Homework #5

**Problem 1)** Implement an edge detector. Your result should have the form:

function [edgeim,edgepointlists]=edgedetect(im,filt,threshold)

The user should be able to pass to your function the image to be filtered (im), the filt (or filter parameters) you will use to produce candidate points, and a threshold value for thresholding the gradient magnitude.

Your function should return two things, an image that is black on edge points and white everywhere else, and an array of edgepoint lists, in which each list contains the indices of a set of points forming an unbroken edge chain.

You will probably benefit from knowing about Matlab’s structure capabilities, and the different kinds of indexing you can perform within images.

For example, consider the following 5x5 image.

```matlab
im = [ 7    28    42    28     7
       28   112   168   112    28
       42   168   252   168    42
       28   112   168   112    28
       7    28    42    28     7]
```

1) using `find()
indices = find(im>100)
% returns:
indices' = [ 7  8  9  12  13  14  17  18  19]
% such that
im(indices) = [112 168 112 168 252 168 112 168 112]
% are the values of the pixels that are greater than 100.
% these 1-D indices correspond to 2-D points according to the following ordering
   1   6  11   16   21
   2   7  12   17   22
   3   8  13   18   23
   4   9  14   19   24
   5  10  15   20   25
% You see then that find has selected the middle 3x3 block.
% Matlab calls this 1-D indexing, index
% But it calls 2-D indexing (e.g. im(3,5)) subscripts.
% to convert to subscripts, use ind2sub()
for instance:
\[ I,J = \text{ind2sub(size(im),indices)}; \]

So that im(I(1),J(1)) is the first pixel, im(I(2),J(2)) is the second pixel, etc.

In general it is more convenient to work with indices, because you quickly grab all the indices with some properties, and quickly return the values at those indices, and you can readily store those pixel locations in a 1-D list.

For instance, chain = \[ 1 \ 2 \ 3 \ 4 \ 9 \ 14 \ 13 \ 12 \ 11 \] represents a U in the image.

To visualize this: imnew = zeros(5,5); imnew(chain) = 256; image(imnew)

Let a second chain =\[21 \ 22 \ 23 \ 24 \ 25 \ 20 \ 15\];

Then we can store both chains in a matlab structure:

edgepointlists(1).chain = \[ 1 \ 2 \ 3 \ 4 \ 9 \ 14 \ 13 \ 12 \ 11 \] \% length 9
edgepointlists(2).chain = \[21 \ 22 \ 23 \ 24 \ 25 \ 20 \ 15\] \% length 7

Structure fields are easy to add:
edgepointlists(1).originalvalues = im( edgepointlists(1).chain );
( matlab takes care of all the memory management for you.)

Finally, structures can be quickly accessed for their contents if all the indices are stored in row format:
Alledgepointindices = [edgepointlists( : ).chain]

Evaluating your edge detector

Use imread() to load housewtree.jpg. run your edge detector at multiple scales and thresholds. How well does your detector find the tree boundary? Flower boundaries? House boundaries?

Problem 2) Segmentation using muliple filters and Bayesian classification

load housewtree_test.jpg and housewtree.jpg. Run 3 or 4 filters of different scale and orientation across the images (you choose the filters). The goal is to identify regions of homogeneous texture in
housewtree.jpg and compute the marginal histograms for each filter output you use on this training data. The goal is to use these histograms to segment housewtree_test.jpg. In particular, use grabpixels() to grab pixels from 3 different regions of the image: 1) flowers; 2) roof; 3) everywhere else. For each filter, determine the histogram for each of the 3 classes H(filter)_class (use \[N=\text{hist(imvalues,bincenters)}\]). Normalize N into a probability vector \(p(filter)_\text{class} = \frac{N}{\text{sum}(N)}\). Determine the prior probability of each class. Segment the test image by filtering it, and computing the optimal Bayesian decision.

**Problem 3 Bayes’ Theorem**
Suppose we have a box containing 8 apples and 4 oranges, and we have a second box containing 10 apples and 2 oranges. One of the boxes is chosen at random (with equal probability) and an item is selected from the box and found to be an apple. Use Bayes’ theorem to find the probability that the apple came from the first box.

**Problem 4 Minimum-risk decision criterion**

Recall that optimal decisions can be made by minimizing the risk:
Risk for each class:
\[ R_k = \sum_{j=1}^{c} L_{kj} \int_{\mathbb{R}_j} p(x \mid C_k) dx \]

Average Risk:
\[ R = \sum_{j=1}^{c} R_k P(C_k) \]

The risk \(R\) is minimized if the integrand is minimized at each point \(x\), that is if the decision regions \(P\) are chosen such that:
\[ x \in \mathbb{R}_j \text{ when } \sum_{k=1}^{c} L_{kj} p(x \mid C_k) P(C_k) < \sum_{k=1}^{c} L_{ki} p(x \mid C_k) P(C_k) \text{ for all } i \neq j \]

Verify that this minimum-risk decision criterion reduces to the decision rule:
\[ p(x \mid C_k) P(C_k) < p(x \mid C_j) P(C_j) \text{ for all } j \neq k \]

for minimizing the probability of the misclassification when the loss matrix is given by:
\[ L_{kj} = 1 - \delta_{kj} \]

where delta is the Kronecker delta function which 1 when \(k=j\) and 0 otherwise.