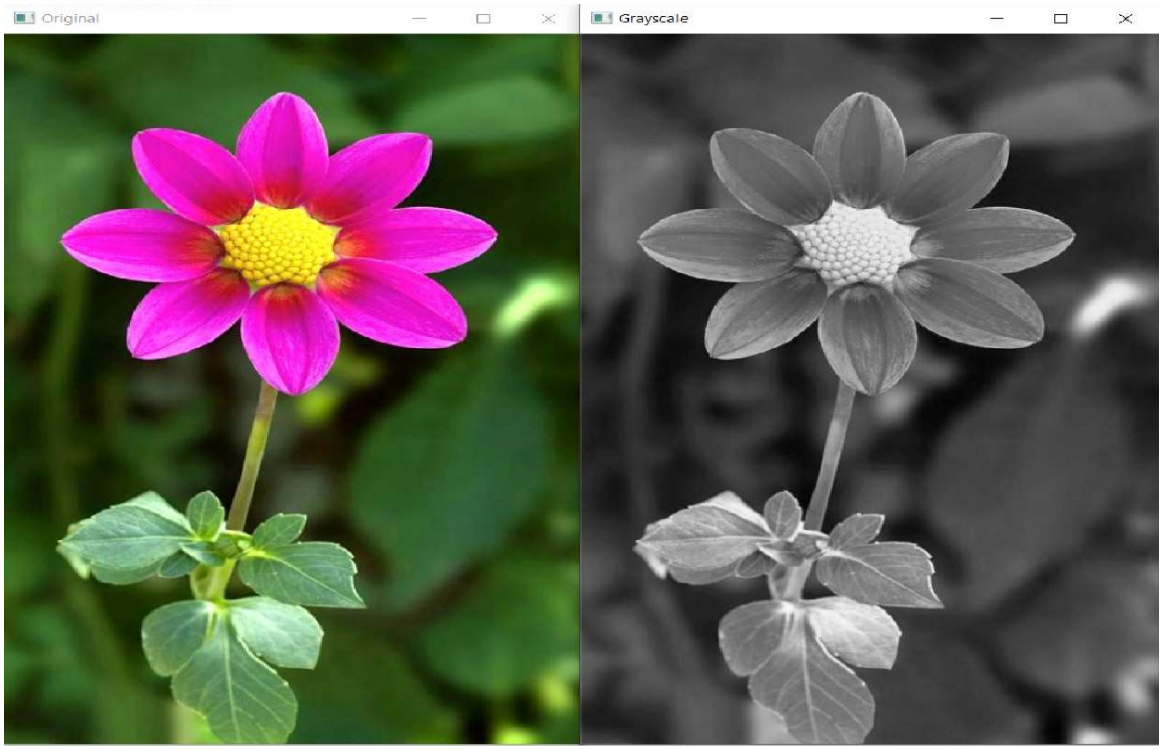
1. Convert an Image to Grayscale:

PROGRAM:

```
import cv2
image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ")
cv2.imshow('Original', image)
cv2.waitKey(0)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Grayscale', gray_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

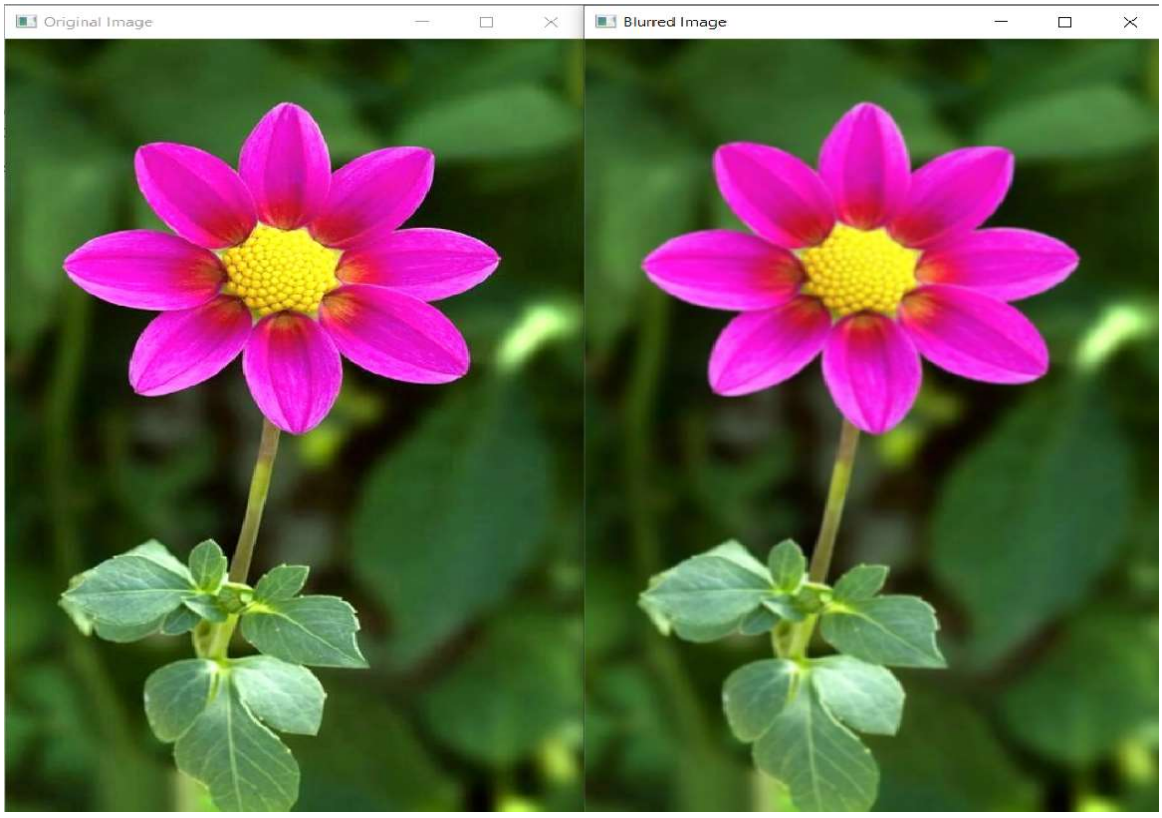


2. CONVERT AN IMAGE TO BLUR:

PROGRAM:

```
import cv2
image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ")
k_size = (5, 5)
sigma_x = 0
blurred_image = cv2.GaussianBlur(image, k_size, sigma_x)
cv2.imwrite('blurred_image.jpg', blurred_image)
cv2.imshow('Original Image', image)
cv2.imshow('Blurred Image', blurred_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

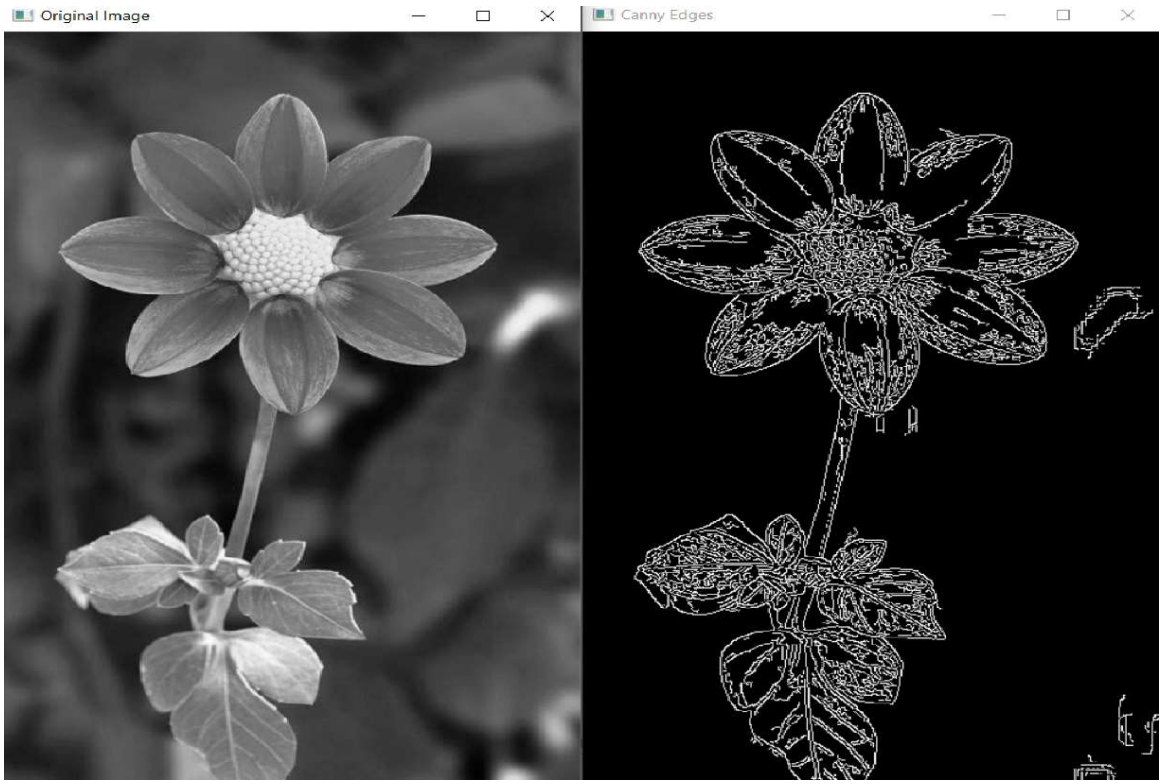


3. Convert an Image to show outline using Canny function.

PROGRAM:

```
import cv2

image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg
cv2.IMREAD_GRAYSCALE)

edges = cv2.Canny(image, threshold1=30, threshold2=100)

cv2.imshow('Original Image', image)

cv2.imshow('Canny Edges', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:



4. Dilate an Image using Dilate function:

PROGRAM:

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ",
cv2.IMREAD_GRAYSCALE)

kernel = np.ones((5, 5), np.uint8)

dilated_image = cv2.dilate(image, kernel, iterations=1)

cv2.imshow('Original Image', image)

cv2.imshow('Dilated Image', dilated_image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:

5. Erode an Image using erode function:

PROGRAM:

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg
cv2.IMREAD_GRAYSCALE)

kernel = np.ones((5, 5), np.uint8)

eroded_image = cv2.erode(image, kernel, iterations=1)

cv2.imshow('Original Image', image)

cv2.imshow('Eroded Image', eroded_image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

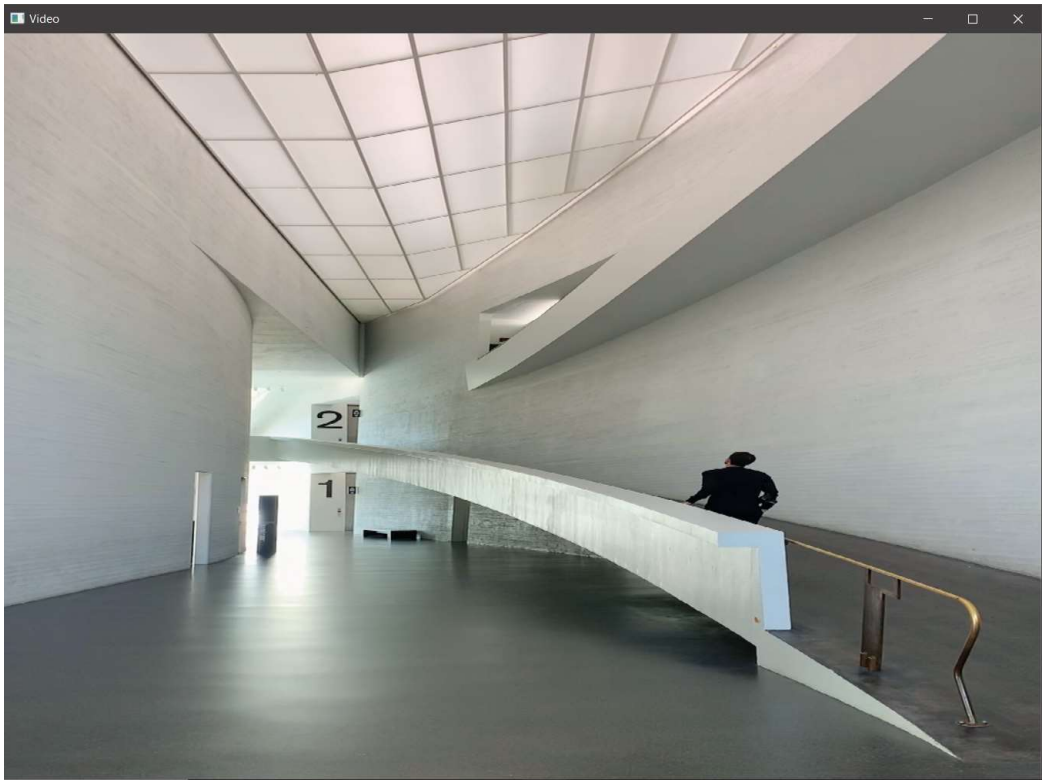
OUTPUT:

6. Perform basic video processing operations.

PROGRAM:

```
import cv2
def play_video(video_path, speed=1.0):
cap = cv2.VideoCapture("C:/Users/thejas/OneDrive/Desktop/COMPUTER VISION/WALK.mp4")
 if not cap.isOpened():
 print("Error: Could not open video.")
 return
 fps = cap.get(cv2.CAP_PROP_FPS)
 cv2.namedWindow("Video", cv2.WINDOW_NORMAL)
  while True:
  ret, frame = cap.read()
  if not ret:
  break
  cv2.resizeWindow("Video", frame.shape[1], frame.shape[0])
   cv2.imshow("Video", frame)
   delay = int(1000 / (fps * speed))
   if cv2.waitKey(delay) & 0xFF == 27:
  break
 cap.release()
 cv2.destroyAllWindows()
play_video('your_video.mp4')
play_video('your_video.mp4', speed=0.1)
play_video('your_video.mp4', speed=10.0)
```

OUTPUT:



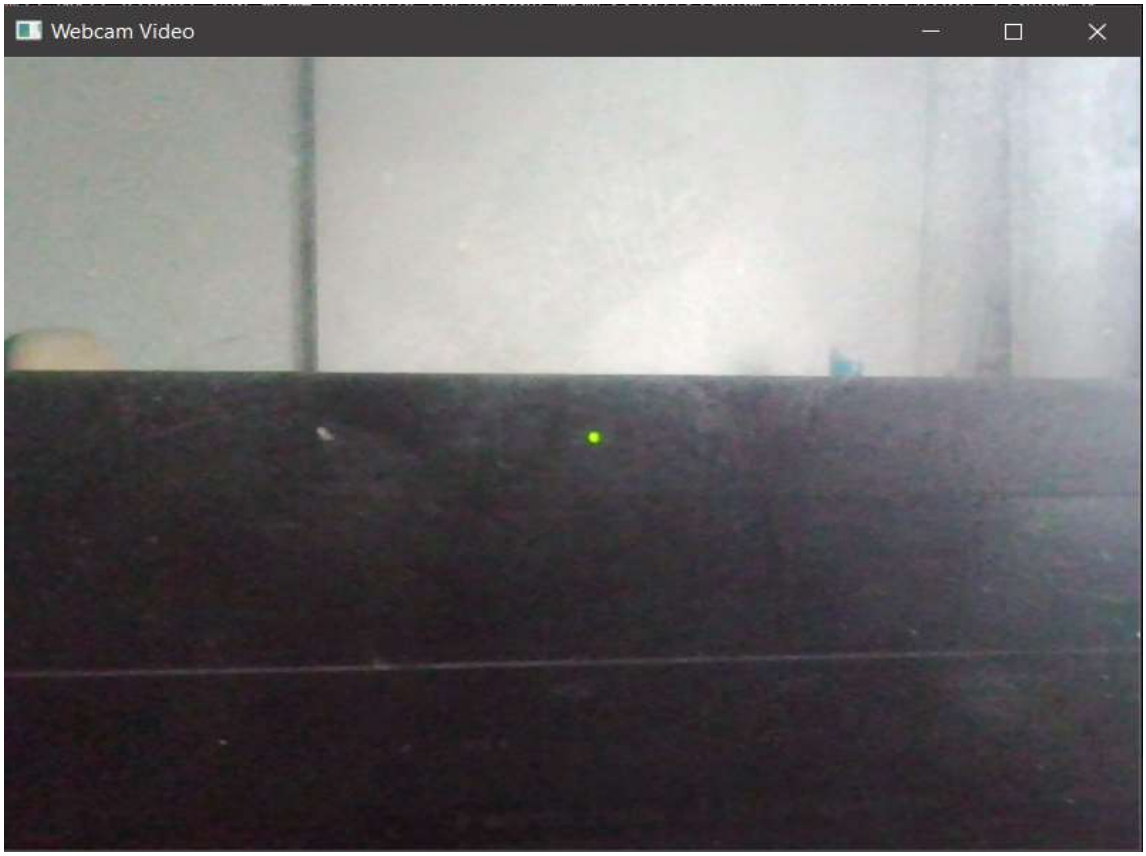
7. Capture video from web Camera and Display the video.

PROGRAM:

```
import cv2
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Could not open the camera")
    exit()
cv2.namedWindow("Webcam Video")
speed_factor = 1.0
while True:
    ret, frame = cap.read()
    if not ret:
```

```
        break
    cv2.imshow("Webcam Video", frame)
    key = cv2.waitKey(1)
    if key == ord("+"):
        speed_factor += 5.0
    elif key == ord("-"):
        speed_factor -= 5.0
    elif key == ord('q'):
        break
    cap.set(cv2.CAP_PROP_FPS, 30 * speed_factor)
cap.release()
cv2.destroyAllWindows()
```

OUTPUT:

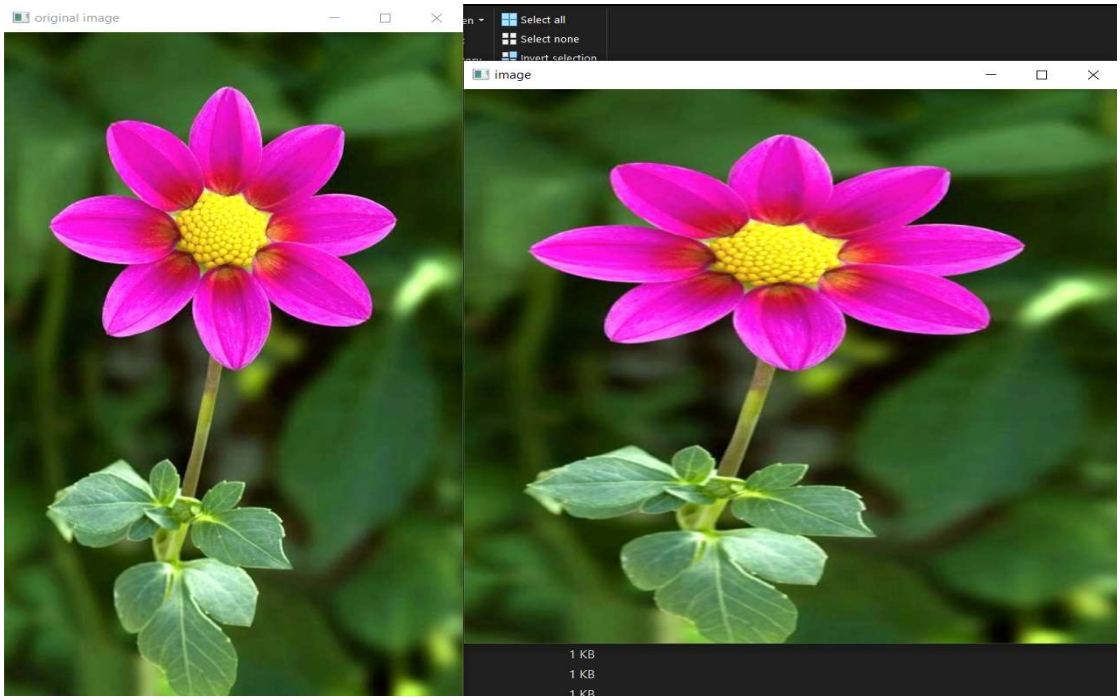


8. Scaling an image to its Bigger and Smaller sizes.

PROGRAM:

```
import cv2
import numpy as np
kernel = np.ones((5,5),np.uint8)
img = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ")
cv2.imshow("original image",img)
cv2.waitKey(0)
img = cv2.resize(img,(600,600))
cv2.imshow("image",img)
cv2.waitKey(0)
```

OUTPUT:

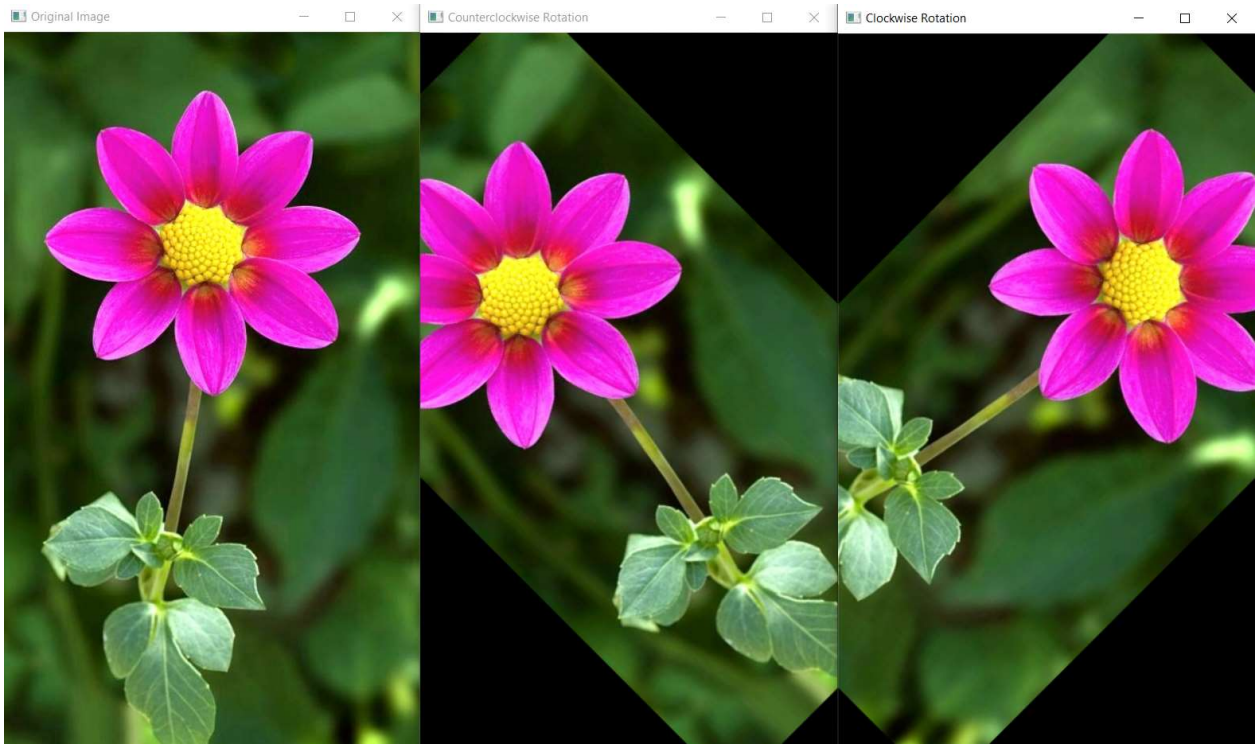


9. Perform Rotation of an image to clockwise and counter clockwise direction:

PROGRAM:

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg”)

height, width = image.shape[:2]

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

angle = 45

clockwise_rotation_matrix = cv2.getRotationMatrix2D(center, -angle, 1.0)

rotated_clockwise = cv2.warpAffine(image, clockwise_rotation_matrix, (width, height))

counterclockwise_rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)

rotated_counterclockwise = cv2.warpAffine(image, counterclockwise_rotation_matrix, (width,
height))

cv2.imshow('Original Image', image)

cv2.imshow('Clockwise Rotation', rotated_clockwise)

cv2.imshow('Counterclockwise Rotation', rotated_counterclockwise)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Output:



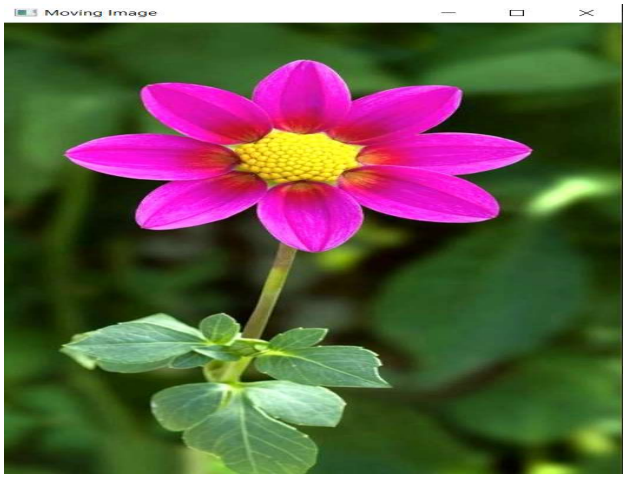
10. Perform moving of an image from one place to another.

Program:

```
import cv2
import numpy as np
image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/draw village.jpg")
x = 100
y = 100
dx = 50
dy = 30
while True:
    image_copy = image.copy()
    x += dx
    y += dy
    cv2.imshow('Moving Image', image_copy)
cv2.waitKey()
```

cv2.destroyAllWindows()

output:

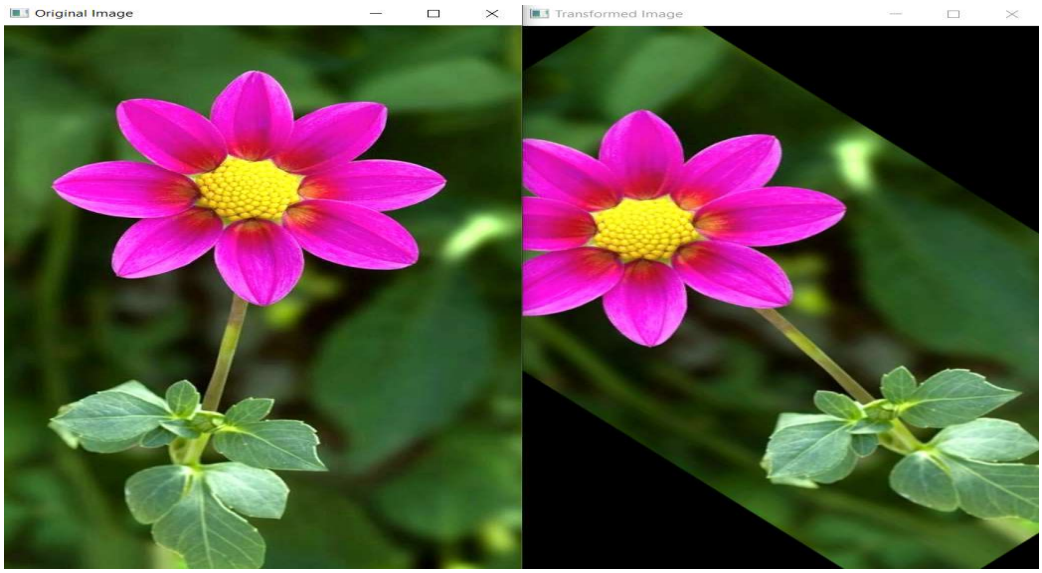


11. Perform Affine Transformation on the image:

Program:

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ")

angle = 45

scale = 1.0

rotation_matrix = cv2.getRotationMatrix2D((image.shape[1] / 2, image.shape[0] / 2), angle, scale)


output_image = cv2.warpAffine(image, rotation_matrix, (image.shape[1], image.shape[0]))


cv2.imshow('Original Image', image)

cv2.imshow('Transformed Image', output_image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

output:

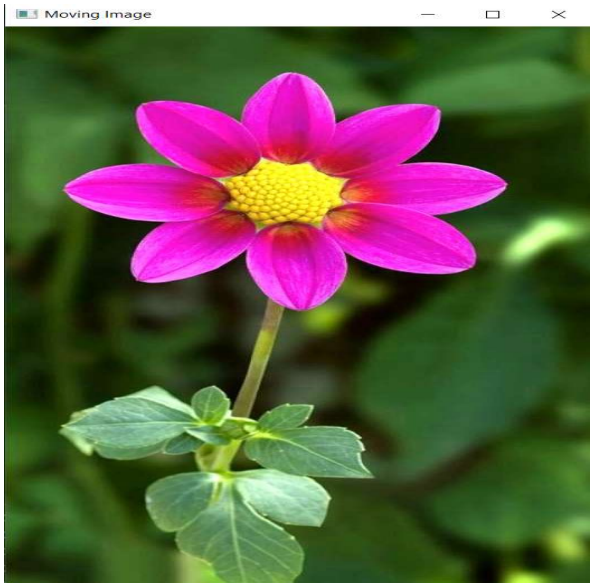


12. Perform Perspective Transformation on the image.

Program:

```
import cv2
import numpy as np
image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ")
x = 100
y = 100
dx = 50
dy = 30
while True:
    image_copy = image.copy()
    x += dx
    y += dy
    cv2.imshow('Moving Image', image_copy)
cv2.waitKey()
cv2.destroyAllWindows()
```

output:



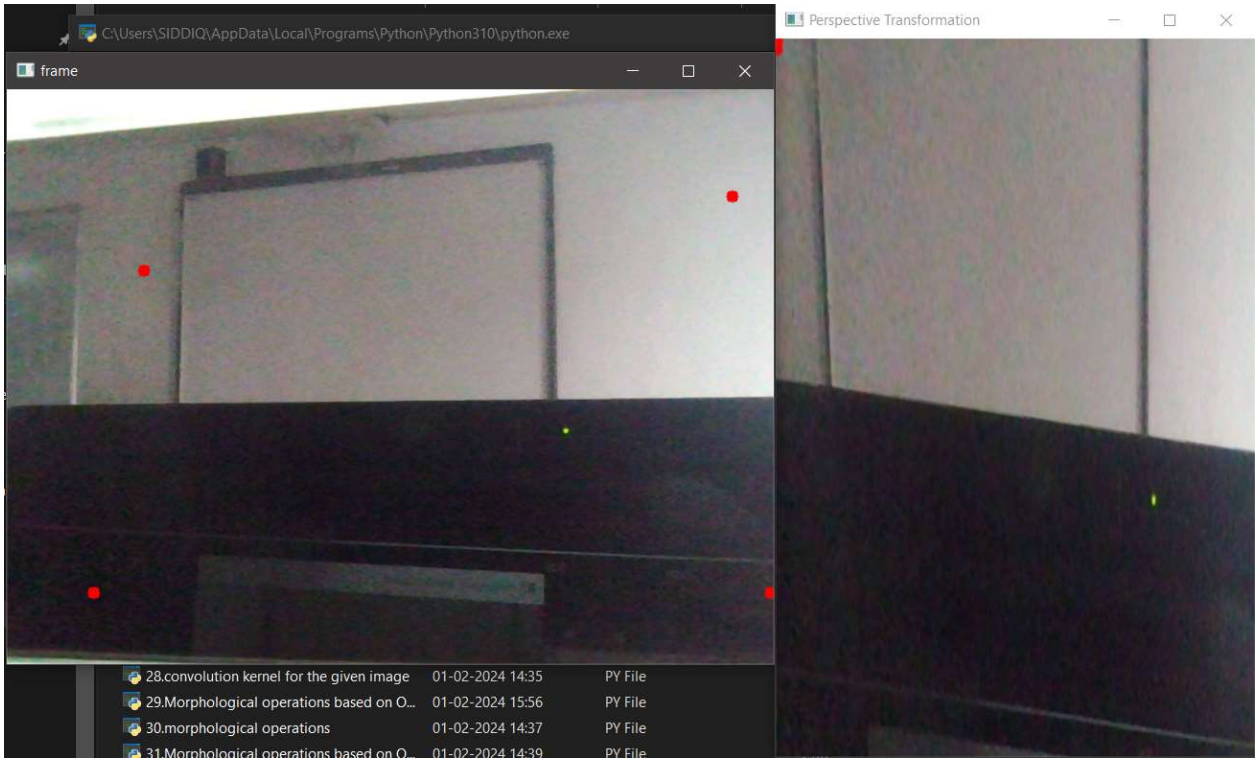
13. Perform Perspective Transformation on the Video.

Program:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
cap = cv2.VideoCapture(0)
while True:
    _,frame = cap.read()
    cv2.circle(frame,(114,151),5,(0,0,255),-1)
    cv2.circle(frame, (605, 89), 5, (0, 0, 255), -1)
    cv2.circle(frame, (72, 420), 5, (0, 0, 255), -1)
    cv2.circle(frame, (637, 420), 5, (0, 0, 255), -1)
```

```
    imgPts = np.float32([[114,151],[605, 89],[72, 420],[637, 420]])

    objPoints = np.float32([[0,0],[420,0],[0,637],[420,637]])
    matrix = cv2.getPerspectiveTransform(imgPts,objPoints)
    result = cv2.warpPerspective(frame,matrix,(400,600))
    cv2.imshow('frame',frame)
    cv2.imshow('Perspective Transformation', result)
    key = cv2.waitKey(1)
    plt.show()
    if key == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

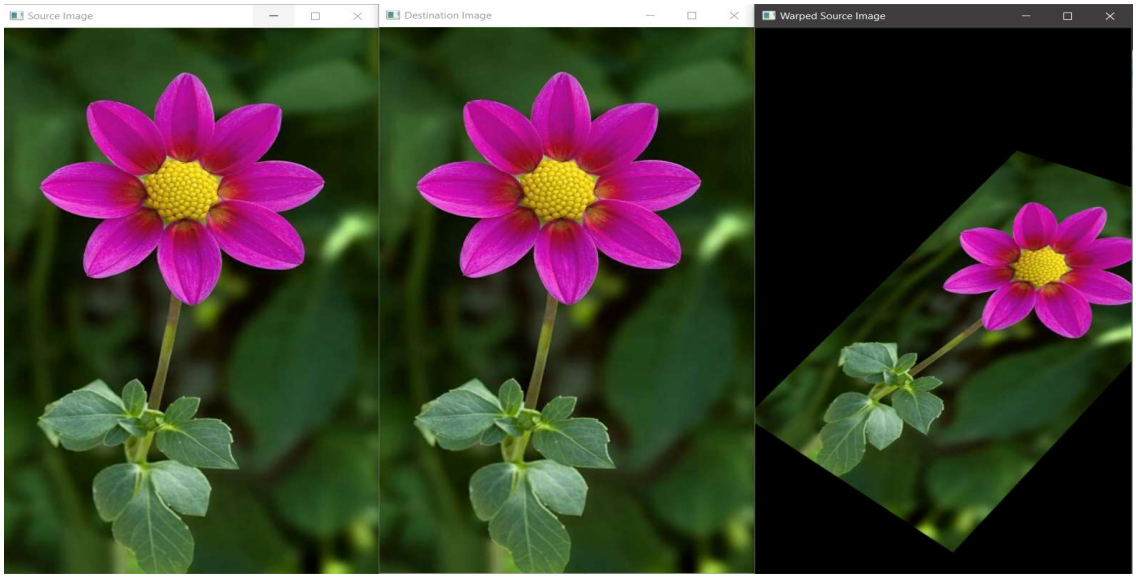
output:



14. Perform transformation using Homography matrix.

```
import cv2
import numpy as np
if __name__ == '__main__' :
    im_src = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg
    pts_src = np.array([[141, 131], [480, 159], [493, 630],[64, 601]])
    im_dst = cv2.imread()
    pts_dst = np.array([[318, 256],[534, 372],[316, 670],[73, 473]])
    h, status = cv2.findHomography(pts_src, pts_dst)
    im_out = cv2.warpPerspective(im_src, h, (im_dst.shape[1],im_dst.shape[0]))
    cv2.imshow("Source Image", im_src)
    cv2.imshow("Destination Image", im_dst)
    cv2.imshow("Warped Source Image", im_out)
    cv2.waitKey(0)
```

output:

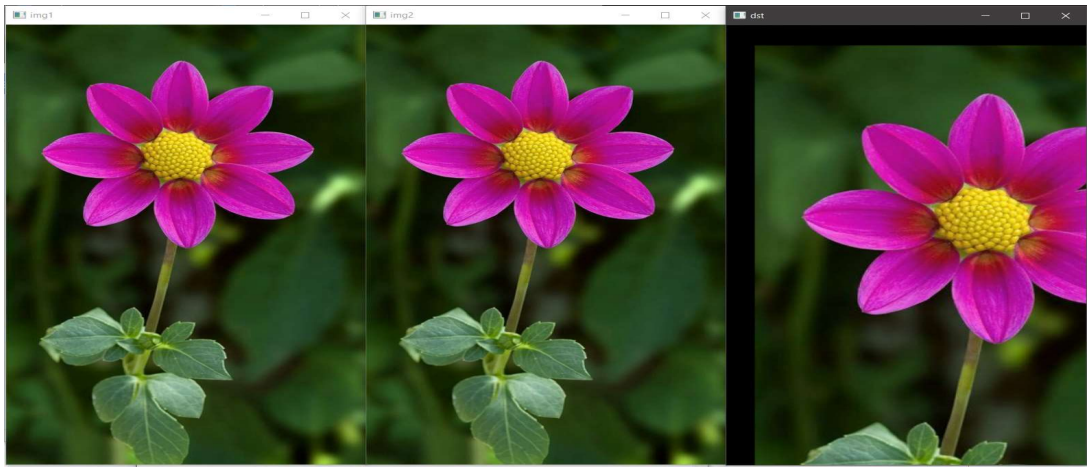


15. Perform transformation using Direct Linear Transformation.

Program:

```
import cv2

import numpy as np

img1 = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ") img2 = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/11.jpg “)# Define corresponding points

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

pts2 = np.array([[100, 100], [300, 100], [100, 300], [300, 300]])

dst = cv2.warpPerspective(img1, H, (img2.shape[1], img2.shape[0])) # Display images

cv2.imshow('img1', img1) cv2.imshow('img2', img2) cv2.imshow('dst', dst)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

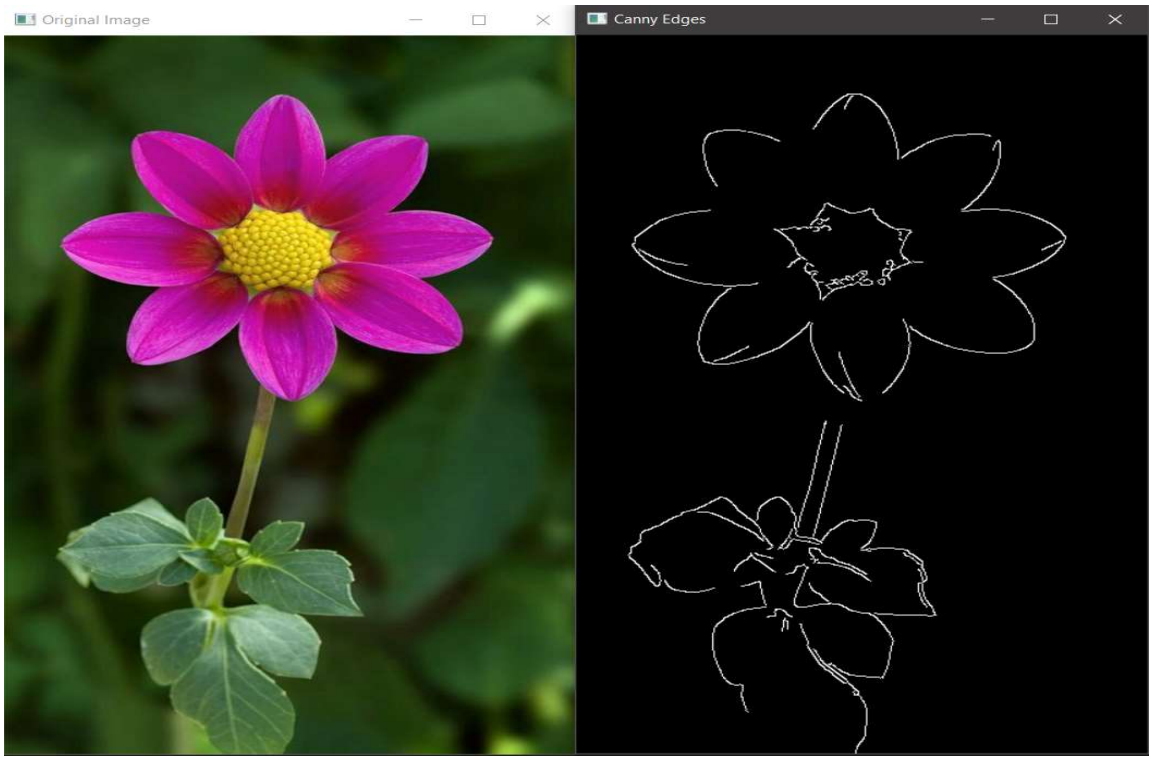
output:



16. Perform Edge detection using canny method:

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ")

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

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

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

cv2.imshow('Original Image', image)

cv2.imshow('Canny Edges', edges)

cv2.imwrite('edges_detected.jpg', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Output:

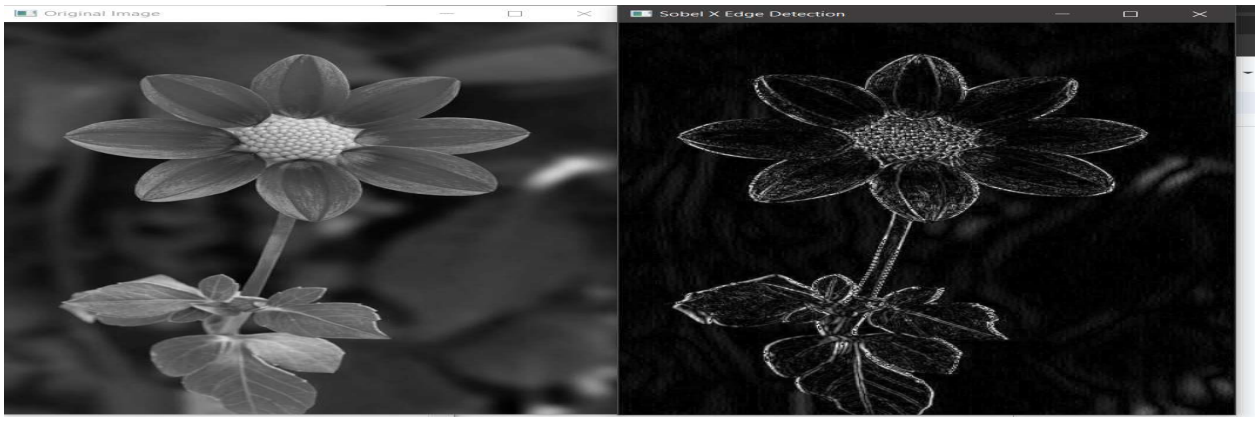


17. Perform Edge detection using Sobel Matrix along X axis:

Program:

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ",
cv2.IMREAD_GRAYSCALE)

sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)

sobel_x = np.absolute(sobel_x)

sobel_x = np.uint8(sobel_x)

cv2.imshow('Original Image', image)

cv2.imshow('Sobel X Edge Detection', sobel_x)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

output:

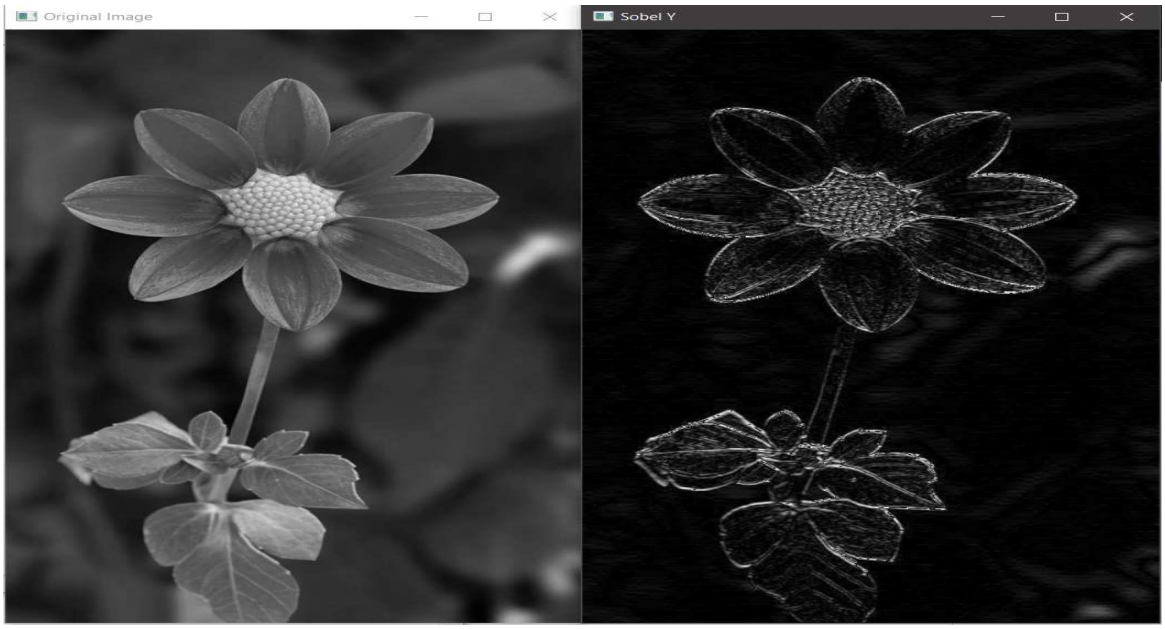


18. Perform Edge detection using Sobel Matrix along Y axis:

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg
cv2.IMREAD_GRAYSCALE)
```

```
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)

sobel_y = np.abs(sobel_y)

sobel_y = np.uint8(sobel_y)

cv2.imshow('Original Image', image)

cv2.imshow('Sobel Y', sobel_y)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

output:



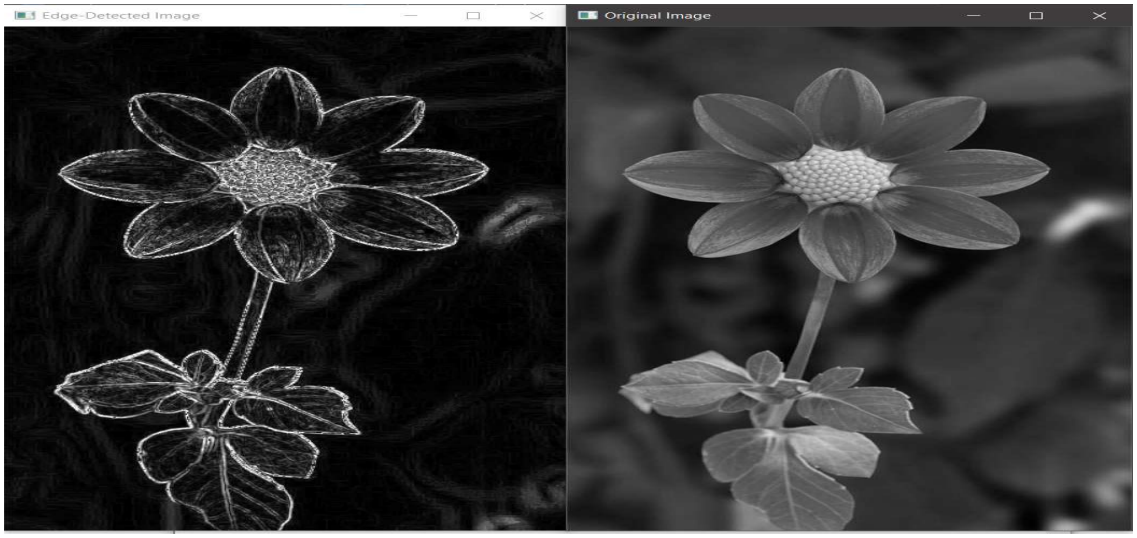
19. Perform Edge detection using Sobel Matrix along XY axis:

Program:

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg
“,cv2.IMREAD_GRAYSCALE)

sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)

abs_sobel_x = np.abs(sobel_x)
```

```
sobel_x = np.uint8(abs_sobel_x)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
abs_sobel_y = np.abs(sobel_y)
sobel_y = np.uint8(abs_sobel_y)
edge_image = cv2.bitwise_or(sobel_x, sobel_y)
cv2.imshow('Original Image', image)
cv2.imshow('Edge-Detected Image', edge_image)
cv2.imwrite('edge_detected_image.jpg', edge_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:



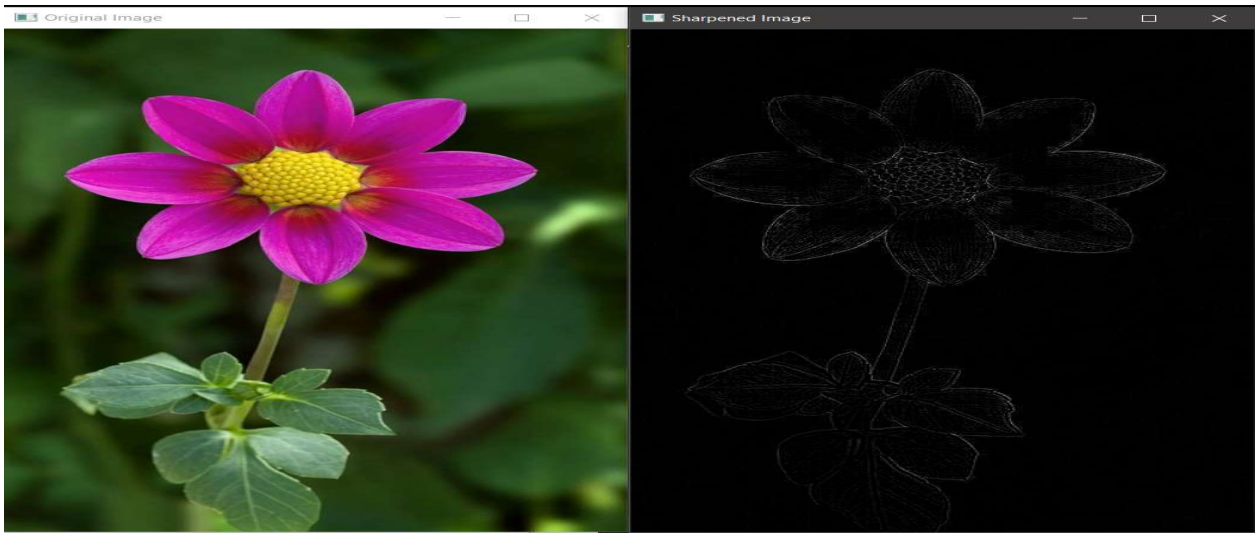
20. Perform Sharpening of Image using Laplacian mask with negative center coefficient.

PROGRAM:

```
import cv2
import numpy as np
image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg”)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
laplacian_kernel = np.array([[0, 1, 0],  [1, -4, 1],[0, 1, 0]], dtype=np.float32)
```

```
sharpened_image = cv2.filter2D(gray_image, -1, laplacian_kernel)
sharpened_image = cv2.cvtColor(sharpened_image, cv2.COLOR_GRAY2BGR)
cv2.imshow('Original Image', image)
cv2.imshow('Sharpened Image', sharpened_image)
cv2.imwrite('sharpened_image.jpg', sharpened_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:



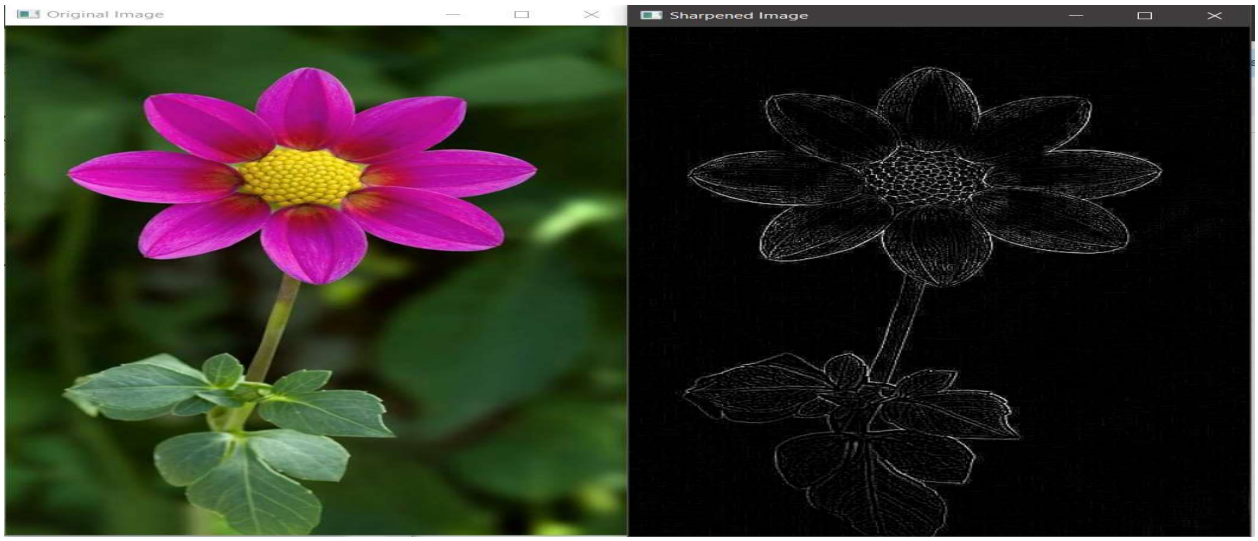
21. Perform Sharpening of Image using Laplacian mask with an extension of diagonals:

Program:

```
import cv2
import numpy as np
image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg")
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
laplacian_kernel = np.array([[1, 1, 1],[1, -8, 1],[1, 1, 1]], dtype=np.float32)
sharpened_image = cv2.filter2D(gray_image, -1, laplacian_kernel)
```

```
sharpened_image = cv2.cvtColor(sharpened_image, cv2.COLOR_GRAY2BGR)
cv2.imshow('Original Image', image)
cv2.imshow('Sharpened Image', sharpened_image)
cv2.imwrite('sharpened_image.jpg', sharpened_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:



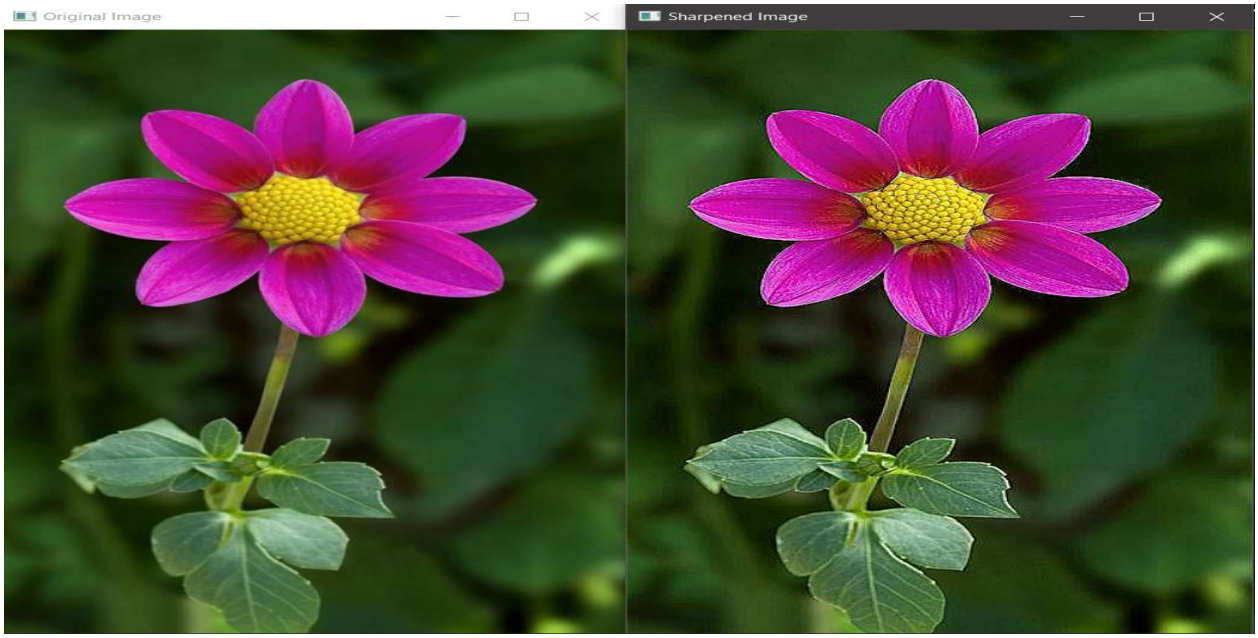
22. Perform Sharpening of Image using Laplacian mask with positive center coefficient:

Program:

```
import cv2
import numpy as np
image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ")
kernel = np.array([[0, -1, 0],
                   [-1, 5, -1],
                   [0, -1, 0]], dtype=np.float32)
```

```
sharpened_image = cv2.filter2D(image, -1, kernel)
cv2.imshow('Original Image', image)
cv2.imshow('Sharpened Image', sharpened_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

output:



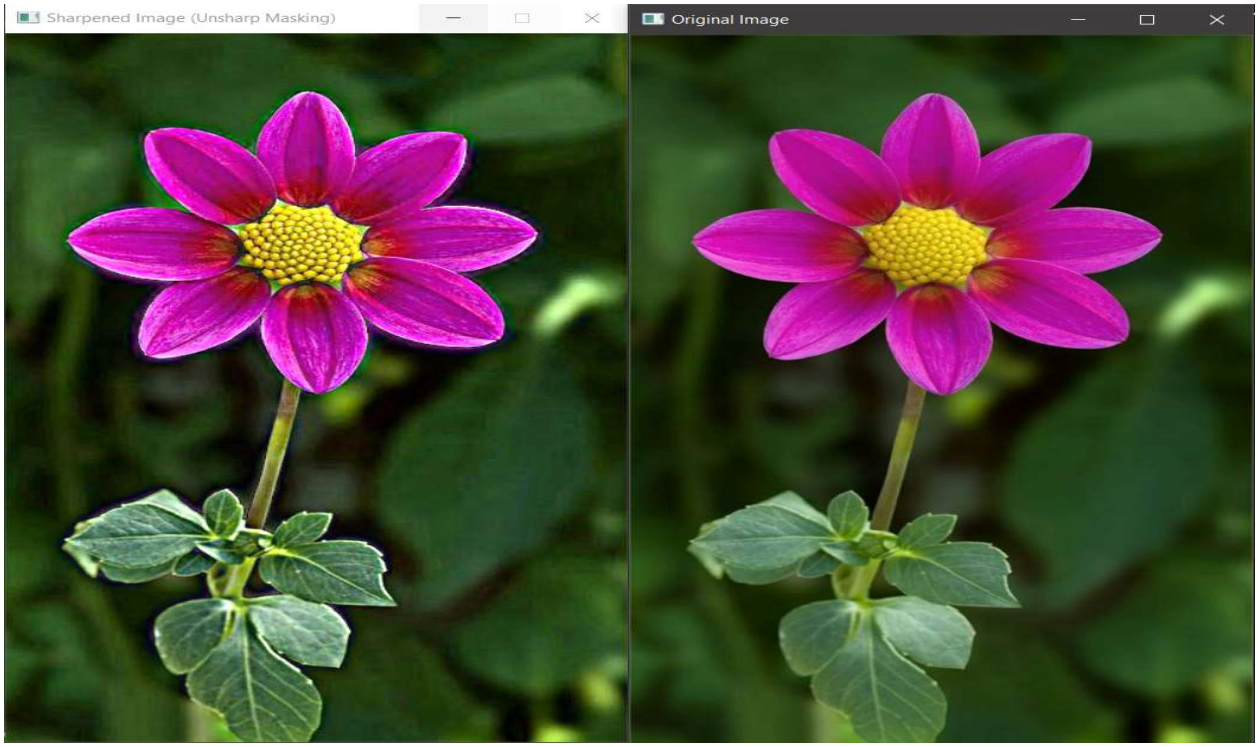
23. Perform Sharpening of Image using unsharp masking:

Program:

```
import cv2
import numpy as np
image = cv2.imread("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ")
image_float = np.float32(image)

blur = cv2.GaussianBlur(image_float, (0, 0), sigmaX=5)
unsharp_mask = cv2.addWeighted(image_float, 2.5, blur, -1.5, 0)
```

```
sharpened_image = cv2.convertScaleAbs(unsharp_mask)

cv2.imshow('Original Image', image)

cv2.imshow('Sharpened Image (Unsharp Masking)', sharpened_image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

output:



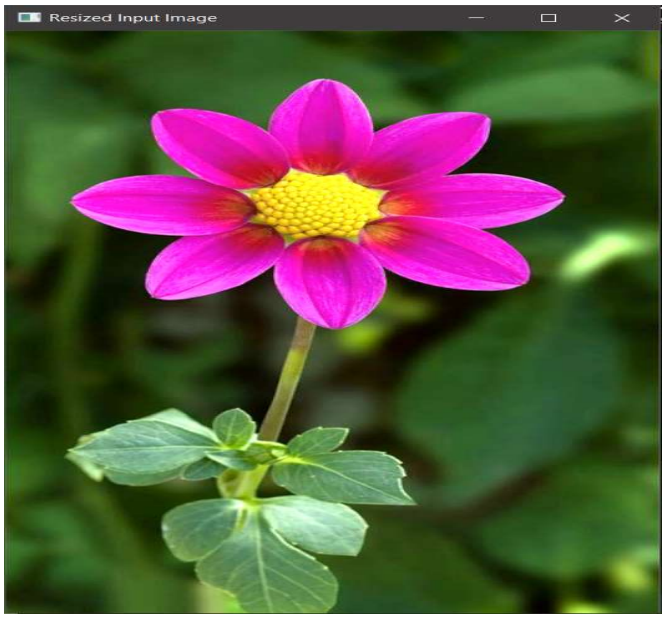
24. Perform Sharpening of Image using High-Boost Masks.

Program:

```
import cv2
resized_img = cv2.imread("C:/Users/theja/OneDrive/Desktop/computervision/th.jpg")
resized_wm = cv2.imread("C:/Users/theja/OneDrive/Desktop/computervision/11.jpg")

h_img, w_img, _ = resized_img.shape
center_y = int(h_img/2)
center_x = int(w_img/2)
h_wm, w_wm, _ = resized_wm.shape
top_y = center_y - int(h_wm/2)
left_x = center_x - int(w_wm/2)
```

```
bottom_y = top_y + h_wm
right_x = left_x + w_wm
roi = resized_img[top_y:bottom_y, left_x:right_x]
resized_wm = cv2.resize(resized_wm, (roi.shape[1], roi.shape[0]))
result = cv2.addWeighted(roi, 1, resized_wm, 0.3, 0)
resized_img[top_y:bottom_y, left_x:right_x] = result
filename = "C:/Users/theja/OneDrive/Desktop/computervision/11.jpg"
cv2.imwrite(filename, resized_img)
cv2.imshow("Resized Input Image", resized_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

output:



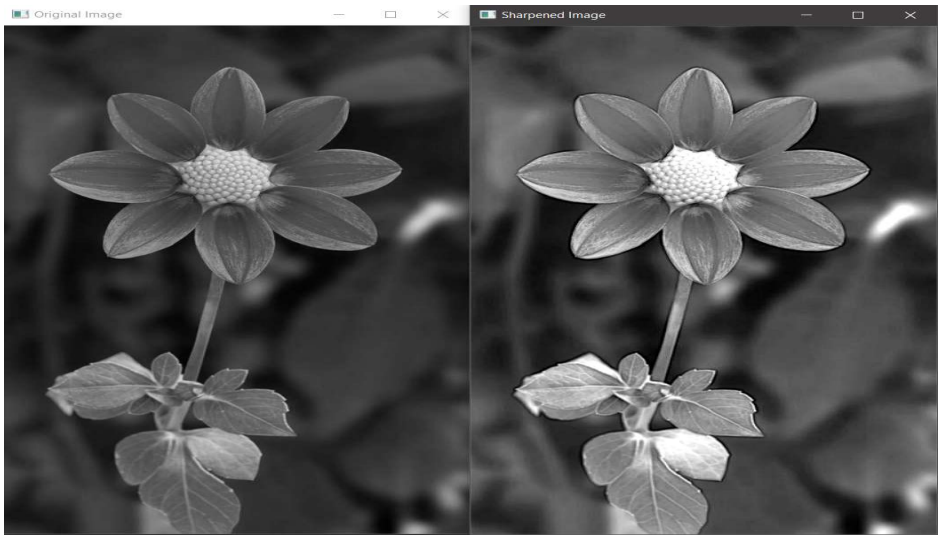
25. Perform Sharpening of Image using Gradient masking:

Program:

```
import cv2

import numpy as np

image = cv2.imread ("C:/Users/theja/onedrive/Desktop/computervision/th.jpg ")

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

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

grad_x = cv2.Scharr(blurred, cv2.CV_64F, 1, 0)

grad_y = cv2.Scharr(blurred, cv2.CV_64F, 0, 1)
```

```
gradient_magnitude = np.sqrt(grad_x**2 + grad_y**2)
gradient_magnitude = cv2.normalize(gradient_magnitude, None, 0, 255, cv2.NORM_MINMAX)
gradient_magnitude = np.uint8(gradient_magnitude)
sharpened = cv2.addWeighted(gray, 1.5, gradient_magnitude, -0.5, 0)
cv2.imshow('Original Image', gray)
cv2.imshow('Sharpened Image', sharpened)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

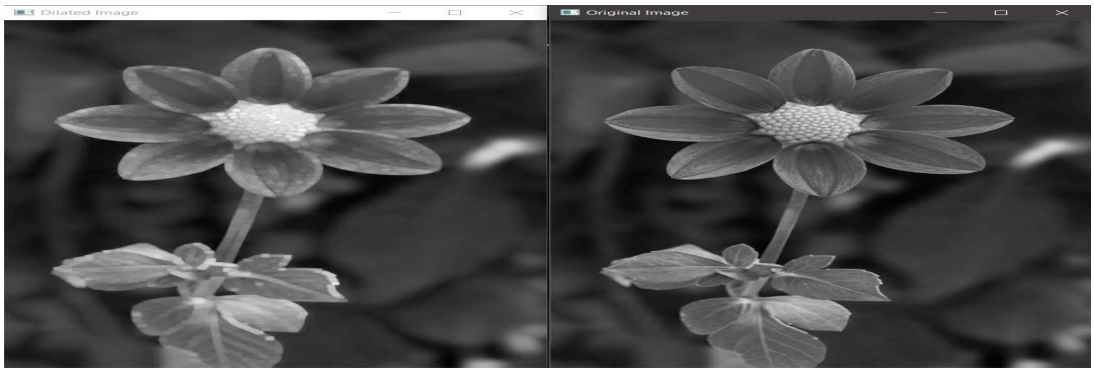


26. Insert water marking to the image using OpenCV.

```
import cv2
logo = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/logo.jpeg")
img = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/cofeee.jpeg")
h_logo, w_logo, _ = logo.shape
h_img, w_img, _ = img.shape
center_y = int(h_img/2)
center_x = int(w_img/2)
top_y = center_y - int(h_logo/2)
left_x = center_x - int(w_logo/2)
```

```
bottom_y = top_y + h_logo

right_x = left_x + w_logo

destination = img[top_y:bottom_y, left_x:right_x]

result = cv2.addWeighted(destination, 1, logo, 0.5, 0)

img[top_y:bottom_y, left_x:right_x] = result

cv2.imwrite("watermarked.jpg", img)

cv2.imshow("Watermarked Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:



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

```
import cv2

main_image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/draw village.jpg")

if main_image is not None:

    main_height, main_width, _ = main_image.shape

    crop_height, crop_width = 80,288

    cropped_region = main_image[0:crop_height, 0:crop_width]

    paste_image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/cofeee.jpeg")

    if paste_image is not None:
```

```
        paste_height, paste_width, _ = paste_image.shape
        paste_x, paste_y = main_width - paste_width, main_height - paste_height
        main_image[paste_y:main_height, paste_x:main_width] = cropped_region
        cv2.imshow("Result", main_image)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
    else:
   print("Error: Could not load paste image.")
else:
    print("Error: Could not load main image.")
```

OUTPUT:

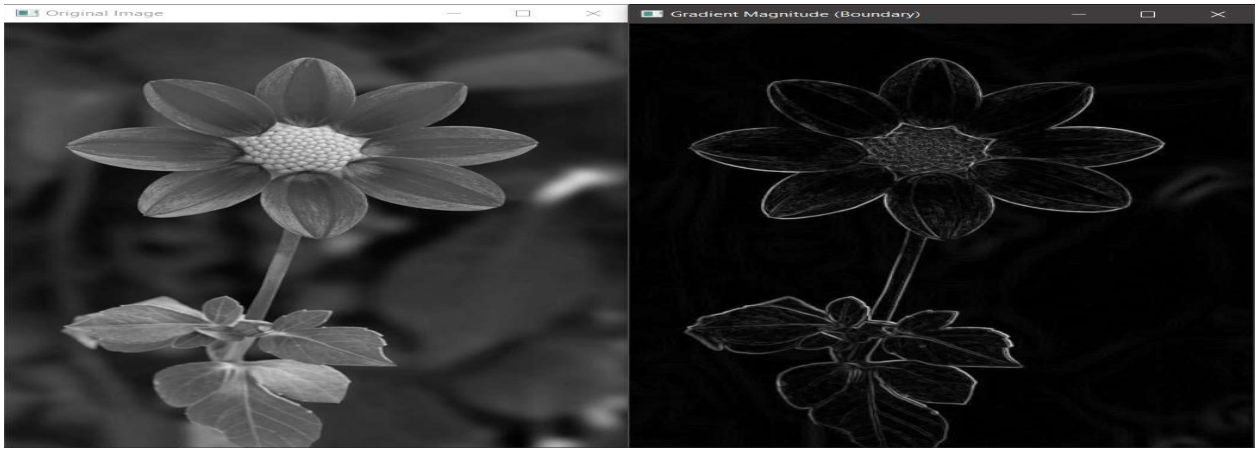
28. Find the boundary of the image using Convolution kernel for the given image.

```
import cv2
import numpy as np
image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/cofeee.jpeg", cv2.IMREAD_GRAYSCALE)
if image is not None:
    sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
    sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
    gradient_magnitude = np.sqrt(sobel_x**2 + sobel_y**2)
    gradient_magnitude = cv2.normalize(gradient_magnitude, None, 0, 255, cv2.NORM_MINMAX)
    gradient_magnitude = np.uint8(gradient_magnitude)
    cv2.imshow("Original Image", image)
    cv2.imshow("Gradient Magnitude (Boundary)", gradient_magnitude)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

else:

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

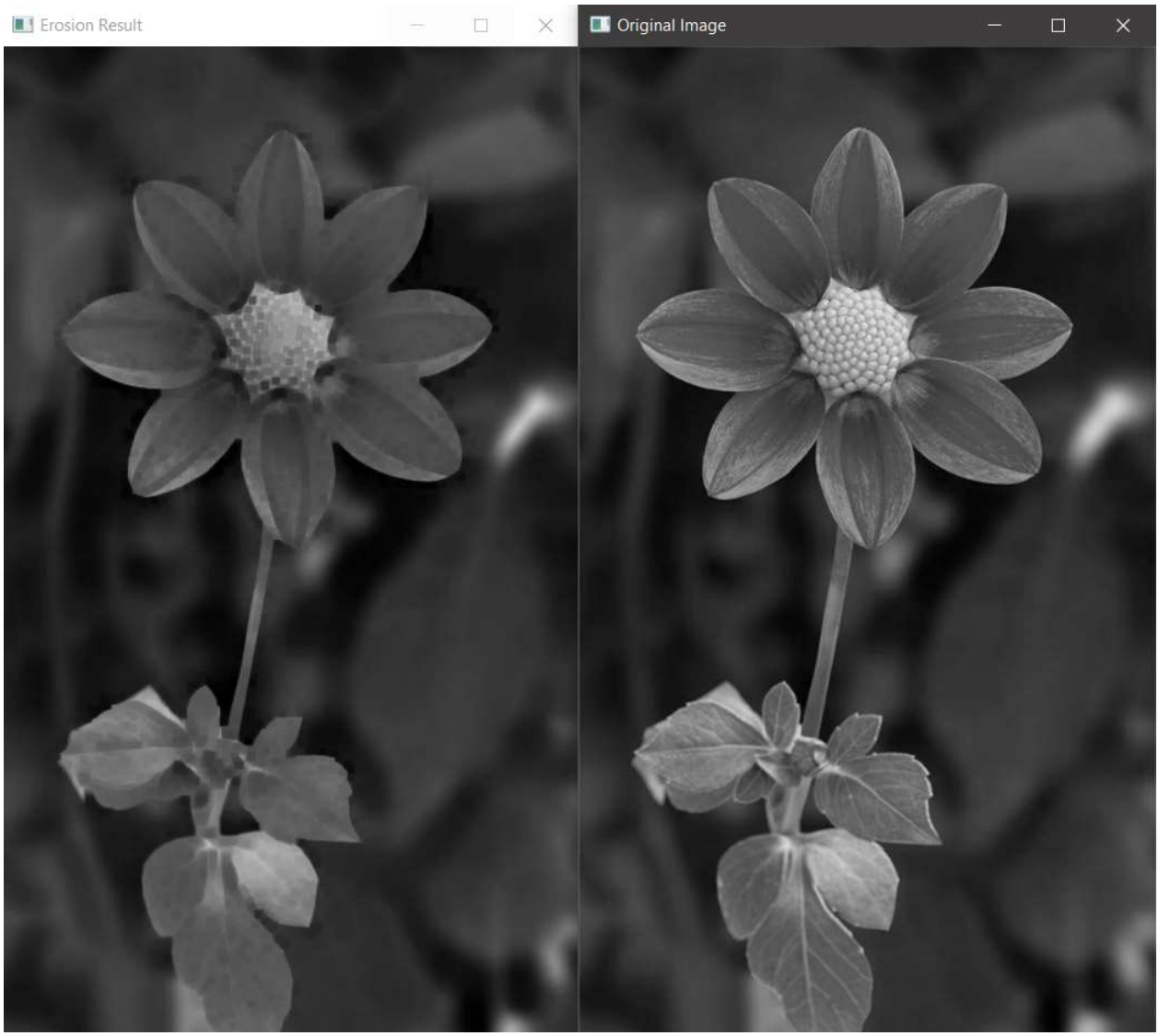
OUTPUT:



29. Morphological operations based on OpenCV using Erosion technique.

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/cofeee.jpeg", cv2.IMREAD_GRAYSCALE)

if image is not None:

    kernel = np.ones((5, 5), np.uint8)

    erosion_result = cv2.erode(image, kernel, iterations=1)

    cv2.imshow("Original Image", image)

    cv2.imshow("Erosion Result", erosion_result)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

else:

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

OUTPUT:

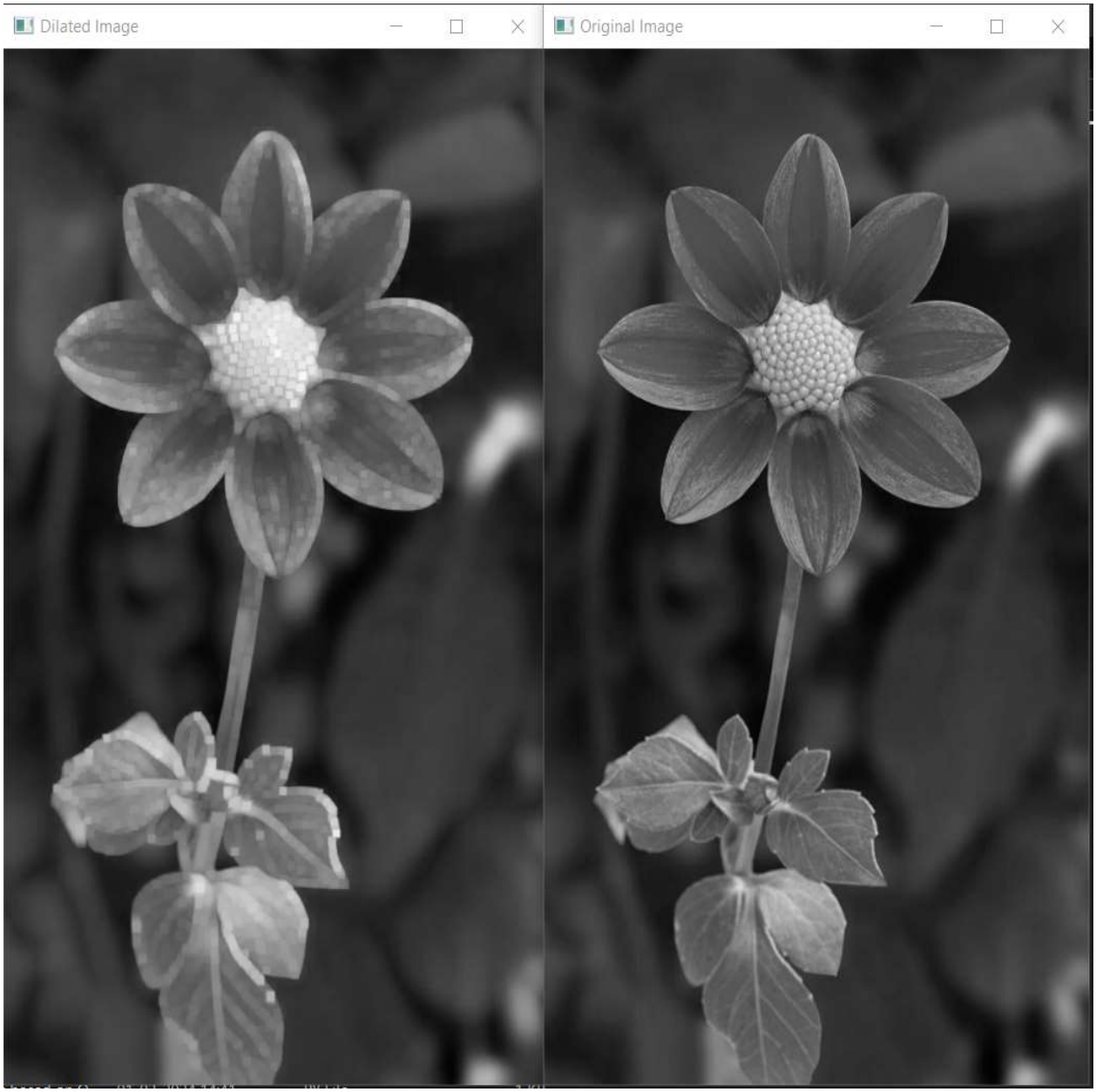


30. Morphological operations based on OpenCV using Dilation technique.

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/cofeee.jpeg",
cv2.IMREAD_GRAYSCALE)

if image is not None:

    kernel = np.ones((5, 5), np.uint8)

    dilated_image = cv2.dilate(image, kernel, iterations=1)
```

```
    cv2.imshow("Original Image", image)
    cv2.imshow("Dilated Image", dilated_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
else:
    print("Error: Could not load the image.")
```

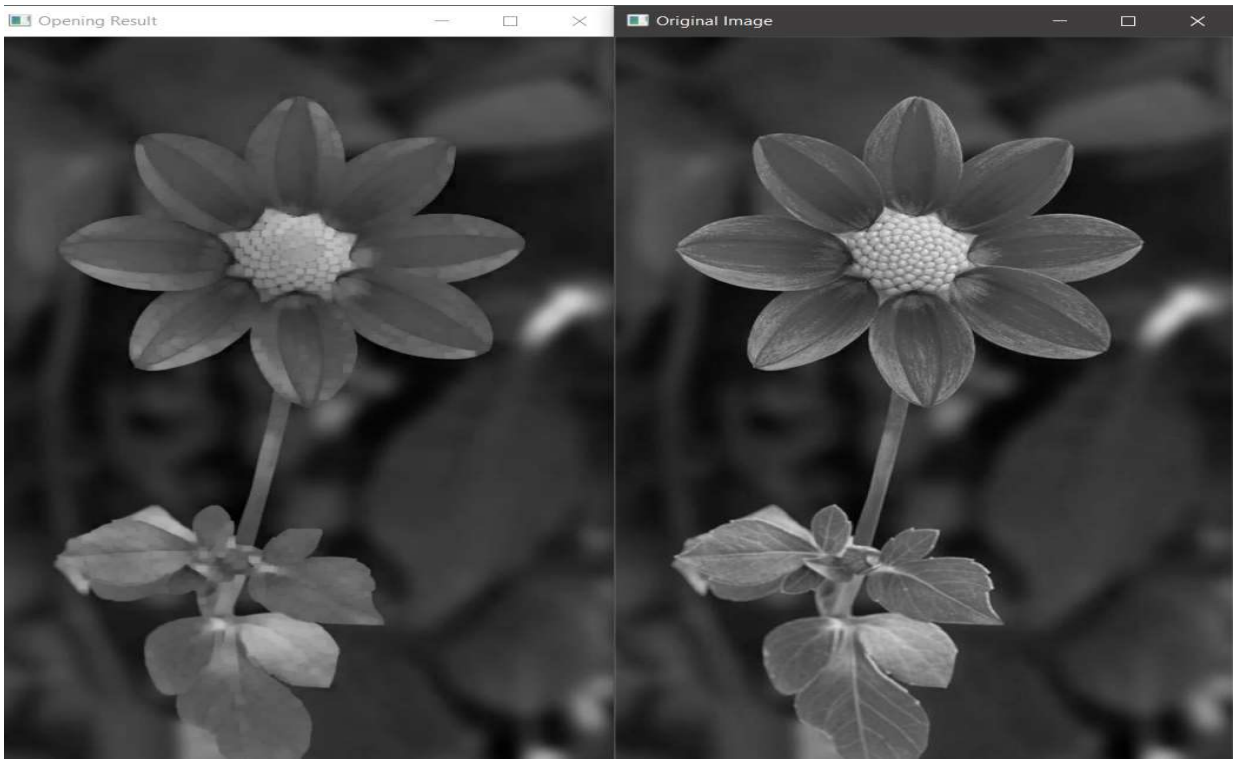
OUTPUT:



31. Morphological operations based on OpenCV using Opening technique.

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/cofeee.jpeg",
cv2.IMREAD_GRAYSCALE)

if image is not None:

    kernel = np.ones((5, 5), np.uint8)

    opening_result = cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)

    cv2.imshow("Original Image", image)

    cv2.imshow("Opening Result", opening_result)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

else:

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

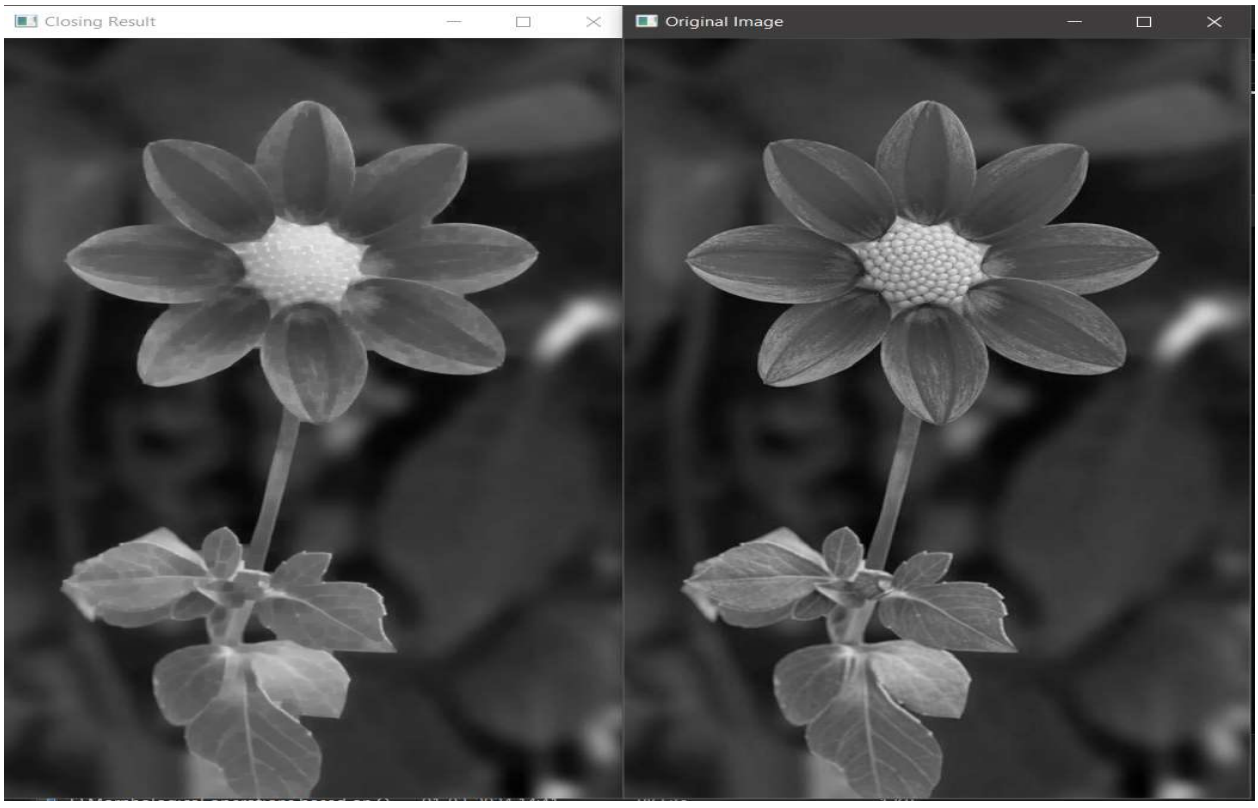
OUTPUT:



32. Morphological operations based on OpenCV using Closing technique.

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/cofeee.jpeg",
cv2.IMREAD_GRAYSCALE)

if image is not None:

    kernel = np.ones((5, 5), np.uint8)

    closing_result = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)

    cv2.imshow("Original Image", image)

    cv2.imshow("Closing Result", closing_result)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

else:

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

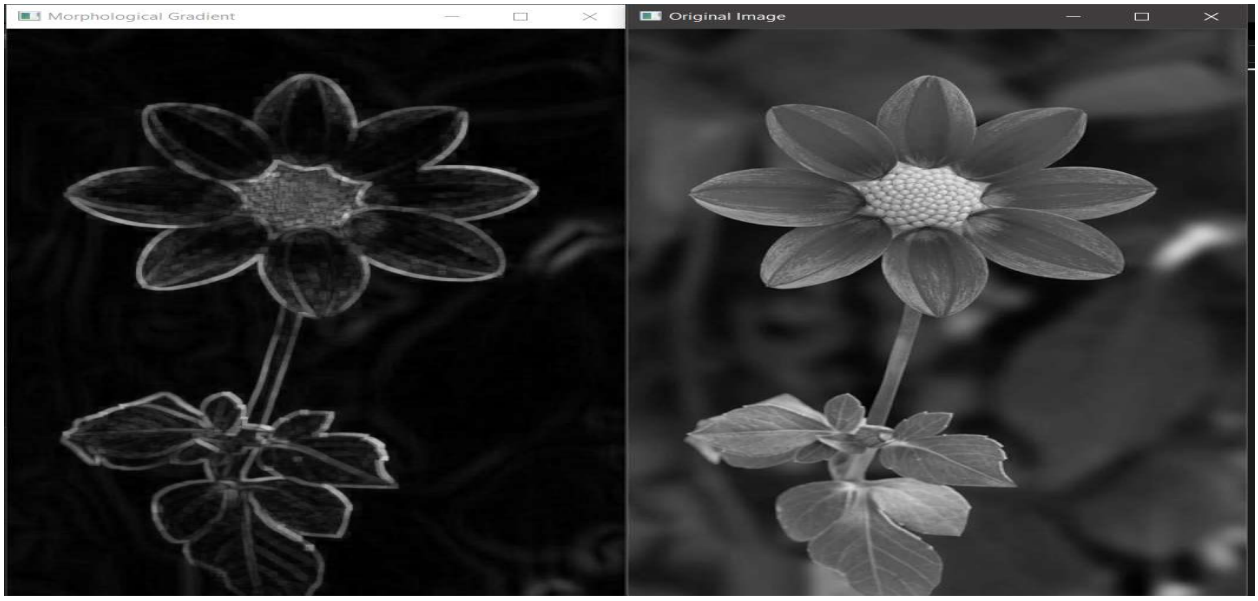
OUTPUT:



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

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/cofeee.jpeg", cv2.IMREAD_GRAYSCALE)

if image is not None:

    kernel = np.ones((5, 5), np.uint8)

    gradient = cv2.morphologyEx(image, cv2.MORPH_GRADIENT, kernel)

    cv2.imshow("Original Image", image)

    cv2.imshow("Morphological Gradient", gradient)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

else:

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

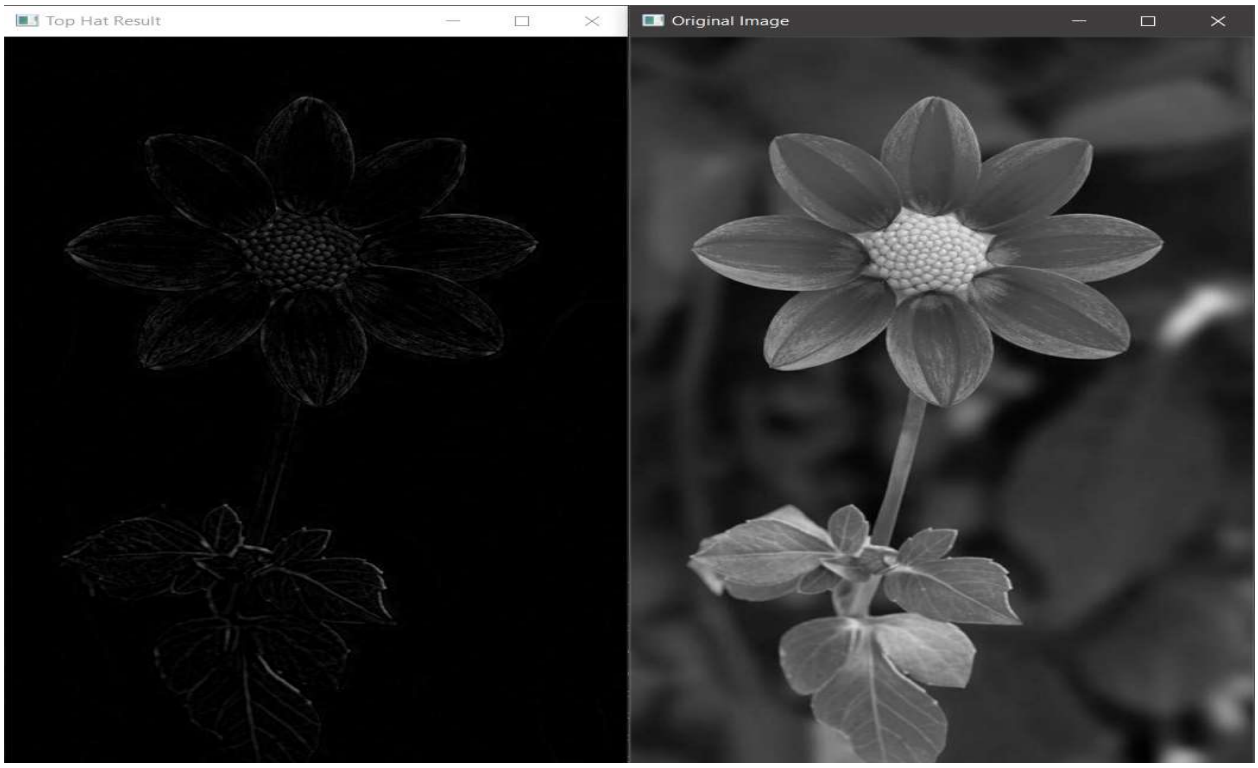
OUTPUT:



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

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/cofeee.jpeg",
cv2.IMREAD_GRAYSCALE)

if image is not None:

    kernel = np.ones((5, 5), np.uint8)

    top_hat = cv2.morphologyEx(image, cv2.MORPH_TOPHAT, kernel)

    cv2.imshow("Original Image", image)

    cv2.imshow("Top Hat Result", top_hat)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

else:

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

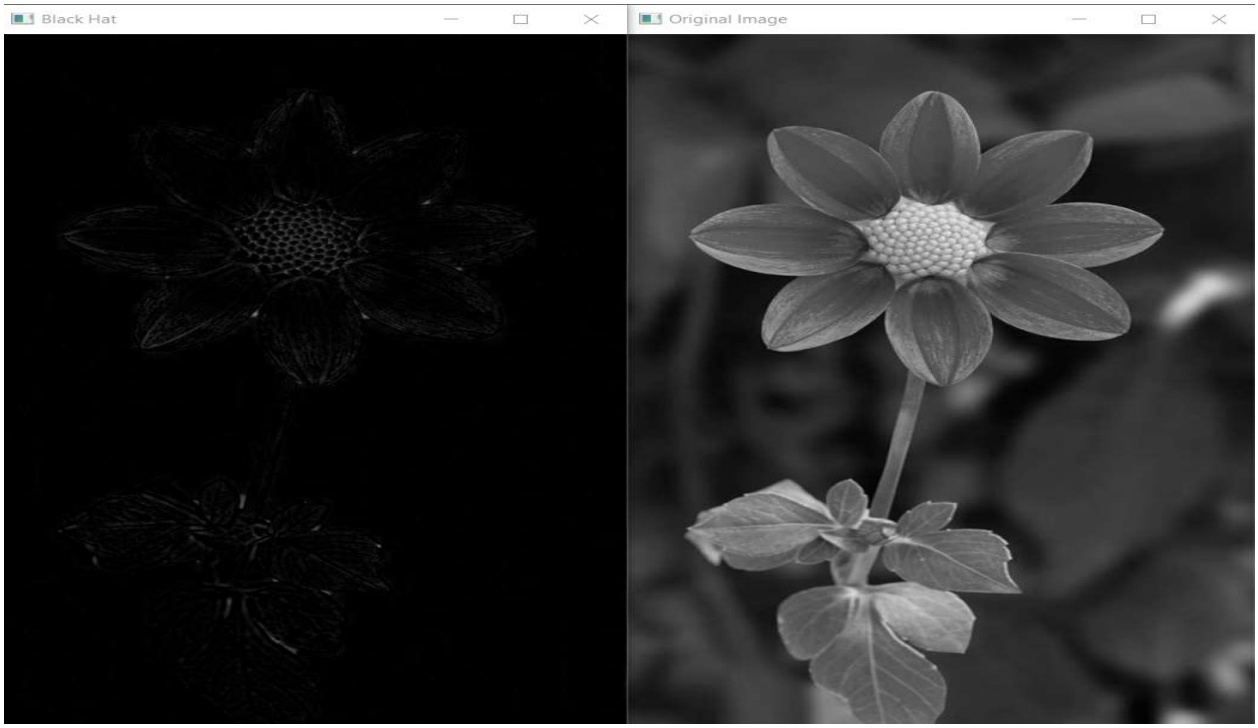
OUTPUT:



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

```
import cv2

import numpy as np

image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/cofeee.jpeg",
cv2.IMREAD_GRAYSCALE)

if image is not None:

    kernel = np.ones((5, 5), np.uint8)

    black_hat = cv2.morphologyEx(image, cv2.MORPH_BLACKHAT, kernel)

    cv2.imshow("Original Image", image)

    cv2.imshow("Black Hat", black_hat)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

else:

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

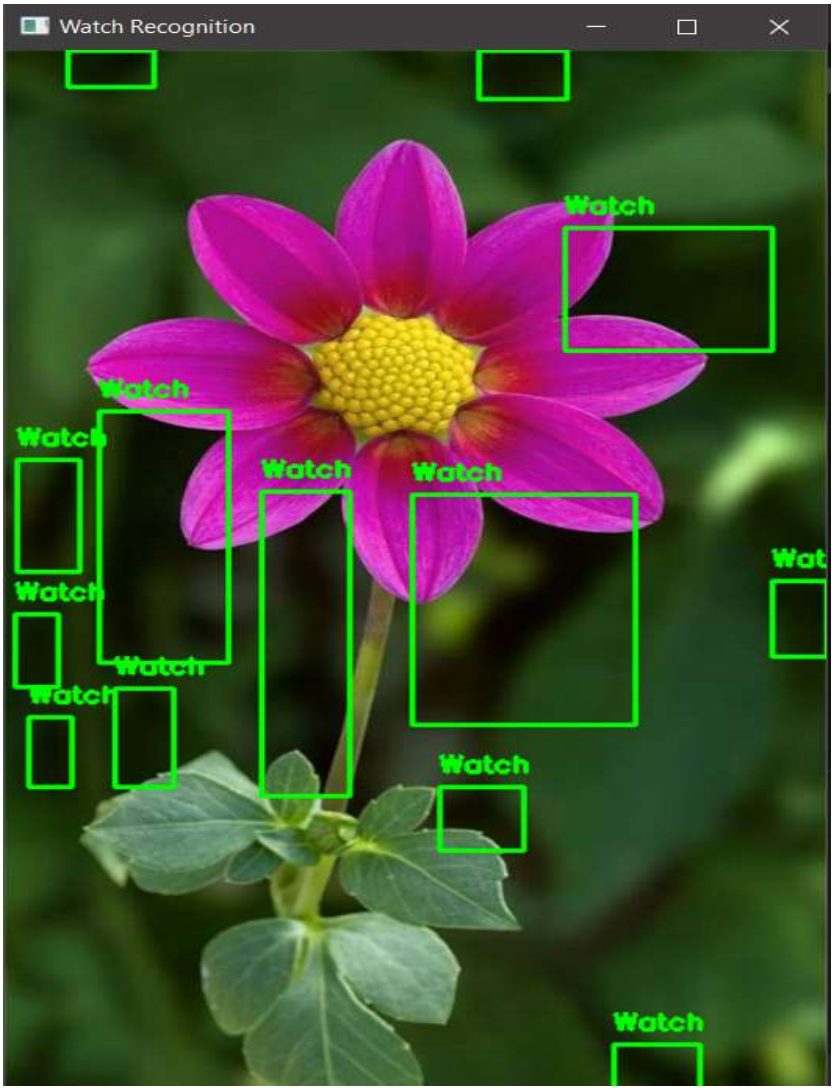
OUTPUT:



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

```
import cv2
import numpy as np
image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/Picture1.jpg")
if image is not None:
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_bound = np.array([0, 0, 0])
    upper_bound = np.array([180, 255, 30])
    mask = cv2.inRange(hsv, lower_bound, upper_bound)
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    for contour in contours:
        area = cv2.contourArea(contour)
        if area > 500:
            x, y, w, h = cv2.boundingRect(contour)
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
          cv2.putText(image, "Watch", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2
    cv2.imshow("Watch Recognition", image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
else:
    print("Error: Could not load the image.")
```

OUTPUT:

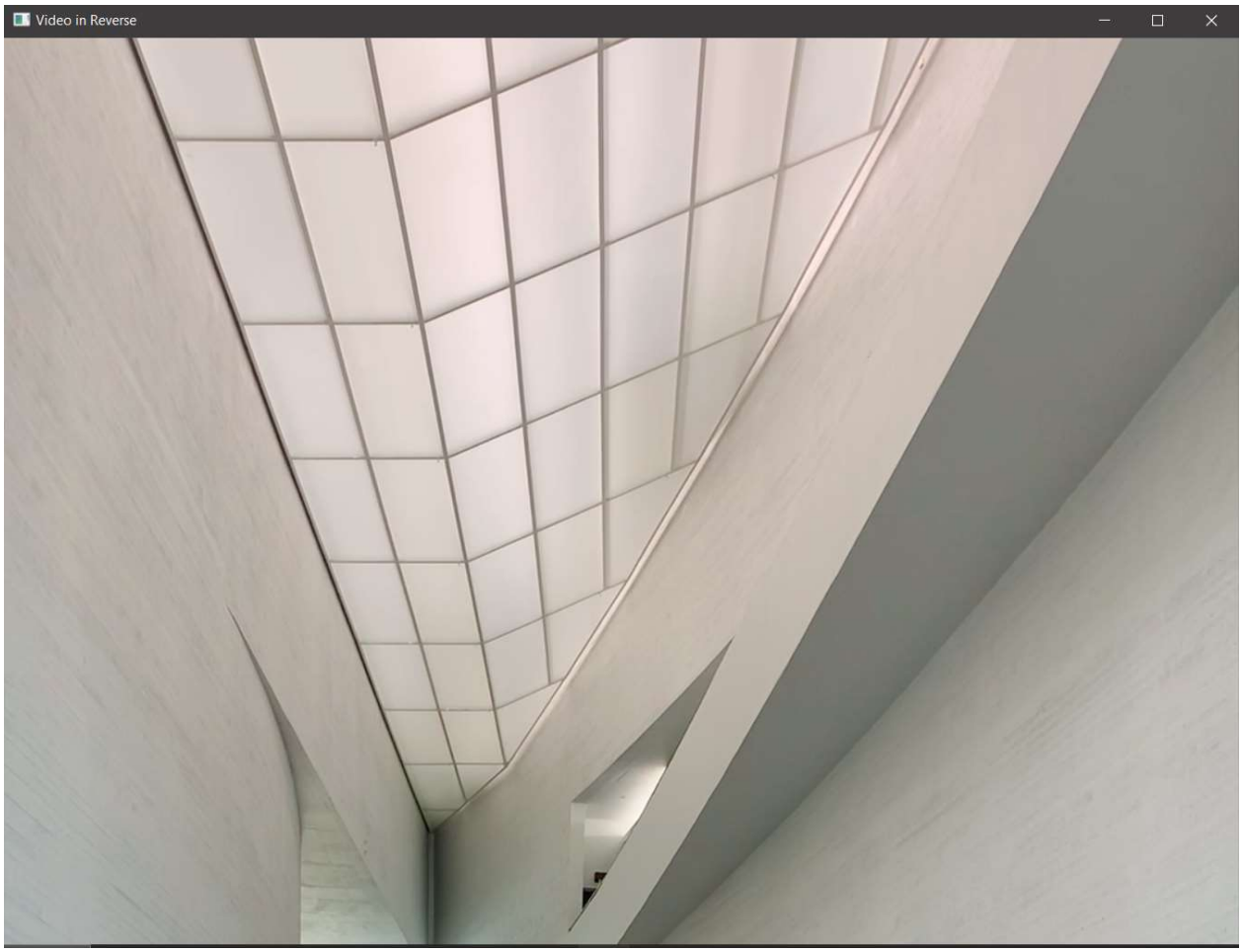


37. Using Opencv play Video in Reverse mode.

```
import cv2
video_path = "C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/mp4.mp4"
cap = cv2.VideoCapture(video_path)
if not cap.isOpened():
    print("Error: Could not open the video file.")
    exit()
frames = []
while True:
```

```
    ret, frame = cap.read()
    if not ret:
        break
    frames.append(frame)
cap.release()
for frame in reversed(frames):
    cv2.imshow("Video in Reverse", frame)
    if cv2.waitKey(30) & 0xFF == 27:
        break
cv2.destroyAllWindows()
```

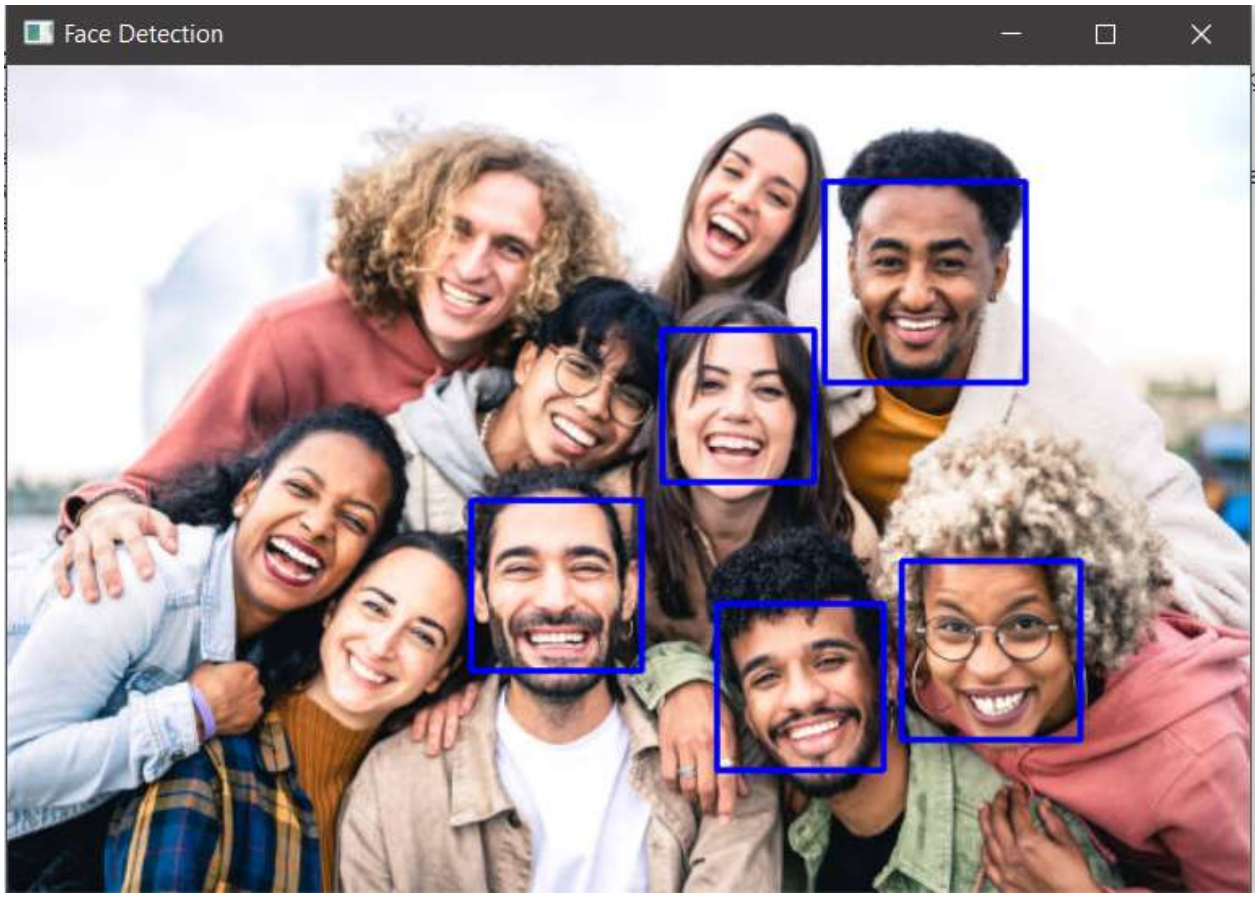
OUTPUT:



38. Face Detection using Opencv.

```
import cv2

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

img = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/face.jpg")

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)

for (x, y, w, h) in faces:

    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

cv2.imshow('Face Detection', img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:



39. Vehicle Detection in a Video frame using OpenCV .

```
import cv2
cap = cv2.VideoCapture("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/car.webm")
fgbg = cv2.createBackgroundSubtractorMOG2()
while True:
    ret, frame = cap.read()
    if not ret:
        break
    fgmask = fgbg.apply(frame)
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel=None)
    contours, _ = cv2.findContours(fgmask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    for contour in contours:
        if cv2.contourArea(contour) > 500:
            x, y, w, h = cv2.boundingRect(contour)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    cv2.imshow('Vehicle Detection', frame)

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

OUTPUT:

40. Draw Rectangular shape and extract objects

```
import cv2
import numpy as np
image = cv2.imread("C:/Users/susri/OneDrive/Desktop/COMPUTER VISION/Virat.jpg")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
    extracted_object = image[y:y + h, x:x + w]
    cv2.imshow('Extracted Object', extracted_object)
    cv2.waitKey(0)
cv2.imshow('Objects with Rectangles', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT: