EMDAT – User Manual

*Samad Kardan, Sébastien Lallé, Dereck Toker, and Cristina Conati. 2021. EMDAT: Eye Movement Data Analysis Toolkit (2.x). The University of British Columbia. DOI: 10.5281/zenodo.4699774*

Manual generated by Sébastien Lallé on June 17, 2021.

Latest update: June 30, 2023.

Contents

[1. Basic Concepts 1](#_Toc118071952)

[2. Features generated by EMDAT 2](#_Toc118071953)

[3. Requirements 5](#_Toc118071954)

[4. Applying EMDAT 5](#_Toc118071955)

[4.1 Input Data 6](#_Toc118071956)

[4.1.1 Eye-tracker export data 6](#_Toc118071957)

[4.1.2 Experiment Metadata 7](#_Toc118071958)

[4.1.3 Input file structure 9](#_Toc118071959)

[4.2 Configure EMDAT 10](#_Toc118071960)

[4.2.1 Params.py 10](#_Toc118071961)

[4.2.2 basicparticipants.py 14](#_Toc118071962)

[4.2.2 Entry Point 15](#_Toc118071963)

The **Eye Movement Data Analysis Toolkit (EMDAT)** is a Python library for processing eye gaze data, developed at the University of British Columbia. Currently EMDAT supports data exported from Tobii Studio, Tobii Lab and SMI BeGaze, albeit additional data formats can be added as needed. EMDAT can be used to calculate a comprehensive list of eye gaze features from eye tracking recordings. 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 and eye trackers. For that reason EMDAT is used as a Python-based library that can be run on its own, and/or integrated into existing data analysis pipeline. In summary the main functionalities of EMDAT are:

* Calculating a comprehensive set of features for eye tracking 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.

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 before using EMDAT. Notice that there is a Developer Manual for EMDAT that is targeted toward developers who need to modify EMDAT or integrate it in their pipeline.

* Link to the root of the GitHub repository: <https://github.com/ATUAV/EMDAT>
* Link to the latest stable release (2.0): <https://github.com/ATUAV/EMDAT/releases/tag/2.0>

[Note: The first stable version (1.0) was developed with Python 2.7. Version 2.x have been ported to Python 3]

# 1. Basic Concepts

An eye-tracker provides eye-gaze information in terms *gaze samples* captured at the eye-tracker sampling frequency, e.g., a 120 Hertz (Hz) eye-tracker registers a gaze sample every 8.33 milliseconds (ms). A gaze sample includes the (x,y) coordinates of the gaze position on the screen. Several eye-trackers also provides information about the size of the pupil (pupillometry) and the distance from the eyes to the screen. The gaze samples are typically collected and saved either via an API or a third-party data collection software such as Tobii Studio or SMI BeGaze (as we elaborate more in Section 4.1).

Most eye-tracking data collection software and tools process the gaze samples into *fixations* (i.e., maintaining eye-gaze at one point on the screen, obtained by clustering together nearby gaze samples) 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. Most eye-tracking data collection software can generate these information.

A *recording* includes all of the eye-tracking data collected in a single session, i.e., from the start of the tracking to the end. In eye-tracking-based experiments, there can either be one recording per participant (i.e., the entire gaze data are collected all at once for a given participant); or several recordings per participant. Multiple recordings per participants can be necessary for instance if the experiments span across several days, or to allow for breaks during the study sessions. Often, recordings are divided into smaller units of time called *Scenes* (or *Segments*), which are meant to capture the meaningful parts of the recording, or the different tasks the participant must undergo.

Gaze samples can be tracked over the entire screen (by default), or over specific Areas of Interests (AOIs) that capture salient areas of the screen.

In EMDAT, the gaze samples, fixations and scenes are required for each recording. Saccades, pupil size, distance of the head to the screen and AOIs are optional, as we will describe in Section 4.1.

# 2. Features generated by EMDAT

EMDAT generates several sets of features for each of the scenes defined for each recording. This section defines all of these features, which are for the most part standard in eye-tracing data analysis and have been extensively used in research (see “*Eye Tracking: A comprehensive guide to methods and measures”* fromHolmqvist et al. [1] for a broad overview).

**Fixation-based features.**

|  |  |
| --- | --- |
| **Feature** | **Definition** |
| numfixations | Total number of fixations |
| fixationrate | Total number of fixations / duration of the recording or scene[[1]](#footnote-1) |
| sumfixationduration  meanfixationduration  stddevfixationduration | Summative features (sum, mean, and standard deviation) over the duration of the fixations |

**Saccade-based features.**

EMDAT capture saccades in terms of:

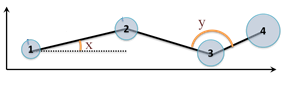
1. A straight line between two fixations (called *path* in EMDAT, or *straight saccade*), inferred from the set of fixations.
2. The actual trajectory formed by the gaze samples of the saccade (called *saccade* or *curved saccade* in EMDAT).

|  |  |
| --- | --- |
| **Feature** | **Definition** |
| sumpathdistance  meanpathdistance  stddevpathdistance | Summative statistics over the length[[2]](#footnote-2) of the *path*s |
| eyemovementvelocity | Mean speed (distance / duration) of the *path*s |
| numsaccades  sumsaccadedistance  meansaccadedistance  stddevsaccadedistance  longestsaccadedistance | Summative statistics over the length of the saccades |
| sumsaccadeduration  meansaccadeduration  stddevsaccadeduration  longestsaccadeduration | Summative statistics over the duration of the saccades |
| meansaccadespeed  stddevsaccadespeed  minsaccadespeed  maxsaccadespeed | Summative statistics over the speed of the saccades (distance / duration) |
| fixationsaccadetimeratio | Total fixation duration / total saccade duration |

**Angles-based features.**

There are two types of angles measured in EMDAT:

1. The angle between the two consecutive straight saccades, called *relative saccade angle*. This is the *y* angle on the Figure below.
2. The angle between a straight saccade and the horizontal, called *absolute saccade angle*. This is the *x* angle on the Figure below.

****

|  |  |
| --- | --- |
| **Feature** | **Definition** |
| meanabspathangles  stddevabspathangles  abspathanglesrate  sumabspathangles | Summative statistics over the absolute path angles |
| meanrelpathangles  stddevrelpathangles  relpathanglesrate  sumrelpathangles | Summative statistics over the relative path angles |

**Pupil-based features.**

Most eye-trackers measure size (diameter) of the pupil at each gaze sample.

|  |  |
| --- | --- |
| **Feature** | **Definition** |
| meanpupilsize  stddevpupilsize  maxpupilsize  minpupilsize | Summative statistics of the pupil size over all gaze sample. |
| startpupilsize  endpupilsize | Pupil size at the start and end of the recording or scene |
| meanpupilvelocity stddevpupilvelocity maxpupilvelocity minpupilvelocity | Summative statistics over the speed of the change in pupil size |

**Head distance-based features.**

Most eye-trackers measure the distance of the eyes to the screen at each gaze sample. In EMDAT the distance of each eye is averaged before computing the features. They are simply called *distance* (future iterations should rename them to *headdistance*).

|  |  |
| --- | --- |
| **Feature** | **Definition** |
| meandistance stddevdistance maxdistance  mindistance | Summative statistics of the head distance over all gaze sample. |
| startdistance enddistance | Head distance at the start and end of the recording or scene |

**Blink-based features.**

Some eye-trackers can automatically detect and export blinks. If not, EMDAT includes a basic blink detector, with blink being defined as a temporary loss of data within specific time thresholds defined as a parameter (see Section 3.2.1).

|  |  |
| --- | --- |
| **Feature** | **Definition** |
| blinknum | Total number of blinks |
| blinkdurationtotal blinkdurationmean blinkdurationstd  blinkdurationmin  blinkdurationmax | Summative statistics over the duration of all blinks |
| blinkrate |  |
| blinktimedistancemean blinktimedistancestd  blinktimedistancemax blinktimedistancemin | Summative statistics over the intervals between successive blinks |

**Interaction-based features.**

These features are only for eye-trackers that also record the interaction events, namely mouse clicks and keyboard strokes.

|  |  |
| --- | --- |
| **Feature** | **Definition** |
| numevents numleftclic numrightclic numdoubleclic numkeypressed | Total number of events, mouse clicks, and key presses |
| leftclicrate rightclicrate doubleclicrate  keypressedrate | Rate of mouse and keyboard events |
| timetofirstleftclic timetofirstrightclic timetofirstdoubleclic timetofirstkeypressed | Timestamp of the first event |

**AOI-based features.**

Most of the above features can be defined over the subset of gaze samples, fixations and saccades that fall within the boundaries of user-defined AOIs. Importantly, all AOI-based features are generated for each AOI defined by the user. Apart from the features defined above, EMDAT can also generate the following features, again for each AOI:

|  |  |
| --- | --- |
| Feature | Definition |
| aoitransfrom | Number of transition from the AOI to each of the defined AOIs (including itself). A transition is a saccade with its start fixation in one AOI and the end fixation in another AOI. |
| aoiproportion | Proportion of all transitions from the AOI to each of the defined AOIs, out of all transitions for the AOI. For example for AOI\_1: if there are 4 transitions from AOI\_1 to AOI\_2 and 20 transitions in total from AOI\_1 to the other AOIs, the aoiproportion for AOI\_1 is *4 / 20 = 0.2* |
| aoisequence | The list of all AOI visits (an AOI is visited if at least one fixation falls into it). For instance if a user’s fixations falls into AOI\_1, then AOI\_2 and finally AOI\_1 again, the final sequence is [AOI\_1, AOI\_2, AOI\_1] |

**Other**

|  |  |
| --- | --- |
| Feature | Definition |
| numsegments | Number of segment in the recording or scene |
| length | Length of the recording or scene |
| numsamples | Number of gaze samples |
| blinktimedistancemean blinktimedistancestd  blinktimedistancemax blinktimedistancemin | Summative statistics over the intervals between successive blinks |

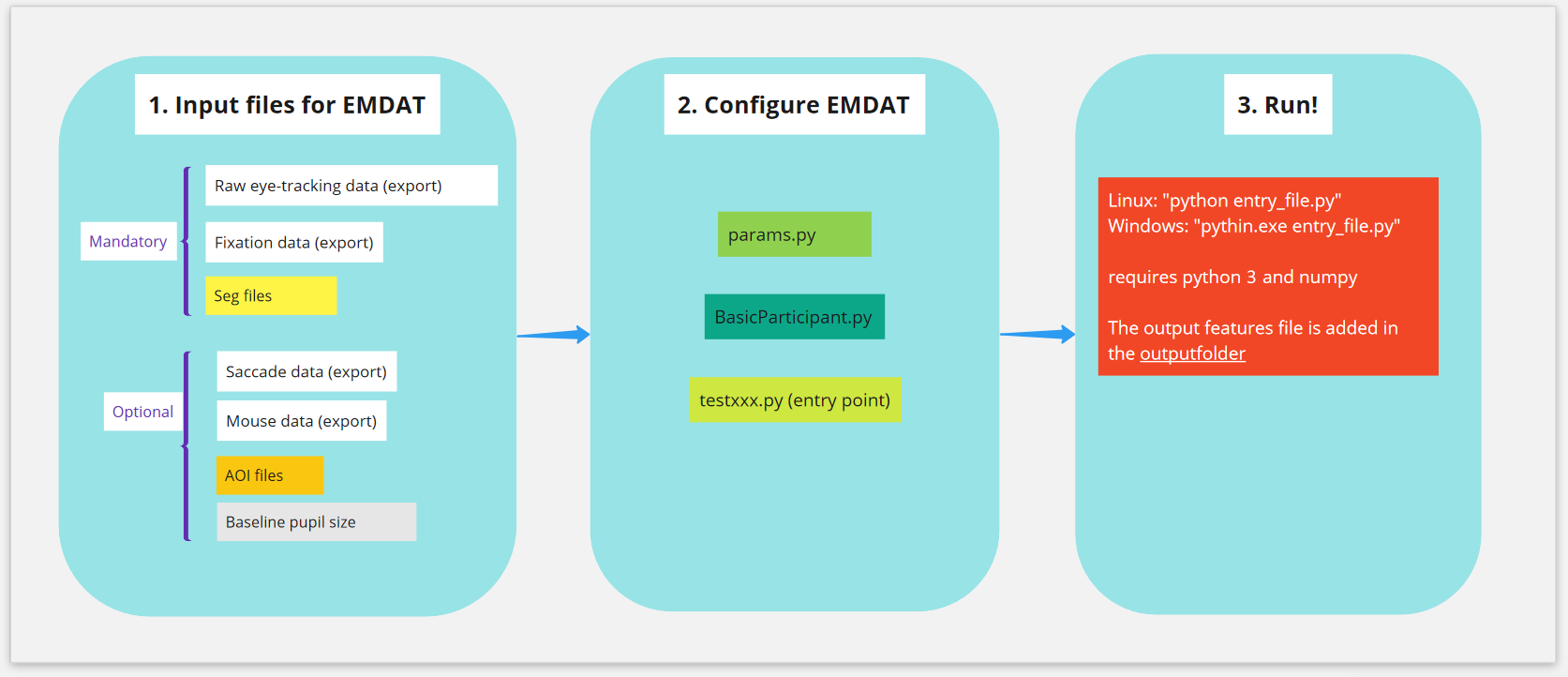
# 3. Requirements

EMDAT (2.x) requires Python 3, numpy and multiprocessing. It works on all OS. It was developed and tested with Python 3.9.

Older versions of EMDAT (1.0) run on Python 2.7.

# 4. Applying EMDAT

There are three main stages required to apply EMDAT to a dataset, summarized in the chart below and described in the next subsections.



## 4.1 Input Data

Several files are inputted into EMDAT for each recording. This means that the input files listed below must be exported/generated for each recording.

### 4.1.1 Eye-tracker export data

EMDAT is meant to process eye-tracking data exported by an eye-tracking data collection software, and require at least the gaze samples and fixations. Currently EMDAT supports several data export format for the following data tracking software:

* *Tobii Studio 2.x*, *Tobii Studio 3.x*, and *Tobii Lab 1.x*, provided by Tobii for Tobii Pro eye-trackers
* *SMI BeGaze 3.x*, that is provided for SMI eye-trackers

EMDAT can be extended to support other data format (see Developer Manual).

The following files can be generated by the above software (where “XX” is a prefix that can be customized in the software) per recording:

* For *Tobii Studio 2.x:*
  + XX-All-Data.tsv: Contains all the gaze samples recorded by the eye-tracker for a given recording.
  + XX-Fixation-Data.tsv: The Fixations generated by Tobii Studio for the recording. The actual algorithm used by Tobii Studio can be selected in the options.
  + XX-Event-Data.tsv (optional): This file includes all the non-gaze events, such as mouse-clicks and key-presses. It also includes all the user-defined events.
* For *Tobii Studio 3.x* and *Tobii Lab 1.x:*
  + *XX-Data\_Export.tsv*: Contains all gaze samples, fixations, saccades, and non-gaze events for a given recording.
* *For SMI BeGaze 3.x:*
  + *XX\_Samples.txt: Contains the gaze samples*
  + *XX\_Events.txt: Contains the fixations, saccades and blinks*

When exporting the above data, it is important to export them \*per recording\* (and not merge all recording together into one file).

### 4.1.2 Experiment Metadata

EMDAT uses two other input data that includes context information about the eye-tracking data.

**a) Scenes and Segments Files:** EMDAT’s features are generated over user-defined scenes to capture specific parts of the interaction. This is in particular to enable analysis of different phases or tasks of experiments. A scene is composed of a list of Segments (at least one) that are defined by a start and end timestamp. The segments of different scenes can overlap.

Scenes and Segments are defined in tab-separated file without header, usually with a “.seg” extension. The format of a line in the Scene and Segment file is:

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

Where *scene\_name* is the unique name (or ID) of the Scene that this Segment belongs to, *segment\_name* is the unique name/ID of the Segment for that scene, and *start\_time* and *end\_time* determines the time interval for the Segment.

Example with three scenes and 7 segments:

Scene\_1 seg\_1 100 200

Scene\_1 seg\_3 600 700

Scene\_1 seg\_5 1100 1200

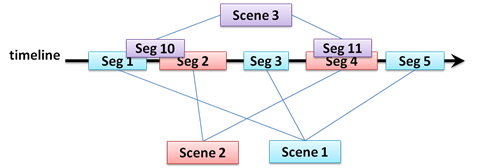
Scene\_2 seg\_2 300 500

Scene\_2 seg\_4 800 1000

Scene\_3 seg\_10 170 350

Scene\_3 seg\_11 850 950

Visual representation of these scenes:



A Segment can also be shared among two or more scenes, for example here seg\_5 is shared by two scenes:

Scene\_1 s eg\_1 600 700

Scene\_1 seg\_5 1100 1200

Scene\_2 seg\_2 300 500

Scene\_2 seg\_5 1100 1200

Importantly, the features listed above in Section 2 are generated for each scene.

Note: In case scenes and segments are not required, it is sufficient to define one Scene and Segment per recording for which the start time and end time captures the entire interaction. In that case the features are generated over the entire recordings. For instance:

Scene\_all seg\_all 0 999999999999

One segment file is required per recording.

**b) Areas of Interests (AOIs) files (optional):** EMDATgenerates by default features over all of the eye-tracking data captured in a scene. In addition, ENDAT can generates features for gaze samples that fall within specific AOIs. An AOI is defined in EMDAT as a set of polygons (list of x,y coordinates on the screen). EMDAT supports both static and dynamic AOIs, as follows.

**Static AOI** are active throughout the entire recording. They are define in tab-separated files without header, typically with a “.aoi” extension, as follows:

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

For instance, the following file defines two rectangular AOI:

Aoi\_1 0,0 10,0 10,10 0,10

Aoi\_2 50,30 50,50 80,50 80,30

AOIs can overlap. The shape of an AOI can be any polygons, by appending the list of x,y coordinates. For examples:

Aoi\_3 0,0 10,3 10,7 10,10 10,20 20,20 20,17 20,5

An AOI can also be defined as multiple polygons, so as to capture different parts of the screen at once. To do so each polygon should be defined in a new line with the same name, e.g.:

Aoi\_4 0,0 10,0 10,10 0,10

Aoi\_4 50,30 50,50 80,50 80,30

Aoi\_4 0,0 10,3 10,7 10,10 10,20 20,20 20,17 20,5

In this example Aoi\_4 is defined as a set of 3 polygons.

**Dynamic AOIs** are defined in a similar way than static AOIs (i.e., one or multiple polygons), but are in addition active only at specific times. The active periods are defined for each dynamic AOI by adding a second line with the list of active periods (defined via their start and end timestamp). The format of a dynamic AOI is:

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

#<tab>start1,end1<tab>start2,end2<tab>...<newline>

For instance, a dynamic AOI with two activation periods:

Aoi\_1 0,0 10,0 10,10 0,10

# 1000,5000 7000,100000

It is possible to define a dynamic AOI with multiple polygons in case the shape of the AOI changes over time. For instance:

Aoi\_1 0,0 10,0 10,10 0,10

# 1000,5000

Aoi\_1 0,0 10,3 10,7 10,10 10,20 20,20 20,17 20,5

# 11000,15000 17000,21000 29000,45000

Static and dynamic AOIs can be defined together in a same AOI file. AOIs can be defined either globally or per recording. Global AOIs are shared across all recordings and are useful when the stimuli do not change across recordings. In that case only one AOI file is required overall. Otherwise AOIs can be defined per recording, and in that case one AOI file must be generated for each recording.

**Important note:** Currently the same unique AOI names must be defined for ALL recordings when using non-global AOI. If names are mismatch across recording EMDAT may generate incorrect output features! This is a limitation to be addressed in later version. Currently, if some AOIs are not required in all recordings, the issue can be tackled by defining them with (0,0) coordinates, e.g.:

Aoi\_recordpecific 0,0 0,0 0,0

**c) Baseline pupil size (optional).** To account forphysiologicaldifferences or changes in lightning, pupil sizes are often normalized. EMDAT offers several way to perform such normalization, one of them being based on baseline pupil sizes collected during the experiment. Baseline pupil sizes are typically collected by having the participants staring at a unicolor screen, so as to measure their pupil size when they are not visually nor mentally stimulated. Importantly, EMDAT can still perform normalization even without such baseline pupil sizes (as we will show in Section 3). Baseline pupil sizes are defined for all recordings in a tab-separated file, for each Scene defined in the Scene and Segment file for that recording. The format is:

recording\_name<tab>baseline\_size<tab>baseline\_size<tab>…<newline>

The header of this file should be:

pid<tab>Scene\_name<tab>Scene\_name<tab>…<newline>

For instance:

Pid Scene\_a Scene\_b

Rec\_1 3 4

Rec\_2 5 6

Note: This format might not be ideal for all settings and could be improved in later iteration of EMDAT.

### 4.1.3 Input file structure

There is no required file structure on the HDD for the input files as the paths will be manually configured in the next section. As a general guideline the following structure can be used:

+---input\_data

| +---eyetracker\_data\_export

| +---scenes\_segments

| +---aois

| +---baseline\_pupil\_size

## 4.2 Configure EMDAT

EMDAT is meant to be integrated into data analysis flow via flexible and customizable scripting. To allow such integration EMDAT requires configuration of three files, as shown in Fig. 1.

### 4.2.1 Params.py

This file includes the main parameters used by EMDAT. All parameters come with default values, that can be changed as needed. The list of parameters is defined next.

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| **Eye tracker type and path** | |
| EYELOGDATAFOLDER | Path of the folder that has the files exported from Tobii  Default is ./sampledata |
| EXTERNALLOGDATAFOLDER | the folder that has the external log files (*not supported anymore*) |
| EYETRACKERTYPE | The type of eye-tracker file format. Currently the supported type are “TobiiV2”, “TobiiV3”, and “SMI” (see Section 1) |
| **Eye tracker specific parameters (depend on the EYETRACKERTYPE)** | |
| NUMBEROFEXTRAHEADERLINES | **TobiiV2 only.** Number of extra metadata lines at the beginning of the files exported from Tobii Studio 2.x  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. Default is 8 (with the basic export) |
| FIXATIONHEADERLINES | **TobiiV2 only.** number of metadata lines at the beginning of the 'Fixation-Data' files exported from Tobii before the actual data. Default is 19 (with the basic export) |
| ALLDATAHEADERLINES | **TobiiV2 only.** number of metadata lines at the beginning of the 'All-Data' files exported from Tobii before the actual data. Default is 26 (with the basic export) |
| EVENTSHEADERLINES | **TobiiV2 only.** number of metadata lines at the beginning of the 'Event-Data' files exported from Tobii before the actual data. Default is 27 (with the basic export) |
| EVENTS\_FIRST\_DATA\_LINE | **SMI only.** the line number of the first data row in Events file. Default is 22 (with the basic export) |
| FIXATION\_HEADER\_LINE | **SMI only.** the line number of the row that contains the table header for fixations. Default is 11 (with the basic export) |
| SACCADE\_HEADER\_LINE | **SMI only.** the line number of the row that contains the table header for fixations. Default is 14 (with the basic export) |
| USER\_EVENT\_HEADER\_LINE | **SMI only.** the line number of the row that contains the table header for user events. Default is 20 (with the basic export) |
| RAW\_HEADER\_LINE | **SMI only.** the line number of the first data row in Raw file. Default is 1 (with the basic export) |
| MONOCULAR\_EYE | **SMI only.** L or R for using left/right eye event when averaging both eyes measures is not possible. Default is “L” |
| **Features generation parameters** | |
| MEDIA\_OFFSET | The (x,y) coordinates of the top left corner of the window showing the interface under study. This is to shift all gaze samples’ coordinates accordingly, by subtracting the x\_offset and Y\_offset, which can be useful to define AOIs relatively to the study interface. (0,0) by default value, corresponding to no offsetting.  This parameter is to be used with caution depending on the settings of the eye-tracker data collection tool (which can include its own offset). |
| featurelist | A list of non-AOI-based features to be generated. By default the list includes all features. To generate less features a subset of the list can be defined. All features are defined in Section 3. The full list is:  ['numsegments','length','numsamples','numfixations','fixationrate','meanabspathangles',  'meanfixationduration','meanpathdistance','meanrelpathangles','stddevabspathangles',  'stddevfixationduration','stddevpathdistance','stddevrelpathangles',  'eyemovementvelocity', 'abspathanglesrate', 'relpathanglesrate',  'sumabspathangles','sumfixationduration','sumpathdistance','sumrelpathangles',  'blinknum', 'blinkdurationtotal', 'blinkdurationmean', 'blinkdurationstd', 'blinkdurationmin',  'blinkdurationmax', 'blinkrate', 'blinktimedistancemean', 'blinktimedistancestd',  'blinktimedistancemax', 'blinktimedistancemin',  'meanpupilsize', 'stddevpupilsize', 'maxpupilsize', 'minpupilsize', 'startpupilsize','endpupilsize',  'meanpupilvelocity', 'stddevpupilvelocity', 'maxpupilvelocity', 'minpupilvelocity',  'meandistance', 'stddevdistance', 'maxdistance', 'mindistance', 'startdistance', 'enddistance',  'numsaccades', 'sumsaccadedistance', 'meansaccadedistance', 'stddevsaccadedistance', 'longestsaccadedistance',  'sumsaccadeduration','meansaccadeduration', 'stddevsaccadeduration', 'longestsaccadeduration',  'meansaccadespeed', 'stddevsaccadespeed','minsaccadespeed', 'maxsaccadespeed',  'fixationsaccadetimeratio',  'numevents', 'numleftclic', 'numrightclic', 'numdoubleclic', 'numkeypressed', 'leftclicrate', 'rightclicrate', 'doubleclicrate', 'keypressedrate',  'timetofirstleftclic', 'timetofirstrightclic', 'timetofirstdoubleclic', 'timetofirstkeypressed'] |
| aoinames | list of the AOI names |
| aoisequencefeat | The list of AOI-sequence-based features, as defined in Section 3. Default is to generate all features:  ['aoisequence'] |
| aoigeneralfeat | A list of general AOI features, as defined in Section 3. Default is the full set of features:  ['fixationrate','numfixations','totaltimespent','proportionnum',  'proportiontime','longestfixation', 'meanfixationduration', 'stddevfixationduration', 'timetofirstfixation','timetolastfixation',  'numevents', 'numleftclic', 'numrightclic', 'numdoubleclic', 'leftclicrate', 'rightclicrate', 'doubleclicrate',  'timetofirstleftclic', 'timetofirstrightclic', 'timetofirstdoubleclic', 'timetolastleftclic', 'timetolastrightclic', 'timetolastdoubleclic',  'meanpupilsize', 'stddevpupilsize', 'maxpupilsize', 'minpupilsize', 'startpupilsize','endpupilsize',  'meanpupilvelocity', 'stddevpupilvelocity', 'maxpupilvelocity', 'minpupilvelocity',  'meandistance', 'stddevdistance', 'maxdistance', 'mindistance', 'startdistance', 'enddistance'] |
| aoinames | List of all AOI-names, required for the next sets of features. |
| aoitransfrom | A list of frequency-based features for transitions between AOIs, as defined in Section 3. By default, it is populated automatically based on the aoinames. |
| aoiproportion | A list of proportion-based features for transitions between AOIs, as defined in Section 3. By default it is populated automatically based on the aoinames. |
| aoifeaturelist | The final list of all AOI-based features which is populated automatically based on aoigeneralfeat, aoitransfrom and aoiproportion, and aoinames. |
| blink\_threshold | Lower and upper bound on size of invalid data gaps to be treated as blinks. Used to automatically compute blinks when they are not exported by the eye-tracker. |
| **Data processing, restoration and validation** | |
| VALIDITY\_METHOD | EMDAT supports several methods to measure data quality within each segment and scene. Current values are:  “1”: data quality is defined as the proportion of valid gaze samples as exported by the eye-tracker. This means validity is eye-tracker-dependent and is read as part of the eye-tracker input files (see Section 4.1).  “2”: data quality is defined in a binary way in terms of maximum time gap allowable, with a time gap being define as a series of gaze samples with missing coordinates or labeled as invalid by the eye-tracker.  “3”: data quality is defined as the proportion of valid gaze samples (including restored samples) as exported by the eye-tracker. Restored samples are samples that are labeled as invalid by the eye-tracker, but EMDAT could extrapolate its coordinates based on the fixation/saccade it likely falls into |
| VALID\_PROP\_THRESH | the minimum proportion of valid samples for a Segment or Scene to be considered valid. This is used for VALIDITY\_METHOD 1 and 3. Scene and segment is validity lower than the threshold will be discarded by EMDAT  Default is 0.8 (i.e., 80% of the gaze samples must be valid). A threshold of 0 means that no segment or scene are discarded. |
| VALID\_TIME\_THRESH | the maximum gap size allowable in samples for a Segment or Scene to be considered valid. The unit depend on the timestamp unit of the eye-tracker exported data. Most eye-trackers exports timestamp at the milliseconds levels, and the default maximum gap is set at 3000 ms (i.e., 3 seconds). |
| MAX\_SEG\_TIMEGAP | maximum gap size (ms) allowable in a segment with auto-partition option. Auto-partition is fully explained below in Section 4.2.3. Default is 1000 ms |
| VALID\_SAMPLES\_PROP\_SACCADE | proportion of valid gaze samples required per saccade. This is currently used only by TobiiV3 EYE\_TRACKRER\_TYPE when estimating saccade path. If less than 1, missing gaze sample will be extrapolated. Default is 1, meaning that a saccade is valid only if all of its gaze samples are valid. |
| MINSEGSIZE | minimum segment length that is considered meaningful for this experiment. Shorter segments will be discarded. Default is 0 (no segment are discarded) |
| INCLUDE\_HALF\_FIXATIONS | A Boolean value determining if a fixation extends between two consecutive Segments, should it be included in those Segments or not. Default is False. |
| PUPIL\_ADJUSTMENT | Pupil adjustment to minimize the pupil size differences among individual users, see Section 4.1. Currently supports 3 values:  “None”: no adjustement (default)  "rpscenter": substraction of the baseline pupil size from the raw pupil size  “PCPS”: Normalization of pupil size based on the baseline pupil size following [Iqbal et al., 2005, doi>10.1145/1054972.1055016] |
| **Developper mode** | |
| DEBUG | If TRUE, all warnings are treated as errors and all exceptions are risen. The verbosity level is set to “VERBOSE” (see below) |
| VERBOSE | Amount of print statements in the console. Currently supports three values:  "QUIET": prints nothing except errors and warnings  "NORMAL": prints essential information (default)  VERBOSE: prints information useful for debugging |

### 4.2.2 basicparticipants.py

This file is meant to initialize processing of the recording. In this file it is important to make sure that the file format are modified to fit the file structure and file name. Namely in *read\_participants\_Basic()*, the following variables must be checked and modified as needed: allfile, fixfile, sacfile, evefile, segfil, aoifile.

The sample Basicparticipant.py file in the repo includes code for each of the supported eye-trackers, for example for TobiiV3:

allfile = "{dir}/TobiiV3/P{rec}\_Data\_Export.tsv".format(dir=datadir, rec=rec)

fixfile = "{dir}/TobiiV3/P{rec}\_Data\_Export.tsv".format(dir=datadir, rec=rec)

sacfile = "{dir}/TobiiV3/P{rec}\_Data\_Export.tsv".format(dir=datadir, rec=rec)

evefile = "{dir}/TobiiV3/P{rec}\_Data\_Export.tsv".format(dir=datadir, rec=rec)

segfile = "{dir}/TobiiV3/TobiiV3\_sample\_{rec}.seg".format(dir=datadir, rec=rec)

aoifile = "{dir}/TobiiV3/TobiiV3\_sample\_{rec}.aoi".format(dir=datadir, rec=rec)

When applying EMDAT to a new dataset, it is best to simply duplicate the sample BasicParticipant.py or inherit from it, and modify the above variables. This is to allow supporting multiple datasets and even multiple ways to process the data from one dataset.

Note that there is a multiprocesses version of BasicParticipant in the repo that is meant to process the recordings in parallel. It works in the exact same way.

### 4.2.2 Entry Point

The entry point needs to:

* Iterate over the recording
* Call the *read\_participants\_Basic()* function for each recording

The EMDAT repository include several sample entry points named *testBasic\*.py*. They can serve as examples.

1. The unit of all duration measures is based on the timestamp found in the raw eye tracking data, e.g., if timestamps are in *ms* the EMDAT duration measures are in *ms* as well. See Section 4.1 for the raw input data. [↑](#footnote-ref-1)
2. Distance for saccades is expressed in pixel. [↑](#footnote-ref-2)