**Lab 03: Edge Detection**

**Lab Outcomes**

After completing this lab, students will be able to:

1. Understand the mathematical foundation of edge detection using first and second derivatives.
2. Implement gradient-based edge detection using Sobel and Prewitt operators.
3. Apply second derivative methods such as the Laplacian operator.
4. Analyze the significance of edge detection in image processing.
5. Develop custom edge detection filters and evaluate their effectiveness.

**Introduction**

Edge detection is a key technique in image processing, used to identify the boundaries of objects in an image. Mathematically, an image can be represented as a function **F(x, y)** where:

* **First Derivative:** Measures the rate of intensity change (gradient) and is useful for detecting edges.
* **Second Derivative:** Measures the rate of change of the gradient, highlighting abrupt intensity changes.

**Common Edge Detection Methods:**

1. **Sobel Operator** – Computes the gradient in horizontal and vertical directions.
2. **Prewitt Operator** – Similar to Sobel but uses different weights.
3. **Laplacian Operator** – A second derivative method that enhances edges.
4. **Canny Edge Detector** – Uses gradient computation and non-maximum suppression for refined edge detection.

Python provides powerful libraries such as **OpenCV** and **NumPy** to implement these edge detection techniques.

**Solved Activities**

**Activity 1: Reading and Displaying an Image**

**Instructions:**

1. Import necessary libraries.
2. Read an image using OpenCV.
3. Convert it to grayscale and display the result.

import cv2

import matplotlib.pyplot as plt

# Load and display an image

image = cv2.imread('image.jpg', cv2.IMREAD\_GRAYSCALE)

plt.imshow(image, cmap='gray')

plt.axis('off')

plt.show()

**Activity 2: Computing the First Derivative Using Sobel Filter**

**Instructions:**

1. Compute the Sobel gradient in the x and y directions.
2. Display the gradient images.

sobel\_x = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=3)

sobel\_y = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=3)

plt.figure(figsize=(10,5))

plt.subplot(1,2,1)

plt.imshow(sobel\_x, cmap='gray')

plt.title('Sobel X')

plt.axis('off')

plt.subplot(1,2,2)

plt.imshow(sobel\_y, cmap='gray')

plt.title('Sobel Y')

plt.axis('off')

plt.show()

**Activity 3: Computing Edge Magnitude Using Gradient**

**Instructions:**

1. Compute the gradient magnitude.
2. Display the edge-detected image.

import numpy as np

gradient\_magnitude = np.sqrt(sobel\_x\*\*2 + sobel\_y\*\*2)

gradient\_magnitude = np.uint8(gradient\_magnitude)

plt.imshow(gradient\_magnitude, cmap='gray')

plt.title('Gradient Magnitude')

plt.axis('off')

plt.show()

**Activity 4: Applying Second Derivative Using Laplacian Operator**

**Instructions:**

1. Compute the Laplacian of the image.
2. Display the result.

laplacian = cv2.Laplacian(image, cv2.CV\_64F)

plt.imshow(laplacian, cmap='gray')

plt.title('Laplacian Edge Detection')

plt.axis('off')

plt.show()

**Activity 5: Applying Canny Edge Detection**

**Instructions:**

1. Apply Canny edge detection with specified thresholds.
2. Display the detected edges.

canny\_edges = cv2.Canny(image, 100, 200)

plt.imshow(canny\_edges, cmap='gray')

plt.title('Canny Edge Detection')

plt.axis('off')

plt.show()

**Activity 6: Smoothing Gradient Magnitude**

**Instructions:**

1. Apply a Gaussian filter to smooth the gradient magnitude.
2. Display the result.

smoothed\_gradient = cv2.GaussianBlur(gradient\_magnitude, (5,5), 0)

plt.imshow(smoothed\_gradient, cmap='gray')

plt.title('Smoothed Gradient Magnitude')

plt.axis('off')

plt.show()

**Activity 7: Applying Thresholding on Gradient Magnitude**

**Instructions:**

1. Apply a threshold to filter strong edges.
2. Display the result.

\_, thresholded = cv2.threshold(gradient\_magnitude, 50, 255, cv2.THRESH\_BINARY)

plt.imshow(thresholded, cmap='gray')

plt.title('Thresholded Gradient Magnitude')

plt.axis('off')

plt.show()

**Activity 8: Getting Orientation Based on Gradient Threshold**

**Instructions:**

1. Compute the gradient orientation.
2. Apply thresholding at minimum gradient magnitude.

orientation = np.arctan2(sobel\_y, sobel\_x)

plt.imshow(orientation, cmap='hsv')

plt.title('Gradient Orientation')

plt.axis('off')

plt.show()

**Activity 9: Before and After Non-Maximum Suppression**

**Instructions:**

1. Apply non-maximum suppression.
2. Compare results before and after suppression.

# Placeholder for non-maximum suppression implementation

plt.subplot(1,2,1)

plt.imshow(gradient\_magnitude, cmap='gray')

plt.title('Before Non-Max Suppression')

plt.axis('off')

# Assume suppressed\_img is the result after applying non-max suppression

plt.subplot(1,2,2)

plt.imshow(gradient\_magnitude, cmap='gray') # Replace with suppressed\_img

plt.title('After Non-Max Suppression')

plt.axis('off')

plt.show()

## ****Home Assignment****

### ****Task 1: Implement Edge Detection on a Noisy Image****

#### ****Scenario:**** Given a noisy image, apply appropriate filters before edge detection to improve results.

### ****Task 2: Compare Sobel and Prewitt Filters****

#### ****Scenario:**** Apply both filters on an image and analyze their differences.

### ****Task 3: Create a Custom Edge Detection Filter****

#### ****Scenario:**** Design a custom filter matrix and apply it for edge detection.

### ****Task 4: Apply Edge Detection to a Real-World Scene****

#### ****Scenario:**** Capture an image using a webcam and detect edges in real-time.

### ****Task 5: Edge Detection for Shape Recognition****

#### ****Scenario:**** Detect and outline geometric shapes in an image.

### ****Task 6: Implement Edge Detection using LoG (Laplacian of Gaussian)****

#### ****Scenario:**** Apply LoG filtering to detect edges in a blurred image.

### ****Task 7: Apply Multi-Scale Edge Detection****

#### ****Scenario:**** Use multiple kernel sizes to detect edges at different scales.

### ****Task 8: Implement Edge Detection for Medical Imaging****

#### ****Scenario:**** Apply edge detection to medical images (e.g., X-ray, MRI) to highlight structures.

### ****Task 9: Detect Moving Edges in a Video Stream****

#### ****Scenario:**** Implement real-time edge detection on a video feed.

### ****Task 10: Edge Detection for Optical Character Recognition (OCR)****

#### ****Scenario:**** Extract edges of text characters in an image to assist OCR preprocessing.