

# Blind Spot Mapping User Manual

## Contents

1. A Quick Introduction .....	3
2. Installation and Required Python Libraries .....	5
2.1. Install Psychopy v3.2.3 .....	5
2.2. Install Python libraries.....	5
2.3. Fix the eye-tracker bug.....	6
2.4. Fix log-displaying bug.....	7
3. Preparation.....	8
3.1. Apparatus .....	8
3.2. Save monitor information to Psychopy.....	8
3.3. Experiment configuration files .....	9
4. During Experiment.....	23
4.1. Run experiment script .....	23
4.2. Configure experiment in dialog boxes .....	24
4.3. Lead-in section .....	27
4.4. Border points detection section .....	28
4.5. Staircase section .....	30
4.6. Validation section.....	32
4.7. Completion section.....	37
5. Data .....	38
5.1. JSON and CSV files .....	38
5.2. PSYDAT files .....	44
6. About the Script .....	45
6.1. Visual stimuli unit .....	45
6.2. Influence of screen and visual stimuli parameters.....	45
6.3. Influence of “inward” and “outward” tests in the “Border points detection section” .....	46

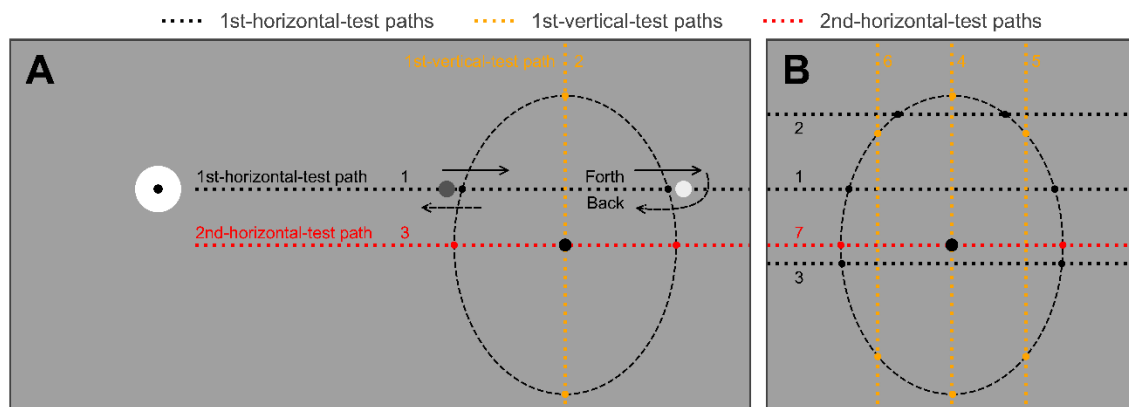
6.4. Excluding in-situ repeated clicks in the “Border points detection section” .....	46
6.5. The blind spot location estimated in the “Border points detection section” will not be adjusted according to the final border points after the “Staircase section” .....	47
6.6. Default staircase rules are unusual.....	48
6.7. Drift correction.....	48
6.8. Combine the script with other experiments.....	48

# 1. A Quick Introduction

This experiment script psychophysically maps the blind spot in the monocular visual field in human subjects. The blind spot refers to a physiological scotoma in the temporal monocular visual field that corresponds to the photoreceptor-free optic disc in the nasal retina. The blind spot is on average  $6^\circ$  wide  $7^\circ$  high, centered about  $16^\circ$  temporal to the fixation and  $2^\circ$  below the horizontal meridian.

People do not experience the hole in their vision. In binocular vision, the missing information is compensated by visual inputs from the fellow eye, while in monocular vision, it is believed to be “filled-in” with information inferred from remote or global retinal inputs.

Due to filling-in, the border points of the blind spot can be estimated by reporting the positions where a visual object perceptually “disappears” and “appears” in the monocular visual field, when the subject is gazing at a fixation object. This experiment script uses this strategy to estimate the blind spot location and size, asking the subject to move a cursor across the blind spot borders and report the positions where the cursor perceptually appears and reappears. These move-and-click tasks are performed at 3 stepwise stages in the order of horizontal, vertical, and horizontal again. By involving varying numbers of motion paths in these 3 stages, varying spatial resolutions of estimates can be achieved (Figure 1 in our paper).



The estimate can be refined in a staircase procedure that calibrates each blind spot border point. Then the estimated blind spot area formed by border points can be validated in a validation procedure that examines the overall quality of estimate. The procedure to perimetrically map the visual sensitivity across the relevant part of the monocular visual field is also provided to accurately map the blind spot. For the full details about these three methods used in this experiment script, please reference our paper.

This script was written with Python 3.6 and tested in PsychoPy v3.2.3 in Windows 7 and 10 systems. Compatibility with higher Psychopy and Python versions and other operating systems is not guaranteed. This script supports interactions with an Eyelink 1000 eye-tracker, but the script can run alone and eye-trackers are not necessarily needed. We only tested Eyelink 1000, so all “eye-tracker”s in the manual refer to “Eyelink 1000” only. Although Psychopy also supports other eye-tracker devices of GazePoint, SR Research, and Tobii, these models have not been tested.

## **2. Installation and Required Python Libraries**

### **2.1. Install Psychopy v3.2.3**

This tool is a Psychopy experiment script, so Psychopy is required. You have two options to get Psychopy installed on your computers: installation as standalone software or as a library of Python 3.

Standalone installation is recommended. You should download a Psychopy version 3.2.3 setup package on the official website <https://www.psychopy.org/>, and follow the instructions.

For add-on installation as a Python 3 library, you should run the pip install command:

```
pip install psychopy==3.2.3
```

Or reference the instructions given by Psychopy through this link <https://www.psychopy.org/download.html>.

### **2.2. Install Python libraries**

To run this script, you should get numpy, pandas, sklearn, webcolors, psychopy, and pylink (if you want to use Eyelink devices) installed in your Python 3 environment.

If you installed a standalone Psychopy v3.2.3, it comes with a built-in Python 3.6 environment with some of the required dependencies. However, sklearn and webcolors are not included and numpy version is not compatible with sklearn, so please install/update the required libraries manually and specifically for the Psychopy Python but NOT for the Python that is already installed on your computer if you do have one. The pylink library is included in a standalone Psychopy.

If you installed psychopy v3.2.3 as an add-on library of your existing Python 3.6, please also manually update/install all the dependencies. Please DO NOT pip install “pylink” through PyPI, because you will get a wrong

library. The correct “pylink” is the one developed by SR Research, which has not gone to PyPI, so you must request this library from SR Research.

### 2.3. Fix the eye-tracker bug

This script uses the Psychopy API to handle the communication between the program and the eye-tracker. If you use Psychopy v3.2.3/v3.2.4, you may have problems in communication between the experiment computer (i.e., the computer running Psychopy) and the eye-tracker computer – the eye-tracker may not respond to the requests sent by the experiment computer.

In this case, you should amend a Psychopy file. The path may look like this:

“...\psychopy\psychopy\iohub\devices\eyetracker\hw\sr\_research\eyelink\eyetracker.py”.

Open the “eyetracker.py” file and remove lines 400 and 401 that are:

```
if starting_state != EyeTrackerConstants.DEFAULT_SETUP_PROCEDURE:  
    printExceptionDetailsToStdErr()
```

Then the bug may be fixed. For more information about this error, please look at the Psychopy forum, especially these two posts:

<https://discourse.psychopy.org/t/nothing-happens-after-calling-runsetupprocedure/9885>, and <https://discourse.psychopy.org/t/typeerror-when-initializing-iohub-for-eyetracking/5148>.

If the problem is not fixed, try to amend the relevant codes and files (“BS\_mapping.py” and “eyetrackingconfig.json”) of this script and directly use the lower-level library “pylink”. The relevant codes of this script should be headed by “Eye\_Tracker” or “psychopy.iohub” in the file “BS\_mapping.py”. Please also amend the configuration file “eyetrackingconfig.json”. The alternative codes to replace the original codes of this script can be found in this post in the Psychopy forum:

<https://discourse.psychopy.org/t/typeerror-when-initializing-iohub-for-eyetracking/5148>. You may need to reference the Psychopy documentation for programming, which can be found on <https://www.psychopy.org/documentation.html>. The section about psychopy.iohub is on <https://www.psychopy.org/api/api.html>.

## 2.4. Fix log-displaying bug

This script supports intra-experiment displaying of some log information on the screen (see the “Preparation” and “During Experiment” sections of this manual for more details). However, this may lead to program crashes on some computers and you may see Psychopy raising an error “AssertionError: Invalid anchor” which is associated with `pyglet.text.layout` module.

In this case, you may turn off the feature by modifying the configuration files and turning off this function on the dialog box (see “Preparation” section regarding “Show\_Raw\_Data” of this manual for more details), and never press “D” during the experiment, because this will turn on the function.

If you want this feature to work anyway, modify the file “BS\_mapping.py”. You should replace default values assigned to arguments “alignHoriz” and “alignVert” with “center” for the method “`visual.TextStim`” that is wrapped in two functions “`create_mndnp`” and “`create_info`”. Then you need to assign a suitable value to the argument “pos” for the method “`visual.TextStim`” in both functions to place the text information at suitable positions on your screen.

## 3. Preparation

### 3.1. Apparatus

To run this experiment, you should equip a visual lab room with:

- (1) An experiment PC running Windows 7/10 and having Psychopy v3.2.3 and all required libraries installed.
- (2) A monitor that can display colorful visual stimuli. Connect this monitor to the experiment PC, and remember to calibrate its gamma values in an experimental setting (e.g., in a dark room if you want to run this experiment in darkness) and measure its resolution and size of the screen (not the outer covering/casing).
- (3) A computer mouse for the human subject to move a cursor and click.
- (4) A keyboard for both the experimenter and human subject to control the experiment and to respond to visual stimuli.
- (5) A speaker/headphone/earphone to display sound indications (optional, the experiment can run without sound-displaying).
- (6) A chinrest to fix the human subject's head. Set the chinrest to the center of the screen and measure the distance between the chinrest and the screen center.
- (7) An eye and/or eyeglass mask to cover one eye of the human subject.
- (8) A chair for the human subject.
- (9) An Eyelink 1000 system or other eye-tracker models (optional), and make sure to connect its computer with the experiment PC.

### 3.2. Save monitor information to Psychopy

You should register the monitor you are about to use in the “Monitor center” of Psychopy, so that Psychopy can accurately display visual stimuli.

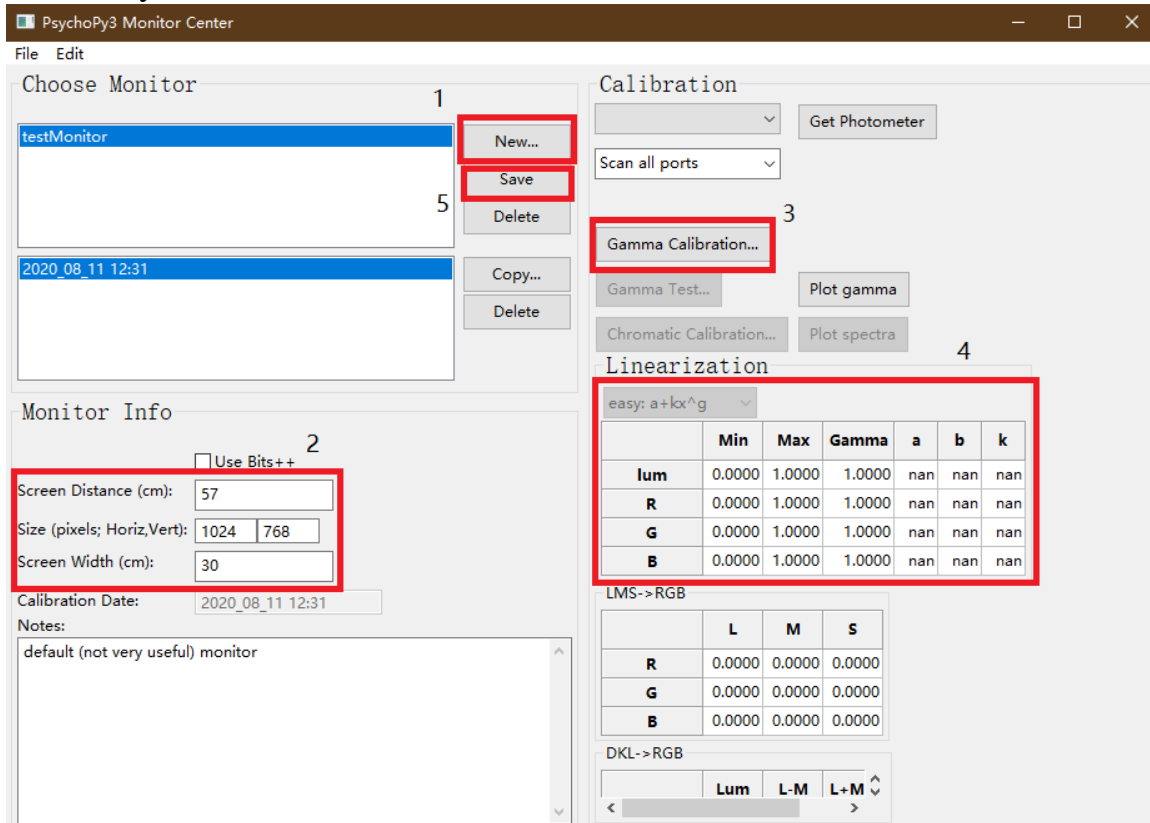
#### 1. The “monitor center” icon in Psychopy GUI.



2. You should input some information of your monitor. First, make a new monitor. Second, input its screen size in cm and pix. Third, do gamma



calibration. You will need a photometer to do this. The “Gamma Calibration” button is used to display a set of full-screen, pure-color graphs, and you have to measure the luminance value for each graph. Fourth, after the gamma calibration, input your results. Fifth, name your monitor and save it. Remember its name, which will be used later on. Finally, exit Monitor center.



### 3.3. Experiment configuration files

Some experiment settings can be done before running the experiment script “BS\_mapping.py”, but note that they will be finalized in the dialog box after you run the script.

There are 5 JSON files: “expconfig.json”, “eyetrackingconfig.json”, “logconfig.json”, “staircaseconfig.json”, and “validationconfig.json”. Please do NOT change their names, otherwise, remember to update their names in “BS\_mapping.py”.

1. “expconfig.json”. This file contains the default settings regarding experiment numbering (“Session”), human subject information

(“Participant”, “Sex”, “Age”), screen information, fixation object, cursor object, the testing mode for the experimental section “Border points detection section” (see our paper for details), and the selection of optional experimental sections.

```
1 {"Session": "test", "Participant": "test", "Sex": "male", "Age": "24", "Monitor_Name":  
  "testMonitor", "Monitor_Size(pix)": "1280, 960", "Monitor_Size(cm)": "40, 30", "Distance(cm)":  
  "57", "BG_Color": "128, 128, 128", "Fix_Color": "white", "Fix_Size(pix)": "40", "Fix_Pos(pix)":  
  "300, 0", "Cursor_Color": "white", "Cursor_Size(pix)": "12", "Flickering(Hz)": "20", "Test_Eye":  
  ["RIGHT", "LEFT"], "Mode": ["Low", "Medium", "High"], "Horizontal_Trials": "Default",  
  "Vertical_Trials": "Default", "Show_Raw_Data": ["False", "True"], "Staircase": ["True", "False"],  
  "Validation": ["Scaling", "Heat_Map", "No"], "Eye_Tracker": ["True", "False"]}]
```

Parameter description:

“Session”: The label for the current experiment, str.

“Participant”: The label for the current human subject, str.

“Sex”: Sex of the current human subject, str.

“Age”: Age of the current human subject, str.

“Monitor\_Name”: Name of the monitor, str.

“Monitor\_Size(pix)”: Size of the screen in pix, str in the form of “width, height”.

“Monitor\_Size(cm)”: Size of the screen in cm, str in the form of “width, height”.

“Distance(cm)”: Viewing distance (face to screen center) in cm, str.

“BG\_Color”: Screen background color in RGB values, str in the form of “R, G, B”.

“Fix\_Color”: Fixation object (a bull’s eye) outer annulus color by color name, str. The inner disk color is the opposite color to the outer annulus color.

“Fix\_Size(pix)”: Fixation object outer diameter in pix, str.

“Fix\_Pos(pix)”: Fixation object position relative to the screen in pix with the screen center being (0, 0), str in the form of “horizontal, vertical”.

“Cursor\_color”: Cursor (used in the “Border points detection section”) color by name, str. If cursor-flickering is not 0, then there will be two colors and another color will be the opposite color to the color specified here.

“Cursor\_Size(pix)”: Cursor diameter in pix, str.

“Flickering(Hz)”: The flickering frequency of the cursor in Hz, str. When set to 0 the cursor will not flicker.

“Test\_Eye”: The eye to be tested, either “RIGHT” or “LEFT”. DO NOT modify parameter values in this file, but you can change the order of values. You can select one option in the dialog box after running the script.

“Mode”: The testing mode for the “Border points detection section”, “Low”, “Medium”, or “High”. DO NOT modify these values in this file, but you can change the order of them. You can select one option in the dialog box after running the script.

In “Low” mode, there will be 1 path for both the “1st-horizontal-test” and “1st-vertical-test”.

The “1st-horizontal-test” path will lay on the screen horizontal meridian (S\_v0). It estimates an intermediate blind spot width (BS\_W1) and the blind spot horizontal center that is the horizontal midpoint of BS\_W1 ( $BS\_h0 = \text{midpoint}_h(BS\_W1)$ ).

The “1st-vertical-test” path will pass the blind spot horizontal center (BS\_h0) and it estimates the blind spot height (BS\_H) and the blind spot vertical center (BS\_v0) that is the vertical midpoint of BS\_H.

Finally, the “2nd-horizontal-test” path will pass the estimated blind spot center ([BS\_h0, BS\_v0]) and it estimates the blind spot width (BS\_W).

In “Medium” mode:

There will be 3 paths for the “1st-horizontal-test”, and they will be:

**the screen horizontal meridian ( $S_{v0}$ ),**  
 **$7.5^\circ/4$  higher than the screen horizontal meridian ( $S_{v0} + 7.5^\circ/4$ ),**  
**and  $7.5^\circ/4$  lower than the screen horizontal meridian ( $S_{v0} - 7.5^\circ/4$ ),**

where  $7.5^\circ$  is an arbitrarily chosen blind spot height. Deg is converted to pix according to “Distance(cm)”, “Monitor\_Size(cm)”, and “Monitor\_Size(pix)”.

These 3 horizontal paths estimate 3 intermediate blind spot widths (BS\_W1, BS\_W2, and BS\_W3). The means of the horizontal midpoints of these 3 intermediate widths determine the blind spot horizontal center:

**$BS_{h0} = \text{mean}(\text{midpoint}_h(BS\_W1), \text{midpoint}_h(BS\_W2), \text{midpoint}_h(BS\_W3)).$**

The 3 paths for the “1st-vertical-test” will pass:

**the blind spot horizontal center ( $BS_{h0}$ ),**  
 **$1/4$  the maximum intermediate blind spot width rightward to the blind spot horizontal center ( $BS_{h1} = BS_{h0} + \max(BS\_W1, BS\_W2, BS\_W3)/4$ ),**  
**and  $1/4$  the maximum intermediate blind spot width leftward to the blind spot horizontal center ( $BS_{h2} = BS_{h0} - \max(BS\_W1, BS\_W2, BS\_W3)/4$ ).**

Similarly, these 3 paths estimate 3 intermediate blind spot heights (path BS\_h0 for BS\_H, BS\_h1 for BS\_H1, BS\_h2 for BS\_H2), in which BS\_H is the estimated blind spot height because path BS\_h0 passes the blind spot horizontal center. The blind spot

vertical center is estimated by the mean of the vertical midpoints of the 3 intermediate heights:

**BS\_v0 = mean(midpoint\_v(BS\_H), midpoint\_v(BS\_H1), midpoint\_v(BS\_H2))).**

Finally, the “2nd-horizontal-test” path will pass the estimated blind spot center ([BS\_h0, BS\_v0]) and estimates the blind spot width (BS\_W).

In “High” mode:

There will be 5 paths for the “1st-horizontal-test”, and they will be:

**S\_v0,**  
**S\_v0 + 7.5°/6,**  
**S\_v0 + 7.5° \* 2/6,**  
**S\_v0 - 7.5°/6,**  
**and S\_v0 - 7.5° \* 2/6.**

The blind spot horizontal center will be:

**BS\_h0 = mean(midpoint\_h(BS\_W1), ..., midpoint\_h(BS\_W5)).**

The 5 paths for the “1st-vertical-test” will be:

**BS\_h0,**  
**BS\_h0 + max(BS\_W1, ..., BS\_W5)/6,**  
**BS\_h0 + max(BS\_W1, ..., BS\_W5) \* 2/6,**  
**BS\_h0 - max(BS\_W1, ..., BS\_W5)/6,**  
**and BS\_h0 - max(BS\_W1, ..., BS\_W5) \* 2/6.**

The blind spot vertical center is

**BS\_v0 = mean(midpoint\_v(BS\_H), midpoint\_v(BS\_H1), ..., midpoint\_v(BS\_H4)).**

Finally, the “2nd-horizontal-test” path will pass the estimated blind spot center ([BS\_h0, BS\_v0]) and estimates the blind spot width (BS\_W).

“Horizontal\_Trials”: The path(s) for the “1st-horizontal-test” (see our paper for details) in the “Border points detection section” in pix, str “Default” or in the form of “[numeric, numeric, numeric, .....]”, and the number of “numeric” must be at least 1 (i.e., do not enter “[ ]”). Usually, you do not need to modify it and you can leave it “Default”, which means path(s) will be planned automatically based on “Mode”. If you change it to other values, the horizontal path(s) will be up to you. For example, if you modified the parameter to “[0, 100.0, -200.5, 50.2]”, then in the “1st-horizontal-test” in the “Border points detection section”, the cursor will be able to be moved along the horizontal straight lines at 4 heights in the order of 0 (central horizontal line) → 100 (100 pix higher than the central horizontal) → -200.5 (200.5 pix lower than central horizontal) → 50.2 (50.2 pix higher than central horizontal). Note that the “2nd-horizontal-test” path cannot be predetermined by you in any way. Note that decimal values may make no sense and Psychopy and/or the graphic card may round them to integers, so you should test to know the exact behavior for sure.

“Vertical\_Trials”: The path(s) for the “1st-vertical-test” in the “Border points detection section” in pix, str “Default” or in the form of “[numeric, numeric, numeric, .....]”, and the number of “numeric” must be at least 1 (i.e., do not enter “[ ]”). Usually, you do not need to modify it and you can leave it “Default”, which means the path(s) will be planned automatically based on “Mode”. If you change it to other values, the vertical path(s) will be up to you. For example, if you modified the parameter to “[0, 100.0, -200.5, 50.2]”, then in the “1st-vertical-test” in the “Border points detection section”, the cursor will be able to be moved along the vertical straight lines at 4 horizontal displacements in the order of 0 (central vertical line) → 100 (100 pix rightward to the central vertical) → -200.5 (200.5 pix leftward to the central vertical) → 50.2 (50.2 pix rightward to the central vertical). Note that decimal values may make no sense and Psychopy and/or the graphic card may round them to integers, so you should test to know the exact behavior for sure.

“Show\_Raw\_Data”: Display experiment progress and subject responses on the screen during an experiment, “True” or “False”. DO NOT modify these values in the file, but you can change the order of them. You can

select one option in the dialog box after running the script. Specifically, you can press “D” to turn on/off this feature during experiment.

“Staircase”: Run the “Staircase section”, “True” or “False”. DO NOT modify these values in the file, but you can change the order of them. You can select one option in the dialog box after running the script.

“Validation”: Specify the procedure to be used for the “Validation section”, where “Scaling” for the validation procedure, “Heat\_Map” for the visual sensitivity mapping procedure, and “No” for no “Validation section”. DO NOT modify these values in the file, but you can change the order of them. You can select one option in the dialog box after running the script.

“Eye\_Tracker”: Use an eye-tracker, “True” or “False”. DO NOT modify these values in the file, but you can change the order of them. You can select one option in the dialog box after running the script.

2. “eyetrackingconfig.json”. Note that only after you selected “True” for “Eye\_Tracker” in the dialog box that will pop up after running the script can you have this dialog box for eye-tracker settings and have these parameters working. Only Eyelink 1000 has been tested and used, so you may need to modify the relevant code blocks of this script and this configuration file if you want to use other devices.

```
1 {"Tolerance(deg)": "1.5", "Calibration_Type": ["HV9"]}
```

Parameter description:

“Tolerance(deg)”: The global eye movement tolerance in visual angle degree, str. When the distance of the recorded eye position from the fixation object is larger than the tolerance level, the fixation object will change color to provide feedback to the subject. If the subject issues a keyboard response whilst the fixation position is outside of the tolerance window, the response will be excluded and the experiment will not progress.

The default tolerance level of  $1.5^{\circ}$  is recommended to provide an optimal balance between accuracy and efficiency. During an extended block of trials, the eyetracker signal may drift slightly so that a fixation offset is

recorded even if the subject is fixating accurately. The program will then become less tolerant of true fixation offsets in the direction of the drift, and this may be frustrating for the subject. The default tolerance level is larger than typical true fixation offsets, and so it allows for a small amount of signal drift during a block. If a smaller tolerance window is desired (e.g. because it is essential that the blind spot map is very accurate), then the user is recommended to enable online drift correction for the eyetracker, and to drift correct regularly during a block.

“Calibration\_Type”: Eyelink-specific. Select the built-in eye-tracker calibration procedure, list of str. At present “HV9” is the only option, which involves 9 points: center, top, lower, left, right, and four corners. You should add the names of other calibration procedures into this parameter in this file if you want to use them. Review Psychopy documentation for all available procedures.

3. “logconfig.json”. The full list of items that are to be saved into data files. Theoretically, Psychopy automatically pickles “expInfo”, which is a dict object containing all experimental information, but this may not always work, and not all items are useful. So you should explicitly specify what you want to save. You cannot set data-saving parameters in the dialog boxes after running the script, but only by modifying this file. Note that some data outputs will depend on other experimental configurations but not solely on this file. For example, if you do not run the “Staircase section” (i.e., “Staircase” set to “False”), you will never get staircase data, even you included “Staircase” in the “logconfig.json” file before running the experiment.

```
1 ["ExpName", "Date", "Session", "Monitor_Name", "Monitor_Size(pix)", "FrameRate", "Distance(cm)",  
  "Participant", "Sex", "Age", "Test_Eye", "BG_Color", "Fix_Pos(pix)", "Fix_Size(pix)",  
  "Fix_Color", "Cursor_Color", "Cursor_Size(pix)", "Flickering(Hz)", "Mode", "H_Trials", "HT_N",  
  "V_Trials", "VT_N", "H_Clicks", "H_Points", "V_Clicks", "V_Points", "BS_Center",  
  "Raw_BS_Vertices", "BS_Vertices", "BS_width", "BS_height", "Staircase", "Validation"]
```

Parameter description:

Some parameters are already described in the above paragraphs, so they are not included here unless they have different meanings here.

“ExpName”: Name of the experiment. It should be “BS\_mapping”.

“Date”: Experiment date time in the form of “Y\_M\_D\_HrMin”.



“FrameRate”: The true framerate of the screen during an experiment.

“H\_Trials”: All horizontal paths for the 1st-horizontal-test and the 2nd-horizontal-test in the “Border points detection section”, all values are in pix.

“HT\_N”: The number of horizontal paths for the 1st-horizontal-test and the 2nd-horizontal-test in the “Border points detection section”.

“V\_Trials”: All vertical path(s) for the 1st-vertical-test in the “Border points detection section”, all values are in pix.

“VT\_N”: The number of vertical path(s) for the 1st-vertical-test in the “Border points detection section”.

“H\_Clicks”: All subject responses (cursor positions where mouse clicks were made) made in the 1st-horizontal-test and the 2nd-horizontal-test in the “Border points detection section”. Each position is a tuple in the form of “(horizontal, vertical)” in pix.

“H\_Points”: All border points estimated from clicks that were made in the 1st-horizontal-test and the 2nd-horizontal-test in the “Border points detection section”. Each point is a tuple in the form of “(horizontal, vertical)” in pix.

“V\_Clicks”: All subject responses (cursor positions where mouse clicks were made) made in the 1st-vertical-test in the “Border points detection section”. Each position is a tuple in the form of “(horizontal, vertical)” in pix.

“V\_Points”: All border points estimated from clicks that were made in the 1st-vertical-test in the “Border points detection section”. Each point is a tuple in the form of “(horizontal, vertical)” in pix.

“BS\_Center”: Estimate of the blind spot center (location) in the form of “(horizontal, vertical)” in pix.

“Raw\_BS\_Vertices”: All border points estimated from clicks that are made in the “Border points detection section”. It is “H\_Points” + “V\_Points”.

“BS\_Vertices”: All border points estimated from clicks that were made in the “Border points detection section” then adjusted by the “Staircase section”. If the “Staircase section” was not run, then it is the same as “Raw\_BS\_Vertices”. All values are in pix.

“BS\_Width”: The estimated blind spot width in pix.

“BS\_Height”: The estimated blind spot height in pix.

“Staircase”: Data for the “Staircase section”. When “Staircase” is “True” you will have these data saved in files. See the “Data” section of this manual for more details.

“Validation”: Data for the “Validation section”. When “Validation” is not “No”, you will have these data saved in files. See the “Data” section of this manual for more details.

“Eye\_Tracker”: Data for the subject’s eye movements during the whole experiment, together with the corresponding responses. When “Eye\_tracker” is “True”, you will have these data saved in files. See the “Data” section of this manual for more details.

4. “staircaseconfig.json”. This file has staircase settings for the “Staircase section”. But this is not the full set of all adjustable parameters. For example, the current tool estimates the threshold value based on the final one trial but not final n trials or n reversals (see our paper and the “Discussion” section of this manual). Adjustable parameters that are not included here cannot be configured here but only by modifying the relevant code block in the file “BS\_mapping.py”. Please review the Psychopy documentation for all adjustable parameters and modify the corresponding code block in the “BS\_mapping.py” file.

```
1 {"InitialVal(pix)": "50", "Step_Sizes(pix)": "30, 15, 7, 3", "Staircase_Trials": "20", "nUp":  
  "1", "nDown": "3", "nReversals": "4", "Staircase_Target_Display(s)": "0.4", "Staircase_Latency(s)": "0.4"}
```

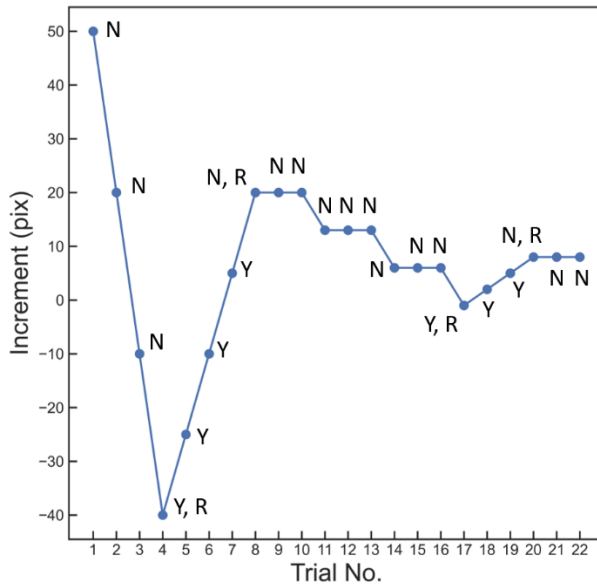
Parameter description:

“InitialVal(pix)”: Initial increment in pix, str. An increment is the distance between the raw border point (i.e., the border point estimated before the “Staircase section”, in the “Border points detection section”) and the current point in one trial of a staircase procedure. A positive increment indicates that the current point is closer to the estimated blind spot center.

“Step\_Sizes(pix)”: A set of step sizes for the absolute values of increments in pix, str in the format of “numeric1, numeric2, numeric3, .....”. The number of “numeric”s must be at least 1, and it must match “nReversals”. Before the first reversal, the increment will always be “numeric1” no matter the up-down rule, after the first reversal, it will become “numeric2”, then “numeric3”, ... based on the up-down rule (see below, “nUp” and “nDown”).

“Staircase\_Trials”: Number of trials for a staircase procedure, str of int. Note that this is the least number of trials for a staircase procedure to complete.

“nUp” and “nDown”: They together set the “up-down rule” for a staircase procedure, str of int. “nDown” is the number of “correct answers” (here “No I did not see it” is the correct answer) needed for the increment to go down by one step size, while “nUp” is the number of “incorrect answers” needed for the increment to go up by one step size. Before the first reversal, the “up-down” rule will be always “1-up-1-down”, but after the first reversal, the “up-down” rule will be what you specified here. For example, given the default settings (initial increment of 50, step sizes of [30, 15, 7, 3], and 1-up-3-down), a staircase trace may be:



Where “Y” = “Yes I saw it”, “N” = “No I did not see it”, and “R” is a reversal. Also see Figure 3 of our paper.

“nReversals”: Number of reversals needed to make before a staircase procedure completes, str of int. A reversal is defined as an opposite answer relative to the last answer. If the “Staircase\_Trials” has been reached but “nReversals” has not, the staircase procedure will continue until “nReversals” has been met.

“Staircase\_Target\_Display(s)”: Target object displaying time, in second, for a single trial of a staircase procedure, str.

“Staircase\_Latency(s)”: Time separation between the completion of the current trial and the initiation of the next trial, in second, str. If it is set to larger than 0, the subject’s response will end the current trial but the initiation of the next trial will be delayed by the latency you determined here.

5. “validationconfig.json”. The configuration file for the “Validation section”. The “Validation section” has two different procedures, so there are two sets of parameters.

```
1 [{"Scaling": {"Scaling_Range(portion)": "(0.6, 1.2)", "Scaling_Steps": "10", "Repeats": "10",
"Scaling_Target_Display(s)": "0.4", "Scaling_Latency(s)": "0.4"}, "Heat_Map": {"Grid_Width
(portion)": "1.1", "Grid_Height(portion)": "1.2", "Rows": "10", "Columns": "10", "Repeats":
"10", "Heat_Map_Target_Display(s)": "0.4", "Heat_Map_Latency(s)": "0.4"}}]
```

Parameter description for “Scaling”:

“Scaling\_Range(portion)”: Range of size-scaling coefficients for the estimated blind spot width and height, str in the form of “(smallest coef, largest coef)”.

“Scaling\_Steps”: Number of size-scaling coefficients, str of int. It must be at least 2. For example, given a “Scaling\_Range(portion)” of (0.6, 1.2) and a “Scaling\_Steps” of 10, then you will have 10 size-scaling coefficients ranging from 0.6 to 1.2 (including 0.6 and 1.2).

“Repeats”: Number of trials to do for one single size-scaling coefficient, str of int.

“Scaling\_Target\_Display(s)”: Target object displaying time, in second, for a single trial in a “Scaling” validation procedure, str.

“Scaling\_Latency(s)”: Time separation between the completion of the current trial and the initiation of the next trial, in second, str.

Parameter description for “Heat\_Map”:

“Grid\_Width(portion)”: Scaling coefficient for the estimated blind spot width, str.

“Grid\_Height(portion)”: Scaling coefficient for the estimated blind spot height, str.

“Rows” and “Columns”: They determine the numbers of rows and columns of the grided monocular visual field that is to be tested, str of int.

“Repeats”: Number of trials to do at a single cell of the grided monocular visual field, str of int.

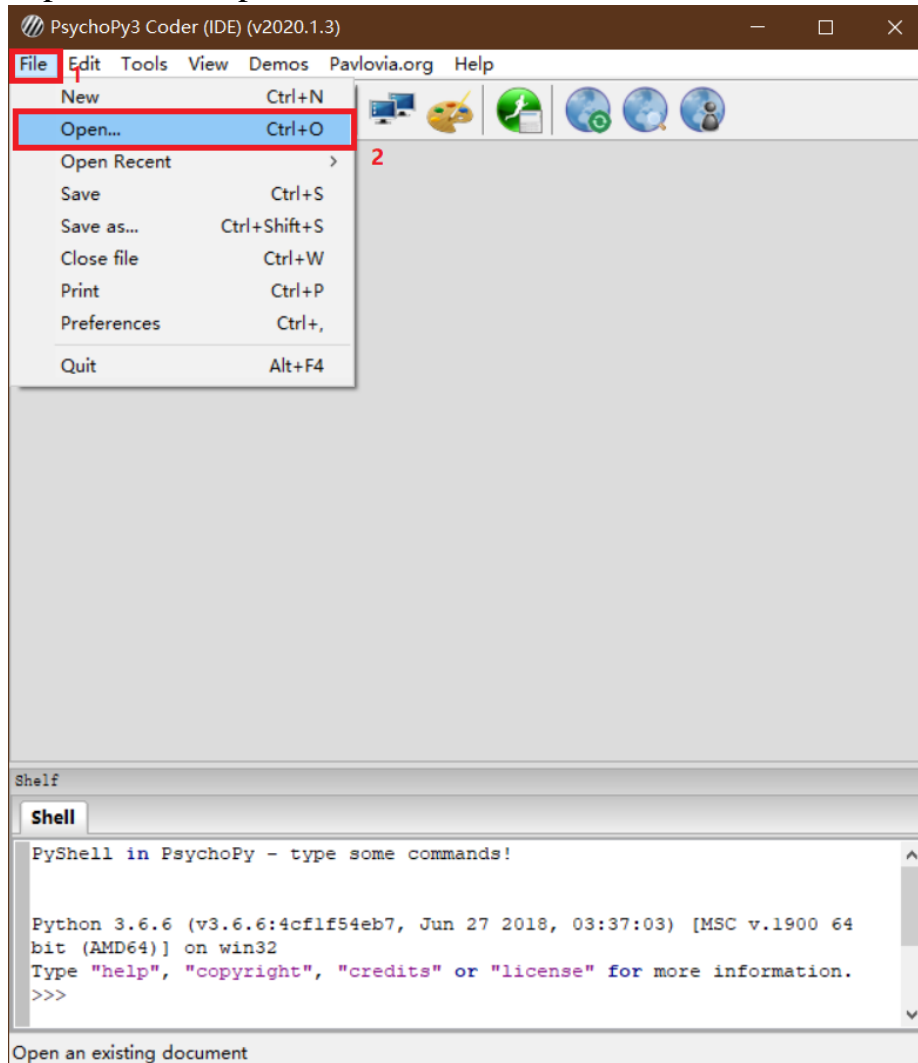
“Heat\_Map\_Target\_Display(s)”: Target object displaying time, in second, for a single trial in the “Heat\_Map” visual sensitivity mapping procedure, str.

“Heat\_Map\_Latency(s)”: Time separation between the completion of the current trial and the initiation of the next trial, in second, str.

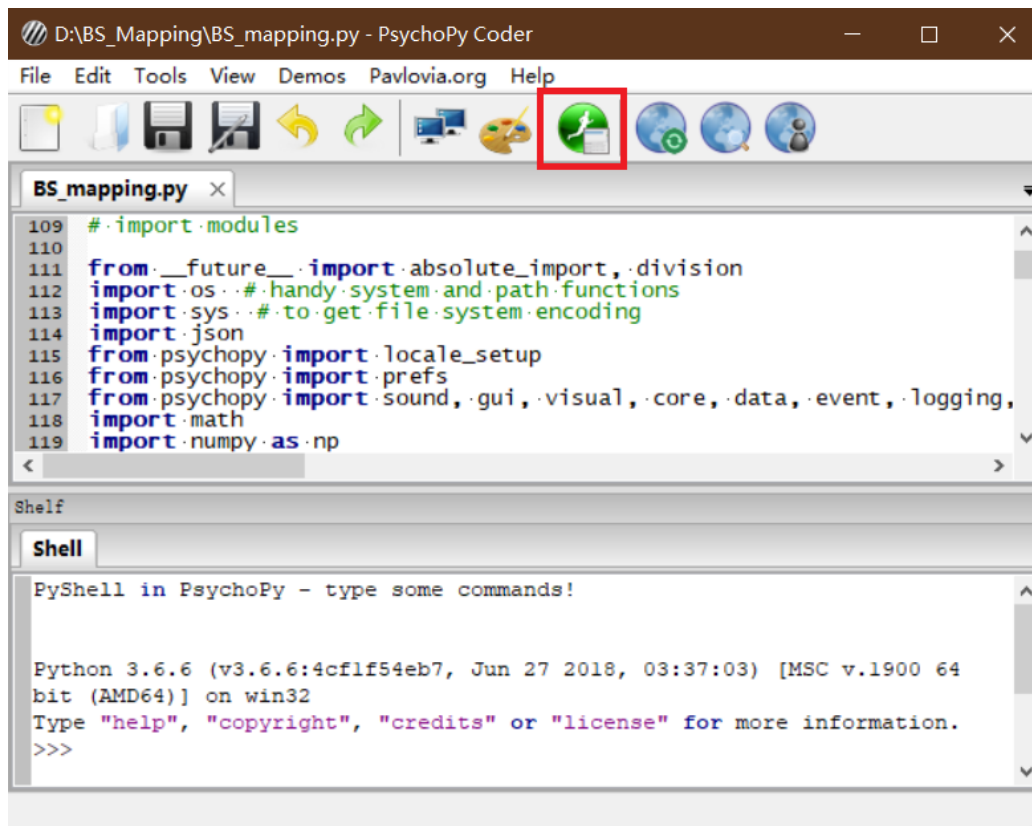
## 4. During Experiment

### 4.1. Run experiment script

1. Launch Psychopy and open “BS\_mapping.py”. Launch Psychopy Coder view and through “File” → “Open” open the “BS\_mapping.py” experiment script.



2. Run the script. Click the “Run current script” icon.



## 4.2. Configure experiment in dialog boxes

After clicking the “Run current script” icon, a series of dialog boxes will pop up. You should confirm your experiment settings there.

1. The “BS\_mapping” dialog box. This dialog box is made from the file “expconfig.json”, and you can confirm your experiment settings here. The default parameter values will be the values you gave in the configuration file, except parameters having a drop-down menu. Parameters with a drop-down menu will use the first value as the default, but you can select other values in the menu. For example, if the “Test\_Eye” in “expconfig.json” was “[‘RIGHT’, ‘LEFT’]”, where “RIGHT” was the first value in the list, then “RIGHT” would be the default value for “Test\_Eye” in the “BS\_mapping” dialog box. After finalizing your settings, click the “OK” button to confirm your settings and go to the next step.

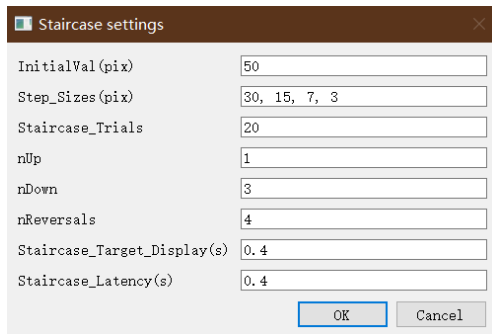


Parameter	Value
Session	test
Participant	test
Sex	male
Age	24
Monitor_Name	testMonitor
Monitor_Size(pixel)	1280, 960
Monitor_Size(cm)	40, 30
Distance(cm)	57
BG_Color	128, 128, 128
Fix_Color	white
Fix_Size(pixel)	40
Fix_Pos(pixel)	300, 0
Cursor_Color	white
Cursor_Size(pixel)	12
Flickering(Hz)	20
Test_Eye	RIGHT
Mode	Low
Horizontal_Trials	Default
Vertical_Trials	Default
Show_Raw_Data	False
Staircase	True
Validation	Scaling
Eye_Tracker	True

Buttons: OK, Cancel

Please reference the “Preparation before Experiment” section of this manual for parameter descriptions.

2. “Staircase settings” dialog box. This dialog box is made from the file “staircaseconfig.json”, and you can confirm your experiment settings here. The default parameter values will be the values you gave in the configuration file. NOTE that only if you selected “True” for “Staircase” in the “BS\_mapping” dialog box will you have this dialog box and run the “Staircase section”. After finalizing your settings, click the “OK” button to confirm your settings and go to the next step.



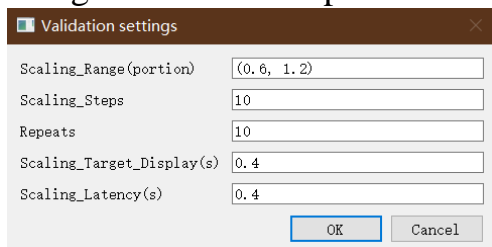
Staircase settings

InitialVal (pix)	50
Step_Sizes (pix)	30, 15, 7, 3
Staircase_Trials	20
nUp	1
nDown	3
nReversals	4
Staircase_Target_Display(s)	0.4
Staircase_Latency(s)	0.4

OK Cancel

Please reference the “Preparation before Experiment” section of this manual for parameter descriptions.

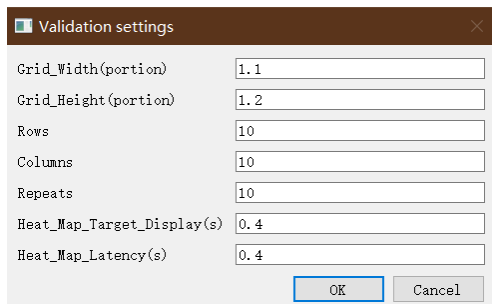
3. “Validation settings” dialog box. This dialog box is made from the file “validationconfig.json”, and you can confirm your experiment settings here. The default parameter values will be the values you gave in the configuration file. NOTE that if you selected “No” for “Validation” in the “BS\_mapping” dialog box you would not have this dialog box nor run the “Validation section”. The contents in this “Validation settings” dialog box depend on your option for the “Validation” in the “BS\_mapping” dialog box (i.e., “Scaling” or “Heat\_Map”). After finalizing your settings, click the “OK” button to confirm your settings and go to the next step.



Validation settings

Scaling_Range (portion)	(0.6, 1.2)
Scaling_Steps	10
Repeats	10
Scaling_Target_Display(s)	0.4
Scaling_Latency(s)	0.4

OK Cancel



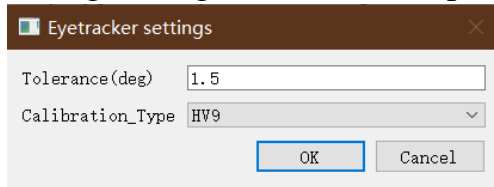
Validation settings

Grid_Width (portion)	1.1
Grid_Height (portion)	1.2
Rows	10
Columns	10
Repeats	10
Heat_Map_Target_Display(s)	0.4
Heat_Map_Latency(s)	0.4

OK Cancel

Please reference the “Preparation before Experiment” section of this manual for parameter descriptions.

4. “Eyetracker settings” dialog box. This dialog box is made from the file “eyetrackingconfig.json”, and you can confirm your experiment settings here. The default parameter values will be the values you gave in the configuration file. NOTE that only if you selected “True” for “Eye\_Tracker” in the “BS\_mapping” dialog box will you have this dialog box, and that you can also change settings on the eye-tracker computer. After finalizing your settings, click the “OK” button to confirm your settings and go to the next step.



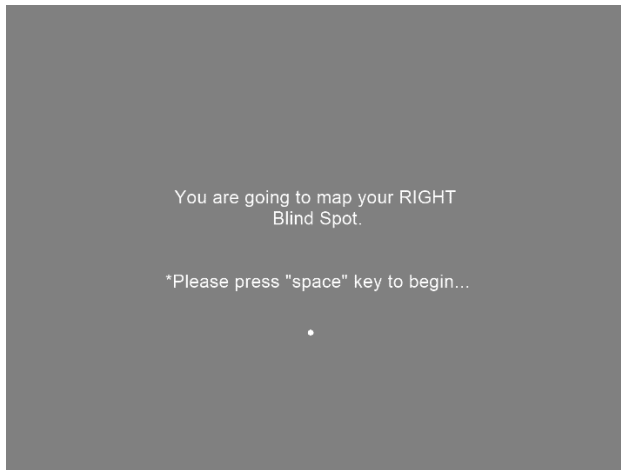
Please reference the “Preparation before Experiment” section of this manual for parameter descriptions.

### 4.3. Lead-in section

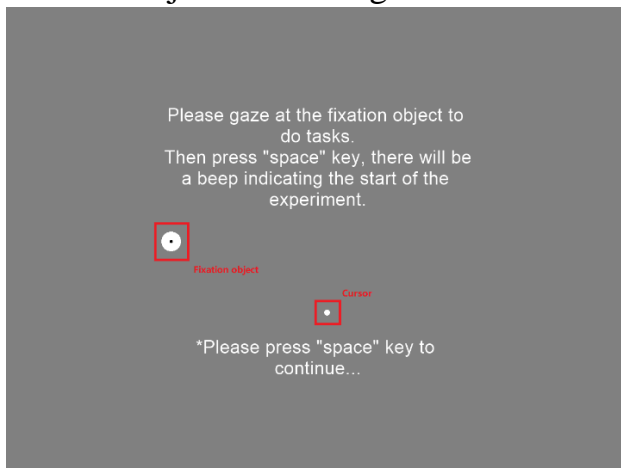
If an eye-tracker is used and you have specified “True” for “Eye\_Tracker” in the “BS\_mapping” dialog box, the eye-tracker will monitor the subject’s eye movements (eye-tracking-on). In this case, an eye-tracker calibration procedure will begin before other experimental sections.

After eye-tracker calibration and before the experimental sections, there will be two lead-in slides for the subject. During this lead-in section, you can press “D” to turn on/off data-displaying on the screen that shows experiment progress and subject responses. During the lead-in section, the cursor will not flick.

The first slide will illustrate the experiment's aim and which eye is the test eye.



The second slide will instruct the subject to gaze at the fixation object during the experiment. This fixation object will be on the left side if the right eye is to be tested, or on the right side if the left eye is to be tested. In eye-tracking-on mode, when the eye movement is smaller than the tolerance, the fixation object colors will be what you set in the “BS\_mapping” dialog box; when the eye movement is larger than the tolerance, the outer annulus of the fixation object will change to red and the inner disk will change to blue.

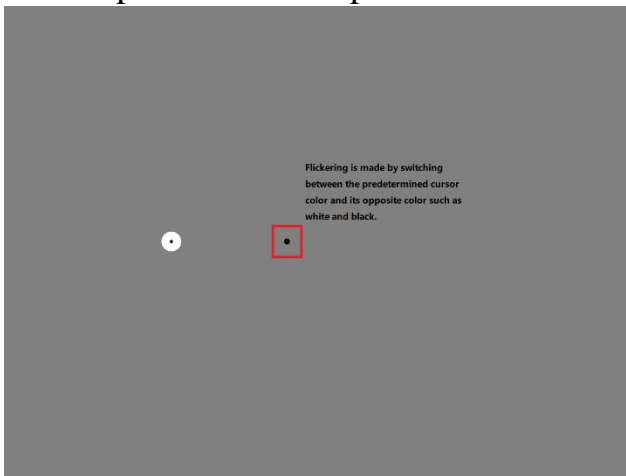


From the second lead-in slide onward, the subject can leave for a break at any time during the experiment. After returning to the experiment, you or the subject should press “E” to re-calibrate the eye-tracker.

#### **4.4. Border points detection section**

This is the first experimental section and it is mandatory. During this section, the subject will have to gaze at the fixation object and simultaneously move the cursor (flickering or not, depending on your setting) along straight horizontal or vertical paths that cross the blind spot.

Note that the subject will not need to move the cursor by perfectly straightly moving the computer mouse, because only the movement along the desired direction will be used to move the cursor. During moving the cursor, the subject will click the mouse when the subject perceived that the cursor disappeared or reappeared. You must monitor the performance of the subject and give proper instructions during this section. After the subject has done a sufficient number of successful trials (moving and clicking, the number depends on your needs) along one path, you should press “Space” so that the experiment will go to the next path or the next experimental section, indicated by a 0.3s beep. Please reference our paper for the principles of this border-point-detection procedure.



Paths will be given based on “Mode” or up to you. There will be 2 horizontal and 1 vertical paths for “Low” mode, 4 horizontal and 3 vertical paths for “Medium” mode, and 6 horizontal and 5 vertical paths for “High” mode. You can explicitly determine your desired paths in “Horizontal\_Trials” and “Vertical\_Trials” in the dialog box “BS\_mapping” or in the file “expconfig.json”. Please reference the “Preparation before Experiment” section of this manual for more details about how to modify these two parameters.

In eye-tracking-on mode, only when the eye movement of the subject is within the tolerance will the mouse-click responses be used to map the blind spot (the responses will be logged for post-experiment analysis, but not used for blind spot mapping). A broken fixation will be indicated by the color changes of the fixation object. A successful click will be indicated by a brief beep. NOTE that among all in-situ repeated clicks, only the first one will be

used (i.e., if the subject did not move the cursor, and during this no-movement period clicked the mouse many times, all clicks were logged for post-experiment analysis but only the first click could be recorded as “successful” and used for blind spot mapping), but if the clicks made at the same position were separated by cursor movement(s), they could be used for blind spot mapping.

In eye-tracking-on mode, the subject can leave for a break at any time during the experiment. The current progression will be saved. After returning to the experiment, you or the subject should press “E” to re-calibrate the eye-tracker.

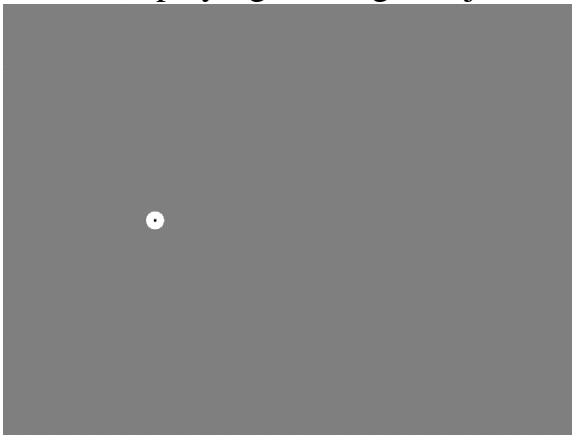
During this “Border points detection section”, you can press “D” to turn on/off data-displaying on the screen.

#### **4.5. Staircase section**

This experimental section is optional. If you selected “True” for “Staircase” in the dialog box “BS\_mapping” and set parameters for the staircase procedure in the dialog box “Staircase settings”, then the “Staircase section” will run, following the “Border points detection section”.

The “Staircase section” is made up of many trials. All trials will be controlled completely by the subject. Each trial has the same three phases:

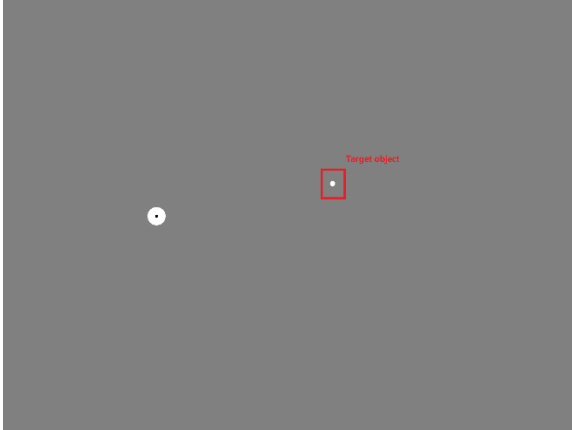
1. Before displaying the target object.



In this phase, only the fixation object will be on the screen. The program will wait for the subject to trigger the onset of target-object-displaying.

When the subject is ready to see the target object, the subject should gaze at the fixation object and press “Space”. In eye-tracking-on mode, only when the eye movement is smaller than the tolerance can the keypress be successful.

2. Displaying the target object.



After the subject pressed “Space”, the target object will be displayed together with the fixation object for a predetermined period. The target object will be the same as the cursor used in the “Border points detection section”.

3. Response.



After the target-object-displaying period, the whole screen will be cleared, and at the same time the program will display a computer-synthesized man’s voice saying “Did you see it?” to indicate the subject to respond by pressing the “left arrow” for “No” or the “right arrow” for “Yes”. In eye-tracking-on mode, note that during this phase, although the fixation object will not be shown, the eye-tracker will still monitor the

eye movement and the program will still judge if the subject's eye movement is larger than the tolerance or not, but this will not influence anything. However, in eye-tracking-on mode, if the subject's eye movement was ever larger than the tolerance at any moment during the "Displaying target object" phase, the subject's response for this trial would be logged only for post-experiment analysis but not used for blind spot mapping. The current trial would re-run until successful.

These three phases will cycle for many trials until the completion of the "Staircase section", which will be indicated by a 0.3s beep. Please reference our paper for details about the principles of the staircase procedure.

In eye-tracking-on mode, the subject can leave for a break at any time during the experiment. The current progression will be saved. After returning to the experiment, you or the subject should press "E" to re-calibrate the eye-tracker.

During this "Staircase section", you can press "D" to turn on/off data-displaying on the screen.

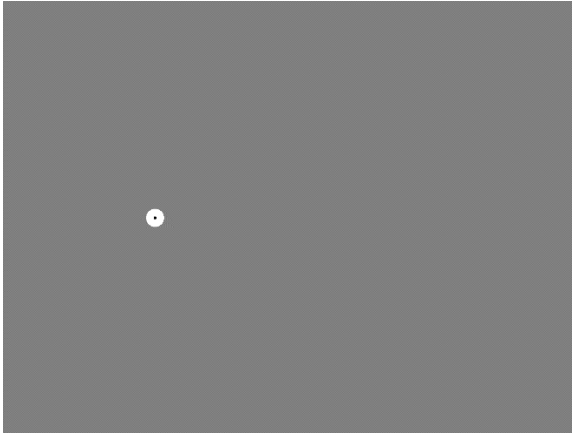
#### **4.6. Validation section**

This experimental section is optional. If you selected "Scaling" or "Heat\_Map" for "Validation" in the dialog box "BS\_mapping" and set parameters for the corresponding procedure in the dialog box "Validation settings", the "Validation section" would run following the "Staircase section" (if the "Staircase section" has run) or the "Border points detection section" (if the "Staircase section" has not run).

The "Validation section" is made up of many trials. All trials will be controlled completely by the subject. Each trial has the same three phases, but they will depend on the procedure chosen. The three phases for the "Scaling" procedure are:

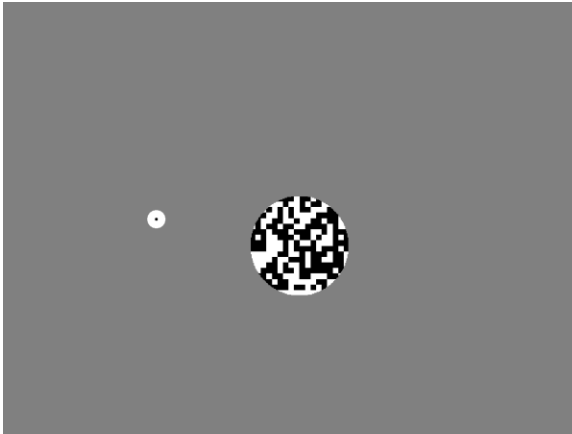
1. Before displaying the target object.





In this phase, only the fixation object will be on the screen. The program will wait for the subject to trigger the onset of target-object-displaying. When the subject is ready to see the target object, the subject should gaze at the fixation object and press “Space”. In eye-tracking-on mode, only when the eye movement is smaller than the tolerance can the keypress be successful.

## 2. Displaying the target object.



After the subject pressed “Space”, the target object will be displayed together with the fixation object for a predetermined period. The target object will be an ellipse whose minor and major axes are determined by the estimated blind spot size (width and height) multiplied by one of the predetermined scaling coefficients. The target object will be filled with randomly distributed squares in the same colors of the cursor used in the “Border points detection section” (i.e., white and its opposite color black, if the cursor color is white and if the flickering frequency is larger than 0 Hz), so the average luminance of this target object equals the luminance of the cursor.

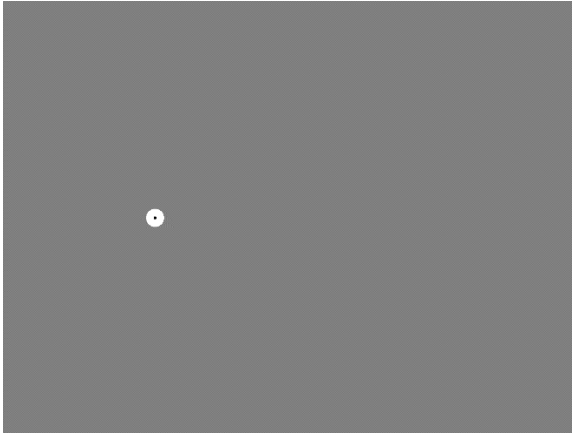
### 3. Response.



After the target-object-displaying period, the whole screen will be cleared, and at the same time the program will display a computer-synthesized man's voice saying "Did you see it?" to indicate the subject to respond by pressing the "left arrow" for "No" or the "right arrow" for "Yes". In eye-tracking-on mode, during this phase, although the fixation object will not be shown, the eye-tracker will still monitor the eye movement and the program will still judge if the subject's eye movement is larger than the tolerance or not, but this will not influence anything. However, in eye-tracking-on mode, if the subject's eye movement was ever larger than the tolerance at any moment during the "Displaying target object" phase, the subject's response for this trial would be logged only for post-experiment analysis but not used for blind spot mapping. The current trial would re-run until successful.

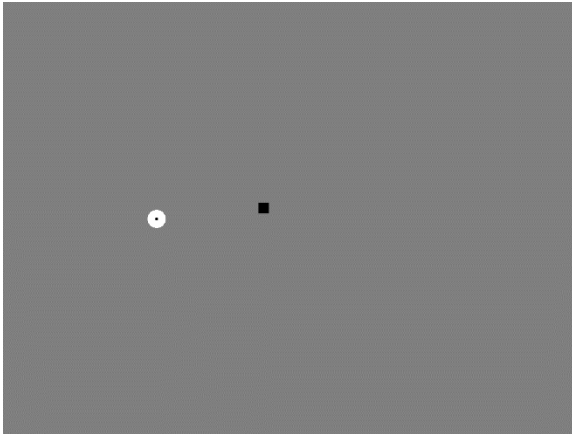
The three phases for the "Heat\_Map" procedure are:

1. Before displaying the target object.



In this phase, only the fixation object will be on the screen. The program will wait for the subject to trigger the onset of target-object-displaying. When the subject is ready to see the target object, the subject should gaze at the fixation object and press “Space”. In eye-tracking-on mode, only when the eye movement is smaller than the tolerance can the keypress be successful.

## 2. Displaying the target object.



After the subject pressed “Space”, the target object will be displayed together with the fixation object for a predetermined period. The target object will be a square whose width and height are determined by the estimated blind spot size (width and height) multiplied by the corresponding predetermined scaling coefficient then divided by the number of columns and rows. The target object will have the same color as the cursor used in the “Border points detection section” (i.e., white if the cursor color is white). The flickering frequency of the target object is also the same as the flickering frequency of the cursor.

### 3. Response.



After the target-object-displaying period, the whole screen will be cleared, and at the same time the program will display a computer-synthesized man's voice saying "Did you see it?" to indicate the subject to respond by pressing the "left arrow" for "No" or the "right arrow" for "Yes". In eye-tracking-on mode, during this phase, although the fixation object will not be shown, the eye-tracker will still monitor the eye movement and the program will still judge if the subject's eye movement is larger than the tolerance or not, but this will not influence anything. However, in eye-tracking-on mode, if the subject's eye movement was ever larger than the tolerance at any moment during the "Displaying target object" phase, the subject's response for this trial would be logged only for post-experiment analysis but not used for blind spot mapping. The current trial would re-run until successful.

These three phases will cycle for many trials until the completion of the "Validation section", indicated by a 0.3s beep. Please reference our paper for details about the principles of the two procedures.

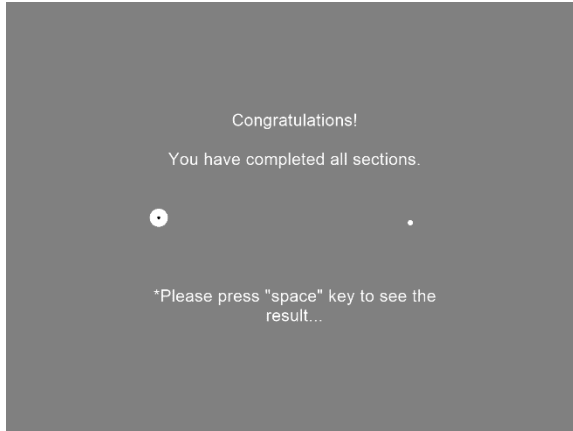
In eye-tracking-on mode, the subject can leave for a break at any time during the experiment. The current progression will be saved. After returning to the experiment, you or the subject should press "E" to re-calibrate the eye-tracker.

During this "Validation section", you can press "D" to turn on/off data-displaying on the screen.

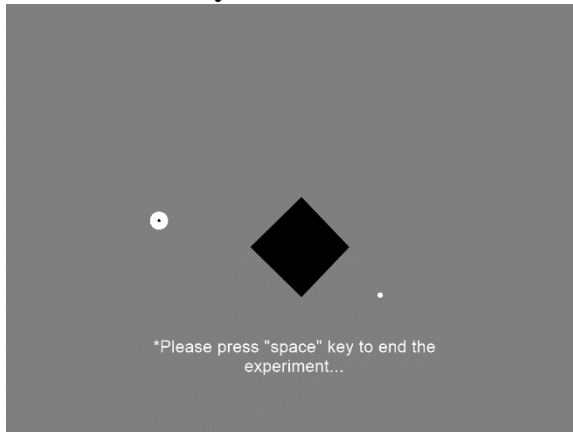
#### 4.7. Completion section

After all experimental section(s) has been completed, the experiment is completed. Completion of the experiment will be indicated by two slides:

The first one will indicate the end of the experiment.



The second one will display the estimated blind spot border points and the area formed by them.



Note that in “Low” mode, although 6 points were estimated by the “Border points detection section”, only 4 cardinal points will be displayed here, because only these 4 cardinal points were adjusted by the “Staircase section”. In “Medium”, “High”, and user-defined (i.e., where the paths in the “Border points detection section” are defined by you) modes, all estimated border points will be displayed here, because all of them will be adjusted by the “Staircase section”. Please reference our paper for the reasons to do so.

## 5. Data

The file “logconfig.json” specifies a part of the data that will be outputted into files. In addition, some data will be saved into files automatically, depending on what experimental sections you will run and whether or not you will use an eye-tracker. For example, if you did not use an eye-tracker during an experiment, you would have no eye-tracking data for that experiment.

Data files will be saved to the path “cwd/data/files”, where “cwd” is the current working document folder of the script “BS\_mapping.py” and “files” is the name(s) of the data file(s).

For a full-section experiment using an eye-tracker, you should have at least 8 JSON, 8 CSV, and several (depending on the number of border points being adjusted by the “Staircase section”) PSYDAT files. Usually, you should also have at least 1 additional PSYDAT file which contains all experimental information excluding eye-tracking data, but sometimes this file can be missing or duplicated, so we do not describe it here.

### 5.1. JSON and CSV files

The JSON and CSV files have the same contents and they only differ in file format. So our descriptions for them are based on the CSV files only. In general, you should have 8 CSV files having names in the form of “Participant\_Session\_expName\_Y\_M\_D\_HrMinExtrainfo.csv”, for example, “p1\_s1\_BS\_mapping\_2021\_Mar\_01\_1201Staircase.csv”. So you can differentiate them based on “Extrainfo”:

1. “Extrainfo” is blank, e.g.,  
“p1\_s1\_BS\_mapping\_2021\_Mar\_01\_1201.csv”. This is the main data file that stores the data you specified in the file “logconfig.json”. Please

reference the “Preparation before Experiment” section of this manual for parameter descriptions.

2. “Extrainfo” is “Staircase”, e.g.,  
 “p1\_sl\_BS\_mapping\_2021\_Mar\_01\_1201Staircase.csv”. This file contains the data for the “Staircase section”.

	A	B	C	D	E
1		Dot_(183.6	Dot_(82.4	Dot_(183.6	Dot_(287.6
2	0	(1, 50)	(1, 50)	(1, 50)	(1, 50)
3	1	(2, 20)	(2, 20)	(2, 20)	(2, 20)
4	2	(3, -10)	(3, -10)	(3, -10)	(3, -10)
5	3	(4, 5)	(4, 5)	(4, 5)	(4, 5)
6	4	(5, 20)	(5, 5)	(5, 5)	(5, 5)
7	5	(6, 20)	(6, 5)	(6, 5)	(6, 20)
8	6	(7, 20)	(7, -2)	(7, 20)	(7, 20)
9	7	(8, 13)	(8, 1)	(8, 20)	(8, 20)
10	8	(9, 13)	(9, 1)	(9, 20)	(9, 13)
11	9	(10, 16)	(10, 4)	(10, 13)	(10, 13)
12	10	(11, 16)	(11, 4)	(11, 13)	(11, 13)
13	11	(12, 16)	(12, 4)	(12, 13)	(12, 6)
14	12	(13, 13)	(13, 1)	(13, 6)	(13, 9)
15	13	(14, 13)	(14, 4)	(14, 6)	(14, 9)
16	14	(15, 13)	(15, 7)	(15, 6)	(15, 9)
17	15	(16, 10)	(16, 7)	(16, -1)	(16, 6)
18	16	(17, 10)	(17, 7)	(17, -1)	(17, 6)
19	17	(18, 10)	(18, 4)	(18, -1)	(18, 6)
20	18	(19, 13)	(19, 4)	(19, -8)	(19, 3)
21	19	(20, 13)	(20, 4)	(20, -5)	(20, 6)
22	20			(21, -2)	
23	21			(22, 1)	
24	22			(23, 1)	
25	23			(24, 1)	

This example datasheet is opened with Office Excel. It has 5 columns, reflecting that 4 border points were adjusted by the staircase procedure. These 4 columns are labeled by names in the form of “Dot\_(horizontal, vertical)” in pix (in the current version, column names are not headed with “Dot\_” anymore), for the estimated border points before the staircase procedure. This datasheet has several rows, with non-blank cells containing values expressed as “(No., Increment)”, where “No.” indicates the numbering of the trial and “Increment” indicates the current increment for that trial. For example, cell B21 has the value pair of “(20, 13)”, and column B is labeled with “Dot\_(183.6, 63.5)”, which means that the “Border points detection section” has estimated one blind spot border point at the position of (183.6 pix, 63.5 pix) relative to the screen (not relative to the fixation object) with the screen center being (0 pix, 0 pix), and that this estimated border point has been adjusted by the “Staircase section” in 20 trials, and the final increment was 13 pix. By default, the final position of the border point is calculated by the final

increment only, so it can be inferred that the original point was moved by the staircase procedure toward the estimated blind spot center by 13 pix.

3. “Extrainfo” is “Validation”, e.g., “p1\_sl\_BS\_mapping\_2021\_Mar\_01\_1201Validation.csv”. This file contains the data for the “Validation section”. Given that this section has two different options of procedures (i.e., “Scaling” and “Heat\_Map”), the data are different depending on the procedure conducted.

	A	B	C	D	E	F	G	H	I	J	K
1		0.6	0.67	0.73	0.8	0.87	0.93	1	1.07	1.13	1.2
2	visible	0	0	0	0	0	0.3	1	1	1	1

This is an example datasheet for a “Validation section” that used a “Scaling” procedure. It has 10 columns that are labeled by numbers from 0.6 to 1.2. This reflects that this “Scaling” procedure used 10 scaling coefficients uniformly distributed within the range from 0.6 to 1.2. The second row shows the visibility scores (likelihood of responding “Yes I saw it”) for each proportional size of the target object. So this datasheet suggests that in this “Scaling” procedure, the 1.00 target object (an elliptical object having its width and height set to 1.00 the estimated blind spot width by 1.00 the estimated blind spot height) was 100% visible, the 0.93 target object (0.93 width by 0.93 height) was 30% visible, while the 0.87 target object (0.87 width by 0.87 height) was 0% visible.

If a “Heat\_Map” procedure was performed, the datasheet would be a matrix of visibility scores.

	A	B	C	D	E	F	G	H	I	J	K
1		119.75	142.83	165.91	188.99	212.08	235.16	258.24	281.32	304.4	327.48
2	112.88	1	1	1	1	1	1	1	1	1	0.9
3	83.74	1	1	1	0.9	0.4	1	1	1	1	1
4	54.6	1	1	0.2	0	0	0	0.1	1	1	1
5	25.46	1	0.6	0	0	0	0	0	0.1	0.9	1
6	-3.68	0.7	0	0	0	0	0	0	0	0.2	1
7	-32.82	0.5	0	0	0	0	0	0	0	0	0.9
8	-61.96	0.9	0	0	0	0	0	0.1	0	0	0.9
9	-91.1	0.9	0.7	0	0	0	0	0	0	0	1
10	-120.24	1	1	1	0.3	0	0	0	0.3	0.6	1
11	-149.38	1	1	1	1	1	1	1	1	1	1

This is an example data matrix for a “Validation section” that used a “Heat\_Map” procedure. It has 10 columns and 10 rows. The column labels are numbers uniformly distributed from 119.75 pix to 327.58 pix, while row indices are from 112.88 pix to -149.38 pix. These labels and



indices are the positions of cell centers relative to the screen (not relative to the fixation object), with the screen center being (0 pix, 0 pix), so you can infer that each cell has the same width of 23.08 pix and the same height of 29.14 pix. In the data matrix, values are the visibility scores for the corresponding cells (i.e., likelihood of responding “Yes I saw it”), so 1 indicates 100% visible while 0 indicates 0% visible.

4. “Extrainfo” is “Section/PHASE\_tracking”, where “Section/PHASE” can be “Staircase” or “Validation” for the “Staircase section” or the “Validation section”, respectively, or it can be “HT”, “VT” or “HD” for the “1st-horizontal-test”, “1st-vertical-test”, or “2nd-horizontal-test” phases of the “Border points detection section”, respectively. These 5 files contain eye-tracking data for different experimental sections or for different phases of the “Border points detection section”.

Datasheets for the “1st-horizontal-test”, “1st-vertical-test”, and “2nd-horizontal-test” phases of the “Border points detection section” (i.e., file name labeled with “HT\_tracking”, “VT\_tracking”, and “HD\_tracking”) are structured in the same way.

	A	B	C	D	E	F	G
1		frameN	routine_time	global_time	gaze	good_fixation	click
2	0	0	0.011683722	9.15877217	[-302.70	TRUE	FALSE
3	1	1	0.023385533	9.17047518	[-301.79	TRUE	FALSE
4	2	2	0.035095756	9.1821848	[-304.0,	TRUE	FALSE
5	3	3	0.046859161	9.19394821	[-303.0,	TRUE	FALSE
6	4	4	0.058652011	9.20574106	[-302.20	TRUE	FALSE
7	5	5	0.070444862	9.21753391	[-301.70	TRUE	FALSE
8	6	6	0.082137358	9.2292261	[-302.20	TRUE	FALSE
9	7	7	0.093916988	9.24100603	[-303.5,	TRUE	FALSE
10	8	8	0.105714946	9.25280369	[-303.39	TRUE	FALSE
11	9	9	0.117446201	9.26453525	[-302.29	TRUE	FALSE
12	10	10	0.129172349	9.27626139	[-301.70	TRUE	FALSE
13	11	11	0.140934252	9.2880233	[-302.10	TRUE	FALSE
14	12	12	0.152707873	9.29979692	[-302.79	TRUE	FALSE
15	13	13	0.164455353	9.3115444	[-304.10	TRUE	FALSE
16	14	14	0.176244298	9.32333304	[-305.29	TRUE	FALSE
17	15	15	0.187870150	9.3350680	[-304.20	TRUE	FALSE

This is part of the example datasheet for the eye-tracking data for the “1st-horizontal-test phase” (labeled by “HT\_tracking”) in a “Border points detection section”, where “frameN” indicates the frame numbering

(relative to the current routine, here the “1st-horizontal-test phase”); “routine\_time” indicates the local time in s (relative to the current routine, here the “1st-horizontal-test phase”); “global\_time” indicates the global time in s (relative to the whole experiment); “gaze” indicates the gazing position (expressed as “[horizontal, vertical]” in pix) relative to the screen with the screen center being [0 pix, 0 pix]; “good\_fixation” indicates whether or not the fixation was broken (“TRUE” for “eye movement smaller than the tolerance” and “FALSE” for “eye movement larger than the tolerance”); and “click” indicates whether or not the subject clicked the computer mouse (“TRUE” for “clicked” and “FALSE” for “did not click”).

Datasheets for the “Staircase section” and “Validation section” (i.e., file name labeled with “Staircase\_tracking”, “Validation\_tracking”) are structured in the same way.

	A	B	C	D	E	F	G	H	I
1		frameN	routine_time	global_time	gaze	good_fixation	button	stimulus_display	recorded_response
2	0	0	1.131283999	179.570411	[-295.5,	TRUE	None	FALSE	None
3	1	1	1.144552721	179.583693	[-294.70	TRUE	None	FALSE	None
4	2	2	1.155384092	179.594524	[-294.29	TRUE	None	FALSE	None
5	3	3	1.16710393	179.60623	[-294.89	TRUE	None	FALSE	None
6	4	4	1.178867635	179.617994	[-294.79	TRUE	None	FALSE	None
7	5	5	1.190731996	179.629858	[-202.79	FALSE	None	FALSE	None
8	6	6	1.202465955	179.641607	[-113.59	FALSE	None	FALSE	None
9	7	7	1.214147635	179.653274	[-24.0, -	FALSE	None	FALSE	None
10	8	8	1.225936279	179.665062	[-37.900	FALSE	None	FALSE	None
11	9	9	1.237699984	179.676826	[-37.599	FALSE	None	FALSE	None
12	10	10	1.249428836	179.688555	[-37.599	FALSE	None	FALSE	None
13	11	11	1.261190438	179.700317	[-37.5, -	FALSE	None	FALSE	None
14	12	12	1.272964059	179.71209	[-36.299	FALSE	None	FALSE	None
15	13	13	1.284712741	179.723839	[-35.0, -	FALSE	None	FALSE	None
16	14	14	1.29647224	179.735611	[-33.900	FALSE	None	FALSE	None
17	15	15	1.308233242	179.747359	[-34.900	FALSE	None	FALSE	None
18	16	16	1.310000039	179.759117	[-37.0,	FALSE	None	FALSE	None

This is part of the example datasheet for the eye-tracking data for the “Staircase section” (labeled by “Staircase\_tracking”), where “frameN” indicates the frame numbering (relative to the current routine, here the “Staircase section”); “routine\_time” indicates the local time in s (relative to the current routine, here the “Staircase section”); “global\_time” indicates the global time in s (relative to the whole experiment); “gaze” indicates the gazing position (expressed as “[horizontal, vertical]” in pix) relative to the screen with the screen center being [0 pix, 0 pix]; “good\_fixation” indicates whether or not the fixation was broken (“TRUE” for “eye movement smaller than the tolerance” and “FALSE”

for “eye movement larger than the tolerance”); “button” indicates whether or not the subject clicked one of functional keys (“left” for “left arrow”, “right” for “right arrow”, “space” for “Space bar”, “e” for “E”, “d” for “D”, and “None” for not any of above); “stimulus\_display” indicates whether or not the target object was being displayed; and “recorded\_response” for the result in response to the subject’s button (0, 1, or “None”).

The value of “recorded\_response” is calculated based on the values of “button”, “good\_gaze”, and “stimulus\_display”. Any functional key other than “left” and “right” will result in “None” for “recorded\_response”. Any “left” or “right” key before the target-object-displaying period (i.e., before a successful “button” == “space” has been made) or during this period (i.e., “stimulus\_display” == “TRUE”) will result in “None” for “recorded\_response”. Any bad fixation (i.e., “good\_gaze” == “FALSE”) during the target-object-displaying period will lead to unsuccessful button responses (i.e., “recorded\_response” == “None”).

Although not shown on the example figure, some eye-tracking datasheet have another column “condition”. For “Staircase\_tracking”, this item is the border point being tested in that trial, in the form of“(horizontal, vertical)” in pix. For “Validation\_tracking”, this item is the scaling coefficient used in that trial when the “Scaling” procedure was used, or the cell position (in the form of“(horizontal, vertical)” in pix), which was also the target object position, being tested in that frame when the “Heat\_Map” procedure was used.

Please pay a lot of attention to an important difference between the data in “Staircase\_tracking” and “Validation\_tracking” files, where successful “left” and “right” keys correspond to 1 and 0, respectively for “recorded\_response” in “Staircase\_tracking” files, but they correspond to 0 and 1 for “recorded\_response” in “Validation\_tracking” files. This is because the “correct answer” (here “left”, or “No I did not see it”) must be set to 1 for the Psychopy StairHandler object, however for both procedures (“Scaling” and “Heat\_Map”) used in the “Validation section”, it is more convenient to

assign 1 to “right” to calculate the visibility score (the likelihood of responding “Yes I saw it”).

## 5.2. PSYDAT files

As mentioned above, you should have several PSYDAT files aside from JSON and CSV files. The number of these PSYDAT files depends on the number of border points that were adjusted by the “Staircase section”. These files are named in the format of

“Participant\_Session\_expName\_Y\_M\_D\_HrMin(horizontal, vertical).psydat”, where “(horizontal, vertical)” indicates the position (in pix) of the border point estimated in the “Border points detection section” that was adjusted by the current StairHandler object. For example, “p1\_s1\_BS\_mapping\_2021\_Mar\_01\_1201(183.6, 63.5).psydat”. These files are pickled files that each contains a StairHandler object, so you can retrieve the object by reading the file and visualize the staircase trace using a Psychopy official demo “JND\_staircase\_analysis.py”.

In addition, you may also have a PSYDAT file named like

“Participant\_Session\_expName\_Y\_M\_D\_HrMin.psydat”, which is the file storing the ExperimentHandler object that contains all variables and objects in the whole experiment, and you can retrieve this object just by reading this file. However, the automatic data-saving does not always work correctly, so you may or may not have this file, or have multiple duplicates.

Finally, you should do post-experiment data analysis and visualization by yourself.

## **6. About the Script**

Here we explain some designs of the experimental sections and procedures, how some default parameter values are determined, and script codes.

### **6.1. Visual stimuli unit**

This script uses “pix” as the window and visual stimuli unit, although Psychopy also supports “deg” (degrees of visual angle) and degFlat (degrees of visual angle corrected for standard flat screens). The latter two units are more convenient, but using “deg” on a flat-screen will change the physical size of the target object at different positions during the “Border points detection section” and the “Staircase section”, so the target object will have varying light intensity at different positions; “degFlat” can correct the error introduced by the flat screen, but the physical shape of the target object will change slightly. So we choose “pix” to precisely draw all visual stimuli. However, if using “deg” or “degFlat” is of great benefits, you can modify the code and use these units. An example experiment script using “deg” is given in the folder “ExampleExp”.

The limitations of using “pix” are that all default settings in pix are only suitable for a viewing distance of 57 cm and that you need to write additional codes to convert pix to deg for data analysis and visualization.

### **6.2. Influence of screen and visual stimuli parameters**

The contrast (such as color and luminance) between the visual stimuli and the background may influence the estimated blind spot, so screen and visual stimuli parameters that determine the contrast matter. These parameters include the screen gamma value, background color, visual stimuli color, visual stimuli flickering frequency, visual stimuli size, and visual stimuli texture (i.e., the texture of the target object used by the “Scaling” procedure in the “Validation section”). Greater contrasts should result in smaller estimates of the blind spot. This can be inferred if you think about the “Heat\_Map” procedure which estimates the visual sensitivity across the tested monocular visual field to a stimulus with a constant contrast – a stimulus with higher contrast against the background should be more visible, so it should narrow the area that is insensitive to it.

### **6.3. Influence of “inward” and “outward” tests in the “Border points detection section”**

In the “Border points detection section”, the subject has to move the cursor and during which click the mouse when the cursor perceptually “disappeared” or “reappeared”. This move-and-click task includes “inward” and “outward” tasks. An “inward” task refers to the subject moving the cursor into the blind spot and clicking to report a perceptual “disappearance”, while an “outward” task refers to moving the cursor outside the blind spot and “reappearance”.

The blind spot estimated based on “inward” tasks should be smaller than that estimated from “outward” tasks. The position of the click is determined by the position of the center of the cursor. A “disappearance” of a cursor (“inward” task) naturally corresponds to a position closer to the blind spot center, compared with a “reappearance” of the same cursor (“outward” task).

To balance the opposite biases, you should ask the subject to move the cursor across the blind spot forth and back for many rounds, so that each border point is estimated from equal numbers of “inward” and “outward” tasks.

### **6.4. Excluding in-situ repeated clicks in the “Border points detection section”**

This design is to reduce biases. This script uses `sklearn.KMeans` to cluster the clicks. Therefore, imagine that the subject did not move the cursor but still clicked at the same position many times – this would lead to incorrectly high weight for this position, resulting in incorrect clustering and averaging.

However, the code implementing this design is more or less flawed. Ideally, the script should continuously detect mouse/cursor movement to determine if the subject is doing in-situ repeated clicks. However, the code is written to compare the current successful click with the last saved successful click – if they are the same, the current successful click will not be saved for blind spot mapping (but still logged). In Python, this is done by:

```
if click_list and click_list[-1] == this_click:
    pass
```

else:

click\_list.append(this\_click)

give feedback indicating a successful saving

Although the implementation has a different logic from the desired feature, it is unlikely to lead to an error, because it is less likely that a subject could click at one position, move the cursor, and then click at the exactly same position in the next click.

### **6.5. The blind spot location estimated in the “Border points detection section” will not be adjusted according to the final border points after the “Staircase section”**

The blind spot location is not adjusted according to the border points after the “Staircase section”. This seems weird, but whether or not to adjust it depends on the definition of “location”. According to the primary logic of a three-step method that we are using in the “Border points detection section”, the blind spot “location” is the point through which the largest width and height can be assessed, in other words, the point should be the intersection of blind spot width and height.

If the blind spot were a perfect ellipse, this “location” would also be the centroid of that ellipse. In this case, it is worth adjusting the estimated “location” after the “Staircase section”. However, the blind spot shape is possible to be irregular and even slanted, so the centroid may not be the “location”.

Here we regard the blind spot “location” as the intersection of width and height and do not make further assumptions about it. For this reason, adjustment of “location” after the staircase procedure is not needed, because the “Staircase section” does not change the intersection of the blind spot width and height.

An alternative way to add staircase effects into location estimate is to mix the “Border points detection section” with the “Staircase section”. For example, after estimating one pair of border points along one path, run the staircase procedure on these two points, then estimate the next pair along the next path. This design should be more accurate, but in this way, all border points will need to be examined by the staircase procedure. This will

increase the time cost for a “Low” mode experiment. In the mixing design, in “Low” mode, all 6 border points estimated in the “Border points detection section” will need to be estimated by the staircase procedure, so the staircase procedure will be run 50% more trials than the current design (4 cardinal points out of 6 points examined by 4 staircase procedures).

### **6.6. Default staircase rules are unusual**

Indeed, the default staircase rules are 3-down-1-up, 20-trial, 4-reversal, and final-1-trial (threshold value computed based on the final trial, however, the common way is to compute the mean of final-n reversals or trials). The trial number of 20 is somewhat small. These rules can lead to a high standard deviation.

However, time-cost and the subject getting bored and fatigue are realistic and serious problems. Before finalizing our designs, we had done a pilot study (3 subjects), where we used 3-down-1-up, 50-trial, 5-reversal, and final-10-trial rules. All subjects reported unbearable eye discomfort and impatience due to the lengthy procedure, and they strongly suggested reducing the number of trials. On the other hand, the staircase trace revealed that after around 10 to 15 trials, 3 reversals had been made, and the staircase had reached the smallest step size (3 pix, around  $0.09^\circ$ ) and the threshold level (also see Figure 3, the example staircase trace for the current staircase rules, in our paper), so additional trials seemed less useful. Our current rules should work and are not too taxing for subjects.

### **6.7. Drift correction**

In addition to the drift correction in eye-tracker calibration procedure, users may wish to turn on the online drift correction function of the eye-tracker, and regularly run a drift correction procedure during the experiment, because the drift of eye-tracker signals can be significant in blind spot mapping. This may be applicable if the user wishes to apply a smaller fixation tolerance level than the default  $1.5^\circ$  (see Section 3.3,2).

### **6.8. Combine the script with other experiments**

You may want to use this script to locate the blind spot to play with it in other experiments.



In our paper, we have recommended several ways to shorten one experiment, including using “Scaling” rather than “Heat\_Map” in the “Validation section”, using “Low” mode for the “Border points detection section”, skip the “Validation section” and/or the “Staircase section”, and locating the blind spot then displaying visual stimuli in the “safe zone”.

You may want to involve the experiment script as part of your larger experiment script. However, you may find the codes in the file “BS\_mapping.py” too lengthy and unreadable, so it is hard to embed them into your codes. This is due to that the programmer was fresh in Psychopy and software development when he was developing this script, and that this experiment was initially developed using Psychopy Builder (a graphic programming tool of Psychopy) and then converted into codes, so the codes involved many useless code blocks made by the Builder. Now it is very difficult to rewrite the codes.

An elegant, easier way to utilize the script “BS\_mapping.py” is to use it as a standalone tool to map the blind spot and output results to files, and then let your experiment scripts read these files to use mapping results.