Computer Vision & Image Processing

Term Assignment

Submitted to:

Sir Usman Ghani

Submitted by:

2018-CS-80

Abbas Ali

Section B

Computer Science and Engineering Department

University of Engineering and Technology, Lahore

TERM ASSIGNMENT

Link to the Code:

Google_Colab_Source_Code

1. Read an image

Description:

Here Reading the Image from the file using maplotlib library.

Code:

```
1. import matplotlib.image as mpimg
2. import matplotlib.pyplot as plt
3.
4. #Reading the Image
5. img = mpimg.imread('/content/SceneImage.jpeg')
```

2. Display an image

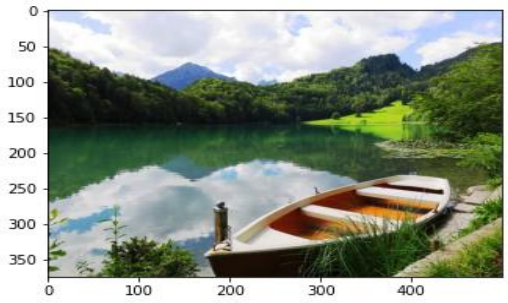
Description:

Here Showing the Image read from the File using matplotlib.

Code:

```
1. plt.imshow(img)
```

Output:



3. Save image to disk

Description:

Here saving the into another file using matplotlib

Code:

```
1. mpimg.imsave('/content/Stored_Screenshot (1).png',img)
```

4. Resize image

Description:

In this Task resizing the image by using the resize method of opencv

Code:

```
1. import cv2
2. from google.colab.patches import cv2_imshow
3.
4. src = cv2.imread('/content/Stored_Screenshot (1).png', cv2.IMREAD
   _UNCHANGED)
5. # newSize
6. dsize = (100, 100)
7.
8. # resize image
9. output = cv2.resize(src, dsize)
10.   cv2_imshow(output)
```

Output:

5. Get image shape

Description:

Getting the Shape of the Image by get the property of the shape of image that is the instance of opencv library.

Code:

```
1. print(output.shape)
```

Output:

```
(100, 100, 4)
```

6. Put text on image

Description:

Here making copy of the Image and writing the text on it by using the opencv that of a specific font and the origin is given to where the text will be printed.

Code:

```
1. textImage = src.copy()
2. textImg=cv2.putText(img=textImage, text='Beautiful', org=(10,200)
   , fontFace=cv2.FONT_HERSHEY_TRIPLEX, fontScale=3, color=(0, 255,
   0),thickness=3)
3. cv2_imshow(textImg)
```

Output:

7. Draw a line on image

Description:

Here making copy of the Image and drawing line on it by using the opencv that of a specific font and the origin is given to where the line will be drowned.

Code:

```
1. lineSrc=src.copy()
2. cv2.line(
3.                lineSrc,
4.                pt1 = (200,100), pt2 = (200,200),
5.                color = (0,0,0),
6.                thickness = 10
7.             )
8. cv2_imshow(lineSrc)
```

Output:

8. Draw a rectangle on image

Description:

Here making copy of the Image and drawing Rectangle on it by using the opencv that of a specific font and the origin is given to where the rectangle will be drowned.

Code:

```
1. imgrect=src.copy()
2. cv2.rectangle(imgrect,(50,50),(250,150),(0,255,0),3)
3. cv2_imshow(imgrect)
```

Output:

9. Draw a circle on image

Description:

Here making copy of the Image and drawing Circle on it by using the opencv that of a specific font and the origin is given to where the circle will be printed.

Code:

```
1. imgcir=src.copy()
2. cv2.circle(imgcir,(150,150), 63, (0,0,255), -1)
3. cv2_imshow(imgcir)
```

Output:

10. Draw a square on image

Description:

Here making copy of the Image and drawing Square on it by using the opencv that of a specific font and the origin is given to where the Square will be drowned.

Code:

```
1. imgsquare=src.copy()
2. cv2.rectangle(imgsquare,(50,50),(150,150),(0,255,0),3)
3. cv2_imshow(imgsquare)
```

Output:

11. Draw a triangle on image

Description:

Here making copy of the Image and drawing Triangle on it by using the opencv that of a specific font and the origin is given to where the Triangle will be drowned.

Code:

```
imgtri=src.copy()
# Three vertices(tuples) of the triangle
p1 = (100, 200)
p2 = (50, 50)
p3 = (300, 100)

# Drawing the triangle with the help of lines
#  on the black window With given points
# cv2.line is the inbuilt function in opencv library
   cv2.line(imgtri, p1, p2, (255, 0, 0), 3)
   cv2.line(imgtri, p2, p3, (255, 0, 0), 3)
   cv2.line(imgtri, p1, p3, (255, 0, 0), 3)
```

```
13.   cv2_imshow(imgtri)
```

Output:

12. Convert RGB Image to Grayscale

Description:

Here converting the color image that has 4 channels into the gray scale image that will have only one channel

Code:

```
1. imgGray=src.copy()
2. imgGray = cv2.cvtColor(imgGray, cv2.COLOR_BGR2GRAY)
3. cv2_imshow(imgGray)
```

Output:

13. Capture livestream from Webcam

Description:

Here getting the image from the webcam and then using some html code to convert for showing into google colab.

Code:

```
start_input()
label_html = 'Capturing...'
img_data = ''
count = 0
while True:
    js_reply = take_photo(label_html, img_data)
    if not js_reply:
        break

    image = js_reply_to_image(js_reply)
```

Output:

14. Read Video from disk

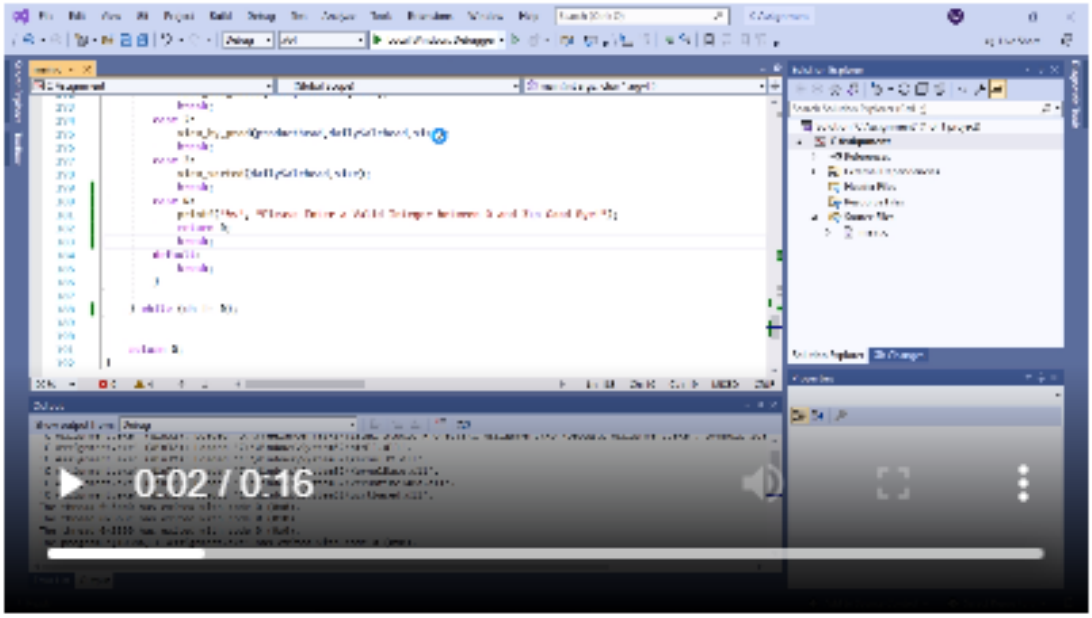
Description:

Here reading the video file from a hard and then using some HTML to show the video on google colab.

Code:

```
from IPython.display import HTML
from base64 import b64encode
mp4 = open('/content/2021-12-11-00-33-07.mp4','rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
HTML("""
<video width=400 controls>
      <source src="%s" type="video/mp4">
</video>
""" % data_url)
```

Output:



15. Blur an image

Description:

Blurring the image by using the opencv blur method and then showing the output.

Code:

```
1. blurImg=src.copy()
2. blurImg = cv2.blur(blurImg,(10,10))
3. cv2_imshow(blurImg)
```

Output:

16. Detect Edges of objects in an image

Description:

Here using the Sobel Operator from the opencv to detect the edges inside the Images.

Code:

```
1. img_gray = cv2.cvtColor(src.copy(), cv2.COLOR_BGR2GRAY)
2. img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
3.
4. sobel = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ks
   ize=5)
5. cv2_imshow(sobel)
```

Output:

17. Detect Contours in an image

Description:

Here using the opencv methods to get the contours inside the image

Code:

```
# convert to RGB
image = cv2.cvtColor(src.copy(), cv2.COLOR_BGR2RGB)
# convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
_, binary = cv2.threshold(gray, 225, 255, cv2.THRESH_BINARY_INV)
# show it
contours, hierarchy = cv2.findContours(binary, cv2.RETR_TREE, cv2
.CHAIN_APPROX_SIMPLE)
# draw all contours
image = cv2.drawContours(image, contours, -1, (0, 255, 0), 2)
  cv2_imshow(image)
```

Output:

18. Crop an image

Description:

Getting the cropped image by slicing the original array and then will get the cropped image of the Original image and will use this image as a template in the next Tasks.

Code:

```
1. imgCrp=src.copy()
2. croped = imgCrp[200:400,200:400]
3. cv2_imshow(croped)
```

Output:

19. Sharpen the image

Description:

Using a filter that has the central value more weightage than other and central pixel contribute more than any other pixel in the output Image. At the end we get the sharper Image.

Code:

```
1. kernel = np.array([[0, -1, 0],
2.                    [-1, 5,-1],
3.                    [0, -1, 0]])
4. image_sharp = cv2.filter2D(src=src.copy(), ddepth=-
   1, kernel=kernel)
5. cv2_imshow(image_sharp)
```

Output:

20. Apply an identity Filter on an image

Description:

Here Applying the Identity Filter using opencv.

Code:

```
1. kernelidentity =  np.array([[0, 0, 0],
2.                     [0, 1, 0],
3.                     [0, 0, 0]])
4.
5. image_identity = cv2.filter2D(src=src.copy(), ddepth=-
   1, kernel=kernelidentity)
6. cv2_imshow(image_identity)
```

Output:

21. Apply Gaussian Filter on an image

Description:

Here making a Gaussian Filter with size of 5 by 5. And then applying on the image using opencv and then showing the resultant Image.

Code:

```
1. kernelgaussian =  np.array([[0, 0, 1/16,0,0],
2.                      [0,0, 2/16, 0,0],
3.                      [1/16, 2/16, 4/16,2/16,1/16],
4.                      [0, 0, 2/16,0,0],
5.                      [0,0, 1/16, 0,0]])
6. gaussian_blur=cv2.filter2D(src=src.copy(), ddepth=-
   1, kernel=kernelgaussian)
7. cv2_imshow(gaussian_blur)
```

Output:

22. Apply Median Filter on an image

Description:

Here applying the Median Filter using opencv and then showing it.

Code:

```
1. median_blur = cv2.medianBlur(src.copy(),5)
2. cv2_imshow(median_blur)
```

Output:

23. Apply Average Filter on an image

Description:

Here applying the Average filter using opencv and showing on the screen.

Code:

```
1. avgFilter=cv2.boxFilter(img, -1, (10, 10), normalize=True)
2. cv2_imshow(avgFilter)
```

Output:

24. Draw Histogram of an image

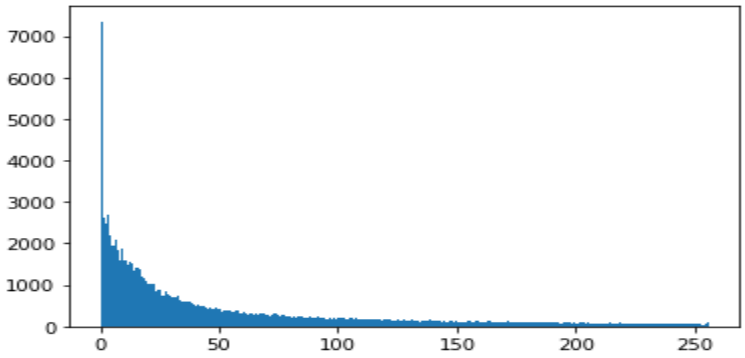
Description:

Here showing the histogram of the image using the matplotlib library and showing the results.

Code:

```
1. plt.hist(sobel.ravel(),256,[0,256])
2. plt.show()
```

Output:



25. Perform template matching on an image

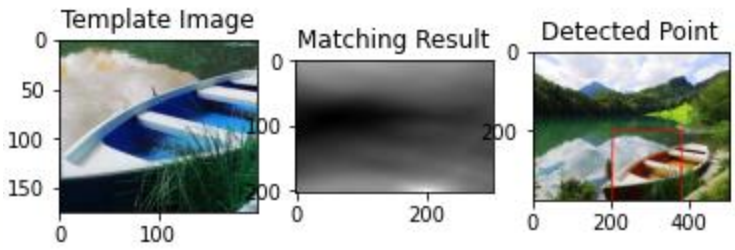
Description:

Performing the template matching an image. In this will use the cropped image of the original image as a template and will do the template matching on that image using opencv.

Code:

```
original=src.copy()
w, h = croped.shape[:-1]
res = cv2.matchTemplate(original,croped,cv2.TM_CCOEFF)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(img,top_left, bottom_right, 255, 2)
plt.subplot(231)
plt.title('Template Image')
    plt.imshow(croped)
    plt.subplot(232)
    plt.imshow(res,cmap = 'gray')
    plt.title('Matching Result')
    plt.subplot(233)
    plt.imshow(img,cmap = 'gray')
    plt.title('Detected Point')
    plt.show()
```

Output:



26. Perform Negative of an Image

Description:

Here looping through the whole image pixel and then subtracting it from the 255 to convert into negative and in this way will get the negative of the image.

Code:

```
1. img=src.copy()
2. cv2_imshow(img)
3. # Read pixels and apply negative transformation
4. for i in range(0, img.shape[0]-1):
5.     for j in range(0, img.shape[1]-1):
6.         # Get pixel value at (x,y) position of the image
7.         pixelColorVals = img[i,j];
8.         # Invert color
9.         redPixel    = 255 - pixelColorVals[0]; # Negate red pixel
10.           greenPixel  = 255 - pixelColorVals[1]; # Negate green
   pixel
11.           bluePixel   = 255 - pixelColorVals[2]; # Negate blue p
   ixel
12.           # Modify the image with the inverted pixel values
13.           img[i,j][0]=redPixel
14.           img[i,j][1]=greenPixel
15.           img[i,j][2]=bluePixel
16.
17.   # Display the negative image
18.   cv2_imshow(img)
```

Output:

27. Perform Thresholding on an image

Description:

Here using the opencv function to add threshold on an image and then showing the results.

Code:

```
1. img = cv2.cvtColor(src.copy(), cv2.COLOR_BGR2GRAY)
2. ret, thresh = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)
3. cv2_imshow(thresh)
```

Output:

28. Apply Log Transformation on an image

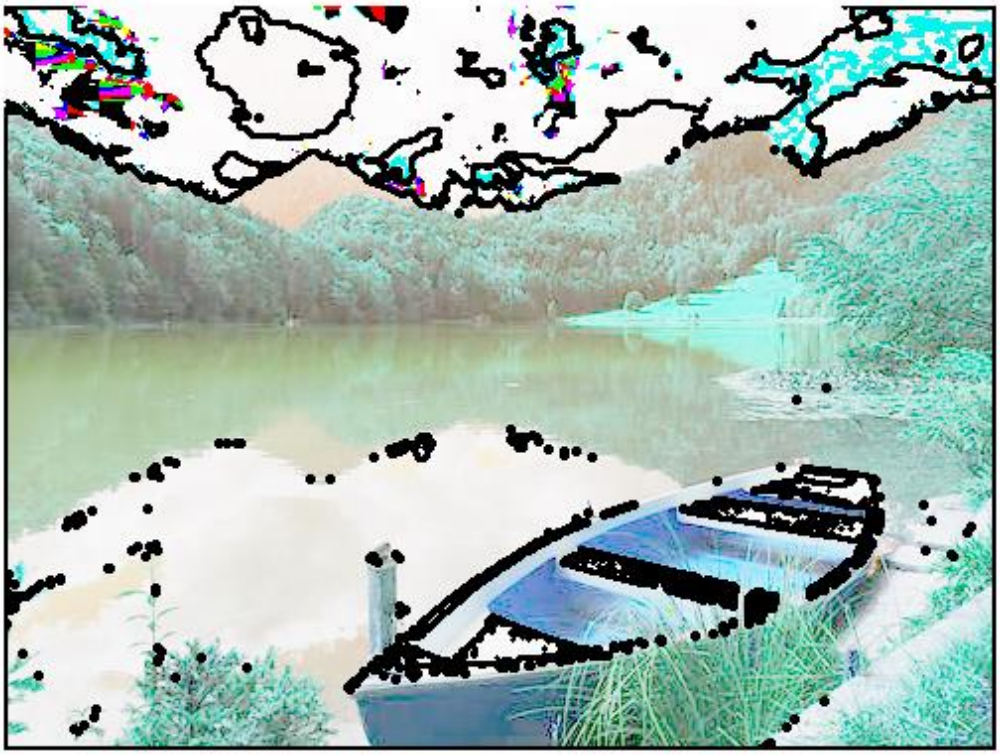
Description:

Here applying the formula of log transformation on each pixel of image and the resultant image will log transformed.

Code:

```
1. c = 255 / np.log(1 + np.max(src.copy()))
2. log_image = c * (np.log(image + 1))
3. log_image = np.array(log_image, dtype = np.uint8)
4. cv2_imshow(log_image)
```

Output:



29. Apply Power Law Transformation on image

Description:

Here performing the Power law transformation and using the gamma value of 0.5 and the showing the results.

Code:

```
1. # Apply gamma correction.
2. gamma_corrected = np.array(255*(src.copy() / 255) ** 0.5, dtype =
    'uint8')
3.
4. # Save edited images.
5. cv2_imshow(gamma_corrected)
```

Output:

30. Apply Erosion

Description:

Here performing the morphological operation on an image. Performing the Erosion operation will make all objects in the image shrink. The filter hare we are using the filter with all one in it.

Code:

```
1. kernel = np.ones((5,5), np.uint8)
2. img_erosion = cv2.erode(src.copy(), kernel, iterations=1)
3. cv2_imshow(img_erosion)
```

Output:

31. Apply Dilation

Description:

Here performing the morphological operation on an image. Performing the Dilation operation will make all objects in the image spread. The filter hare we are using the filter with all one in it.

Code:

```
1. img_dilation = cv2.dilate(src.copy(), kernel, iterations=1)
2. cv2_imshow(img_dilation)
```

Output:

32. Apply a compound Opening Operation

Description:

Here performing the morphological operation on an image. Performing the Opening operation will make all objects in the clear and not overlapping. The filter hare we are using the filter with all one in it.

Code:

```
1. opening = cv2.morphologyEx(src.copy(), cv2.MORPH_OPEN, kernel)
2. cv2_imshow(opening)
```

Output:

33. Apply a compound Closing Operation

Description:

Here performing the morphological operation on an image. Performing the Closing operation will make all objects in the image overlapping each other's. The filter hare we are using the filter with all one in it.

Code:

```
1. closing = cv2.morphologyEx(src.copy(), cv2.MORPH_CLOSE, kernel)
2. cv2_imshow(closing)
```

Output:

34. Boundary Extraction

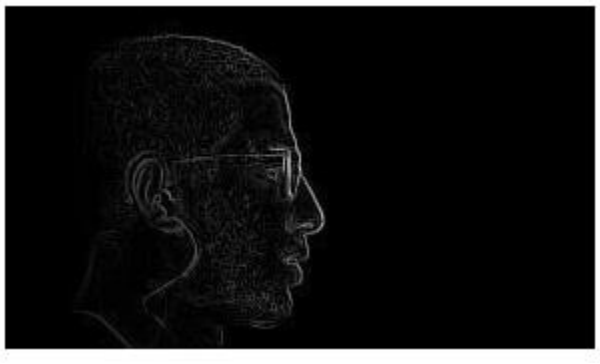
Description:

Here performing the morphological operation on an image to get the boundary of object inside the image. Performing the dilation operation will make all objects in the image shrink. And then subtracting it from original Image will result in a image having boundary lines. The filter hare we are using the filter with all one in it.

Code:

```
1. imagePerson=cv2.imread('/content/person.jpg')
2. kernelboundry = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))
3. dilate = cv2.dilate(imagePerson,kernelboundry,iterations = 1)
4. ebe = np.subtract(dilate, imagePerson)
5. cv2_imshow(ebe)
```

Output:



35. Region Filling

Description:

Here performing the morphological operation on an image to fill the holes inside an object in the image. The filter hare we are using the filter with all one in it.

Code:

```
gray = cv2.cvtColor(imagePerson.copy(), cv2.COLOR_BGR2GRAY)
cnts = cv2.findContours(gray, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX
_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
for c in cnts:
    cv2.drawContours(gray,[c], 0, (255,255,255), -1)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (2,2))
regionFilled = cv2.morphologyEx(gray, cv2.MORPH_OPEN, kernel, ite
rations=2)
cv2_imshow(regionFilled)
```

Output: