MiTgaze

Release 0.0.2

Vahid S. Bokharaie

Oct 12, 2020

CONTENTS:

CHAPTER

ONE

FIRST STEP

MiTgaze, A Python Library to Analyse Gaze Behaviour, Made in Tübingen.

At the moment, it can only import Eye-Tracking data files imported from Tobii Pro Lab recording software. But the main functions can be easily adapted for other types of Eye-Tracking recorders.

1.1 Requirements

```
"seaborn",
"pathlib",
"pandas",
"numpy",
"scipy",
"matplotlib",
"scikit-learn",
"openpyxl",
"scikit-image",
"joblib",
"opencv-python",
"pillow",
"pandas",
"matplotlib",
```

1.2 Compatibility

This code is tested under Python 3.8.

1.3 License

GNU General Public License (Version 3).

1.4 Authors

MiTgaze is maintained by Vahid Samadi Bokharaie.

CHAPTER

TWO

BASIC USAGE

There are a few examples in the *Examples* subfolder that shows how the code can be used for different purposes. If source directories are added properly, the scripts should work for any TSV or CSV files imported from Tobii Lab Pro.

First, you should parse your raw data files. You can copy all your TSV/CSV files in a folder, and then run the *script_parse_df.py*. It will read each data file, parse them based on Presented Media Names and also Participant Names. So, if you have many recordings of different subjects for one project, copy them all in a folder and the script will give you TSV files split based on Media and Participant names.

If you want to get an overview of your TSV/CSV file, you can use *mitgaze.util.describe*.

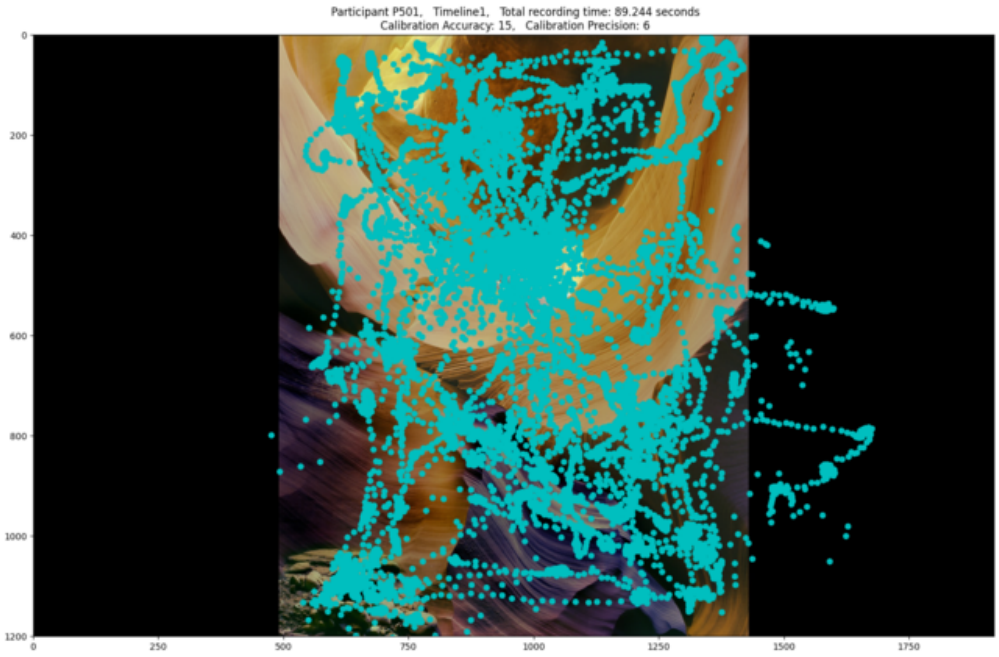
CHAPTER

THREE

BASIC PLOTS

To see how you can generate gaze plots and heatmaps, you can have a look at the *script_plot_basics.py*. This script assumes you are dealing with data files parsed based on the media names.

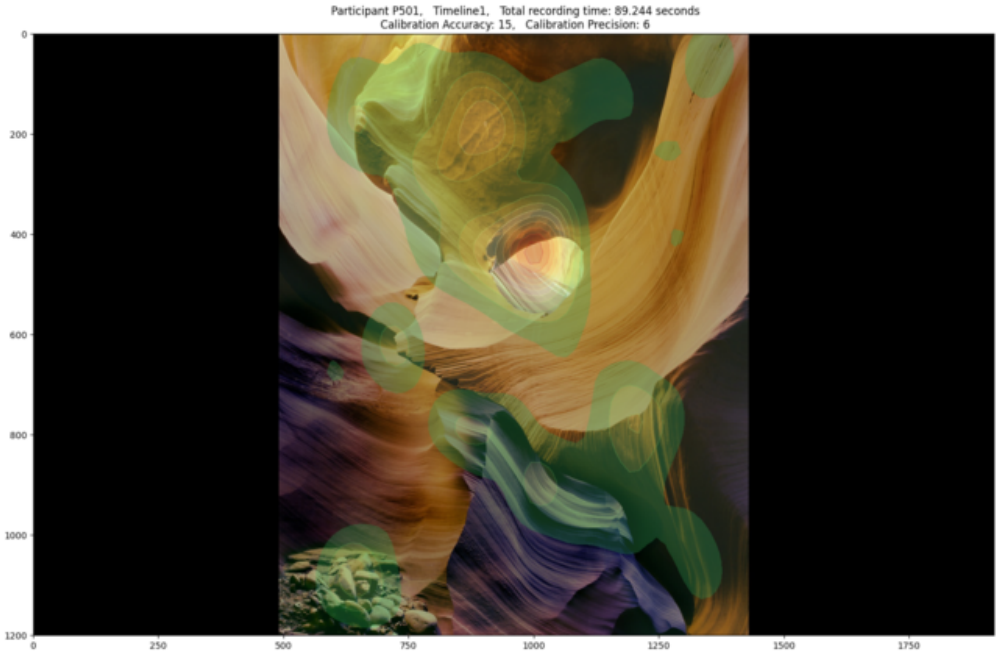
At the moment, the library generates two types of gaze plots, one that highlights order with a colormap, and another type that uses a solid colour for all gaze points. Below, you can see examples of such plots generated by MiTgaze.



And you can also generate heatmaps.

Please note that the heatmap data can also be saved separately for further analysis. The function used to generate the heatmap is *kdeplot* from *seaborn*. Only I have made minor changes to it so it returns the *xx, yy, z* values to the caller function.


Participant P501, Timeline1, Total recording time: 89.244 seconds
Calibration Accuracy: 15, Calibration Precision: 6
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750


Participant P501, Timeline1, Total recording time: 89.244 seconds
Calibration Accuracy: 15, Calibration Precision: 6
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750

CHAPTER

FOUR

AREAS OF INTEREST (AOIS)

If you want to deal with AOIs, the code assumes you have made a copy of your media files (image files), and painted over the Area of Interest with one of R, G, B, C, Y or M colours. Such images can be generated manually using the Tobii Lab Pro Software. Given such images are available, code will generate a pandas Data-frame which includes the time stamps of all occasions in which the gaze has entered and exited the AOI. To avoid the situations in which the effects of saccads or inaccuracies in recording lead to too many in/out time-stamps for a certain AOI, you can use a smoothing factor. Every in/out time-stamp pair which happens faster than the specified threshold would be ignored.

You can look at *script_AOI_events.py in `examples* subfolder.

CHAPTER

FIVE

MITGAZE PACKAGE

5.1 Submodules

5.2 mitgaze.AOI module

Created on Wed Oct 7 19:42:28 2020.

@author: vbokharaie

mitgaze.AOI.**AOI_in_out_indices**(*x*, *y*, *mask*, *smooth_factor=15*)

Find indices in which gaze entered and exits an AOI.

x [1D numpy array or list of int] x coordintaes of gaze.

y [1D numpy array or list of int] y coordintaes of gaze.

mask [set of (int, int)] set of (x,y) indices of AOI pixels.

smooth_factor [int, optional] Any in-out or out-in period less than this value is ignored. The default is 15.

TYPE DESCRIPTION.

mitgaze.AOI.**find_AOI**(*image_name*, *colour='red'*)

Find the AOI based on colour-codes.

If AOI is painted over a photo with one of the main 6 colours, return the set of all (x,y) corrdinates of the AOI pixels.

image_name [Nmpay array] Image containing AOI colour-coded in r,g,b,c,y or m.

colour [str, optional] Can be one of r, g, b, c, m, y letters or fulle colour names. The default is 'red'.

mask [set of (int, int)] Set of all pixels in the mask.

5.3 mitgaze.bplots module

Created on Fri Sep 6 11:03:09 2019.

@author: vbokharaie

mitgaze.bplots.**plot_basics_parsed_media**(*filename*, *dir_save_main*, *dir_source_image_1200*, *dir_source_image_1080=None*)

Get an Eye-Tracking Recording tsv/csv file as input, plot gaze_basic, gaze_order and heatmap.

Saves them in two seperate folders, one under the Media Name folder, other arranged based on participant names.

filename [str or pathlib Path] a tsv or csv file inclduing the Eye-Tarcking Recording parsed per media..

dir_save_main [str or pathlib Path] Where to save.

dir_source_image_1200 [str or pathlib Pathstr or pathlib Path] Where are media files (jpg or png). If recordings are done in different screens, use both parameter.

dir_source_image_1080 [str or pathlib Pathstr or pathlib Path, (Optional)] IF media recordings are done in two screens with two resouloutions, use this extra parameter .

None.

mitgaze.bplots.**plot_gaze**(*x*, *y*, *img*, *order=True*, *ax=None*, *title_str=''*)

Plot the gaze data. Ordered or non-ordered.

x [numpy.array] array of x values.

y [numpy.array] array of y values.

img :png or jpg image. the image used as stimulus while recording x-y values..

order [Bool, optional] If order of gaze values should be highlighted using a certain colormap. The default is True.

ax [matploblib figure axis, optional] axis of the figure to be used for plotting. The default is None.

title_str [str, optional] Optional text to be used as title. The default is ‘’.

ax [matploblib figure axis] axis of the current figure.

mitgaze.bplots.**plot_gaze_df**(*df_media*, *dir_source_image*, *dir_save_main*, *plot_type='GAZE_ORDER'*, *if_save_media=False*)

Get an Eye-Tracking Recording Dataframe as input, plot gaze_basic, gaze_order.

Saves them in two seperate folders, one under the Media Name folder, other arranged based on participant names.

df_media [pandas Dataframe] Dataframe of Eye-tracking data.

dir_source_image [str or pathlib Path] where the media files are.

dir_save_main [str or pathlib Path] where to save.

plot_type [str, optional] what kind of gaze plot. Either color-code order or monochrome plot. The default is ‘GAZE_ORDER’.

if_save_media [Bool, optional] if save the plots. The default is False.

None.

mitgaze.bplots.**plot_heatmap**(*x*, *y*, *img*, *ax=None*, *gridsize=100*, *title_str=''*)

Plot heatmaps based on gaze data.

x [numpy.array] array of x values.

y [numpy.array] array of y values.

img :png or jpg image. the image used as stimulus while recording x-y values..

ax [matplotlib axes, optional] axis of the figure to be used for plotting. The default is None.

gridsize [int, optional] Number of discrete points in the evaluation grid. default is 100 (pixels)

title_str [str, optional] Optional text to be used as title. The default is ‘’.

ax [matplotlib axes,] Axes to plot on, otherwise uses current axes..

xx: numpy array x-grid coordinates for heatmaps.

yy: numpy array y-grid coordinates for heatmaps.

z: 2D numpy array KDE values for each cell.

mitgaze.bplots.**plot_heatmap_df**(*df_media*, *dir_source_image*, *dir_save_main*, *if_save_fig=True*)

df_media [pandas Dataframe] Dataframe of Eye-tracking reocrding.

dir_source_image [str or pathlib Path] where the media fiels are.

dir_save_main [str or pathlib Path] where to save.

if_save_fig [Bool, optional] if save the figures to disk. The default is True.

xx: numpy array x-grid coordinates for heatmaps.

yy: numpy array y-grid coordinates for heatmaps.

z: 2D numpy array KDE values for each cell.

5.4 mitgaze.data_info module

Created on Fri Jun 5 14:49:53 2020

@author: vbokharaie

mitgaze.data_info.**get_dir_source**(*data_type*)

mitgaze.data_info.**get_filename**(*data_type=None*, *subject_ID=None*)

mitgaze.data_info.**get_sub_folder**(*data_type=None*, *subject_ID=None*)

5.5 mitgaze.file_io module

5.6 mitgaze.filters module

Created on Mon Sep 9 14:12:36 2019.

@author: vbokharaie

mitgaze.filters.**edge_detect**(*img*, *alg_type='roberts'*)

Detect edges of an image using various algorithms from skimage.filters.

img [2-D array] Image to process.

alg_type [str, optional] EDge-detection algortihm type. The default is 'roberts'.

edge [2-D array] The Cross edge map..

mitgaze.filters.**plot_edges**(*image_filename*, *dir_save_filters*)

Plot edges of an image and save the result as an image.

image_filename [TYPE] DESCRIPTION.

dir_save_filters [TYPE] DESCRIPTION.

None.

5.7 mitgaze.sns_distributions module

This is a modified version of the Seaborn module.

I changed it so that I can get the data used to generate the heatmap when using kdeplot funciton. @modified by: Vahid Bokharaie

@author: Seaborn team

This is simply _distributions module from Seaborn library. Changed slightly, so kdeplot returns xx, yy and z values used to generate the heatmap. Apart from that, no other chnages are made.

I forgot which exact version of seaborn was it that I used. Module was chnaged in November 2019. @edited by: Vahid Bokharaie

mitgaze.sns_distributions.**distplot**(*a*, *bins=None*, *hist=True*, *kde=True*, *rug=False*, *fit=None*, *hist_kws=None*, *kde_kws=None*, *rug_kws=None*, *fit_kws=None*, *color=None*, *vertical=False*, *norm_hist=False*, *axlabel=None*, *label=None*, *ax=None*)

Flexibly plot a univariate distribution of observations.

This function combines the matplotlib `hist` function (with automatic calculation of a good default bin size) with the seaborn *kdeplot()* and *rugplot()* functions. It can also fit `scipy.stats` distributions and plot the estimated PDF over the data.

a [Series, 1d-array, or list.] Observed data. If this is a Series object with a `name` attribute, the name will be used to label the data axis.

bins [argument for matplotlib hist(), or None, optional] Specification of hist bins, or None to use Freedman-Diaconis rule.

hist [bool, optional] Whether to plot a (normed) histogram.

kde [bool, optional] Whether to plot a gaussian kernel density estimate.

rug [bool, optional] Whether to draw a rugplot on the support axis.

fit [random variable object, optional] An object with *fit* method, returning a tuple that can be passed to a *pdf* method a positional arguments following an grid of values to evaluate the pdf on.

{hist, kde, rug, fit}_kws [dictionaries, optional] Keyword arguments for underlying plotting functions.

color [matplotlib color, optional] Color to plot everything but the fitted curve in.

vertical [bool, optional] If True, observed values are on y-axis.

norm_hist [bool, optional] If True, the histogram height shows a density rather than a count. This is implied if a KDE or fitted density is plotted.

axlabel [string, False, or None, optional] Name for the support axis label. If None, will try to get it from a.namel if False, do not set a label.

label [string, optional] Legend label for the relevent component of the plot

ax [matplotlib axis, optional] if provided, plot on this axis

ax [matplotlib Axes] Returns the Axes object with the plot for further tweaking.

kdeplot [Show a univariate or bivariate distribution with a kernel] density estimate.

rugplot [Draw small vertical lines to show each observation in a] distribution.

Show a default plot with a kernel density estimate and histogram with bin size determined automatically with a reference rule:

Use Pandas objects to get an informative axis label:

Plot the distribution with a kernel density estimate and rug plot:

Plot the distribution with a histogram and maximum likelihood gaussian distribution fit:

Plot the distribution on the vertical axis:

Change the color of all the plot elements:

Pass specific parameters to the underlying plot functions:

`mitgaze.sns_distributions.`**`kdeplot`**(*data*, *data2=None*, *shade=False*, *vertical=False*, *kernel='gau'*, *bw='scott'*, *gridsize=100*, *cut=3*, *clip=None*, *legend=True*, *cumulative=False*, *shade_lowest=True*, *cbar=False*, *cbar_ax=None*, *cbar_kws=None*, *ax=None*, ***kwargs*)

Fit and plot a univariate or bivariate kernel density estimate.

data [1d array-like] Input data.

data2: 1d array-like, optional Second input data. If present, a bivariate KDE will be estimated.

shade [bool, optional] If True, shade in the area under the KDE curve (or draw with filled contours when data is bivariate).

vertical [bool, optional] If True, density is on x-axis.

kernel [{'gau' | 'cos' | 'biw' | 'epa' | 'tri' | 'triw' }, optional] Code for shape of kernel to fit with. Bivariate KDE can only use gaussian kernel.

bw [{'scott' | 'silverman' | scalar | pair of scalars }, optional] Name of reference method to determine kernel size, scalar factor, or scalar for each dimension of the bivariate plot. Note that the underlying computational libraries have different interperetations for this parameter: `statsmodels` uses it directly, but `scipy` treats it as a scaling factor for the standard deviation of the data.

gridsize [int, optional] Number of discrete points in the evaluation grid.

cut [scalar, optional] Draw the estimate to cut * bw from the extreme data points.

clip [pair of scalars, or pair of pair of scalars, optional] Lower and upper bounds for datapoints used to fit KDE. Can provide a pair of (low, high) bounds for bivariate plots.

legend [bool, optional] If True, add a legend or label the axes when possible.

cumulative [bool, optional] If True, draw the cumulative distribution estimated by the kde.

shade_lowest [bool, optional] If True, shade the lowest contour of a bivariate KDE plot. Not relevant when drawing a univariate plot or when `shade=False`. Setting this to `False` can be useful when you want multiple densities on the same Axes.

cbar [bool, optional] If True and drawing a bivariate KDE plot, add a colorbar.

cbar_ax [matplotlib axes, optional] Existing axes to draw the colorbar onto, otherwise space is taken from the main axes.

cbar_kws [dict, optional] Keyword arguments for `fig.colorbar()`.

ax [matplotlib axes, optional] Axes to plot on, otherwise uses current axes.

kwargs [key, value pairings] Other keyword arguments are passed to `plt.plot()` or `plt.contour{f}` depending on whether a univariate or bivariate plot is being drawn.

ax [matplotlib Axes] Axes with plot.

distplot: Flexibly plot a univariate distribution of observations. jointplot: Plot a joint dataset with bivariate and marginal distributions.

Plot a basic univariate density:

Shade under the density curve and use a different color:

Plot a bivariate density:

Use filled contours:

Use more contour levels and a different color palette:

Use a narrower bandwith:

Plot the density on the vertical axis:

Limit the density curve within the range of the data:

Add a colorbar for the contours:

Plot two shaded bivariate densities:

`mitgaze.sns_distributions.`**`rugplot`**(*a*, *height=0.05*, *axis='x'*, *ax=None*, ***kwargs*)

Plot datapoints in an array as sticks on an axis.

a [vector] 1D array of observations.

height [scalar, optional] Height of ticks as proportion of the axis.

axis [{'x' | 'y'}, optional] Axis to draw rugplot on.

ax [matplotlib axes, optional] Axes to draw plot into; otherwise grabs current axes.

kwargs [key, value pairings] Other keyword arguments are passed to `LineCollection`.

ax [matplotlib axes] The Axes object with the plot on it.

5.8 mitgaze.util module

Created on Tue Feb 18 17:58:19 2020.

@author: vbokharaie

mitgaze.util.**describe**(*df*, *media_names=None*)
A summary of an Eye-Tracking recording, per media.

df [Pandas Dataframe] includes the Eye-tracking data.

media_names [name os Presented Media to consider, optional] If not used, all Presented Media in the df would be considered. The default is None.

df_out [Pandas Dataframe] Summary of df. Few cells are dict encapsulated as list to make code more straightforward.

mitgaze.util.**df_columns**(*df*, *if_len=False*, *loc=20*)
Print the columns of Pandas df, sorted alphabetically.

df [pandas Datafame] the df.

if_len [Bool, optional] If print length of df. The default is False.

loc [int, optional] start th eprint from a certain location in the list of column names. The default is 20.

None.

mitgaze.util.**divide_images_in_folder_2_segments**(*dir_source*, *dir_save*, *nw=4*, *nh=4*)
Read all images in a folder and segment them and then save.

dir_source [pathlib Path or str] dir_source.

dir_save [pathlib Path or str] dir_save.

nw [int, optional] number of segments in width. The default is 4.

nh [int, optional] number of segments in height. The default is 4.

None.

mitgaze.util.**divide_img_2_segments**(*img*, *nw=4*, *nh=4*)
Segment each image, gemoterically, uniform steps.

img [numpy array.] The image to be segmented.

nw [int, optional] number of segments in width.. The default is 4.

nh [int, optional] number of segments in height. The default is 4.

list_w [list of int] x corrdinates of segments.

list_h [list of int] y corrdinates of segments.

mitgaze.util.**img_fill**(*img*, *W_screen=1920*, *H_screen=1200*)
Fill image with solid black such that it fits the screen width and height.

img [numpay array] image.

W_screen [int, optional] Screen width. The default is 1920.

H_screen [int, optional] Screen height. The default is 1200.

img_filled: numpy array Filled imge or None if input image dimension is not 2 or 3

mitgaze.util.**parse_tobii_output**(*filename*, *dir_save*, *overwrite=False*)
Read Tobii Eye-tracker output files.

Then splits it to subset of project name / participant name / timeline name and saves.

filename [pathlib Path or str] name of the tsv/csv file.

dir_save [pathlib Path or str] dir_save.

overwrite [Bool, optional] Overwrite if destinations filename already exists? The default is False.

list_files_saved [list of pathlib.Path] list of saved files.

mitgaze.util.**remove_nan**(*x_with_nan*, *y_with_nan*)
Remove nan from x/y data.

x_with_nan [numpy array] x data.

y_with_nan [numpy array] y data.

x [numpy array] x data without nan.

y [numpy array] y data without nan.

mitgaze.util.**rescale_fill_media**(*media_file*, *dir_save*, *W_presented*, *H_presented*, *W_screen=1920*, *H_screen=1200*)
Rescale each image.

media_file [pathlib Path or str] filename for media.

dir_save [pathlib Path or str] save folder.

W_presented [int] Width media.

H_presented [int] Height of media.

W_screen [int, optional] Screen width. The default is 1920.

H_screen [int, optional] Screen height. The default is 1200.

None.

mitgaze.util.**rescale_fill_save_folder**(*dir_source_media*, *tsv_file*, *dir_save*)
Rescale images.

Reads all images in a folder, and scale them based on Screen W/H read from tsv file, fill surrounding areas of image with black, and then save them.

dir_source_media [pathlib Path or str.] folder containig media files.

tsv_file [pathlib Path or str] filename, output of Eye-Tracking recorder tsv or csv.

dir_save [pathlib Path or str] save folder.

None.

5.9 Module contents

MiTgaze - python-based Analysis Tool to study gaze behavior, Made in Tuebingen.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX