DSA0202-COMPUTER VISION WITH OPENCV

NAME : K M SURYA
REG.NO : 192124105

1.Write a python program to eroded an image using opencv.

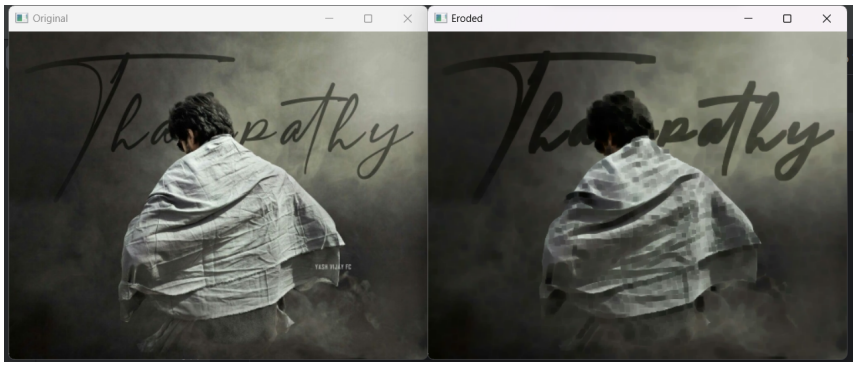
AIM :

To write a python program to eroded a image using opencv

PROGRAM :

```
import cv2
import numpy as np
img =
cv2.imread("C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg")
kernel = np.ones((5, 5), np.uint8)
eroded = cv2.erode(img, kernel, iterations=1)
cv2.imshow("Original", img)
cv2.imshow("Eroded", eroded)
cv2.waitKey(0)
```

OUTPUT :



2.Write a python program to grayscale a image using opencv

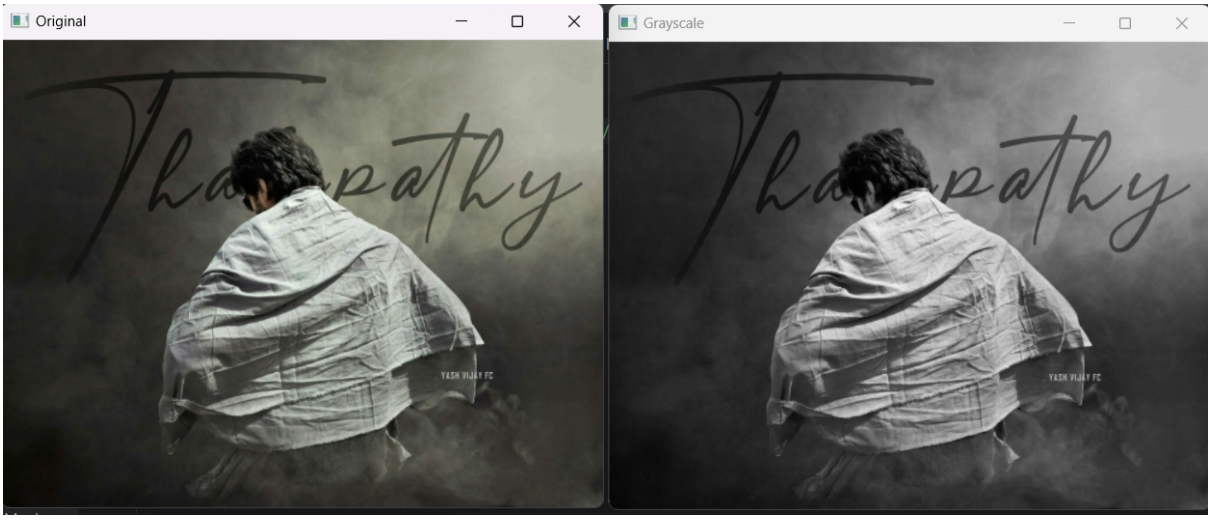
AIM :

To write a python program to grayscale a image using opencv

PROGRAM :

```
import cv2
import numpy as np
img=cv2.imread("C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow("Original", img)
cv2.imshow("Grayscale", gray)
cv2.waitKey(0)
```

OUTPUT :



3.Write a python program to Homography matrix a image using opencv

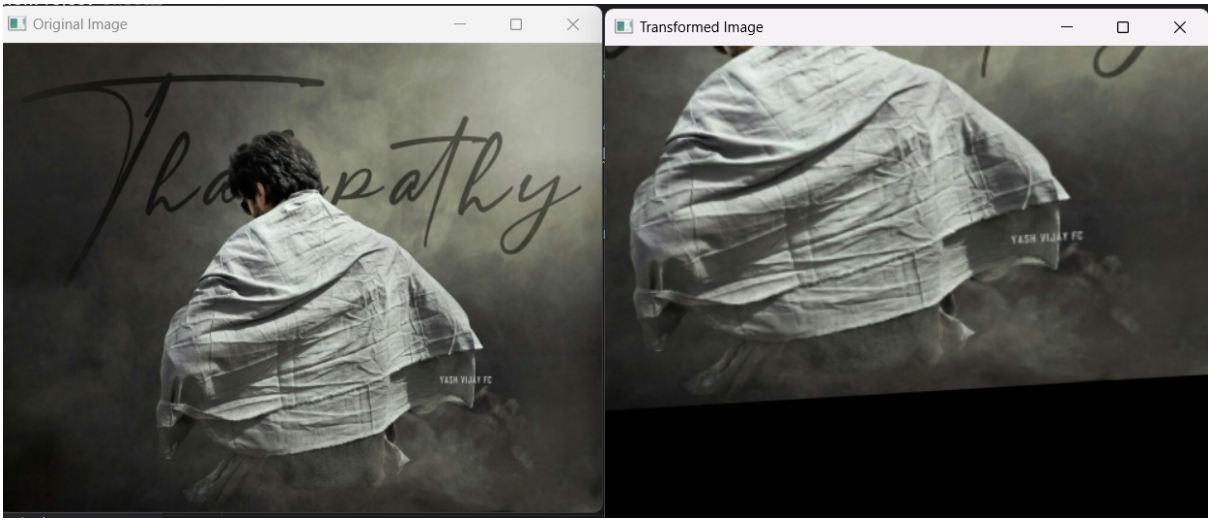
AIM :

To write a python program to Homography matrix a image using opencv

PROGRAM :

```
import cv2
import numpy as np
# Load the image and the target points
image=cv2.imread('C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg')
target_points = np.array([[0, 0], [500, 0], [500, 500], [0, 500]], dtype=np.float32)
# Set the source points to transform
source_points = np.array([[141, 131], [480, 159], [493, 630], [64, 601]], dtype=np.float32)
# Calculate the homography matrix
homography_matrix, _ = cv2.findHomography(source_points, target_points)
# Perform the transformation
transformed_image = cv2.warpPerspective(image, homography_matrix, (500, 500)
# Display the original and transformed images
cv2.imshow('Original Image', image)
cv2.imshow('Transformed Image', transformed_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



4.Write a python program to move a image using opencv

AIM :

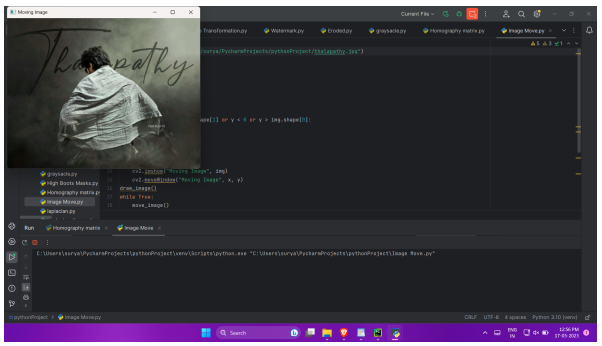
To write a python program to move a image using opencv

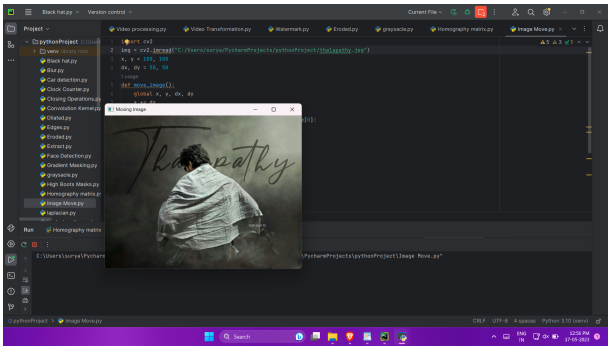
PROGRAM :

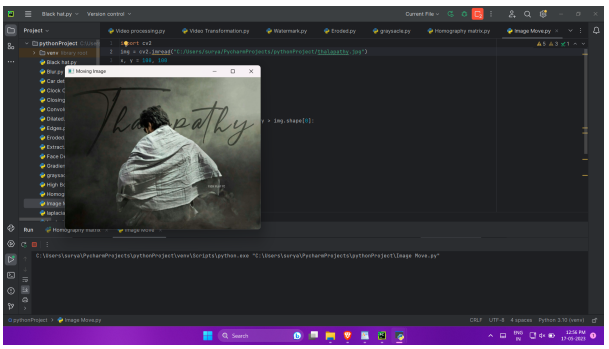
```
import cv2
img =
cv2.imread("C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg")
x, y = 100, 100
dx, dy = 50, 50
def move_image():
    global x, y, dx, dy
    x += dx
    y += dy
```

```
    if x < 0 or x > img.shape[1] or y < 0 or y > img.shape[0]:
        dx = -dx
        dy = -dy
def draw_image():
    global x, y
    cv2.imshow("Moving Image", img)
    cv2.moveWindow("Moving Image", x, y)
draw_image()
move_image()
    draw_image()
    key = cv2.waitKey(50)
    if key != -1:
        break
cv2.destroyAllWindows()
```

OUTPUT :







5.Write a python program to small and big a image using opencv

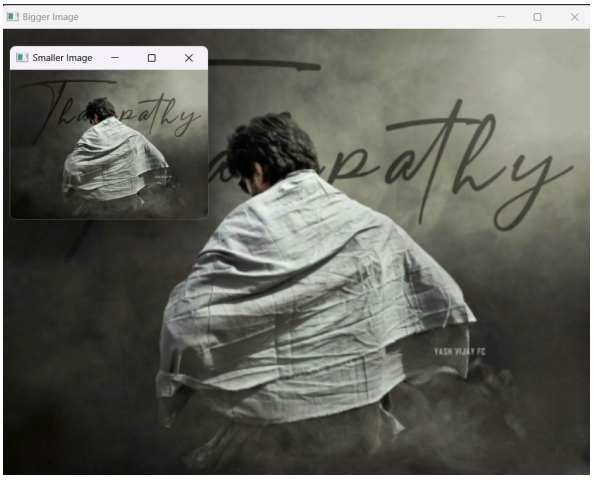
AIM :

To write a python program to small and big a image using opencv3

PROGRAM :

```
import cv2
img=cv2.imread("C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg")
height, width = img.shape[:2]
scale_factor = 1.5
bigger_img = cv2.resize(img, (int(width * scale_factor), int(height * scale_factor)))
scale_factor = 0.5
smaller_img = cv2.resize(img, (int(width * scale_factor), int(height * scale_factor)))
cv2.imshow("Original Image", img)
cv2.imshow("Bigger Image", bigger_img)
cv2.imshow("Smaller Image", smaller_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



6.Write a python program to blur a image using opencv

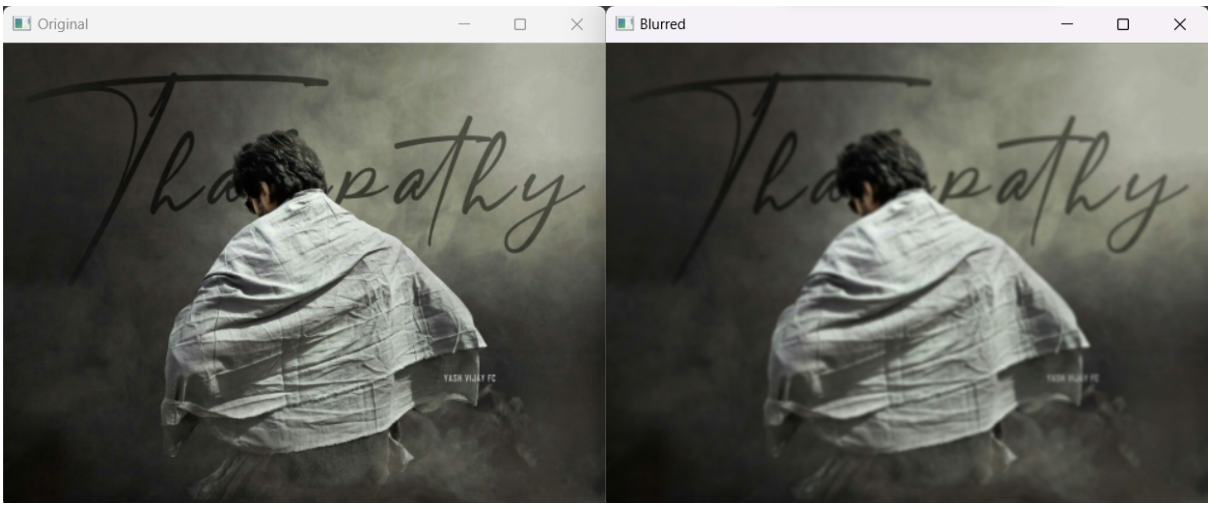
AIM :

To write a python program to blur a image using opencv

PROGRAM :

```
import cv2
import numpy as np
img=cv2.imread("C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg")
blur = cv2.GaussianBlur(img, (5, 5), 0)
cv2.imshow("Original", img)
cv2.imshow("Blurred", blur)
cv2.waitKey(0)
```

OUTPUT :



7.Write a python program to clock counter a image using opencv

AIM :

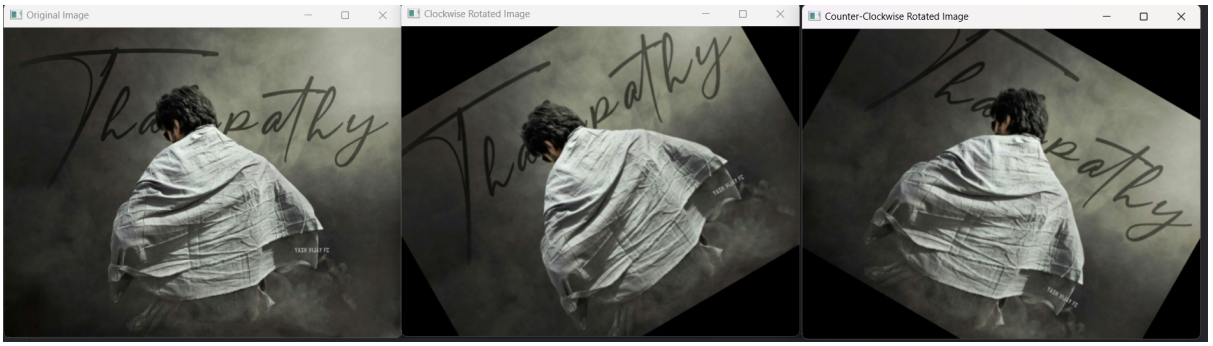
To write a python program to clock counter a image using opencv

PROGRAM :

```
import cv2
img=cv2.imread("C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg")
height, width = img.shape[:2]
angle = 30
centre = (width/2, height/2)
M = cv2.getRotationMatrix2D(centre, angle, 1)
clockwise_img = cv2.warpAffine(img, M, (width, height))
M = cv2.getRotationMatrix2D(centre, -angle, 1)
counter_clockwise_img = cv2.warpAffine(img, M, (width, height))
```

```
cv2.imshow("Original Image", img)
cv2.imshow("Clockwise Rotated Image", clockwise_img)
cv2.imshow("Counter-Clockwise Rotated Image", counter_clockwise_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



8.Write a python program to Edge a image using opencv

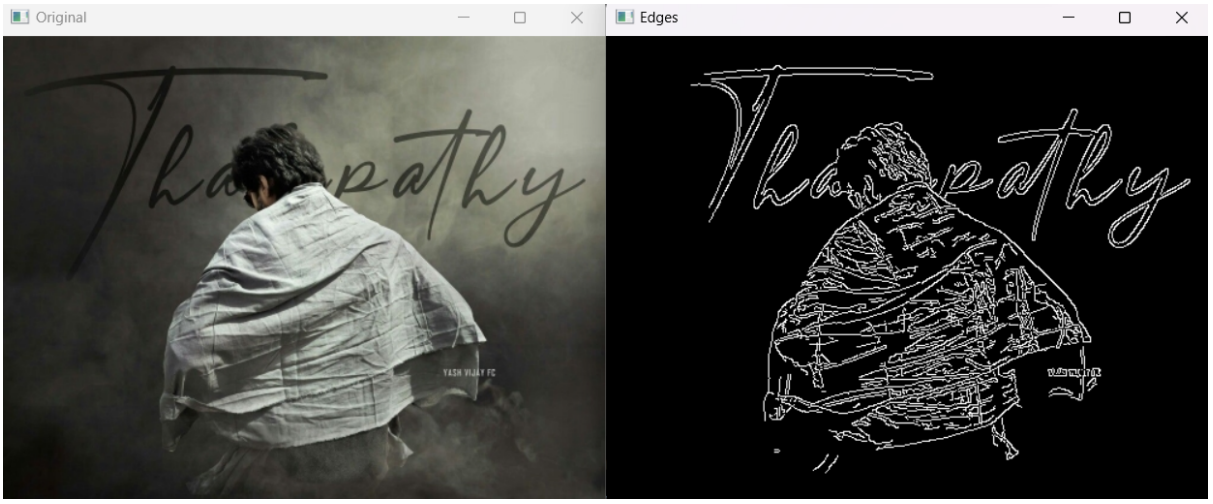
AIM :

To write a python program to Edge a image using opencv

PROGRAM :

```
import cv2
import numpy as np
img=cv2.imread("C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg")
kernel = np.ones((5, 5), np.uint8)
eroded = cv2.erode(img, kernel, iterations=1)
cv2.imshow("Original", img)
cv2.imshow("Eroded", eroded)
cv2.waitKey(0)
```

OUTPUT :



9.Write a python program to Dilated a image using opencv

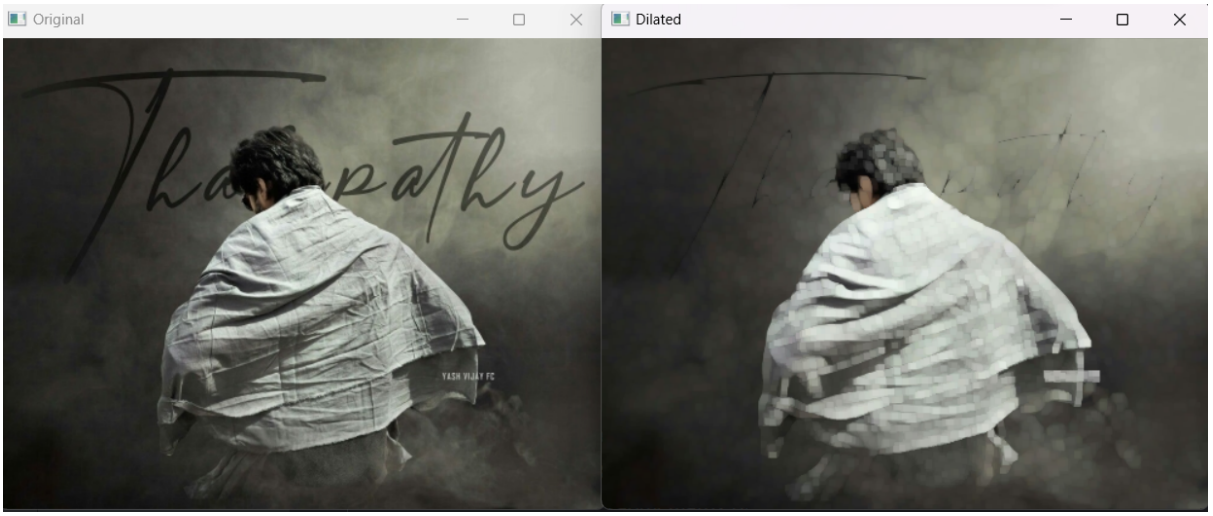
AIM :

To write a python program to Dilated a image using opencv

PROGRAM :

```
import cv2
import numpy as np
img=cv2.imread("C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg")
edges = cv2.Canny(img, 100, 200)
kernel = np.ones((5, 5), np.uint8)
dilated = cv2.dilate(img, kernel, iterations=1)
cv2.imshow("Original", img)
cv2.imshow("Dilated", dilated)
cv2.waitKey(0)
```

OUTPUT :



10. Write a python program to Translate a image using opencv

AIM :

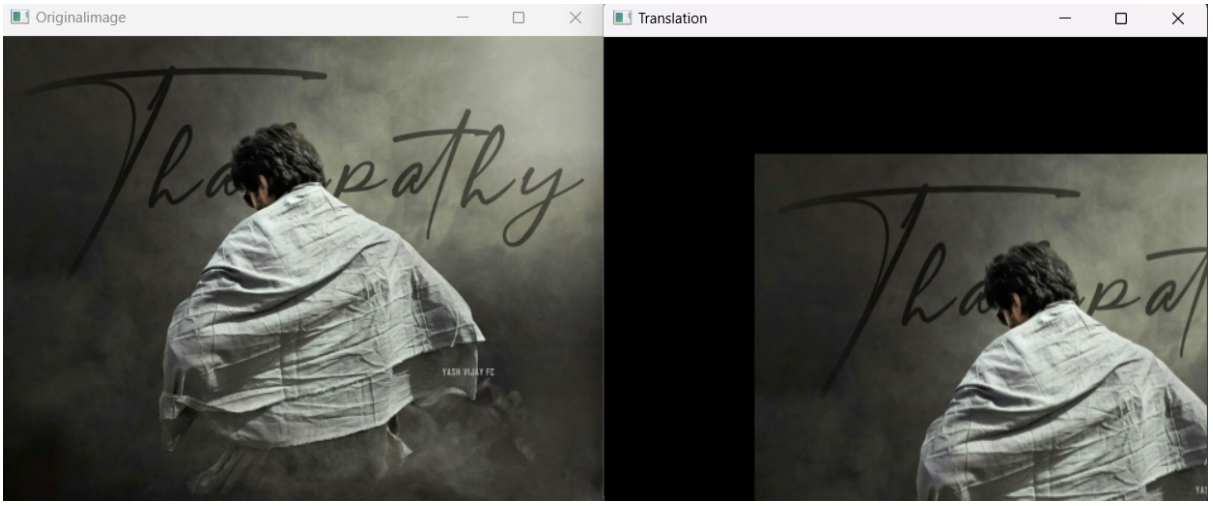
+

To write a python program to Translate a image using opencv

PROGRAM :

```
import cv2
import numpy as np
image=cv2.imread('C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg')
height, width = image.shape[:2]
quarter_height, quarter_width = height / 4, width / 4
T = np.float32([[1, 0, quarter_width], [0, 1, quarter_height]])
img_translation = cv2.warpAffine(image, T, (width, height))
cv2.imshow("Originalimage", image)
cv2.imshow('Translation', img_translation)
cv2.waitKey()
cv2.destroyAllWindows()
```

OUTPUT :



11.Write a python program to affin transform a image using opencv

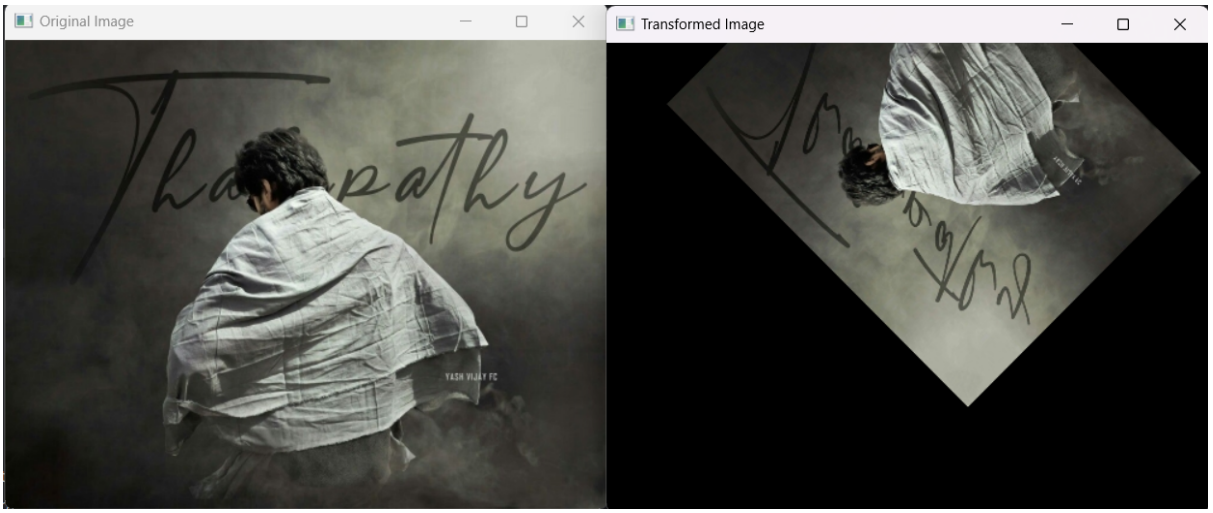
AIM :

To write a python program to affin transform a image using opencv

PROGRAM :

```
import cv2
import numpy as np
img=cv2.imread("C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg")
M = np.float32([[0.5, 0.5, 50], [0.5, -0.5, 50]])
dst = cv2.warpAffine(img, M, (img.shape[1], img.shape[0]))
cv2.imshow("Original Image", img)
cv2.imshow("Transformed Image", dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



12. Write a python program to prespective transform a image using opencv

AIM :

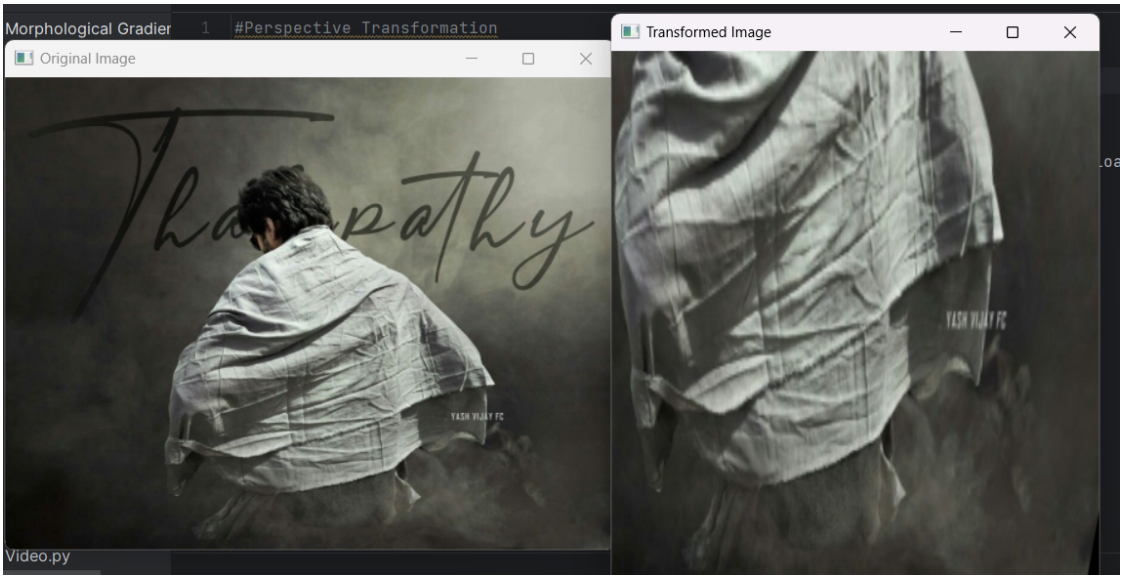
To write a python program to prespective transform a image using opencv

PROGRAM :

```
import cv2
import numpy as np
img=cv2.imread("C:/Users/surya/PycharmProjects/pythonProject/thalapathy.jpg")
roi_points = np.array([(150, 200), (450, 200), (550, 500), (50, 500)])
target_points = np.array([(0, 0), (400, 0), (400, 600), (0, 600)])
M = cv2.getPerspectiveTransform(roi_points.astype(np.float32), target_points.astype(np.float32))
dst = cv2.warpPerspective(img, M, (400, 600))
cv2.imshow("Original Image", img)
cv2.imshow("Transformed Image", dst)
cv2.waitKey(0)
```

cv2.destroyAllWindows()

OUTPUT :



13.Write a python program to prespective transform a video using opencv

AIM :

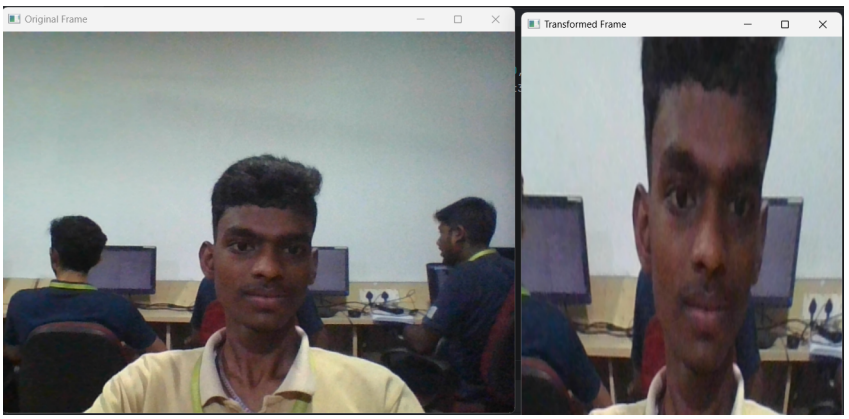
To write a python program to prespective transform a video using opencv

PROGRAM :

```
import cv2
import numpy as np
roi_points = np.array([(150, 200), (450, 200), (550, 500), (50, 500)])
target_points = np.array([(0, 0), (400, 0), (400, 600), (0, 600)])
M = cv2.getPerspectiveTransform(roi_points.astype(np.float32),
target_points.astype(np.float32))
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    dst = cv2.warpPerspective(frame, M, (400, 600))
    cv2.imshow("Original Frame", frame)
```

```
    cv2.imshow("Transformed Frame", dst)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

OUTPUT :



14. Write a python program to transform a image with Homography Matrix using opencv

AIM :

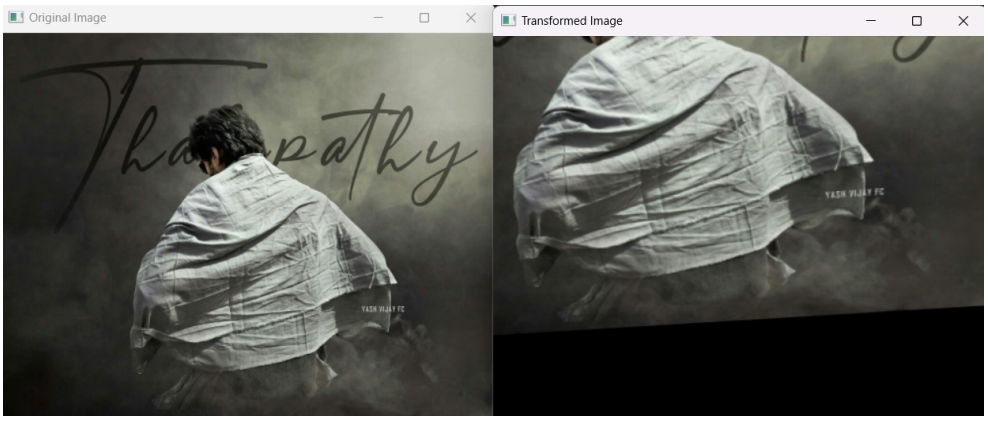
To Write a python program to transform a image with Homography Matrix using opencv

PROGRAM :

```
import cv2
import numpy as np
image = cv2.imread('F:/2.jpg')
target_points = np.array([[0, 0], [500, 0], [500, 500], [0, 500]],
dtype=np.float32)
source_points = np.array([[141, 131], [480, 159], [493, 630], [64, 601]],
dtype=np.float32)
homography_matrix, _ = cv2.findHomography(source_points, target_points)
transformed_image = cv2.warpPerspective(image, homography_matrix, (500,
500))
cv2.imshow('Original Image', image)
```

```
cv2.imshow('Transformed Image', transformed_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



15.Write a python program to direct linear transformation a image using opencv

AIM :

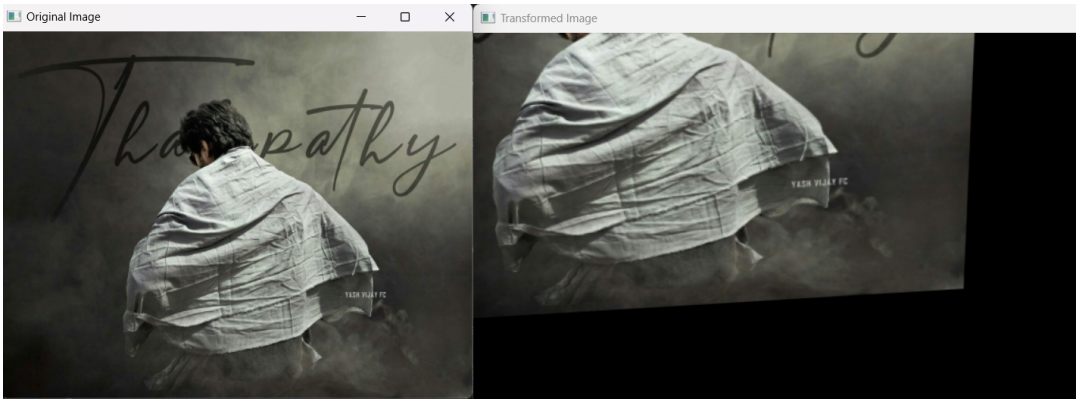
To Write a python program to direct linear transformation a image using opencv.

PROGRAM :

```
import cv2
import numpy as np
image = cv2.imread('E:/1.jpg')
source_points = np.array([[141, 131], [480, 159], [493, 630], [64, 601]],
dtype=np.float32)
target_points = np.array([[0, 0], [500, 0], [500, 500], [0, 500]],
dtype=np.float32)
num_points = source_points.shape[0]
A = np.zeros((2*num_points, 9), dtype=np.float64)
for i in range(num_points):
    x, y = source_points[i]
    u, v = target_points[i]
```

```
    A[2*i] = [x, y, 1, 0, 0, 0, -u*x, -u*y, -u]
    A[2*i+1] = [0, 0, 0, x, y, 1, -v*x, -v*y, -v]
_, _, V = np.linalg.svd(A)
h = V[-1, :]
homography_matrix = h.reshape((3, 3))
transformed_image = cv2.warpPerspective(image, homography_matrix, (500, 500))
cv2.imshow('Original Image', image)
cv2.imshow('Transformed Image', transformed_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



16.Write a python program to detect edge using canny in a image using opencv

AIM :

To Write a python program to detect edge using canny in a image using opencv.

PROGRAM :

```
import cv2
img = cv2.imread('E:/1.jpg', 0)
edges = cv2.Canny(img, 100, 200)
```

```
cv2.imshow('Original Image', img)
cv2.imshow('Canny Edge Detection', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :

17.Write a python program to Edge detection using Sobel Matrix along X axisa image using opencv

AIM :

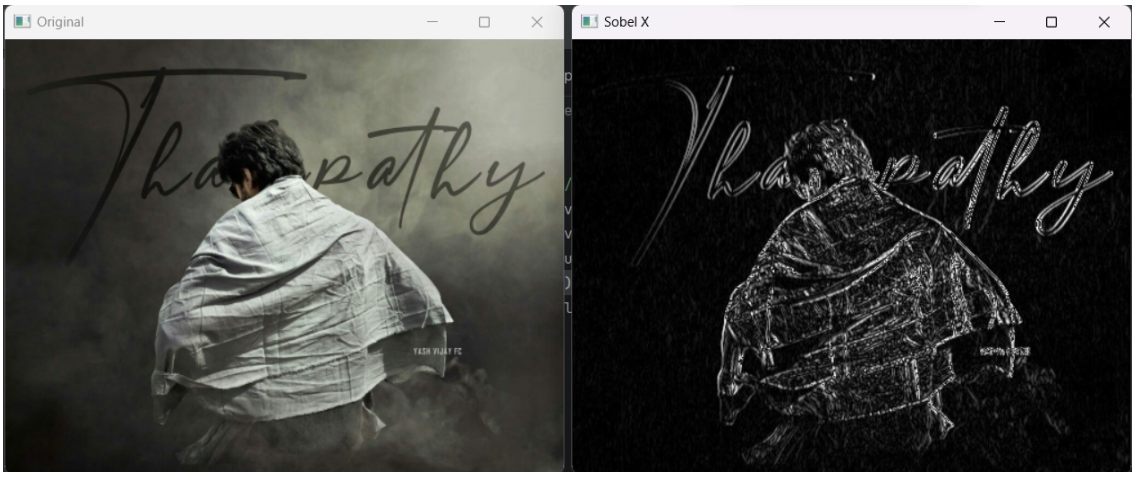
To Write a python program to Edge detection using Sobel Matrix along X axisa image using opencv

PROGRAM :

```
import cv2
import numpy as np
# Load the image
img = cv2.imread('E:/1.jpg', 0)
# Apply Sobel filter along the X-axis
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
# Convert the output to absolute values
sobelx = np.abs(sobelx)
# Display the results
cv2.imshow('Original Image', img)
```

```
cv2.imshow('Sobel Edge Detection (X-axis)', sobelx)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



18. Write a python program to Edge detection using Sobel Matrix along y axis a image using opencv

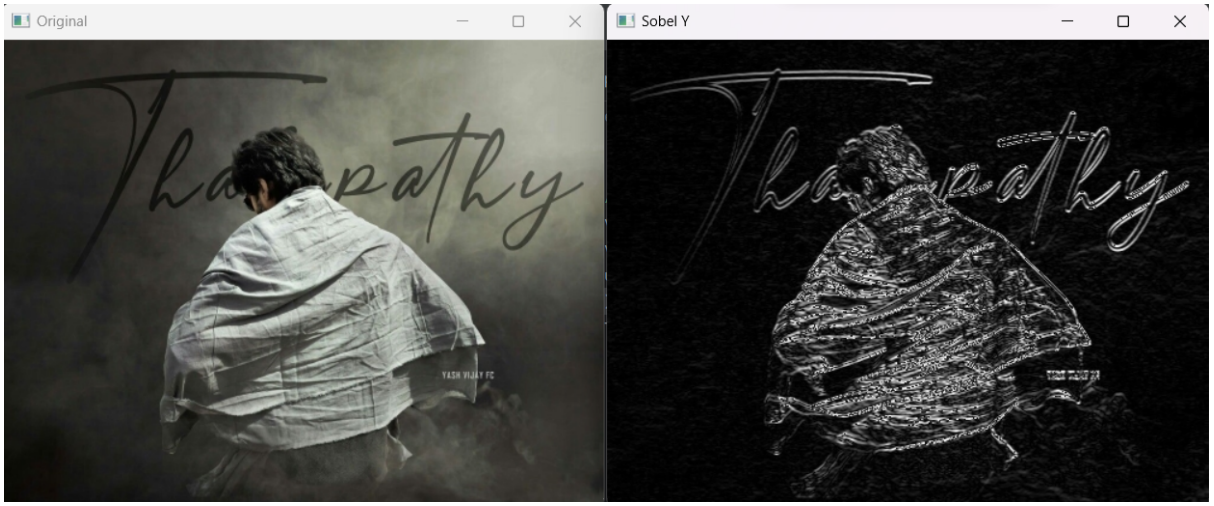
AIM :

To Write a python program to Edge detection using Sobel Matrix along y axis a image using opencv

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg', 0)
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
sobely = np.abs(sobely)
cv2.imshow('Original Image', img)
cv2.imshow('Sobel Edge Detection (Y-axis)', sobely)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



19. Write a python program to Edge detection using Sobel Matrix along XY axis a image using opencv

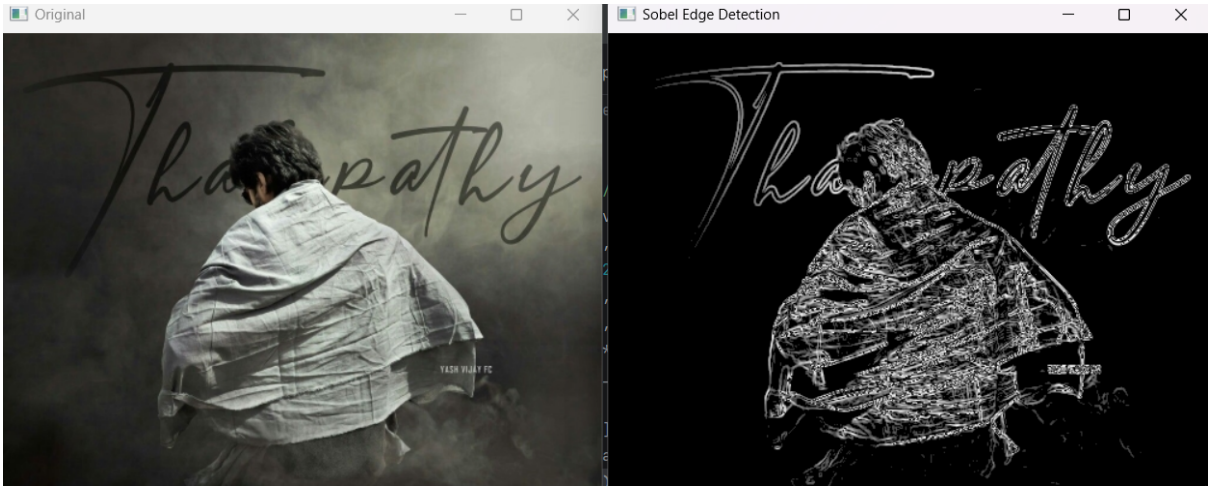
AIM :

To Write a python program to Edge detection using Sobel Matrix along XY axis a image using opencv

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg', cv2.IMREAD_GRAYSCALE)
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
sobel_combined = cv2.addWeighted(sobelx, 0.5, sobely, 0.5, 0)
cv2.imshow('Sobel Edge Detection', sobel_combined)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



20 .Write a python program to Sharpening of Image using Laplacian mask with negative center coefficient a image using opencv

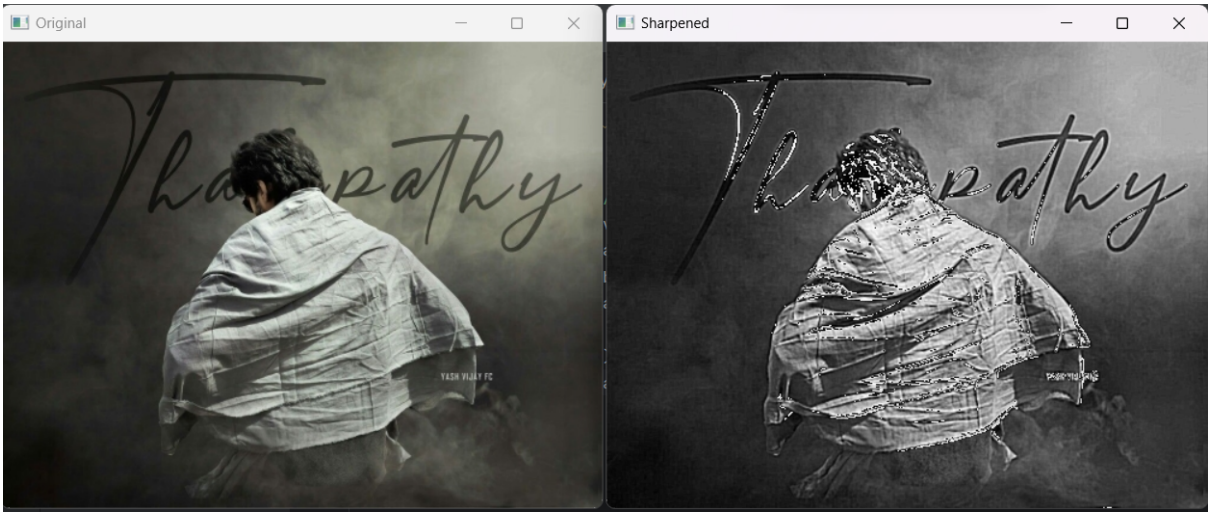
AIM :

To Write a python program to Sharpening of Image using Laplacian mask with negative center coefficient. a image using opencv

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg', cv2.IMREAD_GRAYSCALE)
laplacian_filter = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
sharpened = cv2.filter2D(img, -1, laplacian_filter)
sharpened_img = cv2.addWeighted(img, 1.5, sharpened, -0.5, 0)
cv2.imshow('Original Image', img)
cv2.imshow('Sharpened Image', sharpened_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



21. Write a python program to Sharpening of Image using Laplacian mask with extension of neighbor diagonals . a image using opencv

AIM :

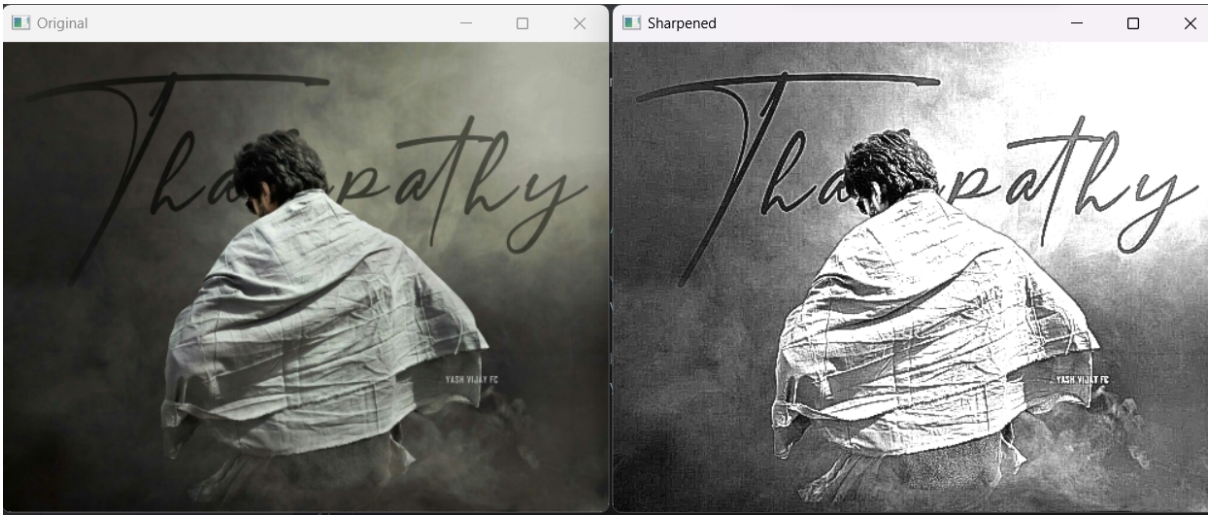
To Write a python program to Sharpening of Image using Laplacian mask with extension of neighbor diagonals . a image using opencv

PROGRAM :

```
import cv2
import numpy as np
image = cv2.imread('E:/1.jpg')
kernel = np.array([[0, -1, 0],
                   [-1, 8, -1],
                   [0, -1, 0]])
sharpened = cv2.filter2D(image, -1, kernel)
cv2.imshow('Original', image)
cv2.imshow('Sharpened', sharpened)
cv2.waitKey(0)
```

cv2.destroyAllWindows()

OUTPUT :



22.Write a python program to laplacian mask with positive center coefficient a image using opencv

AIM :

To Write a python program to laplacian mask with positive center coefficient a image using opencv

PROGRAM :

```
import cv2
import numpy as np
kernnal = np.array([[0,1,0],[-1,5,-1],[0,-1,0]])
img = cv2.imread("3.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
sharpened = cv2.filter2D(gray,-1,kernnal)
sharpened = cv2.multiply(sharpened,0.5)
cv2.imshow("sharpened",sharpened)
cv2.waitKey(0)
```

OUTPUT :

23. Perform Sharpening of Image using unsharp masking.

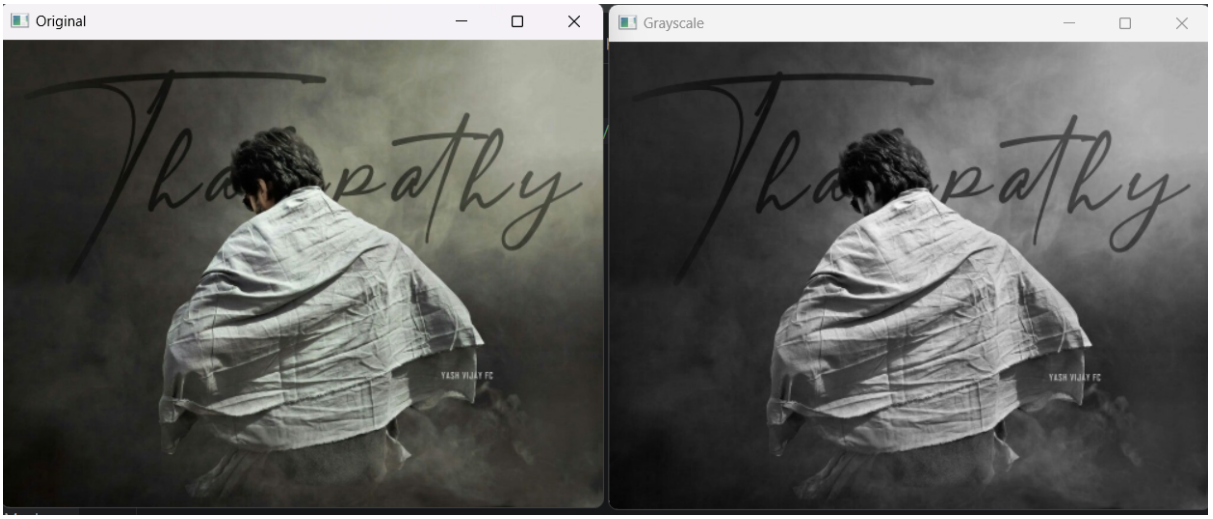
AIM :

To Perform Sharpening of Image using unsharp masking.

PROGRAM :

```
import cv2
import numpy as np
image = cv2.imread('input_image.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 0)
mask = cv2.subtract(gray, blur)
sharpening_factor = 1.5
adjusted_mask = cv2.multiply(mask, sharpening_factor)
sharpened = cv2.add(gray, adjusted_mask)
sharpened = np.clip(sharpened, 0, 255)
cv2.imshow('Sharpened Image', sharpened)
```

OUTPUT :



24. Perform Sharpening of Image using High-Boost Masks.

AIM :

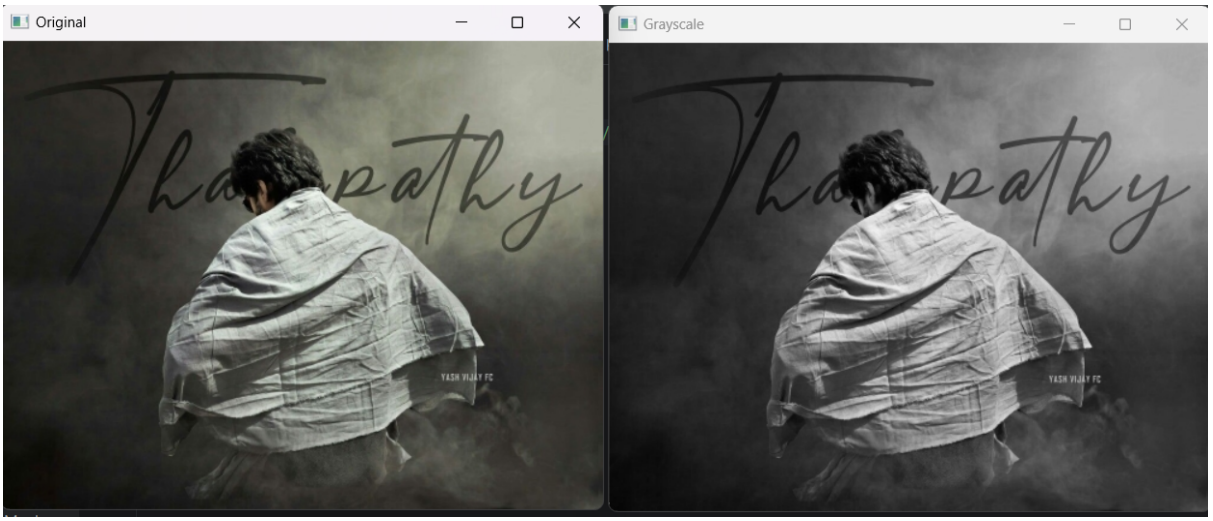
To Perform Sharpening of Image using High-Boost Masks.

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread("3.jpg")
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
kernel = np.array([[0,-1,0],[-1,1020,-1], [0,-1,0]])
kernel_1 = np.array([[-1,-1,-1],[-1,100000,-1],[-1,-1,-1]])
filtered_img = cv2.filter2D(gray_img, -1, kernel)
sharpened_img = cv2.addWeighted(gray_img, 1.5, filtered_img, -0.5, 0)
sharpened_img = cv2.cvtColor(sharpened_img, cv2.COLOR_GRAY2BGR)
filtered_img_1 = cv2.filter2D(gray_img, -1, kernel_1)
sharpened_img_1 = cv2.addWeighted(gray_img, 1.5, filtered_img_1, -0.5, 0)
sharpened_img_1 = cv2.cvtColor(sharpened_img_1,
cv2.COLOR_GRAY2BGR)
```

```
cv2.imshow('Original Image', img)
cv2.imshow('Sharpened Image', sharpened_img)
cv2.imshow('Sharpened_Image', sharpened_img_1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



25. Perform Sharpening of Image using Gradient masking.

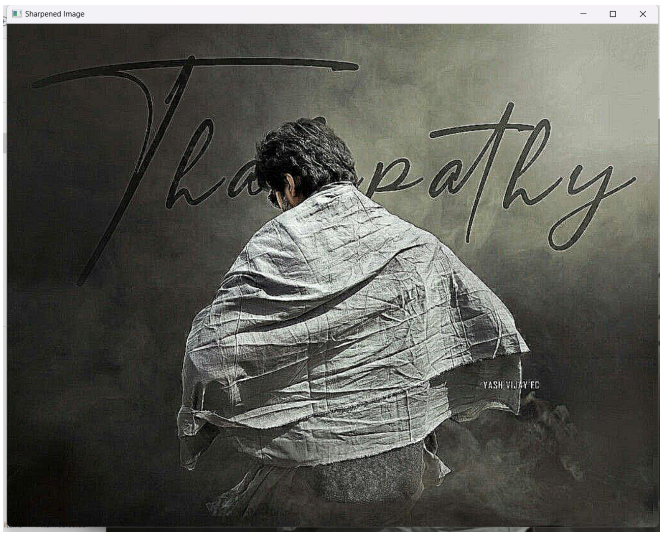
AIM :

To Perform Sharpening of Image using Gradient masking.

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
kernel = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
sharpened_img = cv2.filter2D(img, -1, kernel)
cv2.imshow('Input Image', img)
cv2.imshow('Sharpened Image', sharpened_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



26. Insert water marking to the image using OpenCV.

AIM :

To Insert water marking to the image using OpenCV.

PROGRAM :

```
import cv
img = cv2.imread("E:/1.jpg")
watermark = cv2.imread("E:/1.jpg", cv2.IMREAD_UNCHANGED)
watermark = cv2.resize(watermark, (img.shape[1], img.shape[0]))
result = cv2.addWeighted(img, 1, watermark, 0.3, 0)
cv2.imwrite('path/to/result.jpg', result)
```

27. Do Cropping, Copying and pasting image inside another image using OpenCV.

AIM :

To Do Cropping, Copying and pasting image inside another image using OpenCV.

PROGRAM :

```
import cv2
import numpy as np
img1 = cv2.imread("E:/1.jpg")
img2 = cv2.imread("E:/1.jpg")
x, y, w, h = 100, 100, 200, 200
roi = img1[y:y+h, x:x+w]
roi_resized = cv2.resize(roi, (w, h))
img2gray = cv2.cvtColor(roi_resized,cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 10, 255, cv2.THRESH_BINARY)
mask_inv = cv2.bitwise_not(mask)
img2_bg = cv2.bitwise_and(img2,img2,mask = mask_inv)
img1_fg = cv2.bitwise_and(roi_resized,roi_resized,mask = mask)
dst = cv2.add(img2_bg,img1_fg)
img2[y:y+h, x:x+w] = dst
cv2.imshow('Result', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :

28. Find the boundary of the image using Convolution kernel for the given image.

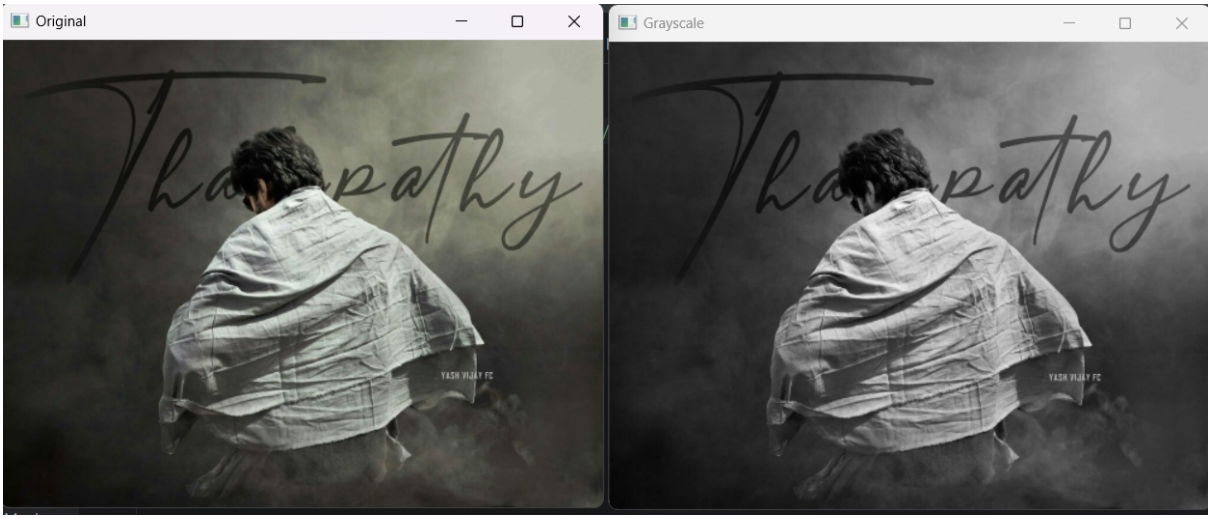
AIM :

To Find the boundary of the image using Convolution kernel for the given image.

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
kernel = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
sharpened_img = cv2.filter2D(img, -1, kernel)
cv2.imshow('Input Image', img)
cv2.imshow('Sharpened Image', sharpened_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



29. Morphological operations based on OpenCV using Erosion technique.

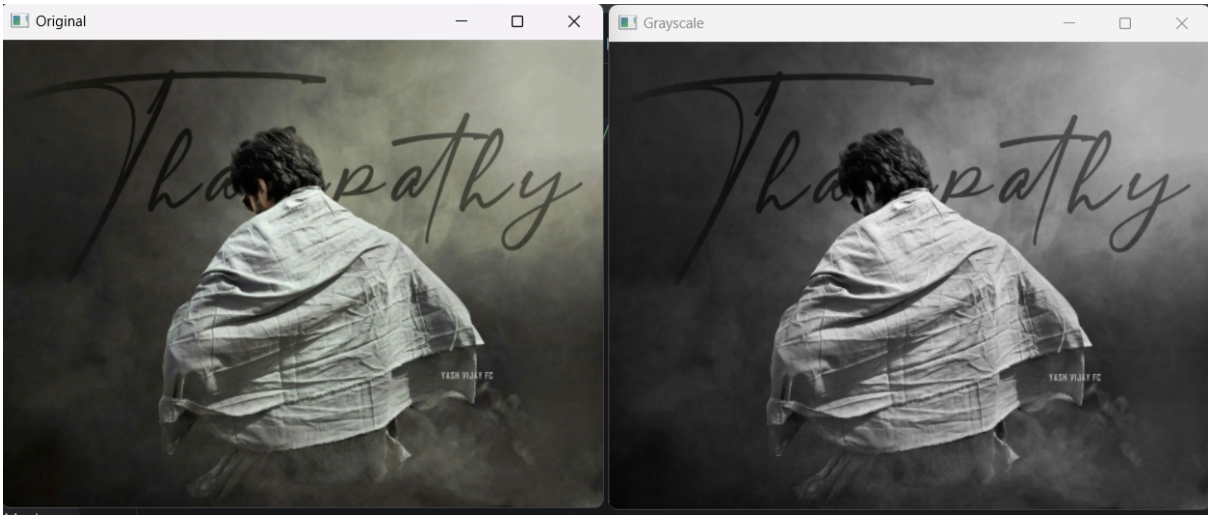
AIM :

To Find the boundary of the image using Convolution kernel for the given image.

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg", cv2.IMREAD_GRAYSCALE)
kernel = np.ones((5, 5), np.uint8)
eroded_img = cv2.erode(img, kernel, iterations=1)
cv2.imshow("Original Image", img)
cv2.imshow("Eroded Image", eroded_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



30. Morphological operations based on OpenCV using Dilation technique.

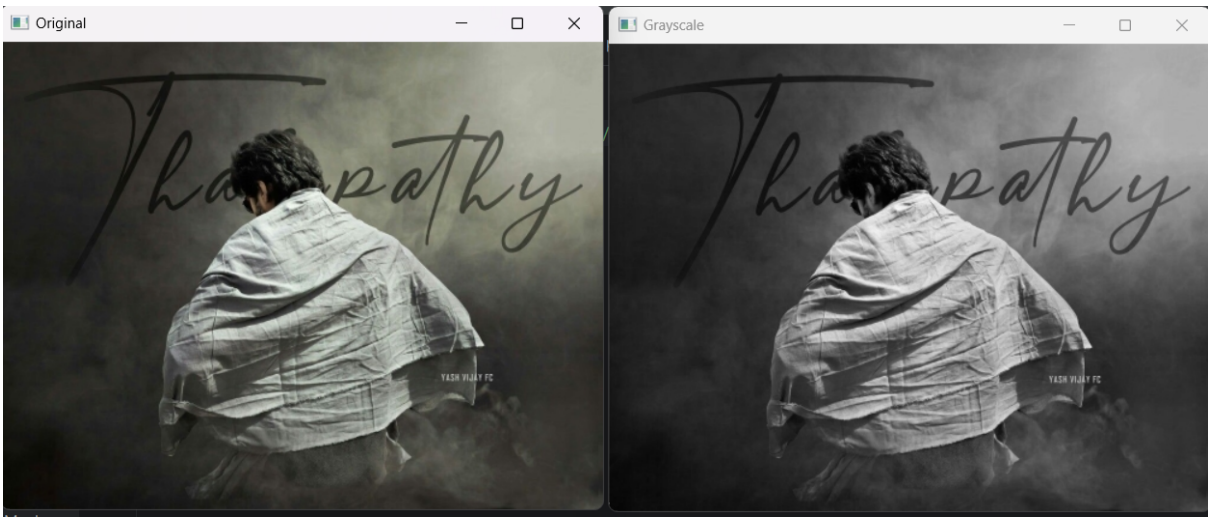
AIM :

To Morphological operations based on OpenCV using Dilation technique.

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
kernel = np.ones((5,5),np.uint8)
dilation = cv2.dilate(img,kernel,iterations = 1)
cv2.imshow('Original', img)
cv2.imshow('Dilated', dilation)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



31. Morphological operations based on OpenCV using Opening technique.

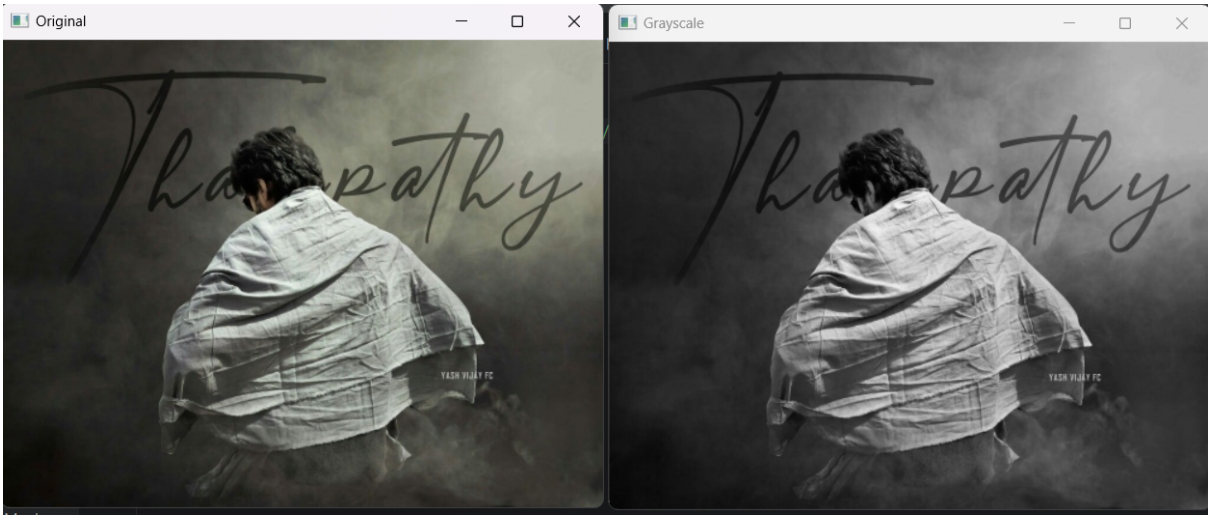
AIM :

To Morphological operations based on OpenCV using Opening technique.

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
kernel = np.ones((5,5),np.uint8)
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
cv2.imshow('Original', img)
cv2.imshow('Opened', opening)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT :



32. Morphological operations based on OpenCV using Closing technique.

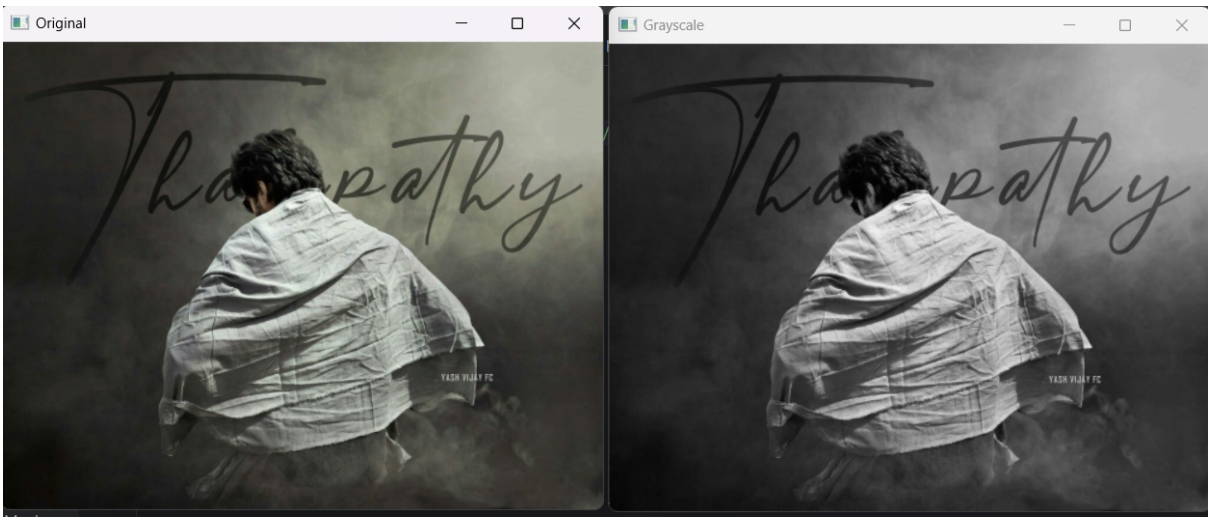
AIM :

To Morphological operations based on OpenCV using Closing technique.

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg", cv2.IMREAD_GRAYSCALE)kernel =
np.ones((5,5), np.uint8)
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
cv2.imshow('Original', img)
cv2.imshow('Closing', closing)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT



33. Morphological operations based on OpenCV using Morphological Gradient technique.

AIM :

To Morphological operations based on OpenCV using Closing technique.

PROGRAM :

```
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg', 0)
kernel = np.ones((3,3), np.uint8)
gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)
cv2.imshow('Morphological Gradient', gradient)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

34. Morphological operations based on OpenCV using Top hat technique.

AIM :

To Morphological operations based on OpenCV using Closing technique.

PROGRAM :

```
import cv2
import numpy as np
image = cv2.imread('2.jpg', 0)  # 0 flag reads the image as grayscale
kernel = np.ones((5, 5), np.uint8)  # 5x5 kernel with all ones
tophat = cv2.morphologyEx(image, cv2.MORPH_TOPHAT, kernel)
cv2.imshow('Top Hat', tophat)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

35. Morphological operations based on OpenCV using Black hat technique.

AIM :

To Morphological operations based on OpenCV using Closing technique.

PROGRAM :

```
import cv2
import numpy as np
image = cv2.imread('1.jpg', 0
kernel = np.ones((5, 5), np.uint8)  # 5x5 kernel with all ones
blackhat = cv2.morphologyEx(image, cv2.MORPH_BLACKHAT, kernel)
cv2.imshow('Black Hat', blackhat)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

36. Recognise watch from the given image by general Object recognition using OpenCV.

AIM :

To Recognise watch from the given image by general Object recognition using OpenCV.

PROGRAM :

```
import cv2
image = cv2.imread('watch.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
watch_cascade = cv2.CascadeClassifier('watch_cascade.xml')
watches = watch_cascade.detectMultiScale(gray)
for (x, y, w, h) in watches:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.putText(image, 'Watch', (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
cv2.imshow('Object Recognition', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

37. Using Opencv play Video in Reverse mode.

AIM :

To Using Opencv play Video in Reverse mode.

PROGRAM :

```
import cv2

video = cv2.VideoCapture('opencv_video.mp4')
total_frames = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
fps = video.get(cv2.CAP_PROP_FPS)
output_filename = 'reversed_video.mp4'
codec = cv2.VideoWriter_fourcc(*'mp4v')
```

```
output = cv2.VideoWriter(output_filename, codec, fps, (int(video.get(3)),
int(video.get(4))))
for frame_number in range(total_frames - 1, -1, -1):
    video.set(cv2.CAP_PROP_POS_FRAMES, frame_number)
    ret, frame = video.read()
    if ret:
        output.write(frame)
    else:
        break
video.release()
output.release()
print("Reversed video saved as:", output_filename)
```

38. Face Detection using Opencv.

AIM :

To Face Detection using Opencv.

PROGRAM :

```
import cv2
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
image = cv2.imread('E:/1.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5)
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
cv2.imshow('Output', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

39. Vehicle Detection in a Video frame using OpenCV .

AIM :

To Vehicle Detection in a Video frame using OpenCV.

PROGRAM :

```
import cv2

cap = cv2.VideoCapture('F:/WhatsApp Video 2023-05-11 at 12.12.02.mp4')
car_cascade = cv2.CascadeClassifier('haarcascade_car.xml')
while True:
    ret, frame = cap.read()
    if not ret:
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cars = car_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5)
    for (x, y, w, h) in cars:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
    cv2.imshow('Vehicle Detection', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
 cap.release()cv2.destroyAllWindows()
```

40. Draw Rectangular shape and extract objects.

AIM :

To Draw Rectangular shape and extract objects.

PROGRAM :

```
import cv2
image = cv2.imread('watch.jpg')
start_point = (100, 100)
```

```
end_point = (300, 400)
color = (0, 255, 0)
thickness = 2
cv2.rectangle(image, start_point, end_point, color, thickness)
roi = image[start_point[1]:end_point[1], start_point[0]:end_point[0]]
cv2.imshow('Image with Rectangle', image)
cv2.imshow('Extracted Object', roi)
cv2.waitKey(0)
cv2.destroyAllWindows()
```