BMI/CS 567 Medical Image Analysis
University of Wisconsin-Madison
Assignment #2

Instructor: Jeanette Mumford
Due Feb 28, 2019 *before* class begins

For this assignment you should turn in a single pdf file containing your code and the specific output requested for each problem. It shouldn't be necessary to run your code to grade the assignment. Name it `hwk2_yourname.pdf` (html is fine too). It should be submitted via canvas. The late homework will not be graded.

(1) For this problem you will use `1008_right.jpg`, which is a retina image similar to the ones you will use for the final project. The goal of the final project is to pull out features of images that help indicate when a retina shows retinopathy. This is a clear example of a diseased retina as both exudates and hemorrages are visible. Use windowing (ch 4) and steps for converting a color image to gray scale (as in 3.7.4) to emphasize the exudates in the image (brighter spots). Please don't loop through the image as the author does in 3.7.4. Make sure you choose settings that maximize the contrast between these spots and everything else in the image. Notably, it isn't like this will work perfectly, but the idea is to get you started processing the retina images.

(a) (1 point) Describe how you chose your windowing parameters and display the parameters.

(b) (1 point) Describe why you chose the weights you used to create the grayscale image.

(c) (2 points) Display the final result.

(2) Use the data in `medfilt_problem_dat.mat` (variable will be called `dat` when you read it in) for this exercise and it looks like Figure 1.

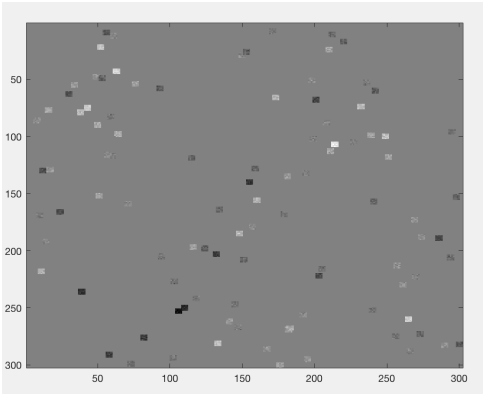

Figure 1: Image you'll be using for median filter problem.

(a) (2 points) Write the code for a median filter using a 3×3 window, using zero padding. Compare your result to what you get using `medfilt2` in MATLAB. To check this, take the difference of the two filtered images and inspect the values of the difference. Viewing the images may not be sufficient, so check the values carefully. Are the images the same? Why or why not?

(b) (1 point) You may use `medfilt2` for this part. What is the minimum filter size that removes all of the "salt and pepper" in the image?

(c) (2 points) I generated the noise in this image by randomly selecting pixels and adding noise to a 5×5 window around that pixel. For an isolated piece of this "salt" or "pepper" (meaning it isn't overlapping or near another chunk of noise), what is the minimum window size that guarantees it will be removed? You should derive this analytically. Did your solution match what you found in the previous part? Why or why not?

(3) (5 points) Linear Hough transform. For this problem you'll be working with the image shown in Figure 2, contained in `dat_hough.mat`. The goal is to apply the linear Hough transform to bring out the edges of the black shape. It is up to you to process the image applying whatever step you think are necessary to provide the proper input for the Hough Transform. Apply the Hough transform using the matlab function from class. You may not use the built in MATLAB Hough transform function.

Please display an image after each preprocessing step you applied as well as the Hough transform result, using `imfuse` to combine the Hough lines and the original image (as we did in class). You must use approaches learned in this class for each step and explain why you chose each of the approaches to process the image.

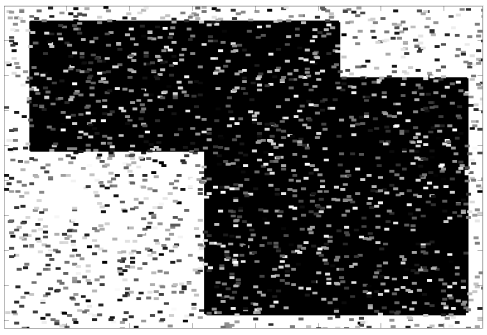

Figure 2: Image you'll be using for linear Hough Transform problem.

(4) (5 points) The Circle Hough transform. The circle Hough is very similar to the linear Hough, but instead of constructing lines through each edge point, we construct circles centered at those points. If the proper radius is chosen, the constructed circles will overlap at the center of the circles you are trying to identify. See class notes for more details. Circles require three parameters: x-coordinate of the center, y-coordinate of the center and the radius. So, unlike the linear Hough, there are 3 parameters to search over instead of 2. You will create a 3D accumulator matrix (x-coordinate, y-coordinate and radius) that counts how often a constructed circle passed through an

(x,y) coordinate for a given radius. I will give you three radius values to try (48, 50 and 52). Your original image is 540×720, so your accumulator matrix is $540 \times 720 \times 3$. Use the `img_circle.mat` file for this problem, which is also shown in Figure 3. There are two circle sizes in the image, but your goal is to find the smaller circles. I have already binarized it, so no preprocessing is required and you may simply start with the Hough transform.

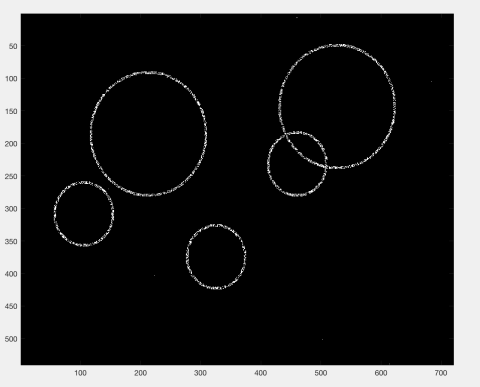

Figure 3: Image you'll be using for the circle Hough Transform problem.