CS 664 – Cornell University
Spring 2008

Assignment 1

Due: Tuesday Feb 26, 2008

In this assignment you will implement and experiment with some filtering and edge detection techniques.

You should submit a write-up of what you did for each problem, and submit a zip or tar file with your code (written in C/C++ with a windows executable or in Matlab) and example images. Include instructions on how to run your code for each part of the assignment.

Attention to issues such as how to accurately approximate image derivatives and how to efficiently compute Gaussian convolutions is important in this assignment. Your programs should take grey-level images in some standard format as an input.

1. Implement a program for Gaussian filtering that uses two one-dimensional passes. You should use an explicit Gaussian convolution for small kernels where that method is faster, and use the Wells box-filtering approximation discussed in class for larger kernels where that method is faster. Copies of Wells' paper will be handed out in class. In implementing box filtering you need to pay attention to numerical issues to ensure that the overall process does not brighten or darken the image (i.e., is equivalent to convolution with a kernel that integrates to 1). Include example output images smoothed with sigma=0.5, sigma=1, and sigma=3. To save gray images you will need to round to the nearest integer value.

2. This is a more open ended question concerning bilateral filtering, a technique that is described in http://scien.stanford.edu/class/psych221/projects/06/imagescaling/bilati.html (or see the original paper by Tomasi and Manduchi cited there). There is a Matlab implementation that you can consider available at http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=12191&objectType=file. If you run this implementation you will see that it is quite slow. Consider ways of speeding up this implementation, including reasonable approximation techniques. Explain your approach in your write-up and provide an implementation and some example output images.

3. Implement a program for edge detection by computing the zero-crossings of the Laplacian of Gaussian. Your program should perform either Gaussian or bilateral filtering as an option (don't round the output of the filtering to the nearest integer value before computing zero crossings). You should pay careful attention to how to estimate where a zero crossing occurs, and to detecting "thin" single-pixel wide edges at zero

crossings. Simply finding all zero crossings yields a lot of edges where there are small changes in the image. Design and implement a method for thresholding to keep only the "strong" edges. Measures of edge strength based on both the slope of the zero crossing and its magnitude generally perform the best. Explain your thresholding method and include example images of un-thresholded and thresholded outputs.