ECE 172a	Machine Vision	Pamela Cosman	2/27/12

HOMEWORK 5: Due Friday March 9 by start of class

1. Segmentation by DT and watershed algorithm:
In Matlab, read in the oranges image with the command: I = imread('oranges.tif');
We want to segment this image so that each orange is separately segmented, including ones that are touching. To begin, we need to convert to grayscale and binarize:

```
Ig = rgb2gray(I);
BW = Ig > 70;
```

(a) Write two functions that will implement the forward and backward Chamfer algorithm. They should take as inputs an image and a 3×3 distance mask.
For example, the functions should be called with the following command lines:

```
out1 = chamferF(BW*1000,Fmask);
out2 = chamferB(out1,Bmask);
```

where

```
Fmask = [4 3 4; 3 0 0; 0 0 0]
Bmask = [0 0 0; 0 0 3; 4 3 4]
```

Explain why we have to call the chamferF routine with `BW*1000` but chamferB gets called directly with out1. Look at out2 using imshow. You may need to rescale:

```
imshow(out2/max(out2(:)));
```

Comment on how it looks. Is it exactly what you expected?
(b) We will segment using Matlab's built-in watershed function. For example `L = watershed(BW)` finds the watershed lines of binary image `BW`. L is called the *label matrix* and it contains positive integers that indicates the label of each segment corresponding to the locations. Label 0 indicates the location of watershed lines. Try

```
L = watershed(out2);
imshow(L == 0);
```

The watershed lines do not correspond to the expected segmentation boundaries. Figure out why not, and see what simple change you need to make in order to get the expected segmentation boundaries.
(c) Plot the correct segmentation result by

```
imagesc(L);
```

Now if you like you can show the k^{th} segment using

```
seg = I;
seg = I .* repmat(uint8((L == k)), [1 1 3]);
imshow(seg);
```

Do you have a fully clean segmentation boundary? If not, explain where the errors come from.

2. Bilevel thresholding:
Read in the image `cell` with the command

```
cell = imread('cell.tif');
```

and look at it. The blobs are cell nuclei which we would like to segment very cleanly, because we are supposing that after the segmentation we will need to make some precise measurements on them. Read in the "correct" segmentation results with the command

```
truth = imread('truth.tif');
```

In Matlab, one binary segmentation map (`map`) can be compared against another (`truth`) by counting separately the number of pixels that are in error either way:

```
sum(sum(map > truth))
sum(sum(truth > map))
```

or by counting the total number of pixels in error using the exclusive OR function:

```
sum(sum(xor(map,truth)))
```

Note before error calculation, you need to make sure `map` and `truth` are BOTH binary images.
We will use the latter as a simple measure of how well the segmentation is doing. Write a routine that steps through a range of thresholds, compares against the binary truth map, and finds a good threshold. How well can you do with segmentation by simple global bilevel thresholding? What is the best global threshold, and what is the lowest error you can get to the truth image?

3. Histogram methods for texture segmentation:
The cell image from the previous part has been corrupted in its background region with noise. The noise values for each pixel are independent and identically distributed with a distribution that is approximately normal. The mean value of the noise matches the mean value of the cells. You can read in the new version as follows:

```
badcell = imread('badcell.tif');
```

In each case, we would like to transform the image into a texture image, where at each pixel the grayscale image is some measure of the local texture. Then, we will use global bilevel thresholding on this texture image, and compare against the truth image.
See how well you can do just using a simple histogram measure. If you type `help stats` you will find that the Statistics Toolbox in Matlab lists a number of different descriptive statistics which can be computed from a histogram.

The measure at a pixel P should be something computed from the histogram of pixel values from a window centered on P. Generate at least two different texture images by varying either the window size or which descriptive measure is computed over the window. In each case, find a good threshold and compute the misclassification error. Interpret your results.

4. Hurst Operator:
Next see what the Hurst operator can do. You are will need to write a routine which computes both the slopes and intercepts for the Hurst transform over some window size. For example, for a 5×5 window, we would have pixels in 5 different distance classes:

```
e d c d e
d b a b d
c a X a c
d b a b d
e d c d e
```

The distances from X to pixels a, b, c, d, and e are 1, $\sqrt{2}$, 2, $\sqrt{5}$, $2\sqrt{2}$. The log values of these are x = [0; 0.3466; 0.6931; 0.8047; 1.0397]. If the vector y of length 5 denotes the log(range) of the pixels up to and including each successive distance class, then the points going into the Hurst plot are

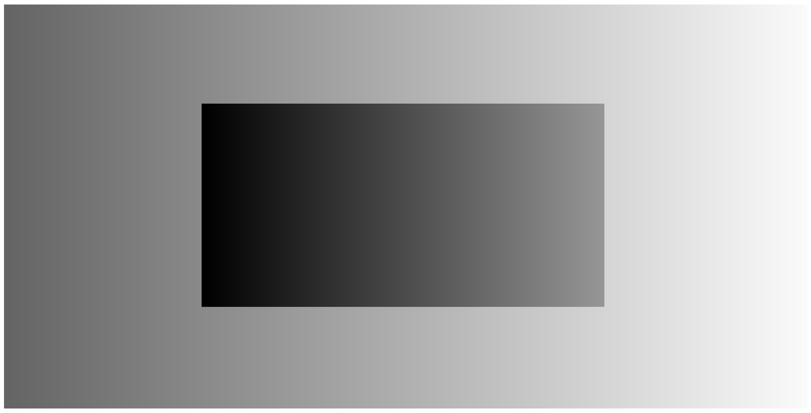
```
plot(x,y)
```

and we need to do a straight line fit to these points. The least squares fit is accomplished in Matlab as follows:

```
X = [1 0; 1 0.3466; 1 0.6931; 1 0.8047; 1 1.0397];
[Q R]=qr(X);
c = R\Q'*y;
slope = c(2);
intercept = c(1);
```

In the computation of the least squares fit, the matrix X is called the design matrix; the first column consists of all 1's, and the second column has the log(distance) for the successively larger distance classes. Try a couple of different window sizes. That is, you will put more or fewer distance classes in to your computation. Again, find good thresholds for these texture measures, report your results, and explain why the results are the way they are. Does the computation over a larger window size give you results that are better or worse?

5. Segmentation:
This is a pencil-and-paper problem of the type that you could be asked on the final. You are given an image which looks like this:



The inner rectangle is shaded uniformly from a value of 0 on the left to a value of 150 on the right. The outer rectangle is shaded uniformly from a value of 100 on the left to a value of 250 on the right. We would like to segment this image into the two constituent rectangles.

1. Is it possible to accomplish this segmentation using global bilevel thresholding? If yes, then describe exactly how it would be done. If it is not possible, then explain why not.
2. Is it possible to accomplish this segmentation using the method described in class called "double thresholding"? If yes, then describe exactly how it would be done. If it is not possible, then explain why not.
3. Is it possible to accomplish this segmentation using pixel aggregation? If yes, then describe exactly how it would work– explain what would be your choice of starting pixel(s), joining criteria, order of joining, and stopping rule. If it is not possible, then explain why not.
4. Is it possible to accomplish this segmentation using quadtree splitting followed by one final merge step? If yes, then describe exactly how it would be done. If it is not possible, then explain why not.