ret_map Documentation

Release 1.0.0

JY

Mar 16, 2018

Contents

CHAPTER 1

About

The retinotopic mapping package is a self-contained module for display visual stimuli in visual physiology experiments and for data analysis on the results of those experiments.

The visual stimuli generation and display is implemented in the modules `MonitorSetup.py`, `StimulusRoutines.py` and `DisplayStimulus.py`. These modules allow you to display flashing circle, sparse noise, locally sparse noise, drifting grading circle, static grading circle and others with spherical correction. The method for spherical correction is the same as Marshel et al. 2011 (2). These stimulus routines are highly customizable and designed to give the user significant flexibility and control in creative experimental design.

Please check the 'examplesvisual_stimulation' folder for example script example_stimulation.py of visual stimulation.

One specific analysis this package can perform is automated segmentation of the mouse visual cortex, which is implemented in `RetinotopicMapping.py` module. The experimental setup and analysis routine was modified from Garrett et al. 2014 (1), and closely follows the protocols and procedures documented in Juavinett et al. 2016 (2).

The analysis takes visual altitude and azimuth maps of mouse cortex as inputs, calculates the visual sign of each pixel and auto-segments the cortical surface into primary visual cortex and multiple higher visual cortices. Ideally, the visual altitude and azimuth maps can be generated by fourier analysis of population cortical responses to periodic sweeping checker board visual stimuli (3, 4).

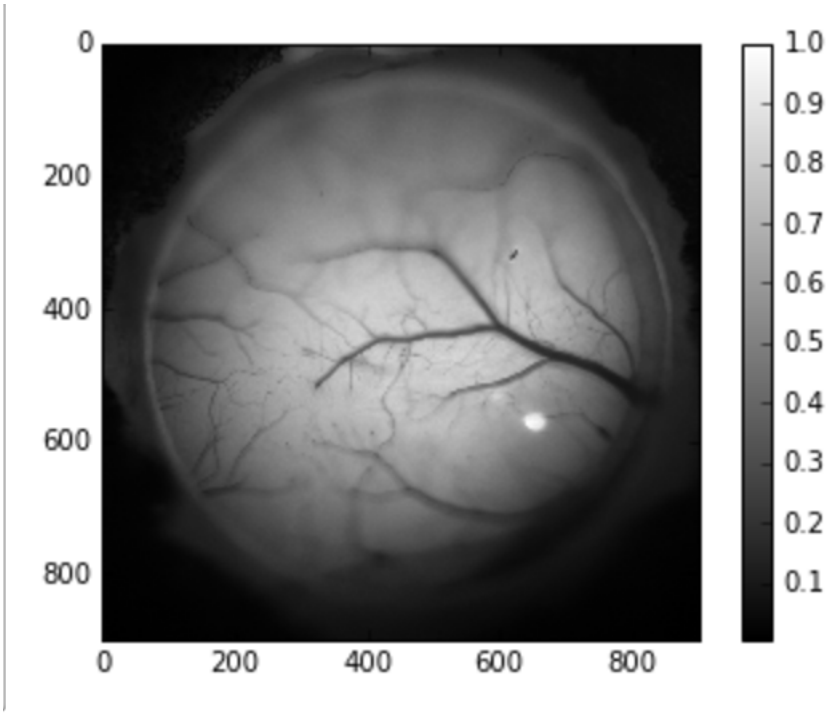
The package also provides some useful plotting functions to visualize the results.

Please check the 'examplessignmap_analysis' folder for a jupyter notebook showing automated visual area segmentation of mouse cortex.

1.1 What is a retinotopic map?

Retinotopic maps are a common tool used in systems neuroscience to understand how receptive fields are mapped onto particular regions of the brain. In the lower visual areas of certain species, neurons are organized as a 2D representation of the visual image that is formed on the retina.

The following video gives an example of a retinopic map, showing how a moving dot is represented in the mouse brain.



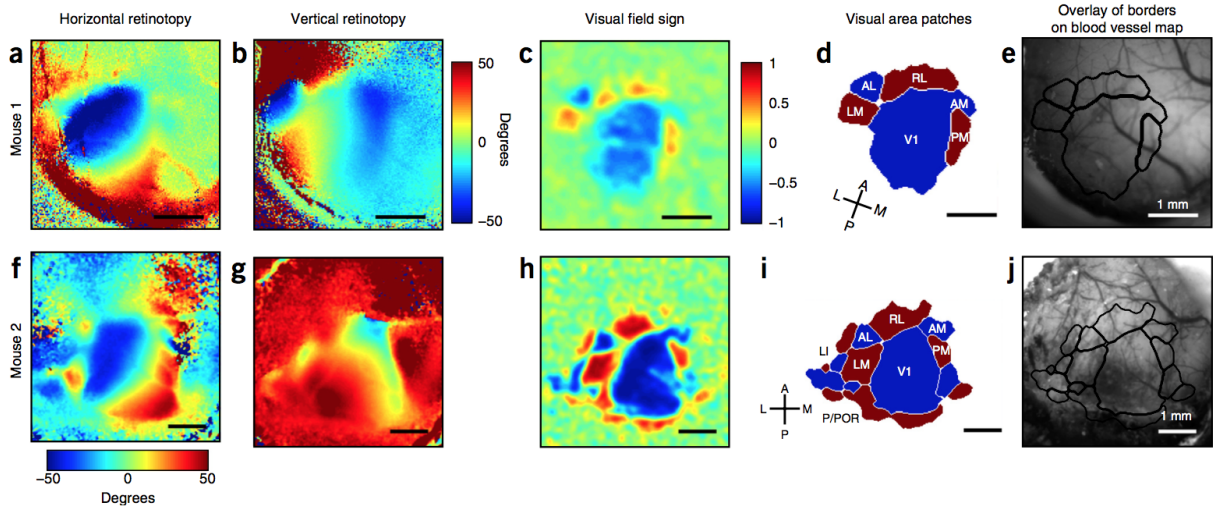


Fig. 1: Image taken from (2) in *Citations*.

1.2 Contributors

- Jun Zhuang @zhuang1981
- John Yearsley @jhyearsley
- Derric Williams @derricw

1.3 Citations

1. Garrett ME, Nauhaus I, Marshel JH, Callaway EM (2014) Topography and areal organization of mouse visual cortex. J Neurosci 34:12587-12600.

2. Juavinett AL, Nauhaus I, Garrett ME, Zhuang J, Callaway EM (2017). Automated identification of mouse visual areas with intrinsic signal imaging. Nature Protocols. 12: 32-43.

3. Kalatsky VA, Stryker MP (2003) New paradigm for optical imaging: temporally encoded maps of intrinsic signal. Neuron 38:529-545.

4. Marshel JH, Kaye AP, Nauhaus I, Callaway EM (2012) Anterior-posterior direction opponency in the superficial mouse lateral geniculate nucleus. Neuron 76:713-720.

5. Sereno MI, Dale AM, Reppas JB, Kwong KK, Belliveau JW, Brady TJ, Rosen BR, Tootell RB (1995) Borders of multiple visual areas in humans revealed by functional magnetic resonance imaging. Science 268:889-893.

6. Sereno MI, McDonald CT, Allman JM (1994) Analysis of retinotopic maps in extrastriate cortex. Cereb Cortex 4:601-620.

CHAPTER 2

Installing

2.1 Manual install (recommended)

To manually install **RetinotopicMapping** you can download the package here.

Then open the command line, move to the directory that the package was installed in and run the *setup.py* file like follows:

```
cd <package_path>
conda env create -f environment.yml     # this will take a while
activate retinotopic_mapping            # Windows
source activate retinotopic_mapping     # Mac or Linux
python setup.py install
```

2.2 PIP install

An even simpler way to install the package is with pip. To do so type the following in the command line:

```
pip install retinotopic_mapping
```

This will install all the dependencies (takes long time and not always work). If you want to install the package without dependencies, run:

```
pip install retinotopic_mapping --no-deps
```

2.3 Dependencies

1. pytest, version 3.3.0 or later
2. numpy, version 1.13.1 or later

3. scipy, version 0.17.1 or later
4. matplotlib, version 1.5.1 or later
5. h5py, version 2.7.1 or later
6. pillow, version 5.0.0 or later
7. psychopy, version 1.85.2 or later
8. pyglet, version 1.2.4
9. OpenCV *[a]*
10. scikit-image, version 0.12.3 or later *[b]*
11. tifffile, version >=0.7.0 *[c]*
12. PIL, version 4.3.0 or later
13. PyDAQmx, version 1.3.2 or later *[d]*
14. configobj, version 5.0.6 or later
15. sphinx, version 1.6.3 or later (just for documentation)
16. numpydoc, version 0.7.0 (just for documentation)

2.4 Operating Systems Supported

- Windows
- Linux
- Mac

2.5 Making sure PyDAQmx works

The PyDAQmx module allows users to integrate National Instruments data-acquisition hardware into their experimental setup (see their documentation for more information).

To get PyDAQmx to function correctly there are a couple of important points to mention:

- The NIDAQmx driver must first be installed
- Once NIDAQmx driver is installed it is not guaranteed that the module will work, so it is important to know how to troubleshoot this issue. The main issue is tracking down where two files are on your computer, a filepath ending with `DAQMX.h` and another path ending with `nicaiu.dll`. The PyDAQmx module tries to find these files for you, but if it cannot, the user needs to manually find and enter the path within the *DAQmxConfig.py* file. See this page for a more thorough explanation).

Note: This is the most likely issue to come up in debugging. Chances are if you are having a related issue it either has something to do with not supplying a correct path or making improper import statements somewhere in your script.

CHAPTER 3

Getting Started

The first thing to do after downloading the retinotopic mapping module is to set up a monitor to display some stimulus routines. Ideally you should have two monitors, one for running your python scripts and another for displaying stimulus. This will allow you to get familiar with the code base in a simple and straightforward manner, as well as get some practice in debugging your experimental setup.

After working through this tutorial you should be able to display simple stimulus routines, and be familiar enough with the basic functionality of the experimental part of the code base to debug your own python scripts.

3.1 Setting up your monitor

The `MonitorSetup` module is used to store and keep track of all your experimental display monitor specs as well as the geometry of the experiment.

There are two classes within the module: `Monitor` and `Indicator`. For now we won't pay much attention to the `Indicator` class, except to mention that it is used for timing purposes.

Instead we will focus on the `Monitor` class, what it does and how it works. But first, here is a visualization of the experimental setup to get an idea of what kind of geometerical parameters are stored in the object.

The specs you will need to initialize a monitor object are as follows:

- Monitor resolution (in pixels).
- Monitor refresh rate (in Hz).
- Monitor width/height (in cm).
- Distance from the mouse's eyeball to the monitor (in cm).
- Distance from the mouse's gaze center to the top of the monitor (in cm).
- Distance from the mouse's gaze center to the anterior edge of the monitor (in cm).
- Angle between the mouse's body axis and the plane that the monitor lies in (in deg).

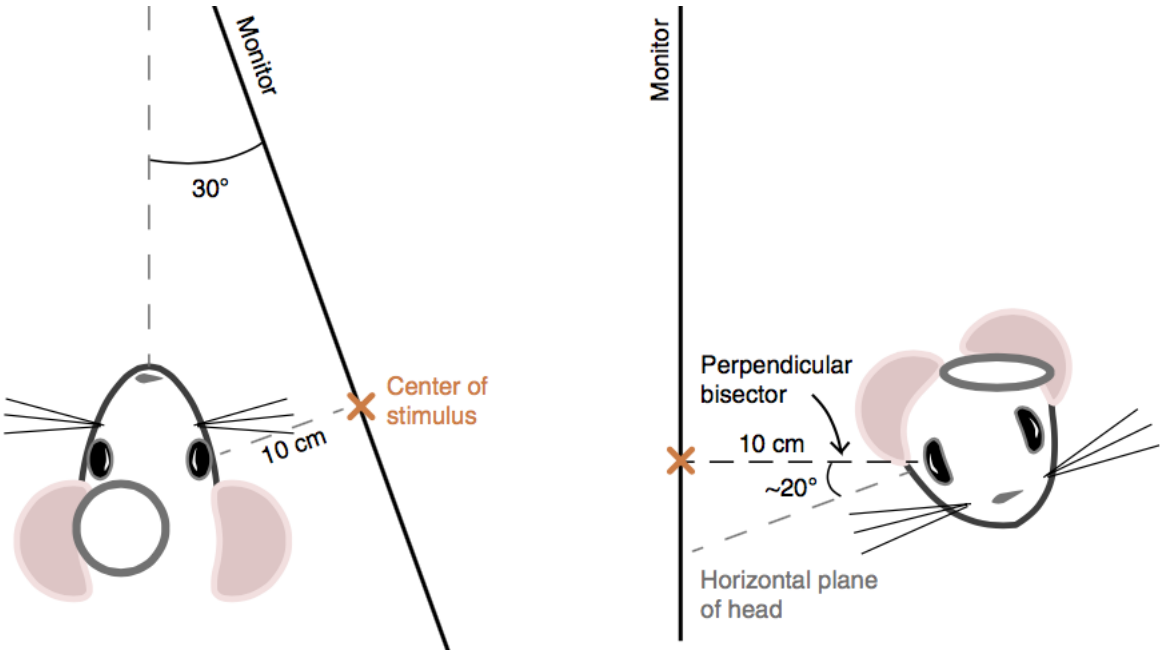


Fig. 1: Image taken from (2) in *Citations*.

It's not important that you know the exact parameters right away but you should definitely check to see if you are able to display stimulus. Since this is a package to both analyze **and** display experimental results, it is important that you are able to play around with all of the parameters that define the visual stimuli. That way you will have significant freedom and creativity in customizing your own routines.

3.2 Testing your display setup

Since we are just interested in trying to display some stimulus quickly, don't worry too much about having exact numbers for all of these parameters. It turns out that PsychoPy will automatically resize your display monitor to fit the screen if you put in the wrong parameters. So just put something in and try to make sure everything works on your machine before going any further.

Let's assume we have a monitor with resolution 1080x1920, refresh rate of 60Hz, and height and width of 90 and 50 cm respectively. Neglect the geometry of the experimental setup for now unless you already know your exact parameters.

Then under these assumptions we can initialize `Monitor` and `Indicator` objects as follows:

```
>>> from MonitorSetup import Monitor, Indicator
>>> mon = Monitor(resolution=(1080,1920),
                  refresh_rate=60.,
                  mon_width_cm=90.,
                  mon_height_cm=50.,
                  dis=20,
                  mon_tilt=10.
                  C2T_cm=30.
                  C2A_cm=40.)
>>> ind = Indicator(mon)
```

3.3 Displaying Stimulus

Now we are ready to display a stimulus routine on your monitor from the `DisplaySequence` class. There are several routines to choose from the `Stimulus` module (each of which can be previewed *here*), but for this example we will initialize the `FlashingCircle` routine which takes the `mon` and `ind` variables as parameters.

```
>>> import StimulusRoutines as stim
>>> from DisplayStimulus import DisplaySequence
>>> flashing_circle = stim.FlashingCircle(mon,ind)
>>> ds = DisplaySequence(log_dir='log_directory')
>>> ds.trigger_display()
```

which will give an output that should look something like this

---

Note: If you are having trouble with setting up your display monitor try experimenting with the `for_debugging.py` script found in the primary module folder. This script can be used to display template routines and debug your setup to get up and running.

CHAPTER 4

Examples

4.1 Kalatsky & Stryker

4.2 Sparse Noise

4.3 Drifting Grating Circle

CHAPTER 5

API Documentation

5.1 retinotopic_mapping.MonitorSetup

Used to store the display monitor and particular geometry used within a given experimental setup. The *Monitor* class holds references to the sizing of the monitor that is used to display stimulus routines and contains the necessary geometrical description of where the subject's eye is placed with respect to the display monitor. The *Indicator* class, on the other hand, is generally used in order to gain precise temporal information of the stimuli display. Basically a indicator is just a small square shows up at one corner of the display screen. It changes color during visual stimuli display. By covering it with a photodiode with high temporal precision and syncing the photodiode signal into data acquisition system, the experimenter will get the ground truth of display timestamps. Ideally the flashes of an indicator will be synchronized with the onsets of stimuli.

The module will most definitely be used in conjunction with the `DisplayStimulus` and `StimulusRoutines` modules.

5.1.1 Monitor

class retinotopic_mapping.MonitorSetup.**Monitor**(*resolution*, *dis*, *mon_width_cm*, *mon_height_cm*, *C2T_cm=None*, *C2A_cm=None*, *center_coordinates=(0.0, 60.0)*, *visual_field='right'*, *deg_coord_x=None*, *deg_coord_y=None*, *name='testMonitor'*, *gamma=None*, *gamma_grid=None*, *luminance=None*, *downsample_rate=10*, *refresh_rate=60.0*)

monitor object created by Jun, has the method "remap" to generate the spherical corrected coordinates in degrees

This object contains the relevant data for the monitor used within a given experimental setup. When initialized, the rectangular coordinates of the pixels on the monitor are computed and stored as *lin_coord_x*, *lin_coord_y*. The rectangular coordinates are then transformed and warped by calling the *remap* method to populate the *deg_coord_x* and *deg_coord_y* attributes.

Parameters **resolution** : tuple

value of the monitor resolution

dis : float

distance from eyeball to monitor (in cm)

mon_width_cm : float

width of monitor (in cm)

mon_height_cm : float

height of monitor (in cm)

C2T_cm : float

distance from gaze center to monitor top

C2A_cm : float

distance from gaze center to anterior edge of the monitor

center_coordinates : tuple of two floats

(altitude, azimuth), in degrees. the coordinates of the projecting point from the eye ball to the monitor. This allows to place the display monitor in any arbitrary position.

visual_field : str from {'right','left'}, optional

the eye that is facing the monitor, defaults to 'right'

deg_coord_x : ndarray, optional

array of warped x pixel coordinates, defaults to *None*

deg_coord_y : ndarray, optional

array of warped y pixel coordinates, defaults to *None*

name : str, optional

name of the monitor, defaults to *testMonitor*

gamma : optional

for gamma correction, defaults to *None*

gamma_grid : optional

for gamme correction, defaults to *None*

luminance : optional

monitor luminance, defaults to *None*

downsample_rate : int, optional

downsample rate of monitor pixels, defaults to 10

refresh_rate : float, optional

the refresh rate of the monitor in Hz, defaults to 60

5.1.1.0.1 Methods

generate_lookup_table()	generate lookup talbe between degree corrdinates and linear corrdinates return two matrix: lookupI: i index in linear matrix to this pixel after warping lookupJ: j index in linear matrix to this pixel after warping
plot_map()	
remap()	warp the linear pixel coordinates to a spherical corrected representation.
set_downsample_rate(downsample_rate)	
set_gamma(gamma, gamma_grid)	
set_luminance(luminance)	
warp_images(imgs, center_coor[, ...])	warp a image stack into visual degree coordinate system

generate_lookup_table()
generate lookup talbe between degree corrdinates and linear corrdinates return two matrix: lookupI: i index in linear matrix to this pixel after warping lookupJ: j index in linear matrix to this pixel after warping

remap()
warp the linear pixel coordinates to a spherical corrected representation.

Function is called when the monitor object is initialized and populate the *deg_coord_x* and *deg_coord_y* attributes.

warp_images(*imgs*, *center_coor*, *deg_per_pixel=0.1*, *is_luminance_correction=True*)
warp a image stack into visual degree coordinate system

Parameters **imgs** : ndarray

should be 2d or 3d, if 3d, axis will be considered as frame x rows x width

center_coor : list or tuple of two floats

the visual degree coordinates of the center of the image (altitude, azimuth)

deg_per_pixel : float or list/tuple of two floats

size of original pixel in visual degrees, (altitude, azimuth), if float, assume sizes in both dimension are the same

is_luminance_correction : bool

if True, wrapped images will have mean intensity equal 0, and values will be scaled up to reach minimum equal -1. or maximum equal 1.

Returns **imgs_wrapped** : 3d array, np.float32

wrapped images, each frame should have exact same size of down sampled monitor resolution. the region on the monitor not covered by the image will have value of np.nan. value range [-1., 1.]

coord_alt_wrapped : 2d array, np.float32

the altitude coordinates of all pixels in the wrapped images in visual degrees. should have the same shape as each frame in 'imgs_wrapped'.

coord_azi_wrapped : 2d array, np.float32

the azimuth coordinates of all pixels in the wrapped images in visual degrees. should have the same shape as each frame in 'imgs_wrapped'.

imgs_dewrapped : 3d array, dtype same as imgs

unwrapped images, same dimension as input image stack. the region of original image that was not got displayed (outside of the monitor) will have value of np.nan. value range [-1., 1.]

coord_alt_dewrapped : 2d array, np.float32

the altitude coordinates of all pixels in the dewrapped images in visual degrees. should have the same shape as each frame in 'imgs_dewrapped'.

coord_azi_dewrapped : 2d array, np.float32

the azimuth coordinates of all pixels in the dewrapped images in visual degrees. should have the same shape as each frame in 'imgs_dewrapped'.

5.1.2 Indicator

class retinotopic_mapping.MonitorSetup.**Indicator**(*monitor*, *width_cm=3.0*, *height_cm=3.0*, *position='northeast'*, *is_sync=True*, *freq=2.0*)

flashing indicator for photodiode

Parameters **monitor** : monitor object

The monitor used within the experimental setup

width_cm : float, optional

width of the size of the indicator in cm, defaults to *3.*

height_cm : float, optional

height of the size of the indicator in cm, defaults to *3.*

position : str from {'northeast','northwest','southwest','southeast'}

the placement of the indicator, defaults to 'northeast'

is_sync : bool, optional

determines whether the indicator is synchronized with the stimulus, defaults to True.

freq : float, optional

frequency of photodiode, defaults to *2.*

5.1.2.0.1 Methods

get_center()	
get_frames()	if not synchronized with stimulation, get frame numbers of each update of indicator
get_size_pixel()	

get_frames()

if not synchronized with stimulation, get frame numbers of each update of indicator

5.2 retinotopic_mapping.StimulusRoutines

This module define a base class `Stim` for visual stimulation. Each subclass of `Stim` defines a particular type of visual stimulus, i.e. `UniformContrast` or `SparseNoise`. When initiated, these subclasses take various parameter inputs to generate stimulus arrays and metadata dictionary which can be passed to the `DisplayStimulus.DisplaySequence` for displaying. Each subclass will have a method called *generate_movie()* or *generate_movie_by_index()* or both. Only when these methods are called, will heavy lifting calculation take place.

5.2.1 Stim

class retinotopic_mapping.StimulusRoutines.**Stim**(*monitor*, *indicator*, *background=0.0*, *coordinate='degree'*, *pregap_dur=2.0*, *postgap_dur=3.0*)

generic class for visual stimulation. parent class for individual stimulus routines.

Parameters **monitor** : monitor object

the monitor used to display stimulus in the experiment

indicator : indicator object

the indicator used during stimulus

background : float, optional

background color of the monitor screen when stimulus is not being presented, takes values in [-1,1] and defaults to *0.* (grey)

coordinate : str {'degree', 'linear'}, optional

determines the representation of pixel coordinates on monitor, defaults to 'degree'

pregap_dur : float, optional

duration of gap period before stimulus, measured in seconds, defaults to *2.*

postgap_dur : float, optional

duration of gap period after stimulus, measured in seconds, defaults to *3.*

5.2.1.0.1 Attributes

postgap_frame_num
pregap_frame_num

5.2.1.0.2 Methods

clear()	
generate_frames()	place holder of function "generate_frames" for each specific stimulus
generate_movie()	place holder of function 'generate_movie' for each specific stimulus
generate_movie_by_index()	place holder of function generate_movie_by_index() for each specific stimulus

Continued on next page

Table 4 – continued from previous page

set_background(background)
set_coordinate(coordinate)
set_indicator(indicator)
set_monitor(monitor)
set_postgap_dur(postgap_dur)
set_pregap_dur(pregap_dur)

5.2.2 UniformContrast

class retinotopic_mapping.StimulusRoutines.**UniformContrast**(*monitor*, *indicator*, *duration*, *color=0.0*, *pregap_dur=2.0*, *postgap_dur=3.0*, *background=0.0*, *coordinate='degree'*)

Generate full field uniform luminance for recording spontaneous activity. Inherits from Stim.

The full field uniform luminance stimulus presents a fixed background color which is normally taken to be grey.

Parameters **monitor** : monitor object

contains display monitor information

indicator : indicator object

contains indicator information

coordinate : str from {'degree','linear'}, optional

specifies coordinates, defaults to 'degree'

background : float, optional

color of background. Takes values in [-1,1] where -1 is black and 1 is white

pregap_dur : float, optional

amount of time (in seconds) before the stimulus is presented, defaults to *2.*

postgap_dur : float, optional

amount of time (in seconds) after the stimulus is presented, defaults to *3.*

color : float, optional

the choice of color to display in the stimulus, defaults to *0.* which is grey

5.2.2.0.1 Attributes

postgap_frame_num
pregap_frame_num

5.2.2.0.2 Methods

clear()

Continued on next page

Table 6 – continued from previous page

generate_frames()	generate a tuple of parameters with information for each frame.
generate_movie()	generate movie for uniform contrast display frame by frame.
generate_movie_by_index()	compute the stimulus movie to be displayed by index.
set_background(background)	
set_coordinate(coordinate)	
set_indicator(indicator)	
set_monitor(monitor)	
set_postgap_dur(postgap_dur)	
set_pregap_dur(pregap_dur)	

generate_frames()
generate a tuple of parameters with information for each frame.

Information contained in each frame:

first element - during display frames, value takes on 1 and value is 0 otherwise

second element - color of indicator during display value is equal to 1 and during gaps value is equal to -1

generate_movie()
generate movie for uniform contrast display frame by frame.

Returns **full_seq** : nd array, uint8

3-d array of the stimulus to be displayed.

full_dict : dict

dictionary containing the information of the stimulus.

generate_movie_by_index()
compute the stimulus movie to be displayed by index.

5.2.3 FlashingCircle

class retinotopic_mapping.StimulusRoutines.**FlashingCircle**(*monitor*, *indicator*, *coordinate='degree'*, *center=(0.0, 60.0)*, *radius=10.0*, *is_smooth_edge=False*, *smooth_width_ratio=0.2*, *smooth_func=<function blur_cos>*, *color=-1.0*, *flash_frame_num=3*, *pregap_dur=2.0*, *postgap_dur=3.0*, *background=0.0*, *midgap_dur=1.0*, *iteration=1*)

Generate flashing circle stimulus.

Stimulus routine presents a circle centered at the position *center* with given *radius*.

Parameters **monitor** : monitor object

contains display monitor information

indicator : indicator object

contains indicator information

coordinate : str from {'degree','linear'}, optional

specifies coordinates, defaults to 'degree'

background : float, optional

color of background. Takes values in [-1,1] where -1 is black and 1 is white

pregap_dur : float, optional

amount of time (in seconds) before the stimulus is presented, defaults to *2.*

postgap_dur : float, optional

amount of time (in seconds) after the stimulus is presented, defaults to *3.*

center : 2-tuple, optional

center coordinate (altitude, azimuth) of the circle in degrees, defaults to (0.,60.).

radius : float, optional

radius of the circle, defaults to *10.*

is_smooth_edge : bool

True, smooth circle edge with smooth_width_ratio and smooth_func False, do not smooth edge

smooth_width_ratio : float, should be smaller than 1.

the ratio between smooth band width and radius, circle edge is the middle of smooth band

smooth_func : function object

this function take to inputs first, ndarray storing the distance from each pixel to smooth band center second, smooth band width

returns smoothed mask with same shape as input ndarray

color : float, optional

color of the circle, takes values in [-1,1], defaults to *-1.*

iteration : int, optional

total number of flashes, defaults to *1*.

flash_frame : int, optional

number of frames that circle is displayed during each presentation of the stimulus, defaults to *3*.

5.2.3.0.1 Attributes

`midgap_frame_num`
`postgap_frame_num`
`pregap_frame_num`

5.2.3.0.2 Methods

clear()	
generate_frames()	function to generate all the frames needed for the stimulation.
generate_movie()	generate movie frame by frame.
generate_movie_by_index()	compute the stimulus movie to be displayed by index.
set_background(background)	
set_center(center)	
set_color(color)	
set_coordinate(coordinate)	
set_flash_frame_num(flash_frame_num)	
set_indicator(indicator)	
set_monitor(monitor)	
set_postgap_dur(postgap_dur)	
set_pregap_dur(pregap_dur)	
set_radius(radius)	

generate_frames()
function to generate all the frames needed for the stimulation.

Information contained in each frame:

first element : during a gap, the value is equal to 0 and during display the value is equal to 1

second element : corresponds to the color of indicator if indicator.is_sync is True, during stimulus the value is equal to 1., whereas during a gap the value isequal to -1.; if indicator.is_sync is False, indicator color will alternate between 1. and -1. at the frequency as indicator.freq

frames [list] list of information defining each frame.

generate_movie()
generate movie frame by frame.

generate_movie_by_index()
compute the stimulus movie to be displayed by index.

5.2.4 SparseNoise

class retinotopic_mapping.StimulusRoutines.**SparseNoise**(*monitor*, *indicator*, *background=0.0*, *coordinate='degree'*, *grid_space=(10.0, 10.0)*, *probe_size=(10.0, 10.0)*, *probe_orientation=0.0*, *probe_frame_num=6*, *subregion=None*, *sign='ON-OFF'*, *iteration=1*, *pregap_dur=2.0*, *postgap_dur=3.0*, *is_include_edge=True*)
generate sparse noise stimulus integrates flashing indicator for photodiode

This stimulus routine presents quasi-random noise in a specified region of the monitor. The *background* color can be customized but defaults to a grey value. Can specify the *subregion* of the monitor where the pixels will flash on and off (black and white respectively)

Parameters **monitor** : monitor object

contains display monitor information

indicator : indicator object

contains indicator information

coordinate : str from {'degree','linear'}, optional

specifies coordinates, defaults to 'degree'

background : float, optional

color of background. Takes values in [-1,1] where -1 is black and 1 is white

pregap_dur : float, optional

amount of time (in seconds) before the stimulus is presented, defaults to *2.*

postgap_dur : float, optional

amount of time (in seconds) after the stimulus is presented, defaults to *3.*

grid_space : 2-tuple of floats, optional

first coordinate is altitude, second coordinate is azimuth

probe_size : 2-tuple of floats, optional

size of flicker probes. First coordinate defines the width, and second coordinate defines the height

probe_orientation : float, optional

orientation of flicker probes

probe_frame_num : int, optional

number of frames for each square presentation

subregion : list or tuple

the region on the monitor that will display the sparse noise, list or tuple, [min_alt, max_alt, min_azi, max_azi]

sign : {'ON-OFF', 'ON', 'OFF'}, optional

determines which pixels appear in the *subregion*, defaults to *'ON-Off'* so that both on and off pixels appear. If *'ON* selected only on pixels (white) are displayed in the noise *subregion while if ''OFF'* is selected only off (black) pixels are displayed in the noise

iteration : int, optional

number of times to present stimulus, defaults to *1*

is_include_edge : bool, default True,

if True, the displayed probes will cover the edge case and ensure that the entire subregion is covered. If False, the displayed probes will exclude edge case and ensure that all the centers of displayed probes are within the subregion.

5.2.4.0.1 Attributes

postgap_frame_num
pregap_frame_num

5.2.4.0.2 Methods

clear()	
generate_frames()	function to generate all the frames needed for SparseNoise stimulus
generate_movie()	generate movie for display frame by frame
generate_movie_by_index()	compute the stimulus movie to be displayed by index.
set_background(background)	
set_coordinate(coordinate)	
set_indicator(indicator)	
set_monitor(monitor)	
set_postgap_dur(postgap_dur)	
set_pregap_dur(pregap_dur)	

generate_frames()
function to generate all the frames needed for SparseNoise stimulus

returns a list of information of all frames as a list of tuples

Information contained in each frame:

first element - int when stimulus is displayed value is equal to 1, otherwise equal to 0,

second element - tuple, retinotopic location of the center of current square,[alt, azi]

third element - polarity of current square, 1 -> bright, -1-> dark

forth element - color of indicator

if synchronized [value equal to 0 when stimulus is not] begin displayed, and 1 for onset frame of stimulus for each square, -1 for the rest.

if non-synchronized: values alternate between -1 and 1 at defined frequency

for gap frames the second and third elements should be 'None'

generate_movie()
generate movie for display frame by frame

generate_movie_by_index()
compute the stimulus movie to be displayed by index.

5.2.5 LocallySparseNoise

class retinotopic_mapping.StimulusRoutines.**LocallySparseNoise**(*monitor*, *indicator*, *min_distance=20.0*, *background=0.0*, *coordinate='degree'*, *grid_space=(10.0, 10.0)*, *probe_size=(10.0, 10.0)*, *probe_orientation=0.0*, *probe_frame_num=6*, *subregion=None*, *sign='ON-OFF'*, *iteration=1*, *repeat=1*, *pregap_dur=2.0*, *postgap_dur=3.0*, *is_include_edge=True*)

generate locally sparse noise stimulus integrates flashing indicator for photodiode

This stimulus routine presents quasi-random noise in a specified region of the monitor. The *background* color can be customized but defaults to a grey value. Can specify the *subregion* of the monitor where the pixels will flash on and off (black and white respectively)

Different from SparseNoise stimulus which presents only one probe at a time, the LocallySparseNoise presents multiple probes simultaneously to speed up the sampling frequency. The sparsity of probes is defined by minimum distance in visual degree: in any given frame, the centers of any pair of two probes will have distance larger than minimum distance in visual degrees. The method generate locally sparse noise here insures, for each iteration, all the locations in the subregion will be sampled once and only once.

Parameters **monitor** : monitor object

contains display monitor information

indicator : indicator object

contains indicator information

coordinate : str from {'degree','linear'}, optional

specifies coordinates, defaults to 'degree'

background : float, optional

color of background. Takes values in [-1,1] where -1 is black and 1 is white

pregap_dur : float, optional

amount of time (in seconds) before the stimulus is presented, defaults to *2.*

postgap_dur : float, optional

amount of time (in seconds) after the stimulus is presented, defaults to *3.*

min_distance : float, default 20.

the minimum distance in visual degree for any pair of probe centers in a given frame

grid_space : 2-tuple of floats, optional

first coordinate is altitude, second coordinate is azimuth

probe_size : 2-tuple of floats, optional

size of flicker probes. First coordinate defines the width, and second coordinate defines the height

probe_orientation : float, optional

orientation of flicker probes

probe_frame_num : int, optional

number of frames for each square presentation

subregion : list or tuple

the region on the monitor that will display the sparse noise, list or tuple, [min_alt, max_alt, min_azi, max_azi]

sign : {'ON-OFF', 'ON', 'OFF'}, optional

determines which pixels appear in the *subregion*, defaults to *'ON-Off'* so that both on and off pixels appear. If *'ON* selected only on pixels (white) are displayed in the noise *subregion while if ''OFF'* is selected only off (black) pixels are displayed in the noise

iteration : int, optional

number of times to present stimulus with random order, the total number a paticular probe will be displayded will be iteration * repeat, defaults to *1*

repeat : int, optional

number of repeat of whole sequence, the total number a paticular probe will be displayded will be iteration * repeat, defaults to *1*

is_include_edge : bool, default True,

if True, the displayed probes will cover the edge case and ensure that the entire subregion is covered. If False, the displayed probes will exclude edge case and ensure that all the centers of displayed probes are within the subregion.

5.2.5.0.1 Attributes

`postgap_frame_num`
`pregap_frame_num`

5.2.5.0.2 Methods

`clear`()	
`generate_frames`()	place holder of function "generate_frames" for each specific stimulus
`generate_movie`()	place holder of function 'generate_movie' for each specific stimulus
`generate_movie_by_index`()	place holder of function generate_movie_by_index() for each specific stimulus
`set_background`(background)	
`set_coordinate`(coordinate)	

Continued on next page

Table 12 – continued from previous page

set_indicator(indicator)	
set_monitor(monitor)	
set_postgap_dur(postgap_dur)	
set_pregap_dur(pregap_dur)	

generate_frames()
place holder of function "generate_frames" for each specific stimulus

generate_movie()
place holder of function 'generate_movie' for each specific stimulus

generate_movie_by_index()
place holder of function generate_movie_by_index() for each specific stimulus

5.2.6 DriftingGratingCircle

class retinotopic_mapping.StimulusRoutines.**DriftingGratingCircle**(*monitor*, *indicator*, *background=0.0*, *coordinate='degree'*, *center=(0.0, 60.0)*, *sf_list=(0.08,)*, *tf_list=(4.0,)*, *dire_list=(0.0,)*, *con_list=(0.5,)*, *radius_list=(10.0,)*, *block_dur=2.0*, *midgap_dur=0.5*, *iteration=1*, *pregap_dur=2.0*, *postgap_dur=3.0*, *is_smooth_edge=False*, *smooth_width_ratio=0.2*, *smooth_func=<function blur_cos>*, *is_blank_block=True*)

Generate drifting grating circle stimulus

Stimulus routine presents drifting grating stimulus inside of a circle centered at *center*. The drifting gratings are determined by spatial and temporal frequencies, directionality, contrast, and radius. The routine can generate several different gratings within one presentation by specifying multiple values of the parameters which characterize the stimulus.

Parameters **monitor** : monitor object

contains display monitor information

indicator : indicator object

contains indicator information

coordinate : str from {'degree','linear'}, optional

specifies coordinates, defaults to 'degree'

background : float, optional

color of background. Takes values in [-1,1] where -1 is black and 1 is white

pregap_dur : float, optional

amount of time (in seconds) before the stimulus is presented, defaults to *2.*

postgap_dur : float, optional

amount of time (in seconds) after the stimulus is presented, defaults to *3.*

center : 2-tuple of floats, optional

coordintes for center of the stimulus (altitude, azimuth)

sf_list : n-tuple, optional

list of spatial frequencies in cycles/unit, defaults to *(0.08)*

tf_list : n-tuple, optional

list of temportal frequencies in Hz, defaults to *(4.)*

dire_list : n-tuple, optional

list of directions in degrees, defaults to *(0.)*

con_list : n-tuple, optional

list of contrasts taking values in [0.,1.], defaults to *(0.5)*

radius_list : n-tuple

list of radii of circles, unit defined by *self.coordinate*, defaults to *(10.)*

block_dur : float, optional

duration of each condition in seconds, defaults to *2.*

midgap_dur : float, optional

duration of gap between conditions, defaults to *0.5*

iteration : int, optional

number of times the stimulus is displayed, defaults to *1*

is_smooth_edge : bool

True, smooth circle edge with smooth_width_ratio and smooth_func False, do not smooth edge

smooth_width_ratio : float, should be smaller than 1.

the ratio between smooth band width and radius, circle edge is the middle of smooth band

smooth_func : function object

this function take two inputs: 1) ndarray storing the distance from each pixel to smooth band center; 2) smooth band width. returns smoothed mask with same shape as input ndarray

is_blank_block : bool

if True, one blank block (full screen background with the same duration of other blocks) will be displayed for each iteration. The frames of this condition will be: (1, 1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0), the meaning of these numbers can be found in self.frame_config

5.2.6.0.1 Attributes

block_frame_num
midgap_frame_num
postgap_frame_num
pregap_frame_num

5.2.6.0.2 Methods

clear()	
generate_frames()	function to generate all the frames needed for DriftingGratingCircle returns a list of information of all frames as a list of tuples
generate_movie()	Generate movie frame by frame
generate_movie_by_index()	compute the stimulus movie to be displayed by index.
set_background(background)	
set_coordinate(coordinate)	
set_indicator(indicator)	
set_monitor(monitor)	
set_postgap_dur(postgap_dur)	
set_pregap_dur(pregap_dur)	

generate_frames()

function to generate all the frames needed for DriftingGratingCircle returns a list of information of all frames as a list of tuples

Information contained in each frame:

first element - value equal to 1 during stimulus and 0 otherwise

second element - on first frame in a cycle value takes on 1, and otherwise is equal to 0.

third element - spatial frequency

forth element - temporal frequency

fifth element - direction, [0, 2*pi)

sixth element - contrast, [-1., 1.]

seventh element - size, float (radius of the circle in visual degree)

eighth element - phase, [0, 2*pi)

ninth element - indicator color [-1, 1]. Value is equal to 1 on the first frame of each cycle, -1 during gaps and otherwise 0.

during gap frames the second through the eighth elements should be 'None'.

generate_movie()
Generate movie frame by frame

generate_movie_by_index()
compute the stimulus movie to be displayed by index.

5.2.7 StaticGratingCircle

class retinotopic_mapping.StimulusRoutines.**StaticGratingCircle**(*monitor*, *indicator*, *background=0.0*, *coordinate='degree'*, *center=(0.0, 60.0)*, *sf_list=(0.08,)*, *ori_list=(0.0, 90.0)*, *con_list=(0.5,)*, *radius_list=(10.0,)*, *phase_list=(0.0, 90.0, 180.0, 270.0)*, *display_dur=0.25*, *midgap_dur=0.0*, *iteration=1*, *pregap_dur=2.0*, *postgap_dur=3.0*, *is_smooth_edge=False*, *smooth_width_ratio=0.2*, *smooth_func=<function blur_cos>*, *is_blank_block=True*)

Generate static grating circle stimulus

Stimulus routine presents flashing static grating stimulus inside of a circle centered at *center*. The static gratings are determined by spatial frequencies, orientation, contrast, radius and phase. The routine can generate several different gratings within one presentation by specifying multiple values of the parameters which characterize the stimulus.

Parameters **monitor** : monitor object

contains display monitor information

indicator : indicator object

contains indicator information

coordinate : str from {'degree','linear'}, optional

specifies coordinates, defaults to 'degree'

background : float, optional

color of background. Takes values in [-1,1] where -1 is black and 1 is white

pregap_dur : float, optional

amount of time (in seconds) before the stimulus is presented, defaults to *2.*

postgap_dur : float, optional

amount of time (in seconds) after the stimulus is presented, defaults to *3.*

center : 2-tuple of floats, optional

coordintes for center of the stimulus (altitude, azimuth)

sf_list : n-tuple, optional

list of spatial frequencies in cycles/unit, defaults to *(0.08)*

ori_list : n-tuple, optional

list of directions in degrees, defaults to *(0., 90.)*

con_list : n-tuple, optional

list of contrasts taking values in [0.,1.], defaults to *(0.5)*

radius_list : n-tuple, optional

list of radii of circles, unit defined by *self.coordinate*, defaults to *(10.)*

phase_list : n-tuple, optional

list of phase of gratings in degrees, default (0., 90., 180., 270.)

display_dur : float, optional

duration of each condition in seconds, defaults to *0.25*

midgap_dur, float, optional

duration of gap between conditions, defaults to *0.*

iteration, int, optional

number of times the stimulus is displayed, defaults to *1*

is_smooth_edge : bool

True, smooth circle edge with smooth_width_ratio and smooth_func False, do not smooth edge

smooth_width_ratio : float, should be smaller than 1.

the ratio between smooth band width and radius, circle edge is the middle of smooth band

smooth_func : function object

this function take two inputs: 1) ndarray storing the distance from each pixel to smooth band center; 2) smooth band width. returns smoothed mask with same shape as input ndarray

is_blank_block : bool, optional

if True, a full screen background will be displayed as an additional grating. The frames of this condition will be: (1, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0 or 0.0), the meaning of these numbers can be found in self.frame_config

5.2.7.0.1 Attributes

display_frame_num
midgap_frame_num
postgap_frame_num
pregap_frame_num

5.2.7.0.2 Methods

clear()	
generate_frames()	place holder of function "generate_frames" for each specific stimulus
generate_movie()	place holder of function 'generate_movie' for each specific stimulus
generate_movie_by_index()	compute the stimulus movie to be displayed by index.
set_background(background)	
set_coordinate(coordinate)	
set_indicator(indicator)	
set_monitor(monitor)	
set_postgap_dur(postgap_dur)	
set_pregap_dur(pregap_dur)	

generate_frames()
place holder of function "generate_frames" for each specific stimulus

generate_movie()
place holder of function 'generate_movie' for each specific stimulus

generate_movie_by_index()
compute the stimulus movie to be displayed by index.

5.2.8 StaticImages

class retinotopic_mapping.StimulusRoutines.**StaticImages**(*monitor*, *indicator*, *background=0.0*, *coordinate='degree'*, *img_center=(0.0, 60.0)*, *deg_per_pixel=(0.1, 0.1)*, *display_dur=0.25*, *midgap_dur=0.0*, *iteration=1*, *pregap_dur=2.0*, *postgap_dur=3.0*, *is_blank_block=True*)

Generate static images stimulus

Stimulus routine presents a sequence of static images in a random order. Currently the input image stack should be a tif file. The size of the image should be exactly same as the pixel dimension of downsized monitor pixel resolution. For example if self.monitor.resolution = (1200,1920) and self.monitor.downsample_rate = 10. The shape of input image stack should be n x 120 x 192. Value of the input image stack should be within the range of [-1., 1.]. The values out of this range will be handled by psychopy.visual.ImageStim() function. The reason of this seemingly stringent requirement is that, for visual physiological experiments, the parameters of

visual stimuli should be very well controlled. Any imaging cropping, zooming, transformating etc. will affect luminance, contrast, spatial resolution etc. and produce unexpected effects.

This stimulus routing provides a method to generate such image stacks. StaticImages.wrap_images() takes a list of image files transform them into a desired spherically corrected and luminance normalized image stack into visual degree coordinates and save it as a tif file.

Parameters **monitor** : monitor object

contains display monitor information

indicator : indicator object

contains indicator information

coordinate : str from {'degree','linear'}, optional

specifies coordinates, defaults to 'degree'

background : float, optional

color of background. Takes values in [-1,1] where -1 is black and 1 is white

pregap_dur : float, optional

amount of time (in seconds) before the stimulus is presented, defaults to *2.*

postgap_dur : float, optional

amount of time (in seconds) after the stimulus is presented, defaults to *3.*

img_center : 2-tuple of floats, optional

coordintes for center of the images (altitude, azimuth)

deg_per_pixel: float, or list/tuple of two floats

pixel size in visual degrees of unwrapped image (altitude, azimuth), if float, assume sizes in altitude and azimuth are the same

display_dur : float, optional

duration of each condition in seconds, defaults to *0.25*

midgap_dur : float, optional

duration of gap between conditions, defaults to *0.*

iteration : int, optional

number of times the stimulus is displayed, defaults to *1*

is_blank_block : bool, optional

if True, a full screen background will be displayed as an additional image. index of this image will be -1.

5.2.8.0.1 Attributes

display_frame_num
midgap_frame_num
postgap_frame_num
pregap_frame_num

5.2.8.0.2 Methods

clear()	
generate_frames()	place holder of function "generate_frames" for each specific stimulus
generate_movie()	place holder of function 'generate_movie' for each specific stimulus
generate_movie_by_index()	compute the stimulus movie to be displayed by index.
set_background(background)	
set_coordinate(coordinate)	
set_imgs_from_hdf5(imgs_file_path)	set 3d arrays from a hdf5 file for display.
set_imgs_from_tif(imgs_path_wrapped[, ...])	
set_indicator(indicator)	
set_monitor(monitor)	
set_postgap_dur(postgap_dur)	
set_pregap_dur(pregap_dur)	
wrap_images(work_dir)	look for the 'images_original.

generate_frames()
place holder of function "generate_frames" for each specific stimulus

generate_movie()
place holder of function 'generate_movie' for each specific stimulus

generate_movie_by_index()
compute the stimulus movie to be displayed by index.

set_imgs_from_hdf5(*imgs_file_path*)
set 3d arrays from a hdf5 file for display. Ideally the hdf5 file should be the result from self.wrap_images() method. Only designed to work with wrapped images

Parameters **imgs_file_path** : str

system path ot the hdf5 file. It should have at least one dataset named 'images_wrapped' containing a 3d array of wrapped images to display

wrap_images(*work_dir*)
look for the 'images_original.tif' in the work_dir, load the images, warp and luminance correct images, save wrapping results in an HDF5 file with name "wrapped_images_for_display.hdf5" in the work_dir

5.2.9 StimulusSeparator

class retinotopic_mapping.StimulusRoutines.**StimulusSeparator**(*monitor*, *indicator*, *coordinate='degree'*, *background=0.0*, *indicator_on_frame_num=4*, *indicator_off_frame_num=4*, *cycle_num=10*, *pregap_dur=0.0*, *postgap_dur=0.0*)
a quick flash of indicator to separate different visual stimuli when displayed in the same session

Parameters **monitor** : monitor object

contains display monitor information

indicator : indicator object

contains indicator information

coordinate : str from {'degree','linear'}, optional

specifies coordinates, defaults to 'degree'

background : float, optional

color of background. Takes values in [-1,1] where -1 is black and 1 is white

pregap_dur : float, optional

amount of time (in seconds) before the stimulus is presented, defaults to *2.*

postgap_dur : float, optional

amount of time (in seconds) after the stimulus is presented, defaults to *3.*

indicator_on_frame_num : int

number of frames the indicator is white, should be positive.

indicator_off_frame_num : int

number of frames the indicator is black, should be positive.

cycle_num : int

number of repeat of the indicator flash, should be positive.

5.2.9.0.1 Attributes

`postgap_frame_num`
`pregap_frame_num`

5.2.9.0.2 Methods

`clear`()	
`generate_frames`()	place holder of function "generate_frames" for each specific stimulus
`generate_movie`()	place holder of function 'generate_movie' for each specific stimulus
`generate_movie_by_index`()	place holder of function generate_movie_by_index() for each specific stimulus
`set_background`(background)	
`set_coordinate`(coordinate)	
`set_indicator`(indicator)	
`set_monitor`(monitor)	
`set_postgap_dur`(postgap_dur)	
`set_pregap_dur`(pregap_dur)	

generate_frames()

place holder of function "generate_frames" for each specific stimulus

generate_movie()
place holder of function 'generate_movie' for each specific stimulus

generate_movie_by_index()
place holder of function generate_movie_by_index() for each specific stimulus

5.2.10 CombinedStimuli

class retinotopic_mapping.StimulusRoutines.**CombinedStimuli**(*monitor*, *indicator*, *background=0.0*, *coordinate='degree'*, *pregap_dur=2.0*, *postgap_dur=3.0*)

the stimulus class that can combine different stimuli into one session.

example: >>> import retinotopic_mapping.StimulusRoutines as stim >>> from retinotopic_mapping.MonitorSetup import Monitor, Indicator >>> from retinotopic_mapping.DisplayStimulus import DisplaySequence >>> mon = Monitor(resolution=(1200, 1920), dis=15., mon_width_cm=52., mon_height_cm=32.) >>> ind = Indicator(mon) >>> uc = stim.UniformContrast(mon, ind, duration=10., color=-1.) >>> ss = stim.StimulusSeparator(mon, ind) >>> cs = stim.CombinedStimuli(mon, ind) >>> cs.set_stimuli([ss, uc, ss]) >>> ds = DisplaySequence(log_dir='C:/data') >>> ds.set_stim(cs) >>> ds.trigger_display()

Parameters **monitor** : monitor object

contains display monitor information

indicator : indicator object

contains indicator information

coordinate : str from {'degree','linear'}, optional

specifies coordinates, defaults to 'degree'

background : float, optional

color of background. Takes values in [-1,1] where -1 is black and 1 is white

pregap_dur : float, optional

amount of time (in seconds) before the stimulus is presented, defaults to *2.*

postgap_dur : float, optional

amount of time (in seconds) after the stimulus is presented, defaults to *3.*

5.2.10.0.1 Attributes

postgap_frame_num
pregap_frame_num

5.2.10.0.2 Methods

clear()

Continued on next page

Table 22 – continued from previous page

generate_frames()	place holder of function "generate_frames" for each specific stimulus
generate_movie()	place holder of function 'generate_movie' for each specific stimulus
generate_movie_by_index()	place holder of function generate_movie_by_index() for each specific stimulus
set_background(background)	
set_coordinate(coordinate)	
set_indicator(indicator)	
set_monitor(monitor)	
set_postgap_dur(postgap_dur)	
set_pregap_dur(pregap_dur)	
set_stimuli(stimuli[, static_images_path])	

Parameters

generate_movie_by_index()
place holder of function generate_movie_by_index() for each specific stimulus

set_stimuli(*stimuli*, *static_images_path=None*)

Parameters **stimuli** : list of above stimulus object

static_images_path : str

system path to the hdf5 file storing the wrapped images for display. If there is StaticImages stimulus in the stimuli list, it will try to load images and display

5.2.11 KSstim

class retinotopic_mapping.StimulusRoutines.**KSstim**(*monitor*, *indicator*, *background=0.0*, *coordinate='degree'*, *square_size=25.0*, *square_center=(0, 0)*, *flicker_frame=10*, *sweep_width=20.0*, *step_width=0.15*, *direction='B2U'*, *sweep_frame=1*, *iteration=1*, *pregap_dur=2.0*, *postgap_dur=3.0*)

generate Kalatsky & Stryker stimulus

Kalatsky & Stryker (KS) stimulus routine presents checkerboard gratings that drift against a fixed *background* color.

Parameters **monitor** : monitor object

object storing experimental monitor setup information

indicator : indicator object

object storing photodiode indicator information

background : float, optional

background color of stimulus, takes values in [-1,1]. defaults to *0.*

coordinate : str, optional

coordinate representation, either 'degree' or 'linear', defaults to 'degree'

square_size : float, optional

size of flickering square, defaults to *25.*

square_center: tuple, optional

coordinate of center point, defaults to *(0,0)*

flicker_frame : int, optional

number of frames in one flicker, defaults to *10*

sweep_width : float, optional

width of sweeps measured in units cm or degs if coordinate value is 'linear' or 'degree' respectively. defaults to *20*

step_width : float, optional

width of steps measured in units cm or degs if coordinate value is 'linear' or 'degree' respectively. defaults to *0.15*

direction : {'B2U','U2B','L2R','R2L'}, optional

the direction of sweep movement, defaults to 'B2U'. 'B2U' means stim is presented from the bottom to the top of the screen, whereas 'U2B' is from the top to the bottom. 'L2R' is left to right and 'R2L' is right to left

sweep_frame : int, optional

roughly determines speed of the drifting grating, defaults to *1*

iteration : int, optional

number of times that the stimulus will be repeated, defaults to *1*

pregap_dur : float, optional

number of seconds before stimulus is presented, defaults to *2*

postgap_dur : float, optional

number of seconds after stimulus is presented, defaults to *2*

5.2.11.0.1 Attributes

`postgap_frame_num`
`pregap_frame_num`

5.2.11.0.2 Methods

`clear`()	
`generate_frames`()	function to generate all the frames needed for KS stimulation returnins a list of information of all frames as a list of tuples
`generate_movie`()	Function to Generate Kalatsky & Stryker visual stimulus frame by frame

Continued on next page

Table 24 – continued from previous page

generate_movie_by_index()	place holder of function generate_movie_by_index() for each specific stimulus
generate_squares()	generate checker board squares
generate_sweeps()	generate full screen sweep sequence
plot_squares()	plot checkerboard squares
set_background(background)	
set_coordinate(coordinate)	
set_direction(direction)	
set_indicator(indicator)	
set_monitor(monitor)	
set_postgap_dur(postgap_dur)	
set_pregap_dur(pregap_dur)	
set_sweep_sigma(sweepSigma)	
set_sweep_width(sweep_width)	

generate_frames()
function to generate all the frames needed for KS stimulation returnins a list of information of all frames as a list of tuples

Information contained in each frame:

first element - during stimulus value is equal to 1 and 0 otherwise

second element - square polarity, 1->not reversed; -1->reversed

third element: sweeps, index in sweep table

forth element -

color of indicator synchronized: gap->-1, sweep on -> 1 non-synchronized: alternating between -1 and 1 at defined frequency

for gap frames the second and third elements should be 'None'

generate_movie()
Function to Generate Kalatsky & Stryker visual stimulus frame by frame

generate_movie_by_index()
place holder of function generate_movie_by_index() for each specific stimulus

generate_squares()
generate checker board squares

generate_sweeps()
generate full screen sweep sequence

plot_squares()
plot checkerboard squares

5.2.12 KSstimAllDir

class retinotopic_mapping.StimulusRoutines.**KSstimAllDir**(*monitor*, *indicator*, *coordinate='degree'*, *background=0.0*, *square_size=25*, *square_center=(0, 0)*, *flicker_frame=6*, *sweep_width=20.0*, *step_width=0.15*, *sweep_frame=1*, *iteration=1*, *pregap_dur=2.0*, *postgap_dur=3.0*)

generate Kalatsky & Stryker stimulation in all four direction contiuously

Generalizes the KS stimulus routine so that the drifting gratings can go in all four directions

Parameters **monitor** : monitor object

contains display monitor information

indicator : indicator object

contains indicator information

coordinate : str from {'degree','linear'}, optional

specifies coordinates, defaults to 'degree'

background : float, optional

color of background. Takes values in [-1,1] where -1 is black and 1 is white

square_size : int, optional

size of flickering square, defaults to 25.

square_center : tuple, optional

coordinate of center point of the square, defaults to (0,0)

flicker_frame : int, optional

number of frames per flicker while stimulus is being presented, defaults to *6*

sweep_width : float, optional

width of sweeps. defaults to *20.*

step_width : float, optional

width of steps. defaults to *0.15*.

sweep_frame : int, optional

roughly determines speed of the drifting grating, defaults to *1*

iteration : int, optional

number of times stimulus will be presented, defaults to *1*

pregap_dur : float, optional

number of seconds before stimulus is presented, defaults to *2.*

postgap_dur : float, optional

number of seconds after stimulus is presented, defaults to *3.*

5.2.12.0.1 Methods

generate_movie()	Generate stimulus movie frame by frame

generate_movie()
Generate stimulus movie frame by frame

5.3 retinotopic_mapping.DisplayStimulus

Visual Stimulus codebase implements several classes to display stimulus routines. Can display frame by frame or compress data for certain stimulus routines and display by index. Used to manage information between experimental devices and interact with `StimulusRoutines` module to produce visual display and log data. May also be used to save and export movies of experimental stimulus routines for presentation.

5.3.1 DisplaySequence

5.4 retinotopic_mapping.DisplayLogAnalysis

This module contains the class(es) and methods that perform offline analysis of the log file (the .pkl file) saved by the `DisplayStimulus.DisplaySequence`.

5.4.1 DisplayLogAnalyzer

class retinotopic_mapping.DisplayLogAnalysis.**DisplayLogAnalyzer**(*log_path*)
class to take display_log (.pkl) file, check its integrity and extract stimuli and display organize into stim_dict dictionary, which is a intermediate step to put visual display information into nwb files.

5.4.1.0.1 Attributes

num_frame_tot

5.4.1.0.2 Methods

analyze_photodiode_onsets_combined(pd_onsets_seq)	**Parameters**
analyze_photodiode_onsets_sequential(stim_dict)	Analyze photodiode onsets in a sequential way
check_integrity()	
get_stim_dict()	**Returns**

analyze_photodiode_onsets_combined(*pd_onsets_seq*, *is_dgc_blocked=True*)

Parameters **pd_onsets_seq: list**

product of self.analyze_photodiode_onsets_sequential()

dgc_onset_type : str

type of onset "block" or "cycle"

Returns **pd_onsets_combined** : dict

analyze_photodiode_onsets_sequential(*stim_dict*, *pd_thr=-0.5*)

Analyze photodiode onsets in a sequential way

Parameters **stim_dict: dictionary**

should be the output of self.get_stim_dict()

pd_thr : float

the threshold to detect photodiode onset, the photodiode color was saved in each displayed frame (the last item of frame tuple) as float with range [-1., 1.]. pd_onset is defined as up crossing the pd_thr. retinotopic_mapping.tools.GenericTools.up_crossing() function is used to detect the up crossing. It detects the frame meeting the following criteria: 1) the current frame has photodiode color larger than pd_thr; 2) the previous frame has photodiode color no larger than pd_thr

Returns pd_onsets: list

list of photodiode onsets in sequential manner (in time). Each element in the list is a dictionary representing one photodiode onset. The dictionary has 3 fields:

1. stim_name: str, the name of the stimulus the onset belongs to
2. global_frame_ind: the index of this frame in the total frame displayed
3. **global_pd_onset_ind: the index of this photodiode onset in the total photodiode onsets series** of the stimuli display
4. **str(s)_stim: string that represents the properties of the onset frame. For most frame it is just a** string, for LocallySparseNoise, it is a set of strings with each string representing one probe on the onset frame.

get_stim_dict()

Returns stim_dict: dictionary

the structure of this dictionary should look like this:

{

'000_UniformContrastRetinotopicMapping': { ... 'stim_name' : '000_UniformContrastRetinotopicMapping', 'index_to_display': <index referencing 'frames_unique' field> 'timestamps': <index referencing entire display sequence,

should match hardware vsync signal>

'frames_unique': list of tuple representing unique frames ... },

'001_StimulusSeparatorRetinotopicMapping: { ... 'stim_name' : '000_UniformContrastRetinotopicMapping', 'index_to_display': <index referencing 'frames_unique' field> 'timestamps': <index referencing entire display sequence,

should match hardware vsync signal>

'frames_unique': list of tuple representing unique frames ... },

... }

CHAPTER 6

Indices and tables

- genindex
- search

CHAPTER 7

Indices and tables

- genindex
- search

Bibliography

[a] This can be 1) OpenCV-Python 3.1.0 or later (pip version) or 2) opencv 2.4.11 (conda cloud version from menpo channel) or 3) opencv3 3.2.0 (conda cloud version from menpo channel)

[b] On windows systems, *conda install scikit-image* is recommended. *pip install scikit-image* will sometimes fail. Due to the `skimage.external.tifffile` module. This solution usually fix the problem.

[c] Same as b.

[d] Requires National Instruments DAQmx driver.

Index

I

K

L

M

P

R

S

U

W