Rensselaer Polytechnic Institute
Department of Electrical, Computer, and Systems Engineering
ECSE 4540: Introduction to Image Processing, Spring 2018

Homework #5: due Monday, Mar. 5^{th}, at the beginning of class.
Show all work for full credit!

For this homework, you'll need the images in `http://www.ecse.rpi.edu/~rjradke/4540/hw5.zip`.

1. (10 points.) Consider the image given in the file `market.png`.
   (a) Determine the gradient, its magnitude, and its angle at the point (205,696), using the estimate $\frac{df}{dx} = f(x+1,y) - f(x,y)$ (and similarly for $\frac{df}{dy}$). Throughout this problem, interpret the x direction as pointing down the rows and the y direction as pointing across the columns (i.e., Matlab/image coordinates).
   (b) Determine the gradient, its magnitude, and its angle at the point (205,696), using the Sobel operators divided by 8 to estimate the derivatives.

2. (15 points.) Use the Sobel operators divided by 8 to estimate the magnitude and angle of the gradient over the entire `market` image; that is, create two new images `mag` and `ang` containing the estimated gradient magnitude and angle at each pixel.
   (a) Display and interpret the results of the binary image created by finding the pixels whose magnitude is above 50.
   (b) Display and interpret the results of the binary image created by finding the pixels whose angle is within 20° of 45° (on either side).
   (c) Display and interpret the results of the logical AND of the previous two images.

3. (30 points.) Here, we'll try to find the long lines in the `deck.png` image.
   (a) (5 points.) First, extract the Canny edges (using the default Matlab thresholds). What does the edge map look like?
   (b) (10 points.) Use these edges as the input to the `hough` function to return the Hough array and corresponding ρ and θ vectors. Then use the `houghpeaks` function to find at most 50 highest peaks. Determine how to superimpose these lines onto the original image and return the result.
   (c) (5 points.) Do these lines make sense? Why or why not? Why are there many lines detected at ±45°?
   (d) (5 points.) Now, extract edges using the `Sobel` option to the `edge` command, instead of Canny, and repeat the process in part (b) (again, using all of Matlab's default thresholds). Explain what you see. Is this better than part (c) and why?
   (e) (5 points.) Tune the threshold option to `houghpeaks` until you're able to extract all the major lines (without detecting any spurious lines). How low did you need to set the threshold to get to this point?

4. (15 points.) The command `imfindcirlces` uses a Hough-transform-like method to find circles in an image. Load up the image `curling.jpg` and tweak the parameters to find as many circles as you can (without generating any spurious circles). You can try changing the `objectpolarity` and `edgethreshold` inputs, as well as trying different color channels of the input image (e.g., the red ring will stand out better in the blue channel). Describe the settings you eventually arrived at to get the best result. Note that you will need several calls to `imfindcirlces` with different radii since the function will complain if you use too wide a range.

5. (15 points.) Threshold the `important.png` image to find pixels above a grayscale level of 180. Display the resulting image. Use `bwlabel` and `regionprops` to automatically isolate the white region with the largest area, and display this as a new image.

6. (15 points.) Consider the image in `page.png`. Use `graythresh` to find Otsu's threshold and apply it to the image (i.e., turn everything darker than the threshold to white). How does it look? Why is a global threshold suboptimal for this task?

   Now, use the `blockproc` command to process each 60×60 block of the image separately, returning white pixels where the pixel intensity $p < \mu - 2\sigma$, where μ is the mean intensity of the block and σ is the standard deviation of the block. Interpret the results. Why does the local threshold work better?