Eye Tracking Data Analysis for Experiment 1

**This is a basic overview of how to use the Exp1 ET analysis Script**

First, to clarify what files you will work with: Eyelink 1000 eyetracker generates edf files that are converted to asc files with the EDFconverter that is built in the data parsing section in the Analyzers script.

Second, if MEG (SA:Birmingham, SB:Peking then the data is binocular otherwise always monocular)

Important variables in this across *Analyzers*, *AnalysisSupport*, *AnalysisHelpers*:

**Params**: all the parameters that will be used for analysis base on the experiment being analyzed

**TrialData**: The gaze data broken down to every trial in the Experiment, with timestamp

**TrialInfo**: Collection of triggers that give you all the information of each and every trial of the Experiment including the duration, category, relevance, orientation of the specific trial

Below, I will describe all the modules this set of scripts uses and describe the functions. Besides this description the comments should guide you what each input and output are designed for and what each function is generated to achieve.

There are 6 different modules/scripts:

1. **DataParser**
   1. *Initparameters*: defines and returns the parameters that will be used for analysis base on the experiment being analyzed
      1. screenWidth, screenHeight, viewDistance, participantName, sampling frequency, which eye is recorded
   2. *ExtractTimeStamps:* this function extracts the onset, stimulus ID, and trial type of each trial
      1. eyeDFs include 6 different dataframes: 1. Recording info (not important), 2. messages (that is used here in this function to get the triggers recorded by the ET), 3. Fixation recorded by Eyelink, 4. Saccades recorded by Eyelink, 5. Blinks recorded by Eyelink, 6. **Raw gaze samples (we use only this for now)**
   3. *SequenceEyeData:* segments the full eye data into trials. So at each trial timestamp, we grab the data from pre-onset (500ms) until sometime post-onset to get a window of data that associate with that trial
      1. It returns the raw samples carved out to each trial, which will be fed into the RemoveBlinks function to further process the data and get rid of blinks based on Hershman method
   4. *RemoveBlinks:* This function takes in the sequenced TrialData and for each trial, it detects and removes blinks. The blink detection script and algorithm used here are based on Hershman et al. 2018 (<https://osf.io/jyz43/>).
      1. *based\_noise\_blinks\_detection:*This function incorporates the Hershman method, which is the following:

*Hershman method for Blink detection*

Edge Case 1: there are no blinks

Edge Case 2: the data starts with a blink. In this case, blink onset will be defined as the first missing value:

* It starts with a blink but does not end with a blink
* It starts with a blink and ends with a blink

Edge Case 3: the data ends with a blink. In this case, blink offset will be defined as the last missing sample

* Ends with a blink but does not start with a blink
* Ends with a blink and starts with a blink ---> Already handled "start with blink" in Edge case 2 so it reduces to i (previous case)

Smoothing the data in order to increase the difference between the measurement noise and the eyelid signal. Finding values <=0 and >=0 in order to find monotonically increasing and decreasing sections of smoothened pupil data. Looking at all values in the monotonically decreasing sequence should be <= 0 and those included in the monotonically increasing sequence >= 0

* 1. *ExtractSaccades:* This is the main function responsible for extracting saccadees and microsaccades
     1. *GetVelocity*
     2. *ExtractMonocularMS*
     3. *GetVelocityThreshold*

The extraction of saccades and microsaccades are explained in details: Engbert, R., & Mergenthaler, K. (2006) Microsaccades are triggered by low retinal image slip

* 1. *RemoveSaccades:* This function takes in the sequenced trial data (after or before blinks are removed) and removes saccade data. If no saccade info (output of ExtractSaccades) is provided. The function makes the call to ExtractSaccades.
  2. *ParseEyeLinkAsc:*This function is the very first function that reads in the raw data from the .asc (ASCII) files the edfconverter already created and parses through all the messages eyelink recorded.
     1. Creates the 6 dataframes I have already mentioned in *ExtractTimeStamps:*
        1. dfRec contains information about recording periods (often trials)
        2. dfMsg contains information about messages (usually sent from stimulus software)
        3. dfFix contains information about fixations
        4. dfSacc contains information about saccades
        5. dfBlink contains information about blinks
        6. dfSamples contains information about individual samples

1. **AnalysisSupport**

This script includes the analysis of the individual features of the ET data and saves the plots for each participant it cycles through.

* 1. *AnalyzeGaze:* This function is designed to calculate fixation distance and fixation density across categories/duration/orientation/Task relevance, and plot them on either line plots or in case of density on heatmaps.

***Disclaimer:*** Many of the datasets have missing values or extra nan values assigned, so due to this defect I am trimming each gazeX and gazeY to an ideal size so later on you can compile all data across all subjects for group plots and group stats

* 1. *AnalyzeSaccades:* This function is designated to calculate saccade amplitude, direction and rate. It again calculates these with the GetSaccData function and then plots them in line plots and the direction in roseplots.
     1. *GetSaccData:* This function computes saccade amplitude per condition (category/duration/orientation/task relevance), saccade rate per condition (category/duration/orientation/task relevance), saccade direction per condition (category/duration/orientation/task relevance), and finally mean saccade amplitude over trials.
  2. *AnalyzeBlinks:* This function is designated to calculate mean blink rate across conditions (category/duration/orientation/task relevance) and plot them in line plots.
  3. *AnalyzePupil:* This function is designated to calculate mean pupil size across conditions (category/duration/orientation/task relevance) and plot them in line plots.
     1. Pupil size is calculated as the sum of the number of pixels inside the detected pupil contour.

1. **AnalysisHelpers**
   1. *deg2pix*: convert degrees of visual angle to pixels
   2. *CalcFixationDensity*: This function divides the screen into bins and sums the time during which a gaze was present at each bin (density heatmaps of fixation)
   3. *plot\_group*: This is a very long function that includes most of the lines from **AnalysisSupport,** meaning it includes gaze analysis, blink analysis, saccade analysis and pupil analysis on the group level. Its inputs are the compiled variables across all the subjects:
      1. meanFixation distance
      2. meanBlink rate
      3. meanSaccade amplitude and direction
      4. meanPupil size

Then plot\_group uses the same plotting functions as **AnalysisSupport** to plot the compiled values of these features (meanFixationdistance, meanBlink rate, meanSaccade amplitude and direction, meanPupil size)

1. **Plotters**: Contains all the functions responsible for plotting including
   1. *AccioFigure*: Returns a figure with appropriate properties for plotting analyses results
   2. *ErrorLinePlot*: plot a line plot with a shaded error area defined by standard error of mean
   3. *HeatMap*: plots a heatmap of data and shows a rectangular roi defined by roi size
   4. *PolarPlot*: plots direction rose plot for saccade direction results
2. **Analyzers:** This script is the one that you run.
   1. The script then starts with defining the paths for Behavior and ET data
   2. Then define the lab specific screen sizes and viewing distances (already given)
   3. Then some of the variables get preallocated for future use
   4. *skip\_et*: These are the participants that have ET data that are not usable, so we skip them
   5. Then the loop starts over all the available subjects
      1. Locate if there is already a parsed pickle file created for the participant
         1. if yes load it and move on
         2. if no run Dataparser and make the file then save it
      2. Plotting starts for each subject
         1. AnalyzeGaze
         2. AnalyzeSaccade
         3. AnalyzeBlink
         4. AnalyzePupil
      3. Stats calculation for fixation/blink/saccades/pupil size across different duration broken down to trials
      4. *group\_plot\_variables:*Collect all the appended variables that are the input for the *plot\_group* in **AnalysisHelpers** to plot the group results
      5. Concatenate all stats results into one big dataframe and save it as a csv file for analysis in R or JASP
3. **QualityChecker**

**Disclaimer:** Even though we calculate the percent of fixation as a measure of quality of ET, we do not exclude participants based on eye-tracking.

* 1. This is a separate script that can be used to parse the datasets for each participant (exact same way as in **Analyzers**) as well as calculate the percent of fixation within the given 3 degrees of visual angle circle –
  2. We do not exclude datasets based on ET QC
  3. *check\_percent\_fixation:* this function calculates and saves the amount of time spent within a specified range (reference angle) based on the distance from fixation
     1. Here too I format the size of GazeX and GazeY to be able to compile the fixation distance results across participants, due to the fact that some datasets have many missing values or nan values added and they do not match in size always across participants

**II. Running the code**

1. **Analyzers**: The bottom of the script has the input function. Here you plug in the:
   1. ‘modality’=fMRI or MEG or ECoG
   2. ‘stats’=True/False
   3. ‘plotting’=True/False

The function will then loop through subjects and create a directory called ‘Figures’ within the eye tracker folder of each subject and will save all the figures there as well as it creates a general ‘Figures’ folder for the group plots. In addition it creates a statistics folder where it saves all the fixation/blink/saccade/pupil size results that can be used for further statistical analysis.

1. **QualityChecker**: The bottom of the script has the input function. Here you plug in the:
   1. ‘modality’=fMRI or MEG or ECoG
   2. group\_QC=True/False - with this you can save all the fixation QC results into the same folder –not really used anymore