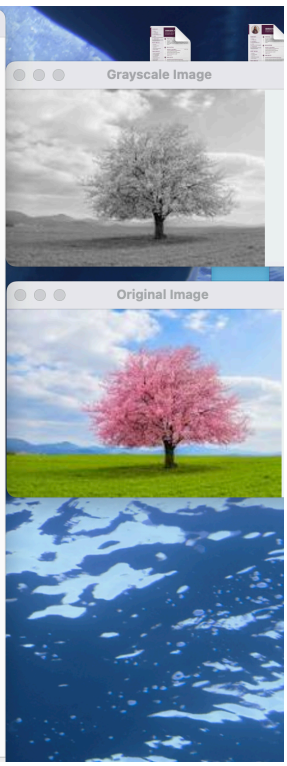
DSA0208 - COMPUTER VISION LAB EXPERIMENTS

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

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

opencv.py

```python
import cv2

# 1. Read an image in Python
image_path = '/Users/srinidhi/Desktop/pinktree.jpeg'  # Replace with your image filename
image = cv2.imread(image_path)
if image is None:
    print("Error: Image not found.")
else:
    cv2.imshow('Original Image', image)
    cv2.waitKey(0)

# 2. Convert image to another format (e.g., PNG)
converted_path = 'converted_image.png'
cv2.imwrite(converted_path, image)
print(f"Image saved as {converted_path}")

# 3. Convert the image to Grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Grayscale Image', gray_image)
cv2.waitKey(0)

# Save grayscale image
cv2.imwrite('grayscale_image.jpg', gray_image)

# 4. Play a video file
video_path = 'sample_video.mp4'  # Replace with your video filename
cap = cv2.VideoCapture(video_path)

if not cap.isOpened():
    print("Error: Cannot open video.")
else:
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            print("Reached end of video or can't fetch the frame.")
            break

        cv2.imshow('Video Playback', frame)

        if cv2.waitKey(25) & 0xFF == ord('q'):
            break

    cap.release()

# Cleanup
cv2.destroyAllWindows()
```

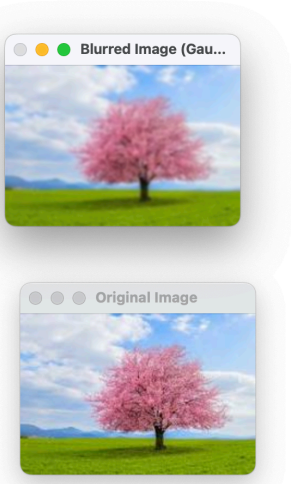


2. Perform basic Image Handling and processing operations on the image

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

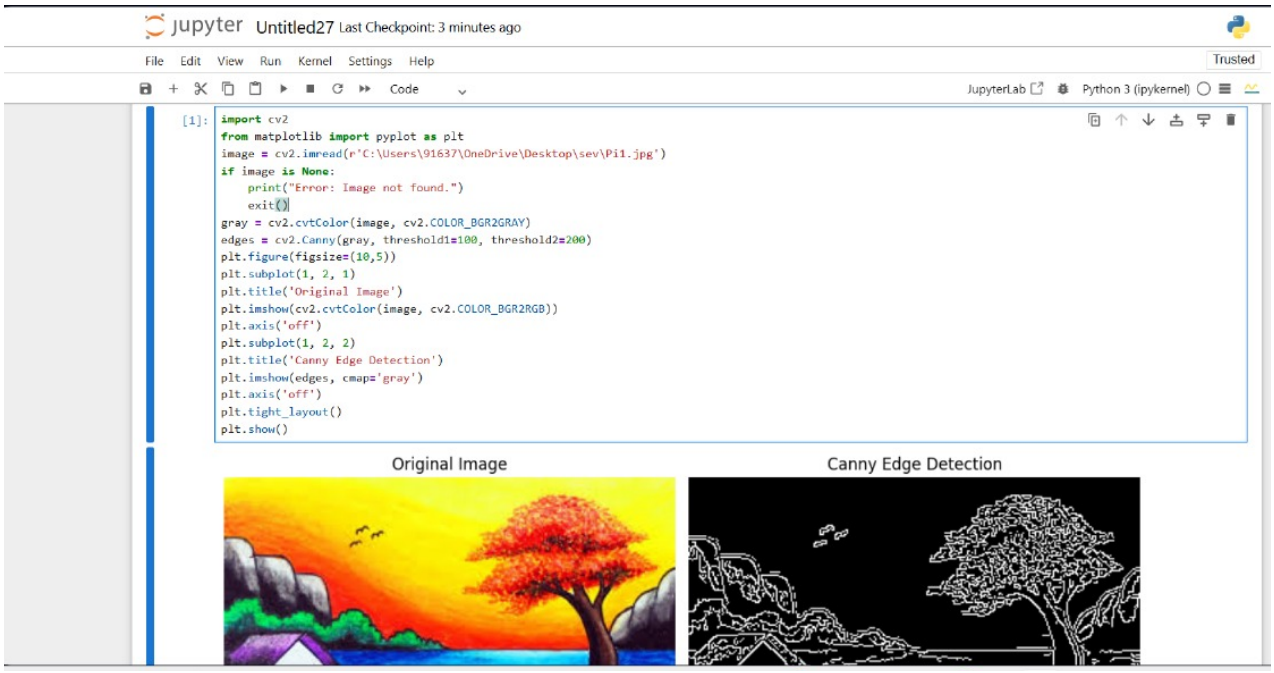
Gaussianblur.py

```python
import cv2
import os

# === Step 1: Read the image ===
image_path = '/Users/srinidhi/Desktop/pinktree.jpeg'  # Replace with your image file name
if not os.path.exists(image_path):
    print(f"Error: File '{image_path}' not found.")
    exit()

image = cv2.imread(image_path)

if image is None:
    print("Error: Could not read the image. Check file format or path.")
    exit()

# Display the original image
cv2.imshow("Original Image", image)
cv2.waitKey(0)

# === Step 2: Apply Gaussian Blur ===
# kernel size must be odd and positive (e.g., 5x5)
blurred_image = cv2.GaussianBlur(image, (5, 5), 0)

# Display the blurred image
cv2.imshow("Blurred Image (Gaussian Blur)", blurred_image)
cv2.waitKey(0)

# Optionally, save the blurred image
cv2.imwrite('blurred_image.jpg', blurred_image)

# Close all OpenCV windows
cv2.destroyAllWindows()
```



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

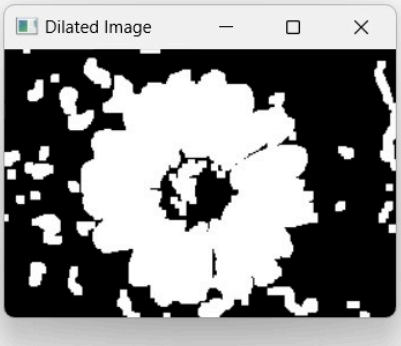
• Read an image in python and Convert an Image to show outline using Canny function.



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

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

```
import cv2
import numpy as np

# Replace this with your actual image file name or full path
image_path = 'C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\sample image.jpg'

# Read the image in grayscale mode
img = cv2.imread(image_path, 0)

# Check if the image was loaded successfully
if img is None:
    print("❌ Error: Image not found. Check the file name and path.")
else:
    # Convert to binary image using thresholding
    _, binary = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)

    # Define a kernel and apply dilation
    kernel = np.ones((5, 5), np.uint8)
    dilated = cv2.dilate(binary, kernel, iterations=1)

    # Show the result
    cv2.imshow('Dilated Image', dilated)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```



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

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

```
import cv2
import numpy as np

# Upload image
uploaded = files.upload()

# Read the image
image = cv2.imread('cvsess.png')  # Change to your filename

# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Define the kernel
kernel = np.ones((5, 5), np.uint8)

# Apply erosion
eroded = cv2.erode(gray, kernel, iterations=1)

# Display eroded image
cv2_imshow(eroded)
```

```
Choose Files cvsess.png
• cvsess.png(image/png) - 43862 bytes, last modified: 6/23/2025 - 100% done
Saving cvsess.png to cvsess (1).png
```

6. Perform basic video processing operations on the captured video

• Read captured video in python and display the video, in slow motion and in fast motion.

```
import cv2
# Path to your captured video file
video_path = "sample_video.mp4"  # Make sure the file is in the same directory or provide full path
# Open the video
cap = cv2.VideoCapture(video_path)
# Check if the video file opened successfully
if not cap.isOpened():
    print("Error: Could not open video.")
    exit()
```

```
# Get original frame rate
fps = cap.get(cv2.CAP_PROP_FPS)
print(f"Original FPS: {fps}")

# Function to play video with a custom delay (in ms)
def play_video(cap, delay, window_name):
    cap.set(cv2.CAP_PROP_POS_FRAMES, 0)  # Restart video
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        cv2.imshow(window_name, frame)
        if cv2.waitKey(delay) & 0xFF == ord('q'):
            break
    cv2.destroyWindow(window_name)

# Display original video
print("Playing original video...")
play_video(cap, int(1000 // fps), "Original Video")

# Play video in slow motion (e.g., 2x slower)
print("Playing slow motion video...")
play_video(cap, int((1000 // fps) * 2), "Slow Motion Video")

# Play video in fast motion (e.g., 2x faster)
print("Playing fast motion video...")
play_video(cap, int((1000 // fps) // 2), "Fast Motion Video")

# Release the capture object
cap.release()
cv2.destroyAllWindows()
```

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

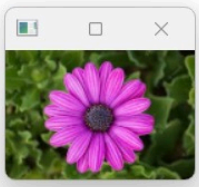
```
import cv2
import time
# Open the default webcam (device index 0)
cap = cv2.VideoCapture(0)
# Check if webcam opened successfully
if not cap.isOpened():
    print("Error: Cannot access webcam.")
    exit()
# Get the frame rate of the webcam (may not be accurate for all cameras)
fps = cap.get(cv2.CAP_PROP_FPS)
if fps == 0:
    fps = 30  # fallback default if camera doesn't provide FPS
print(f"Estimated webcam FPS: {fps}")
# Function to capture and display in real-time
def show_webcam_video(speed="normal"):
    delay = int(1000 // fps)  # default delay
    if speed == "slow":
        delay *= 2  # double the delay
        window = "Webcam - Slow Motion"
    elif speed == "fast":
        delay = max(1, delay // 2)  # halve the delay
        window = "Webcam - Fast Motion"
    else:
        window = "Webcam - Normal Speed"
    print(f"Playing: {window}")
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        cv2.imshow(window, frame)
        if cv2.waitKey(delay) & 0xFF == ord('q'):
```

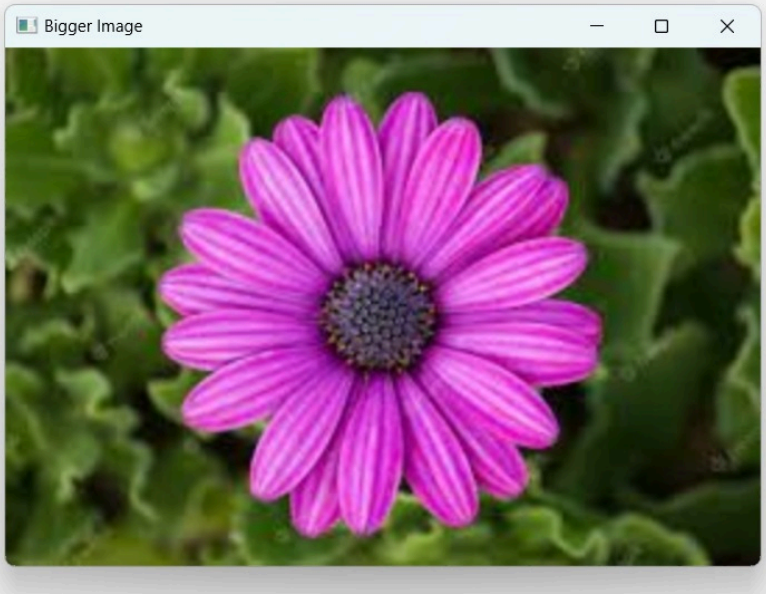
```
            break
    cv2.destroyWindow(window)
# Run in normal, slow, and fast motion
show_webcam_video("normal")
show_webcam_video("slow")
show_webcam_video("fast")
# Release the webcam and close windows
cap.release()
cv2.destroyAllWindows()
```

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

File Edit Format Run Options Window Help

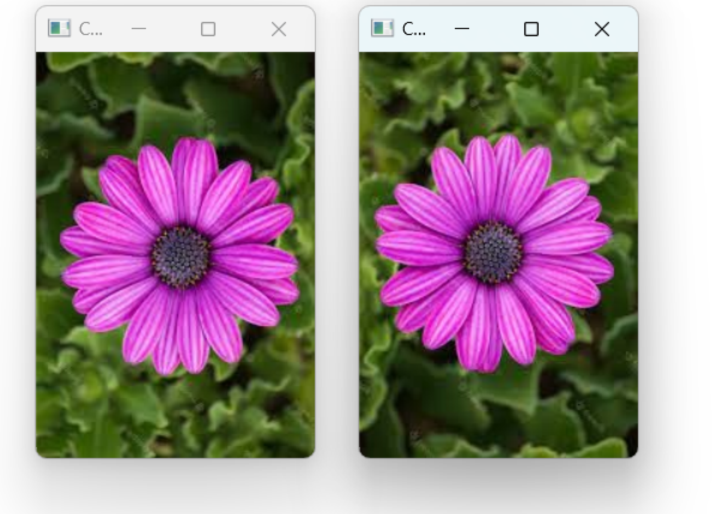
```
import cv2

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\sample image.jpg")
small = cv2.resize(img, None, fx=0.5, fy=0.5)
big = cv2.resize(img, None, fx=2.0, fy=2.0)

cv2.imshow('Smaller Image', small)
cv2.imshow('Bigger Image', big)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



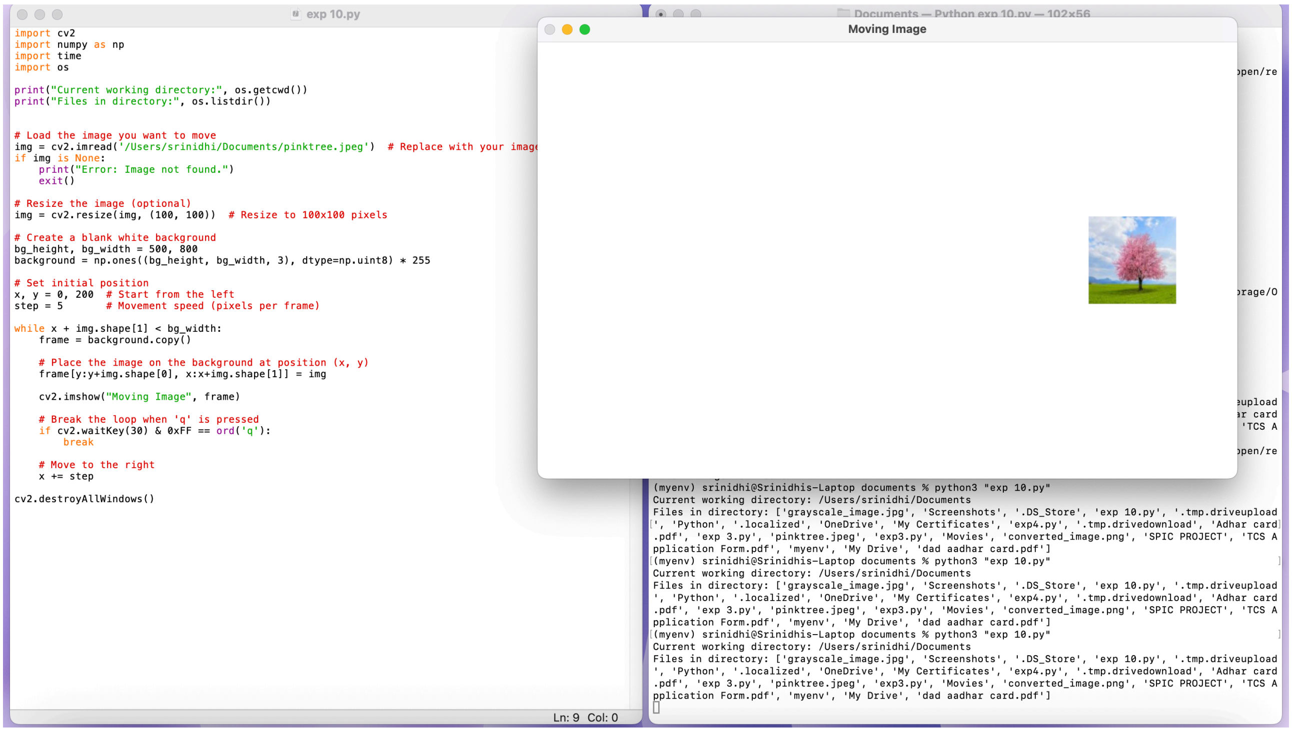


9. Perform Rotation of an image to clockwise and counter clockwise direction.

```
import cv2

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\sample image.jpg")
clockwise = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
counter = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)

cv2.imshow('Clockwise', clockwise)
cv2.imshow('Counter Clockwise', counter)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



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



11. Perform Affine Transformation on the image.

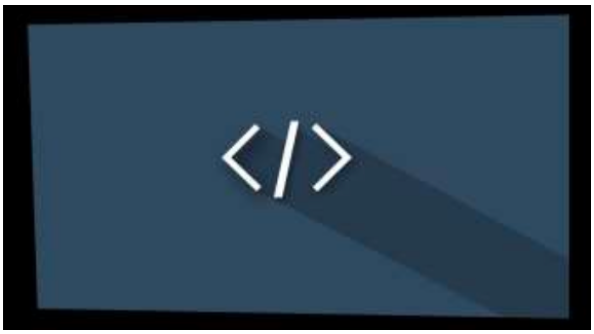
Affine Transformation preserves points, straight lines, and planes. It's defined by 3 points.

```
import cv2
import numpy as np
img = cv2.imread('input.jpg')
rows, cols = img.shape[:2]
# Source and destination points
pts1 = np.float32([[50, 50], [200, 50], [50, 200]])
pts2 = np.float32([[10, 100], [200, 50], [100, 250]])
# Affine transform matrix
M = cv2.getAffineTransform(pts1, pts2)
dst = cv2.warpAffine(img, M, (cols, rows))
cv2.imwrite('affine_output.jpg', dst)
```

affined image

12. Perform Perspective Transformation on the image.

```
img = cv2.imread('input.jpg')
rows, cols = img.shape[:2]
pts1 = np.float32([[0, 0], [cols - 1, 0], [0, rows - 1], [cols - 1, rows - 1]])
pts2 = np.float32([[50, 50], [cols - 50, 30], [70, rows - 50], [cols - 50, rows - 30]])
# Perspective matrix
M = cv2.getPerspectiveTransform(pts1, pts2)
dst = cv2.warpPerspective(img, M, (cols, rows))
cv2.imwrite('perspective_image.jpg', dst)
```



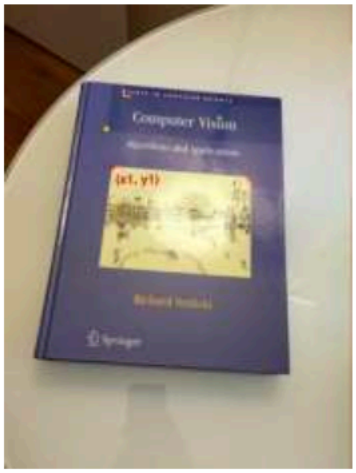
Perspective image

13. Perform Perspective Transformation on the Video.

```
import cv2
import numpy as np
cap = cv2.VideoCapture('input_video.mp4')
pts1 = np.float32([[100, 100], [400, 100], [100, 400], [400, 400]])
pts2 = np.float32([[150, 150], [350, 100], [130, 380], [370, 420]])
M = cv2.getPerspectiveTransform(pts1, pts2)
while cap.isOpened():
ret, frame = cap.read()
if not ret:
break
warped = cv2.warpPerspective(frame, M, (frame.shape[1], frame.shape[0]))
cv2.imshow('Perspective Video', warped)
if cv2.waitKey(1) & 0xFF == ord('q'):
break
```

14. Perform transformation using Homography matrix.

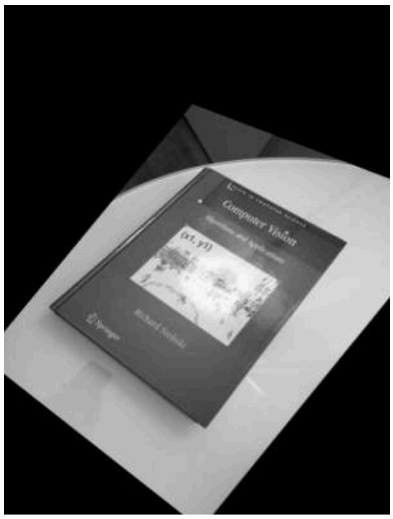
```
import cv2
import numpy as np
cap.release()
cv2.destroyAllWindows()
# Use ORB to detect keypoints
img1 = cv2.imread('img1.jpg', 0)
img2 = cv2.imread('img2.jpg', 0)
orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)
# Match features
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(des1, des2)
matches = sorted(matches, key=lambda x: x.distance)
# Use first 10 good matches
src_pts = np.float32([kp1[m.queryIdx].pt for m in matches[:10]]).reshape(-1, 1, 2)
dst_pts = np.float32([kp2[m.trainIdx].pt for m in matches[:10]]).reshape(-1, 1, 2)
# Get homography
H, _ = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC)
warped = cv2.warpPerspective(img1, H, (img2.shape[1], img2.shape[0]))
cv2.imwrite('homography_output.jpg', warped)
```



Book 1

Book 2

OUTPUT



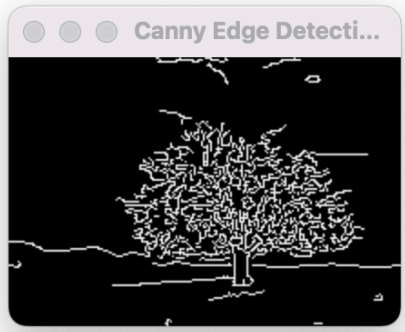
15. Perform transformation using Direct Linear Transformation.

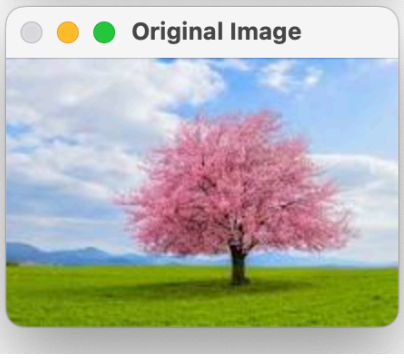
```
import cv2
import numpy as np
# Step 1: Read the image
img = cv2.imread('E://opencv//t and t//456d1b45951425.584281dfe6460.jpg')
if img is None:
exit()
print("Image not found. Make sure 'input.jpg' is in the same folder.")
# Step 2: Define 4 points in the original image
src_pts = np.float32([
[100, 100],
[300, 100],
```

```
[300, 300],
[100, 300]
])
# Step 3: Define where those points should map to in the output image
dst_pts = np.float32([
[80, 150],
[320, 100],
[310, 320],
[90, 330]
])# Step 4: Calculate the Homography matrix using DLT method
H, status = cv2.findHomography(src_pts, dst_pts, method=0) # method=0 means pure DLT
print("Homography Matrix (DLT):\n"
, H)
# Step 5: Apply the perspective transformation
warped = cv2.warpPerspective(img, H, (img.shape[1], img.shape[0]))
# Step 6: Show the result
cv2.imshow("Original Image", img)
cv2.imshow("Warped Image using DLT", warped)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

16. Perform Edge detection using canny method

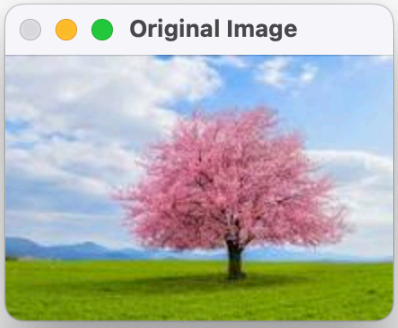
```
import cv2
import os

# (Optional) Change directory if needed
# os.chdir("/Users/srinidhi/Documents")

# Read the image
img = cv2.imread("/Users/srinidhi/Documents/pinktree.jpeg")

# Check if image loaded
if img is None:
    print("Error: Image not found.")
    exit()

# Convert image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Apply Canny edge detection
edges = cv2.Canny(gray, threshold1=100, threshold2=200)

# Display original and edge-detected images
cv2.imshow("Original Image", img)
cv2.imshow("Canny Edge Detection", edges)

cv2.waitKey(0)
cv2.destroyAllWindows()
```





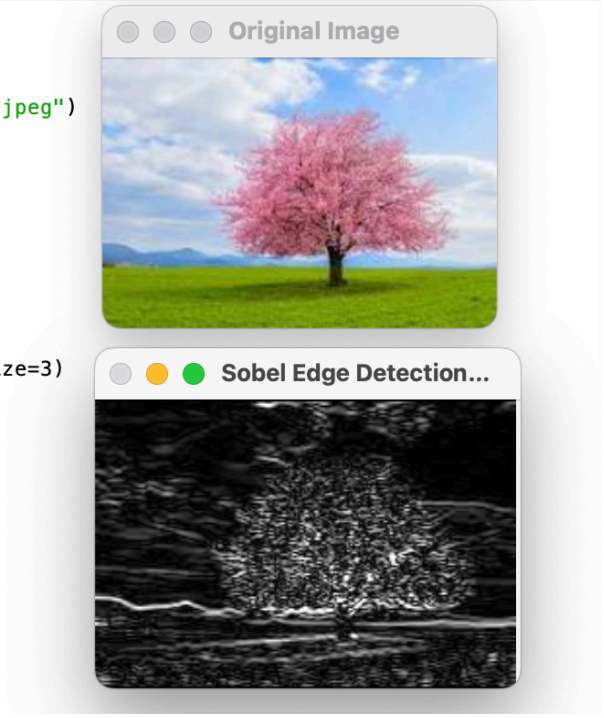
17. Perform Edge detection using Sobel Matrix along X axis

```
import cv2
import os

# (Optional) Change directory if needed
# os.chdir("/Users/srinidhi/Documents")

# Read the image
img = cv2.imread("/Users/srinidhi/Documents/pinktree.jpeg")

# Check if image loaded
if img is None:
    print("Error: Image not found.")
    exit()

# Convert image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Apply Sobel operator along X axis
sobel_x = cv2.Sobel(gray, cv2.CV_64F, dx=1, dy=0, ksize=3)

# Convert scale to absolute values and 8-bit
sobel_x = cv2.convertScaleAbs(sobel_x)

# Display original and Sobel X images
cv2.imshow("Original Image", img)
cv2.imshow("Sobel Edge Detection - X axis", sobel_x)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



18. Perform Edge detection using Sobel Matrix along Y axis

```
import cv2
import os

# Read the image
img = cv2.imread("/Users/srinidhi/Documents/pinktree.jpeg")

# Check if image is loaded
if img is None:
    print("Error: Image not found.")
    exit()

# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Apply Sobel operator along Y axis
sobel_y = cv2.Sobel(gray, cv2.CV_64F, dx=0, dy=1, ksize=3)

# Convert to 8-bit absolute values
sobel_y = cv2.convertScaleAbs(sobel_y)

# Display original and Sobel Y edge-detected images
cv2.imshow("Original Image", img)
cv2.imshow("Sobel Edge Detection - Y axis", sobel_y)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



19. Perform Edge detection using Sobel Matrix along XY axis

```
import cv2
import os


# Read the image
img = cv2.imread("/Users/srinidhi/Documents/pinktree.jpeg")

# Check if image is loaded
if img is None:
    print("Error: Image not found.")
    exit()

# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Apply Sobel along X and Y
sobel_x = cv2.Sobel(gray, cv2.CV_64F, dx=1, dy=0, ksize=3)
sobel_y = cv2.Sobel(gray, cv2.CV_64F, dx=0, dy=1, ksize=3)

# Combine the gradients
sobel_combined = cv2.magnitude(sobel_x, sobel_y)
sobel_combined = cv2.convertScaleAbs(sobel_combined)

# Display results
cv2.imshow("Original Image", img)
cv2.imshow("Sobel Edge Detection - XY Combined", sobel_combined)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

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

0	1	0
1	−4	1
0	1	0

```python
import cv2
import numpy as np

# Load the image
img = cv2.imread("/Users/srinidhi/Documents/pinktree.jpeg")

# Check if the image is loaded
if img is None:
    print("Error: Image not found.")
    exit()

# Convert to grayscale for sharpening
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Define Laplacian kernel with negative center coefficient
laplacian_kernel = np.array([[0,  -1,  0],
                             [-1,  4, -1],
                             [0,  -1,  0]])

# Apply the Laplacian filter
laplacian = cv2.filter2D(gray, -1, laplacian_kernel)

# Sharpen image by subtracting Laplacian from original grayscale
sharpened = cv2.subtract(gray, laplacian)

# Display the images
cv2.imshow("Original Grayscale", gray)
cv2.imshow("Laplacian Mask Output", laplacian)
cv2.imshow("Sharpened Image", sharpened)

cv2.waitKey(0)
cv2.destroyAllWindows()
```