Contrast Documentation

Release 1.0.0

Antonio M. Rodriguez and Richard H. Granger

Jun 25, 2020

CONTENTS

Contents

CHAPTER

ONE

CONTRAST

Contrast is an implementation of the generalized contrast function described in:

Reference Paper

Contrast-dependent crowding
(2020, in submission)

A.M.Rodriguez, R.Granger
Dartmouth College

Abstract

Visual clutter affects our ability to see: objects that would be identifiable on their own, may become unrecognizable when presented close together ("crowding") – but the psychophysical characteristics of crowding have resisted simplification. Image properties initially thought to produce crowding have paradoxically yielded unexpected results, e.g., adding flanking objects can ameliorate crowding (Manassi, Sayim et al. 2012, Herzog, Sayim et al. 2015, Pachai, Doerig et al. 2016) The resulting theory revisions have been sufficiently complex and specialized as to make it difficult to discern what principles may underlie the observed phenomena. A generalized formulation of simple visual contrast energy is presented, arising from straightforward analyses of center and surround neurons in the early visual stream. Extant contrast measures, such as RMS contrast, are easily shown to fall out as reduced special cases. The new generalized contrast energy metric surprisingly predicts the principal findings of a broad range of crowding studies. These early crowding phenomena may thus be said to arise predominantly from contrast, or are, at least, severely confounded by contrast effects. (These findings may be distinct from accounts of other, likely downstream, "configural" or "semantic" instances of crowding, suggesting at least two separate forms of crowding that may resist unification.) The new fundamental contrast energy formulation provides a candidate explanatory framework that addresses multiple psychophysical phenomena beyond crowding.

1.1 Install

This program is written in Python 3.7.3. https://www.python.org/

1.1.1 Ensure that the following Python packages are installed:

- numpy numpy (1.18.1)
- scipy scipy (1.4.1)
- Pillow Pillow (5.4.1) https://pillow.readthedocs.io/en/latest/installation.html
- matplotlib matplotlib (3.1.2)
- pandas pandas (0.25.3)
- PsychoPy Psychopy (2020.1.2)

Install the latest version of these libraries:

```
$ pip3 install numpy scipy Pillow matplotlib pandas psychopy
```

1.2 Running the Herzog experiment:

To run Herzog experiment from the paper:

```
$ cd experiments/herzog
$ ./herzog.py
```

Results will be in the report directory:

```
$ ls report
contrast-Herzog-2012-Figure-1a.pdf
contrast-Herzog-2012-Figure-1b.pdf
contrast-Herzog-2012-Figure-1c.pdf
contrast-Herzog-2012-Figure-1d.pdf
"decision-'herzog'-Figure-1a.pdf"
"decision-'herzog'-Figure-1b.pdf"
"decision-'herzog'-Figure-1c.pdf"
"decision-'herzog'-Figure-1d.pdf"
Herzog-2012-Figure-1a.pdf
Herzog-2012-Figure-1b.pdf
Herzog-2012-Figure-1c.pdf
Herzog-2012-Figure-1d.pdf
$
```

To run the Herzog experiment with a new est_max value one can edit the parameter directly in the source code or one can issue the following command with a command-line option:

```
$ cd experiments/herzog
$ ./herzog.py -model est_max=0.1
```

All the parameters are listed at the top of each experiment file. For the Herzog experiment the parameters are in herzog.py:

```
#foreground_color = [-1,-1,-1]
background_color = [-1,-1,-1]
eccentricity = 3.88
herzog_params = Params(
    name              = 'herzog',
    expName           = 'herzog',
```

(continues on next page)

(continued from previous page)

```
    viewing_distance = 75.0,
    monitor          = 'testMonitor',
    logfile          = 'herzog.log',
    window_color     = [-1,-1,-1],
    exp_info         = {u'session': u'001', u'participant': u'default'},
    cwd              = os.getcwd(),
    target_identifier= ('num_flank',0),
    stair      = Params(startVal=0, stepSizes=1, stepType='lin',
                 nReversals=0, nTrials=4, nUp=1, nDown=1,
                 minVal=0, maxVal=7, autoLog=True,
                 originPath=-1, name='staircase_trials'),
    levels     = [
                   Params(exp_num=[1],
                          num_flank=[0,1,2,4,8],
                          jitter=[0],
                          flank_target_height_ratio=[0.5]),
                   Params(exp_num=[2],
                          num_flank=[0,1,2,4,8],
                          jitter=[0],
                          flank_target_height_ratio=[1]),
                   Params(exp_num=[3],
                          num_flank=[0,1,2,4,8],
                          jitter=[0],
                          flank_target_height_ratio=[2]),
                   Params(exp_num=[4],
                          num_flank=[0,1,2,4,8],
                          jitter=[1],
                          flank_target_height_ratio=[0.5])],
    experiment = Params(eccentricity= eccentricity,
                 nTrialReps= 2,
                 nStaircaseTrials= 8),
    stimulus   = Params(eccentricity= eccentricity,
                       jitters=np.array([-0.1,  0.26, -0.87,  0.24,
                                          0.86, -0.34, 0.5 , -0.51])*0.5*(40/60.0),
                       flank_distance=23.33/60.0,
                       target_orientation= 0,
                       line_height= 40/60.0,
                       line_width= 4/60.0,
                       vertical_gap= 4/60.0,
                       offset= 0.0,
                       filename= ['num_flank','jitter','offset','flank_target_height_
↪ratio','target_orientation'],
                       offset_level= 16.66/60.0,
                       offsets= np.array([16.66, 19.04, 21.42, 23.8,
                                          26.18, 28.56, 30.94, 33.32])),
    model      = Params(eccentricities= [eccentricity], # in deg
                       view_size= (600,600), # in pixels
                       view_pos= (eccentricity,0), # center in degrees of visual␣
↪angle
                       est_max= 0.1,
                       upper_limit= 0.85,
                       lower_limit= 0.0))
```

To recreate the stimuli for Herzog experiment (note: various windows will appear while the stimuli are being generated):

```
$ cd experiments/herzog
$ ./herzog.py -genstim
```

1.3 Running the Kahneman experiment:

To run Herzog experiment from the paper:

```
$ cd experiments/herzog
$ ./herzog.py
```

Results will be in the report directory:

```
$ ls report
contrast-Kahneman-2012-Figure-1.pdf
"decision-'kahneman'-Figure-1.pdf"
Kahneman-2012-Figure-1.pdf
$
```

All the parameters are listed at the top of each experiment file. For the Kahneman experiment the parameters are in kahneman.py:

```
foreground_color = [-1,-1,-1]
background_color = [0.1,0.1,0.1]
eccentricity = 0.0
kahneman_params = Params(
    name             = 'kahneman',
    expName          = 'kahneman',
    exp_num          = 1,
    viewing_distance = 2300.0,
    monitor          = 'testMonitor',
    logfile          = 'kahneman.log',
    window_color     = background_color,
    cwd              = os.getcwd(),
    exp_info         = {u'session': u'001', u'participant': u'default'},
    target_identifier= ('flank_distance',-1/60.0),
    levels = Params(
        flank_distance = np.array([-1., 0.06, 0.12, 0.18, 0.24, 0.6, 1.2, 1.8, 2.4, 3.
↪, 5.4 ])/60,
        offset = [0],
        target_orientation = [0,90,180,270]), # degrees from noon orientation
    experiment = Params(eccentricity= eccentricity,
                  nTrialReps= 1),
    stimulus   = Params(
        eccentricity = eccentricity,
        target_size = 0.0548,
        gap_size = 0.01124,
        line_width= 0.014),
    model      = Params(eccentricities= [5], # in deg
                        view_size= (1000,1000), # in pixels
                        view_pos= (eccentricity,0), # center in degrees of visual␣
↪angle
                        est_max= 0.032,
                        upper_limit= 0.85,
                        lower_limit= 0.0))
```

To recreate the stimuli for Kahneman experiment (note: various windows will appear while the stimuli are being generated):

```
$ cd experiments/kahneman
$ ./kahneman.py -genstim
```

1.4 Running the Pachai experiment:

To run Pachai experiment from the paper:

```
$ cd experiments/pachai
$ ./pachai.py
```

Results will be in the report directory:

```
$ ls report
contrast-1-Pachai-Figure-1.pdf
contrast-5-Pachai-Figure-1.pdf
"decision-1-'pachai'-Figure-a.pdf"
"decision-5-'pachai'-Figure-a.pdf"
Pachai-1-Figure-1.pdf
Pachai-5-Figure-1.pdf
"plot-'pachai'-barplot.pdf"
$
```

All the parameters are listed at the top of each experiment file. For the Pachai experiment the parameters are in pachai.py:

```
eccentricity = 10.0
#background_color = [-1,-1,-1]
background_color = [0,0,0]

pachai_params = Params(
    name             = 'pachai',
    expName          = 'pachai',
    exp_num          = 1,
    viewing_distance = 58,
    monitor          = 'testMonitor',
    logfile          = 'pachai.log',
    exp_info         = {u'session': u'001', u'participant': u'default'},
    cwd              = os.getcwd(),
    window_color     = background_color,
    target_identifier= ('flank_distance',-1),
    levels = Params(flank_distance= [-1,0.5,0.9,1.62,2.58,3.9], # degrees of visual␣
↪angle
                    flank_orientation= [45,135,225,315], # degrees from noon␣
↪orientation
                    target_orientation= [0,90,180,270], # degrees from noon␣
↪orientation
                    gap= [0,1],
                    num_flank= [1,5]),
    experiment  = Params(eccentricity= eccentricity,
                        nTrialReps= 1),
    stimulus    = Params(line_width= 0.4,
                        gap_width= 1.2,
```

(continues on next page)

(continued from previous page)

```
                        target_diameter= 2.0,
                        flank_height= 10.0),
    model       = Params(eccentricities= [eccentricity], # in deg
                        view_size= (600,600), # in pixels
                        view_pos= (eccentricity,0), # center in degrees of visual␣
↪angle
                        upper_limit= 0.85,
                        lower_limit= 0.0))
```

To recreate the stimuli for Pachai experiment (note: various windows will appear while the stimuli are being generated):

```
$ cd experiments/pachai
$ ./pachai.py -genstim
```

1.5 Running the Pelli experiment:

To run Pelli experiment from the paper:

```
$ cd experiments/pelli
$ ./pelli.py
```

Results will be in the report directory:

```
$ ls report
"contrast-'LargeLetters'-Figure-1.pdf"
"contrast-'SmallLetters'-Figure-1.pdf"
"decision-'LargeLetters'-Figure-1.pdf"
"decision-'SmallLetters'-Figure-1.pdf"
"'LargeLetters'-Figure-1.pdf"
"'SmallLetters'-Figure-1.pdf"
$
```

All the parameters are listed at the top of each experiment file. For the Herzog experiment the parameters are in herzog.py:

```
background_color = [1,1,1]
eccentricity = [5,10,15,20]
pelli_params = Params(
    name             = 'pelli',
    expName          = 'pelli',
    exp_num          = 1,
    viewing_distance = 22*2.54,
    monitor          = 'testMonitor',
    logfile          = 'pelli.log',
    exp_info         = {u'session': u'001', u'participant': u'default'},
    cwd              = os.getcwd(),
    window_color     = background_color,
    target_identifier= ('flank_distance',-1.0),
    levels = Params(
        flank_distance = np.array([-1.0,0.05,0.1,0.15,0.2,0.3,0.4,0.6]),
        offset = eccentricity),
    experiment = Params(
        nTrialReps= 2),
```

(continues on next page)

(continued from previous page)

```
    stimulus    = Params(
        name = '',
        target_size = 1.0,
        gap_size = 0.4,
        stim = '02',
        line_width= 0.4),
    model       = Params(eccentricities= eccentricity, # in deg
                        view_size= (500,500), # in pixels
                        view_pos= (0,0), # center in degrees of visual angle
                        upper_limit= 0.85,
                        lower_limit= 0.0))
```

Since we are using two different decision function in the Pelli case, one for 'Small Letters' and one for the 'LargeLetters', the target_contrast and est_max for those values are defined in the following code fragment at the bottom of the pelli.py file:

```
if name == 'SmallLetters':
    pelli_params['model']['target_contrast'] = 0.005
    pelli_params['model']['est_max'] = 0.01
else:
    pelli_params['model']['target_contrast'] = 0.012
    pelli_params['model']['est_max'] = 0.025
```

To recreate the stimuli for Pelli experiment (note: various windows will appear while the stimuli are being generated):

```
$ cd experiments/pelli
$ ./pelli.py -genstim
```

CHAPTER

TWO

OVERVIEW

2.1 Layout of files

2.1.1 Contrast directory

The contents of the Contrast directory are listed below:

Name	Description
doc/	Sphinx Documentation directory for generating documentation
experiments/	Location for individual experiments
model/	Model and Library code

2.1.2 Experiments directory

The Contrast/experiments directory is where the all the experiments live.

The contents of the Contrast/experiments directory is listed below:

Commands for running all experiments:

Name	Description
herzog/	Manassi, M., B. Sayim and M. Herzog (2012). Experiment 1.
kahneman/	Flom, M., F. Weymouth and D. Kahneman (1963). Experiment 1.
pachai/	Pachai, M., A. Doerig and M. Herzog (2016).
pelli/	Pelli, D. and K. Tillman (2008). Experiment in Figure 5.

2.1.3 Sample Experiment directory: Herzog

This section describes the files in an individual experiment directory. In this case, the Herzog experiment but the same format will apply to the other experiments as well.

Name	Description
data/	Place where data from the runs is stored
herzog.log	Log file for the run of the experiment
herzog.py	Code for running the herzog experiment
images/	Directory where heatmaps are stored
report/	Location for final figures from running the herzog experiment
report.py	File to generate the report figures, called automatically from herzog.py
stimuli/	Stimuli for the herzog experiment. Created by ./herzog.py -genstim

2.2 Code for generating the Contrast Jacobian

CHAPTER

THREE

CONTRAST API REFERENCE

3.1 Model

Contains Model Class for building Jacobian operators and processing an image.

class Contrast.model.model.**Model**(*eccentricities=[3.88], viewing_distance=29.5, screen_pixel_size=0.282, view_size=(350, 350), view_pos=(0, 0), target_contrast=0.0, est_max=0.01, K=5.0, upper_limit=0.85, lower_limit=0.0, saveimages=False, cwd=", name=", logging=None, **params*)

Class for building Jacobians and processing an image

build_operators(*eccentricities*, *viewing_distance=22.0*, *screen_pixel_size=0.282*, *field_height=100*, *field_width=100*)

NOTE: Ji and Javg as filters should each sum to one

compute_decision(*contrast*, *sigma=1.0*, *compute_error=False*, *est_max=-1.0*, *compute_relative_to_chance=False*, *chance=0.0*, *update=False*)

get_decision_params()

get_op(*sigma=0.01*, *K=1.0*, *field_height=100*, *field_width=100*, *pixel_eccentricity=0*, *viewing_distance=12.0*, *screen_pixel_size=0.282*)

process(*data=None*, *name="*, *save_conv_filename=None*, *target_data=None*)

response(*data*, *name="*, *correct_answer='left'*, *incorrect_answer='right'*, *save_conv_filename=None*, *target_data=None*)

update_decision_params()

Contrast.model.model.**decision_func**(*contrast*, *target_contrast=0*, *decision_sigma=0*, *decision_K=0*, *upper_limit=0*, *lower_limit=0*, *return_all=False*)

Decision function that maps contrast values to a proportion of correct response.

Parameters contrast : a numpy array or single contrast value target_contrast : the contrast value of the target_alone, or mu_t decision_sigma : the standard deviation of the target_alone, or sigma_tau decision_K : the value K_phi, so sigma_phi = K_phi*sigma_tau upper_limit : the upper limit of proportion correct resp e.g. 0.85 lower_limit : the lower limit of proportion correct resp e.g. 0.0 return_all : if True, returns result and foreground and background Jacobians

Returns a numpy array of proportion correct responses, one for each contrast value in the contrast parameter.

3.2 Library

Contains functions that are independent of any specific experiment.

`Contrast.model.library.`**`get_correct_coords`**(*start_x=0*, *viewing_distance=12.0*, *field_height=10*, *field_width=10*, *pixel_width=0.282*, *pixel_height=0.282*, ***config*)

returns the coords in terms of degree of visual angle converts Euclidean to Polar coordinates based on a fixation point, viewing distance, and a window size polar coordinate conversion:

- r = np.sqrt(np.square(x) + np.square(y))
- th = np.arctan2(y,x)

log-polar coordinate conversion based on degrees of visual angle from fixation:

- r = np.rad2deg(np.arctan2(np.sqrt(np.square(x) + np.square(y)),viewing_distance*25.4))

`Contrast.model.library.`**`get_degrees_at_pixels`**(*pixels=10*, *viewing_distance=24.0*, *screen_pixel_size=0.282*)

pixels - if fovea is centered on an image, pixels is half the image width in pixels returns - half the viewing_angle

`Contrast.model.library.`**`get_image_width_in_degrees`**(*image_width=100*, *viewing_distance=24.0*, *screen_pixel_size=0.282*)

image_width is size of entire in pixels returns: degrees to span the entire image

`Contrast.model.library.`**`get_image_width_in_pixels`**(*degrees=1.0*, *viewing_distance=24.0*, *screen_pixel_size=0.282*)

degrees is viewing angle of the entire image returns: num of pixels that span the entire image

`Contrast.model.library.`**`get_pixels_at_degrees`**(*degrees=1.0*, *viewing_distance=24.0*, *screen_pixel_size=0.282*)

degrees - if fovea is centered on an image, degrees is half the viewing angle returns: pixels - if fovea is centered on an image, pixels is half the image width in pixels

`Contrast.model.library.`**`get_sigma_map`**(*start_x=0*, *field_height=100*, *field_width=100*, *viewing_distance=12.0*, *screen_pixel_size=0.282*, *debug=False*)

For each point on the image (image_height x image_width) returns the sigma associated with each point due to the offset from the fovea of the image. The average of all the sigmas may be used as an approximation to the full set of all sigmas. Each sigma is used as the basis for creating the J operator which is the weighting of all the pixels given one pixel as a focal point.

Parameters **`start_x`** – is in degrees of visual angle

Returns an entire field_height x field_width array of sigma values

`Contrast.model.library.`**`get_viewing_distance_to_span_image`**(*image_width=20*, *degree_span=1.0*, *screen_pixel_size=0.282*)

degrees is viewing angle of the entire image image_width is size of entire in pixels

`Contrast.model.library.`**`normalize`**(*data*)

`Contrast.model.library.`**`sorted_ls`**(*path*)

3.3 Library

NewLibrary for Experiment Components Must run with Python 3.7 or greater

class Contrast.model.newlibrary.**Experiment**(*params=None*, *reportobj=None*, *subroutines=[]*)

Contains settings and routines

displayreport()

end()

run(*runsubject=False*, *monitor=None*, *distance=None*, *store_runtime_info=False*, *display_report=True*)

update_params(*args*, *params*)

class Contrast.model.newlibrary.**ExperimentConditions**(*levels=[]*)

class Contrast.model.newlibrary.**ImageComponent**(*image=None*, *start=0*, *stop=1000000*, *mask=None*, *units='deg'*, *pos=(0.0, 0.0)*, *size=None*, *ori=0.0*, *color=(1.0, 1.0, 1.0)*, *colorSpace='rgb'*, *contrast=1.0*, *opacity=1.0*, *depth=0*, *interpolate=False*, *flipHoriz=False*, *flipVert=False*, *texRes=128*, *name=None*, *autoLog=None*, *maskParams=None*)

create(*win*)

class Contrast.model.newlibrary.**Params**(**args*, ***kwargs*)

flatten(*prefix=''*)

class Contrast.model.newlibrary.**PolygonComponent**(*edges=3*, *radius=0.5*, *start=0*, *stop=1000000*, *units='deg'*, *lineWidth=1.5*, *lineColor='white'*, *lineColorSpace='rgb'*, *fillColor=None*, *fillColorSpace='rgb'*, *vertices=((-0.5, 0), (0, 0.5), (0.5, 0))*, *closeShape=True*, *pos=(0, 0)*, *size=1*, *ori=0.0*, *opacity=1.0*, *contrast=1.0*, *depth=0*, *interpolate=True*, *name=None*, *autoLog=None*, *autoDraw=False*)

create(*win*)

class Contrast.model.newlibrary.**RectComponent**(*width=0.5*, *height=0.5*, *start=0*, *stop=1000000*, *autoLog=None*, *units='deg'*, *lineWidth=1.5*, *lineColor='white'*, *lineColorSpace='rgb'*, *fillColor=None*, *fillColorSpace='rgb'*, *pos=(0, 0)*, *ori=0.0*, *opacity=1.0*, *contrast=1.0*, *depth=0*, *interpolate=True*, *name=None*, *autoDraw=False*)

create(*win*)

class Contrast.model.newlibrary.**Response**(*key=''*, *rt=0*, *correct=False*, *prob=0.0*, *contrast=0.0*, *level=None*)

class Contrast.model.newlibrary.**Routine**(*components=[]*, *timeout=10*)

Contains only StimulusComponents Represents a SCREEN of different stimuli Is associated with a filename for saving the screen Is associated with a keyboard response

addComponent(*component*)

create(*win*, *exp_handler=None*)

end()

get_answer(*trialparams={}*, *loopstate={}*)

get_filename(*trialparams={}*, *loopstate={}*)

run(*runmodel=None*, *genstim=False*, *trialparams={}*, *params={}*, *loopstate={}*)

update_levels(*levels*)

update_params(*trialparams={}*, *loopstate={}*)

class Contrast.model.newlibrary.**SoundComponent**

class Contrast.model.newlibrary.**StaircaseTrialRoutine**(*stair_params=[]*, *level_values=None*, *subroutines=[]*, *saveTrials=False*)

Can contain a sequence of either Routines StairCaseTrialRoutines or TrialRoutine

create(*win*, *exp_handler=None*)

end()

run(*runmodel=None*, *genstim=False*, *trialparams={}*, *params={}*, *loopstate={}*)

update_levels(*levels*)

class Contrast.model.newlibrary.**StimulusComponent**(*start=0*, *stop=1000000*, *name=None*, *params={}*)

Implements a stimulus that allows for a human subject response. Contains Sounds, Images, or some other stimulus.

create(*win*)

end()

run(*genstim=False*, *trialparams={}*, *loopstate={}*)

start(*trialparams={}*, *loopstate={}*)

stop()

update(*trialparams={}*, *loopstate={}*)

class Contrast.model.newlibrary.**TrialRoutine**(*conditions=[]*, *nReps=1*, *subroutines=[]*, *saveTrials=True*)

Can contain a sequence of either Routines StairCaseTrialRoutines or TrialRoutine

create(*win*, *exp_handler=None*)

end()

run(*runmodel=None*, *genstim=False*, *trialparams={}*, *params={}*, *loopstate={}*)

update_levels(*levels*)

Contrast.model.newlibrary.**print_df**(*df*)

3.4 Plotting Library

`Contrast.model.plotting.`**`plot_figure`**(*figure, name='Default', caption='Default caption.', experiment_name='', dirname='report'*)

Saves a figure.

3.5 Report

class `Contrast.model.report.`**`Report`**(*filenames=[], only_most_recent=False, contrast_results=0, dirname=''*)

Documentation for Report

`generate`()

`plot_data`(*df, df_err=[], fname='plot', title='title', xlabel='xlabel', ylabel='ylabel', plot_min=0, plot_max=1, index_marker_shape=['o', 'v'], columns_marker_color=['m', 'g'], columns_title='Column_title', linestyle='–', scale_plot=True, x_scale_factor=1.0, x_scale_addition=0.0, plot_on_number_line=False*)

`plot_data2`(*df, fname='plot', experiment_name='pelli', exp_name='LargeLetters', title='title', xlabel='xlabel', ylabel='ylabel', plot_min=0, plot_max=100, index_marker_shape=['o', 'v'], columns_marker_color=['m', 'g'], columns_title='Column_title', linestyle='–', scale_plot=True, show_target_as_dash=True, x_scale_factor=1.0, x_scale_addition=0.0, target_value=None, **params*)

`plot_data_as_barplot`(*df, fname='plot', title='title', xlabel='xlabel', ylabel='ylabel', target_value=1*)

`plot_data_as_barplot2`(*df, fname='plot', experiment_name='pelli', exp_name='LargeLetters', title='title', xlabel='xlabel', ylabel='ylabel', columns_labels=None, plot_bar_min=0, plot_bar_max=40*)

`plot_decision_func`(*df, df_contrast=None, target_contrast=0.027, sigma=0.05, K=5.0, zoomed=False, fname='plot', experiment_name='pelli', exp_name='LargeLetters', title='title', xlabel='xlabel', ylabel='ylabel', index_marker_shape=['o', 'v'], columns_marker_color=['m', 'g'], columns_title='Column_title', scale_range=[0, 1], upper_limit=0.85, lower_limit=0.0, x_scale_factor=1.0, show_target_as_dash=True, compute_relative_to_chance=False, chance=0.0, compute_error=False, plot_columns_labels=['0', '1', '2', '3', '4'], show_legend=False, decision_prob_label='response', est_max=0, transpose_df=True, use_target_contrast=False, legend_alpha=0.7, legend_loc=8, target_value=False*)

`Contrast.model.report.`**`fit`**(*df*)

`Contrast.model.report.`**`main`**(*reportclass=<class 'Contrast.model.report.Report'>, filenames=None, only_most_recent=False, dirname=''*)

`Contrast.model.report.`**`sorted_ls`**(*path*)

3.6 Stimulus Library

`Contrast.model.stimulus.`**`save_image`**(*data, filename, xlabel='pixels', ylabel='pixels', title='title', overlay=None, reverse_overlay=False, valmax=False*)

PYTHON MODULE INDEX

C

INDEX

A

B

C

D

E

F

G

I

M

N

P

R

S

T

U