Image Wrench Documentation

Release 0.17.0

luphord

Nov 12, 2022

Contents:

CHAPTER 1

Image Wrench

A highly opinionated image processor for the commandline. Multiple subcommands can be executed sequentially to form a processing pipeline.

`imgwrench` is free software available under the MIT license. Detailed documentation can be found at https://imgwrench.readthedocs.io.

1.1 Features

- Subcommands can be executed sequentially to form a pipeline
- Command *blackwhite* for converting images to black and white
- Command *collage* creates a collage from multiple images
- Command *colorfix* for fixing the colors of aged photographs
- Command *crop* for cropping images to give aspect ratio
- Command *dither* for converting images to black and white and dithering
- Command *filmstrip* to stack images horizontally forming a filmstrip
- Command *flip* to flip/mirror images left-right
- Command *frame* to put a monocolor frame around images
- Command *framecrop* top frame and crop an image to a target aspect ratio
- Command *quad* collects four images to a quad
- Command *resize* for resizing images

- Command *save* for no processing, but saving images with the given parameters
- Command *stack* for vertically stacking images

1.2 Installation

Make sure you have Python and pip installed and available in your $PATH. Then `imgwrench` can be installed with

```
pip install imgwrench
```

In order to install for the current user only, you may want to execute

```
pip install --user imgwrench
```

instead. In this case you will have to ensure that the local bin directory (typically `~/.local/bin` on Linux systems) is contained in your $PATH.

Note that legacy Python2 is not supported. If your system still ships Python2 as the default Python interpreter, you may have to execute

```
pip3 install imgwrench
```

or

```
python3 -m pip install imgwrench
```

1.3 Usage

`imgwrench` is used on the command line by piping file paths into it, e.g.

```
ls /path/to/my/images/*.jpg | imgwrench blackwhite
```

Full command line help is

```
Usage: imgwrench [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

A highly opinionated image processor for the commandline. Multiple
subcommands can be executed sequentially to form a processing pipeline.

Options:
-i, --image-list FILENAME  File containing paths to images for processing,
                           defaults to stdin

-r, --repeat INTEGER       repeat every image in input sequence  [default:
                             1]

-p, --prefix TEXT          prefix for all output filenames before numbering
                           [default: img_]

-d, --digits INTEGER       number of digits for file numbering  [default: 4]
-c, --increment INTEGER    increment for file numbering  [default: 1]
-k, --keep-names           keep original file names instead of numbering
                           [default: False]
```

(continues on next page)

(continued from previous page)

```
-f, --force-overwrite      force overwriting output image file if it exists
                         [default: False]

-o, --outdir DIRECTORY     output directory  [default: .]
-q, --quality INTEGER      quality of the output images, integer 0 - 100
                         [default: 88]

-e, --preserve-exif        preserve image exif and xmp metadata if available
                         [default: False]

-j, --jpg / --png          save output images in JPEG format (otherwise PNG)
                         [default: True]

--help                     Show this message and exit.

Commands:
blackwhite  Convert color images to black and white.
collage     Create a collage from multiple images.
colorfix    Fix colors by stretching channel histograms to full range.
crop        Crop images to the given aspect ratio.
dither      Apply black-white dithering to images.
filmstrip   Stack all images horizontally, creating a filmstrip.
flip        Flip/mirror images left-right.
frame       Put a monocolor frame around images.
framecrop   Crop and frame an image to a target aspect ratio.
quad        Collects four images to a quad.
resize      Resize images to a maximum side length preserving aspect...
save        No-op to enable saving of images without any processing.
stack       Stacks pairs of images vertically, empty space in the middle.
```

1.4 Pipelines

`imgwrench` subcommands can be combined into pipelines. This saves you from generating intermediate files cluttering your filesystem or reducing the quality of the final results. For example, if you want to convert all images in the current directory to black and white, put a white frame around them and have them cut to an aspect ratio of 3:2 (for standard format printing), you would execute the following command:

```
ls *.JPG | \
imgwrench -o out -q 95 -p oldschool_img_ \
        blackwhite \
        framecrop -a 3:2 -w 0.03 -c white
```

Please refer to the detailed subcommand documentation for the individual parameters.

1.5 Developer Notes

Should you run into the following exception while running `imgwrench` from an editable install

```
importlib_metadata.PackageNotFoundError: No package metadata was found for imgwrench
```

try executing `make dist` to regenerate the `egg` files required bei `importlib` which have likely been removed by a call to `make clean`.

1.6 Credits

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

CHAPTER 2

Installation

2.1 Stable release

To install Image Wrench, run this command in your terminal:

```
$ pip install imgwrench
```

This is the preferred method to install Image Wrench, 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 Image Wrench can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/luphord/imgwrench
```

Or download the tarball:

```
$ curl  -OL https://github.com/luphord/imgwrench/tarball/master
```

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

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

CHAPTER 3

Usage

3.1 blackwhite

The *blackwhite* subcommand converts color images to black and white.

Assuming image *rainbow.jpg* in the current directory, *blackwhite* can be applied to output to *img_0000.jpg* as follows:

```
ls rainbow.jpg | imgwrench blackwhite
```

At the moment, *blackwhite* supports no further parameters. Conversion is delegated to the PIL *convert('L')* method call.

3.2 collage

The *collage* subcommand creates a collage of all input images. The method for image composition is based on the *Blocked Recursive Image Composition (BRIC)* algorithm by C. Brian Atkins..

Assuming a couple of images in the current directory, *collage* generates a collage in *img_0000.jpg* as follows:

```
ls *.jpg | imgwrench collage -c lightgrey
```

`-w/--width` and `-s/--height` can be used to specify the dimensions of the output image. The parameter `-f/--frame-width` specifies the frame width as fraction of the longer image side, e.g. 0.1 for a frame width that is equal to 10% of the longer image side. Also `-c/--color` is supported which accepts the frame color as either a name (e.g. `white`, `green`), a hex value (e.g. `#ab1fde`) or an rgb function value (e.g. `rgb(120,23,217)`).

```
Usage: imgwrench collage [OPTIONS]

Create a collage from multiple images.

Options:
-w, --width INTEGER          width of the collage  [default: 3072]
-s, --height INTEGER         height of the collage  [default: 2048]
-f, --frame-width FLOAT      width of the frame as a fraction of the longer
                             image side  [default: 0.01]

-c, --color COLOR            color of the frame as a color name, hex value or
                             in rgb(...) function form  [default: white]

-x, --seed INTEGER           seed for random number generator  [default: 123]
-n, --number-tries INTEGER   number of tries for layout generation  [default:
                             100]

--help                       Show this message and exit.
```

3.3 colorfix

The *colorfix* subcommand repairs aged images with a color shift (usually towards red) by shifting the channel histograms back to the full range.

Assuming image *old.jpg* in the current directory, *colorfix* can be applied to repair its colors and output as *img_0000.jpg* as follows:

```
ls old.jpg | imgwrench colorfix
```

The *colorfix* algorithm stretches the channel histogram to specified clipping values (cutoffs). The precise specification depends on the `-m/--method` option.

`--method=quantiles` supports the float parameter `-a/--alpha` representing the quantile within each color channel that is clipped to the minimum and maximum value. It defaults to `0.01`. Increasing `alpha` will stretch the histogram further and will intensify the contrast of the resulting image.

`--method=fixed-cutoff` lets you specify the cutoff colors directly as named color, hex value or in rgb(...) function form. Use `-l/--lower-cutoff` and `-u/--upper-cutoff` to specify.

`--method=quantiles-fixed-cutoff` combines the other two methods and applies the "stronger" cutoff (i.e. the higher value of lower cutoffs and lower value of upper cutoffs).

```
Usage: imgwrench colorfix [OPTIONS]

  Fix colors by stretching channel histograms to full range.

Options:
  -m, --method [quantiles|fixed-cutoff|quantiles-fixed-cutoff]
                                  algorithm method to use; quantiles stretches
                                  all channel histograms between the quantiles
                                  specified by --alpha; fixed-cutoff stretches
                                  channels between the cutoffs specified by
                                  --lower-cutoff and --upper-cutoff;
                                  quantiles-fixed-cutoff combines the two
                                  methods and applies the "stronger" of both
                                  cutoffs (i.e. the higher value of lower
                                  cutoffs and lower value of upper cutoffs)
                                  [default: (dynamic)]
  -a, --alpha FLOAT               quantile (low and high) to be clipped to
                                  minimum and maximum color; relevant for
                                  --method=quantiles and --method=quantiles-
                                  fixed-cutoff  [default: 0.01]
  -l, --lower-cutoff COLOR        lower cutoff as a color name, hex value or
                                  in rgb(...) function form; relevant for
                                  --method=fixed-cutoff and
                                  --method=quantiles-fixed-cutoff  [default:
                                  rgb(127,0,0)]
  -u, --upper-cutoff COLOR        lower cutoff as a color name, hex value or
                                  in rgb(...) function form; relevant for
                                  --method=fixed-cutoff and
                                  --method=quantiles-fixed-cutoff  [default:
                                  white]
  --help                          Show this message and exit.
```

3.4 crop

The *crop* subcommand crops images to a specified aspect ratio.

Assuming image *rainbow.jpg* in the current directory, *crop* can be applied with aspect ratio 2:1 and output to *img_0000.jpg* as follows:

```
ls rainbow.jpg | imgwrench crop -a 2:1
```

crop supports the parameter `-a/--aspect-ratio` which has to be an aspect ratio specified as two numbers separated by a colon, e.g. `2:1`, `3:4`, `117:123`.

```
Usage: imgwrench crop [OPTIONS]

Crop images to the given aspect ratio.

Options:
-a, --aspect-ratio RATIO  aspect ratio to crop to  [default: 3:2]
--help                    Show this message and exit.
```

3.5 dither

The *dither* command converts the image to true black and white (not greyscale) and applies dithering.

Assuming image *lensflare.jpg* in the current directory, *dither* can be applied to output *img_0000.jpg* as follows:

```
ls lensflare.jpg | imgwrench dither
```

The parameter `-b/--brightness-factor` adjusts the brightness of the image before dithering. It is usually recommended to make images brighter before dithering. A value of 1.0 is neutral (i.e. has no effect), larger values will make the image brighter, smaller values will make it darker. It defaults to 1.5.

```
Usage: imgwrench dither [OPTIONS]

Apply black-white dithering to images.

Options:
-b, --brightness-factor FLOAT  adjust brightness before dithering (1.0 is
                                neutral, larger is brighter, smaller is
                                darker)  [default: 1.5]
--help                         Show this message and exit.
```

3.6 filmstrip

The *filmstrip* command stacks all images in the pipeline horizontally to create a filmstrip within a single row. Assuming a couple of images in the current directory, *filmstrip* will create a single output image *img_0000.jpg* width height 800 pixels as follows:

```
ls *.JPG | imgwrench filmstrip -s 800
```

The parameter `-s/--height specifies` the total height of the resulting collage, its width will be inferred. `-w/--frame-width` determines the frame width relative to the specified height and `-c/--color` sets the frame color.

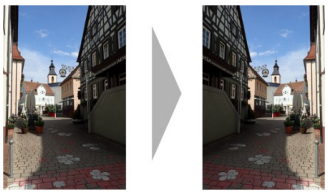
```
Usage: imgwrench filmstrip [OPTIONS]

Stack all images horizontally, creating a filmstrip.

Options:
-s, --height INTEGER     height of the filmstrip  [default: 2048]
-w, --frame-width FLOAT  width of the frame as a fraction of the height  of
                        the filmstrip  [default: 0.025]
-c, --color COLOR        color of the frame as a color name, hex value or in
                        rgb(...) function form  [default: white]
--help                   Show this message and exit.
```

3.7 flip

The *flip* command flips (a.k.a. mirrors) all images in the pipeline horizontally, i.e. what was left is now right and vice versa.

Assuming image *town.jpg* in the current directory, *flip* will output the mirrored image to *img_0000.jpg* as follows:

```
ls town.jpg | imgwrench flip
```



flip takes no parameters.

3.8 frame

The *frame* subcommand puts a monocolor frame around the image. The frame is added to the image size.

Assuming image *saarschleife.jpg* in the current directory, *frame* can be applied with a frame width equal to 3% of the original image width (which is in landscape format, i.e. width > height) and a light grey color to output to *img_0000.jpg* as follows:

```
ls saarschleife.jpg | imgwrench frame -w 0.03 -c '#ddd'
```

frame supports the parameter `-w/--frame-width` which specifies the frame width as fraction of the longer image side, e.g. 0.1 for a frame width that is equal to 10% of the longer image side. Also `-c/--color` is supported which accepts the frame color as either a name (e.g. `white`, `green`), a hex value (e.g. `#ab1fde`) or an rgb function value (e.g. `rgb(120,23,217)`).

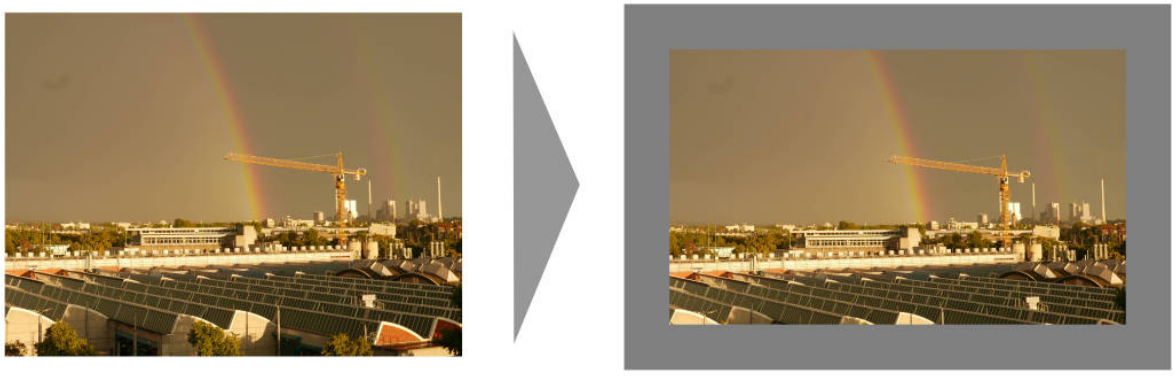
```
Usage: imgwrench frame [OPTIONS]

Put a monocolor frame around images.

Options:
-w, --frame-width FLOAT  width of the frame as a fraction of the longer
                        image side  [default: 0.025]
-c, --color COLOR        color of the frame as a color name, hex value or in
                        rgb(...) function form  [default: white]
--help                   Show this message and exit.
```

3.9 framecrop

The *framecrop* command crops and frames an image to a target aspect ratio. The resulting image will conform to the target aspect ratio so you don't have to precompute the required crop ratio.

Assuming image *rainbow.jpg* in the current directory, *framecrop* can be applied with aspect ratio 3:2, a grey frame of width 10% and output to *img_0000.jpg* as follows:

```
ls rainbow.jpg | imgwrench framecrop -a '3:2' -w 0.1 -c grey
```



framecrop supports the parameter `-a/--aspect-ratio` which has to be an aspect ratio specified as two numbers separated by a colon, e.g. `2:1`, `3:4`, `117:123`. This will be the ratio of the final image *including* the frame.

The parameter `-w/--frame-width` specifies the frame width as fraction of the longer image side after the crop operation. Also `-c/--color` is supported which accepts the frame color as either a name (e.g. `white`, `green`), a hex value (e.g. `#ab1fde`) or an rgb function value (e.g. `rgb(120,23,217)`).

```
Usage: imgwrench framecrop [OPTIONS]

Crop and frame an image to a target aspect ratio.

Options:
-a, --aspect-ratio RATIO  aspect ratio of final image including frame
                            [default: 3:2]
-w, --frame-width FLOAT   width of the frame as a fraction of the longer
                            side of the cropped image  [default: 0.025]
-c, --color COLOR         color of the frame as a color name, hex value or
                            in rgb(...) function form  [default: white]
--help                    Show this message and exit.
```

3.10 quad

The *quad* command creates grids consisting of four images. The primary use case is batch creation of small prints. Images are rotated in order to minimize the area cropped away, i.e. landscape images are rotated if the target image has portrait aspect ratio and portrait images are rotated if the target image has landscape aspect ratio.

```
ls *.jpg | imgwrench quad
```

quad automatically creates the correct amount of target images and leaves remaining space blank (color can be specified using `--color`). Also, the usual `--width`, `--height` and `--frame-width` options are supported.

```
Usage: imgwrench quad [OPTIONS]

Collects four images to a quad.

Options:
-w, --width INTEGER       width of the quad image  [default: 3072]
-s, --height INTEGER      height of the quad image  [default: 2048]
-f, --frame-width FLOAT   width of the frame as a fraction of the longer
                            side of the output image  [default: 0.0]

-d, --double-inner-frame  double inner frame width for even cuts
-c, --color COLOR         color of the frame as a color name, hex value or
                            in rgb(...) function form  [default: white]

--help                    Show this message and exit
```

3.11 resize

The *resize* command resizes images to a maximum side length while preserving the original aspect ratio.

Assuming image *lensflare.jpg* in the current directory, *resize* can be applied with a maximum side length of 300 pixels to *img_0000.jpg* as follows:

```
ls lensflare.jpg | imgwrench resize -m 300
```

The parameter `-m/--maxsize` specifies the new maximum side length of the resized image, i.e. for landscape images it specifies the new width and for portrait images it specifies the new height.

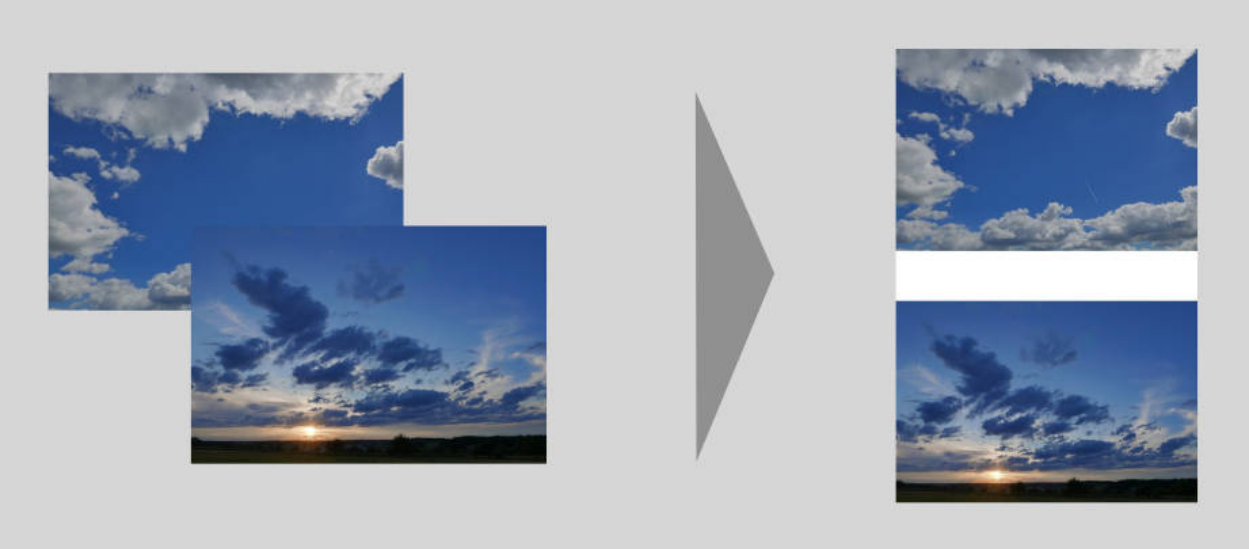
```
Usage: imgwrench resize [OPTIONS]

Resize images to a maximum side length preserving aspect ratio.

Options:
-m, --maxsize INTEGER  size of the longer side (width or height) in pixels
                        [default: 1024]
--help                 Show this message and exit.
```

3.12 stack

The *stack* command stacks pairs of images vertically.

Assuming image *sky.jpg* and *sunset.jpg* in the current directory, *stack* can be applied with a target width of 400 and height 600 pixels to output to *img_0000.jpg* as follows:

```
ls sky.jpg sunset.jpg | imgwrench stack -w 400 -s 600
```



The parameters `-w/--width` and `-s/--height` (attention: it is `-s`, not `-h` to avoid conflicts with `--help`) specify the target width and height of the output image. Remaining space will be white.

```
Usage: imgwrench stack [OPTIONS]

Stack images vertically, empty space in the middle.

Options:
-w, --width INTEGER   width of the stacked image  [default: 2048]
-s, --height INTEGER  height of the stacked image  [default: 3072]
--help                Show this message and exit.
```

CHAPTER 4

imgwrench

4.1 imgwrench package

4.1.1 Subpackages

imgwrench.commands package

Submodules

imgwrench.commands.blackwhite module

Convert color images to black and white.

imgwrench.commands.blackwhite.**blackwhite**(*image*)
Convert color images to black and white.

imgwrench.commands.colorfix module

Fix colors of images by stretching their channel histograms to full range.

imgwrench.commands.colorfix.**colorfix_fixed_cutoff**(*img*, *lower_cutoff*, *upper_cutoff*)
Fix colors by stretching channel histograms between given cutoff colors to full range.

imgwrench.commands.colorfix.**colorfix_quantiles**(*img*, *level=0.01*)
Fix colors by stretching channel histograms between given quantiles to full range.

imgwrench.commands.colorfix.**colorfix_quantiles_fixed_cutoff**(*img*, *level*, *lower_cutoff*, *upper_cutoff*)
Fix colors by stretching channel histogram between inner values of given quantiles and cutoff colors to full range.

imgwrench.commands.colorfix.**quantiles**(*img*, *level=0.01*)
Compute high and low quantiles to the given level

imgwrench.commands.colorfix.**stretch_histogram**(*img*, *cutoffs*)
Stretch channel histograms between given cutoffs to full range.

imgwrench.commands.crop module

Crop images to the given aspect ratio.

imgwrench.commands.crop.**crop**(*image*, *aspect_ratio*)
Crop images to the given aspect ratio.

imgwrench.commands.frame module

Put a monocolor frame around images.

imgwrench.commands.frame.**frame**(*image*, *width*, *color*)
Put a monocolor frame around images.

imgwrench.commands.resize module

Resize images to a maximum side length preserving aspect ratio.

imgwrench.commands.resize.**resize**(*image*, *maxsize*)
Resize image to maxsize (longer) side length preserving aspect ratio.

imgwrench.commands.save module

No-op to enable saving of images using imgwrench without processing.

imgwrench.commands.stack module

Stack images vertically, empty space in the middle.

imgwrench.commands.stack.**stack**(*img1*, *img2*, *width*, *height*)
Stack images vertically, empty space in the middle.

Module contents

4.1.2 Submodules

4.1.3 imgwrench.cli module

Command Line Interface for Image Wrench.

imgwrench.cli.**pipeline**(*image_processors*, *image_list*, *repeat*, *prefix*, *increment*, *digits*, *keep_names*, *force_overwrite*, *outdir*, *quality*, *preserve_exif*, *jpg*)

4.1.4 imgwrench.info module

Image meta information.

class imgwrench.info.**ImageInfo**(*path*, *index*, *exif*, *xmp*)
Bases: object

Container for image meta information

4.1.5 imgwrench.param module

Custom parameter types for the click-based CLI

class imgwrench.param.**Color**
Bases: click.types.ParamType

Parameter type representing a color as name, hex or rgb value

convert(*value*, *param*, *ctx*)
Convert the value to the correct type. This is not called if the value is None (the missing value).

This must accept string values from the command line, as well as values that are already the correct type. It may also convert other compatible types.

The param and ctx arguments may be None in certain situations, such as when converting prompt input.

If the value cannot be converted, call fail() with a descriptive message.

Parameters

- **value** – The value to convert.
- **param** – The parameter that is using this type to convert its value. May be None.
- **ctx** – The current context that arrived at this value. May be None.

name = 'color'

class imgwrench.param.**Ratio**
Bases: click.types.ParamType

Parameter type representing a ratio or rational number

convert(*value*, *param*, *ctx*)
Convert the value to the correct type. This is not called if the value is None (the missing value).

This must accept string values from the command line, as well as values that are already the correct type. It may also convert other compatible types.

The param and ctx arguments may be None in certain situations, such as when converting prompt input.

If the value cannot be converted, call fail() with a descriptive message.

Parameters

- **value** – The value to convert.
- **param** – The parameter that is using this type to convert its value. May be None.
- **ctx** – The current context that arrived at this value. May be None.

name = 'ratio'

4.1.6 Module contents

A command line tool for my image processing needs.

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/luphord/imgwrench/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

Image Wrench could always use more documentation, whether as part of the official Image Wrench 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/luphord/imgwrench/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 *imgwrench* for local development.

1. Fork the *imgwrench* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/imgwrench.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 imgwrench
$ cd imgwrench/
$ 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 imgwrench 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 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.com/github/luphord/imgwrench/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

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

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- luphord <luphord@protonmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.17.0 (2022-11-12)

- `-r/--repeat` option for repeating input files
- Constrain `click` version to `<8.1` due to breaking API change; this will be relaxed in a future version
- Constrain `Pillow` version to `<9.0` due to breaking tests; this will be relaxed in a future version
- Drop support for Python 3.6 and 3.7
- Add support for Python 3.9 and 3.10
- Upgrade dependencies
- Migrate from travis-ci.com to GitHub actions

7.2 0.16.1 (2021-06-19)

- Fix support for `click>=8.0` which has changed its behaviour regarding custom parameter types

7.3 0.16.0 (2021-01-23)

- `quad` subcommand supports doubling inner frame using the `-d/--double-inner-frame` flag

7.4 0.15.0 (2021-01-22)

- `collage` subcommand selects best layout based on score function
- `collage` subcommand supports `-n/--number-tries` parameter to specify number of layout tries

7.5 0.14.0 (2021-01-21)

- BREAKING CHANGE: replace golden collage approach with BRIC algorithm in `collage` subcommand
- BREAKING CHANGE: drop support for Python 3.5
- format code with `black`

7.6 0.13.0 (2020-10-26)

- `quad` subcommand to to collect four images into a quad
- improve documentation

7.7 0.12.0 (2020-07-24)

- `flip` subcommand to flip/mirror images left-right
- Monkey patch `IFDRational.__eq__` method of Pillow in tests to avoid regression with Pillow 7.2.0

7.8 0.11.1 (2020-04-05)

- `-x/--seed` option for collage to control initialization of random number generator

7.9 0.11.0 (2020-03-21)

- collage subcommand for creating a framed collage from images
- BREAKING CHANGE: default method for colorfix is now quantiles-fixed-cutoff
- preserve xmp metadata when `-e/--preserve-exif` is used (in addition to exif metadata)

7.10 0.10.0 (2020-03-04)

- `-m/--method` option to colorfix (default: quantiles)
- add fixed-cutoff as new method to colorfix accepting fixed colors as color cutoff boundaries
- add quantiles-fixed-cutoff as new method to colorfix combining quantiles and fixed-cutoff
- deprecate running colorfix without specifying method (as default will change in next version)

7.11 0.9.0 (2020-02-19)

- add numpy as dependency
- change colorfix algorithm to vectorized numpy code for performance
- support Python 3.8

7.12 0.8.1 (2020-01-12)

- fix crash when orientation is missing in exif headers

7.13 0.8.0 (2019-07-10)

- dither subcommand for dithering
- filmstrip subcommand to stack images horizontally
- images can be saved in PNG format using `--png` CLI flag

7.14 0.7.1 (2019-05-16)

- fix development status

7.15 0.7.0 (2019-05-16)

- option for preserving exif image metadata
- fix error when running with `-k/--keep-names`
- status progress to Alpha

7.16 0.6.0 (2019-03-14)

- framecrop subcommand to crop and frame an image to a target aspect ratio incl. tests and docs
- breaking change: moved command modules to *commands* package
- introduced ImageInfo as a container for additional meta information in the pipeline
- increased test coverage
- more documentation

7.17 0.5.2 (2019-03-10)

- use a custom parameter type for colors

7.18 0.5.1 (2019-03-09)

- changed default frame width to 0.025
- usage doc for frame subcommand
- consistent alphabetic sorting of subcommands
- use a custom parameter type for ratios

7.19 0.5.0 (2019-03-07)

- blackwhite subcommand to convert color images to black and white; incl. doc
- frame subcommand to put a monocolor frame around images; incl. tests

7.20 0.4.0 (2019-02-26)

- convert RGBA mode PNG images to RGB (to enable saving as JPG)
- crop subcommand to crop images to a specified aspect ratio
- documentation for colorfix and crop

7.21 0.3.0 (2019-02-17)

- `-d/--digits` option to specify number of digits in file names
- `-c/--increment` option to define increment for file numbering
- create non-existing output folder instead of complaining

7.22 0.2.0 (2019-01-30)

- no-op save command for only saving images
- raise exception if output image already exists
- `-f/--force-overwrite` flag to enable overwriting output
- tests for cli, pipeline and resize

7.23 0.1.1 (2019-01-29)

- Fix __main__ module

7.24 0.1.0 (2019-01-29)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

i

Index