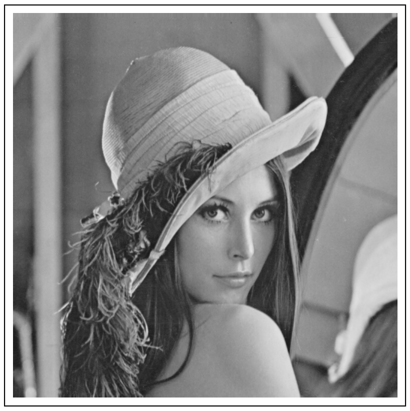
Canny Edge Detector

In this homework assignment, you are to implement the Canny edge detector [1, 2] in ImageJ. You should write an ImageJ PlugInFilter that prompts the user to enter the standard deviation of the Gaussian filter, a high threshold, and a low threshold. Optionally, you can have a check box that specifies whether the output should be a black and white image or a gray-scale image based on the edge strength. If you do not include this option in the dialog, you will have to select one of these two methods and hard code it into your plugin. The input and output to your filter should be a 8-bit grayscale images. Figures 1 and 2 show two example sets of input and output. You need to implement both non-maximal suppression and thresholding with hysteresis.


(a) Original image

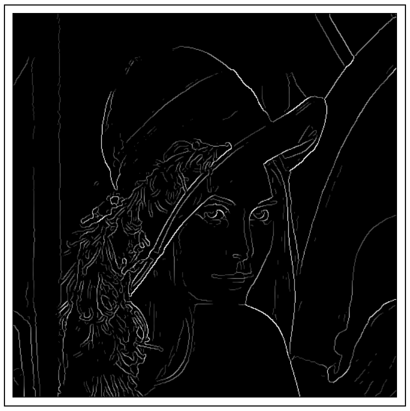
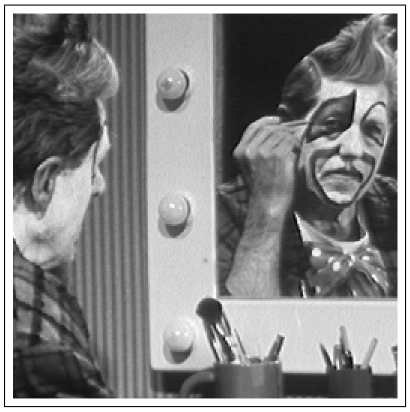

(b) Result of Canny edge detector

Figure 1: Example of the Canny edge detector with gray-scale edges.


(a) Original image

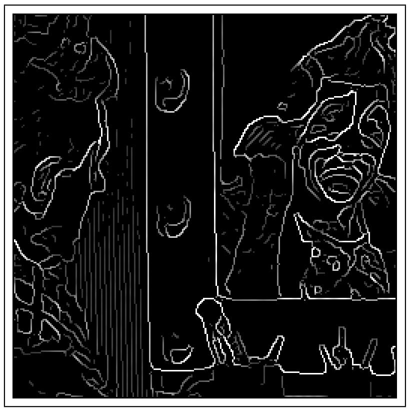

(b) Result of Canny edge detector

Figure 2: Second example of the Canny edge detector with gray-scale edges.

Non-maximum suppression

You should round the edge direction angle to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example). Given estimates of the image gradients, a search is then carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction. So, for example, if the rounded gradient angle is zero degrees (i.e. the edge is in the north-south direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes of its west and east neighbors. Perform similar computations if the rounded angle is 45, 90, and 135 degrees. This stage is referred to as non-maximum suppression and results in a set of edge points in the form of a binary image. These edges are sometimes referred to as "thin edges".

Notes

- You may use the Convolver class to apply filters.
- You will need to use both 8-bit and float ImageProcessor objects. To access pixel values for 8-bit ImageProcessor objects, use the methods putPixel(u,v,value) and getPixel(u,v). To access pixel values for float ImageProcessor objects, use the methods setf(u,v,value) and getf(u,v).
- To convert an 8-bit ImageProcessor object to a float ImageProcessor, use ImageProcessor image = orig.convertToFloat(); where orig is an ImageProcessor of type 8-bit.
- To visualize a float ImageProcessor object, you will need to copy it back to the input ImageProcessor object. Suppose you want to visualize the float ImageProcessor object image and your input 8-bit ImageProcessor object is called ip. You could use the two commands image.restMinAndMax(); ip.insert(image.convertToByte(true), 0, 0);
- You may use code from the DIPUJ book as long as you cite the book and the page that you got the code from.
- You may not any code from the internet.
- We will check your code for plagiarism.
- Don't forget to pick a method for handling boundary conditions.
- You may want to set edge values in binary images to be 0 and 255 instead of 0 and 1 for visualization purposes.
- I found the examples in the DIPUJ book very helpful for writing my solution.

Write a short report that describes your results. The document should include your input and output images. You should show results for different Gaussian standard deviations and explain how changing this value affected your results. Collect all of your files (java files and your report) in a single directory, compress the directory using zip, and submit it using the ICON Dropbox.

References

[1] F. J. Canny, "A Computational Approach to Edge Detection," *IEEE-PAMI*, vol. 8, no. 6, pp. 679–698, 1986.

[2] http://en.wikipedia.org/wiki/Canny_edge_detector.