**Name :** Anto Chriswin G

**Reg No :** 192324093

**Course Code:** DSA0208 **Course Name:** computer vision with opencv for medical applications

**EXP - 1**

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

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

**CODE:**

import cv2

image = cv2.imread(r"C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img1.png")  # Replace with your image file name

if image is None:

    print("Error: Image not found or unable to read.")

else:

    cv2.imshow('Original Image', image)

    gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

    cv2.imshow('Grayscale Image', gray\_image)

    cv2.imwrite('grayscale\_output.jpg', gray\_image)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

**OUTPUT:**

****

****

**EXP - 2**

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

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

**CODE :**

import cv2

image = cv2.imread(r"C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img1.png")  # Replace with your image filename

if image is None:

    print("Error: Image not found or unable to read.")

else:

    cv2.imshow('Original Image', image)

    blurred\_image = cv2.GaussianBlur(image, (15, 15), 0)  # (15,15) is kernel size

    cv2.imshow('Blurred Image', blurred\_image)

    cv2.imwrite('blurred\_output.jpg', blurred\_image)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

**OUTPUT :**

****

****

**EXP - 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.**

**CODE:**

import cv2

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img1.png')  # Replace with your actual image filename

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

blurred\_image = cv2.GaussianBlur(gray\_image, (5, 5), 0)

edges = cv2.Canny(blurred\_image, threshold1=50, threshold2=150)

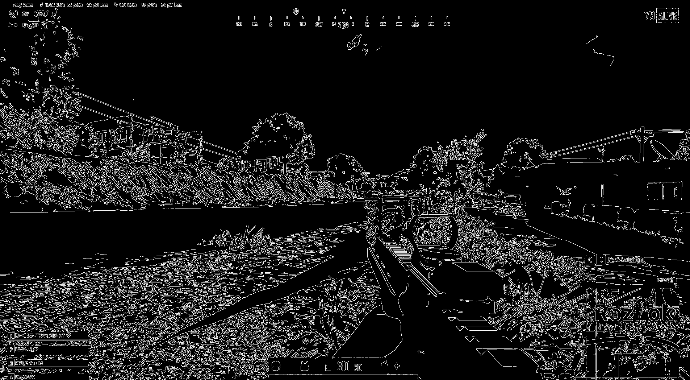
cv2.imshow('Original Image', image)

cv2.imshow('Canny Edge Detection', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

**EXP - 4**

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

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

**CODE:**

import cv2

import numpy as np

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img1.png')  # Replace with your actual image filename

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

kernel = np.ones((5, 5), np.uint8)  # You can adjust the size for stronger/weaker dilation

dilated\_image = cv2.dilate(gray\_image, kernel, iterations=1)

cv2.imshow('Original Image', image)

cv2.imshow('Dilated Image', dilated\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

****

**EXP - 5**

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

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

**CODE:**

import cv2

import numpy as np

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img1.png')  # Replace with your image file

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

kernel = np.ones((5, 5), np.uint8)  # You can adjust the size for different effects

eroded\_image = cv2.erode(gray\_image, kernel, iterations=1)

cv2.imshow('Original Image', image)

cv2.imshow('Eroded Image', eroded\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

****

**EXP - 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.**

**CODE:**

import cv2

video\_path = r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\shotray.

cap = cv2.VideoCapture(video\_path)

if not cap.isOpened():

    print("Error: Could not open video.") exit()

print("Select Playback Mode:")

print("1. Normal Speed")

print("2. Slow Motion")

print("3. Fast Motion")

mode = input("Enter option (1/2/3): ")

if mode == '1':

    delay = int(1000 / 30)  # Normal: 30 FPS

elif mode == '2':

    delay = int(1000 / 10)  # Slow Motion: ~10 FPS

elif mode == '3':

    delay = int(1000 / 60)  # Fast Motion: ~60 FPS

else:

    print("Invalid option. Playing at normal speed.")

    delay = int(1000 / 30)

while True:

    ret, frame = cap.read()

    if not ret:

        break

    cv2.imshow('Video Playback', frame)

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

        break

cap.release()

cv2.destroyAllWindows()

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

**CODE:**

import cv2

cap = cv2.VideoCapture(0)

if not cap.isOpened():

    print("Error: Could not open webcam.")

    exit()

print("Select Webcam Playback Mode:")

print("1. Normal Speed")

print("2. Slow Motion")

print("3. Fast Motion")

mode = input("Enter option (1/2/3): ")

if mode == '1':

    delay = int(1000 / 30)  # Normal: ~30 FPS

elif mode == '2':

    delay = int(1000 / 10)  # Slow Motion: ~10 FPS

elif mode == '3':

    delay = int(1000 / 60)  # Fast Motion: ~60 FPS

else:

    print("Invalid option. Defaulting to Normal Speed.")

    delay = int(1000 / 30)

print("Press 'q' to quit.")

while True:

    ret, frame = cap.read()

    if not ret:

        print("Error: Failed to capture frame.")

        break

    cv2.imshow('Webcam Video', frame)

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

        break

cap.release()

cv2.destroyAllWindows()

**EXP – 8**

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

**CODE :**

import cv2

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img1.png')  # Replace with your image path

if image is None:

    print("Error: Image not found or unable to read.")

    exit()

smaller\_image = cv2.resize(image, None, fx=0.5, fy=0.5, interpolation=cv2.INTER\_AREA)

bigger\_image = cv2.resize(image, None, fx=1.5, fy=1.5, interpolation=cv2.INTER\_LINEAR)

cv2.imshow('Original Image', image)

cv2.imshow('Smaller Image (50%)', smaller\_image)

cv2.imshow('Bigger Image (150%)', bigger\_image)

cv2.imwrite('scaled\_smaller.jpg', smaller\_image)

cv2.imwrite('scaled\_bigger.jpg', bigger\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT :**

**Bigger Scale:**

**Smaller Scale :**

****

**EXP – 9**

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

**CODE:**

import cv2

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img2.jpg')  # Replace with your image file

if image is None:

    print("Error: Image not found or unable to read.")

    exit()

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

center = (w // 2, h // 2)

clockwise\_matrix = cv2.getRotationMatrix2D(center, -90, 1.0)

clockwise\_rotated = cv2.warpAffine(image, clockwise\_matrix, (w, h))

counter\_matrix = cv2.getRotationMatrix2D(center, 90, 1.0)

counter\_rotated = cv2.warpAffine(image, counter\_matrix, (w, h))

cv2.imshow('Original Image', image)

cv2.imshow('Clockwise Rotation (90°)', clockwise\_rotated)

cv2.imshow('Counter-Clockwise Rotation (90°)', counter\_rotated)

cv2.imwrite('rotated\_clockwise.jpg', clockwise\_rotated)

cv2.imwrite('rotated\_counter.jpg', counter\_rotated)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT**

Clockwise:

Counter Clockwise:

**EXP -10**

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

**CODE:**

import cv2

import numpy as np

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img3.jpg')

if image is None:

    print("Error: Image not found or unable to read.")

    exit()

# Move image 100 pixels right and 50 pixels down

tx = 100  # shift along X-axis

ty = 50   # shift along Y-axis

translation\_matrix = np.float32([[1, 0, tx], [0, 1, ty]])

height, width = image.shape[:2]

translated\_image = cv2.warpAffine(image, translation\_matrix, (width + tx, height + ty))

cv2.imshow('Original Image', image)

cv2.imshow('Moved Image', translated\_image)

cv2.imwrite('translated\_image.jpg', translated\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**INPUT:**

**OUTPUT:**

****

**EXP – 11**

**Perform Affine Transformation on the image.**

**CODE:**import cv2

import numpy as np

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img3.jpg')

if image is None:

    print("Error: Image not found or unable to read.")

    exit()

rows, cols = image.shape[:2]

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

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

M = cv2.getAffineTransform(pts1, pts2)

affine\_transformed = cv2.warpAffine(image, M, (cols, rows))

cv2.imshow('Original Image', image)

cv2.imshow('Affine Transformed Image', affine\_transformed)

cv2.imwrite('affine\_transformed.jpg', affine\_transformed)

cv2.waitKey(0)

cv2.destroyAllWindows()

**INPUT:**

**OUTPUT:**

****

**EXP – 12**

**Perform Perspective Transformation on the image.**

**CODE:**

import cv2

import numpy as np

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img3.jpg')

if image is None:

    print("Error: Image not found or unable to read.")

    exit()

rows, cols = image.shape[:2]

pts1 = np.float32([[50, 50], [cols - 50, 50], [50, rows - 50], [cols - 50, rows - 50]])

pts2 = np.float32([[0, 0], [cols, 0], [0, rows], [cols, rows]])

matrix = cv2.getPerspectiveTransform(pts1, pts2)

perspective\_transformed = cv2.warpPerspective(image, matrix, (cols, rows))

cv2.imshow('Original Image', image)

cv2.imshow('Perspective Transformed Image', perspective\_transformed)

cv2.imwrite('perspective\_transformed.jpg', perspective\_transformed)

cv2.waitKey(0)

cv2.destroyAllWindows()

**INPUT:**



**OUPUT:**

**EXP – 13**

**Perform Perspective Transformation on the Video.**

**CODE:**

import cv2

import numpy as np

cap = cv2.VideoCapture(0)  # Replace with 'your\_video.mp4' if using a video file

if not cap.isOpened():

    print("Error: Could not open video.")

    exit()

ret, frame = cap.read()

if not ret:

    print("Error: Couldn't read frame.")

    exit()

height, width = frame.shape[:2]

pts1 = np.float32([[50, 50], [width - 50, 50], [50, height - 50], [width - 50, height - 50]])

pts2 = np.float32([[0, 0], [width, 0], [0, height], [width, height]])

matrix = cv2.getPerspectiveTransform(pts1, pts2)

print("Press 'q' to quit...")

while True:

    ret, frame = cap.read()

    if not ret:

        break

    transformed\_frame = cv2.warpPerspective(frame, matrix, (width, height))

    cv2.imshow('Original Video', frame)

    cv2.imshow('Perspective Transformed Video', transformed\_frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

cap.release()

cv2.destroyAllWindows()

**EXP – 14**

**Perform transformation using Homography matrix.**

**CODE:**

import cv2

import numpy as np

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img3.jpg')

if image is None:

    print("Error: Image not found.")

    exit()

h, w = image.shape[:2]

src\_points = np.float32([[100, 100], [w - 100, 100], [100, h - 100], [w - 150, h - 150]])

dst\_points = np.float32([[0, 0], [w, 0], [0, h], [w, h]])

H, status = cv2.findHomography(src\_points, dst\_points)

homography\_result = cv2.warpPerspective(image, H, (w, h))

cv2.imshow('Original Image', image)

cv2.imshow('Homography Transformed Image', homography\_result)

cv2.imwrite('homography\_result.jpg', homography\_result)

cv2.waitKey(0)

cv2.destroyAllWindows()

**INPUT:**

**OUTPUT:**

****

**EXP – 15**

**Perform transformation using Direct Linear Transformation.**

**CODE:**

import cv2

import numpy as np

def compute\_homography\_dlt(src\_pts, dst\_pts):

    A = []

    for (x, y), (x\_prime, y\_prime) in zip(src\_pts, dst\_pts):

        A.append([-x, -y, -1, 0, 0, 0, x \* x\_prime, y \* x\_prime, x\_prime])

        A.append([0, 0, 0, -x, -y, -1, x \* y\_prime, y \* y\_prime, y\_prime])

    A = np.asarray(A)

    U, S, Vt = np.linalg.svd(A)

    H = Vt[-1].reshape(3, 3)

    return H / H[2, 2]

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img3.jpg')

if image is None:

    print("Error: Could not load image.")

    exit()

h, w = image.shape[:2]

src\_pts = np.float32([[100, 100], [w - 100, 100], [100, h - 100], [w - 150, h - 150]])

dst\_pts = np.float32([[0, 0], [w, 0], [0, h], [w, h]])

H\_dlt = compute\_homography\_dlt(src\_pts, dst\_pts)

transformed\_img = cv2.warpPerspective(image, H\_dlt, (w, h))

cv2.imshow("Original Image", image)

cv2.imshow("DLT Homography Transformed", transformed\_img)

cv2.imwrite("dlt\_transformed.jpg", transformed\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**INPUT:**

**EXP – 16**

**Perform Edge detection using canny method**

**CODE:**

import cv2

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img3.jpg')

if image is None:

    print("Error: Image not found.")

    exit()

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

blurred = cv2.GaussianBlur(gray, (5, 5), 1.4)

edges = cv2.Canny(blurred, threshold1=50, threshold2=150)

cv2.imshow('Original Image', image)

cv2.imshow('Canny Edge Detection', edges)

cv2.imwrite('canny\_edges.jpg', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

**INPUT:**

**OUTPUT:**

****

**EXP – 17**

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

**CODE:**

import cv2

import numpy as np

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img3.jpg'

if image is None:

    print("Error: Image not found.")

    exit()

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

sobel\_x = cv2.Sobel(gray, cv2.CV\_64F, dx=1, dy=0, ksize=3)

abs\_sobel\_x = cv2.convertScaleAbs(sobel\_x)

cv2.imshow('Original Image', image)

cv2.imshow('Sobel X Edge Detection', abs\_sobel\_x)

cv2.imwrite('sobel\_x\_edges.jpg', abs\_sobel\_x)

cv2.waitKey(0)

cv2.destroyAllWindows()

**INPUT:**



**OUTPUT:**

****

**EXP – 18**

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

**CODE:**

import cv2

import numpy as np

image = cv2.imread('your\_image.jpg')

if image is None:

print("Error: Image not found.")

exit()

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

sobel\_y = cv2.Sobel(gray, cv2.CV\_64F, dx=0, dy=1, ksize=3)

abs\_sobel\_y = cv2.convertScaleAbs(sobel\_y)

cv2.imshow('Original Image', image)

cv2.imshow('Sobel Y Edge Detection', abs\_sobel\_y)

cv2.imwrite('sobel\_y\_edges.jpg', abs\_sobel\_y)

cv2.waitKey(0)

cv2.destroyAllWindows()

**INPUT:**

******OUTPUT:**

**EXP – 19**

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

**CODE:**

import cv2

import numpy as np

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img2.jpg')

if image is None:

    print("Error: Image not found.")

    exit()

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

sobel\_x = cv2.Sobel(gray, cv2.CV\_64F, dx=1, dy=0, ksize=3)

sobel\_y = cv2.Sobel(gray, cv2.CV\_64F, dx=0, dy=1, ksize=3)

sobel\_xy = cv2.magnitude(sobel\_x, sobel\_y)

sobel\_xy = cv2.convertScaleAbs(sobel\_xy)

cv2.imshow('Original Image', image)

cv2.imshow('Sobel XY Edge Detection', sobel\_xy)

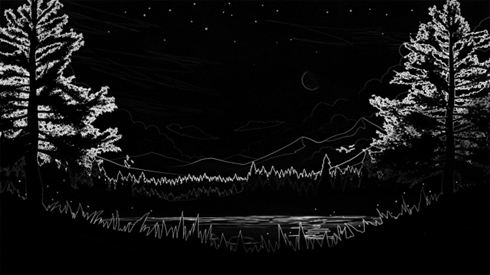
cv2.imwrite('sobel\_xy\_edges.jpg', sobel\_xy)

cv2.waitKey(0)

cv2.destroyAllWindows()

**INPUT:**

**OUTPUT:**

****

**EXP – 20**

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

**CODE:**

import cv2

import numpy as np

image = cv2.imread(r'C:\Users\antoc\OneDrive\Desktop\Course CV\LAB\src\img3.jpg')

if image is None:

    print("Error: Image not found.")

    exit()

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

laplacian\_kernel = np.array([[0,  -1,  0],

                             [-1,  5, -1],

                             [0,  -1,  0]])

sharpened = cv2.filter2D(gray, -1, laplacian\_kernel)

cv2.imshow('Original Grayscale Image', gray)

cv2.imshow('Sharpened Image (Laplacian Mask)', sharpened)

cv2.imwrite('sharpened\_laplacian.jpg', sharpened)

cv2.waitKey(0)

cv2.destroyAllWindows()

**INPUT:**

**OUTPUT:**

****