PyTrack

Release 0.0.1

May 23, 2019

Contents

This is a toolkit to analyse and visualize eye tracking data. It provides the following functionality:

CHAPTER 1

Feature Extraction

This involves extraction of parameters or meta-data related to blinks, fixations, saccades, microsaccades and pupil diameter. The features extracted are as follows:

Blink	Fixations	Saccades	Microsaccades	Pupil	Revisits to AOI/ROI
Count	Count	Count	Count	Size	Count
Avg Duration	Avg Duration	Velocity	Velocity	Time to Peak	First Pass Duration
Max Duration	Max Duration	Amplitude	Amplitude	Peak Size	Second Pass Duration
		Duration	Duration	Avg Size	
				Slope	
				Area Under Curve	

CHAPTER 2

Statistical Analysis

After extraction of features, PyTrack can perform tests such as the student T-Test, Welch T-Test, ANOVA, RMANOVA, n-way ANOVA and Mixed ANOVA. The between and within group factors can be specified.

CHAPTER 3

Visualization

PyTrack can generate a variety of plots. The visualization is through an interactive GUI. The plots that can be generated are as follows:

1. Fixation plot
2. Individual subject gaze heat map
3. Aggregate subject gaze heat map
4. Dynamic pupil size and gaze plot
5. Microsaccade position and velocity plot
6. Microsaccade main sequence plot

3.1 Table of Contents

3.2 Installation

PyTrack is built for Python3 because support for the Python2 is going to be stopped at the end of 2019. In order to install PyTrack please use any of the following:

```
python3 -m pip install PyTrack-NTU
pip install PyTrack-NTU
pip3 install PyTrack-NTU
```

Please make sure that pip is for Python3 and not Python2. Python3 can be found here or Anaconda Python3 can be found here.

NOTE: Python3 can be installed alongside Python2

3.3 Sample Data

In order to test the toolkit some sample data in SMI, EyeLink and Tobii formats can be found here. The .txt file in the folder describes the data found. The SMI and Tobii files have been taken from here.

3.4 Using PyTrack

CHAPTER 4

Setup

Before running the framework, lets setup the folder so PyTrack can read and save all the generated figures in one central location and things are organised.

Create a directory structure like the one shown below. It is essential for the listed directories to be present for the proper functioning of PyTrack.

NOTE: The sample data has a folder called NTU_Experiment which is already organised in the following manner. It can be used as reference.

```
[Experiment-Name]
|
└── Data/
|   |   subject_001.[asc/txt/tsv/...]
|   |   subject_002.[asc/txt/tsv/...]
|   |__ ......
|
└── Stimulus/
|   |   stim_1.[jpg/jpeg]
|   |   stim_2.[jpg/jpeg]
|   |__ ......
|
└── [Experiment-Name].json
```

[Experiment-Name] stands for the name of your experiment. The rest of the steps will use *NTU_Experiment* as the *[Experiment-Name]* folder.

Now, follow these steps:

1. Place the data of all your subjects in the *Data* folder under the main *NTU_Experiment* folder. Make sure the name of each of the data files is the name of the subjects/paticipants. Replace all spaces( ) with underscores (_).

   eg. *waffle_buttersnaps.asc* or *subject_001.asc*

2. For proper visualization of gaze data, its best if you include the stimuli presented during your experiment inside the *Stimuli* folder. Make sure the images have either **jpg, jpeg** or **png** extensions and the names match the names of the stimuli as present in your recorded data.

eg. *stim_1.jpg* or *random_picture.png*

3. The last and final step to setup the experiment directory is to include the experiment description json file. This file should contain the essential details of your experiment. It contains specifications regarding your experiment such as the stimuli you wish to analyse or the participants/subjects you wish to include. Mentioned below is the json file structure. The content below can be copied and pasted in a file called *NTU_Experiment*.json

   - "*Experiment_name*" should be the same name as the json file without the extension and "*Path*" should be the absolute path to your experiment directory without the final "/" at the end.
   - The subjects should be added under the "*Subjects*" field. You may specify one or more groups of division for your subjects (recommended for between group statistical analysis). **There must be atleast 1 group**.
   - The stimuli names should be added under the "*Stimuli*" field and again you may specify one or more types (recommended for between/within stimulus type statistical analysis). **There must be atleast 1 type**.
   - The "*Control_Questions*" field is optional. In case you have some stimuli that should be used to standardise/normalise features extracted from all stimuli, specify the names here. **These stimuli must be present under the "Stimuli" field under one of the types**.
   - **The field marked "Columns_of_interest" should not be altered**.
   - Under "*Analysis_Params*", just change the values of "Sampling_Freq", "Display_height" and "Display_width" to match the values of your experiment.

   Note: If you wish to analyse only a subset of your stimuli or subjects, specify only the ones of interest in the json file. The analysis and visualization will be done only for the ones mentioned in the json file.

NOTE: A sample json file is present in the NTU_Experiment folder in the sample data. You can just edit it to make your work simpler.

```
{
   "Experiment_name":"NTU_Experiment",
   "Path":"abcd/efgh/NTU_Experiment",
   "Subjects":{
      "group1":[
         "Subject_01",
         "Subject_02"
      ],
      "group2":[
         "Subject_03",
         "Subject_04"
      ]
   },
   "Stimuli":{
      "Type_1":[
         "Stim_1",
         "Stim_2"
      ],
      "Type_2":[
         "Stim_3",
         "Stim_4"
      ],
   },
   "Control_Questions":[
         "Stim_1"
    ],
   "Columns_of_interest":{
      "EyeTracker":[
         "GazeLeftx",
```

(continues on next page)

(continued from previous page)

```
            "GazeLefty",
            "GazeRightx",
            "GazeRighty",
            "PupilLeft",
            "PupilRight",
            "FixationSeq",
            "GazeAOI"
        ],
        "Extra":[
            "EventSource"
        ]
    },
    "Analysis_Params":{
        "EyeTracker":{
          "Sampling_Freq": 1000,
          "Display_width": 1920,
          "Display_height": 1280
        }
    }
}
```

CHAPTER 5

Running PyTrack

NOTE: All sample segments shown below are for the NTU_Experiment folder in the sample data.

1. In order to use the features, the **first step is to convert the raw data into a readable format**. In order to do so, the following code segment can be used:

```
from PyTrack.formatBridge import generateCompatibleFormat

generateCompatibleFormat(exp_path="complete/path/to/NTU_Experiment",
                         device="eyelink",
                         stim_list_mode='NA',
                         start='start_trial',
                         stop='stop_trial',
                         eye='B')
```

   To get a detailed understanding of the parameters of *generateCompatibleFormats* and modify it to your needs see the documentation here.

2. The **second step is to create an object of the Experiment class**.

```
from PyTrack.Experiment import Experiment

# Creating an object of the Experiment class
exp = Experiment(json_file="complete/path/to/NTU_Experiment/NTU_Experiment.json")
```

3. Now you can run the **feature extraction and statistical tests**

```
# Instantiate the meta_matrix_dict of an Experiment to find and extract all␣
↪features from the raw data
exp.metaMatrixInitialisation()

# Calling the function for the statistical analysis of the data
exp.analyse(parameter_list={"all"},
            between_factor_list=["Subject_type"],
            within_factor_list=["Stimuli_type"],
            statistical_test="anova",
```

(continues on next page)

(continued from previous page)

```
            file_creation=True)

# Does not run any statistical test. Just saves all the data as csv files.
exp.analyse(parameter_list={"all"},
            statistical_test="None",
            file_creation=True)
```

To get a detailed understanding of the parameters of the *analyse* function: here

To get a detailed understanding of the parameters of the *metaMatrixInitialisation* function: here

4. For **visualization**

```
# This function call will open up a GUI which you can use to navigate the entire␣
↪visualization process
exp.visualizeData()
```

5.1 Advanced Functionality

CHAPTER 6

Statistical Tests

The Experiment class contains a function called analyse() which is used to perform statistical analysis (eg: ANOVA or T test), by default there is only 1 between group factor ("Subject_type") and 1 within group factor ("Stimuli_type") that is considered. If additional factors need to be considered they need to added to the json file.

- For example if Gender is to be considered as an additional between group factor then in the json file, under "Subjects", for each subject, a corresponding dicitionary must be created where you mention the factor name and the corresponding value. Please also note that the square brackets ('[', ']') after group type need to be changed to curly brackets ('{', '}').
- Similarly for Stimuli, for example, if you are showing Words and Pictures to elicit different responses from a user and you additonally have 2 different brightness levels ("High" and "Low") then mention Brightness as an additional within group factor.

```
{
   "Subjects":{
      "group1":{
         "Subject_01": {"Gender": "M"},
         "Subject_02": {"Gender": "F"}
      },
      "group2":{
         "Subject_03": {"Gender": "F"},
         "Subject_04": {"Gender": "M"}
      }
   },
   "Stimuli":{
      "Type_1":{
         "Stim_1": {"Brightness": "High"},
         "Stim_2": {"Brightness": "Low"}
      },
      "Type_2":{
         "Stim_3": {"Brightness": "Low"},
         "Stim_4": {"Brightness": "High"}
      },
   },
}
```

Sample code segment to use the advanced statistical test:

```
from PyTrack.Experiment import Experiment

exp = Experiment(json_file="abcd/efgh/NTU_Experiment/NTU_Experiment.json")

exp.metaMatrixInitialisation()

exp.analyse(parameter_list={"all"},
            between_factor_list=["Subject_type", "Gender"],
            within_factor_list=["Stimuli_type", "Brightness"],
            statistical_test="anova",
            file_creation=True)
```

CHAPTER 7

Accessing extracted features as a dictionary

In case you wish to get the extracted features for a particilar Subject on a particular Stimulus:

```
from PyTrack.Experiment import Experiment

exp = Experiment(json_file="complete/path/to/NTU_Experiment/NTU_Experiment.json")

subject_name = "sub_333" #specify your own subject's name (must be in json file)
stimulus_name = "Alpha1" #specify your own stimulus name (must be in json file)

# Access metadata dictionary for particular subject and stimulus
exp.metaMatrixInitialisation()
single_meta = exp.getMetaData(sub=subject_name,
                              stim=stimulus_name)

# Access metadata dictionary for particular subject and averaged for stimulus types
exp.metaMatrixInitialisation(average_flag=True)
agg_type_meta = exp.getMetaData(sub=subject_name,
                                stim=None)
```

CHAPTER 8

Using PyTrack in Stand-alone mode

The stand-alone design requires only interaction with tyhe Stimulus class. This is recommended if you wish to extract features or visualize data for only 1 subject on a particular stimulus.

```
from PyTrack.Stimulus import Stimulus
from PyTrack.formatBridge import generateCompatibleFormat
import pandas as pd
import numpy as np


# function to convert data to generate csv file for data file recorded using EyeLink␣
↪on both eyes and the stimulus name specified in the message section
generateCompatibleFormat(exp_path="/path/to/smi_eyetracker_freeviewing.txt",
                         device="smi",
                         stim_list_mode='NA',
                         start='12',
                         stop='99')

df = pd.read_csv("/path/to/smi_eyetracker_freeviewing.csv")

# Dictionary containing details of recording. Please change the values according to␣
↪your experiment. If no AOI is desired, set aoi value to [0, 0, Display_width,␣
↪Display_height]
sensor_dict = {
                    "EyeTracker":
                    {
                       "Sampling_Freq": 1000,
                       "Display_width": 1280,
                       "Display_height": 1024,
                       "aoi": [390, 497, 759, 732]
                    }
                }

# Creating Stimulus object. See the documentation for advanced parameters.
stim = Stimulus(path="path/to/experiment/folder",
```

(continues on next page)

(continued from previous page)

```
                data=df,
                sensor_names=sensor_dict)

# Some functionality usage. See documentation of Stimulus class for advanced use.
stim.findEyeMetaData()
features = stim.sensors["EyeTracker"].metadata  # Getting dictioary of found metadata/
↪features

# Visualization of plots
stim.gazePlot(save_fig=True)
stim.gazeHeatMap(save_fig=True)
stim.visualize()

# Extracting features
MS, ms_count, ms_duration = stim.findMicrosaccades(plot_ms=True)
```

See the stimulus class for more details on the functions: here

8.1 Authors

- **Upamanyu Ghose** (github | email)
- **Arvind A S** (github | email)

See also the list of contributors who participated in this project.

8.2 Acknowledgments

- The formatsBridge module was adapted from the work done by Edwin Dalmaijer in PyGazeAnalyser.
- This work was done under the supervision of Dr. Chng Eng Siong - School of Computer Science and Engineering NTU and in collaboration with Dr. Xu Hong - School of Humanitites and Social Sciences NTU.
- We extend our thanks to the **Department of Computer Science and Engineering Manipal Isntitute of Technology**[link] and the **Department of Computer Science and Information Systems BITS Pilani, Hyderabad Campus** [link].

CHAPTER 9

LICENSE

```
GNU GENERAL PUBLIC LICENSE
   Version 3, 29 June 2007
```



```
Preamble
```

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

```
TERMS AND CONDITIONS
```

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

```
a) The work must carry prominent notices stating that you modified
it, and giving a relevant date.

b) The work must carry prominent notices stating that it is
released under this License and any conditions added under section
```

(continues on next page)

(continued from previous page)

```
7.  This requirement modifies the requirement in section 4 to
"keep intact all notices".

c) You must license the entire work, as a whole, under this
License to anyone who comes into possession of a copy.  This
License will therefore apply, along with any applicable section 7
additional terms, to the whole of the work, and all its parts,
regardless of how they are packaged.  This License gives no
permission to license the work in any other way, but it does not
invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display
Appropriate Legal Notices; however, if the Program has interactive
interfaces that do not display Appropriate Legal Notices, your
work need not make them do so.
```

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

```
a) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by the
Corresponding Source fixed on a durable physical medium
customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by a
written offer, valid for at least three years and valid for as
long as you offer spare parts or customer support for that product
model, to give anyone who possesses the object code either (1) a
copy of the Corresponding Source for all the software in the
product that is covered by this License, on a durable physical
medium customarily used for software interchange, for a price no
more than your reasonable cost of physically performing this
conveying of source, or (2) access to copy the
Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the
written offer to provide the Corresponding Source.  This
alternative is allowed only occasionally and noncommercially, and
only if you received the object code with such an offer, in accord
with subsection 6b.

d) Convey the object code by offering access from a designated
place (gratis or for a charge), and offer equivalent access to the
Corresponding Source in the same way through the same place at no
further charge.  You need not require recipients to copy the
Corresponding Source along with the object code.  If the place to
copy the object code is a network server, the Corresponding Source
may be on a different server (operated by you or a third party)
```

(continues on next page)

(continued from previous page)

```
that supports equivalent copying facilities, provided you maintain
clear directions next to the object code saying where to find the
Corresponding Source.  Regardless of what server hosts the
Corresponding Source, you remain obligated to ensure that it is
available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided
you inform other peers where the object code and Corresponding
Source of the work are being offered to the general public at no
charge under subsection 6d.
```

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

```
a) Disclaiming warranty or limiting liability differently from the
terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or
author attributions in that material or in the Appropriate Legal
Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or
requiring that modified versions of such material be marked in
reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or
authors of the material; or

e) Declining to grant rights under trademark law for use of some
trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that
material by anyone who conveys the material (or modified versions of
it) with contractual assumptions of liability to the recipient, for
any liability that these contractual assumptions directly impose on
those licensors and authors.
```

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise

does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work

from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE

WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

```
         END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs
```

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w'` and`show c'` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see https://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read https://www.gnu.org/licenses/why-not-lgpl.html.

CHAPTER 10

Modules

10.1 Experiment

class Experiment.**Experiment**(*json_file*, *reading_method='SQL'*, *aoi='NA'*)

Bases: object

This is class responsible for the analysis of data of an entire experiment. The creation of a an object of this class will subsequently create create objects for each *Subject*. involved in the experiment (which in turn would create object for each *Stimulus* which is viewed by the subject).

This class also contains the *analyse* function which is used for the statistical analysis of the data (eg: Mixed ANOVA, RM ANOVA etc).

Parameters

- **json_file** (*str*) – Name of the json file that contains information regarding the experiment or the database
- **reading_method** (*str {"SQL", "CSV"} (optional)*) – Mentions the format in which the data is being stored
- **sensors** (*list(str) (optional)*) – Contains the names of the different sensors whose indicators are being analysed (currently only Eye Tracking can be done However in future versions, analysis of ECG and EDA may be added)
- **aoi** (*str {'NA', 'e', 'r', 'p'} | tuple (optional)*) – If 'NA' then AOI is the entire display size. If 'e' then draw ellipse, 'p' draw polygon and 'r' draw rectangle. If type is tuple, user must specify the coordinates of AOI in the following order (start_x, start_y, end_x, end_y). Here, x is the horizontal axis and y is the vertical axis.

analyse(*parameter_list={'all'}*, *between_factor_list=['Subject_type']*, *within_factor_list=['Stimuli_type']*, *statistical_test='Mixed_anova'*, *file_creation=True*, *ttest_type=1*)

This function carries out the required statistical analysis.

The analysis is carried out on the specified indicators/parameters using the data extracted from all the subjects that were mentioned in the json file. There are 4 different tests that can be run,

namely - Mixed ANOVA, Repeated Measures ANOVA, T Test and Simple ANOVA (both 1 and 2 way)

Parameters

- **parameter_list** (*set (optional)*) – Set of the different indicators/parameters (Pupil_size, Blink_rate) on which statistical analysis is to be performed, by default it will be "all" so that all the parameter are considered.
- **between_factor_list** (*list(str) (optional)*) – List of between group factors, by default it will only contain "Subject_type" If any additional parameter (eg: Gender) needs to be considered, then the list will be: between_factor_list = ["Subject_type", "Gender"] DO NOT FORGET TO INCLUDE "Subject_type", if you wish to consider "Subject_type" as a between group factor. Eg: between_factor_list = ["factor_x"] will no longer consider "Subject_type" as a factor. Please go through the README FILE to understand how the JSON FILE is to be written for between group factors to be considered.
- **within_factor_list** (*list(str) (optional)*) – List of within group factors, by default it will only contain "Stimuli_type" If any additional parameter, needs to be considered, then the list will be: between_factor_list = ["Subject_type", "factor_X"] DO NOT FORGET TO INCLUDE "Stimuli_type", if you wish to consider "Stimuli_type" as a within group factor. Eg: within_factor_list = ["factor_x"] will no longer consider "Stimuli_type" as a factor. Please go through how the README FILE to understand how the JSON FILE is to be written for within group factors to be considered.
- **statistical_test** (*str {"Mixed_anova","RM_anova","ttest","anova","None"} (optional)*) –

  Name of the statistical test that has to be performed. NOTE:

  - ttest: There are 3 options for ttest, and your choice of factors must comply with one of those options, for more information, please see description of *ttest_type* variable given below.
  - Mixed_anova: Only 1 between group factor and 1 within group factor can be considered at any point of time
  - anova: Any number of between group factors can be considered for analysis
  - RM_anova: Upto 2 within group factors can be considered at any point of time
- **file_creation** (*bool (optional)*) –

  Indicates whether a csv file containing the statistical results should be created. NOTE: The name of the csv file created will be by the name of the statistical test that has been chosen. A directory called "Results" will be created within the Directory whose path is mentioned in the json file and the csv files will be stored within "Results" directory. If any previous file by the same name exists, it will be overwritten.
- **ttest_type** (*int {1,2,3} (optional)*) –

  Indicates what type of parameters will be considered for the ttest NOTE: - 1: Upto 2 between group factors will be considered for ttest - 2: 1 within group factor will be considered for ttest - 3: 1 within group and 1 between group factor will be considered for ttest

Examples

Testing yamana tamana lamana

For calculating Mixed ANOVA, on all the parameters, with standardisation, NOT averaging across stimuli of the same type and considering Subject_type and Stimuli_type as between and within group factors respectively

```
>>> analyse(self, standardise_flag=False, average_flag=False, parameter_list={
↪"all"}, between_factor_list=["Subject_type"], within_factor_list=["Stimuli_
↪type"], statistical_test="Mixed_anova", file_creation = True)
OR
>>> analyse(self, standardise_flag=True) (as many of the option are present␣
↪by default)
```

For calculating 2-way ANOVA, for "blink_rate" and "avg_blink_duration", without standardisation with averaging across stimuli of the same type and considering Subject_type and Gender as the between group factors while NOT creating a new csv file with the results

```
>>> analyse(self, average_flag=True, parameter_list={"blink_rate", "avg_blink_
↪duration"}, between_factor_list=["Subject_type", "Gender"], statistical_
↪test="anova", file_creation = False)
```

columnsArrayInitialisation()

The functions extracts the names of the columns that are to be analysed

Parameters **json_file** (*str*) – Name of the json file which contains details of the experiment

Returns **columns_list** – List of names of columns of interest

Return type list(str)

drawAOI()

Function that allows speicification of area of interest (AOI) for analysis.

fileWriting(*writer*, *csvFile*, *pd_dataframe*, *values_list*)

This function is used to write the statistical results into a csv file

Parameters

- **writer** (*file object*) – File object that is used to write into a csv file
- **csvFile** (*str*) – Name of the csv file
- **pd_dataframe** (*pandas DataFrame*) – Statistical results
- **values_list** (*list*) – Used for labelling the results

getMetaData(*sub*, *stim=None*, *sensor='EyeTracker'*)

Function to return the extracted features for a given subject/participant.

Parameters

- **sub** (*str*) – Name of the subject/participant.
- **stim** (str | None) – Name of the stimulus. If 'str', the features of the given stimulus will be returned. If None, the features of all stimuli averaged for the different stimuli types (as mentoned in json file) is wanted.
- **sensor** (*str*) – Name of the sensor for which the features is wanted.

Returns

Return type dict

Note:

- **If the *stim* is `None`, the returned dictionary is organised as follows** {"Stim_TypeA": {"meta1":[], "meta2":[], ... }, "Stim_TypeB": {"meta1":[], "meta2":[], ... }, ... }
- **If the *stim* is `str`, the returned dictionary is organised as follows** {"meta1":[], "meta2":[], ... }

To get the names of all the metadata/features extracted, look at the *Sensor* class

---

`metaMatrixInitialisation`(*standardise_flag=False*, *average_flag=False*)

This function instantiates the `meta_matrix_dict` with values that it extracts from the `aggregate_meta` variable of each Subject object.

Parameters

- **`standardise_flag`** (*`bool (optional)`*) – Indicates whether the data being considered need to be standardised (by subtracting the control values/baseline value)
- **`average_flag`** (*`bool (optional)`*) – Indicates whether the data being considered should averaged across all stimuli of the same type NOTE: Averaging will reduce variability and noise in the data, but will also reduce the quantum of data being fed into the statistical test

`return_index`(*value_index*, *summation_array*)

This function is used in helping to find the corresponding stimuli for data points in certain parameters, that more than one value for a specific stimuli

Parameters

- **`value_index`** (*`int`*) – Index of an instance of the parameter in the value array of the meta_matrix_dict
- **`summation_array`** (*`list`*) – list of values whose index will indicate which stimuli an instance will correspond to

Returns **summation_index** – Is the index of the stimuli to which an instance of the parameter corresponds to

Return type int

`stimuliArrayInitialisation`()

This functions instantiates the dictionary *stimuli* with the list of names of the different stimuli by category

Parameters **`json_file`** (*`str`*) – Name of the json file which contains details of the experiment

Returns **data_dict** – Dictionary containing the names of the different stimuli categorised by type

Return type dict

`subjectArrayInitialisation`(*reading_method*)

This function initialises an list of objects of class *Subject*.

Parameters **`reading_method`** (*`str {'SQL','CSV'}`*) – Specifies the database from which data extraction is to be done from

Returns **subject_list** – list of objects of class Subject

Return type list(Subject objects)

`summationArrayCalculation`(*meta*, *sub_index*, *stimuli_index*)

This function is used for creating a list that will be used later for identifying the corresponding stimuli for an instance in the meta_matrix_dict

Parameters

- **meta** (*str*) – Is the parameter that is being considered
- **sub_index** (*int*) – Is the index of subject with regard to meta_matrix_dict
- **sub_index** – Is the index of subject with regard to meta_matrix_dict

Returns **summation_array** – list of values whose index will indicate which stimuli an instance will correspond to

Return type list

visualizeData()
Function to open up the GUI for visualizing the data of the experiment.

This function can be invoked by an *Experiment* object. It opens up a window and allows the usee to visualize data such as dynamic gaze and pupil plots, fixation plots and gaze heat maps for individual subjects or aggregate heat maps for a group of subjects on a given stimulus.

welch_ttest(*dv*, *factor*, *subject*, *data*)
This funtion is used to calculate the welch ttest (used when unequal variance of 2 samples exists)

Parameters

- **dv** (*str*) – Name of the parameter that is being considered for statistical analysis
- **factor** (*str*) – Name of the factor on which statistical analysis is being done
- **subject** (*str*) – Name of the subject
- **data** (*pandas DataFrame*) – Data on which the Welch t-test is to be performed

Returns

- **normality** (*pandas DataFrame*) – Data regarding normality of the different categories of the 'factor'
- **results** (*pandas DataFrame*) – Data containing the results of the Welch t-test

class Experiment.**Visualize**(*master*, *subjects*, *exp*)
Bases: object

button_click(*sub=None*)

subFrameSetup()

10.2 Sensor

class Sensor.**Sensor**(*name*, *sampling_freq*)
Bases: object

This class represents the paramters of the sensors used during the experiment.

As of now the sensor class only support the Eye Tracker but in future versions, we plan to include EEG, ECG, EDA and Respiration as well. The class is used to store all the metadata/features extracted during analysis.

sensor_names
List of accepted sensors.

Type list(str)

meta_cols
Dictionary of lists containing the various metadata/features of a given sensor.

Type dict

Parameters

- **name** (*str*) – Name of the sensor.
- **sampling_freq** (*int | float*) – Sampling frequency of the sensor.

```
meta_cols = {'EyeTracker':  ['response_time', 'pupil_size', 'time_to_peak_pupil', 'pea
sensor_names = ['EyeTracker']
```

10.3 Stimulus

class Stimulus.**Stimulus**(*path, name='id_rather_not', stim_type='doesnt_matter', sensor_names=['EyeTracker'], data=None, start_time=0, end_time=-1, roi_time=-1, json_file=None, subject_name='buttersnaps', aoi=None*)

Bases: object

This is the main class that performs analysis and visualization of data collected during presentation of various stimuli during the experiment.

If the framework is used in the *Experiment Design* objects of this class are created implicitly and the user need not worry about the internal functioning of the class methods. However, if using the *Stand-alone Design*, the user needs to explicitly create an object of the class and invoke the functions based on what is needed.

Parameters

- **path** (*str*) – This parameter is the absolute path to the experiment directory containing the json file, stimuli folder with pictures etc. For the *Experiment Design*, this parameter is internally handled. In the *Stand-alone Design*, this parameter needs to be specified while creating the object. All data and plots, if saved, will be stored in folders in this location.
- **name** (*str*) – The name of the stimulus. For the *Experiment Design*, this parameter is internally handled. If using in *Stand-alone Design*, this parameter is optional and will default to *id_rather_not*.
- **stim_type** (*str*) – The type of stimulus, if there are different classes of stimulus in the experiment. For the *Experiment Design*, this parameter is internally handled. If using in *Stand-alone Design*, this parameter is optional and will default to *doesnt_matter*.
- **sensor_names** (*list(str) | dict*) – In the *Experiment Design sensor_names* will default to the sensors being used (as mentioned in the json_file). As of now this only supports EyeTracker. If using in *Stand-alone Design*, this parameter must be a dictionary of dictionaries with the details of the sensors and their attributes. The framework as of now just supports Eye Tracking so in the *Stand-alone Design*, *sensor_names* should be in this format (edit the value of the "Sampling_Freq" according to your eye tracker's value): {"EyeTracker": {"Sampling_Freq":1000}}
- **data** (*pandas DataFrame*) – The data for this stimulus as a Pandas DataFrame. For the *Experiment Design*, this parameter is internally handled. In the *Stand-alone Design* use the *formatBridge.generateCompatibleFormat* module to convert your data into the accepted format and then pass the csv file as a Pandas DataFrame to *data*. This should be the data for just a single stimulus or else the features extracted will not make sense to you. In case you wish to analyse all stimuli for 1 subject, we suggest using the *Experiment Design*.
- **start_time** (*int*) – The onset of stimulus. For the *Experiment Design*, this parameter is internally handled and if -1, it implies that data for stimulus is missing. In the *Stand-alone Design*, this parameter is optional (0 by default) and need not be mentioned.

However, if supplying an entire dataframe and it is desired to analyse data in a given range, supply the index value to start from. Also specify *end_time* or else -1 is used by default i.e end of DataFrame.

- **end_time** (*int*) – The offset of stimulus. For the *Experiment Design*, this parameter is internally handled and if -1, it implies that data for stimulus is missing. In the *Stand-alone Design*, this parameter is optional (-1 by default) and need not be mentioned. However, if supplying an entire dataframe and it is desired to analyse data in a given range, supply the index value to end at. Also specify *start_time* or else 0 is used by default i.e start of DataFrame.
- **roi_time** (*int*) –
- **json_file** (*str*) – Desciption of experiment as JSON file. For the *Experiment Design*, this parameter is internally handled. In the *Stand-alone Design* it is not required (leave as `None`).
- **subject_name** (*str (optional)*) – Name of the subject being analysed. For the *Experiment Design*, this parameter is internally handled. In the *Stand-alone Design* it is optional (Defaults to buttersnaps).
- **aoi** (*tuple*) – Coordinates of AOI in the following order (start_x, start_y, end_x, end_y). Here, x is the horizontal axis and y is the vertical axis.

calculateMSThreshold(*vel*, *sampling_freq*, *VFAC=5.0*)

Function to calculate velocity threshold value for X and Y directions to classify point as a microsaccade point.

Serves as a helper function. See *findMicrosaccades*

Parameters

- **vel** (*array | list*) – Gaze velocity in x or y direction
- **sampling_freq** (*float*) – Sampling frequency of the eye tracking device
- **VFAC** (*float*) – Scalar constant used to find threshold (Defaults to 5.0). See R. Engbert and K. Mergenthaler, "Microsaccades are triggered by low retinal image slip," Proc. Natl. Acad. Sci., vol. 103, no. 18, pp. 7192–7197, 2006.

Returns **radius** – Threshold radius in x or y direction

Return type float

diff(*series*)

Python implementation of Matlab's 'diff' function.

Computes the difference between (n+1)th and (n)th elements of array. Returns (a[n+1] - a[n]) for all n.

Parameters **series** (*list | array (numeric)*) – Numeric list, of type `int` or `float`. Must be atleast of length 2.

Returns The size of the returned list is n-1 where n is the size of *series* supplied to the *diff*.

Return type list | array (numeric)

findBinocularMS(*msl*, *msr*)

Function to find binocular microsaccades from monocular microsaccades.

Serves as helper function. See *findMicrosaccades*

Parameters

- **msl** (*array | list (num_ms, 9)*) – Microsaccade list returned by *findMonocularMS* for the left eye. *num_ms* stands for the number of left eye microsaccades.
- **msr** (*array | list (num_ms, 9)*) – Microsaccade list returned by *findMonocularMS* for the right eye. *num_ms* stands for the number of right eye microsaccades.

Returns **ms** – Dictionary of values containing the number of binary microsaccades, number of left eye microsaccades, number of right eye microsaccades, binary microsaccades list, left microsaccades list and right microsaccades list. - "NB" : int - "NR" : int - "NL" : int - "bin" : array | list (num_ms, 18) - "left" : array | list (num_ms, 9) - "right" : array | list (num_ms, 9)

Return type dict

findBlinkParams()

Function to find blink parameters like count, duration and average duration

Internal function of class that uses its *data* member variable. Does not take any input and can be invoked by an object of the class. Serves as a helper function. See *findEyeMetaData*

Returns Tuple consisting of (blink_cnt, peak_blink_duration, avg_blink_duration)

Return type list

findBlinks(*pupil_size*, *gaze=None*, *sampling_freq=1000*, *concat=False*, *concat_gap_interval=100*, *interpolate=False*)

Finds indices of occurances of blinks and interpolates pupil size and gaze data.

Function to find blinks and return blink onset, offset indices and interpolated pupil size data. Adapted from: R. Hershman, A. Henik, and N. Cohen, "A novel blink detection method based on pupillometry noise," Behav. Res. Methods, vol. 50, no. 1, pp. 107-114, 2018. Goto https://osf.io/jyz43/ for R and Matlab implementation.

Parameters

- **pupil_size** (*array | list*) – Pupil size data for left or right eye
- **gaze** (*dict of list*) – Gaze in x and y direction. {"x" : list , "y" : list }
- **sampling_freq** (*float*) – Sampling frequency of eye tracking hardware (Defaults to 1000).
- **concat** (*bool*) – Concatenate close blinks/missing trials or not. `False` by default. See R. Hershman et. al. for more information
- **concat_gap_interval** (*float*) – Minimum interval between successive missing samples/blinks to consider for concatenation. If *concat* is `False` this parameter does not matter. Default value is 100.
- **interpolate** (*bool*) – Interpolate pupil and gaze data durig blinks (Defaults to *False*').

Returns

- **blinks** (*dict*) – Blink onset and offset indices. {"blink_onset" : list , "blink_offset" : list}
- **interp_pupil_size** (*array | list*) – Interpolated pupil size data for left or right eye after fixing blinks. If *interpolate*'=''*False*', this is the same as *pupil_size* supplied to the function.

- **new_gaze** (*dict*) – Interpolated gaze in x and y direction after fixing blinks. If *interpolate*=*'False'*, this is the same as *gaze* supplied to the function. {"x" : list, "y" : list}

findEyeMetaData(*sampling_freq=1000*)

Function to find all metadata/features of eye tracking data.

Internal function of class that uses its *data* member variable. Can be invoked by an object of the class. The metadata is stored in the sensor object of the class and can be accessed in the following manner.

Examples

The following code will return the metadata dictionary containing all meta features extracted.

```
>>> stim_obj.findEyeMetaData()
>>> stim_obj.sensors["EyeTracker"].metadata
```

This segment allows you to extract individual features

```
>>> stim_obj.sensors["EyeTracker"].metadata["pupil_slope"]
>>> stim_obj.sensors["EyeTracker"].metadata["fixation_count"]
```

findFixationParams()

Function to find fixation parameters like count, max duration and average duration.

Internal function of class that uses its *data* member variable. Does not take any input and can be invoked by an object of the class. Serves as a helper function. See *findEyeMetaData*

Returns Tuple consisting of (fixation_count, max_fixation_duration, avg_fixation_duration)

Return type tuple

findFixations()

Function to extract indices of fixation sequences.

Internal function of class that uses its *data* member variable to compute indices. Does not take any input and can be invoked by an object of the class. Serves as a helper function.

Returns **fixation_indices** – Indices of start and end of fixations. {"start": fixation_onset list, "end": fixation_offset list}

Return type dict

findMicrosaccades(*sampling_freq=1000*, *plot_ms=False*)

Function to detect microsaccades within fixations.

Adapted from R. Engbert and K. Mergenthaler, "Microsaccades are triggered by low retinal image slip," Proc. Natl. Acad. Sci., vol. 103, no. 18, pp. 7192–7197, 2006.

Parameters

- **sampling_freq** (*float*) – Sampling Frequency of eye tracker (Defaults to 1000)
- **plot_ms** (*bool*) – Wether to plot microsaccade plots and main sequence or not (Defaults to `False`). If `True`, the figures will be plot and saved in the folder Subjects in the experiment folder.

Returns

- **all_bin_MS** (return value of *findBinocularMS*) – All the binocular microsaccades found for the given stimuli.

- **ms_count** (*int*) – Total count of all binocular and monocular microsaccades.
- **ms_duration** (*list(sloat)*) – List of durations of all microsaccades.
- **temp_vel** (*list(float)*) – List of peak velocities of all microsaccades.
- **temp_amp** (*list(float)*) – List of amplitudes of all microsaccades.

findMonocularMS(*gaze*, *vel*, *sampling_freq=1000*)

Function to find binocular microsaccades from monocular microsaccades.

Serves as helper function. See *findMicrosaccades*

Parameters

- **gaze** (*array | list*) – Gaze positons in x or y direction
- **vel** (*array | list*) – Gaze velocities in x or y direction
- **sampling_freq** (*float*) – Sampling Frequency of eye tracker (Defaults to 1000)

Returns

- **MS** (*array (num_ms, 9)*) – Array of 9 microsaccade Parameters. These Parameters correspond to the following array indices 0. starting index 1. ending index 2. peak velocity 3. microsaccade gaze vector (x direction) 4. microsaccade gaze vector (y direction) 5. amplitude (x direction) 6. amplitude (y direction) 7. threshold radius (x direction) 8. threshold radius (y direction) num_ms = *ms_count*
- **ms_count** (*int*) – Number of microsaccades
- **ms_duration** (*list(int)*) – List of duration of each microsaccade. Contains as many values as *ms_count*

findPupilParams()

Function to find pupil parameters like size, peak size, time to peak size, area under curve, slope, mean size, downsampled pupil size

Internal function of class that uses its *data* member variable. Does not take any input and can be invoked by an object of the class. Serves as a helper function. See *findEyeMetaData*

Returns Tuple consisting of (pupil_size, peak_pupil, time_to_peak, pupil_AUC, pupil_slope, pupil_mean, pupil_size_downsample)

Return type tuple

findResponseTime(*sampling_freq=1000*)

Function to find the response time in milliseconds based on the sampling frequency of the eye tracker.

Internal function of class that uses its *data* member variable. Serves as a helper function. See *findEyeMetaData*

Parameters **sampling_freq** (*float*) – Sampling Frequency of eye tracker (Defaults to 1000)

Returns Response time in milliseconds

Return type float

findSaccadeParams(*sampling_freq=1000*)

Function to find saccade parameters like peak velocity, amplitude, count and duration.

Internal function of class that uses its *data* member variable. Serves as a helper function. See *findEyeMetaData*

Parameters **sampling_freq** (*float*) – Sampling Frequency of eye tracker (Defaults to 1000)

Returns Tuple consisting of (saccade_count, saccade_duration, saccade_peak_vel, saccade_amplitude).

Return type tuple

findSaccades()

Function to extract indices of saccade sequences.

Saccades are assumed to be interspersed between fixations. Internal function of class that uses its *data* member variable to compute indices. Does not take any input and can be invoked by an object of the class. Serves as a helper function.

Returns **saccade_indices** – Indices of start and end of saccades. {"start": saccade_onset list, "end": saccade_offset list}

Return type dict

gazeHeatMap(*save_fig=False*, *show_fig=True*)

Function to plot heat map of gaze.

Internal function of class that uses its *data* member variable. Can be invoked by an object of the class.

Parameters

- **save_fig** (*bool*) – Save the heat map figure or not (Defaults to `False`). If `True`, will be saved in the Subjects folder of the experiment folder
- **show_fig** (*bool*) – Display the heat map figure or not (Defaults to `True`).

gazePlot(*save_fig=False*, *show_fig=True*)

Function to plot eye gaze with numbered fixations.

Internal function of class that uses its *data* member variable. Can be invoked by an object of the class.

Parameters

- **save_fig** (*bool*) – Save the gaze plot figure or not (Defaults to `False`). If `True`, will be saved in the Subjects folder of the experiment folder
- **show_fig** (*bool*) – Display the gaze plot figure or not (Defaults to `True`).

getData(*data*, *sensor_names*)

Function to extract data and store in local format.

It is invoked by *__init__* when the object of the class is created. This function is used in the *Experiment Design*.

Parameters

- **data** (*pandas DataFrame*) – DataFrame containing the eye tracking data.
- **sensor_names** (*list (str)*) – List of sensors being used for the experiment (currently supports only EyeTracker).

Returns **extracted_data** – Dictionary of extracted data to be used by the functions of the class. - "ETRows" : list, - "FixationSeq" : list, - "Gaze" : dict, - "InterpPupilSize" : list, - "InterpGaze" : dict, - "BlinksLeft" : dict, - "BlinksRight" : dict

Return type dict

getDataStandAlone(*data*, *sensor_names*)

Function to extract data and store in local format.

It is invoked by *__init__* when the object of the class is created. This function is used in the *Stand-alone Design*.

Parameters

- **data** (*pandas DataFrame*) – DataFrame containing the eye tracking data.
- **sensor_names** (*dict*) – Dictionary of dictionaries containing list of sensors being used for the experiment (currently supports only EyeTracker) and their Parameters. See *sensor_names* in Stimulus for details.

Returns **extracted_data** – Dictionary of extracted data to be used by the functions of the class. - "ETRows" : list, - "FixationSeq" : list, - "Gaze" : dict, - "InterpPupilSize" : list, - "InterpGaze" : dict, - "BlinksLeft" : dict, - "BlinksRight" : dict

Return type dict

numberRevisits()

Calculates the number of times the eye revisits within the region of interest, each instance should atleast be 4 milliseconds long

Returns **num_readings** – Number of times the subject revisits the Area of Interest (1 revisit is consecutive fixations within AOI)

Return type int

passDurationCalculation()

Calculates the amount of time spent during the first and second revisit in the region of interest

Returns

- **first_pass_duration** (*int*) – duration spent on first visit to the Area of Interest
- **second_pass_duration** (*int*) – duration spent on the second revisit of the Area of Interest

position2Velocity(*gaze*, *sampling_freq*)

Function to calculate velocity for a gaze point based on a 6 sample window.

Serves as a helper function. See *findMicrosaccades*.

Parameters

- **gaze** (*array | list*) – Gaze positons in x or y direction.
- **sampling_freq** (*float*) – Sampling Frequency of eye tracker.

Returns **velocity** – Gaze velocities in x or y direction.

Return type array | list

setAOICol(*data*)

Function to set values based on a point being inside or outsode the AOI.

Parameters **data** (*list*) – List of size 3 containing gaze_aoi, gaze_x and gaze_y column data.

Returns **gaze_aoi_new** – Modified gaze_aoi column with the mask for points inside and outside the AOI.

Return type list

smooth(*x*, *window_len*)

Smoothing function to compute running average.

Computes the running average for a window size of *window_len*. For the boundary values (*window_len*-1 values at start and end) the window length is reduced to accommodate no padding.

Parameters

- **x** (*list | array (numeric)*) – Numeric list, of type `int` or `float` to compute running average for.
- **window_len** (*int*) – Size of averaging window. Must be odd and >= 3.

Returns **y** – Smoothed version *x*.

Return type list | array (numeric)

smoothGaze(*vel*, *gaze*, *sampling_freq*)

Function to smoothen gaze positions using running average method.

Serves as a helper function. See *findMicrosaccades*

Parameters

- **vel** (*array | list*) – Gaze velocities in x or y direction
- **gaze** (*array | list*) – Gaze positons in x or y direction
- **sampling_freq** (*float*) – Sampling Frequency of eye tracker

Returns **smooth_gaze** – Smoothened gaze positons in x or y direction

Return type array | list

visualize(*show=True*)

Function to create dynamic plot of gaze and pupil size.

Internal function of class that uses its *data* member variable. Does not take any input and can be invoked by an object of the class.

Stimulus.**groupHeatMap**(*sub_list*, *stim_name*, *json_file*, *save_fig=False*)

Function to plot aggregate heat map of gaze for a list if subjects.

Invoked by the *subjectVisualize* function of the *Subject* class.

Parameters

- **sub_list** (*list (Subject)*) – List of *Subject* class objects to plot the gaze heat map for.
- **stim_name** (*dict*) – Dictionary containing the type of stimulus and the number of stimulus of that type. {stim_type:stim_num}
- **json_file** (*str*) – Name of json file containing details of the experiment.
- **save_fig** (*bool*) – Save the figure or not.

10.4 Subject

class Subject.**Subject**(*path*, *name*, *subj_type*, *stimuli_names*, *columns*, *json_file*, *sensors*, *database*, *reading_method*, *aoi*)

Bases: `object`

This class deals with encapsulating all the relevant information regarding a subject who participated in an experiment. The class contains functions that help in the extraction of data from the databases (SQL or CSV) and the creation of the *Stimuli* objects. The class also calculates the control data for the purpose of standardisation.

Parameters

- **name** (*str*) – Name of the Subject
- **subj_type** (*str*) – Type of the Subject
- **stimuli_names** (*list(str)*) – List of stimuli that are to be considered for extraction
- **columns** (*list(str)*) – List of columns that need to be extracted from the database
- **json_file** (*str*) – Name of json file that contains information regarding the experiment/database
- **sensors** (*list(str)*) – Contains the names of the different sensors whose indicators are being analysed
- **database** (*str | SQL object*) – is the SQL object for the SQL database | name of the folder that contains the name csv files
- **manual_eeg** (*bool*) – Indicates whether artifact removal is manually done or not
- **reading_method** (*str*) – Mentions the format in which the data is being stored

dataExtraction(*columns*, *json_file*, *database*, *reading_method*, *stimuli_names*)

Extracts the required columns from the data base and the required stimuli_column and returns a pandas datastructure

Parameters

- **columns** (*list(str)*) – list of the names of the columns of interest
- **json_file** (*str*) – Name of the json file that contains information about the experiment
- **database** (*SQL object| str*) – is the SQL object for the SQL database | name of the folder that contains the name csv files
- **reading_method** (*str {"SQL","CSV"}*) – Describes which type of databse is to be used for data extraction
- **stimuli_names** (*list(str)*) – List of stimuli that are to be considered for extraction

Returns **df** – contains the data of columns of our interest

Return type pandas DataFrame

getControlData()

Function to find the values for standardization/normalization of the features extracte from the different stimuli.

The method is invoked implicitly by the *__init__* method. It extracts the features/metadata for the stimuli mentioned in the json file under the "*Control_Questions*" field. If it is blank the values in the control data structure will be all 0s. During analysis, these control values will be subtracted from the values found for each stimulus.

Returns

control – Dictionary containing the standardised values for all the metadata/features. The keys of the dictionary are the different meta columns for a given sensor type. It can be found under meta_cols in *Sensor*.

Return type dict

stimulusDictInitialisation(*stimuli_names*, *columns*, *json_file*, *sensors*, *database*, *reading_method*)

Creates a list of objects of class *Stimuli*.

Parameters

- **stimuli_names** (*list(str)*) – list of names of different stimulus
- **columns** (*list(str)*) – list of names of the columns of interest
- **json_file** (*str*) – Name of json file that contains information about the experiment/database
- **sensors** (*object of class Sensor*) – Is an object of class sensor and is used to see if EEG extraction is required
- **database** (*SQL object | str*) – Is the SQL object that is created for accessing the SQL database | Name of the folder containing the CSV files
- **reading_method** (*str {"SQL","CSV"}*) – Describes which type of databse is to be used for data extraction

Returns **stimulus_object_dict** – dictionary of objects of class stimulus ordered by category

Return type dict

subjectAnalysis(*average_flag*, *standardise_flag*)

Function to find features for all stimuli for a given subject.

Does not return any value. It stores the calculated features/metadata in its *aggregate_meta* member variable. Can be accessed by an object of the class. For structure of this variable see *Subject*.

Parameters

- **average_flag** (*bool*) – If `True`, average values for a given meta variable under each stimulus type for a given stimulus.
- **standardise_flag** (*bool*) – If `True`, subtract *control_data* values for a given meta variable for each stimulus. See *getControlData* for more details on *control_data*.

subjectVisualize(*master*, *viz_type='individual'*, *sub_list=None*)

Visualization function to open the window for stimulus and plot type selection.

It is invoked internally by the *visualizaData* function.

timeIndexInitialisation(*stimulus_column_name*, *stimulus_name*, *df*)

This function that will retireve the index of the start, end and roi of a question

Parameters

- **stimulus_column_name** (*str*) – Name of the column where the stimuli names are present
- **stimulus_name** (*str*) – Name of the stimulus
- **df** (*pandas dataframe*) – Contains the data from which *start*, *end* and *roi* will be extracted from

Returns

- **start** (*int*) – The index of the start of a queation
- **end** (*int*) – The index of the end of a question
- **roi** (*int*) – The index when the eye lands on the region of interest

class Subject.**SubjectVisualize**(*master*, *subj_name*, *stimuli*, *json_file=None*, *viz_type='individual'*, *sub_list=None*)

Bases: object

button_click(*stim*, *stim_num=-1*)

10.5 etDataReader

etDataReader.**blink_detection**(*x*, *y*, *time*, *missing=0.0*, *minlen=10*)

Detects blinks, defined as a period of missing data that lasts for at least a minimal amount of samples

Parameters

- **x** (*array*) – Gaze x positions
- **y** (*array*) – Gaze y positions
- **time** (*array*) – Timestamps
- **missing** (*float*) – Value to be used for missing data (default = 0.0)
- **minlen** (*int*) – Minimal amount of consecutive missing samples

Returns

- **Sblk** (*list of lists*) – Each containing [starttime]
- **Eblk** (*list of lists*) – Each containing [starttime, endtime, duration]

etDataReader.**fixation_detection**(*x*, *y*, *time*, *missing=0.0*, *maxdist=25*, *mindur=50*)

Detects fixations, defined as consecutive samples with an inter-sample distance of less than a set amount of pixels (disregarding missing data)

Parameters

- **x** (*array*) – Gaze x positions
- **y** (*array*) – Gaze y positions
- **time** (*array*) – Timestamps
- **missing** (*float*) – Value to be used for missing data (default = 0.0)
- **maxdist** (*int*) – Maximal inter sample distance in pixels (default = 25)
- **mindur** (*int*) – Minimal duration of a fixation in milliseconds; detected fixation cadidates will be disregarded if they are below this duration (default = 100)

Returns

- **Sfix** (*list of lists*) – Each containing [starttime]
- **Efix** (*list of lists*) – Each containing [starttime, endtime, duration, endx, endy]

etDataReader.**read_edf**(*filename*, *start*, *stop=None*, *missing=0.0*, *debug=False*, *eye='B'*)

Returns a list with dicts for every trial.

Parameters

- **filename** (*str*) – Path to the file that has to be read
- **start** (*str*) – Trial start string
- **stop** (*str*) – Trial ending string (default = None)
- **missing** (*float*) – Value to be used for missing data (default = 0.0)

- **debug** (*bool*) – Indicating if DEBUG mode should be on or off; if DEBUG mode is on, information on what the script currently is doing will be printed to the console (default = False)
- **eye** (*str {'B','L','R'}*) – Which eye is being tracked? Deafults to 'B'-Both. ['L'-Left, 'R'-Right, 'B'-Both]

Returns data – With a dict for every trial. Following is the dictionary 0. x -array of Gaze x positions, 1. y -array of Gaze y positions, 2. size -array of pupil size, 3. time -array of timestamps, t=0 at trialstart, 4. trackertime -array of timestamps, according to the tracker, 5. events -dict {Sfix, Ssac, Sblk, Efix, Esac, Eblk, msg}

Return type list

etDataReader.**read_idf**(*filename*, *start*, *stop=None*, *missing=0.0*, *debug=False*)

Returns a list with dicts for every trial.

Parameters

- **filename** (*str*) – Path to the file that has to be read
- **start** (*str*) – Trial start string
- **stop** (*str*) – Trial ending string (default = None)
- **missing** (*float*) – Value to be used for missing data (default = 0.0)
- **debug** (*bool*) – Indicating if DEBUG mode should be on or off; if DEBUG mode is on, information on what the script currently is doing will be printed to the console (default = False)

Returns data – With a dict for every trial. Following is the dictionary 0. x -array of Gaze x positions, 1. y -array of Gaze y positions, 2. size -array of pupil size, 3. time -array of timestamps, t=0 at trialstart, 4. trackertime -array of timestamps, according to the tracker, 5. events -dict {Sfix, Ssac, Sblk, Efix, Esac, Eblk, msg}

Return type list

etDataReader.**read_tobii**(*filename*, *start*, *stop=None*, *missing=0.0*, *debug=False*)

Returns a list with dicts for every trial.

Parameters

- **filename** (*str*) – Path to the file that has to be read
- **start** (*str*) – Trial start string
- **stop** (*str*) – Trial ending string (default = None)
- **missing** (*float*) – Value to be used for missing data (default = 0.0)
- **debug** (*bool*) – Indicating if DEBUG mode should be on or off; if DEBUG mode is on, information on what the script currently is doing will be printed to the console (default = False)

Returns data – With a dict for every trial. Following is the dictionary 0. x -array of Gaze x positions, 1. y -array of Gaze y positions, 2. size -array of pupil size, 3. time -array of timestamps, t=0 at trialstart, 4. trackertime -array of timestamps, according to the tracker, 5. events -dict {Sfix, Ssac, Sblk, Efix, Esac, Eblk, msg}

Return type list

etDataReader.**replace_missing**(*value*, *missing=0.0*)

Returns missing code if passed value is missing, or the passed value if it is not missing; a missing value in the

EDF contains only a period, no numbers; NOTE: this function is for gaze position values only, NOT for pupil size, as missing pupil size data is coded '0.0'

Parameters

- **value** (*str*) – Either an X or a Y gaze position value (NOT pupil size! This is coded '0.0')
- **missing** (*float*) – The missing code to replace missing data with (default = 0.0)

Returns Either a missing code, or a float value of the gaze position

Return type float

etDataReader.**saccade_detection**(*x*, *y*, *time*, *missing=0.0*, *minlen=5*, *maxvel=40*, *maxacc=340*)

Detects saccades, defined as consecutive samples with an inter-sample velocity of over a velocity threshold or an acceleration threshold

Parameters

- **x** (*array*) – Gaze x positions
- **y** (*array*) – Gaze y positions
- **time** (*array*) – Timestamps
- **missing** (*float*) – Value to be used for missing data (default = 0.0)
- **minlen** (*int*) – Minimal length of saccades in milliseconds; all detected saccades with len(sac) < minlen will be ignored (default = 5)
- **maxvel** (*int*) – Velocity threshold in pixels/second (default = 40)
- **maxacc** (*int*) – Acceleration threshold in pixels / second**2 (default = 340)

Returns

- **Ssac** (*list of lists*) – Each containing [starttime]
- **Esac** (*list of lists*) – Each containing [starttime, endtime, duration, startx, starty, endx, endy]

10.6 formatBridge

formatBridge.**convertToBase**(*filename*, *sensor_type*, *device*, *stim_list=None*, *start='START'*, *stop=None*, *eye='B'*)

Master function that calls the different converter functions to convert to bas data format.

Internally invoked by *generateCompatibleFormats*.

Parameters

- **filename** (*str*) – Full path of the data file.
- **sensor_type** (*str {'EyeTracker'}*) – Type of sensor. Supports only 'EyeTracker' for now.
- **device** (*str {'eyelink', 'smi', 'tobii'}*) – Make of the sensor. Must be a type of eye tracker.
- **stim_list** (*list (str) | None*) – If None (default) the name of stimuli for a data file wil be generated sequentially as ["stim1", "stim2", "stim2", ...]. If it is a list of str, then the stimulus names will be taken from this list in the given order. Hence, the

length of stim_list and number of events/trials/simuli markers in the data should be the same.

- **start** (*str*) – The start of event marker in the data (Defaults to 'START').
- **stop** (*str*) – The end of event marker in the data (Defaults to None). If None, start of new event will be considered as end of previous event.
- **eye** (*str {'B','L','R'}*) – Which eye is being tracked? Deafults to 'B'-Both. ['L'-Left, 'R'-Right, 'B'-Both]

Returns The extracted data in the base format.

Return type pandas DataFrame

formatBridge.**db_create**(*data_path*, *source_folder*, *database_name*, *dtype_dictionary=None*, *na_strings=None*)

Create a SQL database from a csv file

Parameters

- **data_path** (*string*) – Path to the directory where the database is to be located
- **source_folder** (*string*) – Name of folder that contains the csv files
- **database_name** (*string*) – Name of the SQL database that is to be created
- **dtype_dictionary** (*dictionary*) – Dictionary mapping the names of the columns ot their data type
- **na_strings** (*list*) – Is a list of the strings that are to be considered as null value

formatBridge.**generateCompatibleFormat**(*exp_path*, *device*, *stim_list_mode='NA'*, *start='START'*, *stop=None*, *eye='B'*, *reading_method='SQL'*)

Function to convert data into the base format before starting analysis and visualization.

The function creates a directory called 'csv_files' inside the *Data* folder and stores the converted csv files in it. If *reading_method* is specified as 'SQL' then an SQL database is created inside the 'Data' folder but the user need not worry about it.

Parameters

- **exp_path** (*str*) – Absolute path to the experiment folder. If the path is a folder, the framework will assume its being run in the *Experiment Design* mode. If it is a path to a single data file, then the *Stand-alone Design* mode is assumed.
- **device** (*str {'eyelink', 'smi', 'tobii'}*) – Make of the sensor.
- **stim_list_mode** (*str {'NA', 'common', 'diff'}*) – See the Using PyTrack in the Introduction for details on which of the three to suply.
- **start** (*str*) – The start of event marker in the data (Defaults to 'START').
- **stop** (*str*) – The end of event marker in the data (Defaults to None). If None, start of new event will be considered as end of previous event.
- **eye** (*str {'B','L','R'}*) – Which eye is being tracked? Deafults to 'B'-Both. ['L'-Left, 'R'-Right, 'B'-Both]
- **reading_method** (*str {'CSV', 'SQL'}*) – 'SQL' (default) reading method is faster but will need extra space. This affects the internal functioning of he framework and the user can leave it as is.

formatBridge.**getColHeaders**()

Function to return the column headers for the *PyTrack* base format data representation.

formatBridge.**toBase**(*et_type*, *filename*, *stim_list=None*, *start='START'*, *stop=None*, *eye='B'*)

Bridge function that converts SMI, Eyelink or Tobii raw eye tracking data files to the base CSV format that the framework uses.

Parameters

- **et_type** (*str {"smi","tobii", "eyelink"}*) – Which eye tracker
- **filename** (*str*) – Full file name (with path) of the data file
- **stim_list** (*list (str)*) – Name of stimuli as a list of strings. If there are n trials/events found in the data, the length of stim_list should be n containing the names of stimuli for each trial/event.
- **start** (*str*) – Marker for start of event in the .asc file. Default value is 'START'.
- **stop** (*str*) – Marker for end of event in the .asc file. Default value is None. If None, new event/trial will start when start trigger is detected again.

Returns **df** – Pandas dataframe of the data in the framework friendly base csv format

Return type pandas DataFrame

Python Module Index

Index

S

T

V

W