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)

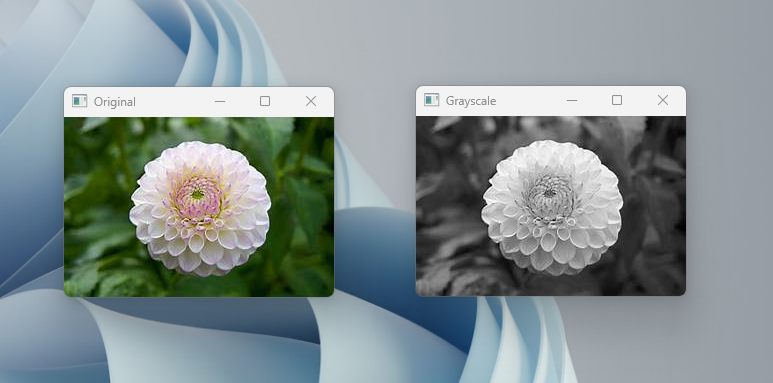
gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

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

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

1. **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)

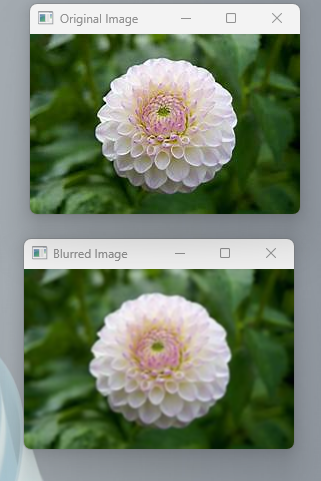
cv2.imshow('Original Image', image)

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

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

1. **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)

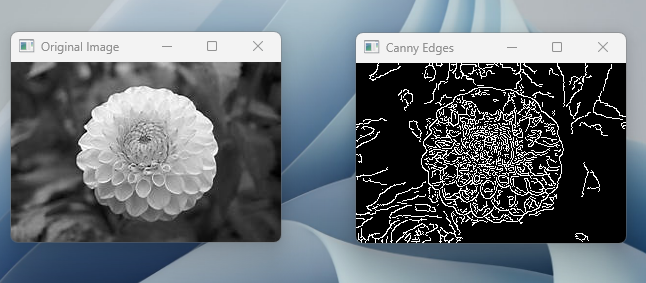
cv2.imshow('Original Image', image)

cv2.imshow('Canny Edges', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

1. **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)

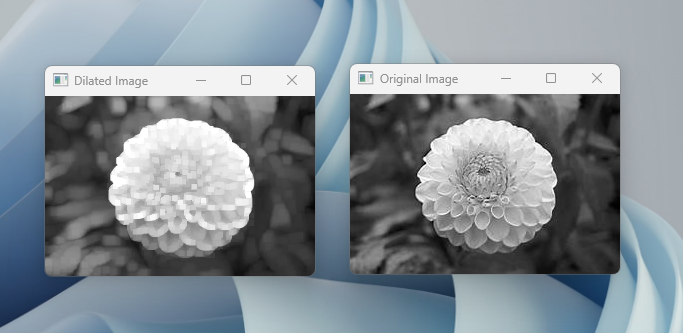
cv2.imshow('Original Image', image)

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

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

1. **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)

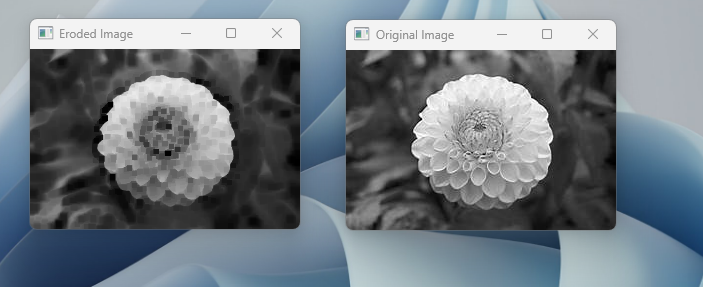
cv2.imshow('Original Image', image)

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

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

1. **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:**

****

**"C:\Users\theja\OneDrive\Desktop\Video.png"**

1. **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:**

1. **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)

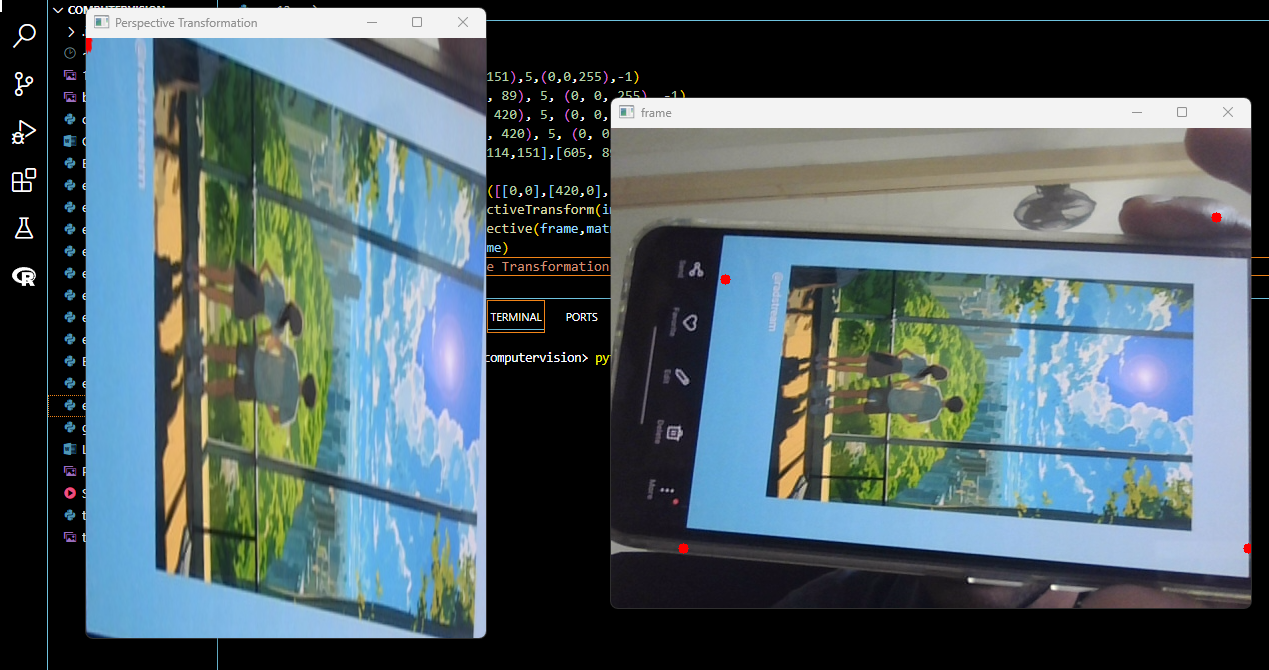
cv2.waitKey(0)

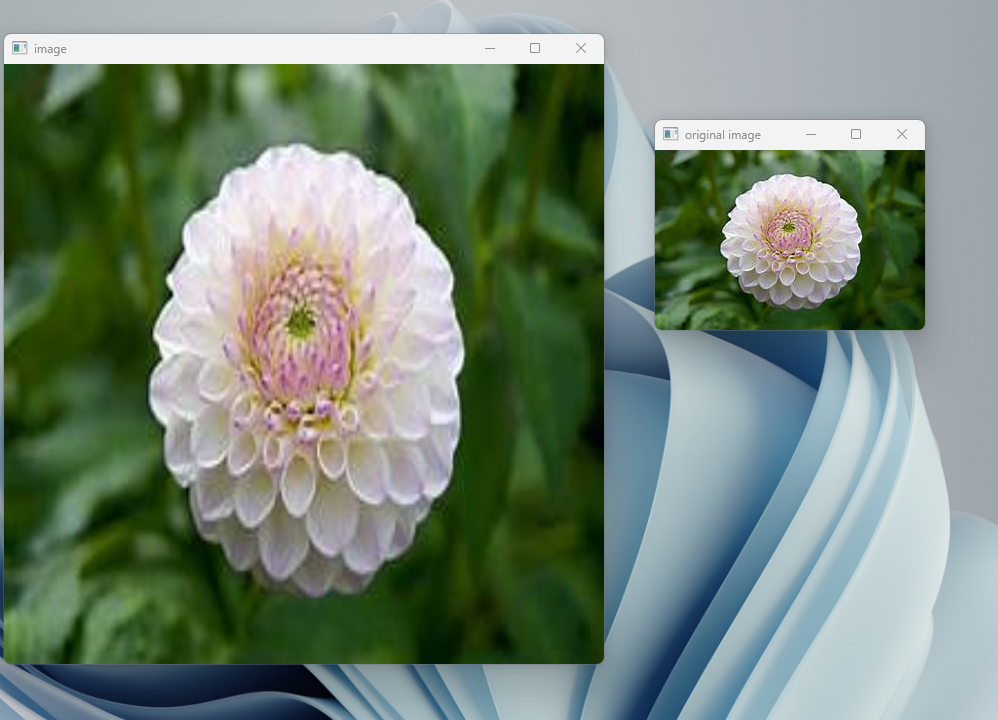
img = cv2.resize(img,(600,600))

cv2.imshow("image",img)

cv2.waitKey(0)

**OUTPUT:**

****

****

1. **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))

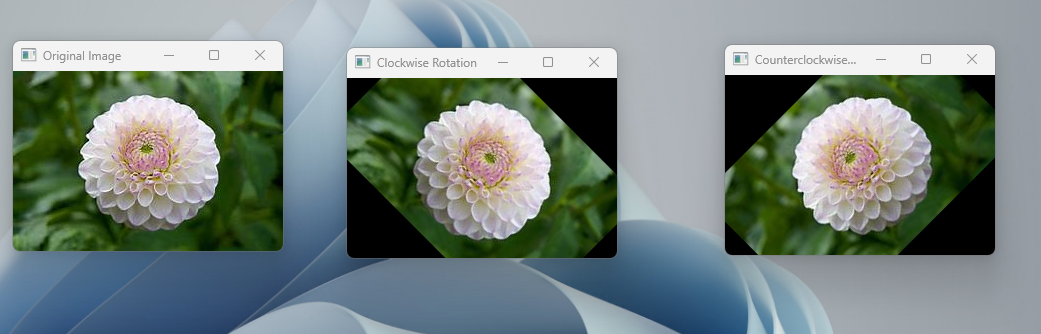
cv2.imshow('Original Image', image)

cv2.imshow('Clockwise Rotation', rotated\_clockwise)

cv2.imshow('Counterclockwise Rotation', rotated\_counterclockwise)

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

1. **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:**

****

1. **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]))

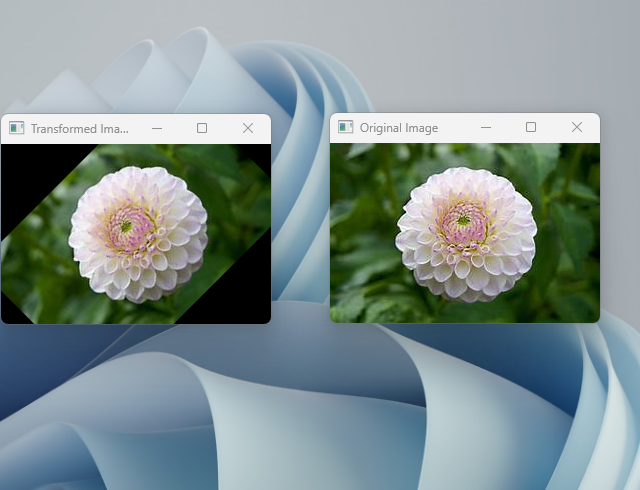
cv2.imshow('Original Image', image)

cv2.imshow('Transformed Image', output\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****

1. **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:**

****

1. **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()

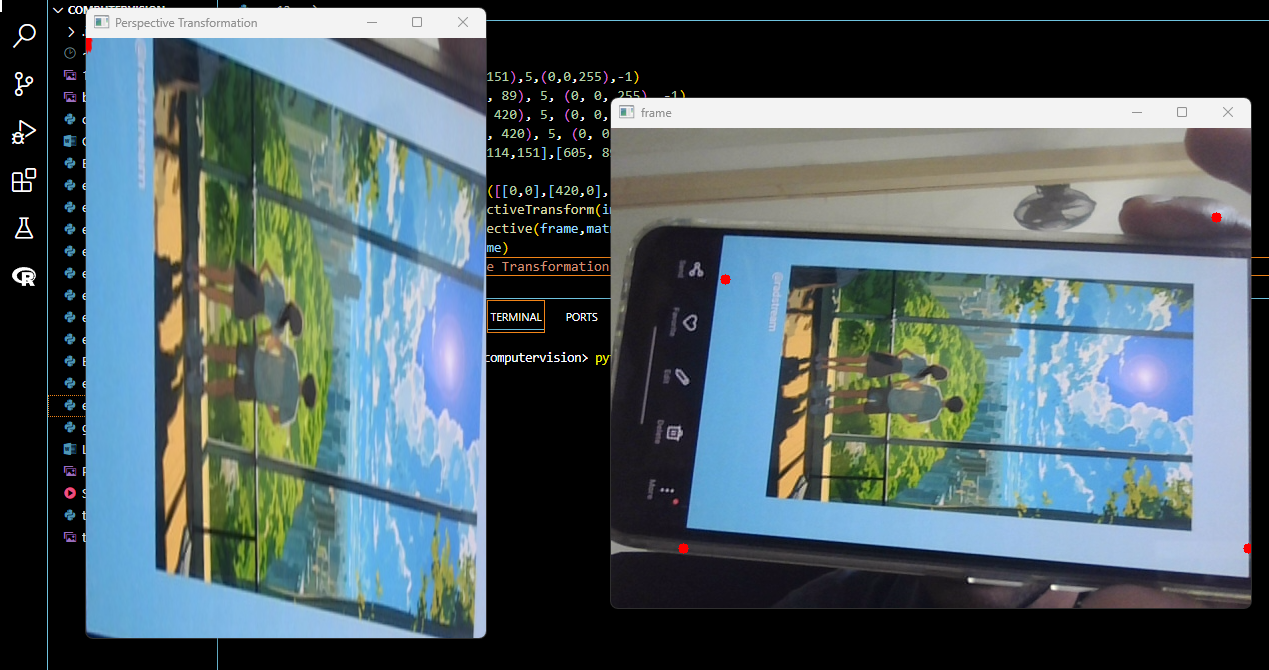
if key == 27:

break

cap.release()

cv2.destroyAllWindows()

**output:**

****

1. 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]))

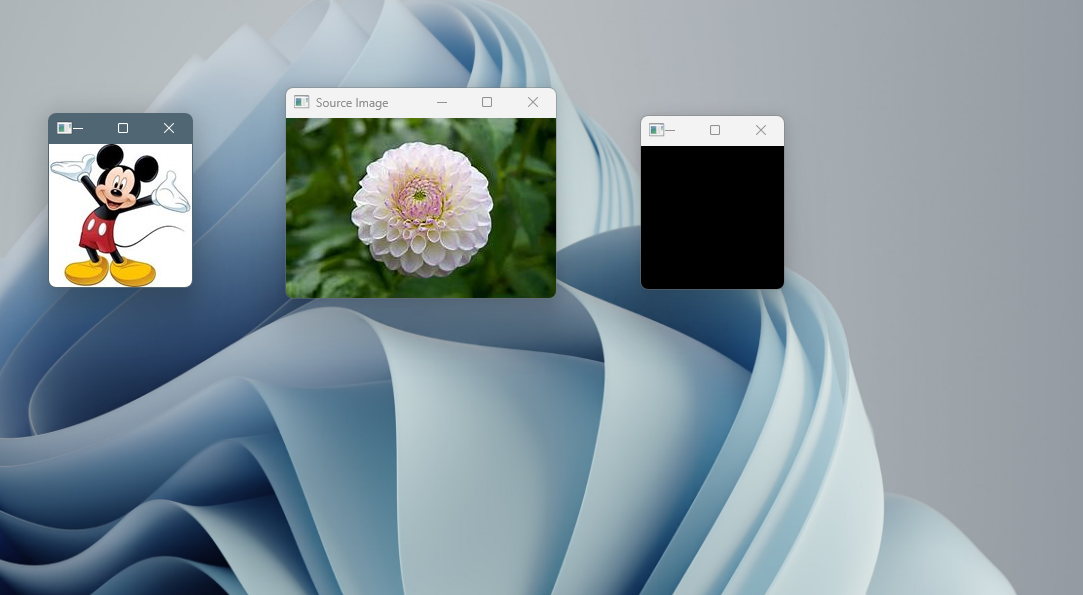
cv2.imshow("Source Image", im\_src)

cv2.imshow("Destination Image", im\_dst)

cv2.imshow("Warped Source Image", im\_out)

cv2.waitKey(0)

**output:**

****

1. **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]])

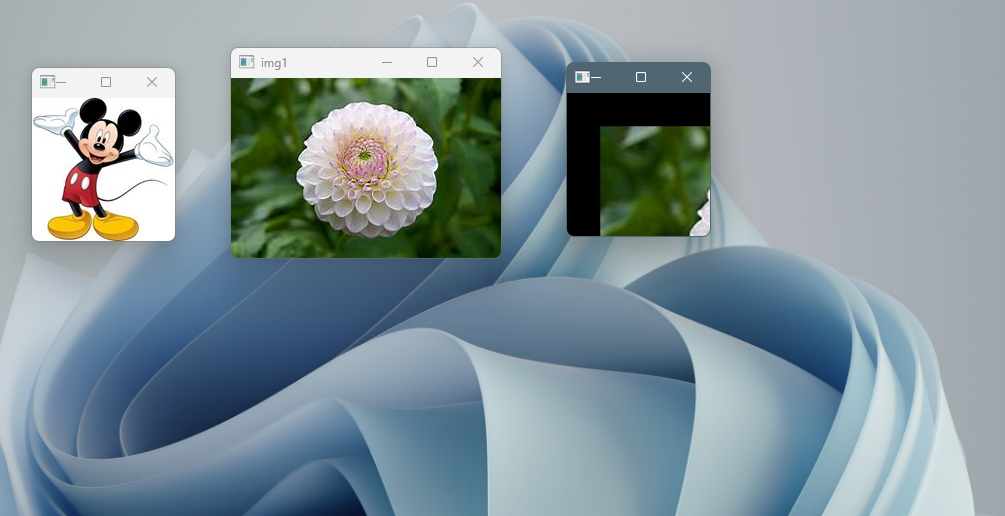
dst = cv2.warpPerspective(img1, H, (img2.shape[1], img2.shape[0])) # Display images

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

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****

1. 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)

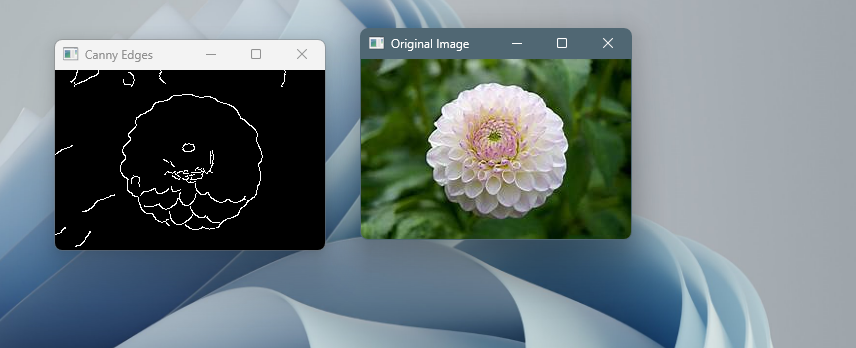
cv2.imshow('Canny Edges', edges)

cv2.imwrite('edges\_detected.jpg', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****

1. **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)

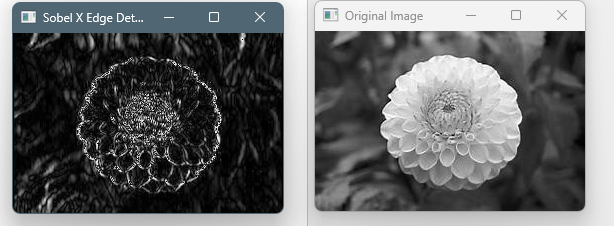
cv2.imshow('Original Image', image)

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

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****

1. 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)

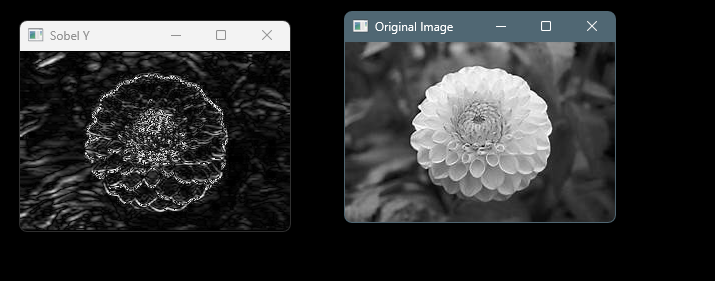
cv2.imshow('Original Image', image)

cv2.imshow('Sobel Y', sobel\_y)

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****

1. **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)

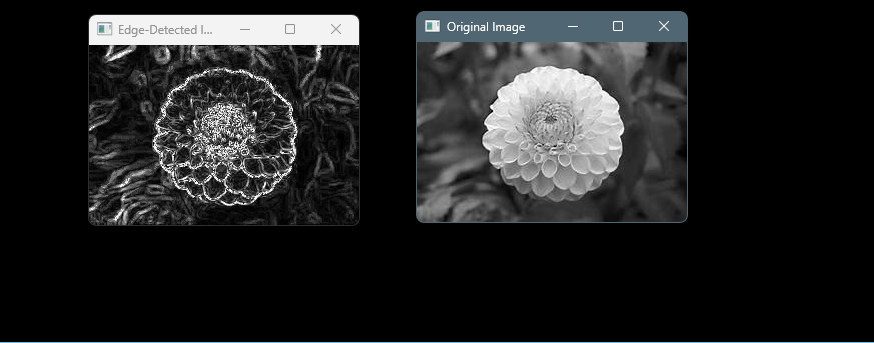
cv2.imshow('Edge-Detected Image', edge\_image)

cv2.imwrite('edge\_detected\_image.jpg', edge\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****

1. **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)

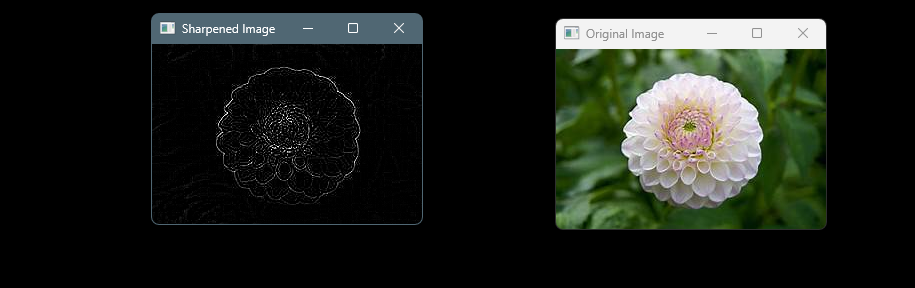
cv2.imshow('Sharpened Image', sharpened\_image)

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

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

1. **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)

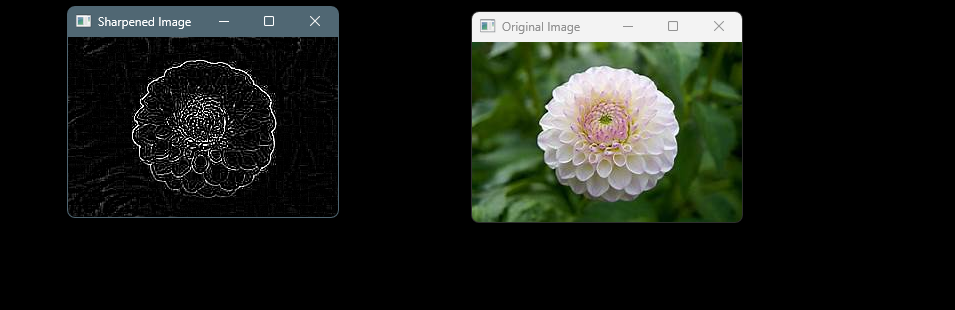
cv2.imshow('Sharpened Image', sharpened\_image)

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

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****

1. **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)

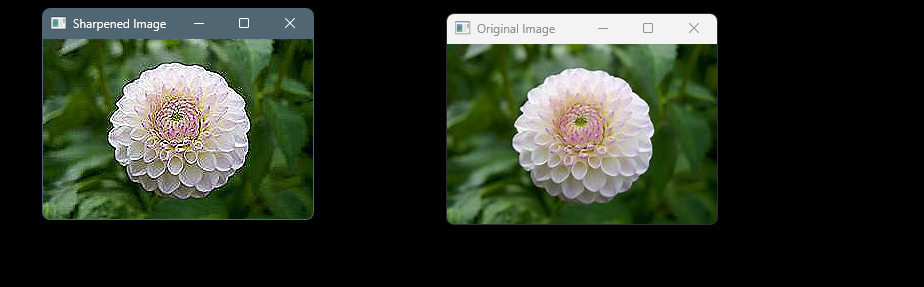
cv2.imshow('Original Image', image)

cv2.imshow('Sharpened Image', sharpened\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****

1. **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)

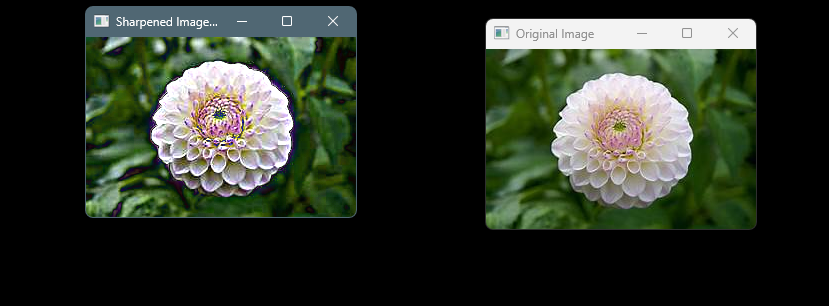
cv2.imshow('Original Image', image)

cv2.imshow('Sharpened Image (Unsharp Masking)', sharpened\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****

1. **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"

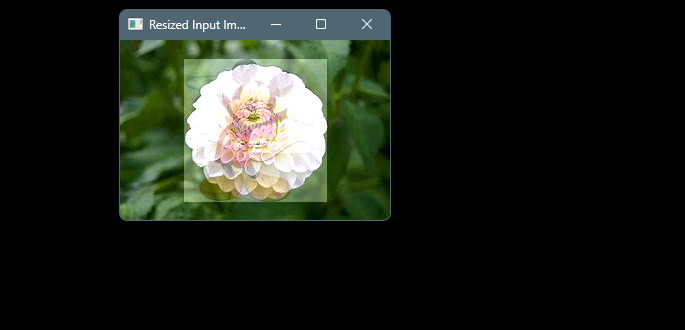
cv2.imwrite(filename, resized\_img)

cv2.imshow("Resized Input Image", resized\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****

1. **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)

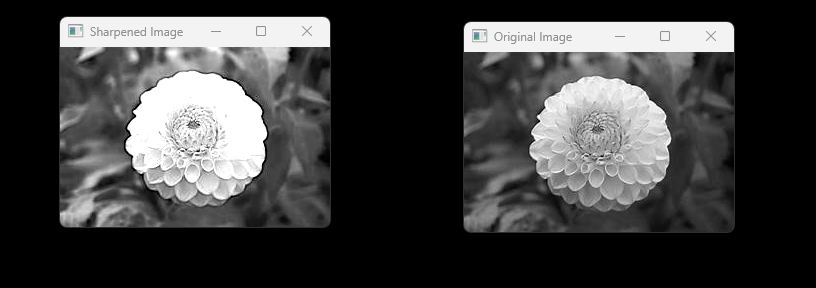
cv2.imshow('Original Image', gray)

cv2.imshow('Sharpened Image', sharpened)

cv2.waitKey(0)

cv2.destroyAllWindows()

**output:**

****