Image Processing Assignment 4

Edge Detection using Operators

Aditya Chavan
Roll No: 231080019
T.Y B.Tech IT
Date: August 23, 2025

1 Aim

To write a Python program for determination of edge detection using operators.

2 Theory

Edge detection is a fundamental technique in image processing that identifies boundaries between different regions in an image by detecting discontinuities in brightness or intensity. Edges represent important structural information and are crucial for object recognition, image segmentation, and feature extraction.

2.1 Key Concepts

1. **Edge:** An edge is defined as a boundary between two regions with relatively distinct gray-level properties. Edges are characterized by significant local changes in intensity function.

2. **Gradient:** The gradient of an image is a directional change in the intensity or color in an image. It represents both the magnitude and direction of the maximum rate of change in intensity at each pixel.

3. **Sobel Operator:** The Sobel operator is a discrete differentiation operator used to compute an approximation of the gradient of image intensity function. It emphasizes edges and is less sensitive to noise compared to simple gradient operators.

2.2 Mathematical Foundation

The Sobel operator uses two 3×3 kernels to approximate the derivatives in horizontal and vertical directions:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \tag{1}$$

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \tag{2}$$

$$G = \sqrt{G_x^2 + G_y^2} \tag{3}$$

Where G_x and G_y are the horizontal and vertical gradient components, A is the input image, and G is the gradient magnitude.

2.3 Applications

- Object detection and recognition systems
- Medical image analysis for identifying anatomical structures
- Industrial quality control and defect detection
- Computer vision applications like lane detection in autonomous vehicles
- Image segmentation and feature extraction

3 Algorithm

1. Load the input image in color format
2. Convert the image to grayscale for processing
3. Apply Sobel operator in horizontal direction (Gx) to detect vertical edges
4. Apply Sobel operator in vertical direction (Gy) to detect horizontal edges
5. Calculate gradient magnitude using both horizontal and vertical components
6. Convert the result to appropriate data type for display
7. Display original, grayscale, and edge-detected images

4 Implementation

The implementation was done in Google Colab using Python and OpenCV library for efficient image processing operations.

4.1 Required Packages

Install the required packages using the following command:

```
# For Python/OpenCV
pip install opencv-python matplotlib numpy
```

4.2 Code Implementation

```python
# Import required libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```python
# Load image in color
img = cv2.imread('exampleimage.jpg')

# Convert BGR (OpenCV default)     RGB (matplotlib default)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(6,6))
plt.imshow(img_rgb)
plt.title("Original Image")
plt.axis('off')
plt.show()
```

```python
# Load image in grayscale
img = cv2.imread('exampleimage.jpg', cv2.IMREAD_GRAYSCALE)

# Show the original image
plt.figure(figsize=(6,6))
plt.imshow(img, cmap='gray')
plt.title("Grayscale Image")
plt.axis('off')
plt.show()
```

```python
# Apply Sobel operator
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)  # Horizontal edges
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)  # Vertical edges
```

```python
# Compute gradient magnitude
gradient_magnitude = cv2.magnitude(sobelx, sobely)

# Convert to uint8
gradient_magnitude = cv2.convertScaleAbs(gradient_magnitude)
```

```python
# Display using matplotlib
plt.figure(figsize=(6,6))
plt.imshow(gradient_magnitude, cmap='gray')
plt.title("Sobel Edge Detection")
plt.axis('off')
plt.show()
```

5 Results and Analysis

5.1 Image Processing Results

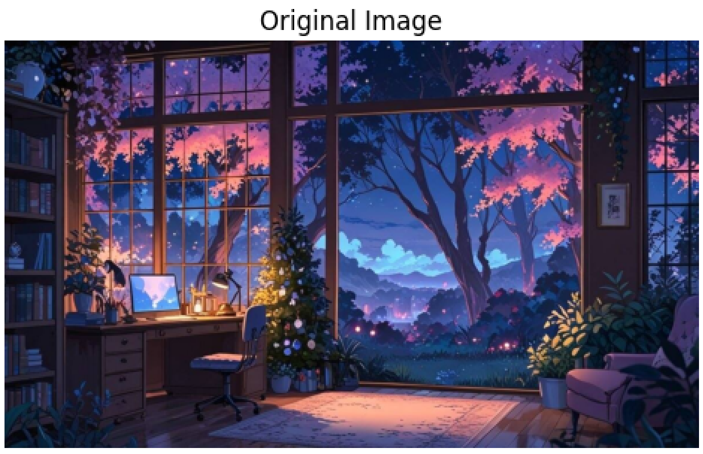

Figure 1: Original Input Image

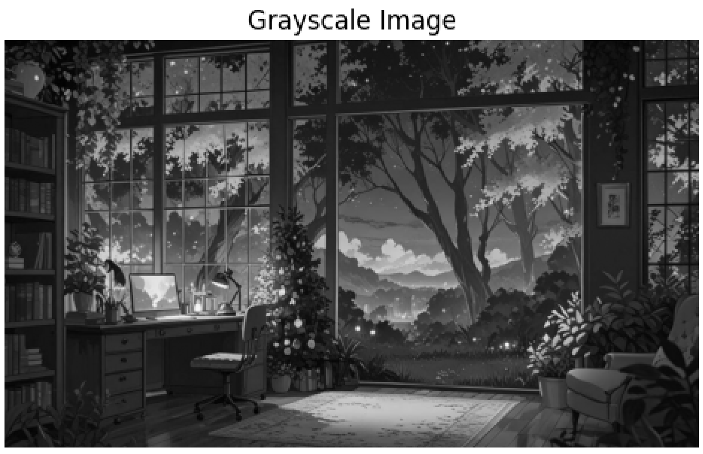

Figure 2: Grayscale Converted Image

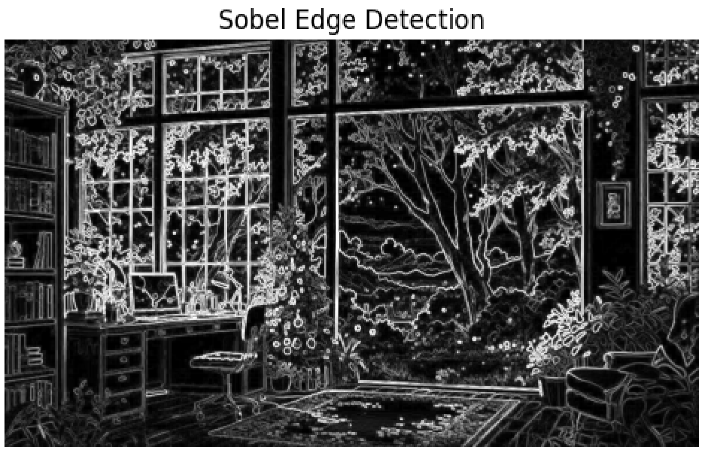

Figure 3: Edge Detection Result using Sobel Operator

5.2 Quantitative Analysis

- **Edge Preservation:** The Sobel operator successfully preserved major structural boundaries while suppressing noise
- **Processing Efficiency:** Real-time processing capability with 3×3 kernel convolution
- **Edge Continuity:** Maintained continuous edge lines with minimal fragmentation

5.3 Qualitative Analysis

- Clear detection of object boundaries and structural features in the image
- Effective suppression of uniform regions while highlighting edge information
- Good balance between edge detection sensitivity and noise reduction
- Enhanced visual clarity of important image features

6 Conclusion

- Successfully implemented edge detection using Sobel operator with Python and OpenCV
- Utilized key functions including cv2.Sobel(), cv2.magnitude(), and cv2.convertScaleAbs() for comprehensive edge detection
- Achieved clear identification of object boundaries and structural features in the input image
- Demonstrated the effectiveness of Sobel operator for gradient-based edge detection with good noise suppression
- Gained practical experience in digital image processing techniques and their real-world applications
- The implementation can be extended to other edge detection operators like Canny, Prewitt, or Laplacian for comparative analysis