Manual

Isaac Kinley

June 2019

Contents

1 Getting started

pupillometry-pipeline is a library of Matlab/Octave functions for processing eye tracking data and a user interface that runs these functions with a click.

1.1 Initializing

Change Matlab's working directory to the pupillometry-pipeline folder (which should contain `pupl_init.m`, `LICENSE.md`, etc.). You can do this either using the "Current Folder" panel or from the Command Window using the `cd` function.

Run `pupl_init` in the Command Window to add the source code to Matlab's path, initialize the global `eyeData` variable, initialize the user interface, and load whatever add-ons are in the add-ons/ folder. Any of these last 3 steps can be skipped by adding `noGlobals`, `noUI`, and/or `noAddOns` as command-line arguments to `pupl_init`.

1.2 Importing raw data

Under `File > Import`, you will see options for importing raw data. Note: if you import raw data from a BIDS-formatted folder, event logs will not also be imported.

1.3 Manipulating data

Loaded data will appear on the user interface. Unselected data is ignored by functions run using the user interface. To delete data from Matlab's workspace, go to `File > Remove datasets`. Data is also accessible in the Command Window through the global variable `eyeData`. If you process data in the command window and want to update the user interface, run `pupl_init noglobals`.

1.4 Importing raw event logs

After you have loaded some data, you will be able to attach event logs to it. Go to `Trials > Event logs > Import` to see your options for importing raw event logs.

1.5 Synchronizing event logs and eye tracker data

When raw eye tracker data is loaded, event onsets are recomputed such that an event coinciding with the first data sample will be assigned an onset of zero. This means that timestamps in event logs cannot be directly copied to eye data. Furthermore, the clock used to measure timestamps in the event log and the clock used to measure timestamps in the eye data may drift relative to one another. To solve this problem, we use regression to find a linear mapping between timestamps the two records. To begin, go to `Trials > Event logs > Synchronize with eye data`. A window will open for you to specify a mapping between event types in the two records (for example, there may be sync markers sent by stimulus presentation software to both the eye tracker and an event log). After this, you will be able to select which events from the event logs should be copied to the eye data, whether they should be copied under different names, and whether the pre-existing events in the eye data should be deleted.

2 Visualizing data

2.1 Plotting continuous datastreams

It is a good idea to visualize data throughout processing to understand how it is being altered. Under `Plot > Plot continuous` there are tools for plotting continuous datastreams. The unprocessed data are plotted using dotted lines while the processed data are plotted using solid lines. For pupil diameter, data from the left eye are plotted in blue, data from the right in red, and merged data from both eyes in black. For gaze data, the gaze x coordinates are plotted in blue, the y coordinates in red.

2.2 Plotting gaze coordinates

To visualize gaze data from all active datasets, go to `Plot > Gaze scatterplot`.

2.3 Plotting pupil size

To get an idea of how pupil size measurements are distributed for each participant, go to `Plot > Pupil diameter histogram`

2.4 Visualizing pupil foreshortening error

The pupil foreshortens as it turns away from the eye tracker, being measured as smaller than it really is. E.g. if the eye tracker is placed below a computer screen, the pupil will tend to be measured as smaller when looking at the top of the screen and either side. This will appear as a (roughly) linear trend in a plot of pupil size vs gaze y coordinate and a (roughly) quadratic trend in a plot of pupil size vs gaze x coordinate. Under `Plot > Pupil foreshortening error`, there are tools to examine whether there is a systematic relationship between gaze location and measured pupil size. The tools explained in 3.8 can correct for such a relationship.

2.4.1 Plotting pupil size as a function of single gaze dimension

To plot pupil size as a function of either gaze x or gaze y coordinate, go to `Plot > Pupil foreshortening error > Pupil size vs gaze [x/y]`.

2.4.2 Plotting pupil foreshortening error surface

To visualize pupil size across the whole gaze field, go to `Plot > Pupil foreshortening error > Error surface`. This divides the gaze field into a grid and computes the mean pupil size for points falling within a square centered on each node.

2.5 Plotting trials from event-related pupillometry

Once epochs have been extracted for event-related pupillometry, they can be plotted using `Plot > Plot trials`. The meanings of the line colours are the same as in the continuous pupil diameter plots.

2.6 Plotting trial sets from event-related pupillometry

Once epochs have been merged into trial sets (as explained in 5.4), they can be plotted using `Plot > Plot trial sets`. The meanings of the line colours are the same as in the continuous pupil diameter plots. The shaded error bars represent one standard error of the mean.

3 Processing raw data

3.1 A note on specifying durations

You may use units and arithmetic to specify lengths of time. Valid units are as follows:

`ms`: milliseconds
`s`: seconds
`m`: minutes
`dp`: datapoints

For example, `1s + 2 dp - 100 ms - 2d` translates to one second plus two datapoints minus 100 milliseconds minus two datapoints.

3.2 A note on specifying relative quantities

You may compute statistics and use arithmetic to specify relative quantities. Valid statistics are as follows:

`mn`: mean
`md`: median
`sd`: standard deviation
`v`: variance
`iq`: interquartile range
`%`: percentile (e.g. "15%" computes the 15th percentile)

E.g. `mn - 2sd` would compute the mean minus two standard deviations.

3.3 Trimming data

3.3.1 Trimming extreme pupil diameter measurements

Go to `Process > Trim data > Trim extreme dilation values` to remove pupil size measurements that are too high or low. In the window that opens, you can specify cutoff points as explained in 3.2. Note: the gaze measurements corresponding to rejected pupil size measurements will also be removed. If you plan to run the same pipeline on multiple participants, it is best to create relative cutoffs since different participants will likely have different average pupil size measurements.

3.3.2 Trimming extreme gaze measurements

Go to `Process > Trim data > Trim extreme gaze values` to remove gaze measurements that are too extreme (e.g. off-screen or too far from a fixation cross). You can specify cutoff points as explained in 3.2. Note: the pupil size measurements corresponding to rejected gaze measurements will also be removed. If you plan to run the same pipeline on multiple participants, it is best to create absolute gaze cutoffs since screen x and y values are likely to have the same meaning across participants.

3.4 Trimming isolated samples

Go to `Process > Trim data > Trim isolated samples` to remove little blips of data that are unlikely to be meaningful or accurate measurements. See below to understand how to specify the max length of these islands of isolated data.

3.5 Trimming blink-adjacent samples

The pupil is partially obstructed shortly before and after the eye fully closes during blinks. On eye trackers with high sample rates, this results in datapoints recorded near blinks being unreliable. To trim these unreliable points, go to `Process > Trim blink-adjacent samples`. See 3.1 for an explanation of how to specify durations in the dialog boxes that appear.

3.6 Filtering data

To apply a moving mean or median (recommended) filter to pupil size or gaze data, go to `Process > Moving average filter`.

3.7 Saccades and fixations

3.7.1 Identifying

Algorithms available for identifying saccades and fixations can be found under `Process > Identify saccades and fixations`. Note: data points themselves are not labelled as saccades or fixations, only the periods of time between them. The justification for this is as follows: if gaze samples 1, 2, and 3 are all at the same coordinates and gaze samples 4, 5, and 6 are at a far-away coordinate, which point ought to be labelled as a saccade? Clearly the participant was fixating at points 1, 2, and 3 and then again at points 4, 5, and 6, and the saccade took place between points 3 and 4.

3.7.2 Mapping to fixations

After identifying fixation periods, the data during these periods can be reassigned to their centroids (spatial means) using `Process > Map gaze data to fixation centroids`.

3.8 Pupil foreshortening error correction

For an explanation of pupil foreshortening error, see 2.4. It can be corrected using the options under `Process > Pupil foreshortening error correction`.

3.9 Interpolating missing data

To linearly interpolate missing pupil size and gaze data, go to `Process > Interpolate missing data`. Note: care must be taken when interpolating gaze data since, if saccades take place during long periods of missing data, linear interpolation will make these appear to be periods of smooth pursuit.

3.10 Merging left and right pupil diameter

To analyze pupil diameter, it is necessary to average the left and right data streams. This can be done using `Process > Merge left and right diameter streams`.

3.11 Adding BIDS information

To add subject, session, task, and acquisition information for saving data to the BIDS format, go to `BIDS > Add BIDS info`.

4 Saving data

To save data, go to `File > Save`. A separate filesave dialog will open for each active recording. The `.eyedata` extension on saved recordings is merely an alias for version 6 `.mat` files.

4.1 Batch saving data

To save all active data to the same folder using filenames corresponding to recording names, go to `File > Batch save`. This will open a single folder selection dialog.

4.2 Saving data to BIDS format

If you have data loaded and active, you can save it in a BIDS-formatted project directory.

4.2.1 Saving raw

To create and populate the raw/ folder in your BIDS-formatted project directory, go to `BIDS > Save > Save raw from current data`.

4.2.2 Saving sourcedata

To create and populate the sourcedata/ folder in your BIDS-formatted project directory, go to `BIDS > Save > Save sourcedata of current data`. Note: this re-loads the raw data using the `getraw` method of the data structure, which can be slow depending on the format of the raw data.

4.2.3 Saving derivatives

To save processed data in a folder withing the derivatives/ folder of your BIDS-formatted project directory, go to `BIDS > Save > Save current data as derivative`. Note: this folder will have the same internal structure and filenames as the sourcedata/ folder and can be loaded using the function explained in 4.3.

4.3 Loading BIDS sourcedata

Under `BIDS > Load sourcedata`, you will have the option to load sourcedata from a BIDS-formatted directory. Note: if you use this option, any event logs corresponding to the data will be imported as well. You can also use this menu option to load data from a derivatives/ subfolder that was saved using the function explained in 4.2.3. Simply provide said subfolder as the sourcedata folder.

5 Event-related pupillometry

5.1 Extracting trials

To extract trial data, go to `Trials > Event-related pupillometry > Fragment continuous data into trials`. Trials are defined and extracted relative to events in the eye data. For example, if you wanted to examine pupillary response to an event called `showImage`, you would select this event name in the window that appears next. The durations of trials can be specified in the next dialog window. For example, to extract data from 200 milliseconds before the occurrence of events until 2 seconds after, you would enter `-200ms` and `2s` in the next dialog window. These durations can be specified as explained in 3.1.

5.2 Baseline-correcting trials

Due to variability in pupil size measurements, it is necessary to baseline correct trial data, either by subtracting the baseline mean or by computing the percentage change from baseline duration. To do this, go to `Trials > Event-related pupillometry > Baseline correction`. The mapping from baseline periods to trials can be one-to-one (e.g. baselines periods are the 200 ms before each event onset), one-to-all (e.g. one single baseline period occurs at the beginning of the experiment), or one-to-some (e.g. one baseline period occurs before a block of 5 trials). If the one-to-some mapping is selected, trials are baseline-corrected using the most recent baseline period. The durations of baseline periods can be specified as explained in 3.1.

5.3 Rejecting trials

Trials can be removed from further analysis using `Trials > Event-related pupillometry > Trial rejection`. Trials can be rejected according to the proportion of data missing or the presence of extreme pupil size measurements.

5.4 Merging trials into sets

It is useful to average together trials of multiple types to get reliable estimates of pupillary response. E.g. you may want to examine responses to both a particular stimulus and a category of stimuli to which it belongs. To define

sets of trials for later analysis, go to `Trials > Event-related pupillometry > Merge trials into sets`. Note: even when you do this the first time, you will be asked if you want to overwrite the pre-existing trial sets. This is because extracting trials initially creates a trial sets with a one-to-one mapping to trial types.

5.5 Calculating statistics

To compute statistics on the trial data, go to `Trials > Event-related pupillometry > Write statistics to spreadsheet`. You can compute multiple statistics, and the time windows to be used for analysis can be specified as explained in 3.1. You can compute the statistics on each of the trials and either save all of these individually or save their average, or you can compute the trial averages and compute statistics on these.

6 Gaze tracking

7 Automating processing pipelines

To export the processing history to a Matlab script, go to `File > Save processing script`. When a processing function is run, it saves the equivalent command as a string to the `history` field of the data. All of these commands together constitute a processing pipeline.

7.1 Running pipeline on BIDS sourcedata

8 Contributing

8.1 Raw data loader

8.2 Raw event log loader

8.3 Adding to the user interface