Symbiosis_basic_open_cv

April 10, 2021

Ventrapragada Sai Shravani
17070123120
ENTC-B

Basic images operations with OpenCV Let's start by importing the OpenCV libary

[1]:
```
!pip install opencv-python
import cv2
import numpy as np
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.7/dist-
packages (4.1.2.30)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-
packages (from opencv-python) (1.19.5)
```

[5]:
```
  from google.colab.patches import cv2_imshow
```

Let's now load our first image

[8]:
```
# Load an image using 'imread' specifying the path to image
input_img= cv2.imread('/taj.jpg')



cv2_imshow(input_img)

# 'waitKey' allows us to input information when a image window is open
# By leaving it blank it just waits for anykey to be pressed before
# continuing. By placing numbers (except 0), we can specify a delay for
# how long you keep the window open (time is in milliseconds here)
cv2.waitKey()

# This closes all open windows
# Failure to place this will cause your program to hang
cv2.destroyAllWindows()
```

0.0.1 Let's take a closer look at how images are stored

[9]:
```
print(input_img.shape)
```

```
(2719, 3989, 3)
```

Shape gives the dimensions of the image array The 2D dimensions are 830 pixels in high bv 1245 pixels wide. The '3L' means that there are 3 other components (RGB) that make up this image.

[10]:
```
# Let's print each dimension of the image

print('Height of Image:', int(input_img.shape[0]), 'pixels')
print('Width of Image: ', int(input_img.shape[1]), 'pixels')
```

```
Height of Image: 2719 pixels
Width of Image:  3989 pixels
```

0.0.2 How do we save images we edit in OpenCV?

[11]:
```
# Simply use 'imwrite' specificing the file name and the image to be saved
cv2.imwrite('output.jpg', input_img)
cv2.imwrite('output.png', input_img)
```

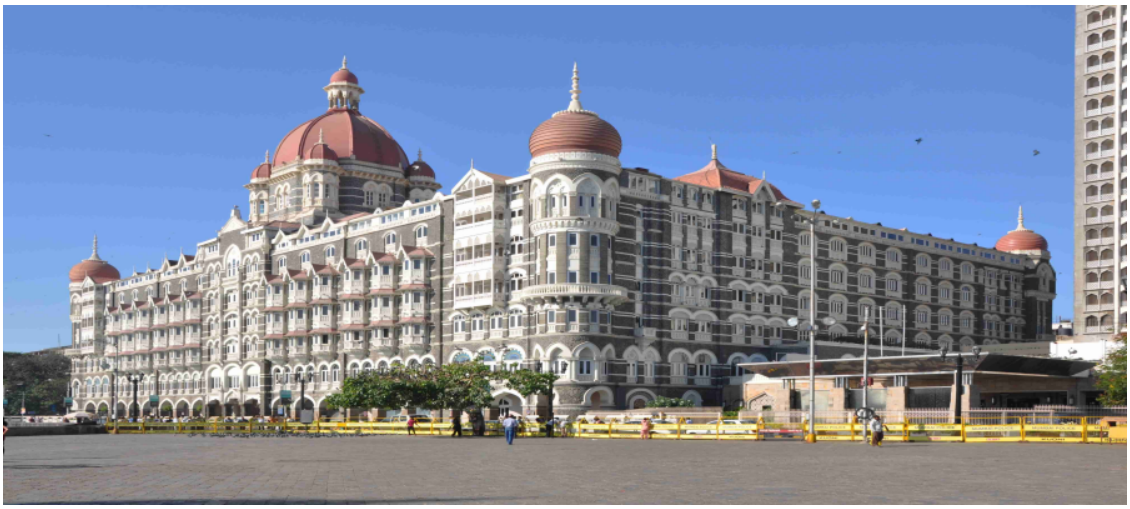
[11]:
```
True
```

1 Scaling and resizing

[12]:
```
dim=(150,150)
```

[14]:
```
image_scaled = cv2.resize(input_img,None,fx=0.75,fy=0.75)
cv2_imshow(image_scaled)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[15]:
```
img_scaled = cv2.resize(input_img, (900, 400), interpolation = cv2.INTER_AREA)
cv2_imshow(img_scaled)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



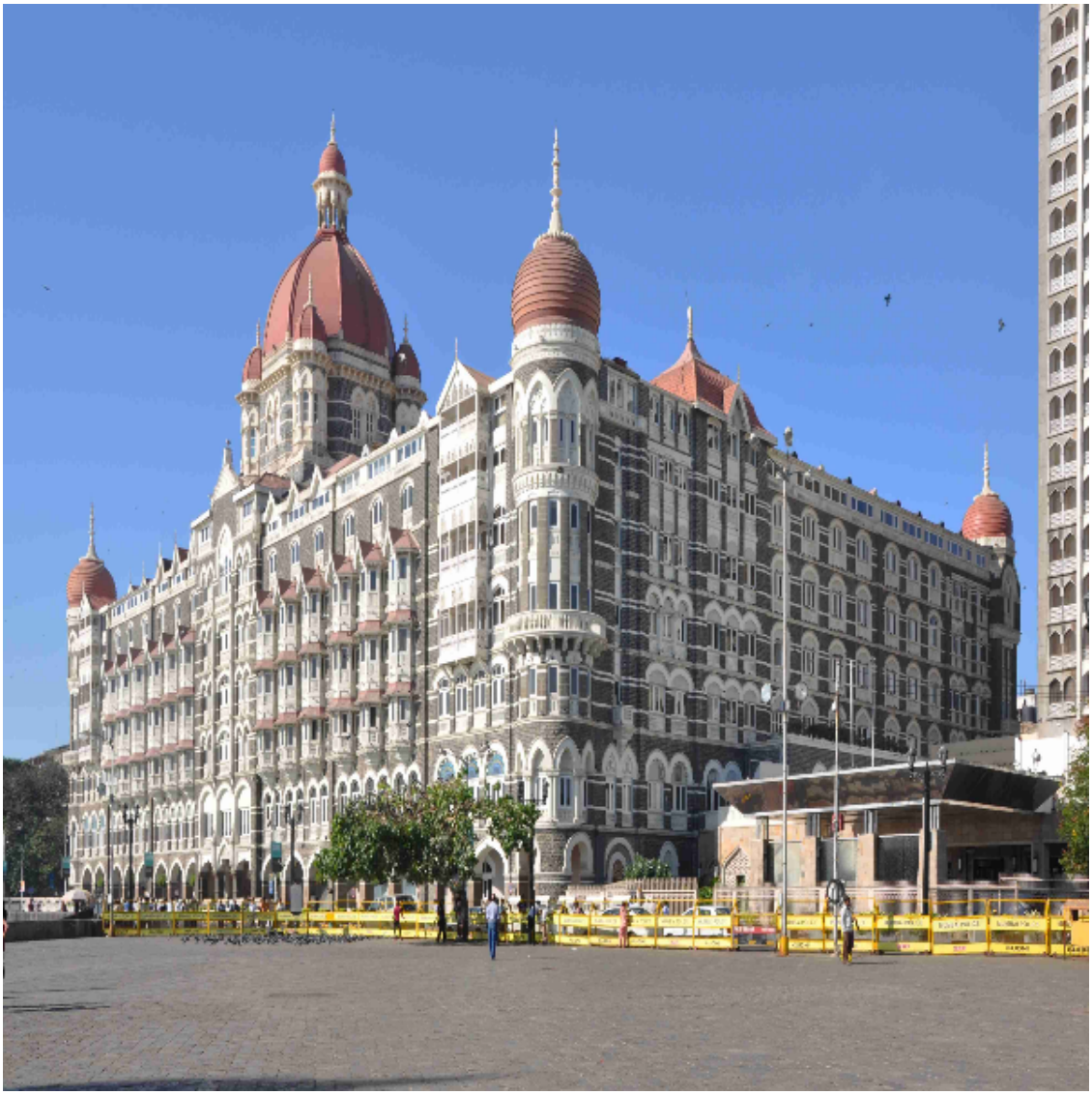
2 Grayscaling

[18]:
```
img=cv2.imread('/taj.jpg')

img=cv2.resize(img,(600,600))
cv2_imshow(img)
cv2.waitKey()

gray_img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
print('shape of the image: ',gray_img.shape)
cv2_imshow(gray_img)
cv2.waitKey()

cv2.destroyAllWindows()
```



[18]:
```
-1
```

3 Let's take a closer look at color spaces

You may have remembered we talked about images being stored in RGB (Red Green Blue) color Spaces. Let's take a look at that in OpenCV.

First thing to remember about OpenCV's RGB is that it's BGR (I know, this is annoying

[19]:
```
image = cv2.imread('/taj.jpg')
```

4 Individual pixel value

[20]:
```
B,G,R = image[0,0]
print('shape of the image',image.shape)
print('BGR value',B,G,R)
```

```
shape of the image (2719, 3989, 3)
BGR value 214 142 100
```

[21]:
```
g_i=cv2.imread('/taj.jpg',0)
gray_value=g_i[0,0]
print('shape of the image',g_i.shape)
print('gray value',gray_value)
```

```
shape of the image (2719, 3989)
gray value 138
```

4.1 HSV Value

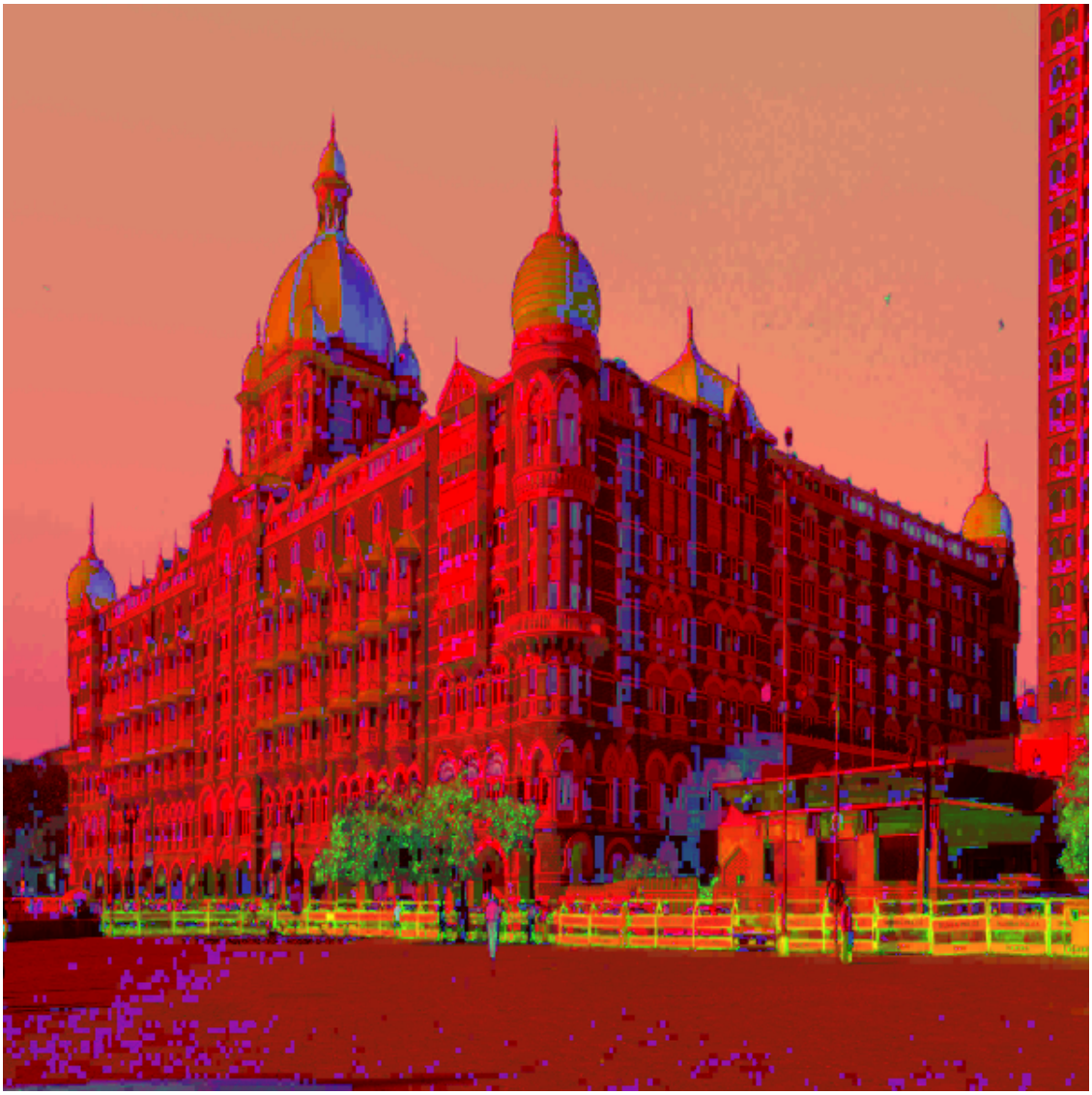
[24]:
```
#H: 0 - 180, S: 0 - 255, V: 0 - 255

image = cv2.imread('/taj.jpg')
image=cv2.resize(image,(600,600))

hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
print('HSV image')
cv2_imshow(hsv_image)
print('Hue channel')
cv2_imshow(hsv_image[:, :, 0])
print('Saturation channel')
cv2_imshow(hsv_image[:, :, 1])
print('Value channel')
cv2_imshow(hsv_image[:, :, 2])

cv2.waitKey()
cv2.destroyAllWindows()
```

```
HSV image
```



4.2 RGB individual color space

[25]:
```
image = cv2.imread('/taj.jpg')
image= cv2.resize(image,(600,600))

# OpenCV's 'split' function splites the image into each color index
B, G, R = cv2.split(image)
print(B.shape)
print("Red")
cv2_imshow(R)
print("Green")
cv2_imshow(G)
```

```
print("Blue")
cv2_imshow(B)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

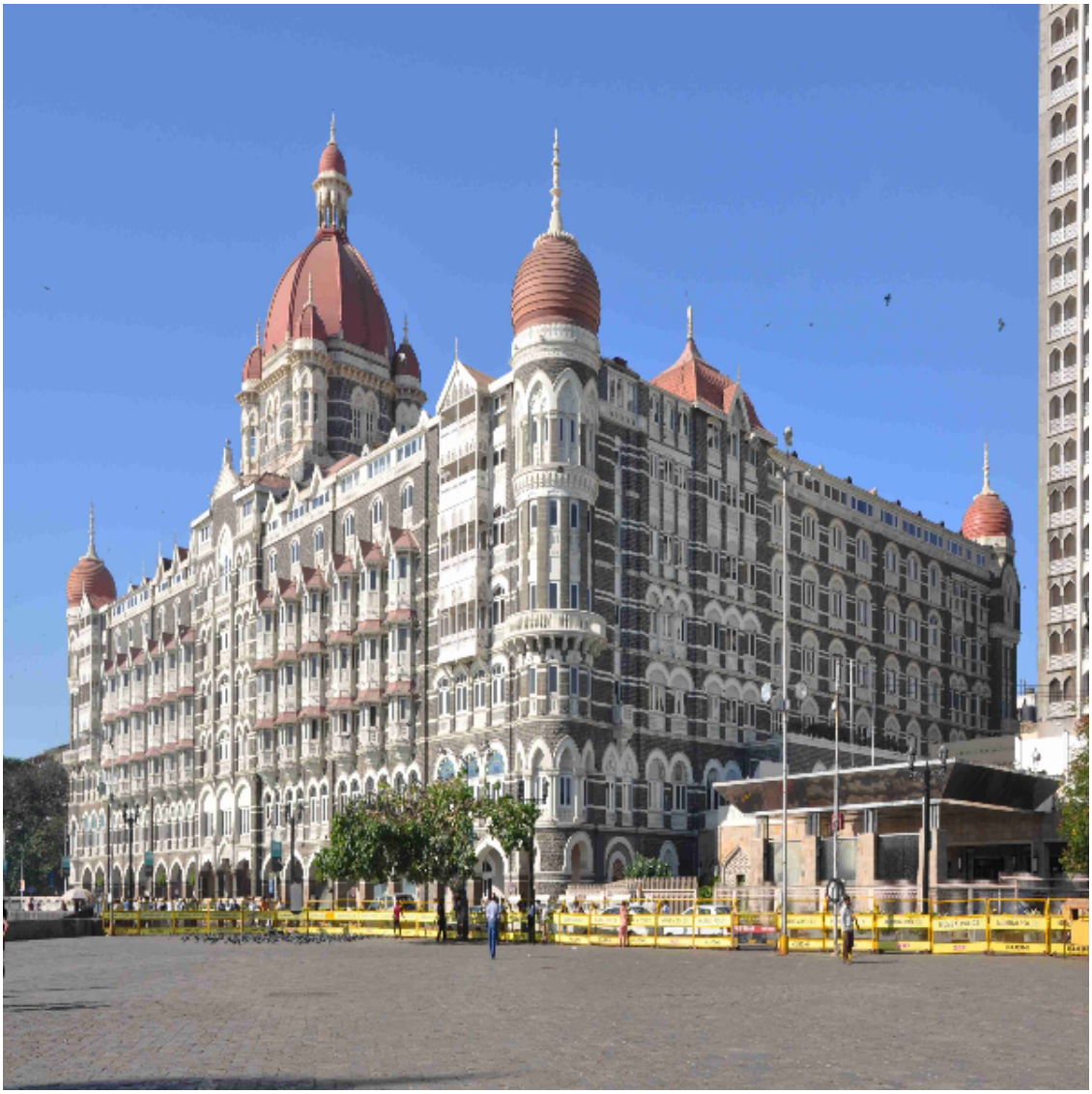
```
(600, 600)
Red
```

[26]:
```
# Let's re-make the original image,
merged = cv2.merge([B, G, R])
print("Merged")
cv2_imshow(merged)
```

```
# Let's amplify the blue color
merged = cv2.merge([B+100, G, R])
print("Merged with Blue Amplified")
cv2_imshow(merged)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Merged
```



[28]:
```
import cv2
import numpy as np
```

```
B, G, R = cv2.split(image)

# Let's create a matrix of zeros
# with dimensions of the image h x w
zeros = np.zeros(image.shape[:2], dtype = "uint8")
print("Red")
cv2_imshow(cv2.merge([zeros, zeros, R]))
print("Green")
cv2_imshow(cv2.merge([zeros, G, zeros]))
print("Blue")
cv2_imshow(cv2.merge([B, zeros, zeros]))

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Red
```

4.3 Task 1

[ ]:
```
# Load the image (download a new image of your choice)
# Convert the image from BGR to RGB  and save it (hint: BGR2RGB)
# Load the RGB saved image and convert split it to individual RGB color space␣
 ↪and convert each of individual color space to HSV color space
```

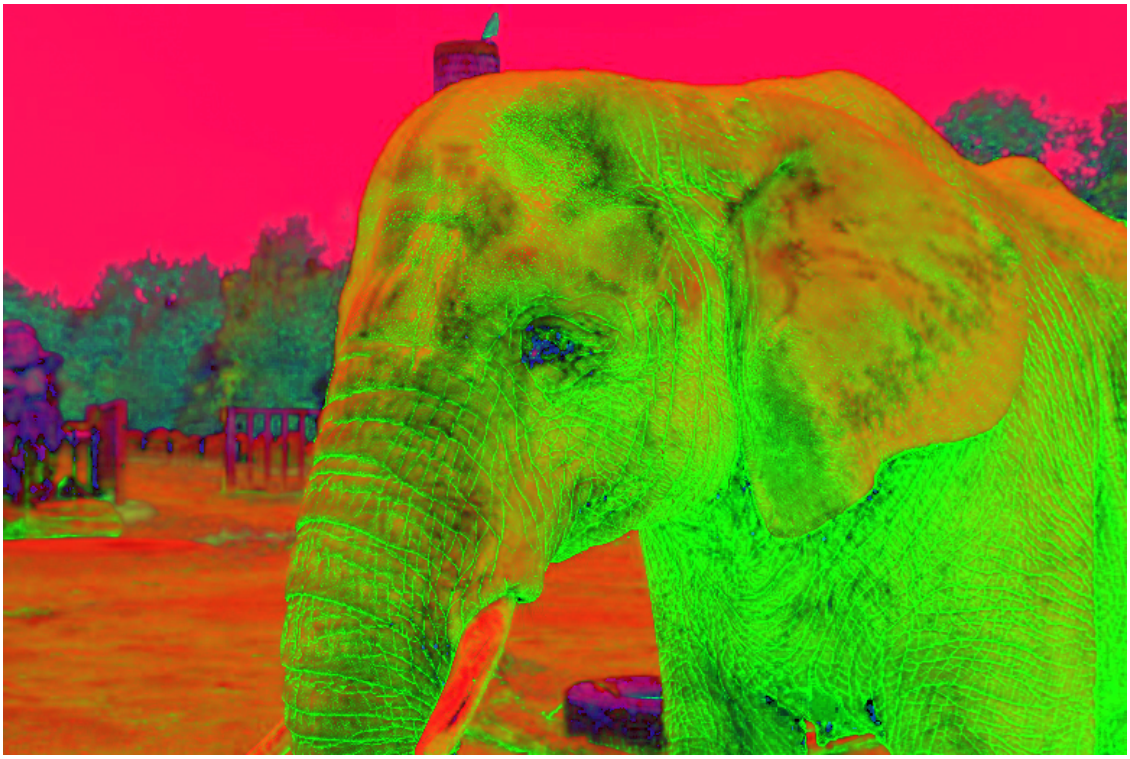
[31]:
```
new_img= cv2.imread('/elephant.jpg')
im_rgb = cv2.cvtColor(new_img, cv2.COLOR_BGR2RGB)
print("Original image")
cv2_imshow(new_img)
print("Converted from BGR to RGB")
cv2_imshow(im_rgb)
```

```
Original image
```

Load the RGB saved image and convert split it to individual RGB color space and convert each of individual color space to HSV color space

[ ]:
```
hsv_nemo = cv2.cvtColor(nemo, cv2.COLOR_RGB2HSV)
print("Converted from RGB")
cv2_imshow(im_rgb)
```

```
Converted from RGB to HSV
```



[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('pandas-assignment.ipynb')
```