CMPT882 Assignment 1: Edge detection and Texture Recognition
Due October 1

1. Implement the Canny edge detector (ref: Trucco and Verri, Ch. 4)
   - Smooth input image \( I \) by convolving with a Gaussian \( G = \exp\left(-\frac{x^2 + y^2}{\sigma^2}\right) \), \( J = I * G \)
   - Compute image derivatives \( J_x, J_y \)
   - Compute edge strength \( E_s = \sqrt{J_x^2 + J_y^2} \) and direction \( E_o = \arctan\frac{J_y}{J_x} \)
   - Perform non-maximum suppression, setting to zero values of \( E_s \) that are not larger than their neighbours along the direction perpendicular to the edge orientation in \( E_o \).
   - Implement hysteresis thresholding: given high threshold \( t_h \) and low threshold \( t_l \) (\( t_h \geq t_l \)), mark as edges all points with either:
     1. \( E_s \) larger than \( t_h \)
     2. \( E_s \) larger than \( t_l \) and connected to an edge point \( \hat{e} \) with \( E_s(\hat{e}) > t_h \) by other edge points with strength \( E_s > t_l \), in the direction of the edge at \( \hat{e} \)

Helpful MATLAB functions (aside from \texttt{edge(\ldots,'canny')}\texttt{)} include \texttt{filter2}, \texttt{gradient}, and \texttt{fspecial(‘gaussian’,\ldots)}. Experiment with running your edge detector on a couple of your favourite images, with different values for \( \sigma, t_h, \) and \( t_l \).

\textbf{Extra}: If you are interested in edge detection, perhaps for use in a course project, try downloading and running the “PB” code from http://www.cs.berkeley.edu/projects/vision/grouping/segbench/
Please see the course webpage for more details.

2. Perform texture recognition using histograms of textons
   - Download the training and test images from the course website
   - Construct a filterbank consisting of 18+18+3 filters (\( L_1 \) normalized), of three different types:
     1. Oriented odd-symmetric filters at 3 scales and 6 orientations, modeled as rotated copies of the horizontal filter \( f(x, y) = G'_{\sigma_1}(y)G_{\sigma_2}(x) \). Use a ratio of 3 for \( \sigma_2 : \sigma_1 \). Set the 3 scales to be a “half-octave” apart, i.e. \( \sigma_1^{i+1} = \sqrt{2}\sigma_1^i \).
     2. Oriented even-symmetric filters at 3 scales and 6 orientations, again rotated copies of a horizontal filter, this time \( f(x, y) = G''_{\sigma_1}(y)G_{\sigma_2}(x) \).
     3. Radially symmetric center-surround filters at 3 scales, each modeled as a “Difference of Gaussians” (DOG), \( f(x, y) = \exp\left(-\frac{x^2 + y^2}{\sigma_1^2}\right) - \exp\left(-\frac{x^2 + y^2}{\sigma_2^2}\right) \).
   - Compute textons by filtering the training images and running kmeans to cluster the output
   - Use these textons for texture recognition: for each test image, compute texton histogram and compare to histograms for training images using \( \chi^2 \) distance. Assign label of closest matching training image as label for test image.

Experiment with a few different values of \( K \), the number of textons computed using kmeans. Look at the texton maps computed for each image, and check that they are sensible.
There is a kmeans routine in the Netlab package, linked from the course website.