National University
Of Computer and Emerging Sciences

Department of Computer Science

Faculty Member: <> **Date: <>**
Semester: <>

Computer Vision

Lab 3: Detection of Edges

Introduction

This laboratory exercise will focus on implementing various edge detection operators using OpenCV and NumPy. You will apply convolution operations to detect edges with classical filters such as Roberts Cross, Prewitt, Sobel, and Laplacian. In addition, you will experiment with Blob Detection—a method for finding regions in an image that differ in properties such as brightness or color compared to their surroundings.

Objectives

- Understand and apply convolution for image processing.
- Implement custom convolution functions with different filters.
- Detect edges using Roberts Cross, Prewitt, Sobel, and Laplacian operators.
- Implement Canny EdgeDetection

Lab Conduct

- Respect faculty and peers through speech and actions
- The lab faculty will be available to assist the students. In case some aspect of the lab experiment is not understood, the students are advised to seek help from the faculty.
- In the tasks, there are commented lines such as #YOUR CODE STARTS HERE# where you have to provide the code. You must put the code between the #START and #END parts of these commented lines. Do NOT remove the commented lines.
- Use the tab key to provide the indentation in python.

Theory

OpenCV is a library that focuses on image processing and computer vision. An image is an array of colored square called pixels. Each pixel has a certain location in the array and color values in BGR format. By referring to the array indices, the individual pixels or a range of pixels can be accessed and modified. OpenCV provides many functions for centroid determination, color space changing, color range selection and perspective transformation.

A brief summary of the relevant keywords and functions in python is provided below. (For more details, check the slides for this lab)

print()	output text on console
input()	get input from user on console
range()	create a sequence of numbers
len()	gives the number of characters in a string
if	contains code that executes depending on a logical condition
else	connects with **if** and **elif**, executes when conditions are not met
elif	equivalent to **else if**

while	loops code as long as a condition is met
for	loops code through a sequence of items in an iterable object
break	exit loop immediately
continue	jump to the next iteration of the loop
def	used to define a function

Lab Task 1 –Robert Cross_________________________________

The Roberts Cross operator is one of the earliest edge detectors and uses two 2×2 convolution kernels to approximate the gradient diagonally. In this task, you will use the convolution function from LAB2 that performs 2D convolution on an image without using any built-in convolution functions. Then, apply the Roberts Cross operator by convolving the grayscale image with the given kernels. Compare the result with built in Roberts Cross. Display the resulting edge maps.

Horizontal Roberts Cross Kernel (Gx):

$$Gx = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Vertical Roberts Cross Kernel (Gy):

$$Gy = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Lab Task 2 –Prewitt

The Prewitt operator is a gradient-based method similar to Sobel but with simpler kernels. It uses 3×3 masks to estimate edges in both horizontal and vertical directions. In this task, create the Prewitt filters and apply them to your images using your implemented convolution function. Compare the results with built in Prewitt function.

Horizontal Prewitt Kernel (Gx):

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Vertical Prewitt Kernel (Gy):

$$Gy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Lab Task 3 – Sobel

The Sobel operator improves upon the Prewitt method by giving more weight to the central pixels of the 3×3 kernel, making it more robust to noise. In this task, you will apply the Sobel operator in both horizontal and vertical directions to detect edges.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Lab Task 4 – Laplacian

The Laplacian operator is a second-order derivative method that detects regions of rapid intensity change. In this task, you will smooth the image with your Gaussian function from Task 3 and then apply the Laplacian operator to highlight the edges. Use both common Laplacian kernels (4-neighbor and 8-neighbor) to see the difference in the resulting edge maps. Provide code, screenshots, and explanations of the output.

4-neighbor Kernel

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

8-neighbor Kernel

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Lab Task 5 –Canny Edge Detector

Implement Canny Edge detector from scratch and compare the result with built in Canny Edge detector.