**CS-515 Lab on Digital Image Processing**

**#Read an Image**

import cv2  
import matplotlib.pyplot as plt  
i=cv2.imread('pink.jpg')  
plt.subplot(122)  
plt.title('Flower Image')  
plt.imshow(i)  
plt.axis('off')  
plt.show()

**Assi.1 Write Python Script to perform basic operations on images.**

from PIL import Image  
import matplotlib. pyplot as plt  
img=Image.open('pink.jpg')  
#Image Resize  
resize\_img=img.resize((200,300))  
  
#Image Rotation  
rotated\_img=img.rotate(45)  
  
#Image Crop  
width, height=img.size  
area=(0,0,width/2, height/2)  
crop\_img=img.crop(area)  
  
#Display Images  
plt.subplot(221)  
plt.title('Original Image')  
plt.imshow(img)  
plt.axis('off')  
plt.subplot(222)  
plt.title('Resize Image')  
plt.imshow(resize\_img)  
plt.axis('off')  
plt.subplot(223)  
plt.title('Rotated Image')  
plt.imshow(rotated\_img)  
plt.axis('off')  
plt. subplot(224)  
plt.title('Crop Image')  
plt.imshow(crop\_img)  
plt.axis('off')  
plt.show()

**Assi.2 Write Python Script to perform conversion between color spaces.**

import cv2  
import matplotlib. pyplot as plt  
img= cv2.imread('pink.jpg')  
plt.subplot(231)  
plt.title('Input Image')  
plt.imshow( img)  
plt.axis('off')  
  
# Using cv2.COLOR\_BGR2GRAY color space  
  
gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY )  
  
plt.subplot(232)  
plt.title('Gray Image’')  
plt.imshow( gray\_img)  
plt.axis('off')  
  
# Using cv2.COLOR\_BGR2HSV color space  
  
hsv\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)  
plt.subplot(233)  
plt.title('HSV Image')  
plt.imshow( hsv\_img)  
plt.axis('off')  
  
# Using cv2.COLOR\_BGR2RGB color space  
bgr\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
plt.subplot(234)  
plt.title('RGB Image')  
plt.imshow( bgr\_img)  
plt.axis('off')  
  
  
# Using cv2.COLOR\_BGR2LAB color space  
  
lab\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2LAB)  
plt.subplot(235)  
plt.title('LAB Image')  
plt.imshow( lab\_img)  
plt.axis('off')  
plt.show()

**Assi.3 Write Python Script to perform Histogram Equalization.**

import cv2   
import matplotlib. pyplot as plt  
  
img=cv2.imread('img.png',cv2.IMREAD\_GRAYSCALE)  
plt.subplot(221)  
plt.title('Original image')  
plt.imshow(img)  
  
# Calculate the histogram  
histogram1 = cv2.calcHist([img], [0], None, [256], [0, 256])  
  
# Plot the original histogram  
plt.subplot(222)  
plt.title('Original Histogram')  
plt.plot(histogram1)  
plt.xlabel('Pixel Values')  
plt.ylabel('Frequency')  
  
  
#Histogram Equalization  
eq\_img=cv2.equalizeHist(img)  
plt.subplot(223)  
plt.title('Equalize image')  
plt.imshow(eq\_img)  
  
  
#plot the equalize histogram  
histogram2 = cv2.calcHist([eq\_img], [0], None, [256], [0, 256])  
plt.subplot(224)  
plt.title('Equalize Histogram')  
plt.plot(histogram2)  
plt.xlabel('Pixel Values')  
plt.ylabel('Frequency')  
plt.show()

**Assi.4 Write a python script to perform image filtering using spatial domain.**

**#Spatial domain filtering using smoothing**  
import cv2  
import matplotlib.pyplot as plt  
I=cv2.imread('rose.jpg',)  
cv2.imshow('Original Image',I)  
#Averaging Filter  
blur=cv2.blur(I,(5,5))  
cv2.imshow('Blurred Image',blur)  
#Gaussian Filter  
GB=cv2.GaussianBlur(I,(5,5),0)  
cv2.imshow('Gaussian Filter Image',GB)  
med=cv2.medianBlur(I,5)  
#Median Filter  
cv2.imshow('Median Filter Image',med)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

**#Spatial domain filtering using sharpning**

import cv2

import numpy as np

I=cv2.imread('img.png')

cv2.imshow('Original Image',I)

#generating the kernels

kernel\_sharpen1=np.array([[0,-1,0],[-1,5,-1],[0,-1,0]])

kernel\_sharpen2=np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])

#applying different kernels to the input image

output1=cv2.filter2D(I,-1,kernel\_sharpen1)

output2=cv2.filter2D(I,-1,kernel\_sharpen2)

cv2.imshow('Sharpening image 1',output1)

cv2.imshow('Sharpening image 2 ',output2)

cv2.waitKey(0)

**Assi.6 Write Python Script to perform image restoration.**

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
I=cv2.imread('rose.jpg',0)  
blurred=cv2.GaussianBlur(I,(9,9),10)  
noise=np.random.normal(0,10,I.shape)

noisy\_blurred=blurred + noise  
dft\_img=np.fft.fftshift(np.fft.fft2(noisy\_blurred))  
psf=np.fft.fftshift(np.fft.fft2(np.ones((9,9))/81,s=I.shape))  
wiener\_filter=np.conj(psf)/(np.abs(psf)\*\*2+0.01)  
restored=np.fft.ifft2(np.fft.ifftshift(dft\_img\*wiener\_filter))  
plt.subplot(131),plt.title('Blurred Image'),plt.imshow(blurred)  
plt.subplot(132),plt.title('Blurred+ Noisy Image'),plt.imshow(noisy\_blurred)  
plt.subplot(133),plt.title('Restored Image'),plt.imshow(np.abs(restored),cmap='gray')  
plt.show()

**Assi.7 Write python script for edge detection using varour**

import cv2  
import numpy as np  
  
#Read an Image  
image = cv2.imread('Lenna.png')  
gray\_image = cv2.cvtColor(image,cv2.COLOR\_BGR2GRAY)  
blurred\_image = cv2.GaussianBlur(gray\_image,(5,5),0)  
  
#Canny Edge Detection  
canny\_edges = cv2.Canny(blurred\_image,threshold1=100,threshold2=200)  
  
#Sobel Edge Detection  
sobel\_x = cv2.Sobel(gray\_image,cv2.CV\_64F,1,1,ksize=5)  
sobel\_y = cv2.Sobel(gray\_image,cv2.CV\_64F,1,1,ksize=5)  
sobel\_edges = cv2.magnitude(sobel\_x,sobel\_y)  
sobel\_edges = np.uint8(sobel\_edges)  
  
#Laplacian Operator  
laplacian = cv2.Laplacian(blurred\_image,cv2.CV\_64F)  
  
#Display Results  
cv2.imshow('Canny',canny\_edges)  
cv2.imshow('Sobel',sobel\_edges)  
cv2.imshow('Laplacian',laplacian)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

**Assi 8. Write Python Script to perform global, adapative and Otsu’s thresholding.**

import cv2

import matplotlib.pyplot as plt

# Load the image in grayscale

image = cv2.imread('img\_1.png', cv2.IMREAD\_GRAYSCALE)

# Global Thresholding

\_, global\_thresh = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

# Adaptive Thresholding

adaptive\_thresh = cv2.adaptiveThreshold(image, 255,

cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,

cv2.THRESH\_BINARY, 199, 5)

# Otsu's Thresholding

\_, otsu\_thresh = cv2.threshold(image, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

# Display the results

plt.figure(figsize=(10, 8))

plt.subplot(221)

plt.imshow(image, cmap='gray')

plt.title('Original Image')

plt.axis('off')

plt.subplot(222)

plt.imshow(global\_thresh, cmap='gray')

plt.title('Global Thresholding')

plt.axis('off')

plt.subplot(223)

plt.imshow(adaptive\_thresh, cmap='gray')

plt.title('Adaptive Thresholding')

plt.axis('off')

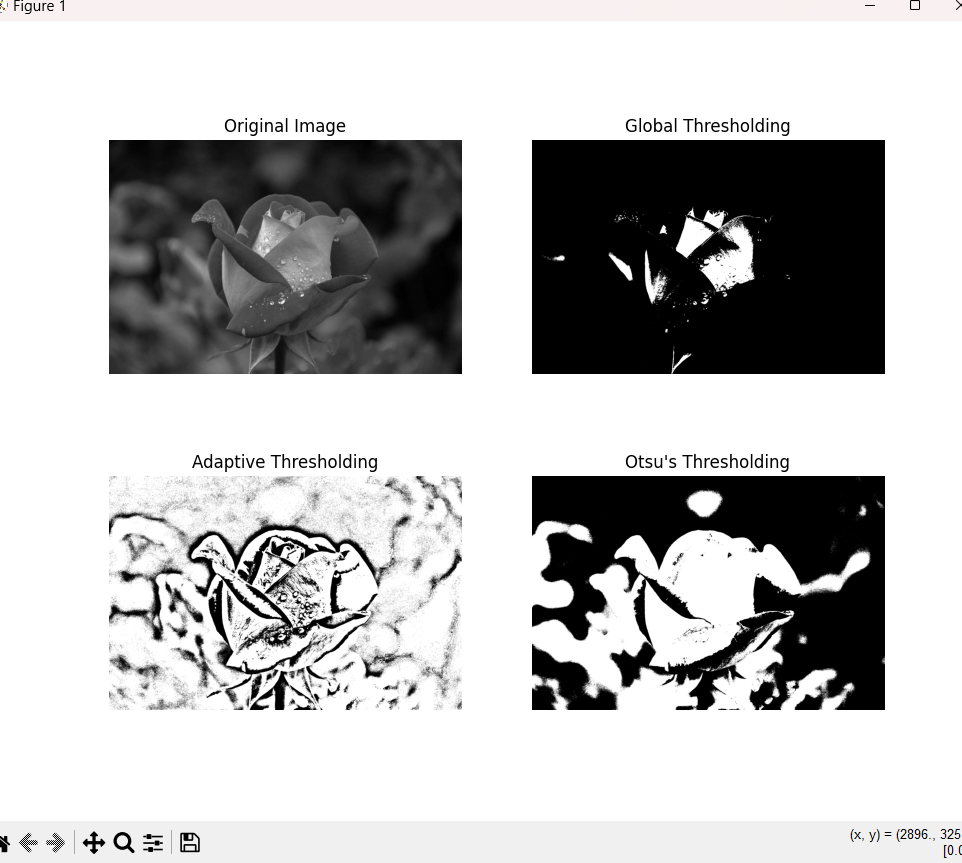
plt.subplot(224)

plt.imshow(otsu\_thresh, cmap='gray')

plt.title("Otsu's Thresholding")

plt.axis('off')

plt.show()



**10. Write python script to extract texture and color features of an image**

#Extract texture feature using LBP

from skimage import io, feature  
import matplotlib.pyplot as plt  
  
# Load the image  
image = io.imread('img\_1.png', as\_gray=True)  
  
# Apply Local Binary Pattern (LBP)  
lbp = feature.local\_binary\_pattern(image, P=8, R=1, method='uniform')  
  
# Display the original image and the LBP result  
plt.figure(figsize=(10, 5))  
plt.subplot(1, 2, 1)  
plt.title('Original Image')  
plt.imshow(image, cmap='gray')  
plt.subplot(1, 2, 2)  
plt.title('LBP')  
plt.imshow(lbp, cmap='gray')  
plt.show()

**11. Write a python code to perform character recognition.**

from PIL import Image  
import pytesseract  
pytesseract.pytesseract.tesseract\_cmd=r'C:\Program Files (x86)\Tesseract-OCR\tesseract.exe'  
image=Image.open("C:/Users/admin/Desktop/text.jpg")  
image.save('New\_Text.jpg')  
text = pytesseract.image\_to\_string(image)  
#print the text  
print('String:',text)

**Output:**

String: Noisy,image

to test

Tesseract OCR

import cv2  
import numpy as np  
from matplotlib import pyplot as plt  
image = cv2.imread('Lenna.png',0)  
  
kernel = np.ones((5,5),np.uint8)  
  
erosion = cv2.erode(image,kernel,iterations=1)  
  
dilation = cv2.morphologyEx(image,cv2.MORPH\_OPEN,kernel)  
  
opening = cv2.morphologyEx(image,cv2.MORPH\_OPEN,kernel)  
  
closing = cv2.morphologyEx(image,cv2.MORPH\_CLOSE,kernel)  
  
titles = ['Original Image','Erosion','Dilation','Opening','Closing']  
images = [image,erosion,dilation,opening,closing]  
  
for i in range(5):  
 plt.subplot(2,3,i+1)  
 plt.imshow(images[i],cmap='gray')  
 plt.title(titles[i])  
 plt.xticks([]),plt.yticks([])  
plt.show()