About

This matlab utility can parse recording files from the Tobii Glasses 2 as stored on the SD card (no need for intervening Tobii software). Once parsed, it will display this data in a MATLAB GUI. Optional eye videos are also supported. In this GUI, sync events can be shown, and data can be optionally annotated (e.g. as slow and fast phase intervals) by hand or by a classifier algorithm, or such annotations can be loaded from file.

Cite as: Niehorster, D.C., Hessels, R.S., and Benjamins, J.S. (in prep). GlassesViewer: Open-source software for viewing and analyzing data from the Tobii Pro Glasses 2 eye tracker.

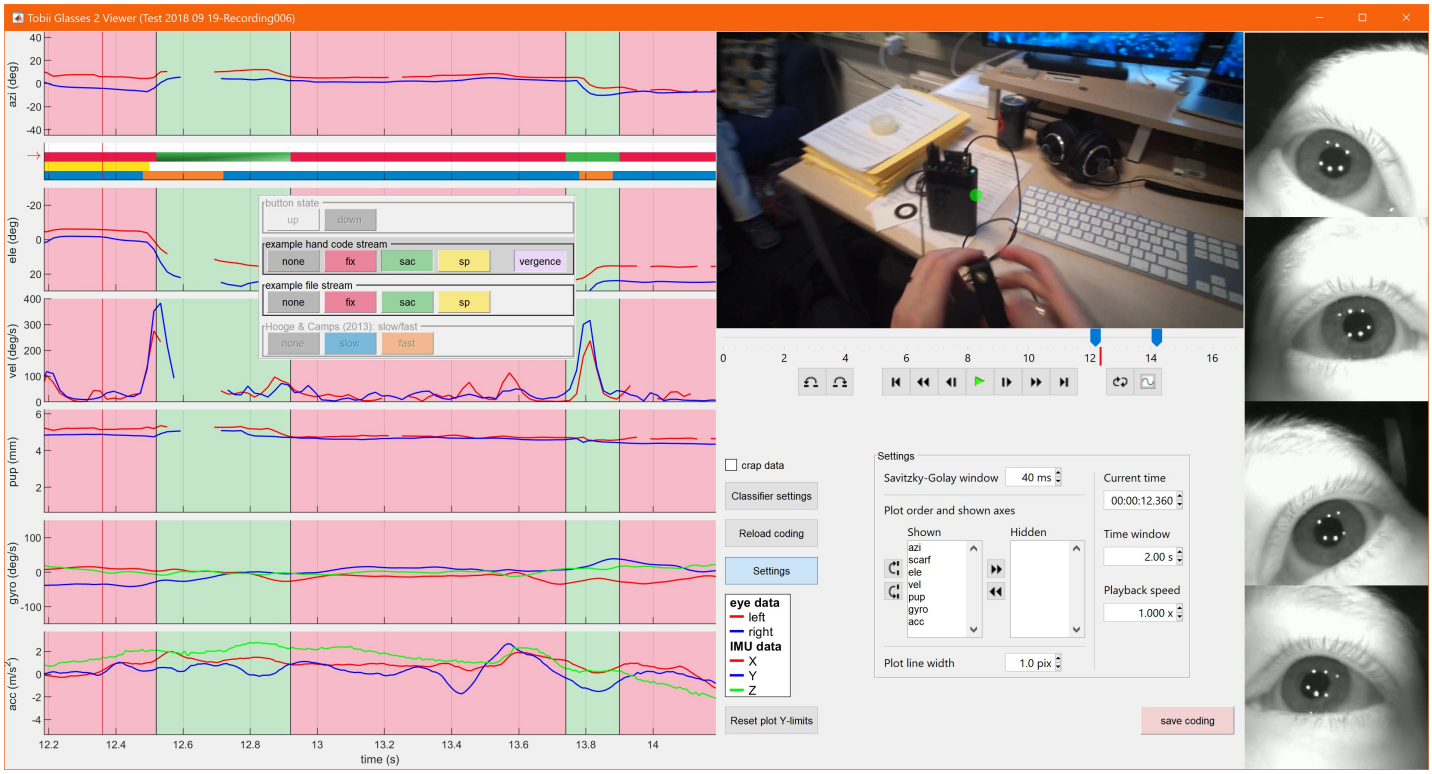
GlassesViewer offers integration with the GazeCode manual mapping tool for easily and efficiently mapping participant gaze or other event episodes to the visual stimulus. This integration is demoed in the manual, see the GazeCode repository for further details on how to acquire and use GazeCode.

Tested on MATLAB R2017b and R2018b. Because a bunch of java hacks are used for the GUI, it is quite possible that the viewer part of this repository is not all that compatible with different matlab versions. Pull requests welcomed! Some used functionality was only introduced with R2015b (JSON decoder), so the code in this repository is not expected to run on versions of matlab older than that.

NB: do not expect fluent playback. I get a few frames a second on my 4K display, the matlab GUI renderer can't handle all this.

Screenshot

Click the screenshot to see a full-size version.



Usage

When running the viewer GUI, `glassesViewer.m`, a file picker will appear. Select the folder of a recording to view. This needs to point to a specific recording's folder. If "projects" is the project folder on the SD card, an example of a specific recording is: `projects/rkamrkb/recordings/zi4xmt2`.

If you just wish to parse the Tobii glasses data into a MATLAB readable file, you can directly call `./function_library/TobiiGlassesRecordings/getTobiiDataFromGlasses.m` with the same specific recording as above as the input argument.

An overview of the viewer GUI's functionality is provided in the manual.

Default settings for the reader are in the `default.json` file. To alter these settings, read in the json file with `jsondecode()`, change any values in the resulting struct, and pass that as an input argument to `glassesViewer`. The options that can be set through this interface are listed below.

Annotations are stored to a file `coding.mat` in the same folder as where the displayed Tobii Glasses data is loaded from. This file is created upon loading the GUI if any annotations are defined, and can be updated by pressing the "save coding" button in the GUI. When opening the same folder again in the GUI, annotations are loaded from this file (and possibly overridden depending on the settings of some annotation streams).

Viewer interface

mouse: - can drag the time indicator (red line in data axes) or annotation markers with left mouse button. If the control key is held down while starting the drag, and there is an event marking for the same sample in one or more other streams, all these aligned markers are dragged together. The `escape` key cancels all drag actions. - left-clicking on an axis opens an annotation panel if any annotation streams are defined in the settings - shift-clicking on the time axis inside an existing annotation starts adding an intervening annotation. Make a second shift-click to close the added interval. Use this, e.g., to add a missing saccade in a too long fixation. The `escape` key cancels this action, as does clicking anywhere without the shift key held down. Shift-clicking, keeping the mouse button down, dragging and releasing adds an intervening annotation for the range over which the mouse moved without a need for the second click. - control-shift-clicking starts adding intervening annotation across all streams instead of only current stream. - dragging with right mouse button pans the plot, moves the visible data in the window - double-click on data axis sets time to clicked time - using mouse scrollwheel when the pointer is on a data axis has two functions: 1. if holding down `ctrl`, the time window is zoomed around the cursor position 2. if holding down `shift`, the value range of the vertical axis is zoomed around the cursor position

keyboard: - left arrow or `a` key: show previous time window - right arrow or `d` key: show next time window - `space bar`: start/stop playback - `z`: enter/exit zoom mode - `ctrl` + `r`: reset vertical axes limits - `escape`: closes open dialogues, cancels drag actions and the adding of intervening events

Settings

The settings are stored in a nested structure that directly reflects the `defaults.json` file's structure (in fact, they are directly read in from there unless `settings` are provided by the user as an input argument to `glassesViewer()`). As such, here we document the structure of this file and the understood options.

plot

In `settings.plot`, the following fields are understood:

settings	description
`initPanelOrder`	Sets which data panels are initially shown on the left side of the interface and the order in which they are shown, as an array. E.g. `["azi", "ele"]`. Understood panels are `"azi"`, `"ele"`, `"vel"`, `"pup"`, `"gyro"`, `"acc"` and `"scarf"`. `"scarf"` is a special panel showing the annotations in one or multiple coding streams and allows switching for which stream annotations are shown in the other panels.
`panelNames`	Table in which you can provide a name to be shown for each panel. E.g. `{"azi":"azimuth", "ele":"elevation"}`. Understood panels are `"azi"`, `"ele"`, `"vel"`, `"pup"`, `"gyro"`, `"acc"` and `"scarf"`.
`timeWindow`	Length (s) of the time window
`aziLim`	The range (deg) of the `"azi"` panel.
`eleLim`	The range (deg) of the `"ele"` panel.
`velLim`	The range (deg) of the `"vel"` panel.
`gyroLim`	The range (deg) of the `"gyro"` panel.
`lineWidth`	Width of the lines of the data plots in the panels
`removeAccDC`	If true, the mean value of the three traces in the acceleration plot is removed. This removes the 1g acceleration that is always measured due to earth's gravity, making it easier to see accelerations due to subject movement
`SGWindowVelocity`	Length (ms) of the Savitzky-Golay filter used to calculate gaze velocity
`scarfHeight`	Height (pixels) of a stream in the scarf plot

VCR

In `settings.VCR`, the following fields are understood:

settings	description
`seekShort`	Number of samples to seek backward or forward when pressing the `<` and `>` buttons
`seekLong`	Number of samples to seek backward or forward when pressing the `<<` and `>>` buttons

dataQuality

In `settings.dataQuality`, the following field is understood:

settings	description
`windowLength`	Length (ms) of moving windows used to calculate RMS-S2S of the dataset

coding

In `settings.coding`, the following fields are understood:

settings	description
`streams`	Array of annotation stream settings
`colors`	Array of RGB colors used for coloring events, format: `[[R, G, B],[R, G, B]]`
`closePanelAfterCode`	Boolean indicating whether coding panel is closed after adding a new annotation (true), or not (false).

annotation streams

Each of the annotation streams takes labels and a color coding in their `categories` field. This is an array with the names of each annotation category, followed by a one-based index into the `settings.coding.colors` array of colors indicating the color in which the annotation should be shown. Color can be `null` if an annotation should not be color coded. Example: `[ "none", null, "fix", 1, "sac", 2, "sp", 3 ]`.

Events are identified by consecutive power-of-two labels. So for the above example `categories` field, the events would be (internally) coded as follows:

events	code
none	1
fix	2
sac	4
sp	8

A stream can furthermore contain a single special code acting as a flag modifying a base event. An event whose name is suffixed with '+' can take as flag an event whose name is prefixed with `*`. These codes can then be bit-anded together to denote an annotated flag. So for the following scheme:

events	code
none	1
fix	2
sac+	4
sp+	8
*vergence	16

With this setup, a vergence flag can be added to a saccade or a smooth pursuit event, which will be represented as the code 4+16=20 for a saccade with a change in viewing distance and 8+16=24 for smooth pursuit.

User-provided codes that are to be shown from a `fileStream` or a `classifier` should output a stream of integers using this coding scheme. Labels and colors are attached to these output categories using the `categories` field of a stream's settings.

Each annotation stream in the array of defined streams should have a `type` field. The following annotation stream types are understood:

1. syncIn and syncOut

`syncIn` and `syncOut` streams are special streams that are view-only, and cannot be edited through the GUI. The Tobii Glasses 2 has a sync port that takes a single-channel TTL input and provides a single-channel TTL output. These streams visualize the activity of those channels. Understood options are:

setting	description
`type`	Stream type. `syncIn` or `syncOut`
`lbl`	String to be shown in GUI, naming the stream
`categories`	Array defining the labels for each code, and the colors in which to show them. `syncIn` and `syncOut` streams have two events, up/inactive (1) and down/active (2).

2. handStream

`handStream` streams start empty and annotations are made in them through hand coding.

setting	description
`type`	Stream type. `handStream`
`lbl`	String to be shown in GUI, naming the stream
`locked`	Boolean indicating whether annotations can be made, altered or deleted in this stream.
`categories`	Array defining the labels for each code, and the colors in which to show them.

3. fileStream

`fileStream` streams allow showing annotations loaded from a file. These files should contain two tab-separated columns. The first denotes the sample index of an annotation start mark, and the second column the code attached to the interval started at that mark. See the included file `./testCodingFile.txt` for an example.

setting	description
`type`	Stream type. `fileStream`
`lbl`	String to be shown in GUI, naming the stream
`locked`	Boolean indicating whether annotations can be made, altered or deleted in this stream.
`file`	String containing full path to file from which coding should be loaded. If only a filename is provided, the file is assumed to be in the current directory. The path can start with the special string `!!recordingDir!!` to indicate that the path is relative to the directory of the selected recording that is opened in the GUI.
`skipIfDoesntExist`	If set to true, code will not error when the specified file does not exist, it will silently remove the coding stream instead.
`alwaysReload`	Boolean indicating whether coding should be reloaded from file when opening the GUI if a `coding.mat` file already exists for the loaded Tobii Glasses data directory (true), or whether the coding stored in the `coding.mat` file should be shown instead (false). Setting this to true ensures that the displayed fileStream is updated if the file contents have changed (when the GUI is reloaded for the recording)
`needToCorrectT0`	Set to true if t=0 in the file to be loaded corresponds to the first data sample instead of to the start of the video (which is t0 for this GUI and its output)
`categories`	Array defining the labels for each code, and the colors in which to show them.

4. classifier

`classifier` streams allow showing annotations produced by a MATLAB function. An example classifier is included in the file `./user_functions/HoogeCamps2013/HC13.m`. This implementation of Hooge, I. T. C., & Camps, G. (2013). Scan path entropy and arrow plots: Capturing scanning behavior of multiple observers. Frontiers in Psychology, 4(996). doi:10.3389/fpsyg.2013.00996 was provided by GazeCode.

setting	description
`type`	Stream type. `classifier`
`lbl`	String to be shown in GUI, naming the stream
`locked`	Boolean indicating whether annotations can be made, altered or deleted in this stream.
`function`	String containing function to be called to produce event classification. This function should be on the MATLAB path.
`alwaysRecalculate`	Boolean indicating whether coding should be reclassified with the classifier function if a `coding.mat` file already exists for the loaded Tobii Glasses data directory (true), or whether the coding stored in the `coding.mat` file should be shown instead (false). Setting this to true ensures that the displayed classifier stream is updated if the classifier or its settings have changed (when the GUI is reloaded for the recording)
`alwaysRecalculateUseDefaultSettings`	Boolean indicating whether recalculating classification (see `alwaysRecalculate`) should use the last-applied parameter settings stored in the `coding.mat` file (false) or instead use parameter settings from the user provided settings (true), overriding what may have been done in a previous session.
`categories`	Array defining the labels for each code, and the colors in which to show them.
`parameters`	Array of parameters configuring the classifier, which can be marked as user settable in the GUI. See Classifier parameters.

Classifier parameters

The following settings are understood for classifier parameters:

setting	description
`name`	String denoting fieldname in which parameter value will be stored when passing to the classifier function
`label`	String to be shown in GUI, naming the parameter
`type`	Datatype of parameter. Understood are `double`, `int` and `bool`
`value`	Initial value of parameter
`settable`	Boolean indicating whether this parameter is changeble in the GUI by the user
`range`	For settable parameters of types `double` and `int`, denotes the range by means of a two element array `[min, max]`
`granularity`	For settable parameters of types `double` and `int`, denotes the stepsize by which the parameter changes when moving the spinner a tick in the GUI

TODOs

The below items could be considered to be implemented. Currently, none are planned to actually be executed - user own json decoder function (or get from mathworks FEX), so we can support matlab version a little older than what i do currently. In terms of JSON parsing, we however currently already support R2015b. This old version is otherwise untested, and I am not interested in support even older versions. - progress bar when loading in data Tobii data (not linear time but can indicate steps completed or so)

Known issues

- matlab VideoReader (at least in R2017b on Windows 10) cannot read the last few frames of some video files, when seeking to a specific time of frame, even though it can read them when just reading through the file frame by frame. This *appears* to occur for files where the last few frames are marked as keyframes in the mp4 stss table, even though they are not. All read frames are correct, so not a big problem.

Requirements

- Firmware versions that are too old do not contain some of the fields in their data file that are required by this reader. Known to work are 1.22.0-zucchinipaj and several later firmware releases.

Relation GlassesViewer export to Tobii Pro Lab

export

Tobii Pro Glasses 2 recordings can be exported to a plain text file and a scene video with overlaid gaze marker using the Tobii Pro Lab software. When doing so, a different time code is used than produced by GlassesViewer's recording parser: - For GlassesViewer's export, t=0 is at the start of the scene or the eye video, whichever is later. That means that often there are some gaze, gyroscope and/or accelerometer data with negative timestamps, as the gaze recording stream often appears to start earlier than the scene video. - For the Tobii Pro Lab export, t=0 appears to be when the gaze stream starts. As this is often earlier than the start of the scene video, the scene video exported by Tobii Pro Lab often contains a few black frames at its beginning. While i have not tested this, converting timestamps between the two exports should thus be a matter of adding or subtracting the time difference between the timestamp of the first gaze sample and the timestamp of the first eye- or scene video frame (whichever is later).

License details

Acknowledgments

mp4 parsing code extracted from https://se.mathworks.com/matlabcentral/fileexchange/28028-mpeg-4-aac-lc-decoder, and further extended for our purposes.