NATIONAL UNIVERSITY OF MODERN LANGUAGES ISLAMABAD

Computer Vision (Assignments)

Assignment: 01

Submitted to
Mr.Muhammad Waqar

Submitted By
Adeel Naeem
(BSAI-146)

Submission Date: May 20th ,2025

ASSIGMENT 1

1. Gaussian Filter:
This filter reduces image noise and detail by smoothing the image using a Gaussian kernel. It helps eliminate high-frequency content and is particularly effective at reducing Gaussian-type noise. However, it may also blur edges slightly.

2. Median Filter:
This nonlinear filter replaces each pixel's value with the median value of its neighboring pixels. It is especially effective for removing salt-and-pepper noise while preserving edges better than the Gaussian filter.

3. Sobel Edge Detection:
Sobel operator calculates the gradient of the image intensity at each pixel, emphasizing edges in horizontal and vertical directions. It uses convolution with Sobel kernels and highlights regions of high spatial frequency which correspond to edges.

4. Prewitt Edge Detection:
Like the Sobel operator, the Prewitt operator detects edges using convolution kernels, but with a simpler implementation. It’s computationally less intensive and suitable for detecting vertical and horizontal edges.

5. Canny Edge Detection:
A multi-stage edge detection algorithm that includes noise reduction, intensity gradient calculation, non-maximum suppression, and hysteresis thresholding. It provides highly accurate and thin edge maps, making it one of the most powerful edge detection techniques.

Comparative Analysis

Technique	Noise Reduction	Edge Preservation	Accuracy	Speed	Best For
Gaussian Filter	Good	Moderate	N/A	Fast	Smoothing natural noise
Median Filter	Excellent	Good	N/A	Moderate	Removing salt-and-pepper noise
Sobel Operator	N/A	Good	Moderate	Fast	Quick edge detection
Prewitt Operator	N/A	Moderate	Moderate	Fast	Basic edge detection
Canny Detector	Excellent	Excellent	High	Slower	Precise and detailed edge maps

- **Gaussian vs. Median:** Median filtering is better for salt-and-pepper noise, while Gaussian is smoother but can blur important features.
- **Sobel vs. Prewitt:** Both detect edges, but Sobel gives slightly stronger results due to its kernel design.

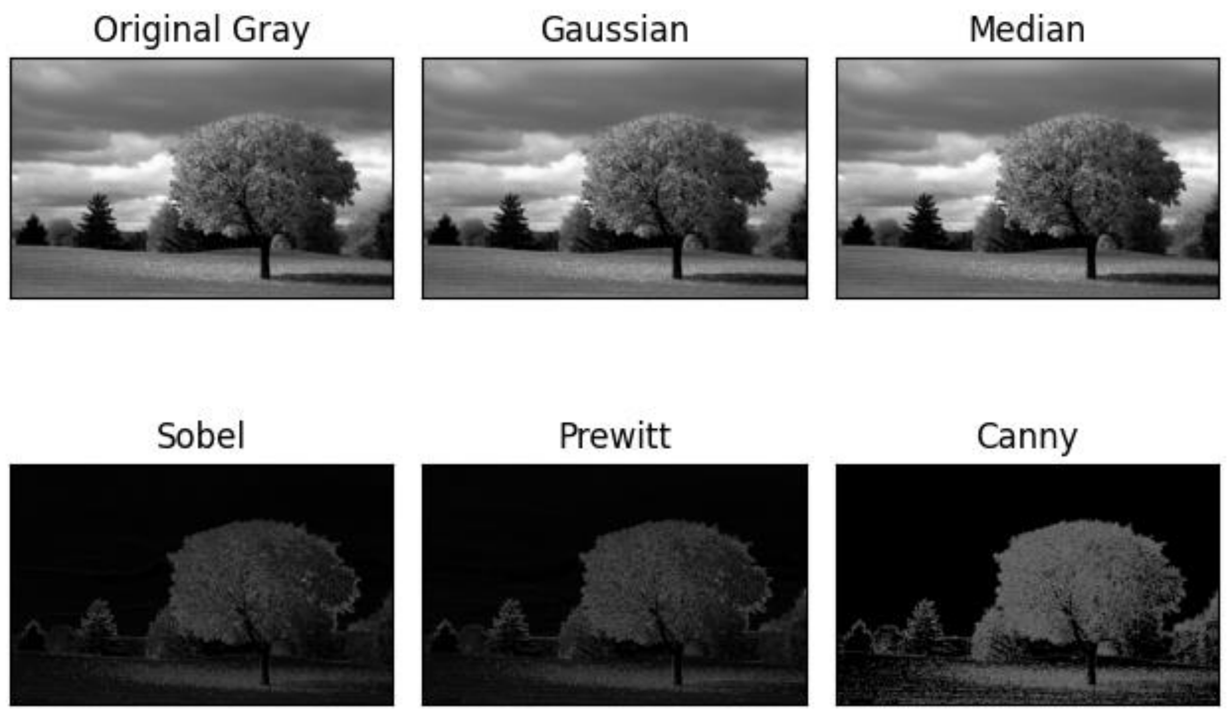
- **Canny:** The best technique in terms of precision, as it combines noise reduction and edge tracking, though it is computationally more expensive.

Code:

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load image (replace 'your_image.jpg' with your file)

img_color = cv2.imread(r'C:\Users\Adeel\Desktop\NLP\tree.jpg')

img_gray = cv2.cvtColor(img_color, cv2.COLOR_BGR2GRAY)

# Noise Reduction

gaussian = cv2.GaussianBlur(img_gray, (5, 5), 0)

median = cv2.medianBlur(img_gray, 5)

# Edge Detection

# Sobel

sobel_x = cv2.Sobel(img_gray, cv2.CV_64F, 1, 0, ksize=3)

sobel_y = cv2.Sobel(img_gray, cv2.CV_64F, 0, 1, ksize=3)

sobel = cv2.magnitude(sobel_x, sobel_y)

# Prewitt

kernelx = np.array([[1,0,-1],[1,0,-1],[1,0,-1]])

kernely = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])

prewitt_x = cv2.filter2D(img_gray, -1, kernelx)

prewitt_y = cv2.filter2D(img_gray, -1, kernely)

prewitt = cv2.magnitude(np.float32(prewitt_x), np.float32(prewitt_y))
```

```
# Canny

canny = cv2.Canny(img_gray, 100, 200)

# Show results

titles = ['Original Gray', 'Gaussian', 'Median', 'Sobel', 'Prewitt', 'Canny']

images = [img_gray, gaussian, median, sobel, prewitt, canny]

for i in range(6):

    plt.subplot(2, 3, i+1)

    plt.imshow(images[i], cmap='gray')

    plt.title(titles[i])

    plt.xticks([]), plt.yticks([])

plt.tight_layout()

plt.show()
```

Output:


Original Gray
Gaussian
Median
Sobel
Prewitt
Canny