UBC’s

Eye Movement Data Analysis Toolkit (EMDAT)

User Manual

Version 0.9

Samad Kardan

[skardan@cs.ubc.ca](mailto:skardan@cs.ubc.ca)

Last modified September 5, 2012

Acknowledgement

Development of EMDAT has been done under the supervision of Dr. Cristina Conati.

EMDAT is partially based on Python Eye Tracking Library (PETL) developed by Nicholas FitzGerald.

Table of Contents

[1 Introduction 3](#_Toc334603912)

[1.1 Basic Concepts 3](#_Toc334603913)

[2 Input files for using EMDAT 4](#_Toc334603914)

[2.1 Files exported from Tobii Studio 4](#_Toc334603915)

[All-Data.tsv: 4](#_Toc334603916)

[Fixation-Data.tsv: 4](#_Toc334603917)

[Event-Data.tsv: 4](#_Toc334603918)

[‘.aoi’ file: 5](#_Toc334603919)

[2.2 EMDAT-specific files 5](#_Toc334603920)

[‘.seg’ file: 5](#_Toc334603921)

[Extended ‘.aoi’ file: 5](#_Toc334603922)

[3 EMDAT core functionalities 5](#_Toc334603923)

[3.1 Calculating eye gaze features 5](#_Toc334603924)

[3.1.1 Non-AOI features 5](#_Toc334603925)

[3.1.2 AOI-based features 6](#_Toc334603926)

[3.2 Preprocessing and clean up of the eye gaze data 6](#_Toc334603927)

[3.2.1 Measuring the quality of eye gaze samples in a Segment 7](#_Toc334603928)

[3.2.2 Automatic Restoration of Invalid samples 7](#_Toc334603929)

[3.2.3 Automatic Splitting of Invalid Segments 7](#_Toc334603930)

[4 EMDAT Classes 7](#_Toc334603931)

[4.1 Segment Class 7](#_Toc334603932)

[4.2 Scene Class 7](#_Toc334603933)

[4.3 AOI and AOI\_Stat classes 7](#_Toc334603934)

[4.4 Recording Class 7](#_Toc334603935)

[4.5 Participant Class 7](#_Toc334603936)

[5 Using EMDAT 7](#_Toc334603937)

[5.1 Exporting files from Tobii studio 7](#_Toc334603938)

[5.2 Generating ‘.seg’ and ‘.aoi’ files 7](#_Toc334603939)

[5.3 Changing the BasicParticipant class 7](#_Toc334603940)

[5.4 Project specific Configurations 7](#_Toc334603941)

[5.4.1 params.py 7](#_Toc334603942)

[5.5 Running EMDAT 7](#_Toc334603943)

[6 References 7](#_Toc334603944)

# Introduction

Eye Movement Data Analysis Toolkit (EMDAT) is a library for processing eye gaze data, developed in the University of British Columbia. Currently EMDAT is able to use data exported from Tobii studio software for analysis. While Tobii studio enables researchers to perform limited analysis on the eye gaze data collected by a Tobii eye tracker in an experiment, EMDAT can be used to calculate a comprehensive list of eye gaze features for each participant. Additionally, EMDAT has built-in mechanisms for data preprocessing and clean up which makes it a valuable toolkit for researchers. EMDAT is developed with generalizability in mind, so that it can be used for a variety of experiments, with no or minimal amount of changes in the code provided. In summary the main functionalities of EMDAT are:

* Calculating a comprehensive set of features for eye gaze data, including:
  + General features for the whole experiment window, and
  + Features for specific areas on the screen (a.k.a., Areas Of Interest) and transitions between them.
* Eye gaze Preprocessing and clean up, including:
  + Evaluation of the quality of eye gaze samples collected during the experiment (data validation) with different methods, and
  + Automatic restoration of certain invalid eye gaze samples and improving the quality of data used in analysis.

While there are works in the literature, that report calculating different sets of features for eye gaze data (e.g., [1]), we could not find any publicly available library that would provide the above functionalities. We hope that EMDAT enables researchers with less coding experience, to analyze their eye tracking data more comprehensively.

The rest of this manual describes how to use the EMDAT library for analyzing eye tracking data collected by a Tobii eye tracker. It is important for the reader to be familiar with eye tracking concepts and Tobii Studio software before using EMDAT.

## Basic Concepts

An eye-tracker provides eye-gaze information in terms of *fixations* (i.e., maintaining eye-gaze at one point on the screen) and *saccades* (i.e., a quick movement of gaze from one fixation point to another), which are analyzed to derive a viewer’s attention patterns. EMDAT uses a large set of basic eye-tracking features, described by [1] as the building blocks for comprehensive eye-data processing. These features are built by calculating a variety of statistics upon the basic eye-tracking measures described in Table 1.

**Table 1.** Description of basic eye tracking measures

|  |  |
| --- | --- |
| Measure | Description |
| Fixation rate | Rate of eye fixations per milliseconds |
| Number of Fixations | Number of eye fixations detected during an interval of interest |
| Fixation Duration | Time duration of an individual fixation |
| Saccade Length | Distance between the two fixations delimiting the saccade (d in Fig. 1) |
| Relative Saccade Angles | The angle between the two consecutive saccades (e.g., angle y in Fig. 1) |
| Absolute Saccade Angles | The angle between a saccade and the horizontal (e.g., angle x in Fig. 1) |

Of these measures, *Fixation rate*, *Number of Fixations* and *Fixation Duration* are widely used (e.g., [2–5]); we also included *Saccade Length* (e.g., distance *d* in Figure 1), *Relative Saccades Angle* (e.g., angle *y* in Figure 1) and *Absolute Saccade Angle* (e.g., angle *x* in Figure 1), as suggested in [1], because these measures are useful to summarize trends in user attention patterns within a specific interaction window (e.g., if the user’s gaze seems to follow a planned sequence as opposed to being scattered). Statistics such as sum, average and standard deviation can be calculated over these measures with respect to: (*i*) the full experiment window, to get a sense of a user’s overall attention; (*ii*) specific areas of interest (AOI from now on), which identify parts of the interface that are of specific relevance for understanding a user’s attention processes.

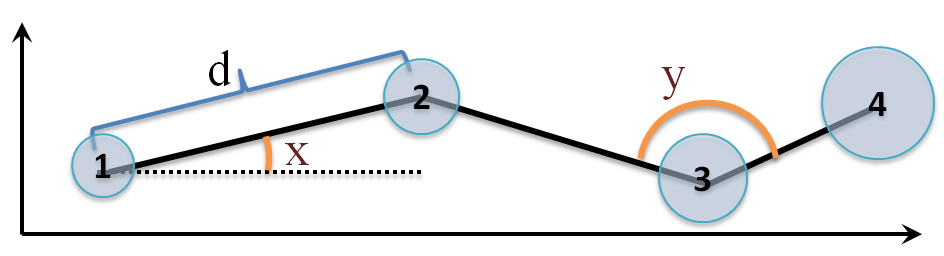


Figure - Saccade based eye measures

We are using EMDAT for different projects in our research group. As of September 2012, the following publications used EMDAT to analyze gaze data:

* the results of using EMDAT to analyze gaze data in an interactive simulation for learning is published in [6].

# Input files for using EMDAT

## Files exported from Tobii Studio

There are four types of files that are used by EMDAT, which can be directly exported from the Tobii studio software. The file types that are mandatory for every project are:

All-Data.tsv: This file contains all the data recorded by the eye-tracker. The eye-tracker samples at a constant rate (e.g., every 8 milliseconds for a Tobii T120 sampling at 120Hz), and records values for a wide variety of features including gaze location, pupil dilation etc.

Fixation-Data.tsv: The individual points of gaze-location data output in the All-Data file can be aggregated into "Fixations" according to a clustering algorithm. The algorithm groups gaze samples which are close enough in distance and time into Fixations, which have a location (centre of the corresponding gaze samples), and a duration.

**Note:** When exporting the data, the Tobii software allows you to choose one of the available methods for clustering Fixations (varies in different versions of Tobii studio). Make sure you use the same method for every subject in a given experiment, to ensure consistency.

In addition to **All-Data.tsv** and **Fixation-Data.tsv**, there are two other file types that can be exported from Tobii studio which may be needed depending on the project and the type of analysis intended:

Event-Data.tsv: This file includes all the non-gaze events, such as mouse-clicks and key-presses. It also includes all the user-defined events that are added after the experiment using Tobii studio. This is important for aligning the experimental timing data with the eye-tracking data.

‘.aoi’ file: This file contains the definition of the Areas of Interest that were defined and exported using the AOI tool in Tobii studio. AOIs are defined as a single polygon. EMDAT also supports dynamic AOIs, which are active at certain time intervals. For defining dynamic AOIs you need to use an extended version of the ‘.aoi’ file as defined in the next section.

## EMDAT-specific files

In addition to the files exported from Tobii studio, EMDAT uses the three other file types that are specific to EMDAT and needs to be created by the researcher. The mandatory file type needed for every project is:

‘.seg’ file:

The '.seg' files have lines of the form:

scene\_name<tab>segment\_name<tab>start\_time<tab>end\_time<newline>

Where *scene\_name* is the id of the Scene that this Segment belongs to, *segment\_name* is the id of the Segement, and *start\_time* and *end\_time* determines the time interval for the Segment.

Additionally, if you want to use dynamic AOIs (i.e., AOIs which are active at certain time intervals only) for your analysis, you will need to create files of the following type:

Extended ‘.aoi’ file: EMDAT supports dynamic AOIs. For defining dynamic AOIs you need to use the extended version of the ‘.aoi’ file. The extended version of '.aoi' files has pairs of lines for each AOI of the form:

aoiname<tab>point1x,point1y<tab>point2x,point2y<tab>...<newline>

#<tab>start1,end1<tab>...<newline>

where in the first line, *aoiname* is the name of the AOI, *point***i***x* and *point***i***y* define the x and y coordinates of the each vertex in the polygon that defines boundaries of the AOI. In the second line, *start***j** and *end***j** define the start and end of each time interval that this AOI is active in milliseconds. The second line starts with ’#' and is optional. If the second line does not exist the AOI will be active throughout the whole session (a.k.a. a global AOI).

Pupil dilation file: If you want to calculate pupil dilation with corresponding adjustments using pupil size in rest, a corresponding .tsv file should be provided. This file should contain rest pupil sizes for each participant for each scene. It is organized in the following format:

Header:

pid<tab><scene name 1><tab><scene name 2>....<newline>

The rest of the file is a collection of lines specified by the following format:

<pid><tab><rest pupil size 1><tab><rest pupil size 2><newline>

File with rest pupil sizes is optional. If not available, the corresponding features will be calculated with adjustments applied.

# EMDAT main functionalities

## ****Calculating eye gaze features****

EMDAT calculates a large number of features for each Segment/Scene. In this section we will list them and describe them briefly. These features are based on the eye tracking measures described in section ‎1.1. All the available features can be accessed through [get\_features()](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\Segment.html#Segment-get_features) method that returns two lists one with the feature names and another with their corresponding values (please see sections 4.1 and 4.2 for definition of the Segment and Scene classes accordingly).

### Non-AOI features

EMDAT calculates 15 general features for each Segment/Scene. These features are the only features that can be extracted without defining any AOIs. For a Segment/Scene called S these features and their meaning will be:

*'length':* Duration of S in milliseconds.

*'numfixations'*: Number of fixations in S

*'fixationrate'*: Number of fixations in S over time

*'sumfixationduration'*, *'meanfixationduration'* and *'stddevfixationduration'*: Sum, average and standard deviation for duration of fixations (in milliseconds) in S

*'sumpathdistance'*, *'meanpathdistance'* and *'stddevpathdistance'*: Sum, average and standard deviation for length of saccades (in pixels) in S

*'sumrelpathangles'*, *'meanrelpathangles'* and *'stddevrelpathangles'*: Sum, average and standard deviation for relative angles between consecutive saccades (in radian) in S

*'sumabspathangles'*, *'meanabspathangles'* and *'stddevabspathangles'*: Sum, average and standard deviation for absolute angles saccades make with the horizontal line (in radian) in S

*'eyemovementvelocity'*: length of path scanned in S over time (pixel/milliseconds) **[to be added soon]**

### In addition to general features EMDAT calculates for each S a set of measures related representing pupil dilation. Before calculating the features listed below EMDAT adjusts pupil size if a file with corresponding information is available. The adjusted value of pupil size is calculated as a difference of actual pupil size and rest pupil size.

### *'meanpupilsize' and 'stddevpupilsize': mean and standard deviation for average pupil dilation in S*

### *'maxpupilsize' and 'minpupilsize': max and min pupil dilation in S*

### *'startpupilsize' and 'endpupilsize': start and end pupil dilation in S*

### Raw and adjusted pupil sizes with corresponding timestamps can be exported in a .tsv file by using plot\_pupil\_dilation\_all() function.

### AOI-based features

In addition to general features, EMDAT also calculates a set of features for each of Segment/Scene S’s active AOIs. In order to calculate these features EMDAT needs AOI definition information (i.e., at least one ‘.aoi’ file). These features can be grouped into two categories:

·         Fixation-based features

·         Transition-based features

For an AOI called A, the fixation-based features will be:

*'numfixations'*: Number of fixations in S that are inside A

*'fixationrate'*: Number of fixations in S that are inside A over time

*'proportionnum'*: Proportion of fixations in S that are inside A over total fixations in S.

*'totaltimespent'*: Sum of the duration of fixations in S that are inside A (i.e., time spent looking inside A during S) in milliseconds.

*'proportiontime'*: Proportion of time spent looking inside A during S over total duration of S.

*'longestfixation'*: Longest fixation inside A during S.

*'timetofirstfixation'*: Time (ms) before first fixation inside A during S

*'timetolastfixation'*: Time (ms) of last fixation inside A during S

A transition is a saccade with its start fixation in one AOI and the end fixation in another AOI. The transition-based features will be calculated based on transitions from all other active AOIs in S to A. For each AOI, the following two features will be calculated for A:

*Numtransfrom<aid>*: Number of transitions from the AOI identified by *aid* to A.

*Proptransfrom<aid>*: Proportion of transitions from the AOI identified by *aid* to A over total transitions in S.

## Preprocessing and clean up of the eye gaze data

In this section, we will explain the specific concepts regarding how EMDAT handles invalid eye tracking samples and the method it uses to improve the quality of the data being analyzed.

### Measuring the quality of eye gaze samples in a Segment

The collection of eye-tracking data is prone to error. This is particularly true for eye-trackers that are not head-mounted, especially if during the experiment the movement of the participant’s head is not restricted (e.g., by a chin rest). Data can be lost due to the subject looking off the screen or due to loss of calibration from rapid movement, blinking or other such events and finally due to blocking of the infra red beam from reaching to the user’s eyes (e.g., by user’s hands).

Segments in which a large amount of data has failed to be captured could skew results to an unacceptable degree. Therefore, it is necessary to remove these Segments from analysis. There are two ways in which validity can be determined:

1. A Segment is valid as long as a certain proportion of time-samples (e.g., 90%) contain valid data.
2. A Segment is valid as long as there is no period of time (e.g., 300ms) in which there is no collected data.

Each of these has advantages and disadvantages. Using the first method will ensure that there is solid data for the whole Segment. However, it might unnecessarily screen out Segments where the data is largely valid, but there are many very small discontinuities throughout (for example if 1 time-sample in 10 is invalid but these are equally distributed throughout, this should not be a problem). The second method will ensure that there are no large gaps in the data for a given Segment.

In the following two sections we will describe two different ways that EMDAT uses to improve the quality of the gaze data samples in Segments.

### Automatic Restoration of Invalid samples

EMDAT can optionally look for any restorable invalid eye gaze samples and add them to the valid samples. “Restored samples” are samples that are initially deemed invalid (e.g., the red star in Figure 2), but which can be restored to be part of a Fixation (e.g., the green boxes in Figure 2). The rationale for such restorations is as follows: if the user was fixating at the same point before and after a short period of “lost” gaze data, it can be assumed that the user was looking at that same point during this “loss” period.

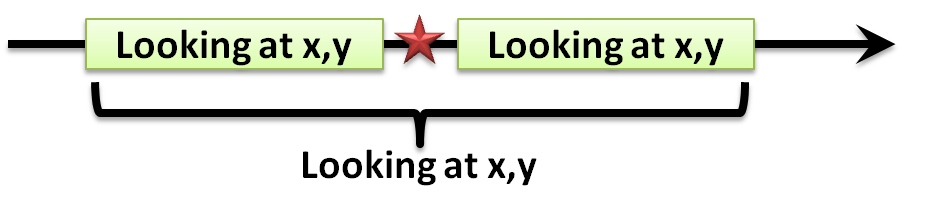


Figure - Recovering invalid samples using fixation data

### Automatic Splitting of Invalid Segments

**EMDAT comes with an additional option for improving the quality of overall data used in the analysis by splitting a Segment that does not meet the minimum quality threshold because of a large continues gap of invalid samples in it (e.g., Segment1 in** Figure 3**, the invalid samples are shown with red starts). The auto-partition option enables EMDAT to automatically split Segments of low sample quality into two new sub-Segments (e.g., Segments 11 and 12 in** Figure 3**), and discarding the largest gap of invalid samples for that Segment. EMDAT will continue to perform the splitting on the Segments until there is no gap larger than MAX\_SEG\_TIMEGAP parameter left in the data (see section 5.1 for description of parameters).**



Figure - Automatic splitting of an invalid Segment into two Segments with higher quality

# EMDAT’s main classes

All of the EMDAT files have been documented according to Google’s documentation style for Python[[1]](" \l "_ftn1" \o "). You can access the description of the methods and attributes of each class by clicking on the links provided in the text.

## Segment Class

A [Segment](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\segment.html) is a class that represents the smallest unit of aggregated eye data samples with a conceptual meaning related to the experiment attached to it by the experimenter (e.g., performing a sub task, the interval in which user was looking at the screen uninterrupted, etc.). This class is the equivalent of segments as defined in Tobii studio.

## Scene Class

A [Scene](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\Scene.html#Scene) is a class that represents one scene (e.g., when certain visual elements are present at the screen) in the experiment. The Scene is designed to aggregate [Segment](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\segment.html)s related to a target conceptual entity or activity in the experiment. A Scene should have at least one [Segment](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\segment.html) assigned to it. From the technical point of view, the [Scene](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\scene.html) class is used to combine multiple [Segment](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\segment.html)s and calculate the aggregated features for this new entity as a whole. This class is the equivalent of a scene as defined in Tobii studio.

While Scene and Segment classes have many similarities (e.g., they share a lot of features), it is important to note that they have different intended purposes. The Segment class covers one basic continuous interval of eye gaze samples and calculates the features for that interval. The Scene class enables the researcher to look at the eye gaze patterns in a higher level of abstraction (e.g. reaction of user whenever a specific event happens on the screen) rather than looking at small pieces of information provided by the Segments. For example, if a researcher is interested in seeing how participants react to a specific visual element (e.g., a virtual agent) that appears under certain conditions on the screen, each instance of its appearance will be covered by a Segment, whereas the overall picture is captured by a Scene that includes all the relevant Segments. As another example, if two activities of interest are happening across two separate Scenes, their respective Segments enable the analysis of these two activities separately even if in some cases they overlap (see Figure 4).

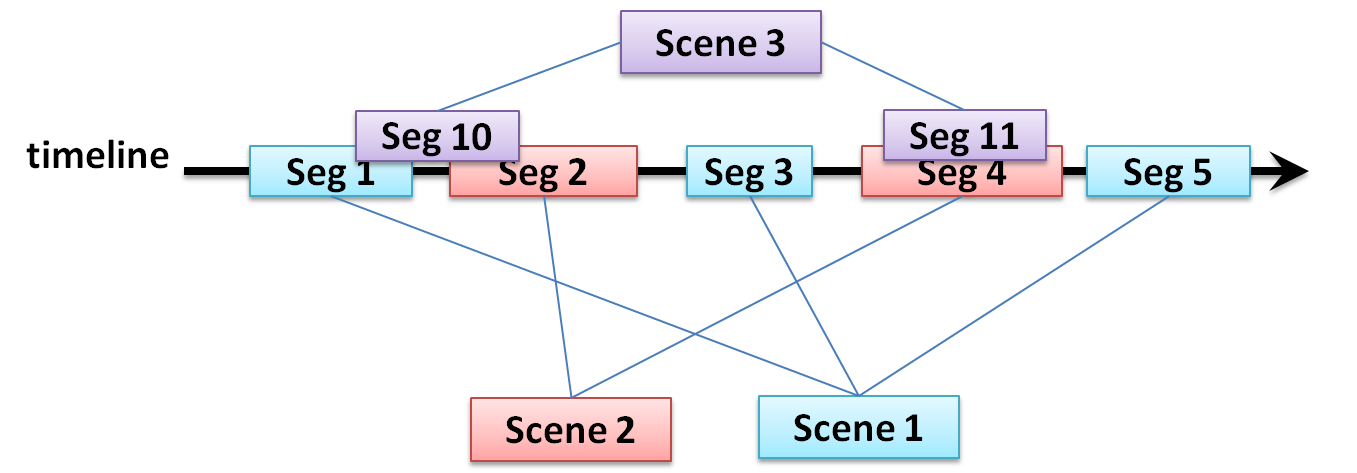


Figure - An example of overlapping Scenes

## AOI and AOI\_Stat classes

In EMDAT, the boundary of an Area of Interest ([AOI](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\AOI.html)) is defined as a polygon on the screen. You can optionally define a second polygon inside the first polygon to be excluded from the [AOI](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\AOI.html) when initializing an AOI (see Figure 2). Please note that the second polygon is not supported in the ‘.aoi’ file format.

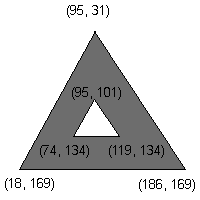


Figure - A sample AOI with outer and inner polygons

An [AOI](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\AOI.html) can be always active (a global [AOI](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\AOI.html)) or can be active during certain time intervals (a dynamic AOI). If there are dynamic AOIs in your experiment you will need to use the extended version of ‘.aoi’ files for your project as explained in section 4.2.

In order to calculate the features for an [AOI](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\AOI.html) instance, you need to create an [AOI\_Stat](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\AOI.html#AOI_Stat) instance and map it to a target AOI object by passing it to the [AOI\_Stat](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\AOI.html#AOI_Stat) constructor. The resulting [AOI\_Stat](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\AOI.html#AOI_Stat) will calculate all features related to the given [AOI](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\AOI.html) and store them for later reference.

## Recording Class

[Recording](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\Recording.html) is a class used to hold all the data from one [Recording](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\Recording.html) (i.e., one complete experiment session) for one participant. It also has some useful functions for reading different types of files.

## Participant Class

[Participant](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\Participant.html) is the parent class for a project specific class that holds the information for one Participant in the experiment. There are some project specific functionalities that should be overridden by the child class in order to use EMDAT. This class is the only component of EMDAT that is not fully implemented by design in order to address the needs of different projects. The following two methods cover the project specific aspects of EMDAT:

**Partition: This method is responsible for generating the list of Scenes and their corresponding Segments. Please see the** [BasicParticipant](file:///C:\Temp\skardan\workspace\python\EMDAT\src\doc\BasicParticipant.html) **class for a sample implementation of this function that reads ‘.seg’ files and generates the Scene list.**

**read\_participants****: This method is responsible for generating the full name of the files used for each participant (e.g., the full name of the ‘All-Data.tsv’ file for the first participant is ‘P1\_All-Data.tsv’ and so forth). Then it generates a list of the Participant objects by passing the file names to the Participant’s constructor method.**

# Using EMDAT

## Exporting files from Tobii studio

As explained in Chapter 2, in order to use EMDAT on your data, you will need to export ‘**All-Data.tsv’**, ‘**Fixation-Data.tsv’** and **‘Event-Data.tsv’** files for each participant in your experiment. Please refer to your Tobii studio manual for details of how this can be done.

## Generating ‘.seg’ and ‘.aoi’ files

The other two types of information that may be needed by EMDAT are the definition of AOIs (‘.aoi’ file) and the definition of Scene and Segments (‘.seg’ files).

If you are using AOIs that are active throughout the experiment for one user, then you can easily define and export AOIs using AOI tool in Tobii studio. However, if your AOIs are active only in certain time intervals for each user, you may need to create customized ‘.aoi’ files for each user according to the extended ‘.aoi’ format as explained in section 2.2.

When it comes to defining Scene and Segments, depending on the experiment design you can have a very basic one line ‘.seg’ file for each user such as:

MainScene<tab>Seg1<tab>0<tab>120000<newline>

This line defines one Scene with only one Segment that covers the whole experiment for the user (120 seconds in the above example). You can also have more complex set of Scenes with multiple Segments or even Scenes with overlapping time intervals. All these different designs are supported by the ‘.seg’ file format described in Section 2.2.

## Changing the BasicParticipant class

The EMDAT distribution includes an example implementation of the Participant class, called BasicParticipant. This class implements a typical scenario where all the input files are provided and the experiment only includes general AOIs. You can directly use this code for your project if you have a similar use case. The only change that may be necessary is in the **read\_participants method where the full names of the files are generated.**

**If you have a more complicated scenario** you will need to implement a more complicated **Partition method that generates the Scene list based on the ‘Events-Data.tsv’ files and/or some external log files generated by the software that was under study. It is important to note that you may need to add a mechanism to generate ‘.aoi’ files for your dynamic AOIs as well.**

## Project specific Configurations

EMDAT uses a large number of parameters that enables the researchers to customize different aspects of the process of calculation of features without any need for any changes in the code. In this section we will look at the list of the parameters EMDAT uses and their meaning.

### params.py

Each project should assign a value to the following parameters that are found in the params.py module. While most of the parameters can be left to their default current values, it is important that a researcher goes through all of them and ensures that the behaviour dictated by these values is what s/he is expecting from EMDAT.

|  |  |
| --- | --- |
| Parameter | Description |
| EYELOGDATAFOLDER | the folder that has the files exported from Tobii |
| EXTERNALLOGDATAFOLDER | the folder that has the external log files (if any) |
| NUMBEROFEXTRAHEADERLINES | number of extra lines at the beginning of the files exported from Tobii  this is specific to the experiment and is based on the number of control variables defined in Tobii studio for the experiment (e.g., age, vision, etc.) and is recorded at the beginning of the files for each participant |
| FIXATIONHEADERLINES | number of lines at the beginning of the 'Fixation-Data' files exported from Tobii before the actual data |
| ALLDATAHEADERLINES | number of lines at the beginning of the 'All-Data' files exported from Tobii before the actual data |
| EVENTSHEADERLINES | number of lines at the beginning of the 'Event-Data' files exported from Tobii before the actual data |
| MEDIA\_OFFSET | The coordinates of the top left corner of the window showing the interface under study. (0,0) if the interface was in full screen (default value) |
| featurelist | A list of non-AOI features. It should be a subset of the following list:  [*'numsegments'*,*'length'*,*'numfixations'*,*'fixationrate'*,    *'meanfixationduration'*,*'meanpathdistance'*,*'meanrelpathangles'*,*'meanabspathangles'*,    *'stddevabspathangles'*,*'stddevfixationduration'*, *'stddevpathdistance'*,*'stddevrelpathangles'*,    *'numsamples'*,*'sumabspathangles'*,*'sumfixationduration'*,  *'sumpathdistance'*,*'sumrelpathangles'*]  *'meanpupilsize', 'stddevpupilsize', 'maxpupilsize', 'minpupilsize', 'startpupilsize','endpupilsize'* |
| aoinames | list of the AOI names |
| aoigeneralfeat | A list of general AOI features. It should be a subset of the following list:  [*'fixationrate'*,*'numfixations'*,*'totaltimespent'*,  *'proportionnum'*,*'proportiontime'*,  *'longestfixation'*,  *'timetofirstfixation'*,*'timetolastfixation'*] |
| aoitransfrom | A list of frequency-based features for transitions between AOIs. It is populated automatically based on the aoinames. |
| aoiproportion | A list of proportion-based features for transitions between AOIs. It is populated automatically based on the aoinames. |
| aoifeaturelist | a list of all AOI-based features which is populated automatically based on aoigeneralfeat, aoitransfrom and aoiproportion. |
| VALID\_PROP\_THRESH | the minimum proportion of valid samples for a Segment or Scene to be considered valid |
| VALID\_TIME\_THRESH | the maximum gap size (ms) allowable in samples for a Segment or Scene to be considered valid |
|  |  |
| VALIDITY\_METHOD | 1: porportion; 2:time gap; 3: porportion with (valid + restored) samples |
|  |  |
| MAX\_SEG\_TIMEGAP | maximum gap size (ms) allowable in a segment with auto-partition option |
| MINSEGSIZE |  |
| INCLUDE\_HALF\_FIXATIONS | A Boolean value determining if a Fixation extends between two consecutive Segments, should it be included in those Segments or not |
| DEBUG |  |
| NONTEMP\_FEATURES\_SEG, NONTEMP\_FEATUES\_PUPIL and NONTEMP\_FEATURES\_AOI | Three predefined lists of general, pupil related and AOI-based features that are not accumulative and are controlled for time spent |

## Running EMDAT

In order to run EMDAT you will need a Python version 2.6.\* or 2.7.\* distribution [[2]](#_ftn1) with the NumPy [[3]](#_ftn1) package. EMDAT has been developed and tested on Python version 2.6.7 and should be working on any 2.6 or 2.7 versions of Python.

After implementing a subclass of the Participant class with the required methods (e.g. changing the BasicParticipant class), you will need a script to generate a list of Participant objects and then extract the features for them. The module testBasic.py contains an example of such script that runs EMDAT for two sample participants and writes the feature vales in a tab-separated format in ‘smaple\_features.tsv’ file.

# References

1. Goldberg, J.H., Helfman, J.I.: Comparing Information Graphics: A Critical Look at Eye Tracking. Proceedings of the 3rd BELIV’10 Workshop: BEyond time and errors: novel evaLuation methods for Information Visualization. pp. 71–78. ACM, Atlanta, GA, USA (2010).

2. Hegarty, M., Mayer, R.E., Monk, C.A.: Comprehension of Arithmetic Word Problems: A Comparison of Successful and Unsuccessful Problem Solvers. Journal of Educational Psychology. 87, 18–32 (1995).

3. Canham, M., Hegarty, M.: Effects of knowledge and display design on comprehension of complex graphics. Learning and Instruction. 20, 155–166 (2010).

4. Jarodzka, H., Scheiter, K., Gerjets, P., van Gog, T.: In the eyes of the beholder: How experts and novices interpret dynamic stimuli. Learning and Instruction. 20, 146–154 (2010).

5. Loboda, T.D., Brusilovsky, P.: User-adaptive explanatory program visualization: evaluation and insights from eye movements. User Modeling and User-Adapted Interaction. 20, 191–226 (2010).

6. Kardan, S., Conati, C.: Exploring Gaze Data for Determining User Learning with an Interactive Simulation. In: Masthoff, J., Mobasher, B., Desmarais, M., and Nkambou, R. (eds.) User Modeling, Adaptation, and Personalization. pp. 126–138. Springer Berlin / Heidelberg (2012).

[[1]](#_ftnref1) Available at [http://google-styleguide.googlecode.com/svn/trunk/pyguide.html#Comments](http://google-styleguide.googlecode.com/svn/trunk/pyguide.html%23Comments)

[[2]](#_ftn2) Available at <http://www.python.org/getit/>

[[3]](#_ftn3) Available at <http://www.scipy.org/Download>