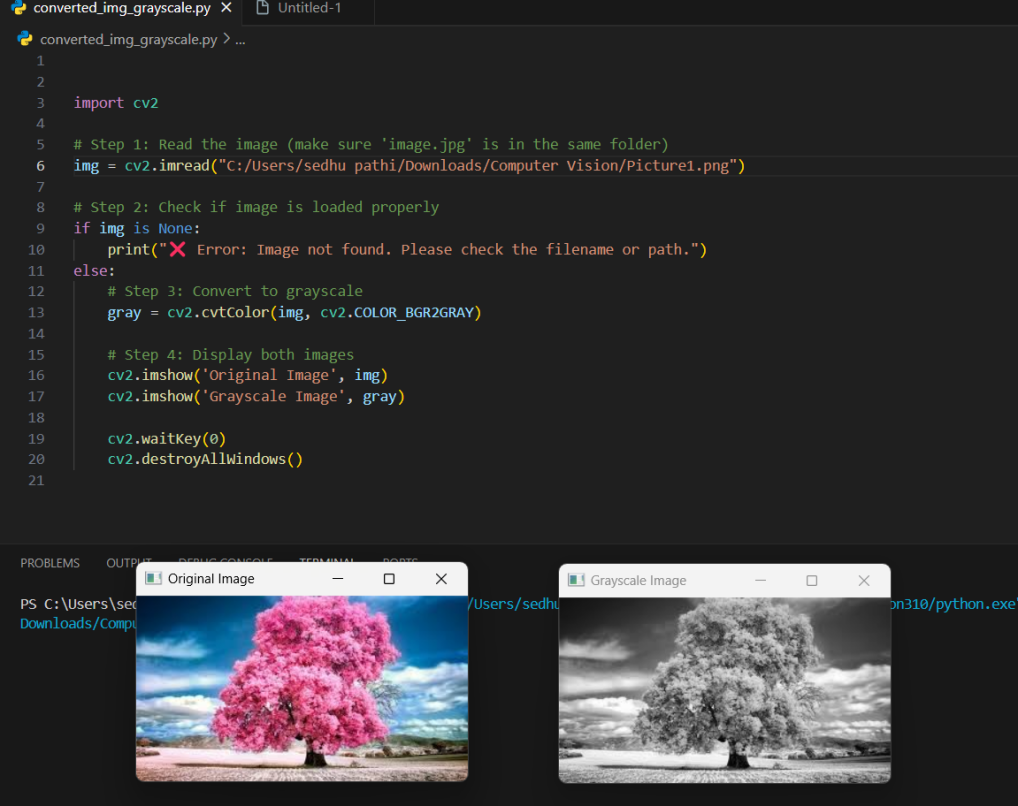
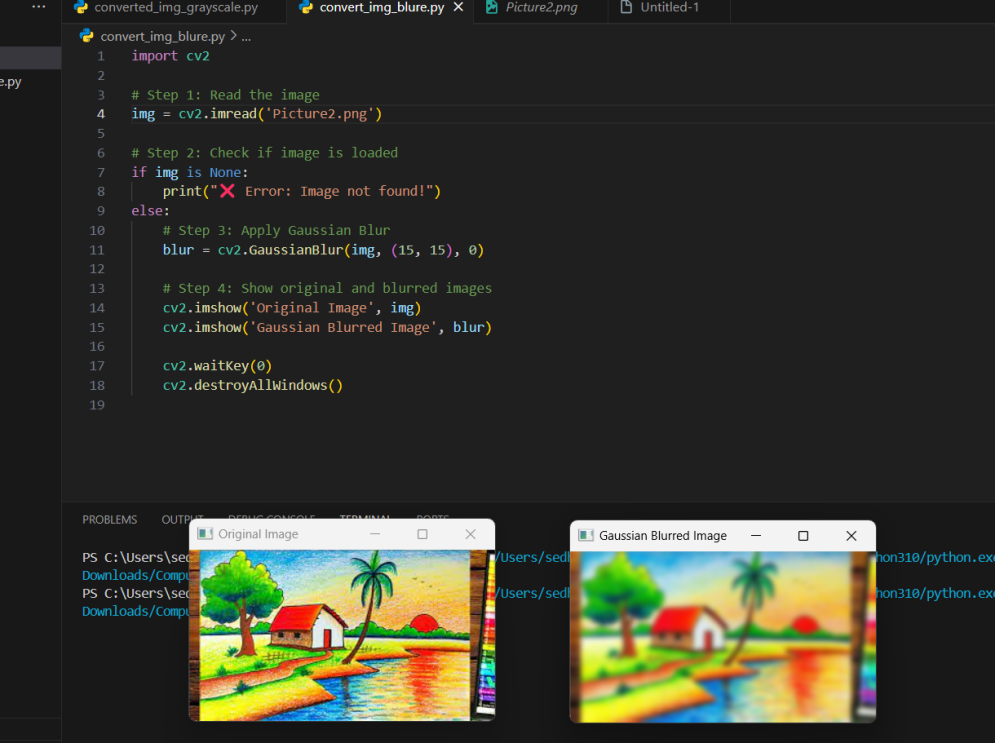
Lab Programs

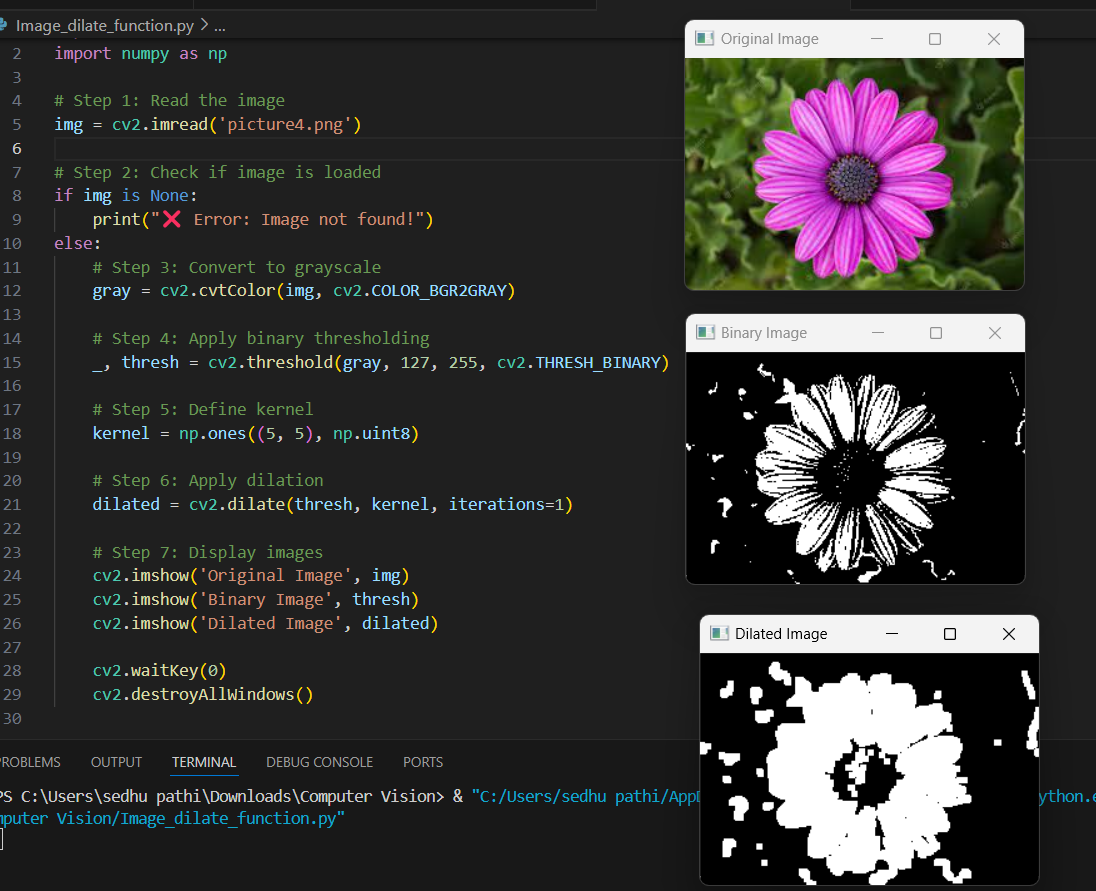
1.Convert the into grayscale



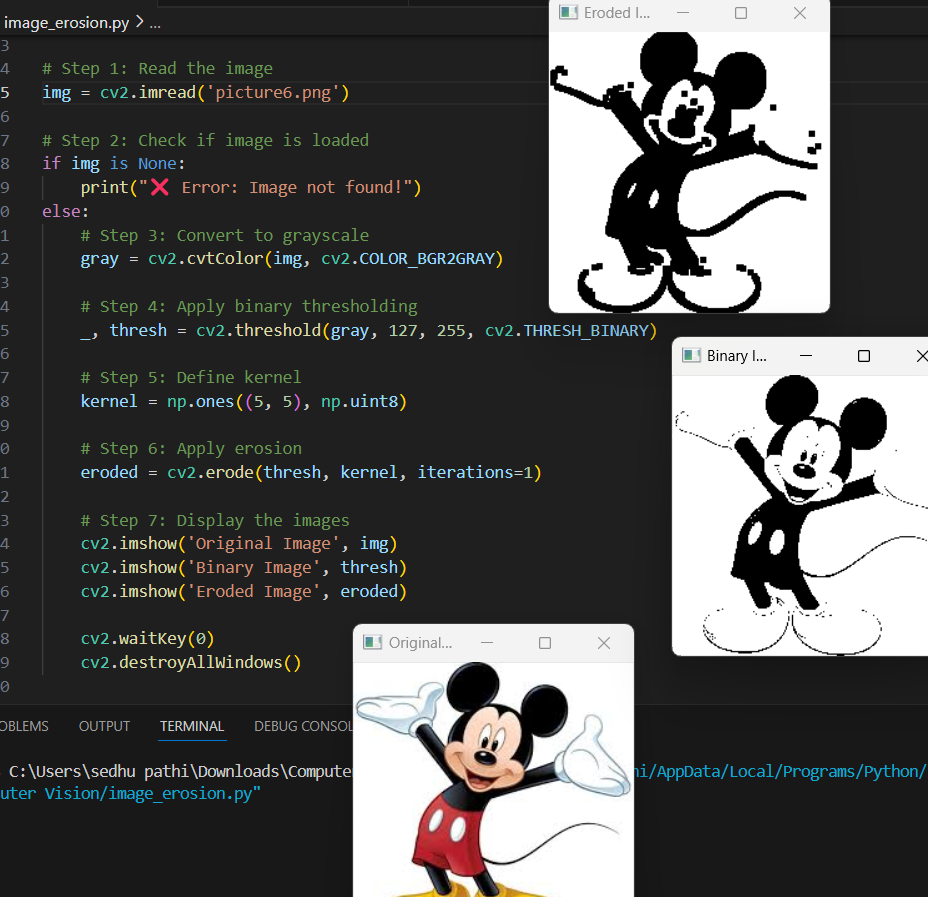
1. Convert the img into gaussian blur



1. Convert an Image to show outline using Canny function.  
     
   
2. Dilate an Image using Dilate function



1. read an image and apply erosion using OpenCV in Python.



6. display the video, in slow motion and in fast motion.

Code:

import cv2

# Step 1: Read the video file

cap = cv2.VideoCapture('video.mp4') # Replace with your video file name

# Step 2: Check if video opened successfully

if not cap.isOpened():

print("❌ Error: Cannot open video file.")

else:

# Step 3: Choose playback mode

print("Choose playback speed:")

print("1. Normal Motion")

print("2. Slow Motion")

print("3. Fast Motion")

choice = int(input("Enter choice (1/2/3): "))

# Step 4: Set delay based on choice

if choice == 1:

delay = 30 # ~30 fps (Normal)

elif choice == 2:

delay = 100 # Slower playback

elif choice == 3:

delay = 10 # Faster playback

else:

print("Invalid choice. Defaulting to normal.")

delay = 30

# Step 5: Read and display frames

while True:

ret, frame = cap.read()

if not ret:

break

cv2.imshow('Video Playback', frame)

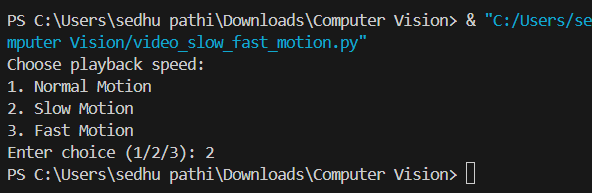
if cv2.waitKey(delay) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

Output:



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

Code:

import cv2

# Step 1: Open webcam (0 = default camera)

cap = cv2.VideoCapture(0)

# Step 2: Check if webcam opened

if not cap.isOpened():

print("❌ Error: Cannot access webcam.")

else:

# Step 3: Ask user for playback mode

print("Choose playback speed:")

print("1. Normal Motion")

print("2. Slow Motion")

print("3. Fast Motion")

choice = int(input("Enter choice (1/2/3): "))

# Step 4: Set delay based on choice

if choice == 1:

delay = 30

elif choice == 2:

delay = 100

elif choice == 3:

delay = 10

else:

print("Invalid choice. Defaulting to normal.")

delay = 30

print("📷 Press 'q' to quit webcam feed.")

# Step 5: Loop to read frames from webcam

while True:

ret, frame = cap.read()

if not ret:

break

cv2.imshow('Webcam Video', frame)

# WaitKey controls speed

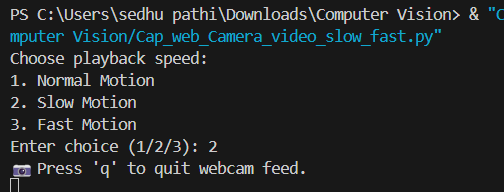
if cv2.waitKey(delay) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

Output:



8.Scaling an image to its Bigger and Smaller sizes

Code:

import cv2

# Step 1: Read the image

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

# Step 2: Check if image is loaded

if img is None:

print("❌ Error: Image not found!")

else:

# Step 3: Resize to smaller size (e.g., 50% of original)

small = cv2.resize(img, None, fx=0.5, fy=0.5, interpolation=cv2.INTER\_AREA)

# Step 4: Resize to bigger size (e.g., 200% of original)

big = cv2.resize(img, None, fx=2.0, fy=2.0, interpolation=cv2.INTER\_CUBIC)

# Step 5: Display all images

cv2.imshow('Original Image', img)

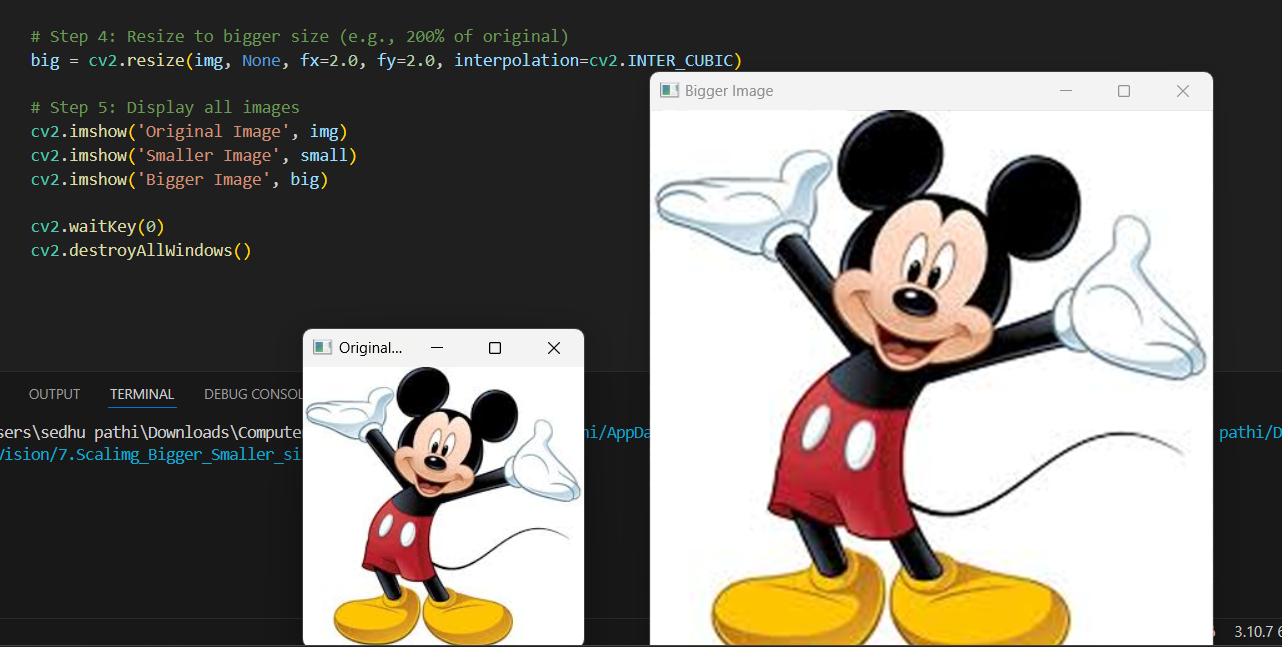
cv2.imshow('Smaller Image', small)

cv2.imshow('Bigger Image', big)

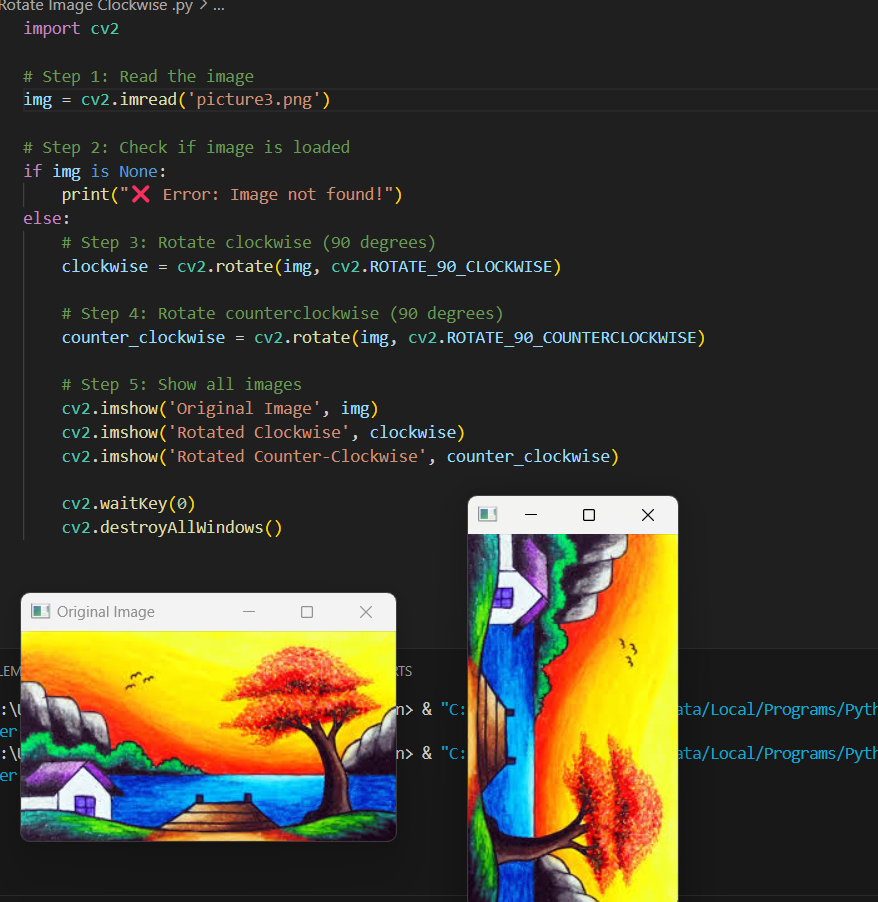
cv2.waitKey(0)

cv2.destroyAllWindows()

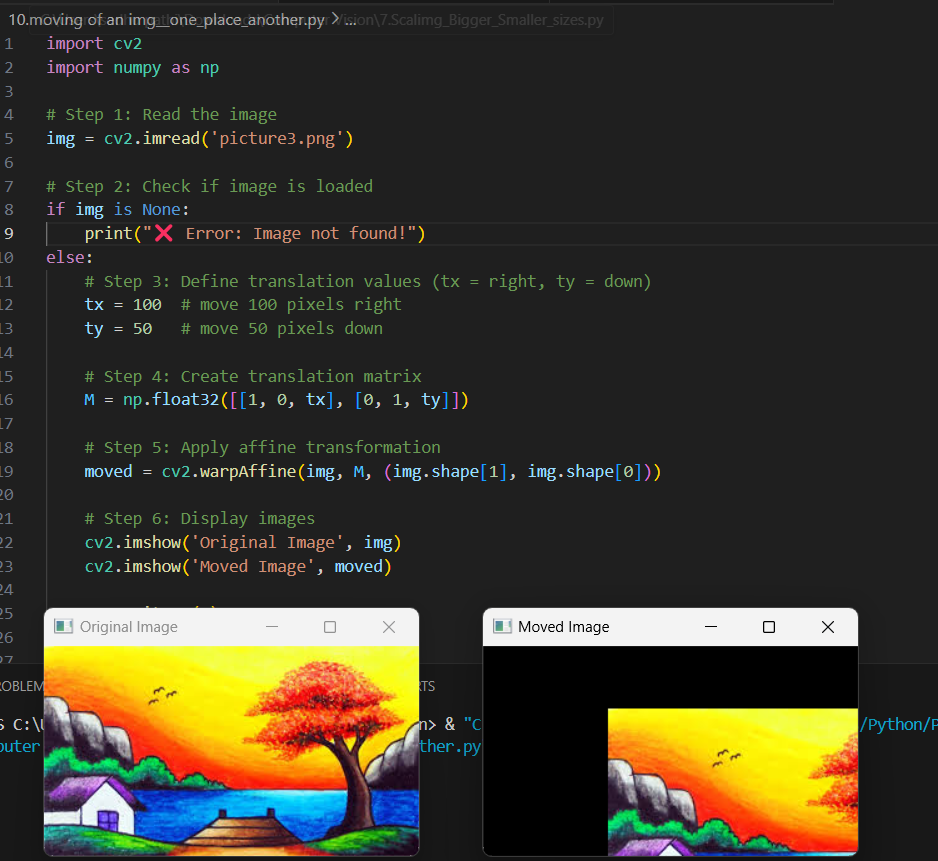
Output:



1. Rotate the image 90 degree



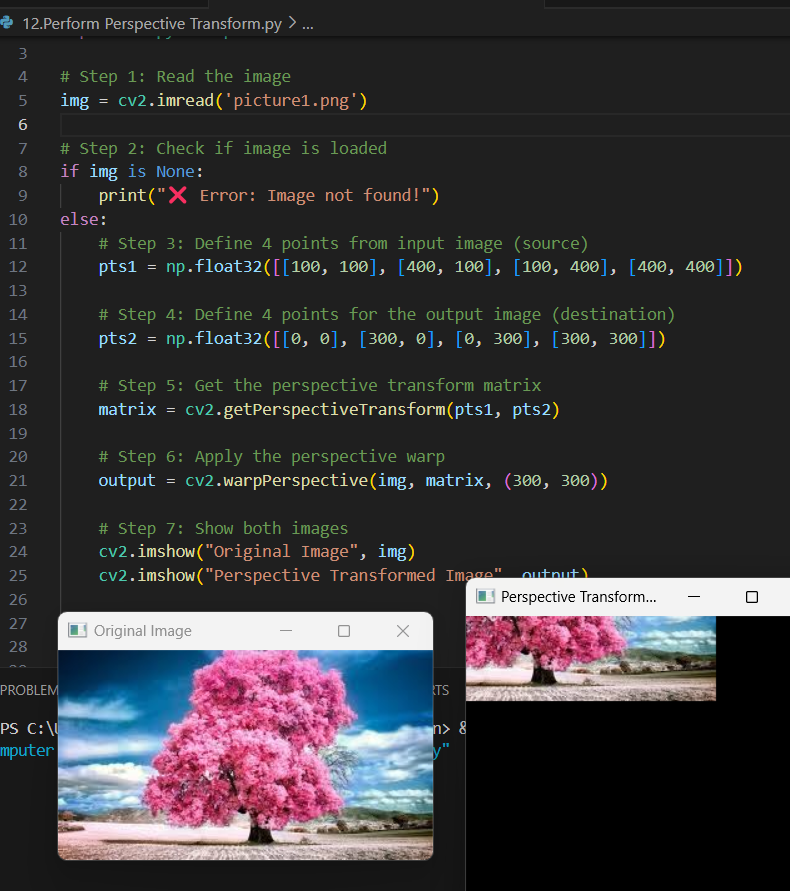
1. Move a image from one place to another



1. Perform Affine Transformation on the image.



12.Perform Perspective Transformation on the image.



13. Perform Perspective Transformation on the Video.

Code:

import cv2

import numpy as np

# Step 1: Open video file or webcam (0 = default webcam)

cap = cv2.VideoCapture('video.mp4') # or use 0 for live camera

# Step 2: Check if video is opened

if not cap.isOpened():

print("❌ Error: Cannot open video.")

exit()

# Step 3: Define source and destination points for transformation

# These should be inside the frame resolution (adjust as needed)

src\_points = np.float32([[100, 100], [500, 100], [100, 400], [500, 400]])

dst\_points = np.float32([[0, 0], [400, 0], [0, 300], [400, 300]])

# Step 4: Get the perspective transform matrix

M = cv2.getPerspectiveTransform(src\_points, dst\_points)

# Step 5: Read and transform each frame

while True:

ret, frame = cap.read()

if not ret:

break

# Apply perspective transformation

warped\_frame = cv2.warpPerspective(frame, M, (400, 300))

# Display the original and transformed frames

cv2.imshow("Original Frame", frame)

cv2.imshow("Perspective Transformed Frame", warped\_frame)

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

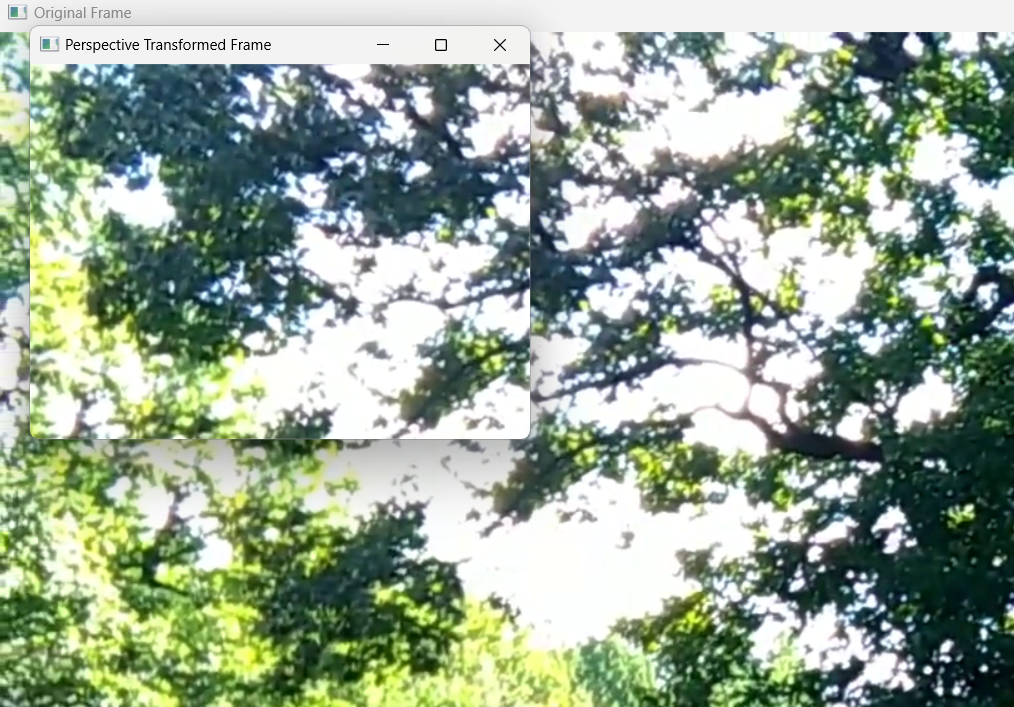
break

# Step 6: Release and close

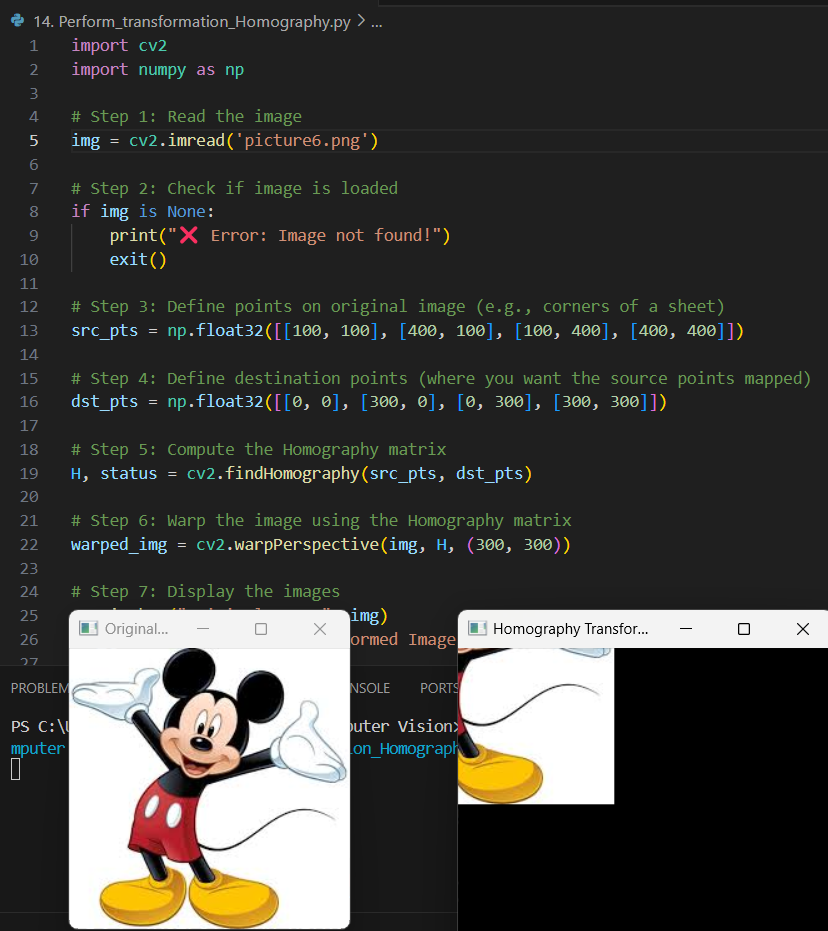
cap.release()

cv2.destroyAllWindows()

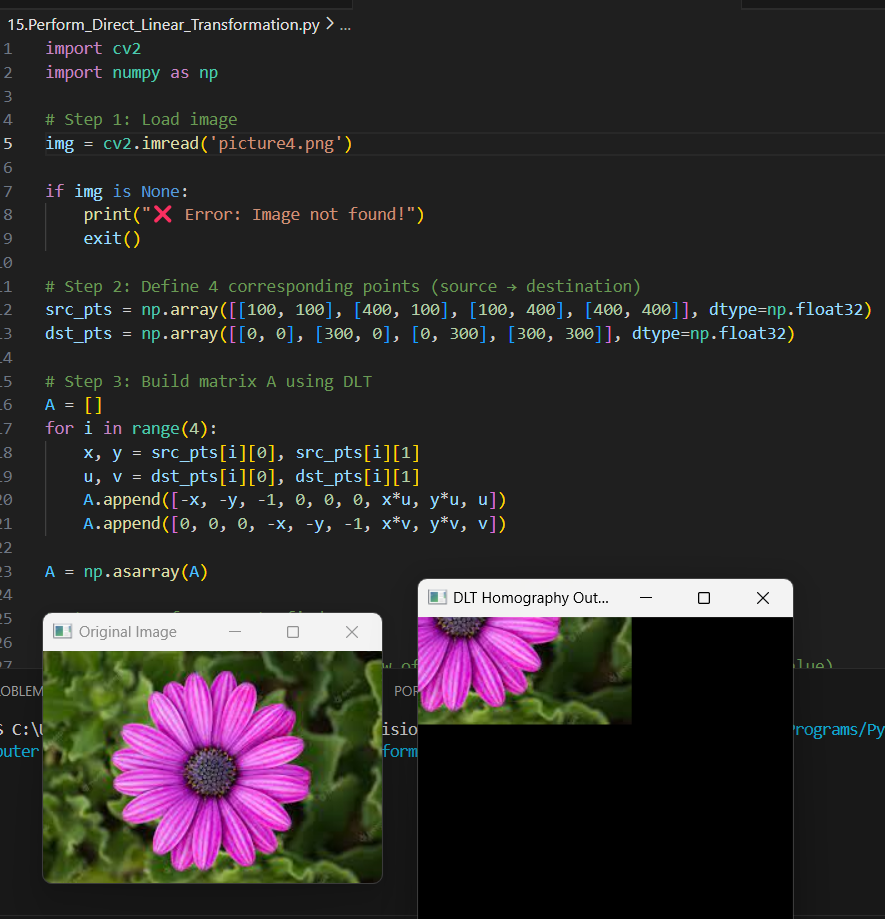
Output:



14. Perform transformation using Homography matrix.



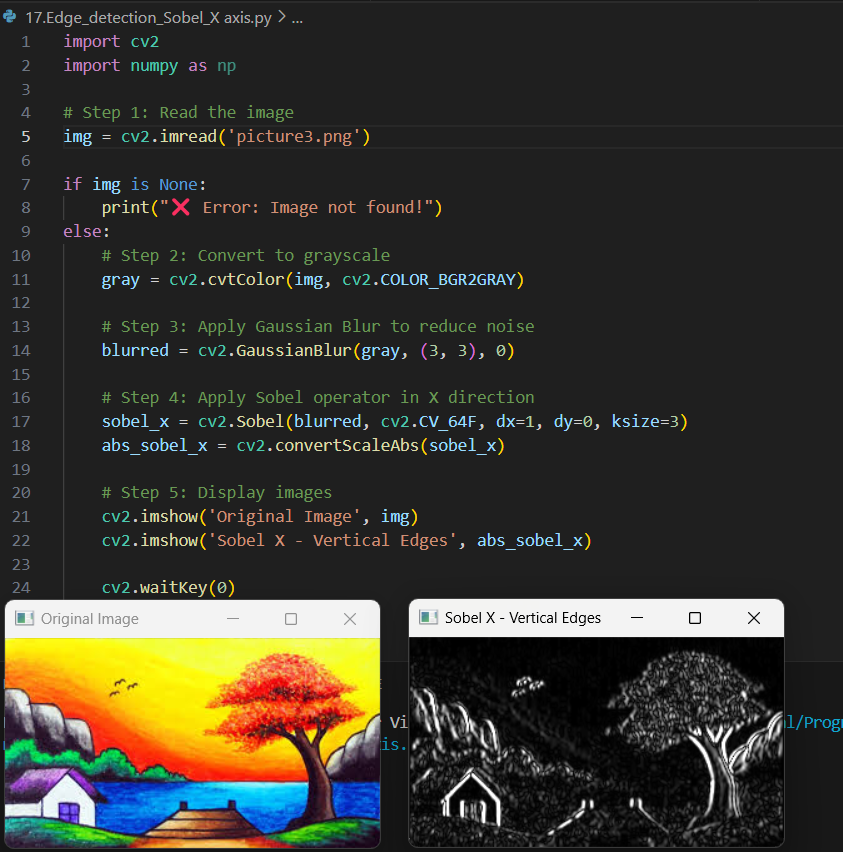
15. Perform transformation using Direct Linear Transformation.



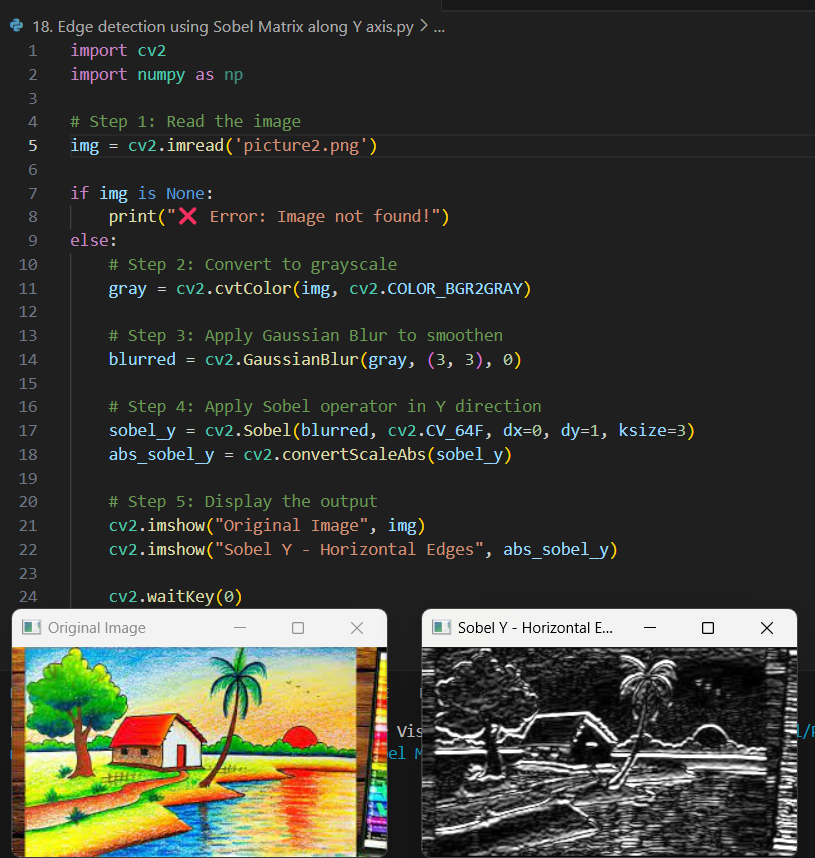
1. Perform Edge detection using canny method



17. Perform Edge detection using Sobel Matrix along X axis



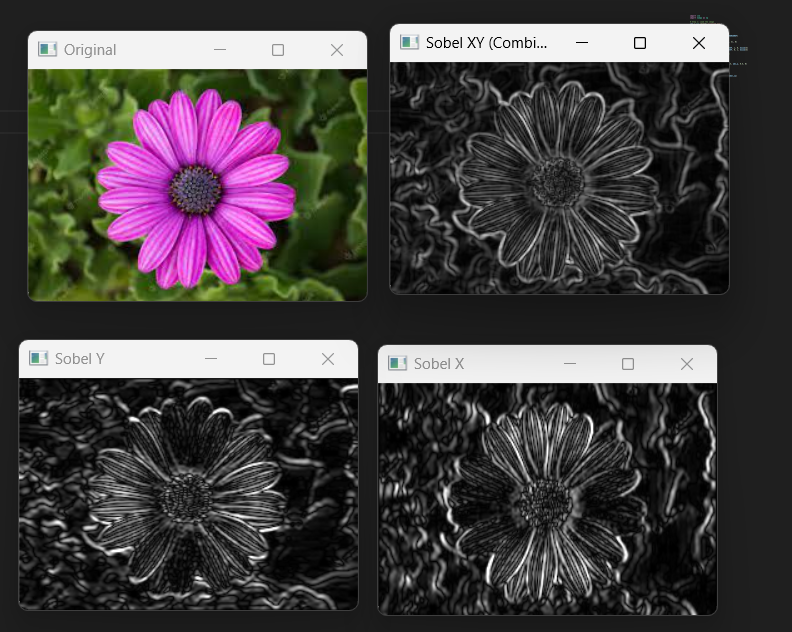
18. Perform Edge detection using Sobel Matrix along Y axis



19. Perform Edge detection using Sobel Matrix along XY axis



Output:



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

