Eye Tracker Option Manual

Tim Van Wesemael

April 2018

Contents

1 Screen options **1**

2 Data options **1**

3 Task options **2**

4 Calibration validation options **5**

5 Plot options **5**

6 Filter options **6**

1 Screen options

This set of options concern the properties of the screen used to show the stimuli to the participant. In listing 1 example values for the screen options can be found.

`xres` *integer*
The width in pixels of the used screen.

`yres` *integer*
The height in pixels of the used screen.

`one_pxl_mm` *double*
The diameter of one pixel of the screen measured in millimetres. Found by dividing the screen height/width in pixels by its height/width in millimetre.

`scrSz` *1x2 double array*
The width, respectively the height of the screen in centimetres. This information may be calculated from the above parameters (as done in line 4 of code snippet 1) and may thus seem redundant, however this argument is needed to make the scripts compatible with the I2MC filter.

2 Data options

These parameters depend on the way the data is recorded and needs to be evaluated. In listening 2 examples on data options attributions can be found.

Listing 1: Screen options example. In this case the screen has a height of 1920 pixels, a width of 1200 pixels and a pixel diameter of 0.26972 millimetre. The screen size in centimetre is calculated from the other parameters in this case.

```
opt.xres       = 1920;
opt.yres       = 1200;
opt.one_pxl_mm = 0.26972;
opt.scrSz      = [opt.xres opt.yres]*opt.one_pxl_mm/10;
```

`freq` *integer*
The frequency at which samples are taken in hertz. This number equals the number of gaze points the eye tracker records every second.

`raw` *boolean*
Choose whether the data needs to be filtered before being interpreted. If set to true (*strongly discouraged*) the I2MC will not be used, instead the raw recorded gaze points will be used.

`expected_eye_pos` *1x3 double array*
The position in three dimensional space where the eye is expected. The coordinate system this points needs to be defined in, uses pixels as distance unit. The first coordinate is the horizontal distance to the right from the left border of the screen. The second coordinate is the distance along the horizontal border, starting from the top of the screen and down is the positive direction. The third coordinate is the perpendicular distance from the screen. So, an object which sits in front of the screen has a negative third coordinate, while an object behind the screen has a positive one.

`experiments` *1xN string cell*
The collection of all names (or abbreviations) of the experiments that need to be evaluated (where N is the number of experiments). An experiment is a collection of tasks which share the same areas of interest names. The exact locations of the area of interest may differ from trail to trail within an experiment. An experiment may contain discontinuities in time and thus be stopped, restarted or alternated with another experiment.

`keys.`***`exp`*** *1xN string cell*
For every experiment name (*exp*) defined in `experiments`, one of these arrays needs to be initialised. It must contain the key for every trail (in total N) that constitutes the experiment, *exp*. The keys given in this cell must be found in the *start* column of the input file and refer to a trail. A trail is always part of an experiment and thus inherits the names for its areas of interest. In contrast to an experiment, a trail must contain a fixed number of succeeding data points without any discontinuities. The position of the areas of interest may or may not change during the trail. The way to define this difference will be explained next section.

3 Task options

The options in this section concern the way AOIs are defined during the task and how to interpret them. Many of the entries rely on coordinates. The coordinate system's distance unit is a pixel. Its first coordinate is the distance to the left side of the screen and its second one is the distance from the top. In listening 3 an example for filling in the following options is worked out.

`aois.`***`exp`***`.names.voronoi` *1xN string cell*
This option contains the names for N the voronoi areas of interest of experiment *exp*. In case this experiment does not have any voronoi AOIs, this variable should refer to an empty cell.

Listing 2: Example for data options. In this case the eye tracker sampled at 120 *hz*, the data needs to be filtered and the position of the eye is expected to be right over the middle of the screen, in front of it at a distance of 60 *cm*. Two experiments will be evaluated, abbreviated by *HF* and *UL*. The *HF* experiment consists out of two trails, namely *facesL* and *housesL*. The HF experiment only has one, *M*, which can still occur several times in the dataset.

```
opt.freq                = 120;
opt.raw                 = false;
opt.expected_eye_pos    = [opt.xres/2 opt.yres/2 -600/opt.one_pxl_mm];
opt.experiments         = {'HF', 'UL'};
opt.keys.UL             = {'M'};
opt.keys.HF             = {'facesL', 'housesL'};
```

`aois.`***exp.***`names.rect` *1xN string cell*
This option contains the names for *N* the rectangular areas of interest of experiment *exp*. In case this experiment does not have any rectangular AOIs, this variable should refer to an empty cell.

`length.`***exp.key*** *integer*
If the trail defined by *key* does not consists of multiple subtrails (thus its AOIs does not change location), this variable saves the total length of the trail in samples. If the trail does consists of subtrails, this variable must not be initialised.

`length.`***exp.key.***`total` *integer*
If the trail defined by *key* does consists of multiple subtrails (thus its AOIs does change location over time), this variable saves the total length of the trail in samples. If the trail does not consists of subtrails, this variable must not be initialised.

`length.`***exp.key.subkey*** *integer*
If the trail defined by *key* does consists of multiple subtrails (thus its AOIs does change location over time), this variable saves the length of subtrail *subkey* in samples. If the trail does not consists of subtrails, this variable must not be initialised.

`aois.`***exp.key.(subkey.)***`voronoi` *Nx2 integer array*
This variable saves the coordinates of the *N* points that define the voronoi tessellation. As an effect *N* has the same value as it has in `aois.`*exp*`.names.voronoi`. The first column contains the horizontal coordinates and the second column the vertical ones. So in every row, exactly one point is saved. If the trail denoted by *key* does not consists of subtrails, the subkey field must be ignored. In the other case, a corresponding entry for every subtrail needs to be created.

`aois.`***exp.key.(subkey.)***`max_vor` *double*
The maximum allowed distance there may be between a centre of a voronoi area of interest and a point, for that point to still be a part of the voronoi AOI.

`aois.`***exp.key.(subkey.)***`rect` *Nx4 integer array*
In this matrix the coordinates of the upper left and lower right corner of every rectangular area of interest is saved. Each of the *N* (same value as *N* in `aois.`*exp*`.names.rect`) AOIs in a row. The columns contain from left to right respectively the following values: outermost left and right horizontal distance and outermost upper and lower vertical distance. If the trail denoted by *key* does not consists of subtrails, the subkey field must be ignored. In the other case, a corresponding entry for every subtrail needs to be created.

`aois.`***exp.key.(subkey.)***`transition_times` *1*x*N integer array*
If one is interested in the first AOI gazed at after a transition of stimulus, this list needs to contain the number of seconds after which each transition occurs relative to the start of the task. If this task consists of subtasks, the given times will be relative to the start of each subtask. As an effect it is not possible to perform analysis of first fixations after transition in tasks consisting of subtasks when the transitions do not occur at exactly the same time relative to the start of the subtask.

`opt.aois.first_fixation` *boolean*
Choose whether analysis of first gaze points after stimulus transition will be performed. When set to `true` there will be three extra tables included with the results: *single, five* and *timed*. The information in these tables is the same: the duration of a gaze at an AOI directly after a stimulus transition. The *single* table will only contain information about the first AOI at which is gazed after a stimulus transition. The *five* table will contain the same information, but of the five first AOIs. The *timed* table will contain this information of all fixations that happened before a given time limit (see `aois.transition_delay`).

`aois.transition_delay` *double*
The upper time limit in seconds after a stimulus transition for the start of a fixation to occur and still be included in the *timed* list.

`opt.aois.timeframe.enable` *boolean*
Choose if every trail is subdivided into time windows which will each be interpreted on their own.

`opt.aois.timeframe.length` *double*
The length in seconds that each time frame is, if the previous option is enabled.

`opt.aois.timeframe.shift` *double*
Delay in seconds each time frame and the one before it.

`aois.type` *1,2,3*
Select the method which will be used to assign the fixation points to areas of interest. There are three available option, 1, 2 and 3, which refer respectively to the classical method, the broad borders method (discouraged) and the normal distribution method.

`aois.min_qmc_samples` *integer*
The minimum number of samples that will be evaluated during the normal distribution method.

`aois.mid_qmc_samples` *integer*
If, during the normal distribution method, this number of evaluated samples is reached during the normal distribution method, the method is slightly changed to only take samples into account that help the approximate integral to converge to one.

`aois.max_qmc_samples` *integer*
If, during the normal distribution method, this number of evaluated samples is reached, the method will stop evaluating samples and display a warning that the desired accuracy could not be reached, including the accuracy it did reach.

`aois.qmc_bases` *1*x*2 integer array*
The two numbers saved in this list will be used as bases for the quasi-random Halton sample generator when using the normal distribution method. They ought to be primes and as small as possible.

`aois.qmc_error` *double*
When the relative difference between the integral of the normal distribution and the approximation by sampling reaches this value, it will stop.

4 Calibration validation options

This section describes the options for the calibration validation. The coordinate system is the same as described is section 3. In listing 4 example values can be found.

`angle` *double*
The maximum allowed visual angle in degrees which a fixation point might deviate from the calibration validation centre for it to count as valid.

`acc_threshold` *double in range 0..1*
Minimum allowed accuracy for the data to be tested. If the ratio of valid angles (see option directly above) to all recorded angles falls below this value, the program will stop and show an error.

`m` *Nx2 double array*
Every row in this matrix contains the horizontal, respectively vertical coordinate of the centre that need to be tested during calibration validation. The N points needs to be put in chronological order.

Listing 4: Example values for calibration validation options. These are set in a way that when half of the fixation points deviate more than one visual degree from the validation point, the calibration validation will not be valid. In this case five different calibration points are tested of which the last one shares its location with the first one.

```
1 opt.angle         = 1;
2 opt.acc_threshold = 0.5;
3 opt.m             = [730.5 243.5; 527.5 359.5; 528.5 493.5; 557.5 414.5; 730.5 243.5];
```

5 Plot options

Options regarding the visualisation and plotting of results are explained in this section. Again, the coordinate system described in section 3 is used. In order for all the plot functions to work, a folder named `complete/` must contain

`visualize` *boolean*
If this value is set to true a visual representation of the calibration validation and trails denoted by the keys supplied in `opt.plot` will be plot at the end of the script. If set to false, nothing will be plot.

`xmin` *integer*
The horizontal pixel from which the stimulus screen will be plotted.

`xmax` *integer*
The horizontal pixel from which the stimulus screen will no longer be plotted.

`ymin` *integer*
The vertical pixel from which the stimulus screen will be plotted.

`ymax` *integer*
The vertical pixel from which the stimulus screen will no longer be plotted.

`plotpos` *1x4 integer array*
The position at which to plot is shown on the screen. The four values are respectively the minimum x, minimum y, maximum x and maximum y values of the window that is created.

`mesh` *integer*
Accuracy with which the calibration validation ellipses are plotted. The higher the more detailed, but he slower as well.

`plot` *1xN string cell*
Collection of the keys referring to the trails that need to be plotted. If `opt.visualize` is set to `true`, a visual representation of these trail's data is shown.

`plot_AOIs` *boolean*
If set to true the used areas of interest will be shown on the plot as well. Depending on the method used, this can make the execution time for the script noticeably longer. If set to false these will not be calculated and plotted.

Listing 5: Example values for plot option. When the finishes in this case, all instances of the *housesL* trail and the two faces of the *M* trail as well as a visualisation of the calibration validation will be plotted, along the AOIs. The visual part of the stimulus screen is limited in the horizontal direction by 560 and 1360 and in the vertical direction by 300 and 900.

```
opt.visualize       = true;
opt.xmin            = 560;
opt.xmax            = 1360;
opt.ymin            = 300;
opt.ymax            = 900;
opt.plotpos         = [100, 100, 700, 650];
opt.mesh            = 100;
opt.plot            = {'housesL', 'M'};
opt.plot_AOIs       = true;
```

6 Filter options

These options concern the identification by two means clustering filtering [1]. For more information, consult `https://zenodo.org/record/159456#.WtZwfHWA6Ht`. Example values for these options are given in listing 6.

`missingx` *integer*
Missing value for horizontal position in eye-tracking data (recommended value is -xres). Used throughout functions as signal for data loss.

`missingy` *integer*
Missing value for vertical position in eye-tracking data (recommended value is -yres). Used throughout functions as signal for data loss.

`downsamples` *1xN integer array*
Collection of the N divisors of the frequency for downsampling the data. Note that if the divisor is too large, there is a risk there are too little samples for the algorithm to run. This collection may be empty.

Listing 6: Example values for filter options. For the missing values the recommended options are given. Only one small divisor is used for downsampling to prevent having too little samples.

```
opt.missingx    = -opt.xres;
opt.missingy    = -opt.yres;
opt.downsamples = [2];
```

References

[1] Hessels, R.S., Niehorster, D.C., Kemner, C. et al. *Noise-robust fixation detection in eye movement data: Identification by two-means clustering (I2MC)*, Behav Res (2017) 49: 1802. https://doi.org/10.3758/s13428-016-0822-1

Listing 3: Example values for the task options. The experiment and trail keys are the same as first described is listening 2. In this case the *UL* experiment has four voronoi AOIs and two rectangle AOIs. The *HF* experiment only has two rectangular AOIs. The *M* trail of the two subtrails, denoted by *face1* and *face2*. The *M* trail runs for sixty seconds of which *face1* takes forty-five and *face2* takes fifteen. The voronoi AOIs differ between *face1* and *face2*, but the maximum distance and rectangular AOIs don't. Nevertheless they need to be specified for every subkey. The *HF* experiment does consist of two trails. They each take thirty seconds and their AOIs span the same areas, only switched in name. The used method for AOI attribution will be the normal distribution method. The *M* task will only change of stimulus when a new subtask starts. The stimulus transitions of the two different tasks of the *HF* experiment are not identical. For the *facesL* task they happen 5, 14 and 18 seconds after the start of the trail. During the *housesL* task they happen after 0, 6.5 and 15 seconds. For the timed stimulus transition results, only the fixations happening before one second has past will be included. The trails will also be divided into different time windows of three seconds, with one second in between (so a two second overlap). The AOI determination methob will be applied to these windows as well.

```
opt.aois.UL.names.voronoi = {'right_eye', 'left_eye', 'mouth', 'nose'};
opt.aois.UL.names.rect    = {'upper', 'lower'};
opt.length.UL.M.total  = 60*opt.freq;
opt.length.UL.M.face1  = round(3*opt.length.UL.M.total/4);
opt.length.UL.M.face2  = round(opt.length.UL.M.total/4);

opt.aois.UL.M.face1.voronoi  = [872 532; 1044 532; 960 730; 960 638];
opt.aois.UL.M.face2.voronoi  = [880 530; 1038 530; 964 714; 964 628];
opt.aois.UL.M.face1.max_vor  = 125;
opt.aois.UL.M.face2.max_vor  = 125;
opt.aois.UL.M.face1.rect     = [775 1150 350 600; 775 1150 601 850];
opt.aois.UL.M.face2.rect     = [775 1150 350 600; 775 1150 601 850];
opt.aois.UL.M.transition_times = 0;

opt.aois.HF.names.voronoi = {};
opt.aois.HF.names.rect    = {'faces', 'houses'};
opt.length.HF.facesL        = 30*opt.freq;
opt.aois.HF.facesL.rect     = [635  885 475 725; 1035 1285 475 725];
opt.aois.HF.facesL.voronoi  = [];
opt.aois.HF.facesL.max_vor  = 125;
opt.aois.HF.facesL.transition_times = [0, 5, 14, 18];

opt.length.HF.housesL        = 30*opt.freq;
opt.aois.HF.housesL.rect     = [1035 1285 475 725; 635  885 475 725];
opt.aois.HF.housesL.voronoi  = [];
opt.aois.HF.housesL.max_vor  = 125;
opt.aois.HF.housesL.transition_times = [0, 6.5, 15];

opt.aois.first_fixation   = true;
opt.aois.transition_delay = 1;

opt.aois.timeframe.enable = true;
opt.aois.timeframe.length = 3;
opt.aois.timeframe.shift  = 1;

opt.aois.type             = 3;
opt.aois.min_qmc_samples  = 100;
opt.aois.mid_qmc_samples  = 1000;
opt.aois.max_qmc_samples  = 10000;
opt.aois.qmc_bases        = [2,3];
opt.aois.qmc_error        = 10^-4;
```