Histonets Computer Vision Documentation

Release 0.1.0

Javier de la Rosa

May 18, 2018

Contents

1 Histonets Computer Vision **3**
- 1.1 Features . . . 3
- 1.2 Commands . . . 3
- 1.3 Credits . . . 13

2 Installation **15**
- 2.1 Stable release . . . 15
- 2.2 From sources . . . 15

3 Usage **17**

4 histonets_cv **19**
- 4.1 histonets_cv package . . . 19

5 Contributing **23**
- 5.1 Types of Contributions . . . 23
- 5.2 Get Started! . . . 24
- 5.3 Pull Request Guidelines . . . 25
- 5.4 Tips . . . 25

6 History **27**
- 6.1 0.1.0 (2016-11-17) . . . 27

7 Indices and tables **29**

Python Module Index **31**

Contents:

CHAPTER 1

Histonets Computer Vision

Computer vision part of the Histonets project

- Free software: Apache Software License 2.0
- Documentation: https://histonets.readthedocs.io.

1.1 Features

Usage: histonets [OPTIONS] COMMAND [ARGS]…

Histonets computer vision application for image processing

Options:

--rst	Show help in ReST format.
--version	Show the version and exit.
--help	Show this message and exit.

1.2 Commands

1.2.1 binarize

Usage: histonets binarize [OPTIONS] [IMAGE]

Binarize IMAGE using a thresholding method.

Example:

```
histonets binarize -m otsu file://...
```

- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-m, –method [sauvola|isodata|otsu|li] Thresholding method to obtain the binary image. For reference, see http://scikit-imag e.org/docs/dev/auto_examples/xx_applications /plot_thresholding.html. Defaults to 'li'.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.2 blobs

Usage: histonets blobs [OPTIONS] [IMAGE]

Binarize using threshold and remove white blobs of contiguous pixels of size between min and max from IMAGE, turning them into black.

Example:

```
histonets blobs -max 100 -c 8 file://...
```

- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-min, --minimum-area INTEGER Minimum area in pixels of the white blobs to detect. Defaults to 0.

-max, --maximum-area INTEGER Maximum area in pixels of the white blobs to detect. Defaults to 9223372036854775807.

-th, –threshold INTEGER RANGE Threshold to binarize before detecting blobs. Ranges from 0 to 255. Defaults to 128.

-c, –connectivity [4|8|16] Connectivity method to consider blobs boundaries. It can take adjacent pixels in a 4 pixels cross neighborhood (top, right, bottom, left), 8 pixels (all around), or 16 pixels (anti-aliased). Defaults to 4 neighbors.

-m, --mask Returns a black and white mask instead.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.3 brightness

Usage: histonets brightness [OPTIONS] VALUE [IMAGE]

Adjust brightness of IMAGE.

- VALUE ranges from 0 to 200.
- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.4 clean

Usage: histonets clean [OPTIONS] [IMAGE]

Clean IMAGE automatically with sane defaults and allows for parameter fine tunning.

- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-bv, –background-value INTEGER RANGE Threshold value to consider a pixel background. Ranges from 0 to 100. Defaults to 25.

-bs, –background-saturation INTEGER RANGE Threshold saturation to consider a pixel background. Ranges from 0 to 100. Defaults to 20.

-c, –colors INTEGER RANGE Number of output colors. Ranges from 2 to

128. Defaults to 8.

-f, –sample-fraction INTEGER RANGE Percentage of pixels to sample. Ranges from 0 to 100. Defaults to 5.

-w, --white-background Make background white.

-s, –saturate / -ns, –no-saturate Saturate colors (default).

-p, --palette TEXT Local file, URL, or JSON string representing a palette of colors encoded as lists of RGB components or hexadecimal strings preceded by the hash character (#). Ex: '["#fa4345", "[123, 9, 108]", [1, 2, 3]]'. If a palette is passed in, colors are ignored.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.5 contrast

Usage: histonets contrast [OPTIONS] VALUE [IMAGE]

Adjust contrast of IMAGE.

- VALUE ranges from 0 to 200.
- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.6 denoise

Usage: histonets denoise [OPTIONS] VALUE [IMAGE]

Denoise IMAGE.

- VALUE ranges from 0 to 100.
- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.7 dilate

Usage: histonets dilate [OPTIONS] [IMAGE]

Thicken (dilate) IMAGE using dilation as the radius for the kernel to use. The number of times the dilation process is applied can be changed via the passes parameter (defautls to 1). If the image is not lack and white, it will be binarized using a binarization-method, which by default it's Li's algorithm (see the binarize command). The black and white image can also be thickened (dilated) by adjusting the dilation parameter before extracting the skeleton image.

Example:

```
histonets dilate -d 6 -p 1 -b otsu file://...
```

- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-d, –dilation INTEGER RANGE Dilation radius to thicken the binarized image prior to perform skeletonization. Ranges from 0 to 100. Defaults to 1.

-p, –passes INTEGER RANGE Number of times the dilation is applied. Ranges from 0 to 100. Defaults to 1.

-b, –binarization-method [sauvola|isodata|otsu|li] Thresholding method to obtain the binary image. For reference, see http://scikit-imag e.org/docs/dev/auto_examples/xx_applications /plot_thresholding.html. Defaults to 'li'.

-i, --invert Invert the black and white colors of the binary image prior to dilation.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.8 download

Usage: histonets download [OPTIONS] [IMAGE]

Download IMAGE.

- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.9 enhance

Usage: histonets enhance [OPTIONS] [IMAGE]

Clean IMAGE automatically with sane defaults.

- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-p, --palette TEXT Local file, URL, or JSON string representing a palette of colors encoded as lists of RGB components or hexadecimal strings preceded by the hash character (#). Ex: '[“#fa4345”, “[123, 9, 108]”, [1, 2, 3]]'.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.10 equalize

Usage: histonets equalize [OPTIONS] VALUE [IMAGE]

Histogram equalization on IMAGE.

- VALUE ranges from 0 to 100.
- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.11 graph

Usage: histonets graph [OPTIONS] REGIONS [IMAGE]

Build a undirected graph using the center points of REGIONS as nodes and the paths in the binary grid expressed in IMAGE as edges.

Example:

```
histonets graph '[[[50,50],[120,50]],[[120, 82],[50,82]]]' -sm vw file://
```

- **REGIONS is a path to a local (file://) or remote (http://, https://) JSON** file representing a list of bounding boxes expressed as two [x, y] coordinates points in pixels with regards to IMAGE, one for the top-left corner and a second for the bottom-left one. For example, '[[[50, 50], [120, 50]], [[120, 82], [50, 82]]]' is a list that contains two regions.
- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-pm, –pathfinding-method [floyd-warshall|astar] Specify the pathfinding algorithm to create edges between the matched templates. Available algorithms are 'floyd-warshall' for Floyd-Warshall all shortest paths algorithm, and 'astar' for grid pathfinding A*. Defaults to 'floyd-warshall'.

-sm, –simplification-method [rdp|vw] Specify the line simplification algorithm to reduce the number of pixels in each path. Available algorithms are 'rdp' for the Ramer–Douglas–Peucker's algorithm, and 'vw' for Visvalingam–Whyatt's algorithm. Defaults to 'vw'.

-st, –simplification-tolerance INTEGER RANGE Exponent of the inverse simplification method tolerance, e.g., 3 involves a tolerance of 10^(-3)). Ranges from 0 to 10. Defaults to 0.

-f, –format [edgelist|gexf|gml|graphml|nodelink] Format to save the graph in. All formats are derived from NetworkX's "Reading and Writing graphs": http://networkx.readthedocs.io/en/s table/reference/readwrite.html. Defaults to 'graphml'

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.12 histogram

Usage: histonets histogram [OPTIONS] [IMAGE]

Extract the histogram of IMAGE in a JSON string representing a dictionary with colors as keys and the count (pixels) of those colors as values. Colors can be given as a list of its RGB components (default), or in hexadecimal format preceded by the hash character (#).

Example:

```
histonets histogram -c hex file://...
```

- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-m, –mode [rgb|hex] Color code to represent the colors in the histogram. The option 'rgb' returns colors as lists of the R, G, and B components that range from 0 to 255. If set to 'hex', an hexadecimal representation will be used. Defaults to 'rgb'.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.13 match

Usage: histonets match [OPTIONS] TEMPLATES… [IMAGE]

Look for TEMPLATES in IMAGE and return the bounding boxes of the matches. Options may be provided after each TEMPLATE.

Example:

```
histonets match http://foo.bar/tmpl1 -th 50 http://foo.bar/tmpl2 -th 95
```

- TEMPLATE is a path to a local (file://) or remote (http://, https://) image file of the template to look for.
- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-th, –threshold INTEGER RANGE Threshold to match TEMPLATE to IMAGE. Ranges from 0 to 100. Defaults to 80.

-f, –flip [horizontal|h|vertical|v|both|b|all|a] Whether also match TEMPLATE flipped horizontally. vertically, or both. Defaults to not flipping.

-e, --exclude-regions TEXT JSON list of polygons expressed as [x, y] points to specify regions to cut out when matching. For example, [[[50,50],[120,50],[120,82],[50,82]]] is a list that contains one single polygon.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.14 palette

Usage: histonets palette [OPTIONS] [HISTOGRAM]

Extract a palette of colors from HISTOGRAM.

- HISTOGRAM path to local file, URL, or JSON string representing a dictionary with colors as keys and the count (pixels) of those colors as values. Colors can be given as a list of its RGB components, or in hexadecimal format preceded by the hash character (#).

  Example:

  ```
  histonets palette '{"#fa4345": 3829, "[123, 9, 108]": 982}'
  ```

- HISTOGRAM path to a local (file://) or remote (http://, https://) JSON file representing histogram. A JSON string can also be piped as input

Options:

-c, –colors INTEGER RANGE Number of output colors. Ranges from 2 to

128. Defaults to 8.

-m, –method [auto|kmeans|median|linear|max|octree] Method for computing the palette. 'auto' runs an optimized K-Means algorithm by sampling the histogram and detecting the background color first; 'kmeans' performs a clusterization of the existing colors using the K-Means algorithm; 'median' refers to the median cut algorithm; 'max' runs a maximum coverage process (also aliased as 'linear'); and 'octree' executes a fast octree quantization algorithm. Defaults to 'auto'.

-f, –sample-fraction INTEGER RANGE Percentage of pixels to sample. Ranges from 0 to 100. Defaults to 5.

-bv, –background-value INTEGER RANGE Threshold value to consider a pixel background. Ranges from 0 to 100. Defaults to 25.

-bs, –background-saturation INTEGER RANGE Threshold saturation to consider a pixel background. Ranges from 0 to 100. Defaults to 20.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.15 pipeline

Usage: histonets pipeline [OPTIONS] ACTIONS [IMAGE]

Allow chaining a series of actions to be applied to IMAGE. Output will depend on the last action applied.

- ACTIONS is a JSON list of dictionaries containing each an 'action' key specifying the action to apply, a 'arguments' key which is a list of arguments, and a 'options' key with a dictionary to set the options for the corresponding action.

  Example:

```
histonets pipeline '[{"action": "contrast", "options": {"value": 50}}]'
```

- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.16 posterize

Usage: histonets posterize [OPTIONS] [COLORS] [IMAGE]

Posterize IMAGE by reducing its number of colors.

- COLORS, the number of colors of the output image, ranges from 0 to 64.
- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-m, –method [kmeans|median|linear|max|octree] Method for computing the palette. ‘kmeans’ performs a clusterization of the existing colors using the K-Means algorithm; ‘median’ refers to the median cut algorithm; ‘max’ runs a maximum coverage process (also aliased as ‘linear’); and ‘octree’ executes a fast octree quantization algorithm. Defaults to ‘kmeans’.

-p, --palette TEXT Local file, URL, or JSON string representing a palette of colors encoded as lists of RGB components or hexadecimal strings preceded by the hash character (#). Ex: ‘[“#fa4345”, “[123, 9, 108]”, [1, 2, 3]]’. If a palette is passed in, colors are ignored.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.17 ridges

Usage: histonets ridges [OPTIONS] [IMAGE]

Remove ridges from IMAGE, turning them into black.

Example:

```
histonets ridges --width 6 file://...
```

- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-w, –width INTEGER RANGE Width in pixels of the ridges to detect. Ranges from 1 to 100. Defaults to 6.

-th, –threshold INTEGER RANGE Threshold to binarize detected ridges. Ranges from 0 to 255. Defaults to 128.

-d, –dilation INTEGER RANGE Dilation radius to thicken the mask of detected ridges. Ranges from 0 to 100. Defaults to 1.

-m, --mask Returns a black and white mask instead.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.18 select

Usage: histonets select [OPTIONS] COLORS… [IMAGE]

Select COLORS in IMAGE, turning the rest into black.

Example:

```
histonets select "[225, 47, 90]" "#8ad70e" -t 80  file://...
```

- **COLOR is a JSON string representing a color as a list of** its RGB components or a hexadecimal string starting with #.
- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-t, –tolerance INTEGER RANGE Tolerance to match COLOR in IMAGE. Ranges from 0 to 100. Defaults to 0 (exact COLOR).

-m, --mask Returns a black and white mask instead.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.19 skeletonize

Usage: histonets skeletonize [OPTIONS] [IMAGE]

Extract the morphological skeleton of IMAGE. If the image is not black and white, it will be binarized using a binarization-method, which by default it's Li's algorithm (see the binarize command). The black and white image can also be thickened (dilated) by adjusting the dilation parameter before extracting the skeleton image.

Example:

```
histonets skeletonize -m thin -d 0 -b otsu file://...
```

- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-m, –method [3d|combined|medial|regular|thin] Method to extract the topological skeleton of IMAGE. For reference, see http://scikit-i mage.org/docs/dev/auto_examples/xx_applicati ons/plot_thresholding.html. Defaults to a 'combined' approach of '3d', 'medial', and 'regular'.

-d, –dilation INTEGER RANGE Dilation radius to thicken the binarized image prior to perform skeletonization. Ranges from 0 to 100. Defaults to 6.

-b, –binarization-method [sauvola|isodata|otsu|li] Thresholding method to obtain the binary image. For reference, see http://scikit-imag e.org/docs/dev/auto_examples/xx_applications /plot_thresholding.html. Defaults to 'li'.

-i, --invert Invert the black and white colors of the binary image prior to skeletonization.

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.2.20 smooth

Usage: histonets smooth [OPTIONS] VALUE [IMAGE]

Smooth IMAGE using bilateral filter.

- VALUE ranges from 0 to 100.
- IMAGE path to a local (file://) or remote (http://, https://) image file. A Base64 string can also be piped as input image.

Options:

-o, --output FILENAME File name to save the output. For images, if the file extension is different than IMAGE, a conversion is made. When not given, standard output is used and images are serialized using Base64; and to JSON otherwise.

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Histonets Computer Vision, run this command in your terminal:

```
$ pip install histonets
```

This is the preferred method to install Histonets Computer Vision, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Histonets Computer Vision can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/versae/histonets
```

Or download the tarball:

```
$ curl  -OL https://github.com/versae/histonets/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use Histonets Computer Vision in a project:

```
import histonets
```

CHAPTER 4

histonets_cv

4.1 histonets_cv package

4.1.1 Submodules

4.1.2 histonets_cv.api module

`histonets_cv.api.`**`histogram_image`**(*image*, *method='rgb'*)
  Calculate the color histogram of image (colors and their counts)

`histonets_cv.api.`**`histogram_palette`**(*histogram*, *n_colors=8*, *method='auto'*, *sample_fraction=5*, *background_value=25*, *background_saturation=20*)
  Return a palette of at most n_colors unique colors extracted after sampling histogram by sample_fraction.

4.1.3 histonets_cv.cli module

4.1.4 histonets_cv.utils module

`class` `histonets_cv.utils.`**`Choice`**(*choices*)
  Bases: `click.types.Choice`

  Fix to click.Choice to be able to use integer choices

  `get_metavar`(*param*)
    Returns the metavar default for this param if it provides one.

`class` `histonets_cv.utils.`**`Image`**(*content=None*, *image=None*)
  Bases: `object`

  Proxy class to handle image input in the commands

  `format`

classmethod get_images(*values*)
Helper to process local, remote, and base64 piped images as input, and return Image objects

image

class histonets_cv.utils.**JSONNumpyEncoder**(*skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)
Bases: `json.encoder.JSONEncoder`

Enable serialization of basic Numpy arrays

default(*obj*)
Implement this method in a subclass such that it returns a serializable object for `o`, or calls the base implementation (to raise a `TypeError`).

For example, to support arbitrary iterators, you could implement default like this:

```python
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

class histonets_cv.utils.**JSONStream**(*mode='r'*)
Bases: *histonets_cv.utils.Stream*

JSON Stream Click option type to handle and decode JSON input and files coming (compressed or not) from the Internet (http:// and https://) or locally (file://, absolute, or relative paths).

convert(*param=None*, *ctx=None*, *value=None*)
Converts the value. This is not invoked for values that are *None* (the missing value).

class histonets_cv.utils.**Stream**(*mode='r'*)
Bases: `click.types.ParamType`

Click option type for http/https/file inputs

Based on https://github.com/moshe/click-stream

SUPPORTED_SCHEMES = ('http', 'https', 'file')

convert(*param=None*, *ctx=None*, *value=None*)
Converts the value. This is not invoked for values that are *None* (the missing value).

name = 'stream'

histonets_cv.utils.**argfirst2D**(*arr*, *item*)
Return the index of the first element of the 2D array arr matching the row item, or None if not found.

histonets_cv.utils.**astar**(*grid*, *start*, *end*)
Run A* algorithm from start to end to find a path in grid. It uses squared Euclidean distance as the distance method and the cost estimate heuristic, and it uses the Von Neumann method to assess the 8-neighbors. Returns a predecessors dictionary from which a path can be built.

histonets_cv.utils.**convert**(*image*)
Convert a scikit-image binary image matrix to OpenCV

`histonets_cv.utils.`**`edges_to_graph`**(*edges*, *fmt=None*)
: Build a graph based on a list of edges and serialize it to format. Each edge is a dictionary with at least keys defined for source_key and target_key, expressing the source and the target of the edge, respectively. The graph is built and serialized using NetworkX, therefore only a subset of its formats are available: 'edgelist', 'gexf', 'gml', 'graphml', 'nodelink'. See http://networkx.readthedocs.io/en/stable/reference/readwrite.html for more information.

`histonets_cv.utils.`**`get_color_histogram`**(**args*, ***kwargs*)
: Calculate the color histogram of image (colors and their counts)

`histonets_cv.utils.`**`get_images`**(*ctx*, *param*, *value*)
: Callback to retrieve images by either their local path or URL

`histonets_cv.utils.`**`get_inner_paths`**(*grid*, *regions*)
: Create 1 pixel width paths connecting the loose ends surrounding the regions to their center. Each region is defined by its top-left and bottom-right corners points expressed in [x, y] coordinates. Grid must be a black and white image

`histonets_cv.utils.`**`get_mask_polygons`**(*polygons*, *height*, *width*)
: Turn a list of polygons into a mask image of height by width. Each polygon is expressed as a list of [x, y] points.

`histonets_cv.utils.`**`get_palette`**(**args*, ***kwargs*)
: Calculate a palette of n_colors from RGB values from an array of colors. Parameters background_value and background_saturation are ignored for methods other than 'auto'. When method='auto', the first palette entry is always the background color; the rest are determined from foreground pixels by running K-Means clustering. Returns the palette.

`histonets_cv.utils.`**`get_quantize_method`**(*method*)
: Transform a string ('median', 'octree', 'linear', 'max') to the corresponding PIL quantize method constant

`histonets_cv.utils.`**`get_shortest_paths`**(*grid*, *look_for*)
: Traverse the grid, where 0's represent holes and 1's paths, and return the paths to get from sources to targets, expressed in look_for in the form of ((start1, end1), (start2, end2)), where each 'start' and 'end' are coordinates of the grid in the form [x, y] pairs. It uses the Floyd-Warshall algorithm to find first all shortest paths and then returns only those in look_for

`histonets_cv.utils.`**`get_shortest_paths_astar`**(*grid*, *look_for*)
: Traverse the grid, where 0's represent holes and 1's paths, and return the paths to get from sources to targets, expressed in look_for in the form of ((start1, end1), (start2, end2)), where each 'start' and 'end' are coordinates of the grid in the form [x, y] pairs. It uses the A* algorithm and it only computes the paths in the look_for.

`histonets_cv.utils.`**`grid_to_adjacency_matrix`**(*grid*, *neighborhood=8*)
: Convert a boolean grid where 0's express holes and 1's connected pixel into a sparse adjacency matrix representing the grid-graph. Neighborhood for each pixel is calculated from its 4 or 8 more immediate surrounding neighbors (defaults to 8).

`histonets_cv.utils.`**`image_as_array`**(*f*)
: Decorator to handle image as Image and as numpy array

`histonets_cv.utils.`**`io_handler`**(*input=None*, **args*, ***kwargs*)
: Decorator to handle the 'input' argument and the 'output' option. If input is other than 'image', it is considered to be a JSON file or URL. Defaults to 'image'.

`histonets_cv.utils.`**`kmeans`**(*X*, *n_clusters*, ***kwargs*)
: Classify vectors in X using K-Means algorithm with n_clusters. Arguments in kwargs are passed to scikit-learn MiniBatchKMeans. Returns a tuple of cluster centers and predicted labels.

`histonets_cv.utils.`**`local_decode`**(*value*)
: Decode bytes into a string by using the system preferred encoding. Defaults to utf8 otherwise.

histonets_cv.utils.**local_encode**(*value*)
Encode a string to bytes by using the system preferred encoding. Defaults to utf8 otherwise.

histonets_cv.utils.**match_template_mask**(**args*, ***kwargs*)
Match template against image applying mask to template using method. Method can be either of (None, 'laplacian', 'sobel', 'scharr', 'prewitt', 'roberts', 'canny'). Returns locations to look for max values.

histonets_cv.utils.**output_as_mask**(*f*)
Decorator to add a return_mask option when image and mask are being returned

histonets_cv.utils.**pair_options_to_argument**(*argument*, *options*, *args=None*, *args_slice=None*)
Enforces pairing of options to an argument. Only commands with one argument with nargs=-1 are supported. Not paired options do still work.

Options is a dictionary with the option name as key and the default value as value. A slice to specify where in the arguments the argument and the options are found can be used. By default it will ignore first and last.

Example:

```
@click.command()
@click.argument('arg', nargs=-1, required=True)
@click.option('-o', '--option', multiple=True)
@pair_options_to_argument('arg', {'option': 0})
def command(arg, option):
    pass
```

histonets_cv.utils.**parse_colors**(*ctx*, *param*, *value*)
Callback to parse color values from a JSON list or hexadecimal string to a RGB tuple.

histonets_cv.utils.**parse_histogram**(*histogram*)
Parse a dictionary or JSON string representing a histogram of colors by parsing the keys that codify colors into lists of RGB components and the values to integer numbers

histonets_cv.utils.**parse_jsons**(*ctx*, *param*, *value*)
Callback to load a list JSON strings as objects

histonets_cv.utils.**parse_palette**(*ctx*, *param*, *value*)
Callback to turn a JSON representing a palette of colors in hexadecimal or by its RGB components, into a list of all RGB components

histonets_cv.utils.**parse_pipeline_json**(*ctx*, *param*, *value*)
Parse the actions JSON used mainly in the pipeline command

histonets_cv.utils.**sample_histogram**(*histogram*, *sample_fraction=0.05*)
Sample a sample_fraction of colors from histogram

histonets_cv.utils.**serialize_json**(*obj*)
Serializes object, containing Numpy basic arrays and types, to JSON

histonets_cv.utils.**unique**(*array*, *axis=None*)
Like np.unique but preserving order of first apparition

4.1.5 Module contents

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/sul-cidr/histonets-cv/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

Histonets Computer Vision could always use more documentation, whether as part of the official Histonets Computer Vision docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/sul-cidr/histonets-cv/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *histonets-cv* for local development.

1. Fork the *histonets-cv* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/histonets-cv.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv histonets
   $ cd histonets/
   $ python setup.py develop
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

   ```
   $ flake8 histonets tests
   $ python setup.py test or py.test
   $ tox
   ```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5. Check https://travis-ci.org/sul-cidr/histonets-cv/pull_requests and make sure that the tests pass for the supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_histonets
```

CHAPTER 6

History

6.1 0.1.0 (2016-11-17)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

histonets_cv, 22
histonets_cv.api, 19
histonets_cv.cli, 19
histonets_cv.utils, 19

Index

A

argfirst2D() (in module histonets_cv.utils), 20
astar() (in module histonets_cv.utils), 20

C

Choice (class in histonets_cv.utils), 19
convert() (histonets_cv.utils.JSONStream method), 20
convert() (histonets_cv.utils.Stream method), 20
convert() (in module histonets_cv.utils), 20

D

default() (histonets_cv.utils.JSONNumpyEncoder method), 20

E

edges_to_graph() (in module histonets_cv.utils), 20

F

format (histonets_cv.utils.Image attribute), 19

G

get_color_histogram() (in module histonets_cv.utils), 21
get_images() (histonets_cv.utils.Image class method), 19
get_images() (in module histonets_cv.utils), 21
get_inner_paths() (in module histonets_cv.utils), 21
get_mask_polygons() (in module histonets_cv.utils), 21
get_metavar() (histonets_cv.utils.Choice method), 19
get_palette() (in module histonets_cv.utils), 21
get_quantize_method() (in module histonets_cv.utils), 21
get_shortest_paths() (in module histonets_cv.utils), 21
get_shortest_paths_astar() (in module histonets_cv.utils), 21
grid_to_adjacency_matrix() (in module histonets_cv.utils), 21

H

histogram_image() (in module histonets_cv.api), 19
histogram_palette() (in module histonets_cv.api), 19
histonets_cv (module), 22
histonets_cv.api (module), 19
histonets_cv.cli (module), 19
histonets_cv.utils (module), 19

I

Image (class in histonets_cv.utils), 19
image (histonets_cv.utils.Image attribute), 20
image_as_array() (in module histonets_cv.utils), 21
io_handler() (in module histonets_cv.utils), 21

J

JSONNumpyEncoder (class in histonets_cv.utils), 20
JSONStream (class in histonets_cv.utils), 20

K

kmeans() (in module histonets_cv.utils), 21

L

local_decode() (in module histonets_cv.utils), 21
local_encode() (in module histonets_cv.utils), 21

M

match_template_mask() (in module histonets_cv.utils), 22

N

name (histonets_cv.utils.Stream attribute), 20

O

output_as_mask() (in module histonets_cv.utils), 22

P

pair_options_to_argument() (in module histonets_cv.utils), 22
parse_colors() (in module histonets_cv.utils), 22
parse_histogram() (in module histonets_cv.utils), 22
parse_jsons() (in module histonets_cv.utils), 22
parse_palette() (in module histonets_cv.utils), 22
parse_pipeline_json() (in module histonets_cv.utils), 22

S

sample_histogram() (in module histonets_cv.utils), 22
serialize_json() (in module histonets_cv.utils), 22
Stream (class in histonets_cv.utils), 20
SUPPORTED_SCHEMES (histonets_cv.utils.Stream attribute), 20

U

unique() (in module histonets_cv.utils), 22