Lab Report 5 : Edge Detection Using Prewitt, Sobel, Roberts, and Isotropic Operators

Course Title: Digital Image Processing Lab

Course Code: CSE-406

Date of Performance: September 22, 2024

Date of Submission: September 29, 2024

Submitted to:

Dr. Morium Akter
Professor

Dr. Md. Golam Moazzam
Professor

Department of Computer Science and Engineering
Jahangirnagar University

Submitted by:

Class Roll	Exam Roll	Name
370	202182	Rubayed All Islam

Experiment Name

a) Edge Detection Using Prewitt, Sobel, Roberts, and Isotropic Operators

Objectives

a) To implement and compare various edge detection techniques, including Prewitt, Sobel, Roberts, and Isotropic (Laplacian of Gaussian), for detecting edges in an image.

b) To implement and compare various edge detection techniques, including Prewitt, Sobel, Roberts, and Isotropic (Laplacian of Gaussian), for detecting edges in an image.

c) To analyze the effects of different edge detection operators on visualizing image details and boundaries.

Methodology

The experiment involves the implementation of four edge detection techniques: Prewitt, Sobel, Roberts, and Isotropic (Laplacian of Gaussian). The steps followed in the methodology are outlined below:

Image Acquisition

An image named `baby2.jpg` is used as the input for this experiment. The image is first read into both MATLAB and Python environments. If the image is in RGB format, it is converted to grayscale to simplify the edge detection process.

Edge Detection Techniques

The following edge detection methods are applied to the grayscale image:

- **Prewitt Edge Detection:** The Prewitt operator applies two 3x3 convolution kernels to calculate the horizontal and vertical gradients. The final edge map is obtained by combining the results of these gradients.
- **Sobel Edge Detection:** Similar to Prewitt, the Sobel operator applies horizontal and vertical kernels to compute edge gradients, with the main difference being that Sobel uses larger weight values to increase the edge's response.
- **Roberts Edge Detection:** The Roberts operator uses 2x2 convolution kernels to detect edges by calculating differences in pixel values diagonally. This is particularly effective for detecting sharp edges.
- **Isotropic Edge Detection (Laplacian of Gaussian):** This method applies a Gaussian filter to smooth the image before using a Laplacian operator to find areas of rapid intensity change. This helps in detecting edges that are not well-defined.

Experiment 1: Edge Detection Using Prewitt, Sobel, Roberts, and Isotropic Operators

MATLAB Code:

Listing 1: Edge_Detection.m

```
% Read the image
img = imread('baby2.jpg'); % Replace 'your_image.jpg' with your image file
% Check if the image is already grayscale
if size(img, 3) == 3
    grayImg = rgb2gray(img); % Convert to grayscale if it's a color image
else
    grayImg = img; % Use the image as is if it's already grayscale
end

% Prewitt Edge Detection
prewitt_x = fspecial('prewitt'); % Horizontal Prewitt filter
prewitt_y = prewitt_x'; % Vertical Prewitt filter
prewitt_edge = sqrt(imfilter(double(grayImg), prewitt_x).^2 + imfilter(double(grayImg), prewitt_y).^2);

% Sobel Edge Detection
sobel_x = fspecial('sobel'); % Horizontal Sobel filter
sobel_y = sobel_x'; % Vertical Sobel filter
sobel_edge = sqrt(imfilter(double(grayImg), sobel_x).^2 + imfilter(double(grayImg), sobel_y).^2);

% Roberts Edge Detection
roberts_x = [1 0; 0 -1]; % Roberts horizontal filter
roberts_y = [0 1; -1 0]; % Roberts vertical filter
roberts_edge = sqrt(imfilter(double(grayImg), roberts_x).^2 + imfilter(double(grayImg), roberts_y).^2);

% Isotropic Edge Detection (Laplacian of Gaussian)
h = fspecial('gaussian', [5 5], 1); % Gaussian filter
smoothedImg = imfilter(double(grayImg), h); % Smooth the image
laplacian = fspecial('laplacian', 0.5); % Laplacian filter
log_edge = imfilter(smoothedImg, laplacian);

% Display results
figure;
subplot(3,2,1); imshow(img); title('Real Image');
subplot(3,2,2); imshow(prewitt_edge, []); title('Prewitt Edge Detection');
subplot(3,2,3); imshow(sobel_edge, []); title('Sobel Edge Detection');
```

```
subplot(3,2,4); imshow(roberts_edge, []); title('Roberts Edge Detection');
subplot(3,2,5); imshow(log_edge, []); title('Laplacian of Gaussian Edge Detection');

% Adjust figure properties
sgtitle('Edge Detection Methods');
```

Python Code:

Listing 2: Edge_Detection.py

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read the image
img = cv2.imread('baby2.jpg')  # Replace with your image file
# Convert to RGB (OpenCV loads images in BGR format)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Check if the image is already grayscale
if len(img.shape) == 3:
    gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)  # Convert to grayscale if it's a color image
else:
    gray_img = img  # Use the image as is if it's already grayscale

# Prewitt Edge Detection
prewitt_x = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])
prewitt_y = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]])
prewitt_edge_x = cv2.filter2D(gray_img.astype(float), -1, prewitt_x)
prewitt_edge_y = cv2.filter2D(gray_img.astype(float), -1, prewitt_y)
prewitt_edge = np.sqrt(prewitt_edge_x**2 + prewitt_edge_y**2)

# Sobel Edge Detection
sobel_x = cv2.Sobel(gray_img.astype(float), cv2.CV_64F, 1, 0, ksize=5)
sobel_y = cv2.Sobel(gray_img.astype(float), cv2.CV_64F, 0, 1, ksize=5)
sobel_edge = np.sqrt(sobel_x**2 + sobel_y**2)

# Roberts Edge Detection
roberts_x = np.array([[1, 0], [0, -1]])
roberts_y = np.array([[0, 1], [-1, 0]])
roberts_edge_x = cv2.filter2D(gray_img.astype(float), -1, roberts_x)
```

```
roberts_edge_y = cv2.filter2D(gray_img.astype(float), -1,
    roberts_y)
roberts_edge = np.sqrt(roberts_edge_x**2 + roberts_edge_y**2)

# Isotropic Edge Detection (Laplacian of Gaussian)
gaussian_blur = cv2.GaussianBlur(gray_img.astype(float), (5, 5),
    1)
log_edge = cv2.Laplacian(gaussian_blur, cv2.CV_64F)

# Display results
plt.figure(figsize=(10, 10))
plt.subplot(3, 2, 1); plt.imshow(img.astype(np.uint8));
    plt.title('Real Image'); plt.axis('off')
plt.subplot(3, 2, 2); plt.imshow(prewitt_edge, cmap='gray');
    plt.title('Prewitt Edge Detection'); plt.axis('off')
plt.subplot(3, 2, 3); plt.imshow(sobel_edge, cmap='gray');
    plt.title('Sobel Edge Detection'); plt.axis('off')
plt.subplot(3, 2, 4); plt.imshow(roberts_edge, cmap='gray');
    plt.title('Roberts Edge Detection'); plt.axis('off')
plt.subplot(3, 2, 5); plt.imshow(log_edge, cmap='gray');
    plt.title('Laplacian of Gaussian Edge Detection');
    plt.axis('off')

# Adjust figure properties
plt.suptitle('Edge Detection Methods', fontsize=16)
plt.tight_layout()
plt.show()
```

Output:

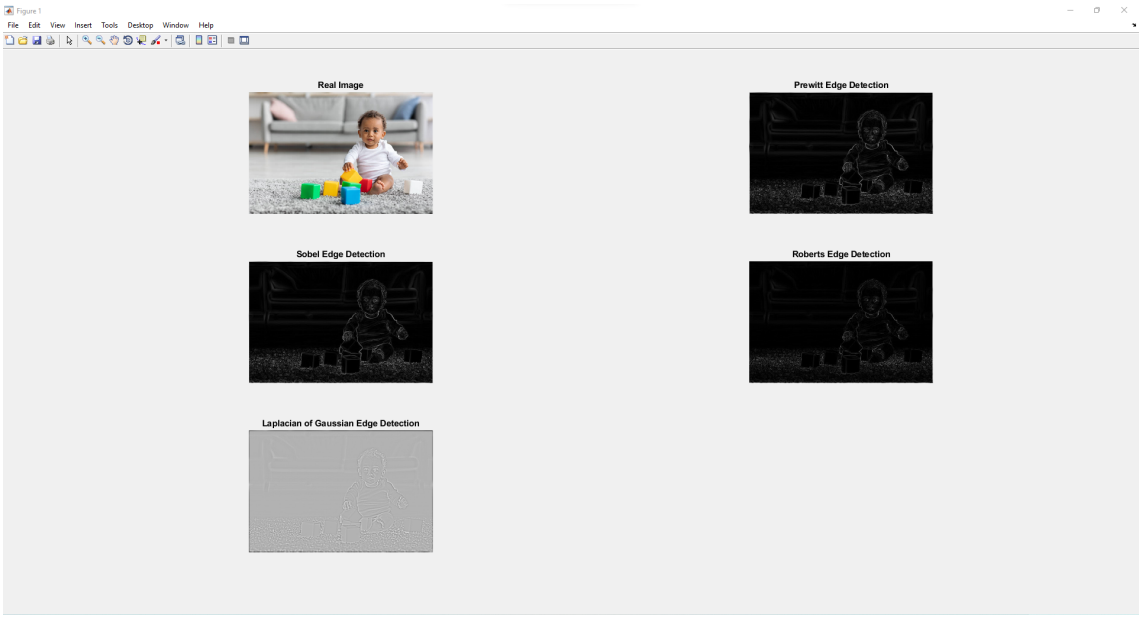

Figure 1: Edge Detection Output for Matlab Code

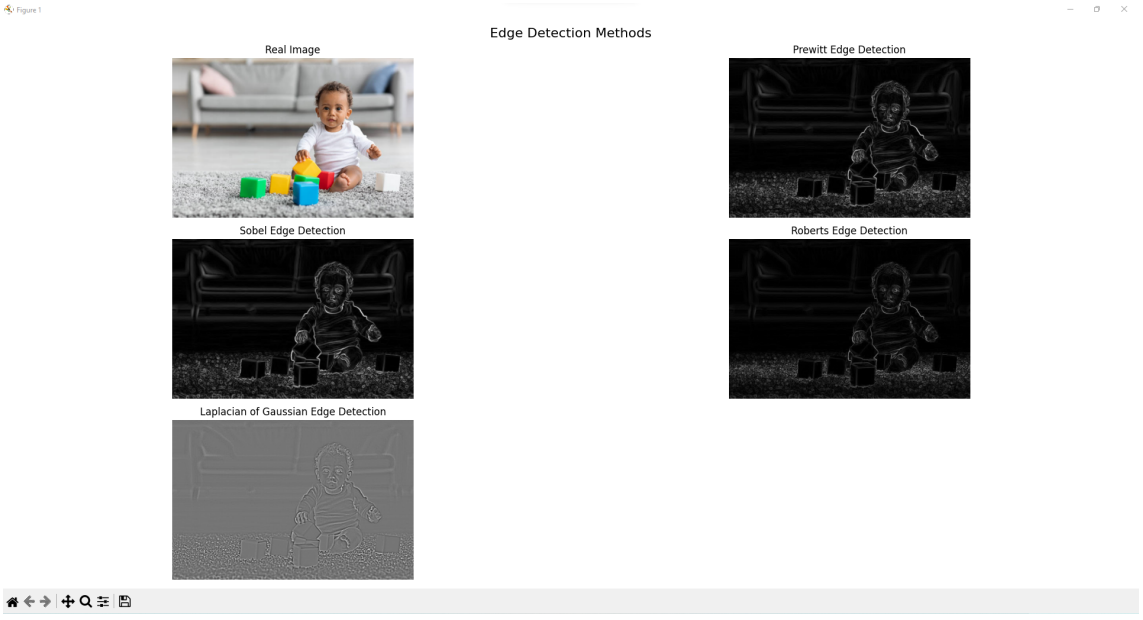

Figure 2: Edge Detection Output for Python Code

Results

- **Original Image**: The original image was a natural image of a baby used for edge detection. The RGB image was converted to grayscale for uniform edge detection.
- **Prewitt Edge Detection**: Both MATLAB and Python implementations produced similar results for the Prewitt operator. The edges in both horizontal and vertical directions are clearly identified, emphasizing sharp transitions between light and dark areas.
- **Sobel Edge Detection**: The Sobel operator produced more refined edges than the Prewitt operator due to larger convolution masks. The edges were smoother and more continuous, especially in regions with gradual intensity changes.
- **Roberts Edge Detection**: The Roberts operator provided a higher sensitivity to diagonal edges and sharper transitions. This method detected subtle edges that were not prominent in other methods. However, the output is slightly noisier due to its smaller kernel size.
- **Laplacian of Gaussian (Isotropic Edge Detection)**: This method performed well in detecting edges while maintaining smooth transitions between regions. The use of Gaussian smoothing prior to edge detection reduced noise, but the detected edges were slightly blurred compared to the other methods.

Conclusion

In this experiment, four edge detection techniques (Prewitt, Sobel, Roberts, and Laplacian of Gaussian) were implemented in MATLAB and Python. Each method has its strengths:

- **Prewitt**: Simple and effective but may miss finer details.
- **Sobel**: Better for smoother transitions, producing more accurate edges.
- **Roberts**: Sensitive to noise but effective for sharp diagonal edges.
- **Laplacian of Gaussian**: Reduces noise but slightly blurs edges.

The choice of method depends on the application, with Sobel and Roberts suited for sharp edges, while the Laplacian of Gaussian (LoG) method works well in noisy environments. Both MATLAB and Python produced similar results, with MATLAB being more intuitive, while Python offered flexibility through OpenCV.