CSE 4/573 Fall 2010 v2 Sep 10 2010

Homework assignment #1 due Wenesday September 22

1. **Bounding box construction:** submit an m-file BoundingBoxes.m with first line

```
function BB = BoundingBoxes(InIm)
```

where

InIm: A uint8 binary type logical input image containing a one or more blobs (black object pixels = 0, white background pixels = 1);

BB: an 2x2xN array, where N is the number of 8-connected blobs in the image InIm. The (1,1,n) entry of BB is the x-coord and (2,1,n) is the y-coord of the upper left hand corner pixel of the bounding box of blob (8-connected region) n in the image; the (1,2,n) entry is the x-coord and the (2,2,n) entry is the y-coord of the lower right hand corner pixel of the bounding box.

The bounding box for a blob is defined to be the smallest rectangle such that no blob pixels lie outside the rectangle. Use SIZE to determine the number of rows and columns in InIm. The blobs should be ordered in BB in size order, ie. (:,:,1) should contain the bounding box of the biggest blob (measured by area of its bounding box), while (:,:,N) should be the smallest. Where there are ties, use any order.

2. Submit an m-file Equidistant.m which finds all the pixels that are equidistant from three given blobs. Specifically, the first line of the m-file should be

```
function OutImage = Equidistant(InImage,DistMeas)
```

where

InImage: uint8 input image containing exactly 3 connected blobs. The pixels of one blob all have value 1, the second all have value 2, the third 3, and the background pixels all have value 0.

DistMeas: a string, either 'cityblock' or 'chessboard'.

OutImage: double normalized image in which the original 3 blobs all have greylevel 1.0, the background has greylevel 0.0, and the pixels that are equidistant from the three blobs all have greylevel 0.5.

Code the chamfer algorithm described on page 28 (20-21 in Ed. 3) of the text and use the result of the chamfer algorithm in your solution to this problem. If DistMeas='cityblock', the pixels you find should be equidistant in the sense that their cityblock distances from the 3 blobs are equal, and similarly with DistMeas='chessboard'. Note that for certain InImages, there may be no pixels that are equidistant.

3. **Exact run-length code for binary image:** submit an m-file BinRLC.m with first line

```
function BRLC=BinRLC(InIm);
```

where

InIm: A uint8 binary type logical image (black object pixels = 0);

BRLC: The exact RLC for InIm.

Your answer should be a single concatenated string (use STRCAT) of the format shown in Fig. 3.2 p. 46, except that you should add a space between adjacent numbers within a given row. For instance, BRLC for the example in Fig. 3.2 should be the string with value ((1 1 1 4 4)(2 1 4)(5 2 3 5 5)), not ((11144)(214)(52355)). This permits runs longer than 10 to be decoded cor rectly.

4. **Approximate run length code (RLC) for 8-bit greylevel images:** submit an m-file ApproxRLC.m whose first line is

```
function ARLC = ApproxRLC(InIm)
```

where

InIm: An 8-bit greylevel intensity image;

ARLC: Your output string representing the approximate RLC of InIm.

To solve this problem, you will divide InIm into four images each containing a selected greylevel range, find the exact RLCs of each, and then combine them.

First construct a binary image with black pixels at each (i,j) where 0<=InIm(i,j)<=63. Find the exact RLC for this image using the function BinRLC you wrote for problem 3 above. Then construct a binary image with black pixels at each (i,j) such that 64<=InIm(i,j)<=127, and find the RLC for that image. Repeat for 128<=InIm(i,j)<=191 and 192<=InIm(i,j)<=255. Use the command STRV-CAT to vertically concatenate the four exact RLCs. The result is ARLC. From this data structure, you could reproduce a compressed version of InIm, which would look like InIm but have only 4 greylevels compared to InIm's 256.

Note that if you used 256 different exact RLCs rather than just the 4 you are asked to combine above, you would end up with an exact RLC for the 256 grey-level image InIm (you could use it to exactly reconstruct InIm). And if InIm was an RGB image rather than greylevel, by using 256x3=768 exact RLCs, one for each value of each primary color, you could get an exact RLC for any 24 bit RGB iamge.

Please concatenate the m-file BinRLC.m to the end of ApproxRLC.m before you submit ApproxRLC.m if you call the function BinRCL as part of your solution here. This will permit us to run this solution stand-alone.

Note 1: The way your submissions will be graded is by running them against a set of test data of the grader's choice which you will not know. For instance, in problem 1 a set of InIm's will be used by the grader, each of different size and containing different numbers and shapes of objects. If your output array BB matches the correct BB for each InIm the grader tries, you will get an E for the problem. If it does not, you will get S or U depending on how serious the errors in your code were. You should not submit any test images or test results with your m-file, just the m-file itself.

Note 2: for each of these problems, do not call any of the functions in the image processing toolbox. Write your own code using the only functions available in the standard matlab library.