



Department of Computer Science

Faculty Member: <>
Semester: <>

Date: <>

Computer Vision

Lab 5: Hough Transform

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 – Hough Transform for Line Detection in the m–c Domain

Implement the Hough Transform in the slope-intercept form of a line ($y = mx + c$).

Steps:

1. Load a binary edge image (e.g., from Canny detector).
2. Define parameter space: slope m (finite range, e.g. -1 to 1) and intercept c .
3. Build the accumulator array by iterating over edge points and voting in (m, c) space.
4. Extract peaks in the accumulator to identify dominant lines.
5. Overlay the detected lines on the original image.

Lab Task 2 – Hough Transform for Line Detection in the Polar Domain

Implement the Hough Transform in polar representation ($\rho = x \cos\theta + y \sin\theta$).

Steps:

1. Use the same edge image.
2. Define θ range (0 – 180°) and ρ range (based on image diagonal).
3. Construct a low- and high-resolution accumulator array (to test the effect of resolution).
4. Detect peaks in the accumulator to identify lines.
5. Draw detected lines on the image and compare with Task 1.



Lab Task 3 – Hough Transform for Circle Detection

Detect circles of varying radii using Hough Transform.

1. Use an image containing circles.
2. Define parameter space: center (a, b) and radius r .
3. Iterate over edge pixels and vote in (a, b, r) space.
4. Apply thresholding or peak detection in accumulator to identify circles.
5. Compare with OpenCV's built-in HoughCircles() function.