
imagecolor Documentation

Release 2.0.0rc1

Rhys Hansen

Feb 11, 2018

Contents:

1	imagecolor - Extract colors from images	1
1.1	Important notes	1
1.2	Installation	1
1.3	Usage examples	2
1.4	Development	2
2	License	5
2.1	The MIT License	5
3	imagecolor reference	7
3.1	imagecolor package	7
4	Indices and tables	15
	Python Module Index	17

imagecolor - Extract colors from images

imagecolor is a python module for averaging images using [pillow](#). When processing of multiple images it uses [concurrent.futures](#) for multiprocessing.

1.1 Important notes

1.1.1 Warnings

Warning: imagecolor only supports python3.6 currently.

1.1.2 Notes

Note: imagecolor is only tested on macOS and Linux currently.

Note: imagecolor only works on 3 channel RGB images.

1.2 Installation

1.2.1 Basic Installation

Install imagecolor with **pip**:

```
$ pip install imagecolor
```

Depending on your platform you might need to install the required [dependencies](#) for pillow before pillow (and imagecolor) will install fully.

1.3 Usage examples

To use imagecolor import it with

```
import imagecolor
```

1.3.1 average an image

Average a single image file to a dict containing name, red, green, & blue

```
imagecolor.file_average(image)
```

If you are not interested in the file name you can use `core_average()` instead.

```
imagecolor.core_average(image)
```

1.3.2 average all images in a directory

Averages all images in a directory to a list of dicts containing name, red, green, & blue

```
imagecolor.directory_average(directory)
```

1.3.3 average a directory

Averages all images in a directory to a dict containing name, red, green, & blue

```
imagecolor.single_directory_average(directory)
```

1.3.4 average nested directories

Uses `single_directory_average` to average the directory and all subdirectories containing images to a list of dicts containing name, red, green, & blue

```
imagecolor.nested_directory_average(directory)
```

For more details read the full module [reference](#)

1.4 Development

1.4.1 Development Installation

image color uses `pipenv` to manage development dependencies.

Install development dependencies with **pipenv**:

```
$ pipenv install --dev
```

1.4.2 make commands

- `all`: calls `release` and `html`
- `release` checks the code with `test` and then calls `source-dist` and `wheel-dist`
- `source-dist` builds a source distribution
- `wheel-dist` builds a wheel distribution
- `lint` lints imagecolor with `pylint`, `pycodestyle`, `pydocstyle`
- `test` lints the code and then runs `pytest`
- `html` builds html docs with `Sphinx`
- `clean` cleans the build and dist directories

2.1 The MIT License

Copyright 2017-2018 Rhys Hansen (Tathorack)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

3.1 imagecolor package

3.1.1 Module contents

imagecolor initialization.

`imagecolor.core_average` (*image*, *downsample=True*, *max_size=100*, *alpha_threshold=245*)

Average a single image.

Averages a single image from a file or file-like object. By default downsamples images that are larger than 100px on the long side for speed. Ignores pixels that are more transparent than the `alpha_threshold`.

Parameters

- **image** (*str*) – A filename, `pathlib.Path` object or a file object.
- **downsample** (*bool*, *optional*) – if downsampling is enabled to speed up iteration.
- **max_size** (*int*, *optional*) – max length of longest side if `downsample == True`.
- **alpha_threshold** (*int*, *optional*) – level at which transparent pixels are excluded.

Returns A dictionary with the following keys: red, green, blue.

Return type dict

Raises

- `IOError` – If the file cannot be found, or the image cannot be opened and identified.
- `ImageAveragingError` – If the image could not be averaged.

`imagecolor.file_average` (*image*, *name=None*, *downsample=True*, *max_size=100*, *alpha_threshold=245*)

Average a single image and keep track of its file name.

Averages a single image from a file or file-like object. `name` is extracted from the filepath unless set. By default downsamples images that are larger than 100px on the long side for speed. Ignores pixels that are more transparent than the `alpha_threshold`.

Parameters

- **image** (*str*) – A filename, `pathlib.Path` object or a file object.
- **name** (*str*, *optional*) – auto generated from path unless set.
- **downsample** (*bool*, *optional*) – if downsampling is enabled to speed up iteration.
- **max_size** (*int*, *optional*) – max length of longest side if `downsample == True`.
- **alpha_threshold** (*int*, *optional*) – level at which transparent pixels are excluded.

Returns A dictionary with the following keys: `name`, `red`, `green`, `blue`.

Return type `dict`

Raises

- `AttributeError` – If `name` is not passed in and cannot be set from filepath.
- `IOError` – If the file cannot be found, or the image cannot be opened and identified.
- `ImageAveragingError` – If the image could not be averaged.

`imagecolor.directory_average` (*path*, *image_formats*=(`'jpeg'`, `'png'`))

Average all images in a directory.

Accepts the path to a directory and averages each individual image. Uses `concurrent.futures` to process images in parallel. If images fail to average successfully, the exceptions are caught and logged allowing other images to finish. By default only averages `jpeg` and `png` images.

Parameters

- **path** (*str*) – Path to directory.
- **image_formats** (*tuple of str*, *optional*) – tuple of image formats used by `imgHDR` to determine what types of images to average. Defaults: (`'jpeg'`, `'png'`)

Returns For each image averaged, returns a list of dictionaries each with the following keys: `name`, `red`, `green`, `blue`.

Return type `list`

Raises `ImageAveragingError` – If no images were averaged successfully.

`imagecolor.single_directory_average` (*path*, *image_formats*=(`'jpeg'`, `'png'`))

Average all images in a directory into a single average.

Accepts the path to a directory and averages each all images together into a single directory average. Uses `concurrent.futures` to process images in parallel. If images fail to average successfully, the exceptions are caught and logged allowing other images to finish. By default only averages `jpeg` and `png` images.

Parameters

- **path** (*str*) – Path to directory.
- **image_formats** (*tuple of str*, *optional*) – tuple of image formats used by `imgHDR` to determine what types of images to average. Defaults: (`'jpeg'`, `'png'`)

Returns A dictionary with the following keys: `name`, `red`, `green`, `blue`.

Return type `dict`

Raises *DirectoryAveragingError* – If the directory could not be averaged.

`imagecolor.nested_directory_average` (*path*, *image_formats*=(*'jpeg'*, *'png'*))

Averages all subdirectories into a directory average for each directory.

Accepts the path to a directory and walks all the enclosed directories calling `single_directory_average` for each one that contains images. Uses `concurrent.futures` to process images in parallel. If images fail to average successfully, the exceptions are caught and logged allowing other images to finish. By default only averages jpeg and png images.

Parameters

- **path** (*str*) – path to directory
- **image_formats** (*tuple of str, optional*) – tuple of image formats used by `imghdr` to determine what types of images to average. Defaults: (*'jpeg'*, *'png'*)

Returns For each directory averaged, returns a list of dictionaries each with the following keys: `name`, `red`, `green`, `blue`.

Return type `list`

exception `imagecolor.ImageColorException`

Bases: `Exception`

Base Exception for all imagecolor exceptions.

exception `imagecolor.ImageAveragingError`

Bases: `imagecolor.exceptions.ImageColorException`

Raised when an image was unable to be averaged.

exception `imagecolor.DirectoryAveragingError`

Bases: `imagecolor.exceptions.ImageColorException`

Raised when a directory was unable to be averaged.

exception `imagecolor.NoResultsError`

Bases: `imagecolor.exceptions.ImageColorException`

Raised when a list of results is empty or invalid.

`imagecolor.results_line` (*results*)

Create a line of pixels from a list of results.

Accepts a list of results and creates an image that is 1 pixel tall and the length of the number of results. The image contains a pixel of the color of each result in the list of results.

Parameters **results** (*list*) – a list of imagecolor results

Returns linear image containing the results

Return type `PIL.Image.object`

`imagecolor.results_rectangle` (*results*, *aspectratio*=(*3*, *2*))

Create a rectangle of pixels from a list of results.

Accepts a list of results and creates an image that is rectangular. The aspect ratio can be set by passing a list formatted as [16,9] to `aspectratio`. The default is 3x2. The image contains a pixel of the color of each result in the list of results.

Parameters

- **results** (*list*) – a list of imagecolor results.

- **aspectratio** (*tuple of int*) – the aspect ratio of the image being created. Format (3, 2)

Returns rectangular image containing the results.

Return type PIL.Image.object

`imagecolor.results_save_csv(results, path)`

Create a csv file from a list of results.

Accepts the path to a new csv file and a list containing results. Writes the current results to a csv file which can be re-loaded again by using `csv_to_results`. The csv created is formatted as follows: ‘File or Folder’, ‘Red’, ‘Green’, ‘Blue’

Parameters

- **results** (*list*) – a list of imagecolor results.
- **path** (*str*) – the path to the file to be created.

`imagecolor.results_load_csv(path)`

Create a list of results from a csv file.

Accepts the path to a csv file formatted as follows: ‘File or Folder’, ‘Red’, ‘Green’, ‘Blue’ parses the file line by line skipping the header. Returns a list containing an list for each line in the csv. Does not do any input checks other than converting the r, g, b columns to ints.

Parameters **path** (*str*) – the path to the file to be loaded.

Returns a list of imagecolor results.

Return type list

3.1.2 Submodules

3.1.3 imagecolor.average module

imagecolor functions for averaging images.

`imagecolor.average.core_average(image, downsample=True, max_size=100, alpha_threshold=245)`

Average a single image.

Averages a single image from a file or file-like object. By default downsamples images that are larger than 100px on the long side for speed. Ignores pixels that are more transparent than the `alpha_threshold`.

Parameters

- **image** (*str*) – A filename, `pathlib.Path` object or a file object.
- **downsample** (*bool, optional*) – if downsampling is enabled to speed up iteration.
- **max_size** (*int, optional*) – max length of longest side if `downsample == True`.
- **alpha_threshold** (*int, optional*) – level at which transparent pixels are excluded.

Returns A dictionary with the following keys: red, green, blue.

Return type dict

Raises

- `IOError` – If the file cannot be found, or the image cannot be opened and identified.

- `ImageAveragingError` – If the image could not be averaged.

`imagecolor.average.directory_average` (*path*, *image_formats*=(*'jpeg'*, *'png'*))

Average all images in a directory.

Accepts the path to a directory and averages each individual image. Uses `concurrent.futures` to process images in parallel. If images fail to average successfully, the exceptions are caught and logged allowing other images to finish. By default only averages jpeg and png images.

Parameters

- **path** (*str*) – Path to directory.
- **image_formats** (*tuple of str, optional*) – tuple of image formats used by `imghdr` to determine what types of images to average. Defaults: (*'jpeg'*, *'png'*)

Returns For each image averaged, returns a list of dictionaries each with the following keys: `name`, `red`, `green`, `blue`.

Return type list

Raises `ImageAveragingError` – If no images were averaged successfully.

`imagecolor.average.file_average` (*image*, *name*=*None*, *downsample*=*True*, *max_size*=*100*, *alpha_threshold*=*245*)

Average a single image and keep track of its file name.

Averages a single image from a file or file-like object. `name` is extracted from the filepath unless set. By default downsamples images that are larger than 100px on the long side for speed. Ignores pixels that are more transparent than the `alpha_threshold`.

Parameters

- **image** (*str*) – A filename, `pathlib.Path` object or a file object.
- **name** (*str, optional*) – auto generated from path unless set.
- **downsample** (*bool, optional*) – if downsampling is enabled to speed up iteration.
- **max_size** (*int, optional*) – max length of longest side if `downsample == True`.
- **alpha_threshold** (*int, optional*) – level at which transparent pixels are excluded.

Returns A dictionary with the following keys: `name`, `red`, `green`, `blue`.

Return type dict

Raises

- `AttributeError` – If `name` is not passed in and cannot be set from filepath.
- `IOError` – If the file cannot be found, or the image cannot be opened and identified.
- `ImageAveragingError` – If the image could not be averaged.

`imagecolor.average.nested_directory_average` (*path*, *image_formats*=(*'jpeg'*, *'png'*))

Averages all subdirectories into a directory average for each directory.

Accepts the path to a directory and walks all the enclosed directories calling `single_directory_average` for each one that contains images. Uses `concurrent.futures` to process images in parallel. If images fail to average successfully, the exceptions are caught and logged allowing other images to finish. By default only averages jpeg and png images.

Parameters

- **path** (*str*) – path to directory

- **image_formats** (*touple of str, optional*) – tuple of image formats used by `img_hdr` to determine what types of images to average. Defaults: ('jpeg', 'png')

Returns For each directory averaged, returns a list of dictionaries each with the following keys: name, red, green, blue.

Return type list

`imagecolor.average.single_directory_average` (*path, image_formats=('jpeg', 'png')*)
Average all images in a directory into a single average.

Accepts the path to a directory and averages each all images together into a single directory average. Uses `concurrent.futures` to process images in parallel. If images fail to average successfully, the exceptions are caught and logged allowing other images to finish. By default only averages jpeg and png images.

Parameters

- **path** (*str*) – Path to directory.
- **image_formats** (*touple of str, optional*) – tuple of image formats used by `img_hdr` to determine what types of images to average. Defaults: ('jpeg', 'png')

Returns A dictionary with the following keys: name, red, green, blue.

Return type dict

Raises `DirectoryAveragingError` – If the directory could not be averaged.

3.1.4 imagecolor.exceptions module

`imagecolor` module containing all publically raised exceptions.

exception `imagecolor.exceptions.DirectoryAveragingError`
Bases: `imagecolor.exceptions.ImageColorException`

Raised when an directory was unable to be averaged.

exception `imagecolor.exceptions.ImageAveragingError`
Bases: `imagecolor.exceptions.ImageColorException`

Raised when an image was unable to be averaged.

exception `imagecolor.exceptions.ImageColorException`
Bases: `Exception`

Base Exception for all `imagecolor` exceptions.

exception `imagecolor.exceptions.NoResultsError`
Bases: `imagecolor.exceptions.ImageColorException`

Raised when a list of results is empty or invalid.

3.1.5 imagecolor.loadsave module

`imagecolor` functions for loading and saving results.

`imagecolor.loadsave.results_line` (*results*)
Create a line of pixels from a list of results.

Accepts a list of results and creates an image that is 1 pixel tall and the length of the number of results. The image contains a pixel of the color of each result in the list of results.

Parameters **results** (*list*) – a list of `imagecolor` results

Returns linear image containing the results

Return type PIL.Image.object

`imagecolor.loadsave.results_load_csv(path)`

Create a list of results from a csv file.

Accepts the path to a csv file formatted as follows: 'File or Folder', 'Red', 'Green', 'Blue' parses the file line by line skipping the header. Returns a list containing an list for each line in the csv. Does not do any input checks other than converting the r, g, b columns to ints.

Parameters `path` (*str*) – the path to the file to be loaded.

Returns a list of imagecolor results.

Return type list

`imagecolor.loadsave.results_rectangle(results, aspectratio=(3, 2))`

Create a rectangle of pixels from a list of results.

Accepts a list of results and creates an image that is rectangular. The aspect ratio can be set by passing a list formatted as [16,9] to `aspectratio`. The default is 3x2. The image contains a pixel of the color of each result in the list of results.

Parameters

- **results** (*list*) – a list of imagecolor results.
- **aspectratio** (*tuple of int*) – the aspect ratio of the image being created. Format (3, 2)

Returns rectangular image containing the results.

Return type PIL.Image.object

`imagecolor.loadsave.results_save_csv(results, path)`

Create a csv file from a list of results.

Accepts the path to a new csv file and a list containing results. Writes the current results to a csv file which can be re-loaded again by using `csv_to_results`. The csv created is formatted as follows: 'File or Folder', 'Red', 'Green', 'Blue'

Parameters

- **results** (*list*) – a list of imagecolor results.
- **path** (*str*) – the path to the file to be created.

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

i

`imagecolor`, 7
`imagecolor.average`, 10
`imagecolor.exceptions`, 12
`imagecolor.loadsave`, 12

C

core_average() (in module imagecolor), 7
core_average() (in module imagecolor.average), 10

D

directory_average() (in module imagecolor), 8
directory_average() (in module imagecolor.average), 11
DirectoryAveragingError, 9, 12

F

file_average() (in module imagecolor), 7
file_average() (in module imagecolor.average), 11

I

ImageAveragingError, 9, 12
imagecolor (module), 7
imagecolor.average (module), 10
imagecolor.exceptions (module), 12
imagecolor.loadsave (module), 12
ImageColorException, 9, 12

N

nested_directory_average() (in module imagecolor), 9
nested_directory_average() (in module imagecolor.average), 11
NoResultsError, 9, 12

R

results_line() (in module imagecolor), 9
results_line() (in module imagecolor.loadsave), 12
results_load_csv() (in module imagecolor), 10
results_load_csv() (in module imagecolor.loadsave), 13
results_rectangle() (in module imagecolor), 9
results_rectangle() (in module imagecolor.loadsave), 13
results_save_csv() (in module imagecolor), 10
results_save_csv() (in module imagecolor.loadsave), 13

S

single_directory_average() (in module imagecolor), 8

single_directory_average() (in module imagecolor.average), 12