Computer Vision

LAB 1

Submitted By:

Gaurav Agarwal	**BT17CSE001**
Abhibha Gupta	**BT17CSE020**
Thogaru Himabindu	**BT17CSE056**

Tasks:

1. Capture images using your webcam.
2. Check how to find the color space of an image.
3. Perform several basic operations on the images.
    1. Resizing
    2. Plot Histogram
    3. Perform Affine operations on images

The original pic used in the whole assignment is given below:

Original pic:

Task 1

Code

```
import cv2

#because only one camera so '0', if more than one cams then '1', etc
cam = cv2.VideoCapture(0)

cv2.namedWindow("test")
# to name images
img_counter = 0

while True:
   # Capture frame-by-frame, returns bool value,
    #True if it has captured frame successfully
   #ret stores return value, and frame is a numpy array of dim (480, 848,
3)
   ret, frame = cam.read()
   #print(frame.shape)
   if not ret:
       print("failed to grab frame")
       break
   cv2.imshow("test", frame)

   #keyboard binding function,argument:
   #time, here waits for 1 sec to detect key press
   k = cv2.waitKey(1)
   if k%256 == 27:
       # ESC pressed
       print("Escape hit, closing...")
       break
   elif k%256 == 32:
```

```
        # SPACE pressed
        img_name = "opencv_frame_{}.png".format(img_counter)
        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img_counter += 1

cam.release()

cv2.destroyAllWindows()
```

Output frame:

Task 2

Code

```
import cv2
import numpy as np
image = cv2.imread('opencv_frame_0.png')

print('no of channels:',image)
b_channel, g_channel, r_channel = cv2.split(image)
```

```
alpha_channel = np.ones(b_channel.shape, dtype=b_channel.dtype) * 50
#creating a dummy alpha channel image.

img_BGRA = cv2.merge((b_channel, g_channel, r_channel, alpha_channel))

print('after adding alpha channel:',img_BGRA.shape)

B, G, R, A = cv2.split(img_BGRA)
#Corresponding channels are seperated

cv2.imshow("original", image)
cv2.waitKey(0)
cv2.imshow("blue", B)
cv2.waitKey(0)
cv2.imshow("green", G)
cv2.waitKey(0)
cv2.imshow("red", R)
cv2.waitKey(0)

#alpha channel
cv2.imshow("alpha", A)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Blue:

green:

red:

Task 3

```
                                    Resizing:
import cv2
img = cv2.imread('opencv_frame_0.png')

print('Original Dimensions : ',img.shape)

cv2.imshow("Original image", img)
cv2.waitKey(0)
scale_percent = 120 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
 # resize image
resized = cv2.resize(img, dim)
print('Resized Dimensions : ',resized.shape)
cv2.imshow("Resized image", resized)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

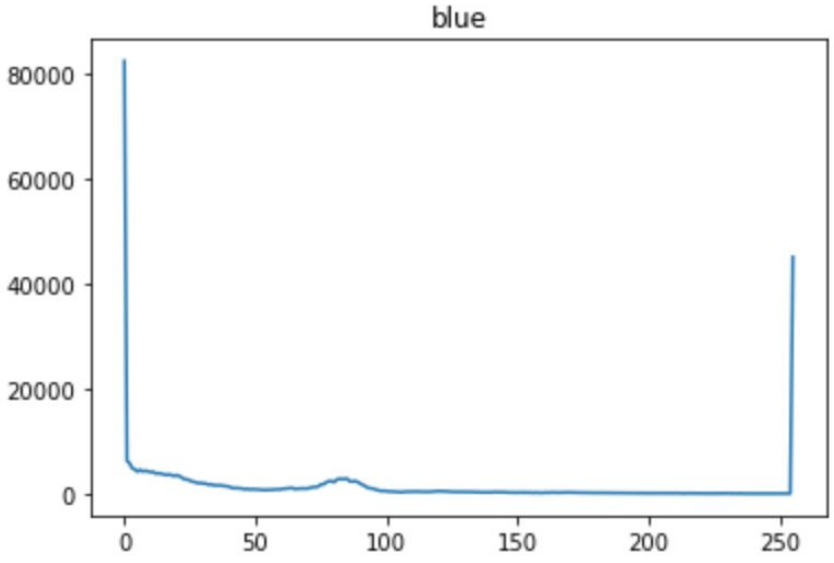
Resized Image:

Plot Histogram

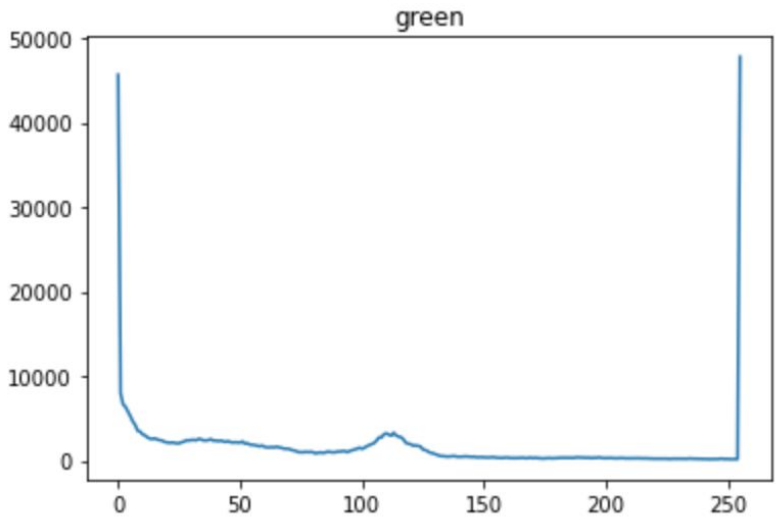
```
import cv2
img = cv2.imread('opencv_frame_0.png')
histr1 = cv2.calcHist([img],[0],None,[256],[0,256]) #blue
histr2 = cv2.calcHist([img],[1],None,[256],[0,256]) #green
histr3 = cv2.calcHist([img],[2],None,[256],[0,256]) #red

# show the plotting graph of an image
plt.plot(histr1)
plt.title('blue')
plt.show()
plt.plot(histr2)
plt.title('green')
plt.show()
plt.plot(histr3)
plt.title('red')
plt.show()
```

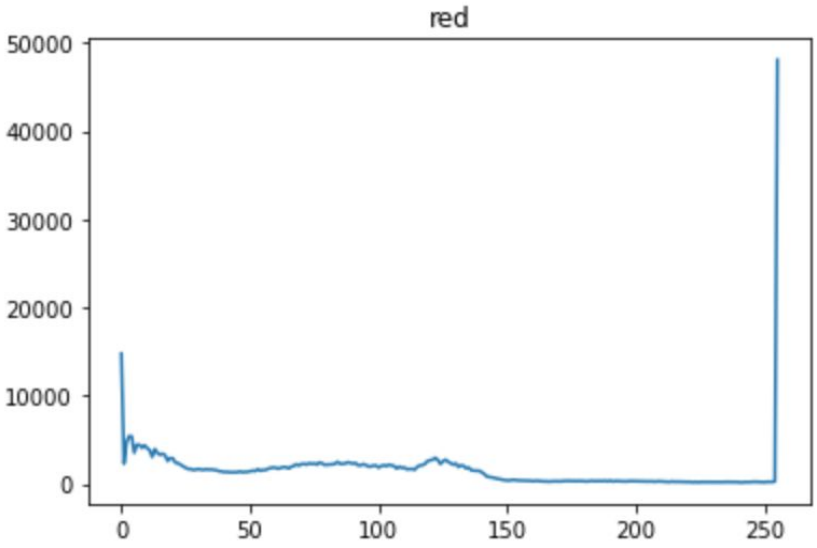
Blue Histogram:



Green Histogram:



Red Histogram:



Scaling

```
import cv2
img = cv2.imread('opencv_frame_0.png')
height, width = img.shape[:2]
res = cv2.resize(img,(int(0.5*width), int(0.5*height)), interpolation =
cv2.INTER_CUBIC)# a bicubic interpolation over 4×4 pixel neighborhood
```

```
cv2.imshow('img',res)
cv2.waitKey(0)
```

Scaled Image:

Translation

```
import cv2
img = cv2.imread('opencv_frame_0.png')
img_=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow('img',img_)
cv2.waitKey(0)
rows,cols = img_.shape

M = np.float32([[1,0,100],[0,1,50]]) #translation coordinates
dst = cv2.warpAffine(img_,M,(cols,rows))

cv2.imshow('img',dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Translated Image:

Rotation

```
#rotation
import cv2
import numpy as np
img = cv2.imread('opencv_frame_0.png', cv2.IMREAD_UNCHANGED)

img_=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
rows,cols = img_.shape
cv2.imshow('img',img_)
cv2.waitKey(0)
M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1) #center, angle, scale
print(M)
dst = cv2.warpAffine(img_,M,(cols,rows))
cv2.imshow('img',dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Rotated Image:

Affine Transformation

```
#affine transform
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('opencv_frame_0.png')
rows,cols,ch = img.shape

pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])
# finding the 2x3 matrix M
M = cv2.getAffineTransform(pts1,pts2)
#passing M as an argument to warpAffine function -> Applies an affine transformation to an image.
#The function warpAffine transforms the source image using the specified matrix M
dst = cv2.warpAffine(img,M,(cols,rows))
```

```
plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.show()
```

Affine Transformation: