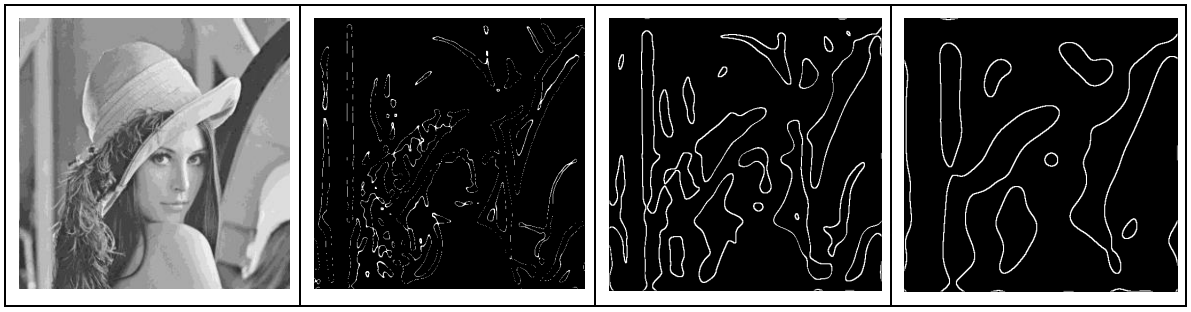
# Homework Assignment #1

## DUE: 5:00pm, Sunday January 20th (Electronic turnin required)

**IMPORTANT NOTE for ALL programming assignments**: You can use Matlab, Python, or C (C++) for programming assignments. Matlab is probably the easiest to learn and use. If you use any programming languages other than Matlab, you must include all files (headers, sources, libraries, Makefile, etc.) that are needed to properly compile and run your programs. You should also include a README file so that the reader knows how to compile and run your program. You must test your programs on a Linux machine in the ECI Lab or in CSIL to make sure that it compiles and runs. Basically,it is your responsibility to resolve any potential system incompatibility issues before you turn your codes in for grading.



The main purpose of this homework is to make sure that you have a valid CoE computer account, know how to use Matlab, Python, C++ etc. and the turnin program. This is a warm-up exercise that should not take much of your time. Do be warned the following assignments will be much harder.

We will implement the Marr-Hildreth edge operator in this programming assignment. First, you will need to write a function to create a pyramid of Gaussian filter masks. These masks should have progressively larger sigma of 3, 6, 12, 24, and 48 pixels. (Note that the discrete filters should cover at least 1 sigma of the continuous Gaussian function, which means that the filter size is not a constant and should increase with sigma.)Then, convolve the pyramid of the Gaussian masks with a Laplacian operator mask $(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2})$ to generate the LoG (Laplacian of a Gaussian) or the "Mexican Hat" operator at five different scales. (If you like, you can also implement the LoG operator directly. You will find the LoG equation in the note.) Finally, convolve the LoG operators with your input image and detect the zero-crossings (not zeros!) in the output as your edge points. Output five binary edge maps at five different scales.

Your main function takes one input: an input image. If the input image is a color image, convert it to a grayscale image first. Output your edge maps as binary images with the filename <old-filename>_n.bmp. (E.g., if the input image filename is “MyImage.jpg”, the output filenames should be “MyImage_n.bmp”, where $0 \le n < 5$.) The zeroth image corresponds to the smallest sigma filter and the 4th image corresponds to the largest sigma filter.

Sample test images can be found in http://www.cs.ucsb.edu/~cs181b/testimages/prog1/ (or follow the local image archive link from the class web page). Make sure that your programs work on images in that directory.