Image Processing and Computer Vision
CSCI 55700
Spring 2018
Project 3
Due: Sunday, April 8, 11:59pm

1 Introduction

In this project you will detect low level image features (edges) and detect larger structures formed by these low level features (lines). You will have three weeks to implement and test your algorithms and write a report for the project.

1.1 Corner Detection

Implement the Harris-Stephens corner detection method. Test your implementation on images. You will need to implement the thresholding and nonmaxima suppression to obtain your corners in the image. Matlab has an implementation of the corner detector called `corner()`. (see http://www.mathworks.com/help/images/ref/corner.html) You can compare the results of your implementation to that of Matlab function.

1.2 Edge detection

Implement the Canny edge detector. This will include implementing the directional Gaussian derivatives, non-maxima suppression, and hysteresis thresholding to find the true edges. You should attempt to automate the process as much as possible including the threshold selection. Compare the results you obtain using your implementation with those of a standard package. Matlab has a Canny edge detector implemented which you can call with various parameter values. The function is `edge()`. The parameters will include σ for the Gaussian and the two threshold values for the hysteresis thresholding. (see the link.) You can compare your results to that of the Matlab implementation.

1.3 Line detection and fitting

Write a program that will take the output edges detected in Section 1.2 and organize them into more abstract structures. The higher level structures you are to detect are straight lines. You will do this by using the Hough transform. Implement the Hough transform to detect straight lines from the edge images. In order to work with a reasonably clean transform space, implement the version of Hough transform that uses the gradient direction of the edges to build the Hough transform. The Hough transform for straight lines gives you the parameters of infinite straight lines. Now you have to decide which points in the image belong to which straight lines. This is called the back-projection. After you have identified the edge points that belong to one straight line, do an eigenvector line fitting to obtain the optimal straight line parameters. Use the normal equations of the line for building the Hough parameter space.

Some of the issues you have to deal with are (I am giving the issues here, but figuring out the solutions is up to you and part of the project):

1. How to get the parameters from the position and gradient direction information? For example, consider the effect of the sign of the contrast on which bins to update. It could be that there are object edges that lie on the same line but their contrasts are reversed (for example on a checkerboard). Do you consider them the same line or not?

2. What quantization to use for the parameter space?
3. How to deal with counts being split into adjacent bins. Your textbook hints at a solution.
4. How to perform the peak detection? Thresholding?
5. How to do the back projection? etc.

Matlab has a Hough transform function called `hough()`. (See the link.) Compare the results of your implementation to that of Matlab.

1.4 Data

Run your program on at least the following images:

- hinge.pnm
- hinges.pnm
- building.pnm
- keys.pnm
- pillsetc.pnm

If you can find some other images (particularly some with good lines in them), run your program on them also.

2 Bonus

If you have the time and want to do it, you can implement the line detection from the edge points using the RANSAC algorithm as well. If you do, include a discussion of how it compares with the Hough transform, in terms of performance and ease of implementation.

3 What to hand in

You will hand in a report for this project. It will contain the following:

1. A short introduction and a brief description of the theory and the algorithms you are implementing.
2. The source code of your program. Make sure you have the proper makefiles or shell scripts that we can compile your program and run and test it on tesla. Then zip the entire folder and submit.
3. The results of running your implementations on the images given above. You are welcome to test other images as well. The results in your report should include the following clearly labeled.
   - Input image.
   - Any intermediate results (e.g., gradient magnitude and direction, thresholded edge image, thinned edge image, Hough transform detected lines, accumulator arrays, detected peaks, etc, etc).
   - Results of your program with all the necessary information such as parameters, etc.

   Come up with a way to display or present your results. Remember that the results should be the straight lines your program detects. It would be nice if you can somehow present this in an image form (preferably superposed on the original image) so that the correctness of your results would be easy to judge.
4. A short discussion and any conclusions you may draw.

Please submit to Canvas your source code, etc as a single zipped file separately from your report document.

4 Grading Rubric

Report	20%
Corner detection	25%
Edge detection	25%
Line detection	15%
Fitting the line	15%
Bonus (RANSAC)	15%
Total	100%+15% bonus