**HW#3 (Canny edge detection)**

* + Write code to perform low-pass and high-pass filtering on an image.
    - Your code should accept a single scale parameter (sigma) as input, along with the name of the image file. (An optional third parameter should specify the name of the file to use for chamfer matching; if the third parameter is not provided, then no chamfer matching is performed.)  Convolve the image with an isotropic Gaussian kernel and with derivatives of the Gaussian along the x and y directions. You should use the separability property of the Gaussian and Gaussian derivative to speed computation, and you should construct the kernels automatically based on the sigma parameter. Do not worry about image borders -- the simplest solution is to simply set the border pixels in the convolution result to zero, but if you want to extend the image to improve the quality of the results, that is fine, too.
    - Implement the Canny edge detector.  There should be three steps to your code:  gradient estimation, non-maximum suppression, and thresholding with hysteresis (i.e., double-thresholding).    The gradient estimation has already been done in the previous step (Be sure to convolve with the derivative of a Gaussian rather than to compute finite differences of the smoothed image). Non-maximum suppression sets to zero any pixel that is not a local maximum along the direction of the gradient. For hysteresis thresholding, which is similar to floodfill, automatically compute the threshold values based upon image statistics.  Run your code on the following images:  [cat.pgm](https://cecas.clemson.edu/~stb/ece847/fall2012/cat.pgm) and [cameraman.pgm](https://cecas.clemson.edu/~stb/ece847/fall2012/cameraman.pgm).
    - \* Implement the chamfer distance algorithm using the Manhattan distance.  Convert the [cherrypepsi.jpg](https://cecas.clemson.edu/~stb/ece847/fall2012/cherrypepsi.jpg) image from color to grayscale before computing the Canny edges, then compute the minimum chamfer distance from each pixel to the nearest Canny edge pixel. Perform an exhaustive search (for simplicity, only consider locations for which the template is completely in bounds) for the best location of [cherrypepsi\_template.jpg](https://cecas.clemson.edu/~stb/ece847/fall2012/cherrypepsi_template.jpg) template .    This is done by computing, for each location, the sum of the distances from all pixels in the template (when centered at that location) to the nearest Canny edge pixel in the image. The best location is the one that yields the minimum value. (If these values are subtracted from a large constant, then they can be considered as a probability map.)
  + Your output should look like this:
    - Print the values of your 1D Gaussian and Gaussian derivative kernels.
    - One figure should show the original image. Another figure should show the smoothed image (after convolving with the Gaussian in both x and y). Four additional figures should show the gradient components in the x and y directions, along with the gradient magnitude and phase (angle).
    - Another figure should show the edges after non-maximum suppression, and another figure should show the final Canny result.
    - If chamfer matching is performed, then display in two separate figures the probability map, and the original image with the rectangle corresponding to the peak overlaid.
  + For this assignment, you may not use any Blepo functionality contained or prototyped in ImageAlgorithms.h (e.g., Chamfer), and you may not use the Gauss\*, Grad\*, Convolve, Correlate, Smooth, etc. functions prototyped in ImageOperations.h.
  + No report is due for this assignment.