# CS 415 Machine Problem #2

## 1 Canny edge detector

The purpose of this MP is to give you a chance to play with pixels by implementing the Canny edge detector which consists of many smart ideas in digital image processing. You do not need to consider the speed of your implementation at this point. To implement it, what you need to do is to follow the steps below[1]. The due date of this assignment is 09/23/2019 (Wed).

1. *Gaussian Smoothing*. Use a Gaussian filter to smooth a given gray scale image. Your prototype could be `S = GaussianSmoothing(I, Kernel_size, Sigma)`.

2. *Calculating Image Gradient*. Once you get the smoothed version of the input image, you need to get the image gradient of it. You can use the Sobel operator to produce the magnitude and direction of your edge map, and optionally set a threshold to get rid of some noise. The prototype could be `Mag, Theta = ImageGradient(S)`.

3. *Suppressing Nonmaxima*. One important idea in Canny edge detector is finding local maxima of image gradient based on nonmaxima suppressing techniques. You can use the interpolation method discussed in the class. The prototype could be `Mag = NonmaximaSuppress(Mag, Theta)`.

4. *Thresholding and Edge Linking*. The nonmaxima suppressing techniques will give you quite good (thin) edges. The next step is to link the edges based on two thresholds. The high threshold produces strong edges, and the low threshold produces weak edges. You can set the low threshold as half of the high threshold. The idea of Canny detector is to first recursively link the strong edges. When a strong edge ends, keep growing the weak edges to fill the gaps between strong edges. We've discussed that in class. The prototype could be `E = EdgeLinking(Mag_weak, Mag_strong)`.

5. *Comparisons of Different Parameters*. You should play with different parameters, mainly, the Gaussian smoothing parameters and the thresholds in nonmaximal suppression, and, to see different outputs.

Two testing images (`test.png` and `lena_gray.png`) can be downloaded from our course website. Since this MP is difficult, I suggest you test and debug each of your smaller functions before putting them together. It is a good idea to check every intermediate result by plotting them out. Using the build-in convolution or correlation is allowed, but you need to specify all kernels by yourself.

## 2 What to turn in

Each individual student should turn in their own solution. What you need to turn in includes:

- your code in Python (recommended) or MATLAB;
- a short report.

[1] Steps 4 and 5 are optional, but finishing them will give you 20 points bonus.