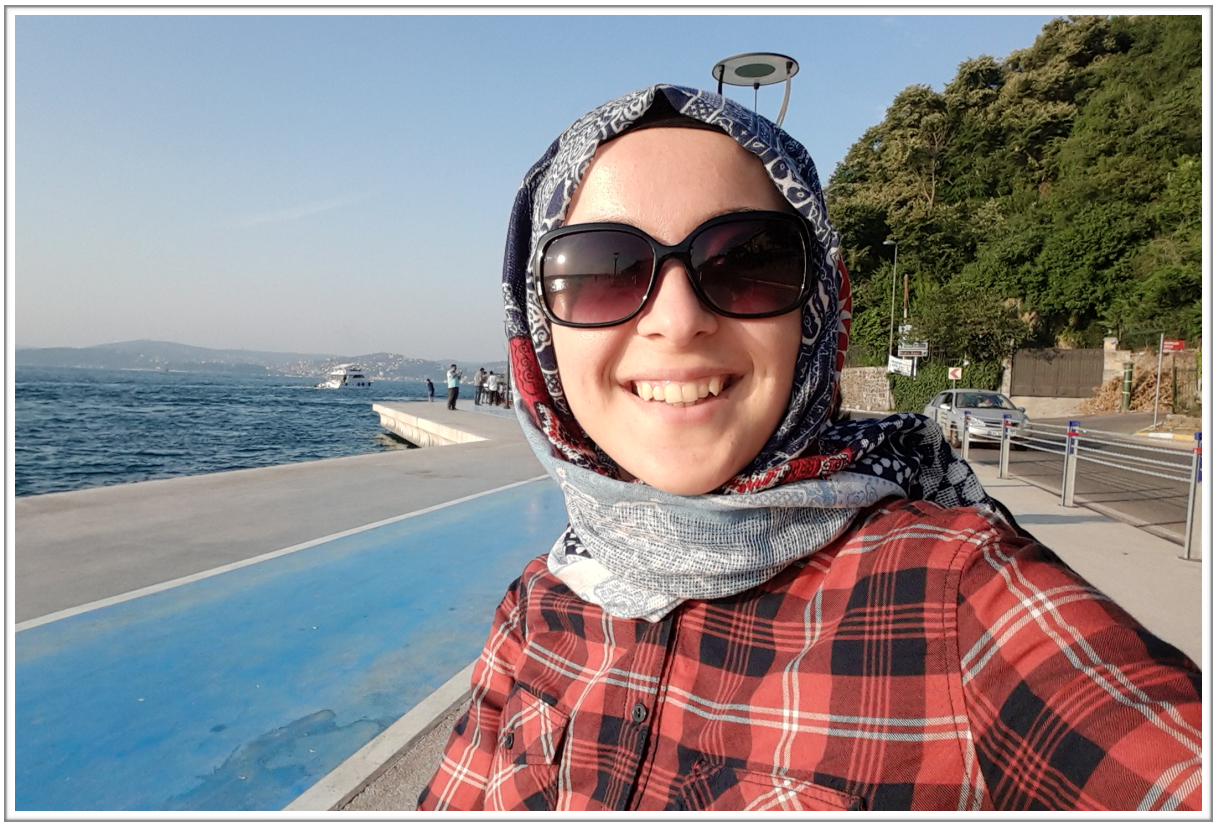
Computer Vision

homework 0



Tuğba Özkal

150120053

Homework 0 Report

Computer Vision

Question 1

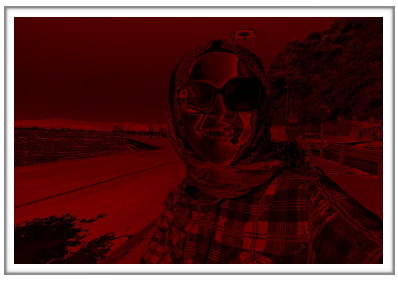
- Display each channel (Red, Green, and Blue) and their two-by-two combinations (Red+Green, Red+Blue, Green+Blue) separately.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('Icolor.jpg')
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

red=[[[0,0,255%j] for j in i] for i in img_gray]
dt = np.dtype('f8')
red = np.array(red, dtype=dt)

green=[[[0,255%j,0] for j in i] for i in img_gray]
dt = np.dtype('f8')
green = np.array(green, dtype=dt)

blue=[[[255%j,0,0] for j in i] for i in img_gray]
dt = np.dtype('f8')
```

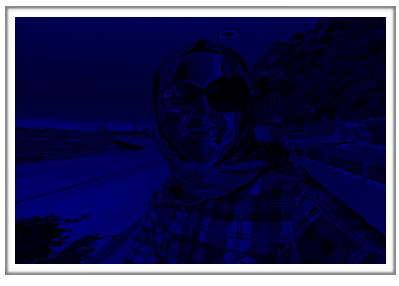
```
blue = np.array(blue, dtype=dt)

titles = ['RED','GREEN','BLUE']
images = [red, green, blue]

for i in range(3):
        plt.subplot(1,3,i+1),plt.imshow(images[i],'gray')
        plt.title(titles[i])
        plt.xticks([]),plt.yticks([])

plt.show()

cv2.imwrite('00255.jpg',red)
cv2.imwrite('02550.jpg',green)
cv2.imwrite('25500.jpg',blue)
```

In this script, open cv and numpy libraries, pyplot function from matplotlib library are imported. When OpenCV library has been used, images are read as BGR instead of RGB. So, if the values come as [255:0:0] on the grayscale image, blue seems instead red. If the values come as [0:0:255] on the grayscale image, red seems instead blue. Therefore, many codes are arranged according to this.

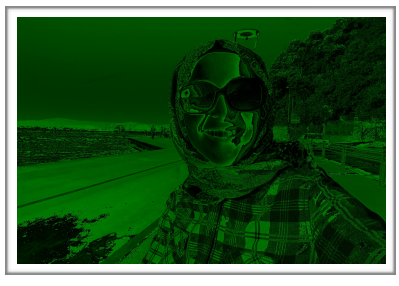
Icolor.jpg is read as 'img' and it is converted to grayscale image as 'togray'. The value of 'togray' array is changed and as a results, red, green and blue colors are obtained separately.



red



blue



green

```
import cv2
img = cv2.imread("Icolor.jpg")

#b:blue, g:green, r:red
b,g,r = cv2.split(img)
```

```
#blue-green-blue
bgb = cv2.merge((b,g,b))
cv2.imwrite("bgb.jpg",bgb)
#blue-green-green
bgg = cv2.merge((b,g,g))
cv2.imwrite("bgg.jpg",bgg)
#green-green-red
ggr = cv2.merge((g,g,r))
cv2.imwrite("ggr.jpg",ggr)
#red-green-red
rgr = cv2.merge((r,g,r))
cv2.imwrite("rgr.jpg",rgr)
#blue-red-red
brr = cv2.merge((b,r,r))
cv2.imwrite("brr.jpg",brr)
```

Here opencv library is imported and by using it the image which is named Icolor.jpg is read as BGR and splitted b(blue), g(green) and r(red). Then it is merged as blue-green, blue-red and green-red. There is 6 different images, because green replaces with red, blue replaces with red, green replaces with blue, red replaces with blue, blue replaces with green and blue replaces with green.

blue-blue-red

blue-green-blue

blue-green-green

blue-red-red

green-green-red

red-green-red

Question 2

- Create an average grayscale image from the color image of yourself and name it Igray. What is the range of this image (i.e. its min, max values)? With how many bits this image can be represented?

```
import cv2
import numpy as np

img = cv2.imread('Icolor.jpg')

togray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imwrite('Igray.jpg', togray)

avg_color_per_row = np.average(togray, axis=0)
avg_color = np.average(avg_color_per_row, axis=0)
print("avarage color: ", avg_color)

smallest = np.amin(img)
print ("image min: ", smallest)
smallest0 = np.amin(avg_color_per_row)
print ("avg_color_per_row min: ", smallest0)
smallest1 = np.amin(avg_color)
print ("avg_color min: ", smallest1)

biggest = np.amax(img)
print ("image max: ", biggest)
biggest0 = np.amax(avg_color_per_row)
print ("avg_color_per_row max: ", biggest0)
biggest1 = np.amax(avg_color)
print ("avg_color max: ", biggest1)
```

Igray.jpg

OpenCV and numpy libraries are used to calculate max and min values and to convert to grayscale image. Icolor image is read and converted to Igray image. Minimum and maximum values of average colors are calculated. The result is given below in figure.

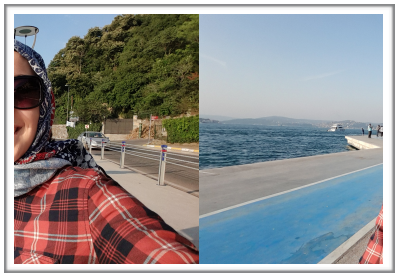
```
Q2 — -bash — 80×24
  File "average.py", line 7, in <module>
    togray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.error: /Users/travis/build/skvark/opencv-python/opencv/modules/imgproc/src/c
olor.cpp:10638: error: (-215) scn == 3 || scn == 4 in function cvtColor

Tugba-MBP:Q2 tugba$ python3 average.py
Invalid SOS parameters for sequential JPEG
avarage color:  130.674215766
image min:  0
avg_color_per_row min:  75.4965277778
avg_color min:  130.674215766
image max:  255
avg_color_per_row max:  175.373611111
avg_color max:  130.674215766
Tugba-MBP:Q2 tugba$ python3 average.py
Invalid SOS parameters for sequential JPEG
avarage color:  130.674215766
image min:  0
avg_color_per_row min:  75.4965277778
avg_color min:  130.674215766
image max:  255
avg_color_per_row max:  175.373611111
avg_color max:  130.674215766
Tugba-MBP:Q2 tugba$
```

figure 1

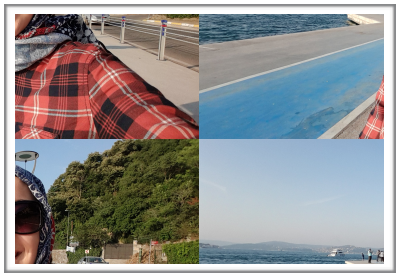
Question 3

- Flip your image horizontally and vertically to obtain your picture in a similar way as in the images of Figure 1.

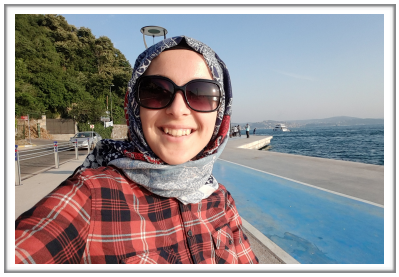
original



croppedw



croppedh



mirror

```
import numpy as np
import cv2

img = cv2.imread("Icolor.jpg")

width = img.shape[1]
height = img.shape[0]

halfw = int(width/2)
halfh = int(height/2)

crop_img1 = img[0:height, 0:halfw]
crop_img2 = img[0:height, halfw:width]
```

```
vis=np.concatenate((crop_img2, crop_img1), axis=1)
cv2.imwrite("croppedw.jpg", vis)

img2= cv2.imread("croppedw.jpg")
crop_img3 = img2[0:halfh, 0:width]
crop_img4 = img2[halfh:height, 0:width]
vis2=np.concatenate((crop_img4, crop_img3), axis=0)
cv2.imwrite("croppedh.jpg", vis2)

mimg=cv2.flip(img,1)
cv2.imwrite('mirror.jpg', mimg)
```

OpenCV and numpy libraries are also used here. Original image named Icolor.jpg is read as “img" and its both width and height are calculated in pixels. Image is cropped vertically firstly and then a new image named ‘croppedw.jpg' was created by replacing and concatenating two pieces. The same operation is applied horizontally. Created image name is ‘croppedh.jpg’.

At the end, original image is flipped and it is written as 'mirror.jpg'.