**Date:29.06.25**

**Assignment : 2 Lab problems(1-20)**

1**. Perform basic Image Handling and processing operations on the image.**

**• Read an image in python and Convert an Image to Grayscale**



import cv2

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

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow('Gray', gray); cv2.waitKey(0)

**2. Perform basic Image Handling and processing operations on the image**

**• Read an image in python and Convert an Image to Blur using GaussianBlur.**



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

blur = cv2.GaussianBlur(img, (5, 5), 0)

cv2.imshow('Blur', blur); cv2.waitKey(0)

**3. Perform basic Image Handling and processing operations on the image**

**• Read an image in python and Convert an Image to show outline using Canny function.**



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

edges = cv2.Canny(img, 100, 200)

cv2.imshow('Canny', edges); cv2.waitKey(0)

**4. Perform basic Image Handling and processing operations on the image**

**• Read an image in python and Dilate an Image using Dilate function.**



img = cv2.imread('image.jpg', 0)

dilated = cv2.dilate(img, None)

cv2.imshow('Dilated', dilated); cv2.waitKey(0)

**5. Perform basic Image Handling and processing operations on the image**

**• Read an image in python and Erode an Image using erode function.**



img = cv2.imread('image.jpg', 0)

eroded = cv2.erode(img, None)

cv2.imshow('Eroded', eroded); cv2.waitKey(0)

**6. Perform basic video processing operations on the captured video**

**• Read captured video in python and display the video, in slow motion and in fast motion.**

cap = cv2.VideoCapture('video.mp4')

while cap.isOpened():

ret, frame = cap.read()

if not ret: break

cv2.imshow('Video', frame)

if cv2.waitKey(100) & 0xFF == ord('q'): break # Slow (100 ms)

cap.release(); cv2.destroyAllWindows()

**7. Capture video from web Camera and Display the video, in slow motion and in fast motion.**

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

cv2.imshow('Webcam', frame)

if cv2.waitKey(100) & 0xFF == ord('q'): break

cap.release(); cv2.destroyAllWindows()

**8. Scaling an image to its Bigger and Smaller sizes.**

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

small = cv2.resize(img, None, fx=0.5, fy=0.5)

big = cv2.resize(img, None, fx=2, fy=2)

cv2.imshow('Small', small); cv2.imshow('Big', big); cv2.waitKey(0)

**9. Perform Rotation of an image to clockwise and counter clockwise direction.**

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

(h, w) = img.shape[:2]

M = cv2.getRotationMatrix2D((w//2, h//2), 90, 1) # Clockwise

rotated = cv2.warpAffine(img, M, (w, h))

cv2.imshow('Rotated', rotated); cv2.waitKey(0)

**10. Perform moving of an image from one place to another.**

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

M = np.float32([[1, 0, 100], [0, 1, 50]])

moved = cv2.warpAffine(img, M, (img.shape[1], img.shape[0]))

cv2.imshow('Moved', moved); cv2.waitKey(0)

**11. Perform Affine Transformation on the image.**

pts1 = np.float32([[50,50],[200,50],[50,200]])

pts2 = np.float32([[10,100],[200,50],[100,250]])

M = cv2.getAffineTransform(pts1, pts2)

dst = cv2.warpAffine(cv2.imread('image.jpg'), M, (300,300))

cv2.imshow('Affine', dst); cv2.waitKey(0)

**12. Perform Perspective Transformation on the image.**

pts1 = np.float32([[50,50],[200,50],[50,200],[200,200]])

pts2 = np.float32([[10,100],[200,50],[100,250],[200,250]])

M = cv2.getPerspectiveTransform(pts1, pts2)

dst = cv2.warpPerspective(cv2.imread('image.jpg'), M, (300,300))

cv2.imshow('Perspective', dst); cv2.waitKey(0)

**13. Perform Perspective Transformation on the Video.**

cap = cv2.VideoCapture('video.mp4')

M = cv2.getPerspectiveTransform(pts1, pts2)

while True:

ret, frame = cap.read()

if not ret: break

warp = cv2.warpPerspective(frame, M, (300,300))

cv2.imshow('Video Warp', warp)

if cv2.waitKey(30) & 0xFF == ord('q'): break

cap.release(); cv2.destroyAllWindows()

**14. Perform transformation using Homography matrix.**

H, \_ = cv2.findHomography(pts1, pts2)

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

warped = cv2.warpPerspective(img, H, (300,300))

cv2.imshow('Homography', warped); cv2.waitKey(0)

**15. Perform transformation using Direct Linear Transformation.**

# Simplified using cv2.getPerspectiveTransform as DLT under the hood

M = cv2.getPerspectiveTransform(pts1, pts2)

dst = cv2.warpPerspective(cv2.imread('image.jpg'), M, (300,300))

cv2.imshow('DLT', dst); cv2.waitKey(0)

**16. Perform Edge detection using canny method**

img = cv2.imread('image.jpg', 0)

edges = cv2.Canny(img, 50, 150)

cv2.imshow('Canny', edges); cv2.waitKey(0)

**17. Perform Edge detection using Sobel Matrix along X axis**

img = cv2.imread('image.jpg', 0)

sobelx = cv2.Sobel(img, cv2.CV\_64F, 1, 0)

cv2.imshow('Sobel X', sobelx); cv2.waitKey(0)

**18. Perform Edge detection using Sobel Matrix along Y axis**

sobely = cv2.Sobel(img, cv2.CV\_64F, 0, 1)

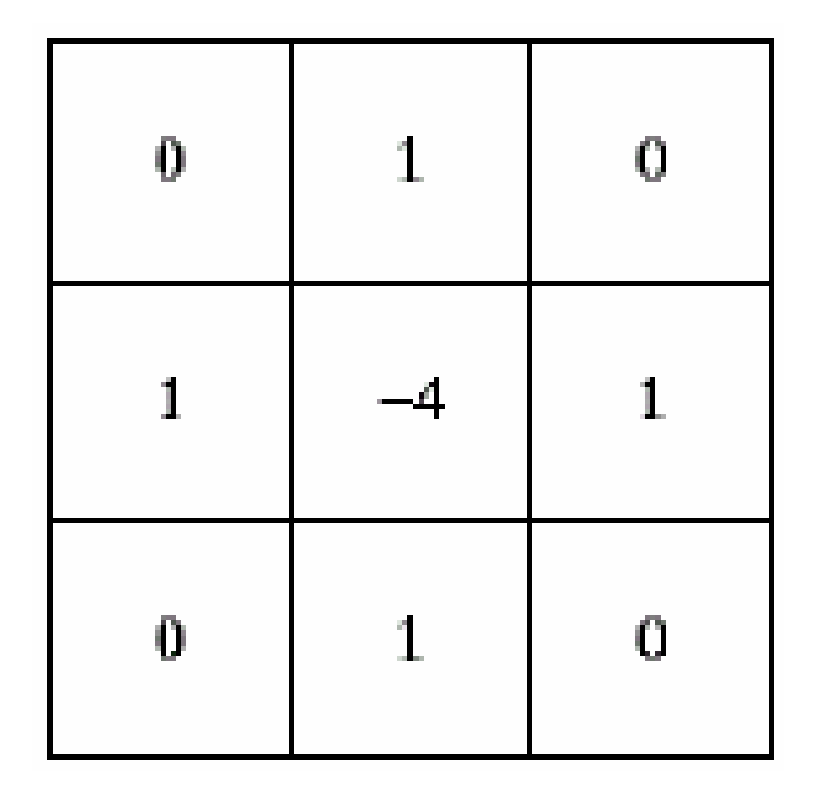
cv2.imshow('Sobel Y', sobely); cv2.waitKey(0)

**19. Perform Edge detection using Sobel Matrix along XY axis**

sobelxy = cv2.Sobel(img, cv2.CV\_64F, 1, 1)

cv2.imshow('Sobel XY', sobelxy); cv2.waitKey(0)

**20. Perform Sharpening of Image using Laplacian mask with negative center coefficient.**

****

img = cv2.imread('image.jpg', 0)

lap = cv2.Laplacian(img, cv2.CV\_64F)

sharp = cv2.convertScaleAbs(img - lap)

cv2.imshow('Sharpened', sharp); cv2.waitKey(0)