Homework #4
Due Th. 10/15

Be sure to show all your work for credit. You must turn in your code as well as output files (code attached at the end of the report).

Please generate a report that contains the code and output in a single readable format using Latex.

0. Getting Started

- Download the homework images from the class website.

  http://www.ee.unlv.edu/~b1morris/ecg782/hw/hw04

1. (GW 10.6)
   
   **Solution**

   It is given that the location of the edge relative to the size of the mask is such that image border effects can be ignored. Assume that n is odd and keep in mind that an ideal step edge transition takes place between adjacent pixels. Then, the average is 0 until the center of the mask is \((n-1)/2\) pixels or more to the left of the edge. The average is 1 when the center of the mask is further away than \((n-1)/2\) pixels to the right of the edge. When transitioning into the edge, (say from left to right) the average picks up one column of the mask for every pixel that it moves to the right, so the value of the average grows as \(n/n^2, 2n/n^2, \ldots, (n-1)n/n^2, n^2/n^2\), or \(1/n, 2/n, \ldots, (n-1)/n, 1\). This is a simple linear growth with slope equal to \(1/n\). Figure P10.4 shows a plot of the original profile and what the profile would look like after smoothing. Thus, we get a ramp edge, as expected.

2. (GW 10.22)
   
   **Solution**

   (a) Express \(x \cos \theta + y \sin \theta = \rho\) in the form \(y = -(\cot \theta)x + \rho/\sin \theta\). Equating terms with the slope-intercept form, \(y = ax + b\), gives \(a = -\cot \theta\) and \(b = \rho/\sin \theta\). This gives \(\theta = \cot^{-1}(a)\) and \(\rho = b \sin \theta\). Once obtained from \(a\) and \(b\) of a given line, the parameters \(\theta\) and \(\rho\) completely specify the normal representation of that line.

   (b) \(\theta = \cot^{-1}(2) = 26.6^\circ\) and \(\rho = (1) \sin \theta = 0.45\).

3. (GW 10.36)
   
   **Solution**

   The simplest solution is to use the given means and standard deviations to form two Gaussian probability density functions, and then to use the optimum thresholding approach discussed in Section 10.3.5 (in particular, see Eqs. (10.311) through (10.313)). The probabilities \(P_1\) and \(P_2\) can be estimated by visual analysis of the images (i.e., by determining the relative areas of the image occupied by objects and background). It is clear by looking at the image that the probability of occurrence of object points is less than that of background points. Alternatively, an automatic estimate can be obtained by thresholding the image into points with values greater than \(170 - 30 = 140\) and less than \(60 + 30 = 90\) (see problem statement). Using the given parameters, the results would be good estimates of the relative probability of occurrence of object and background points due to the separation between means, and the relatively tight standard deviations. A more sophisticated approach is to use the Chow-Kaneko procedure discussed in Section 10.3.5.
4. Thesholding

Suppose an image has the gray-level pdf shown below. $p_1(z)$ corresponds to objects and $p_2(z)$ to background. Assume $P_1 = P_2$, find the optimal threshold between object and background pixels. Be sure to derive the optimal value mathematically to get the optimal value.

![Gray-level pdf](image)

**Solution**

This is a simple problem where the optimal threshold should be found using the conditional distributions. Find the value of $z$ such that

$$p_1(z)P_1 = p_2(z)P_2$$

$$p_1(z) = p_2(z).$$

This can easily be solved by inspection for threshold $T = 1.5$.

5. Canny Edge Detection

(a) Give the convolution kernels for determining the gradient. You may examine the function `gradient.m` to help with the explanation. (It may be easiest to apply the `gradient` to an impulse and inspect the results).

(b) Implement the simplified version of the Canny edge detector (no hysteresis thresholding). The syntax of the function should be

$$[E,M,A] = \text{canny}(I, \text{sig}, \text{tau}),$$

where $E$ contains the detected edges, $M$ the smoothed gradient magnitude, $A$ contains the gradient angle, $I$ is the input image, `sig` is the $\sigma$ parameter for the smoothing filter, and `tau` is a single threshold.

(c) Apply your Canny detector on `wirebond_mask.tif` using $\tau = 0.8$ and $0.6$ with the following values for $\sigma^2 = [0.5, 1, 3]$. Show your results in a (2,3) subplot. Invert the color, white for 0 and black for 1, to save ink. Discuss how the choice of $\sigma$ affects the results.

(d) Apply your Canny detector on `city.jpg`. Adjust the $\sigma$ and $\tau$ parameters as you see fit. Display the resulting edges and the parameter settings used. Also use the built-in Matlab function `edge.m` with default parameters on the `city.jpg` image. Compare.

(Bonus) Modify your Canny detector to implement hysteresis thresholding with $\text{tau} = [\tau_h, \tau_l]$. 

**Solution**
Figure 1: Top Row: $\tau = 0.8$ and $\sigma = \{0.5, 1.0, 3.0\}$ Bottom Row: $\tau = 0.6$ and $\sigma = \{0.5, 1.0, 3.0\}$

(a) This is the centered first difference kernel $g(x, y) = 0.5(f(x + 1, y) - f(x - 1, y))$.

(b) Do it.

(c) Obviously, the choice in $\tau$ changes the amount of edges with higher $\tau$ resulting in fewer edges. The choice of $\sigma$ affects which lines remain. By changing $\sigma$ we are changing the amount of smoothing and reducing the strength of edges. This most comes into play on the diagonal lines which tend to disappear with greater smoothing. See Fig. 1.

(d) The parameters use are $\tau = 0.3$ and $\sigma = 3.0$. This was selected to only get large scale lines (houses) but to return many lines. By comparison, the default Canny parameters give more smaller scale lines. See Fig. 2.

6. Hough Transform

Figure 2: Left: homemade $\tau = 0.3$ and $\sigma = 3.0$ Right: Default parameters.
Figure 3: Homemade Canny Parameters: (a) Accumulator array (b) Original image with top-5 lines (c) Edge image with top 5-lines

Figure 4: Default Canny Parameters: (a) Accumulator array (b) Original image with top-5 lines (c) Edge image with top 5-lines

(a) Study the Matlab function `hough.m`. Compute the Hough transform of the `city.jpg` image. Display the Hough accumulator image and the original image with the top 5 lines as an overlay. Each overlay line should be a different color and a legend should be included.

(b) Write your own Hough transform implementation for circle detection. The function should take an image and radius as input. Test your function on the `quarters.bmp` image. Display the accumulator image and the original image with the top 3 circles as an overlay.

(Bonus) i) Use gradient magnitude accumulation instead individual count. ii) Upgrade the detector to find circles of different sizes. You may test results on `us.silver.coins.jpg`.

Solution

(a) Fig. 3 shows the results using the homemade canny detection scheme while Fig. 4 gives the results using the default parameters. Notice that the top lines are not always what is expected. Some lines are formed by parts of different structures in different parts of the image. This results in a global line even though locally it isn’t consistent. This is why there is generally some sort of check on extent.

(b)