Pre-processing Examples

April 26, 2018

1 Cropping and Resizing

1.0.1 Import resources

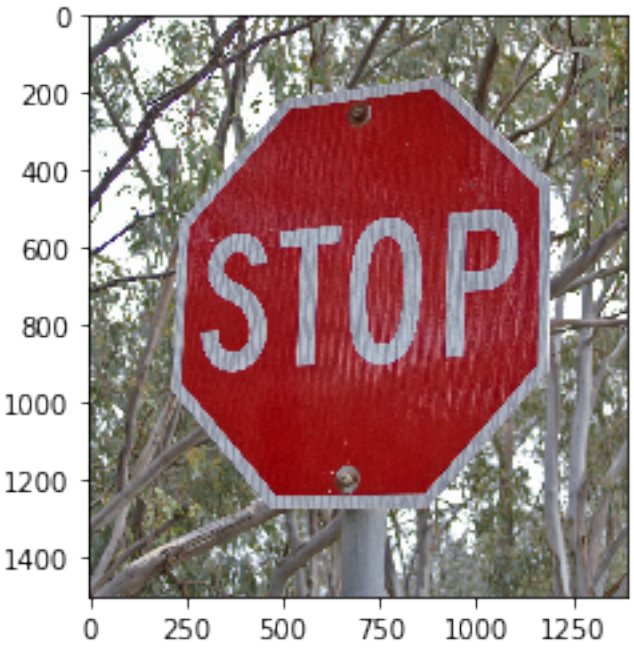
In [1]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

import numpy as np
import cv2

%matplotlib inline
```

1.0.2 Read in the first image of a stop sign

In [2]:
```
# Read in the image
stop1 = mpimg.imread('images/stop_sign.jpg')

print('Image shape: ', stop1.shape)
plt.imshow(stop1)
```

```
Image shape:  (1500, 1389, 3)
```

Out[2]:
```
<matplotlib.image.AxesImage at 0x7f6f2e1be0f0>
```



1.0.3 Read in the second image

```
In [3]: # Read in the image
        stop2 = mpimg.imread('images/stop_sign2.jpg')

        print('Image shape: ', stop2.shape)
        plt.imshow(stop2)
```

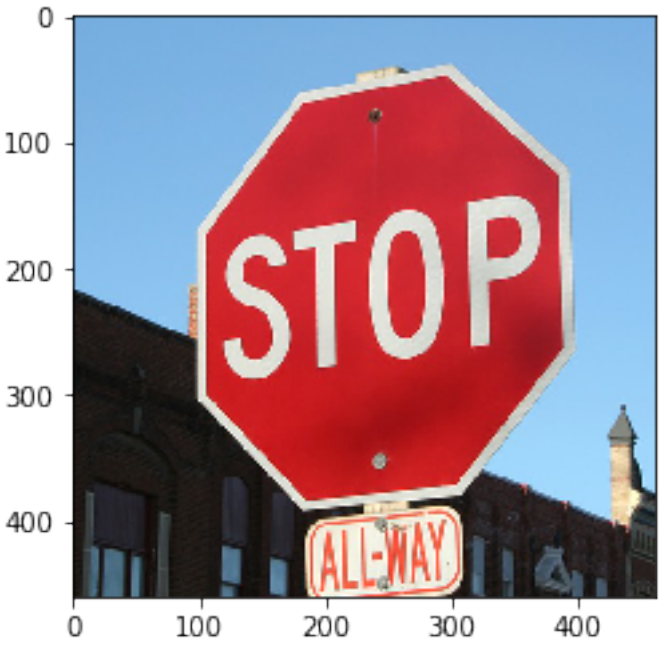
```
Image shape:  (640, 960, 3)
```

```
Out[3]: <matplotlib.image.AxesImage at 0x7f6f2e0de940>
```

1.1 Crop this image so that it resembles the first image

In [7]:
```
# To crop and image, you can use image slicing
# which is just slicing off a portion of the image array

# Make a copy of the image to manipulate
image_crop = np.copy(stop2)

# Define how many pixels to slice off the sides of the original image
row_crop = 90
col_crop = 250

# Using image slicing, subtract the row_crop from top/bottom and col_crop from left/righ
image_crop = stop2[row_crop:-row_crop, col_crop:-col_crop, :]

plt.imshow(image_crop)
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x7f6f2dea5c18>
```



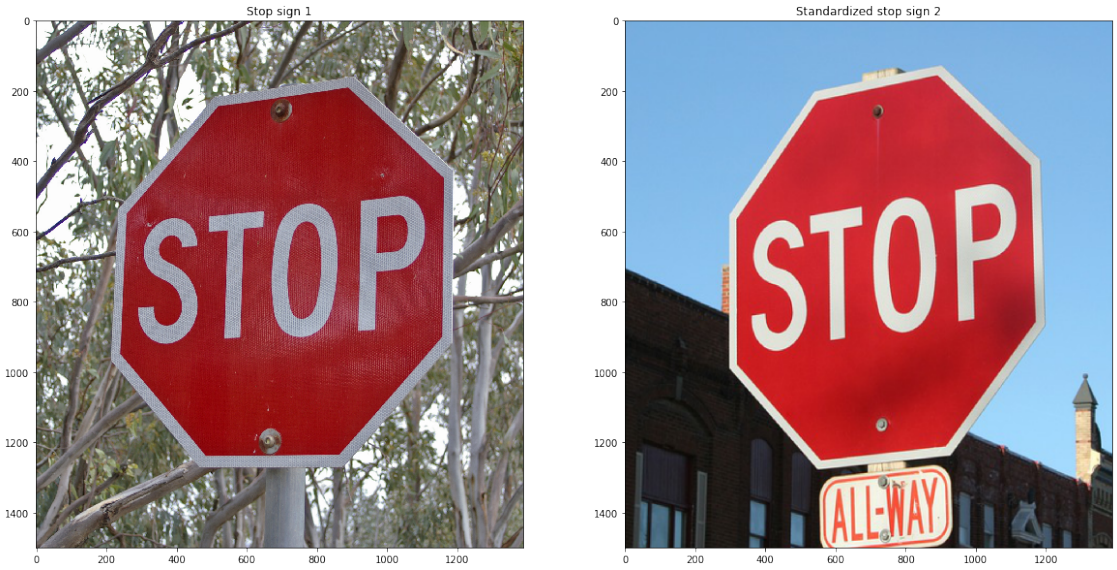
1.2 Resize the cropped image to be the same as the first

Recall that the shape of the first image is `(1500, 1389, 3)`.

```
In [8]: # Use OpenCV's resize function
        standardized_im = cv2.resize(image_crop, (1389, 1500))

        print('Image shape: ', standardized_im.shape)

        # Plot the two images side by side
        f, (ax1, ax2) = plt.subplots(1, 2, figsize=(20,10))
        ax1.set_title('Stop sign 1')
        ax1.imshow(stop1)
        ax2.set_title('Standardized stop sign 2')
        ax2.imshow(standardized_im)
```

```
Image shape:  (1500, 1389, 3)
```

```
Out[8]: <matplotlib.image.AxesImage at 0x7f6f2de3d780>
```



1.3 Compare these images

Now you should be able to compare these images pixel by pixel! We'll add up the red channel values in each image, and they should be fairly close, which means we can use this similarity to characterize these images.

For comparison, we'll also show what happens when you perform this comparison using the orginal `stop_sign2.jpg`.

```
In [9]: # Sum all the red channel values and compare
        red_sum1 = np.sum(stop1[:,:,0])
        red_sum2 = np.sum(standardized_im[:,:,0])

        print('Sum of all red pixel values in the first stop sign image: ', red_sum1)
        print('Sum of red pixel values in the second, standardized image: ', red_sum2)

        red_sum_orig = np.sum(stop2[:,:,0])

        print('\nFor comparison, the sum of red pixels in the non-standardized image: ', red_sum
```

```
Sum of all red pixel values in the first stop sign image:  294214944
Sum of red pixel values in the second, standardized image:  300109548

For comparison, the sum of red pixels in the non-standardized image:  67011798
```

```
In [1]: ## Note: you have been given two other images:
        # `yield.jpg` and `walk.jpg`
        # You can look at these images and see what kind of RGB values might distinguish them
```