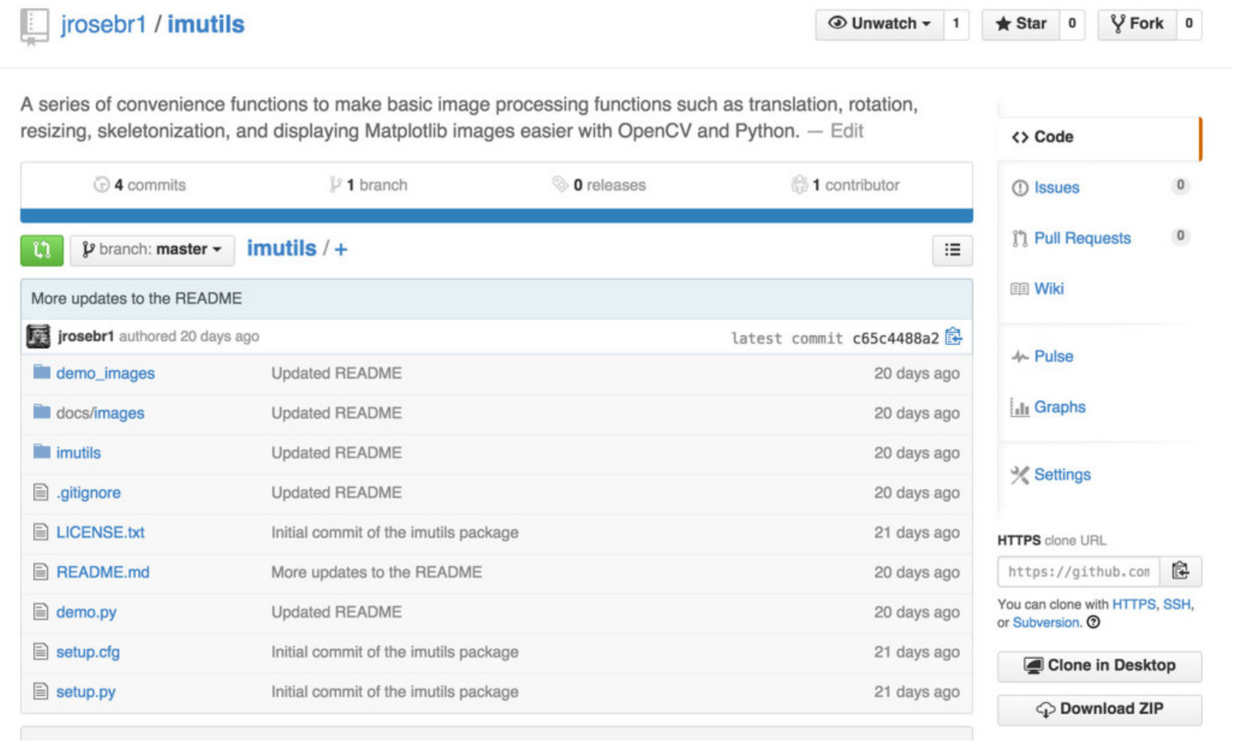
My imutils package: A series of OpenCV convenience functions

pyimagesearch.com/2015/02/02/just-open-sourced-personal-imutils-package-series-opencv-convenience-functions

Adrian Rosebrock

February 2, 2015



A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and Python. — Edit

More updates to the README

jrosebr1 authored 20 days ago — latest commit c65c4488a2

demo_images	Updated README	20 days ago
docs/images	Updated README	20 days ago
imutils	Updated README	20 days ago
.gitignore	Updated README	20 days ago
LICENSE.txt	Initial commit of the imutils package	21 days ago
README.md	More updates to the README	20 days ago
demo.py	Updated README	20 days ago
setup.cfg	Initial commit of the imutils package	21 days ago
setup.py	Initial commit of the imutils package	21 days ago

HTTPS clone URL

https://github.com

You can clone with HTTPS, SSH, or Subversion.

You know what’s a really good feeling?

Contributing to the open source community.

PyPI, the Python Package Index repository is a wonderful thing. It makes downloading, installing, and managing Python libraries and packages a breeze.

And with all that said, I have pushed my own personal imutils package online. I use this package nearly every single day when working on computer vision and image processing problems.

This package includes a series of OpenCV + convenience functions that perform basics tasks such as translation, rotation, resizing, and skeletonization.

In the future we will (probably, depending on feedback in the comments section) be performing a detailed code review of each of the functions in the

imutils

package, but for the time being, take a look at the rest of this blog post to see the functionality included in

imutils

, then be sure to install it on your own system!

Installing

This package assumes that you already have NumPy and OpenCV installed (along with matplotlib, if you intend on using the

opencv2matplotlib
function).
To install the the

imutils
library, just issue the following command:
I just open sourced my personal imutils package: A series of OpenCV convenience functions.

```
$ pip install imutils
```

Let’s go ahead and take a look at what we can do with the

imutils
package.

Translation

Translation is the shifting of an image in either the *x* or *y* direction. To translate an image in OpenCV you need to supply the *(x, y)*-shift, denoted as (t_x, t_y) to construct the translation matrix *M*:

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

And from there, you would need to apply the

cv2.warpAffine
function.
Instead of manually constructing the translation matrix *M* and calling

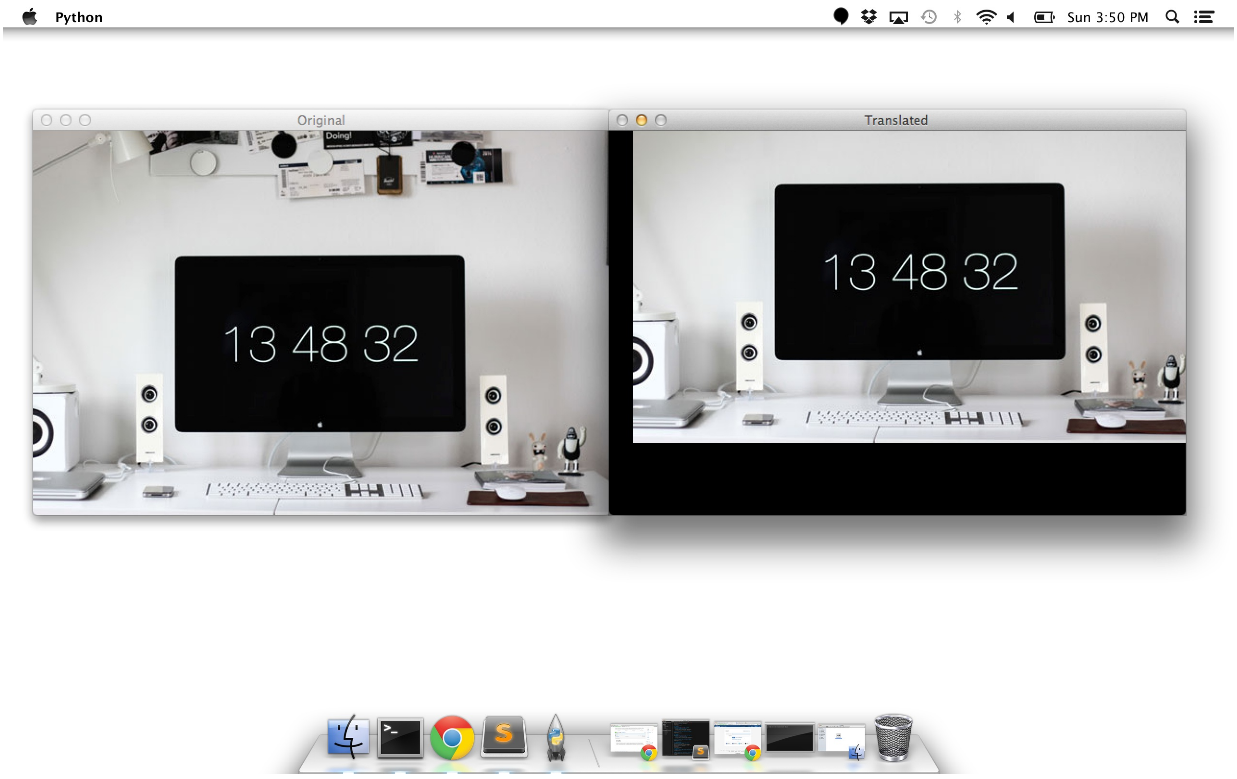
cv2.warpAffine
, you can simply make a call to the
translate
function of
imutils
.

Example:

I just open sourced my personal imutils package: A series of OpenCV convenience functions.

```
# translate the image x=25 pixels to the right and y=75 pixels up
translated = imutils.translate(workspace, 25, -75)
```

Output:



Rotation

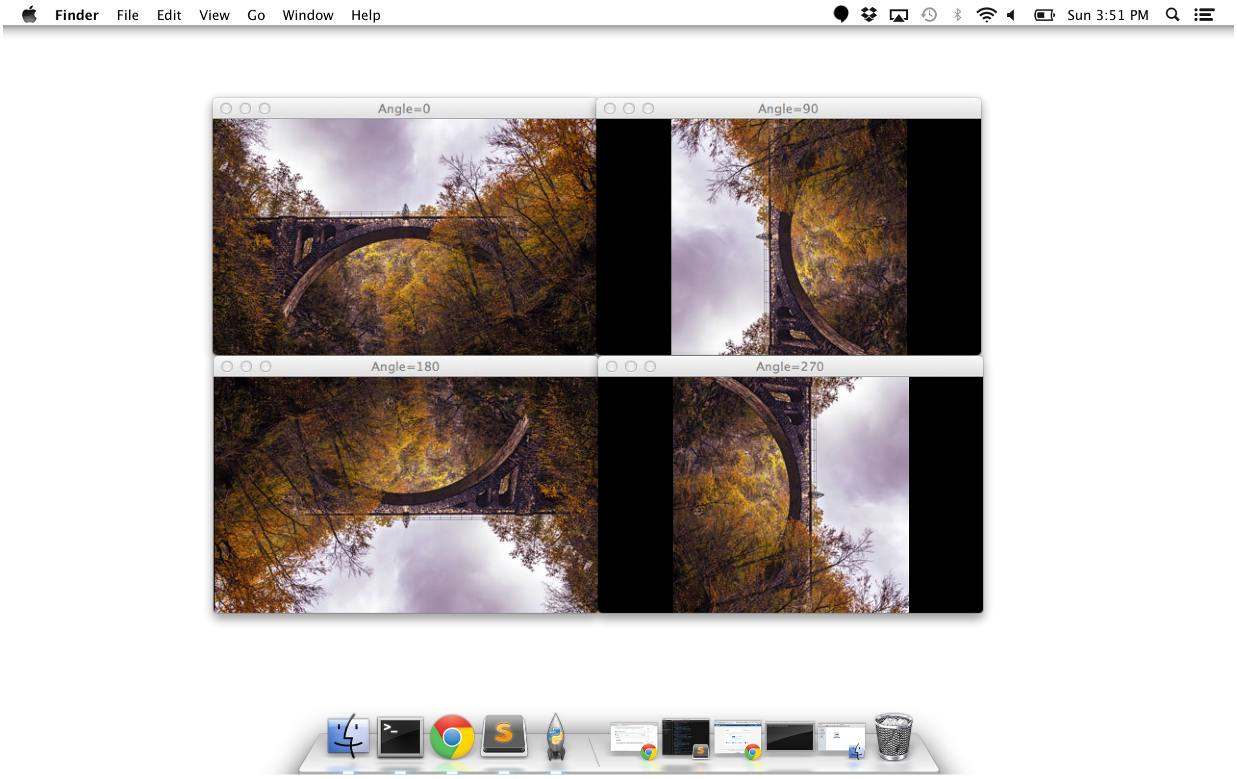
Rotating an image in OpenCV is accomplished by making a call to cv2.getRotationMatrix2D and cv2.warpAffine . Further care has to be taken to supply the *(x, y)*-coordinate of the point the image is to be rotated about. These calculation calls can quickly add up and make your code bulky and less readable. The rotate function in imutils helps resolve this problem.

Example:

I just open sourced my personal imutils package: A series of OpenCV convenience functions.

```
# loop over the angles to rotate the image
for angle in xrange(0, 360, 90):
# rotate the image and display it
rotated = imutils.rotate(bridge, angle=angle)
cv2.imshow("Angle=%d" % (angle), rotated)
```

Output:



Resizing

Resizing an image in OpenCV is accomplished by calling the

cv2.resize
 function. However, special care needs to be taken to ensure that the aspect ratio is maintained. This
resize
 function of
imutils
 maintains the aspect ratio and provides the keyword arguments
width
 and
height
 so the image can be resized to the intended width/height while (1) maintaining aspect ratio and (2) ensuring the dimensions of the image do not have to be explicitly computed by the developer.
Another optional keyword argument,

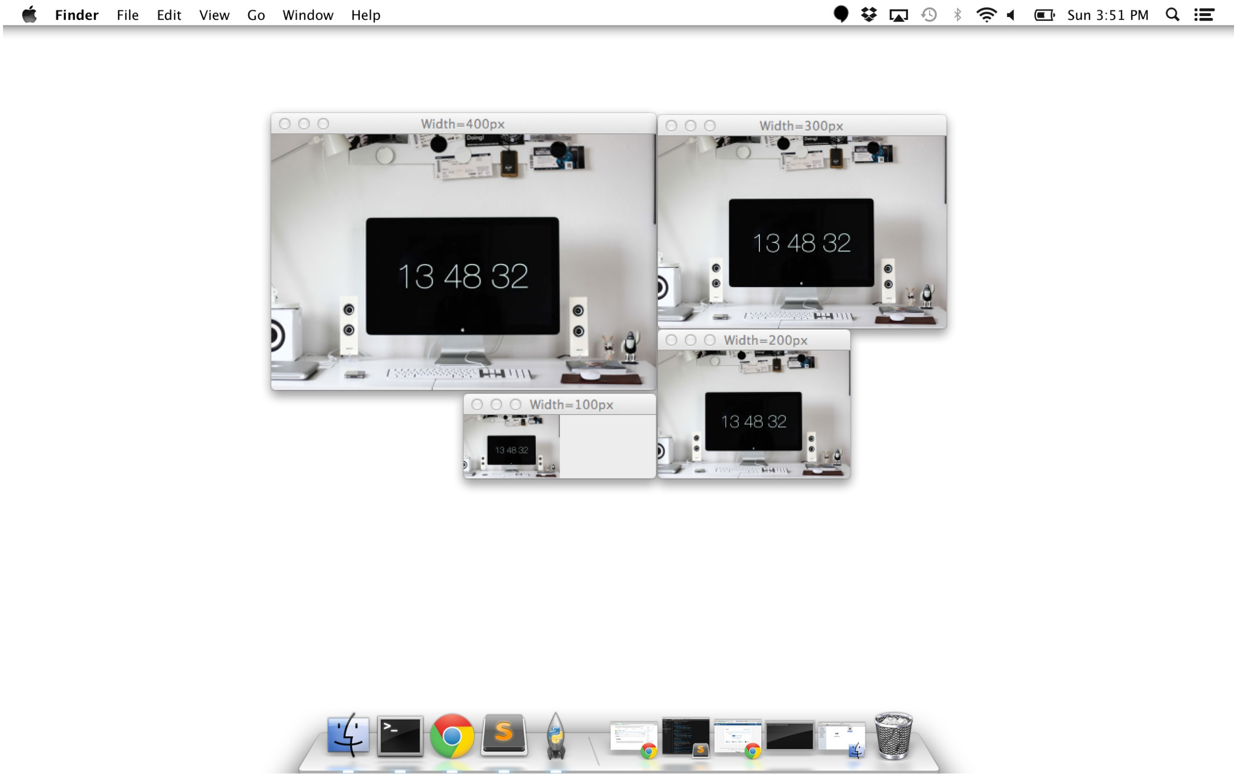
inter
, can be used to specify interpolation method as well.

Example:

I just open sourced my personal imutils package: A series of OpenCV convenience functions.
# loop over varying widths to resize the image to

```
for width in (400, 300, 200, 100):
# resize the image and display it
resized = imutils.resize(workspace, width=width)
cv2.imshow("Width=%dpx" % (width), resized)
```

Output:



Skeletonization

Skeletonization is the process of constructing the “topological skeleton” of an object in an image, where the object is presumed to be white on a black background. OpenCV does not provide a function to explicity construct the skeleton, but does provide the morphological and binary functions to do so.

For convenience, the

skeletonize
 function of
imutils
 can be used to construct the topological skeleton of the image.
The first argument,

size
 is the size of the structuring element kernel. An optional argument,
structuring
, can be used to control the structuring element — it defaults to
cv2.MORPH_RECT

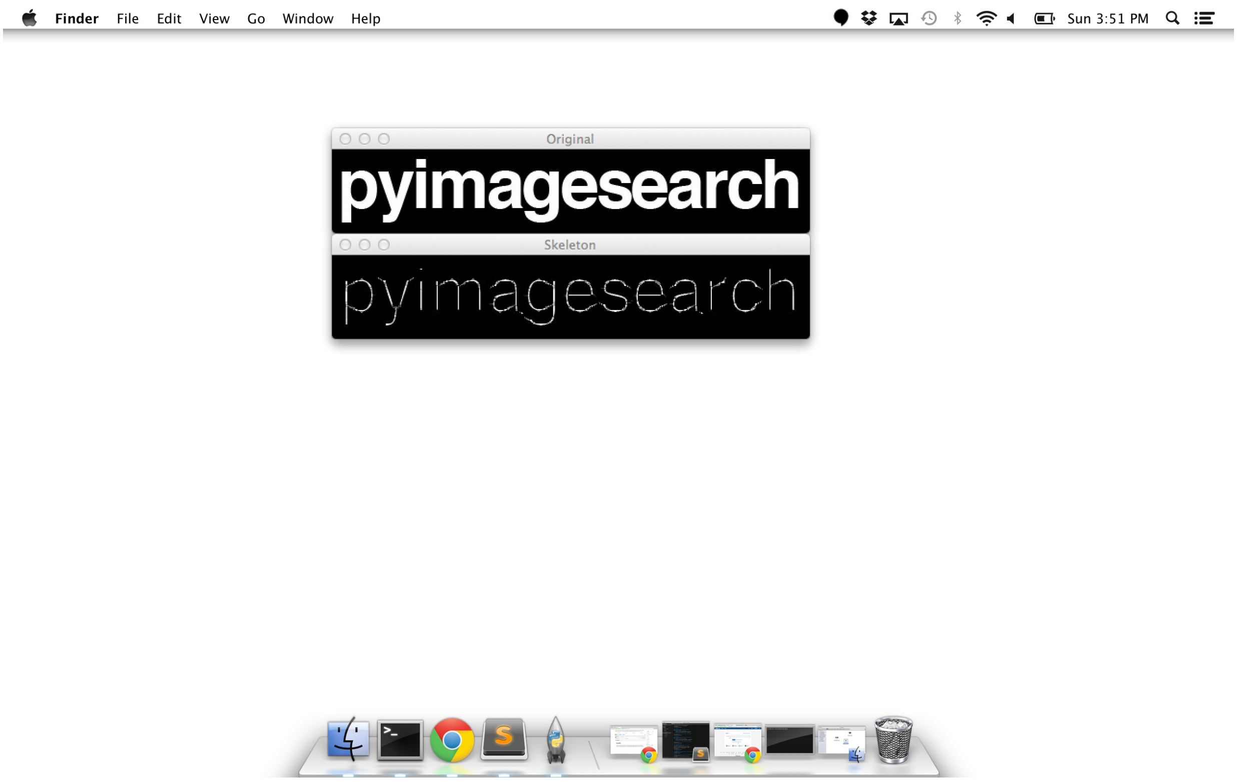
, but can be any valid structuring element.

Example:

I just open sourced my personal imutils package: A series of OpenCV convenience functions.

```
# skeletonize the image
gray = cv2.cvtColor(logo, cv2.COLOR_BGR2GRAY)
skeleton = imutils.skeletonize(gray, size=(3, 3))
cv2.imshow("Skeleton", skeleton)
```

Output:



Displaying with Matplotlib

In the Python bindings of OpenCV, images are represented as NumPy arrays in BGR order. This works fine when using the

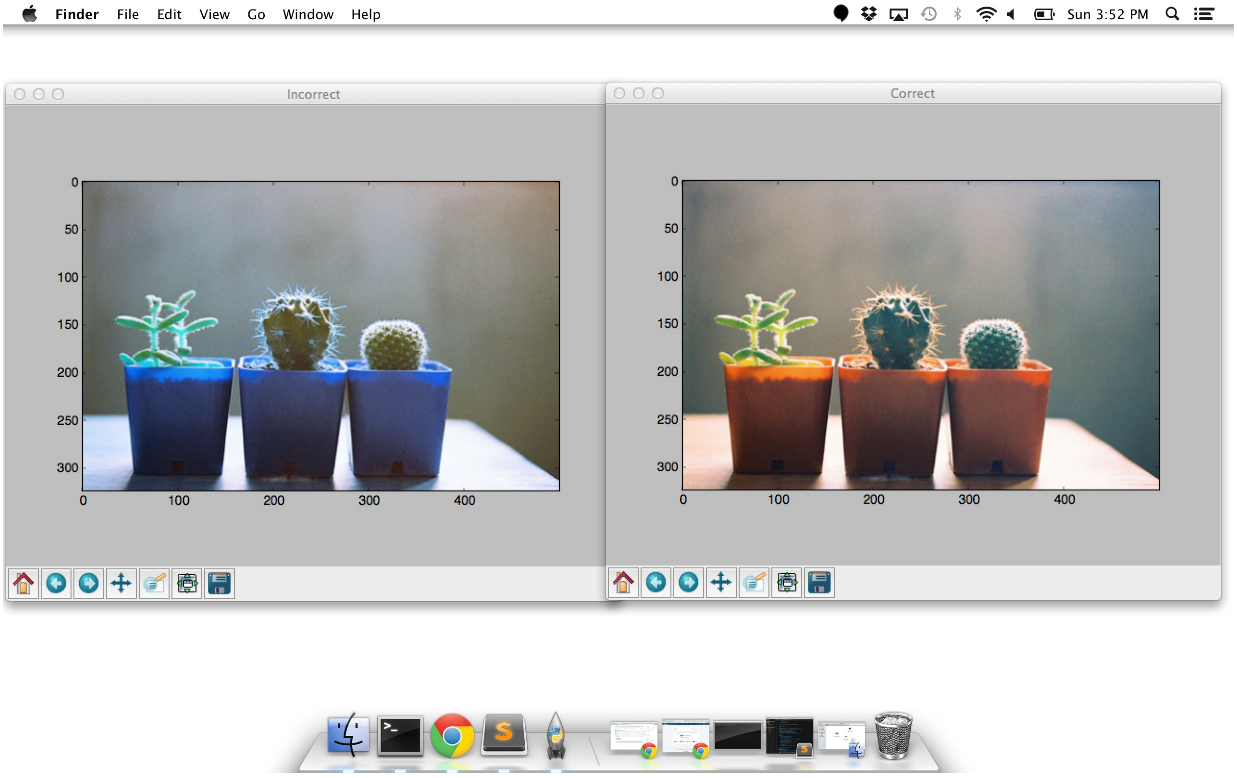
cv2.imshow
 function. However, if you intend on using Matplotlib, the
plt.imshow
 function assumes the image is in RGB order. A simple call to
cv2.cvtColor
 will resolve this problem, or you can use the
opencv2matplotlib
 convenience function.

Example:

I just open sourced my personal imutils package: A series of OpenCV convenience functions.

```
# INCORRECT: show the image without converting color spaces
plt.figure("Incorrect")
plt.imshow(cactus)
# CORRECT: convert color spaces before using plt.imshow
plt.figure("Correct")
plt.imshow(imutils.opencv2matplotlib(cactus))
plt.show()
```

Output:



Summary

So there you have it — the imutils package!

I hope you install it and give it a try. It will definitely make performing simple image processing tasks with OpenCV and Python substantially easier (and with less code).

In the coming weeks we'll perform a code review of each of the functions and discuss what is going on under the hood.

Until then!

Downloads:

Grab the imutils package from GitHub.