(A36607) COMPUTER VISION LAB B.Tech CSE(AI&ML) LAB EXPERIMENTS

1. Write programs for the following

a) Loading and displaying an image.

```
import cv2
img = cv2.imread("flower.jpeg")
cv2.imshow("Image", img)
print("Image Properties:")
print("Number of pixels:"+str(img.size))
print("Image dimensions:"+str(img.shape))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

b) Reading and writing video files.

```
import cv2
cap = cv2.VideoCapture("vtest.mp4")
if cap.isOpened() == False:
    print("Error opening file")
else:
    fps = cap.get(5)
    print("Frames per seconds:", fps,"FPS")
    frm_cnt = cap.get(7)
```

```
    print("Frame count:", frm_cnt)
c = 30
while(cap.isOpened()):
    ret,frame = cap.read()
    i = cv2.resize(frame,(150,120))
    cv2.imshow("Frame",i)
    k = cv2.waitKey(10)
    if k=='q':
        break
    if ret:
        cv2.imwrite('extra'+str(i)+'.png',frame)
cap.realse()
cv2.destroyAllWindows()
```

c) Image enhancement.

```
from PIL import Image
from PIL import ImageEnhance
import matplotlib.pyplot as plt
i = Image.open("flower.jpeg")
l = []
new_bri = 1.68
i_ori = ImageEnhance.Brightness(i)
```

```
i_bri = i_ori.enhance(new_bri)
new_col = 2.45
col_ori = ImageEnhance.Color(i)
i_col = col_ori.enhance(new_col)
new_con = 4.84
con_ori = ImageEnhance.Contrast(i)
i_con = con_ori.enhance(new_con)
new_sharp = 6.18
sharp_ori = ImageEnhance.Sharpness(i)
i_sharp = sharp_ori.enhance(new_sharp)
l = [i, i_bri, i_col, i_con, i_sharp]
t = ['original', 'Brightness', 'color', 'contrast', 'sharpness']
for x in range(5):
    plt.subplot(1, 5, x+1)
    plt.imshow(l[x], 'gray')
    plt.title(t[x])
    plt.xticks([]), plt.yticks([])
plt.show()
```

2. Write a code for basic Statistical Analysis of Images(To find sum , average , standard deviation , min and max)

3. Write a program study contrast adjustment of a given image

```
from PIL import Image,ImageEnhance
img = Image.open("flower.jpeg")
img.show()
enhancer = ImageEnhance.Contrast(img)

f = 0.4
img_output = enhancer.enhance(f)
img_output.show()
f = 4
img_output = enhancer.enhance(f)
img_output.show()
```

4. Write a code to apply Different Filtering Operations on Images.

```
import cv2
import numpy
from matplotlib import pyplot as plt
img = cv2.imread('white.jpg')

#gaussian filtering
gblur = cv2.GaussianBlur(img,(5,5),0)
cv2.imshow("Original",img)
cv2.imshow("Gaussian blurred",gblur)
```

```
#median filtering
mblur = cv2.medianBlur(img,7)
cv2.imshow('Median image', mblur)

#bilateral filtering
blur = cv2.bilateralFilter(img,7,75,75)
cv2.imshow("bilateral-blur image",blur)

cv2.waitKey(0)
cv2.destoryAllWindows()
```

5. Write a code to apply morphological operations like dilation, erosion, opening and closing on the given image.

```
#emorphing
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('pick.jpg', cv2.IMREAD_GRAYSCALE)

_,mask = cv2.threshold(img, 210,  255, cv2.THRESH_BINARY_INV)
```

```
kernal = np.ones((2,2), np.uint8)

dilation = cv2.dilate(mask, kernal, iterations = 4)

erode = cv2.erode(mask, kernal, iterations = 3)

opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernal)

closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernal)

titles = ['original image', 'masked image', 'dilation image', 'erosion
image','opening image', 'closing image','gradient', 'tophat', 'blackhat']

images = [img, mask, dilation, erode,opening,closing]


for i in range(6):
    plt.subplot(3,3,i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])


plt.show()

cv2.waitKey(0)

cv2.destroyAllWindows()
```

6. Write a code for detection of an edge / curvature in a given image and curve fitting

detection of an edge:

```
import cv2
```

```
img =  cv2.imread("flower.jpeg")

#canny operator
canny = cv2.canny(img, 150, 250)
cv2.imshow('Original',img)
cv2.imshow('canny', canny)

#laplacian
img1 = cv2.GaussianBlur(img,(3,3),0)
laplacian = cv2.Laplacian(img, cv2.cv_64F)
cv2.imshow('laplacian', laplacian)

#sobel
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
img2 = cv2.GaussianBlur(gray,(3,3),0)
sobelx = cv2.Sobel(img, cv2.CV_64F, 1,0, ksize = 5)
sobely = cv2.Sobel(img, cv2.CV_64F, 0,1, ksize = 5)
cv2.imshow('SobelX', Sobelx)
cv2.imshow('SobelY', Sobely)

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

Curvature:

```
import cv2
import numpy as np

image = cv2.imread('shape.jpeg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

image_flt = np.float32(gray)
dst = cv2.cornerHarris(image_flt, 2, 3, 0.04)
dst = cv2.dilate(dst, None)

image[dst>0.01 * dst.max()] = [0, 0, 255]

cv2.imshow('Detected corners', image)

cv2.waitKey(0)

cv2.destroyAllWindows
```

7. Write a code to implement SURF / SIFT / HOG detector

```
import numpy as np
```

```
import cv2 as cv2
i = cv2.imread("flower.jpeg")
i = cv2.resize(i,(350,300))
g = cv2.cvtColor(i,cv2.COLOR_BGR2GRAY)

sift_ob = cv2.SIFT_create()
kp = sift_ob.detect(g,None)
img = cv2.drawKeypoints(g, kp, i)
img = cv2.resize(img, (350,300))
cv2.imshow('Output', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8. Implement histogram calculation and equalization for the given image.

```
import cv2
from matplotlib import pyplot as plt
import numpy as np
img = cv2.imread('flower.jpeg',0)

#creating a histogram equalization
equ = cv2.equalizeHist(img)
```

```
#stacking img side by side
res = np.hstack((img,equ))

#show img input vs output
cv2.imshow('histogram img',res)
cv2.waitKey(0)
cv2.destroyAllWindows()

#calculate freq of pixels in range 0-255
histg = cv2.calcHist([img],[0],None,[250],[0,256])
plt.plot (histg)
plt.show()
```

9. Write a code to perform 2-D spatial transformation to image

```
#2-D spatial transformation

import cv2
image = cv2.imread("lilly.jpeg")
height,width = image.shape[:2]

center = (width/2, height/2)
```

```
newsize = (120,200)
out = cv2.resize(image,newsize)
cv2.imshow("resized image",out)

rotate_matrix = cv2.getRotationMatrix2D(center=center, angle=65, scale=1)
rotated_image = cv2.warpAffine(src = image, M=rotate_matrix,
                               dsize=(width,height))
cv2.imshow('Original image', image)
cv2.imshow('Rotate image', rotated_image)

cv2.waitKey(0)
cv2.imwrite('rotated_image.jpg',rotated_img)
```

10. Convert the input image from RGB color space to CMY and HSV color space.

```
import cv2
import numpy as np
image = cv2.imread("flower.jpeg")
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
img = image.astype(np.float64)/255
c = (1-img[...,2])
m = (1-img[...,1])
```

```
y = (1-img[...,0])
cmy_image = (np.dstack((c, m, y))*255).astype(np.int8)
cv2.imshow("Original image", image)
cv2.imshow("HSV image", hsv_image)

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

11. Write a code for feature Extraction from Images.

```
import cv2

imageread = cv2.imread('balls.jpeg')
imagegray = cv2.cvtColor(imageread,cv2.COLOR_BGR2GRAY)
imageedges = cv2.Canny(imagegray, 10, 100)

cv2.imshow('original image',imageread)
#finding the contours

contours, hierarchy = cv2.findContours(imageedges,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

#drawing the contours
```

```
cv2.drawContours(imageread, contours,-1, (0,0,255),6)

cv2.imshow('image_with_contours',imageread)

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

12. Write a code for basic Shape Analysis of an image.

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

img = cv2.imread('shape.jpeg')
img = cv2.resize(img,(550,550))

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#setting threshold of gray image
_,threshold = cv2.threshold(gray, 127, 240, cv2.THRESH_BINARY)

#using a findContours() function
contours,_ = cv2.findContours(threshold, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

```
i = 0

# list for storing names of shapes
for contour in contours:

    if i == 0:
        i = 1
        continue

    # cv2.approxPloyDP() function to approximate the shape
    approx = cv2.approxPolyDP(contour, 0.01 * cv2.arcLength(contour, True), True)

    # using drawContours() function
    cv2.drawContours(img, [contour], 0, (210, 100, 100), 5)

    # finding center point of shape
    M = cv2.moments(contour)

    if M['m00'] != 0.0:
        x = int(M['m10']/M['m00'])
```

```
            y = int(M['m01']/M['m00'])-10


        # putting shape name at center of each shape
        if len(approx) == 3:
            cv2.putText(img, 'Triangle', (x, y),
        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (210, 0, 100), 2)


        elif len(approx) == 4:
            cv2.putText(img, 'Quadilateral', (x,
y),cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0,100), 2)


        elif len(approx) == 5:
            cv2.putText(img, 'Pentagon', (x,
y),cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 0, 200), 2)


        elif len(approx) == 6:
            cv2.putText(img, 'Hexagon', (x,
y),cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 200, 0), 2)
        elif len(approx) == 8:
            cv2.putText(img, 'Octagon', (x,
y),cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 200, 0), 2)


        else:
```

```
                cv2.putText(img, 'circle', (x,
y),cv2.FONT_HERSHEY_SIMPLEX, 0.6, (25, 205, 25), 2)


# displaying the image after drawing contours

cv2.imshow('shapes', img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

13. Write a program to reduce dimensionality using PCA for the given images.

14. Write a code for object detection using Hough transform / Template matching.

15. Object classification using SVM / Adaboost classifier.

16. Object tracking using Kalman filter approach