2017. [Online]. Available: https://github.com/google/gson. [Accessed: 02-Feb-2017].