Homework 3

Computational Vision, Spring 2025

Due Date 3/14/2025

Total Points: 26

Please **follow the instructions** provided in the Programming Rules document on Piazza (under Resources). Note that failure to follow the rules will result in loss of points in the assignment. This is an individual assignment and all code should be your work. You can use code I have provided. If you want to use any external source code, please consult me first. Please submit via GradeScope. Make sure that your code compiles and runs correctly in the cslab machines before submitting on Gradescope. Note that we will check your code for plagiarism using Gradescope. For the programming assignment submit only your source files (.cc and .h) as well a Makefile, README.txt, and thresholds.txt file. For this homework you should submit h1.cc, h2.cc, h3.cc, h4.cc, image.h, image.cc, DisjSets.h (if you used it), DisjSets.cc (if you used it), Makefile, README.txt and thresholds.txt. Your thresholds.txt should contain two numbers (each in a new line), one for the threshold used for Program 2, and one for the one used for Program 4. The command "make all" in the terminal should generate all the executables listed below. Note that you should use the exact names of the executables as described below (h1, h2, h3, and h4). To visualize the input/output images you can use the free software gimp (www.gimp.org)

(TOTAL: 26 points)

Programming Assignment

Your task is to develop a vision system that recognizes lines in an image using the Hough Transform. Such a system can be used to automatically interpret engineering drawings etc. We will call it the "line finder." Three gray-level images are provided to you: **hough_simple_1.pgm, hough_simple_2.pgm** and **hough_complex_1.pgm**. It is enough that your line finder works on the "simple" images. If the results are good, you can try the "complex" image. Your program will be tested using not only these two images but test images we have taken. The task is divided into four parts, each corresponding to a program you need to write and submit. Each program must accept arguments in the format specified as

The task is divided into four parts, each corresponding to a program you need to write and submit. Each program must accept arguments in the format specified as

program_name {*1st argument*} {*2nd argument*} ... {*Nth argument*}.

Program 1 First you need to find the locations of edge points in the image. Write a program named **h1** that locates edges in a gray-level image and generates an "edge" image where the intensity at each point is proportional to edge magnitude:

h1 {*input gray–level image*} {*output gray–level edge image*}

For this you may either use the squared gradient operator or the Laplacian. Since the Laplacian requires you to find zero–crossings in the image, you should choose to use the squared gradient operator. The convolution masks proposed by Sobel should work reasonably well. Else, try your favorite masks. **(6 points)**

Program 2 Threshold the edge image so that you are left with only strong edges. You should have a program named **h2** that thresholds a gray-level image at a certain threshold value:

h2 {*input gray–level edge image*} {*input gray–level threshold*} {*output binary edge image*}

For that you could just reuse **p1.cc** from the previous programming assignment (rename your source file **h2.cc**).

Program 3 Next, you need to implement the Hough Transform for line detection. Write a program named **h3** that generates an image of the Hough Transform space of a given binary edge image:

h3 {*input binary edge image*} {*output gray-level Hough image*} {*output Hough–voting array*}

The brightness of each pixel (voting bin) in the output gray-level Hough image should be proportional to the number of votes it receives. The last output filename will store the votes of the voting array in a representation of your choice. Note that the gray-level output will be used just for visualization. As discussed in class, the equation $y = mx+c$ is not suitable as it requires the use of a huge accumulator array. So, use the line equation $x\cos\theta + y\sin\theta = \rho$. Note that θ must lie between 0 and π, and very large values of ρ correspond to lines that lie outside the image (use the diagonal as the maximum falue for ρ). You can use these constraints to limit the size of the accumulator array. The resolution of the array must be selected carefully. Low resolution will not give you sufficient accuracy in estimated parameters, and very high resolution will increase computations and reduce the number of votes in each cell (or bin). You may want to vote for small patches rather than points in the accumulator array. **(10 points)**

Program 4 Write a program named **h4** that finds lines in the image from its Hough Transform space, using a given threshold, and draw the detected lines on a copy of the original scene image:

h4 {*input original gray–level image*} {*input Hough–voting array*} {*input Hough threshold value*} {*output gray–level line image*}

Make sure that the lines you draw are clearly visible irrespective of pixel brightness values (dark or bright).

Straight lines in the image will produce blurred "areas of brightness" in the Hough space not single points, as predicted by the theory. The theoretical model uses the approximation that lines are infinitely long and infinitely thin - which is not exactly true in practice. The suggested approach to deal with this problem is to first threshold the Hough space image by cutting out all pixels below the given threshold value, and then segment the result into a number of "areas of brightness", each area presumably corresponding to a particular straight line (for that you could adapt the code developed in the previous assignment). Then in order to calculate the parameters of the line, you would have to find the "center" of its bright area. Notice that the approach we used when calculating positions of binary objects will probably not work, as different points in the Hough space will have different brightness (number of votes), and should have different contribution into the position of the "center". You may want to use a weighted average based on points intensities to locate the "center" (similar to how you would locate the center of mass of a non-homogenuous body **(10 points)**.