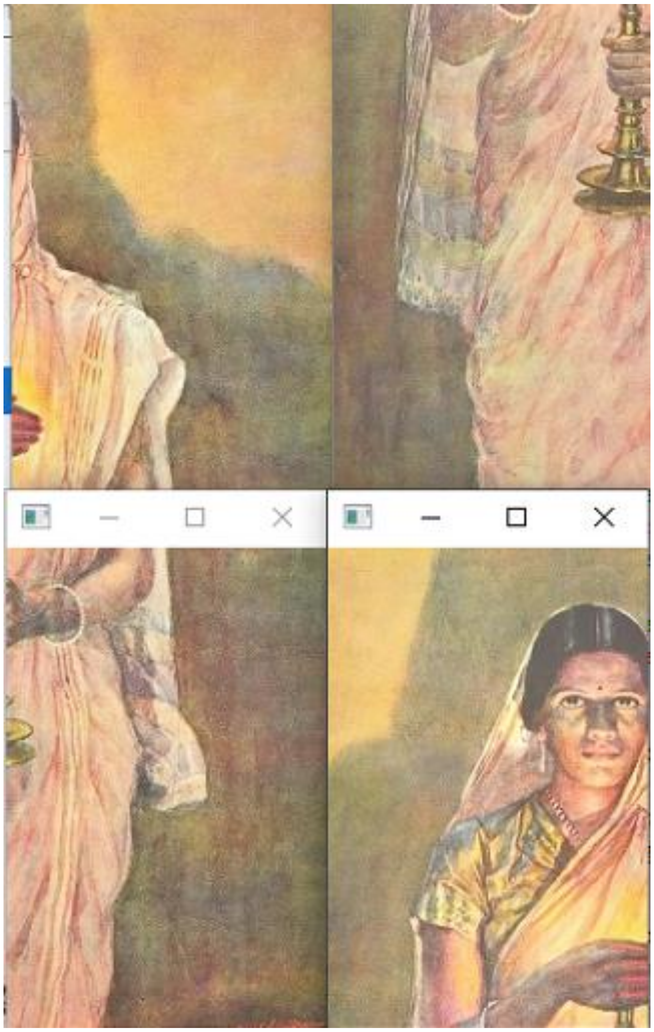
Program7: Write a Program to read a digital image. Split and display image into 4 quadrants, up, down, right and left.

```
import cv2

# Function to split the image into four quadrants
def split_image(image):
    height, width, _ = image.shape
    half_height = height // 2
    half_width = width // 2

    # Split the image into four quadrants
    top_left = image[:half_height, :half_width]
    top_right = image[:half_height, half_width:]
    bottom_left = image[half_height:, :half_width]
    bottom_right = image[half_height:, half_width:]

    return top_left, top_right, bottom_left, bottom_right

# Function to display images
def display_images(images, window_names):
    for img, name in zip(images, window_names):
        cv2.imshow(name, img)

    print("Press any key to terminate.")
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Read the image
image_path = "E:/2023_2024_Even/Pgm_Images/Fig2.jpg"  # Replace "image.jpg" with the path to your image
image = cv2.imread(image_path)
if image is None:
    print("Failed to load the image.")
else:
    # Split the image into quadrants
    top_left, top_right, bottom_left, bottom_right = split_image(image)
```

```
# Display the quadrants
    display_images([top_left, top_right, bottom_left, bottom_right], ["Top Left",
"Top Right", "Bottom Left", "Bottom Right"])
```

Output:

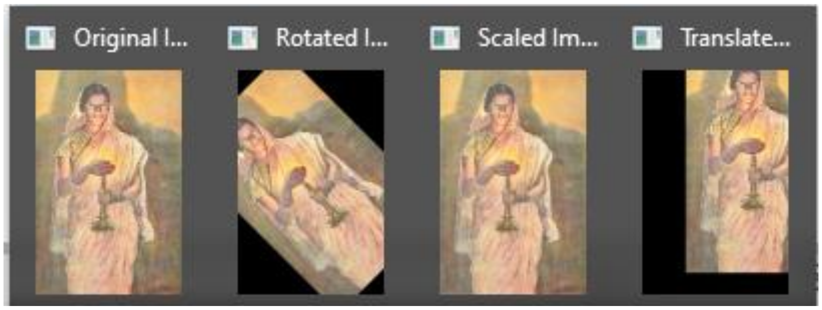


Program 8: Write a program to show rotation, scaling, and translation on an image.

```
import cv2
import numpy as np

# Read the image
image_path = "E:/2023_2024_Even/Pgm_Images/Fig2.jpg" # Replace "your_image.jpg" with the path to your image
image = cv2.imread(image_path)

if image is None:
    print("Failed to load the image.")
else:
    # Display the original image
    cv2.imshow("Original Image", image)

    # Rotation
    angle = 45  # Rotation angle in degrees
    center = (image.shape[1] // 2, image.shape[0] // 2)  # Center of rotation
    rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)  # Rotation matrix
    rotated_image = cv2.warpAffine(image, rotation_matrix, (image.shape[1], image.shape[0]))

    # Scaling
    scale_factor = 0.5  # Scaling factor (0.5 means half the size)
    scaled_image = cv2.resize(image, None, fx=scale_factor, fy=scale_factor)

    # Translation
    translation_matrix = np.float32([[1, 0, 100], [0, 1, -50]])  # Translation matrix (100 pixels right, 50 pixels up)
    translated_image = cv2.warpAffine(image, translation_matrix, (image.shape[1], image.shape[0]))

    # Display the transformed images
    cv2.imshow("Rotated Image", rotated_image)
    cv2.imshow("Scaled Image", scaled_image)
    cv2.imshow("Translated Image", translated_image)
```

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

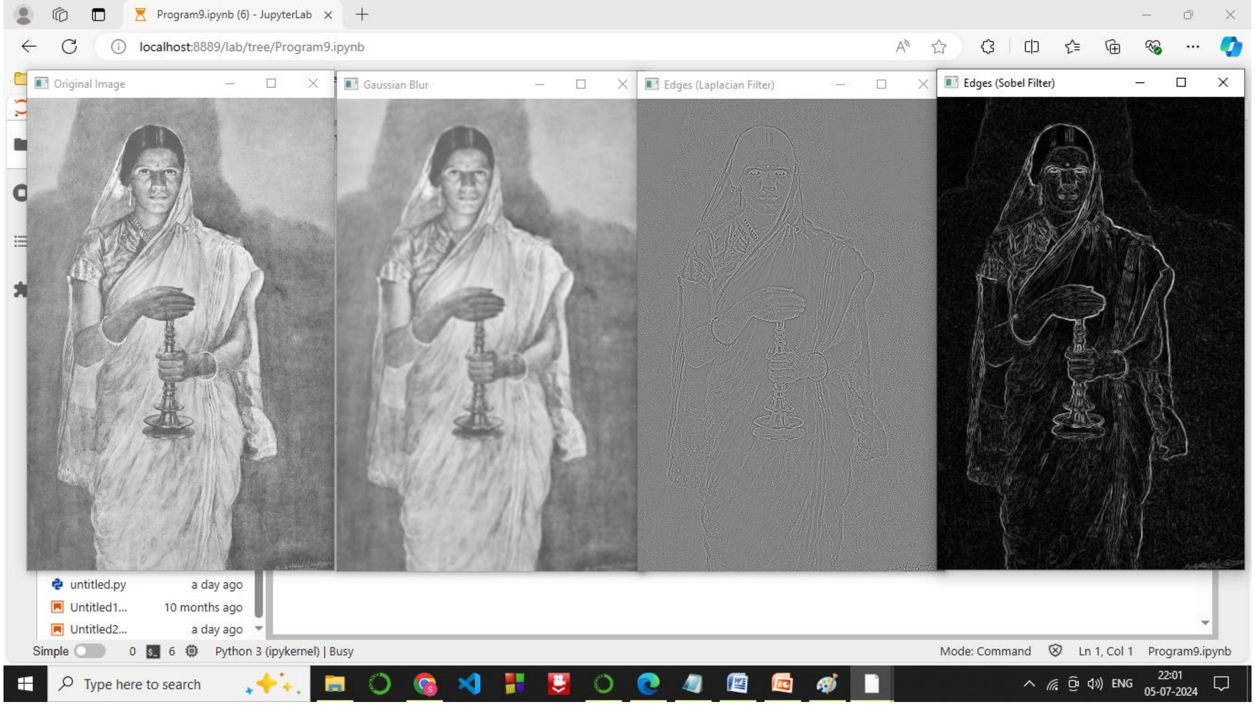
Output:



Program 9: Read an image and extract and display low-level features such as edges, textures using filtering techniques.

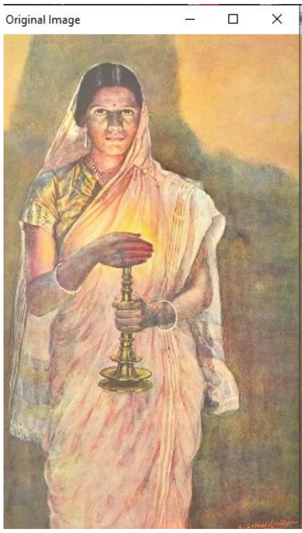
```
import cv2
import numpy as np

# Read the image
image_path = "E:/2023_2024_Even/Pgm_Images/Fig2.jpg"  # Replace
"your_image.jpg" with the path to your image
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

if image is None:
    print("Failed to load the image.")
else:
    # Display the original image
    cv2.imshow("Original Image", image)

    # Apply Sobel filter to extract edges
    sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
    sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
    sobel_edges = cv2.magnitude(sobel_x, sobel_y)
    sobel_edges = cv2.normalize(sobel_edges, None, 0, 255,
cv2.NORM_MINMAX, dtype=cv2.CV_8U)

    # Display edges extracted using Sobel filter
    cv2.imshow("Edges (Sobel Filter)", sobel_edges)

    # Apply Laplacian filter to extract edges
    laplacian_edges = cv2.Laplacian(image, cv2.CV_64F)
    laplacian_edges = cv2.normalize(laplacian_edges, None, 0, 255,
cv2.NORM_MINMAX, dtype=cv2.CV_8U)

    # Display edges extracted using Laplacian filter
    cv2.imshow("Edges (Laplacian Filter)", laplacian_edges)

    # Apply Gaussian blur to extract textures
    gaussian_blur = cv2.GaussianBlur(image, (5, 5), 0)

    # Display image with Gaussian blur
```

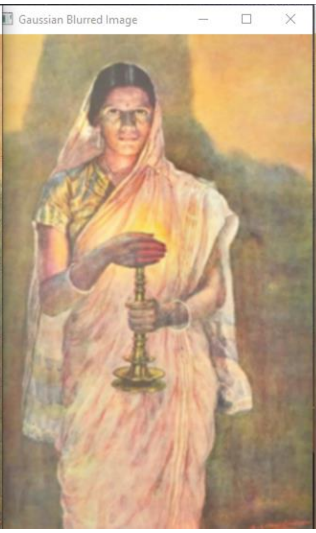
```
cv2.imshow("Gaussian Blur", gaussian_blur)

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

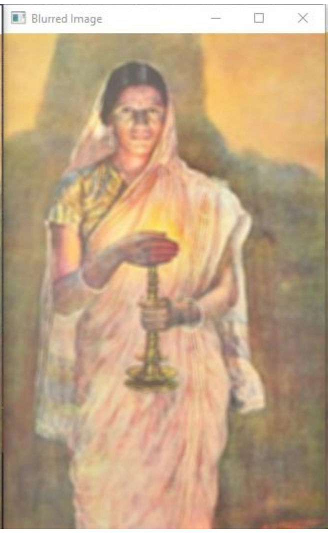


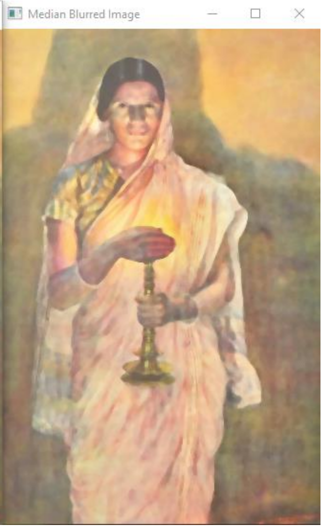
Program10: Write a program to blur and smoothing an image.

```
import cv2

# Read the image
image_path = "E:/2023_2024_Even/Pgm_Images/Fig2.jpg"  # Replace
"your_image.jpg" with the path to your image
image = cv2.imread(image_path)

if image is None:
    print("Failed to load the image.")
else:
    # Display the original image
    cv2.imshow("Original Image", image)

    # Apply blur to the image
    blur_kernel_size = (5, 5)  # Kernel size for blur filter
    blurred_image = cv2.blur(image, blur_kernel_size)

    # Display the blurred image
    cv2.imshow("Blurred Image", blurred_image)

    # Apply Gaussian blur to the image
    gaussian_blur_kernel_size = (5, 5)  # Kernel size for Gaussian blur filter
    gaussian_blurred_image = cv2.GaussianBlur(image, gaussian_blur_kernel_size,
0)

    # Display the Gaussian blurred image
    cv2.imshow("Gaussian Blurred Image", gaussian_blurred_image)

    # Apply median blur to the image
    median_blur_kernel_size = 5  # Kernel size for median blur filter (should be
odd)
    median_blurred_image = cv2.medianBlur(image, median_blur_kernel_size)

    # Display the median blurred image
    cv2.imshow("Median Blurred Image", median_blurred_image)

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


Original Image

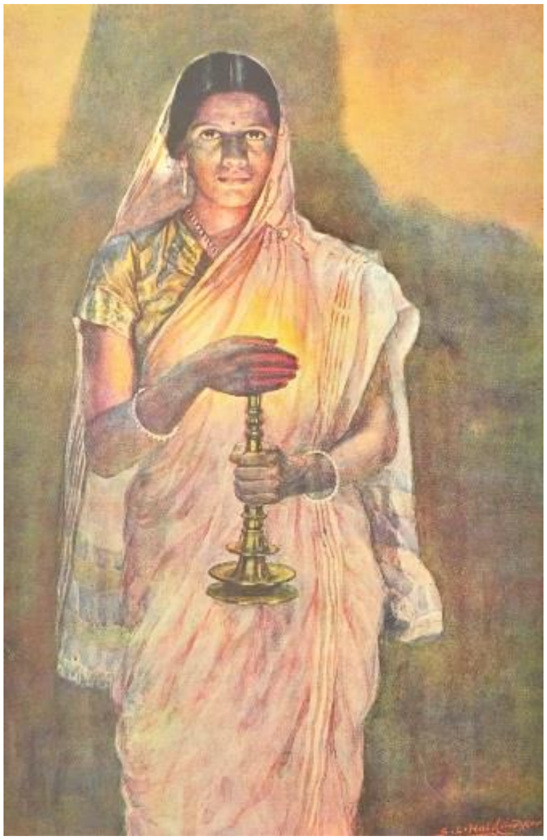

Gaussian Blurred Image


Blurred Image


Median Blurred Image

Program11: Write a program to contour an image.

```
import cv2

# Read the image
image_path = "E:/2023_2024_Even/Pgm_Images/Fig2.jpg"  # Replace
"your_image.jpg" with the path to your image
image = cv2.imread(image_path)

if image is None:
    print("Failed to load the image.")
else:
    # Convert the image to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply adaptive thresholding
    _, thresh = cv2.threshold(gray_image, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)

    # Find contours in the thresholded image
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # Draw contours on the original image
    contour_image = image.copy()
    cv2.drawContours(contour_image, contours, -1, (0, 255, 0), 2)  # Draw all
contours with green color and thickness 2

    # Display the original image with contours
    cv2.imshow("Image with Contours", contour_image)

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



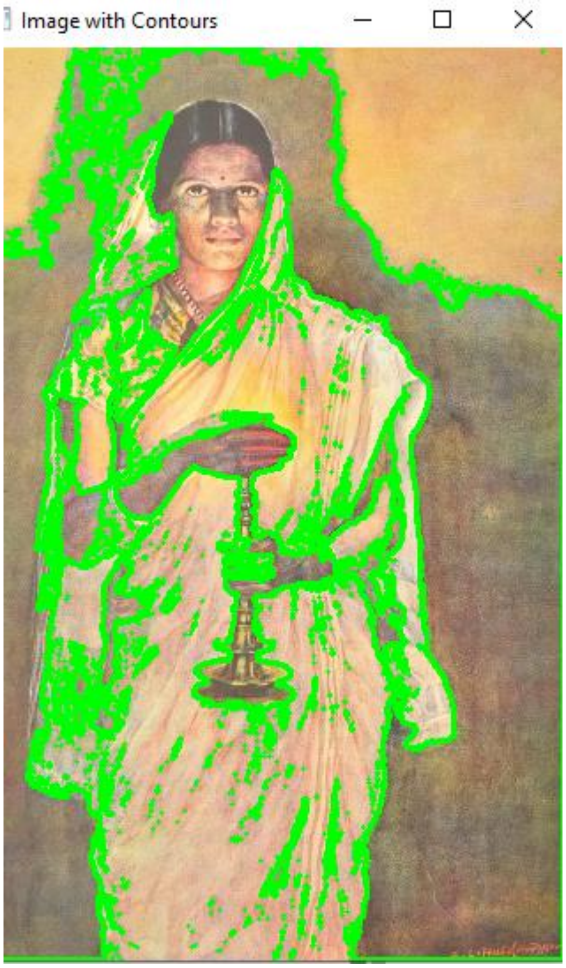


Image with Contours

Program12: Write a program to detect a face/s in an image.

```
import cv2

# Load the cascade classifier for face detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

# Load the image
image = cv2.imread("E:/2023_2024_Even/Pgm_Images/Fig2.jpg")

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

# Detect faces in the grayscale image
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
minSize=(30, 30))

# Draw rectangles around the detected faces
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Display the image with detected faces
cv2.imshow('Face Detection', image)

# Wait for a key press to close the window
cv2.waitKey(0)
cv2.destroyAllWindows()
```