EXPERIMENT NO. 4

Title:

To perform Edge Detection.

Objective:

- To understand the importance of edge detection in image processing.
- To implement and compare different edge detection techniques.
- To apply Canny, Sobel, and Laplacian edge detection techniques on images.
- To analyze the effectiveness of each method based on different parameters.

.

Brief Theory:

Edge detection is a crucial preprocessing step in image processing and computer vision applications. It aims to identify points in an image where the brightness changes sharply, often corresponding to object boundaries. The main edge detection techniques include:

1. **Canny Edge Detection:**
    - A multi-stage process involving Gaussian filtering, gradient computation, non-maximum suppression, and hysteresis thresholding.
    - It provides a robust detection mechanism with noise reduction.
2. **Sobel Edge Detection:**
    - Uses convolution masks to detect edges in horizontal and vertical directions.
    - The gradient magnitude is computed to highlight edge intensities.
3. **Laplacian Filter:**
    - A second-order derivative operator that detects edges based on zero-crossings.
    - More sensitive to noise, often requiring smoothing before application.

Steps: -

Step 1: Import Necessary Libraries

Use OpenCV, NumPy, and Matplotlib for image processing and visualization.

Code: -

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
```

Step 2: Load the Image

Read an input image in grayscale mode.

Check if the image is loaded successfully; if not, display an error message..

Code: -

```
# Load the image
    image = cv2.imread('sample.jpg', cv2.IMREAD_GRAYSCALE)

# Check if image is loaded
    if image is None:
        print("Error: Could not load image.")
        return
```

Step 3: Apply Edge Detection Methods:

- **Canny Edge Detection:** Use `cv2.Canny(image, lower_threshold, upper_threshold)`.
- **Sobel Edge Detection:** Compute horizontal (`cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)`) and vertical (`cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)`) edges.
- **Laplacian Edge Detection:** Apply `cv2.Laplacian(image, cv2.CV_64F)` to highlight intensity variations.

Code: -

```
# Apply edge detection methods
    edges_canny = cv2.Canny(image, lower_threshold, upper_threshold)
    edges_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
    edges_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
    edges_sobel = cv2.magnitude(edges_x, edges_y)
    edges_laplacian = cv2.Laplacian(image, cv2.CV_64F)
```

Step 4: Compute Magnitude for Sobel Edge Detection.

Combine horizontal and vertical Sobel results using `cv2.magnitude(edges_x, edges_y)`.

Code: -

```
edges_sobel = cv2.magnitude(edges_x, edges_y)
```

Step 5: Display Results:

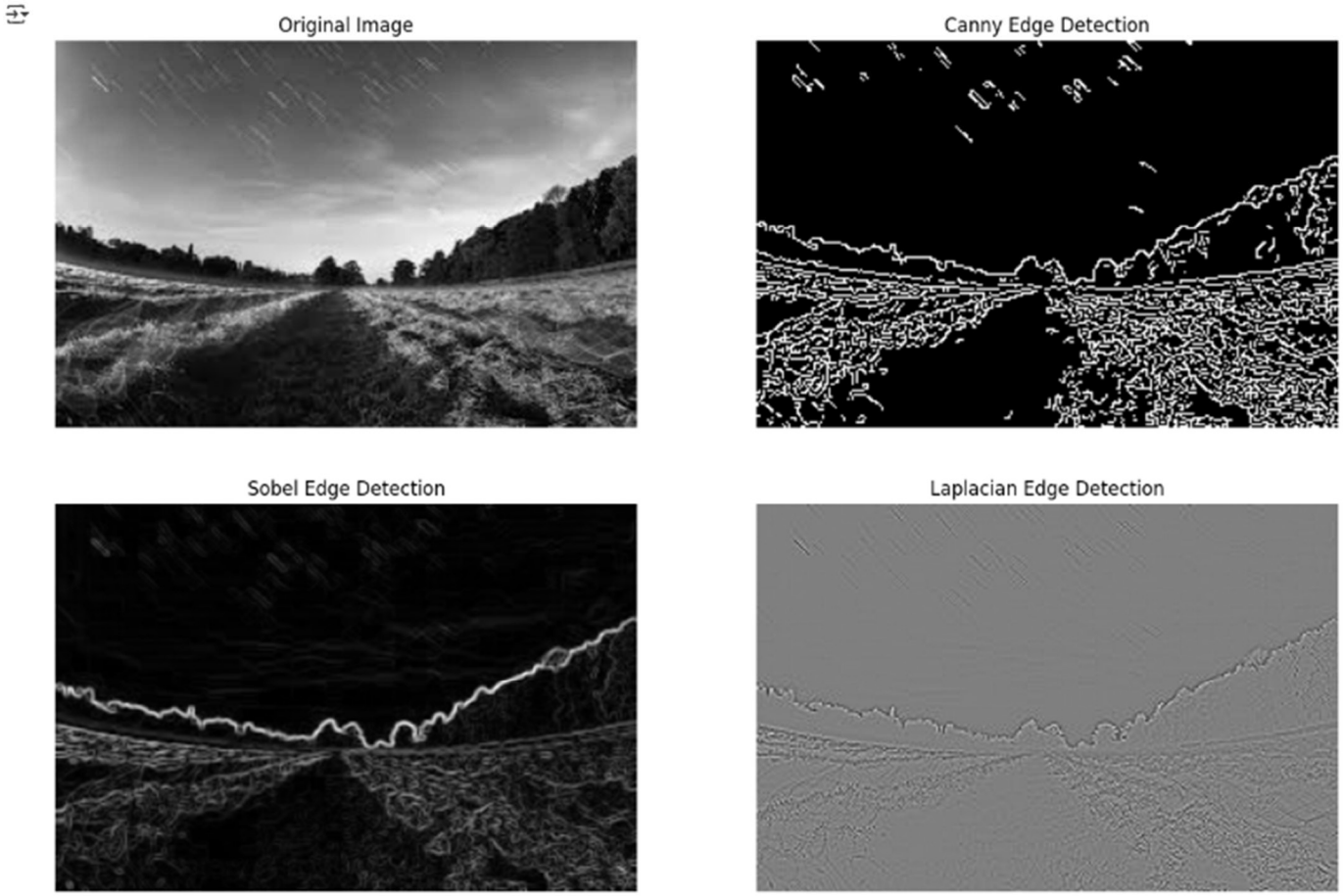
Code: -

- Use `matplotlib` to plot the original image and edge-detected outputs.
- Arrange results in a 2x2 subplot format and label each output.

```
 # Display the original and edge-detected images
using matplotlib
    plt.figure(figsize=(15,10))
    plt.subplot(2,2,1)
    plt.imshow(image, cmap='gray')
    plt.title('Original Image')
    plt.axis('off')

    plt.subplot(2,2,2)
    plt.imshow(edges_canny, cmap='gray')
    plt.title('Canny Edge Detection')
    plt.axis('off')
```

```
plt.subplot(2,2,3)
  plt.imshow(edges_sobel, cmap='gray')
  plt.title('Sobel Edge Detection')
  plt.axis('off')

  plt.subplot(2,2,4)
  plt.imshow(edges_laplacian, cmap='gray')
  plt.title('Laplacian Edge Detection')
  plt.axis('off')

  plt.show()
```

Output:



Conclusion:

By implementing Canny, Sobel, and Laplacian edge detection techniques, we observed their strengths and weaknesses. The Canny method provides the most refined edges with noise suppression. Sobel gives directional gradients, while Laplacian highlights rapid intensity changes but is more noise-sensitive. Choosing the best technique depends on the application requirements.

Practice Questions:

1. Implement Canny edge detection on a grayscale image and experiment with different threshold values.
2. Apply Sobel edge detection on an image and display separate horizontal and vertical edges.
3. Implement Laplacian filtering and compare its results with Canny and Sobel.
4. Combine Sobel and Canny outputs to enhance edge detection in an image.
5. Apply Gaussian blur before using Canny edge detection and analyze the effect.
6. Implement edge detection on a real-time webcam feed using OpenCV.

Expected Oral Questions

1. What is the purpose of edge detection in image processing?
2. Explain the steps involved in Canny edge detection.
3. How does Sobel edge detection work, and what are its advantages?
4. Why is the Laplacian filter more sensitive to noise?
5. How do threshold values affect the output of Canny edge detection?
6. Can you explain the difference between first-order and second-order derivative edge detection methods?

FAQs in Interviews

1. **What are the key differences between Canny, Sobel, and Laplacian edge detection?**
    - Canny is a multi-step process that includes noise reduction and thresholding, while Sobel computes gradients and Laplacian detects intensity variations.
2. **Why is Gaussian filtering used in Canny edge detection?**
    - It helps in noise reduction before computing gradients to avoid false edges.
3. **What is the role of non-maximum suppression in Canny edge detection?**
    - It removes weak edges that are not part of the strongest contours.
4. **How can we improve the robustness of edge detection in noisy images?**
    - By applying Gaussian blur before edge detection and using adaptive thresholding.
5. **What are some real-world applications of edge detection?**
    - Object detection, medical imaging, license plate recognition, and fingerprint analysis.
6. **How can edge detection be applied in deep learning models?**
    - Edge maps can be used as feature inputs for CNN-based object detection and segmentation models.