Keone Hiraide
October 15, 2013
ICS 683

Homework 3

1. (10 points)
A binary image contains straight lines oriented horizontally, vertically, at 45°, and at –45°. Give a set of 3 × 3 masks that can be used to detect 1-pixel-long breaks in these lines. Assume that the gray level of the lines is 1 and that the gray level of the background is 0.

The Laplacian masks W_7 and W_8 would respond with high energy over the 1-pixel long breaks in the image below. The mask below can also be used to detect these 1-pixel-long breaks. If we applied cross correlation on the image shown below, then we can expect that around the breaks, the outputted image pixels would have values equal to 2 around the pixel breaks.

0	0	1
0	1	0
1	0	0

2. (15 points) The three images (a) ~ (c) shown below were blurred using square averaging masks of sizes n = 23, 25, and 45, respectively. In the original image, the vertical bars on the left lower region(red circled area) are 5 pixels wide, 100 pixels high, and their separation is 20 pixels. The vertical bars in (a) and (c) are blurred, but a clear separation exists between them. However, the bars have merged in image (b), in spite of the fact that the mask that produced this image is significantly smaller than the mask that produced image (c). Explain why.
The reason why there is a clear separation between the bars in image (c) even though its n value is significantly larger than the n value for (b) is because when n=45, more pixels are calculated in the average, and thus, a large portion of the pixels are background pixels. I suspect that for n = 25, … n = 29, there won't be a clear separation between the outputted image. Once n > 29, the mask's width will go beyond the bars, and thus start grabbing more background pixels to use in its average.

3. (15 points)
(1) Represent the intensity neighborhood

```
      ------------------
     | 10 | 10 | 10 |
N=   | 10 | 20 | 10 |
     | 10 | 10 | 10 |
     ------------------
```

in terms of the nine Frei-Chen basis vectors. Is all the energy distributed along certain basis vectors as you expect?

N o W_1 = (10 + 10 * sqrt(2) + 10 – 10 – 10 * sqrt(2) – 10) / sqrt(8) = 0 Energy = 0
N o W_2 = (10 – 10 + 10 * sqrt(2) – 10 * sqrt(2) + 10 – 10) / sqrt(8) = 0 Energy = 0
N o W_3 = (-10 + 10 * sqrt(2) + 10 – 10 – 10 * sqrt(2) + 10) / sqrt(8) = 0 Energy = 0
N o W_4 = (10 * sqrt(2) – 10 – 10 + 10 + 10 – 10 * sqrt(2)) / sqrt(8) = 0 Energy = 0
N o W_5 = (10 – 10 – 10 + 10) / 2 = 0 Energy = 0
N o W_6 = (-10 + 10 + 10 – 10) / 2 = 0 Energy = 0
N o W_7 = (10 – 20 + 10 – 20 + 80 – 20 + 10 – 20 + 10) / 6 = 6.67 Energy = 44
N o W_8 = (-20 + 10 -20 + 10 + 80 + 10 -20 +10 -20) / 6 = 6.67 Energy = 44

As we can see, the energy is distributed along the lapacian basis vectors W_7 and W_8. I expected the lapacian basis vector to respond in this way because the laplacian operator responds strongly to areas of rapid change. Thus, we see that N mostly consists of intensities of 10, but there is a sudden change in intensity with the 20 valued pixel at the center, so the W_7 and W_8 basis vectors react accordingly.

(2) Would the interpretation of the intensity neighborhood

$$\mathbf{M} = \begin{matrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{matrix}$$

be different? Why or why not?

The interpretation would be the same, since M's intensity neighbor has the same intensity structure as the intensity neighborhood of N.

4. (60 points) Implement Canny's Edge Detection Algorithm. In your implementation set the width and height of the Gaussian filter to 2σ and handle the boundary case by padding rows and columns of zeros. Download the image called "board.tif" from the course website and run your code with σ = 1.5, 2.5, 3.5 and low and high threshold of 0.1 and 0.3, respectively.
Note that the threshold values should be applied to the normalized gradient magnitude values (i.e. the magnitude values between 0 and 1). In the report,
• describe implementation details for each step of the algorithm
• show the outputs of your implementation in which detected edges are marked as white and the weak edges (edges with magnitude between the low and the high thresholds) that did not survive in the hysteresis process as gray. (Non-edges, i.e. the background, are black.)

1) The first step in my implementation was to compute a Gaussian kernel determined by the size and sigma inputted. In my program, the size of the Gaussian kernel was: (sigma * 2). To do this, I used the two-dimensional Gaussian function:

$$g(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

2) I then read in "board.tif" and used a convolution operator on board.tif using my Gaussian kernel in order to get a smoothed image.

3) I then applied used Sobel's convolution mask in order to retrieve the gradient magnitudes first along the x-axis, then the y-axis called Gx and Gy. I then used:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

in order to retrieve an image of board.tif with its gradient magnitude.

4) To retrieve the gradient direction of the magnitudes, I used:

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x),$$

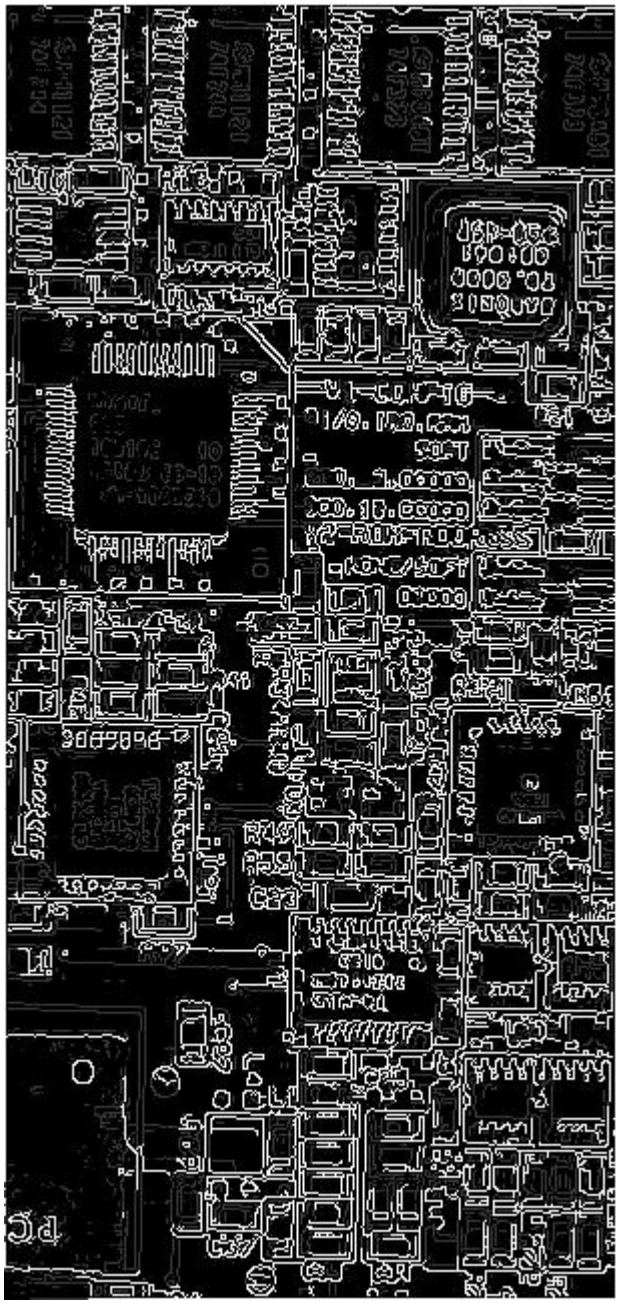
If theta fell below 0, I added (2 * pi) to the values to convert them into positive values. The theta values were then converted into degrees by multiplying theta by (180/Math.PI). With the theta values in degrees, I rounded their values to either: 0, 135, 90, or 45.

5) Non-maximum suppression was then performed. If the angle was 0 degrees, or in the south-north direction, I checked its neighboring pixel values to the west and east of it. If the angle was 90 degrees or in the east-west direction, I checked its neighboring pixel values in the north and south directions. If the angle was 135, or in the north-east south-west position, I checked its neighboring pixel values in the north west and south east directions. Lastly, if the angle was 45 degrees or in the north-west south-east direction, I checked its neighboring pixel values in the north-east and south-west positions. Note that if the current pixel being checked was not greater than both of its neighboring pixels gradient magnitude, then the current pixel gradient magnitude was set to 0.

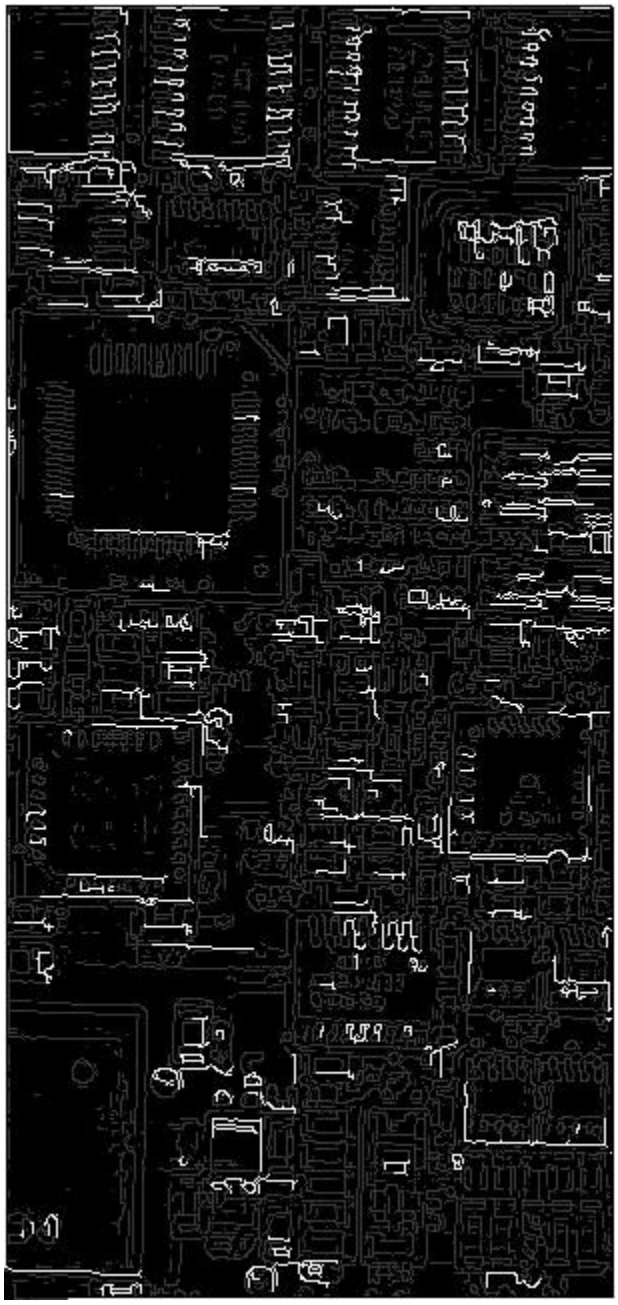
6) Hysteresis was then performed. If the current pixel's gradient magnitude was less than the lower threshold, we moved on to the next pixel. If it was greater than the high threshold, then the corresponding output image's pixel was set as a foreground pixel. If the current pixel was less than the high threshold but greater but greater than the low threshold, then it was marked gray. I then revisited those gray pixels and checked to see if any of its neighbors were edge pixels. If any of its neighbors were edge pixels, then the gray pixel turned into an edge.

My output after post hysteresis with low threshold: 0.1 and high threshold: 0.3:

σ = 1.5



σ = 2.5



σ = 3.5