ECE 172	Digital Image Processing	Pamela Cosman	1/8/13

HOMEWORK 1

Due Friday January 18 in class

In Matlab, if you type “help images” you will see the commands of the Image Processing Toolbox. Within this long list, the commands under the header “Morphological operations (binary images)” are the binary image operators useful in this homework set. One of these is bwmorph; type `help bwmorph` to see a list of the various operations which it provides.
Note: if you try to do some operation on a variable named, say, im, and Matlab tells you that it’s illegal to do that operation on a variable of type uint8, then try it with double(im), which converts the variable im to type double.
For all of the Matlab problems, give your Matlab commands as well as showing your numerical and image results.

1. **Boundary extraction:**

   Read the image `eyechart` into Matlab and look at it with the commands

   ```
   eye = imread('eyechart.tif','tif');
   imshow(eye)
   ```

   The array eye is of datatype uint8, so it is integer-valued, but the pixels take on only two values: 0 and 255. To put it in the form which matlab wants for binary image processing, use: `bw = 1 - round(double(eye)/255);`
   Now it has values of 0 and 1, and the object pixels (letters) have value 1. Find 3 ways to extract the boundaries of the letters, either with individual commands or with combinations of operations.

2. **Adding salt-and-pepper noise:**

   Write a routine which takes as input arguments a binary image and a percentage p of pixels to be randomly flipped. Flipping a pixel means that if it is white it will be made black, and if it is black it will be made white. This is sometimes called salt-and-pepper noise. Use your routine to generate 3 different versions of the binary eyechart: one with low levels of noise ($p = 1\%$), one with a medium level of noise ($p = 5\%$), and one with lots of noise ($p = 20\%$).

3. **Isolated pixel cleaning:**

   We would like to clean up these noisy versions. Write a function that does isolated pixel cleaning. That is, a black pixel entirely surrounded by white pixels should be flipped, as should a white pixel entirely surrounded by black pixels. Apply this to the three noisy versions. How well does it do?

4. **Cleaning using opening and closing:**

   Opening followed by closing can also be used for cleaning. Try it on the three noisy versions. How does it compare to the isolated pixel cleaning? Are the results different if we do closing first and then opening? Why or why not?

5. **Thinning:**

   Write a function to perform thinning based on the sequence of 8 structuring elements discussed in class. A single call to your thin function should do thinning once with each of the 8 elements. How many times do you have to call your routine before you reach convergence? Include the final thinned image in your results. Now use the thinning operation provided with bwmorph: how many times do you have to call it to reach convergence? How does its final thinned output compare?

6. **Filling an outline:**

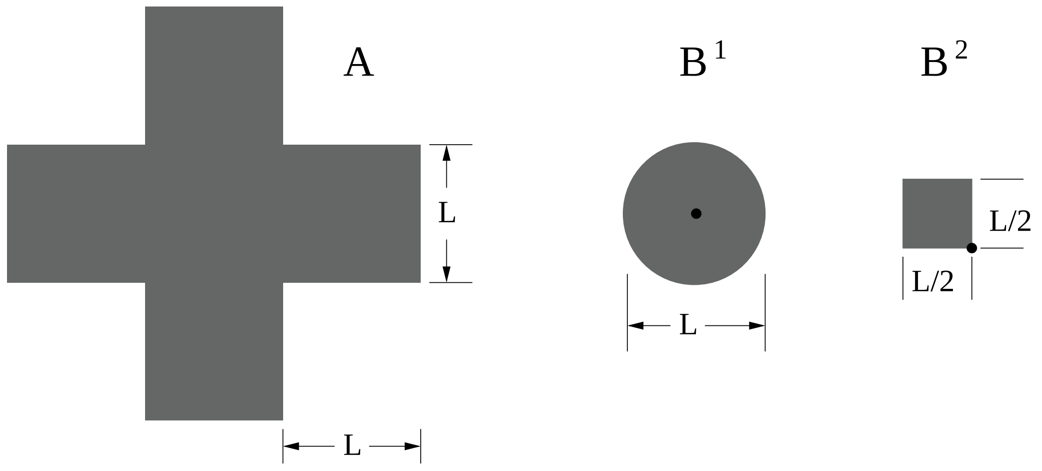
   Read in the binary image Euler.tif. Use any combination of bwmorph operations and your own routines to clean up the noise, make the letter outlines look nice, and fill in the letter "e" with black.

7. **Binary Erosion and Dilation:**

   This is a pencil-and-paper problem, no Matlab. Let A denote the set shown below left. There are two structuring elements shown; the black dots denote the origin of each structuring element.

   In this problem, do *not* think of the sets A, B^1, and B^2 as being discrete (sets of pixels). Rather, they are meant to be continuous sets in R^2.

   Sketch the result of the following morphological operation: $(A \ominus B^1) \oplus B^2$


A
L
L
B 1
L
B 2
L/2
L/2