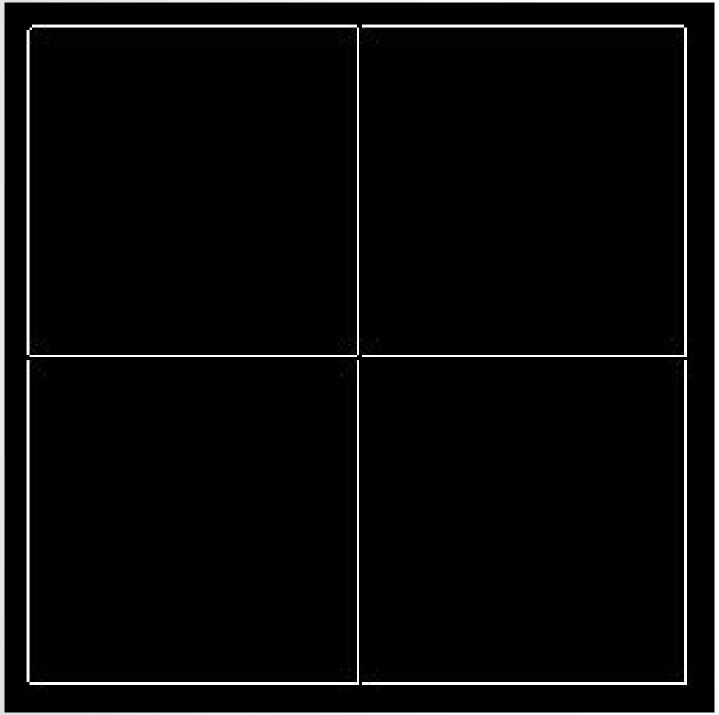
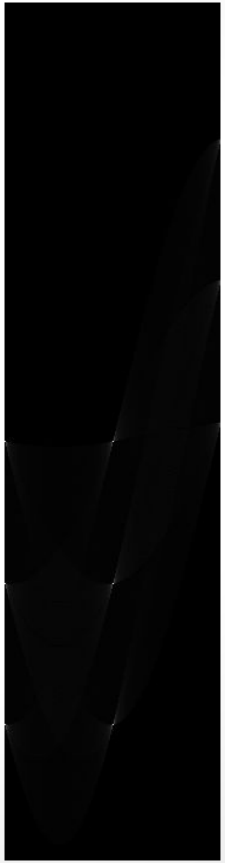
Computer Vision
Spring 2017
Problem Set #2

Yonathan Halim
yonathan@gatech.edu

1a: Edge Image

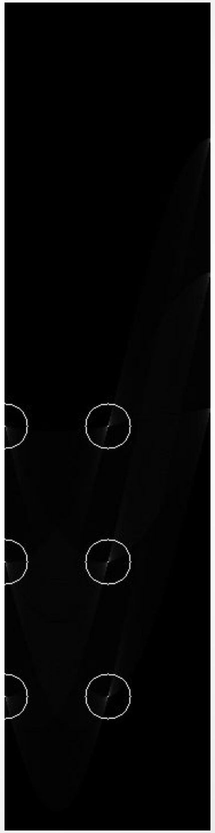

img_edges - ps2-1-a-1.png

2a: Hough Accumulator Array



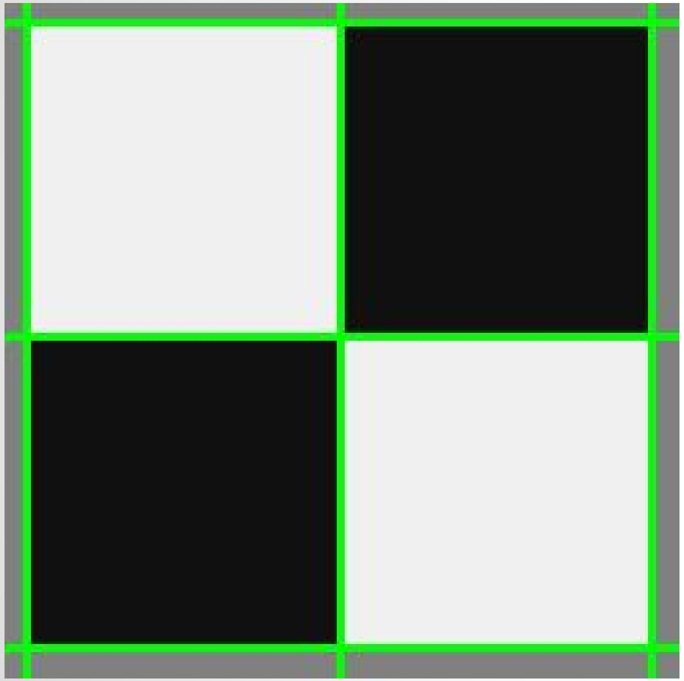
Normalized Accumulator Array - ps2-2-a-1.png

2b: Accumulator Array w/Peaks

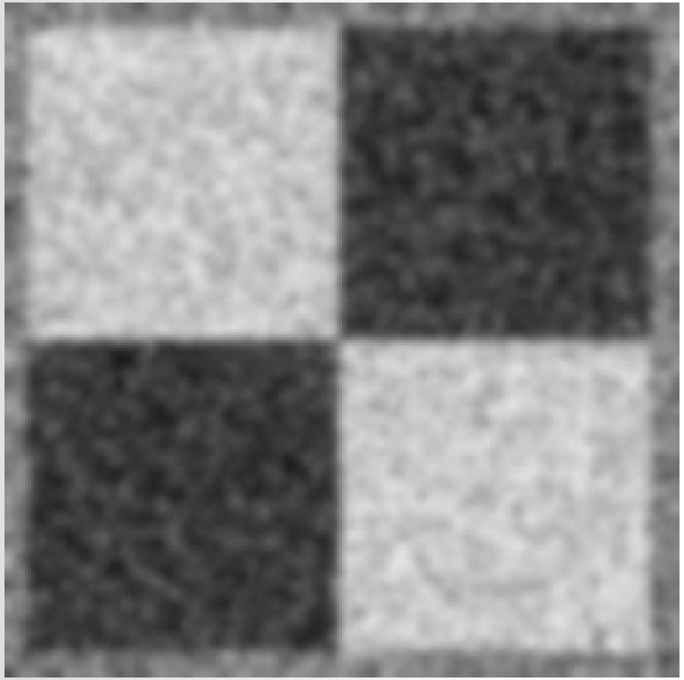


Accumulator Array with peaks highlighted - ps2-2-b-1.png

2c: Image with Lines Drawn

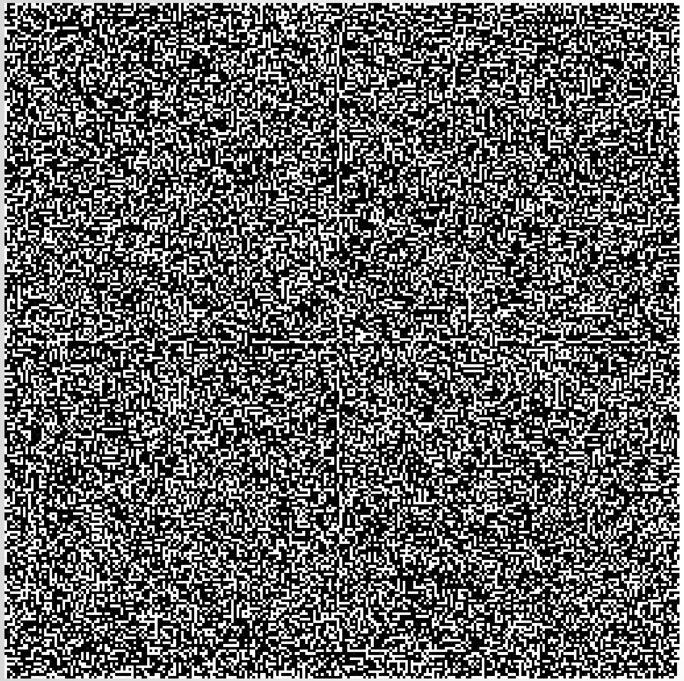

Original grayscale with lines drawn - ps2-2-c-1.png

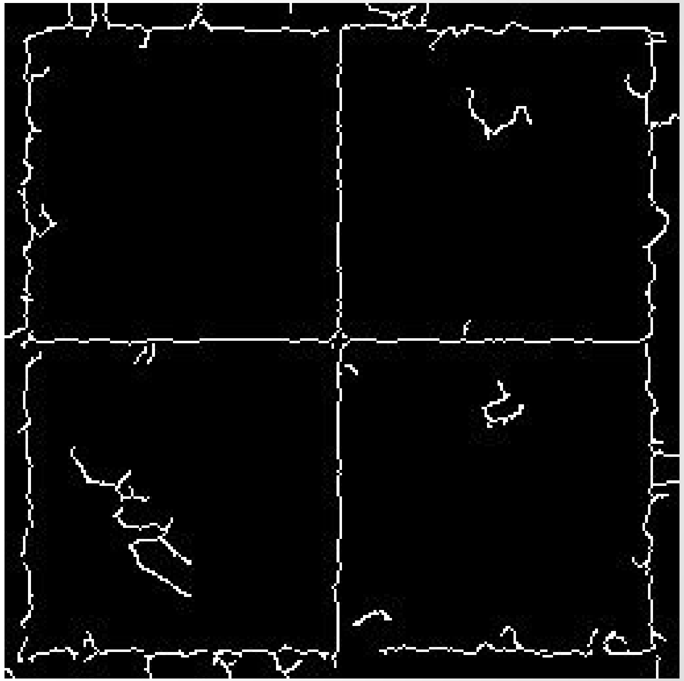
3a: Smoothed Intensity Image



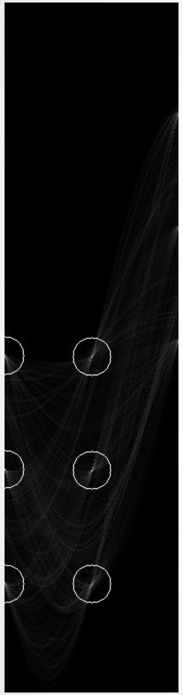
ps2-3-a-1.png

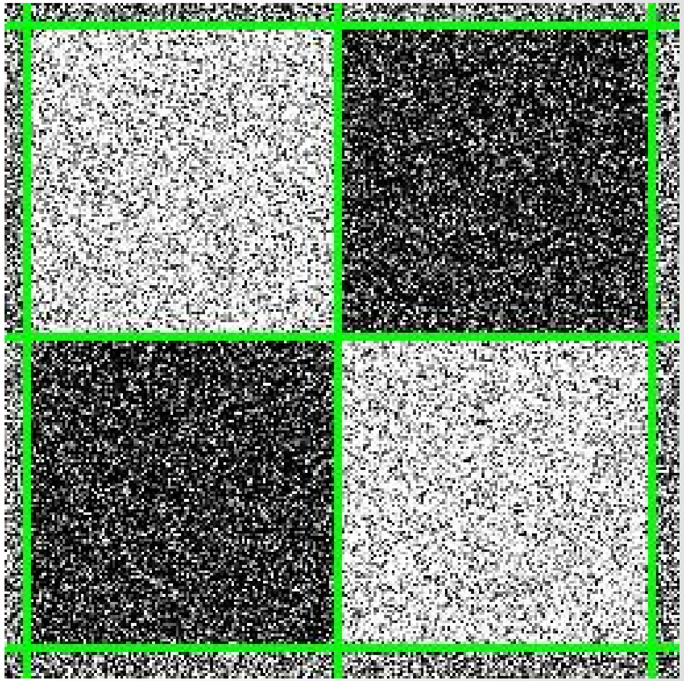
3b: Edge Image Comparison


Edge Image (original) - ps2-3-b-1.png

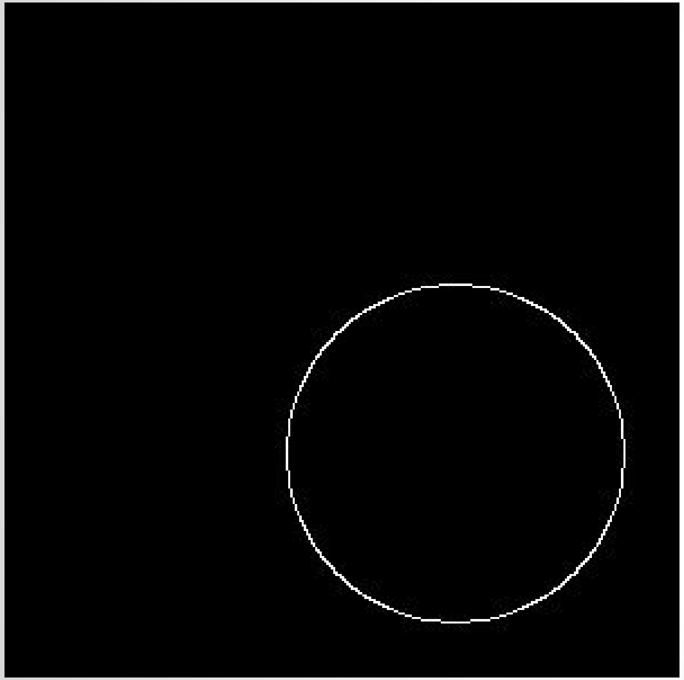

Edge Image (smoothed) - ps2-3-b-2.png

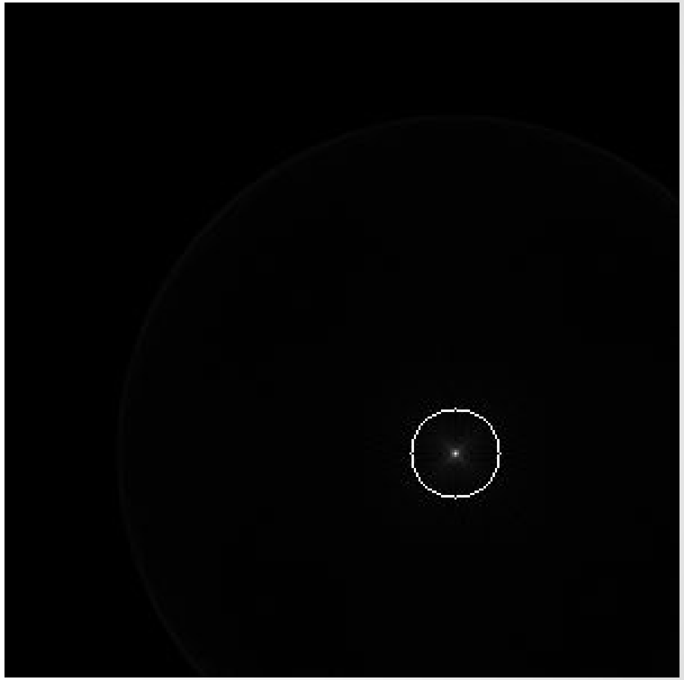
3c: Hough Transform Smoothed


Accumulator Array with peaks highlighted - ps2-3-c-1.png

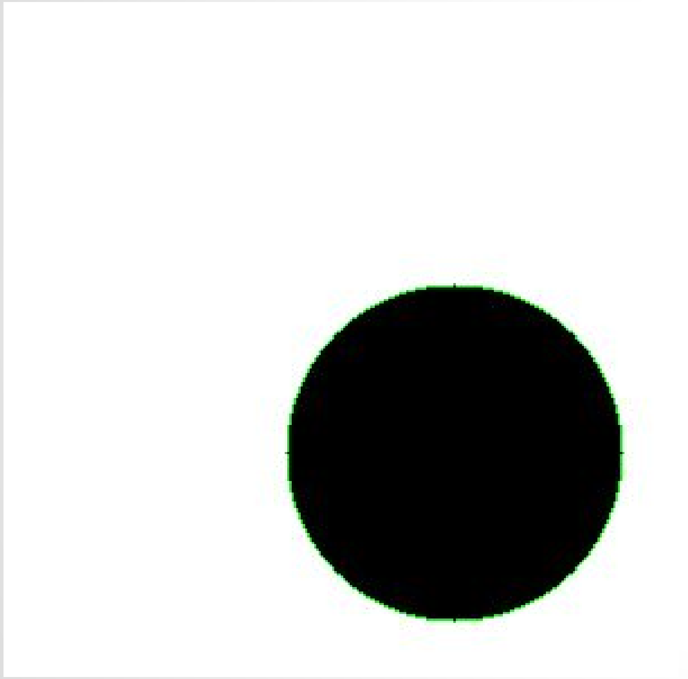

Original intensity image with lines drawn - ps2-3-c-2.png

4a: Single-Point Method


Edge Image - ps2-4-a-1.png


Accumulator Array with peaks highlighted - ps2-4-a-2.png

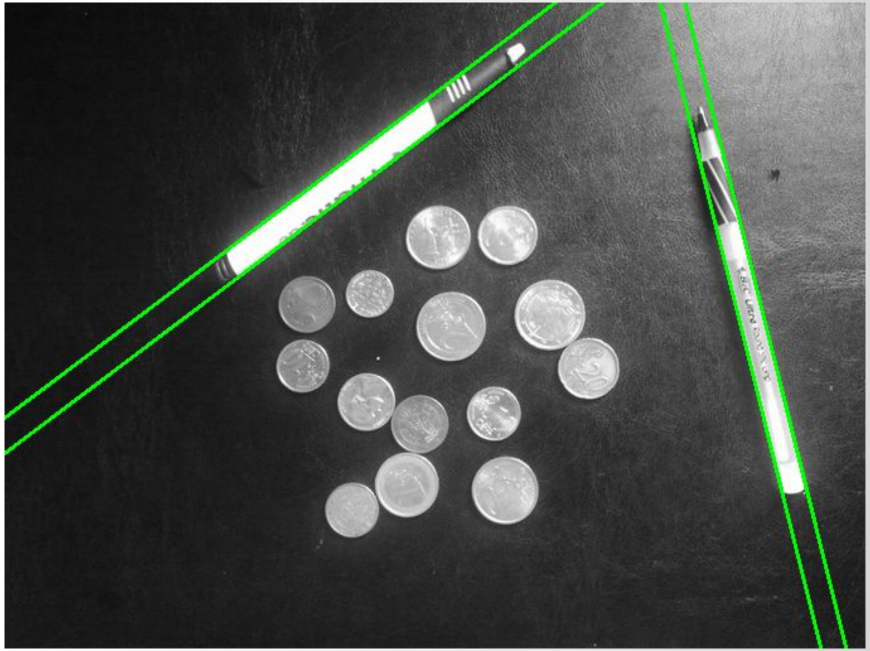
4b: Point-Plus Method



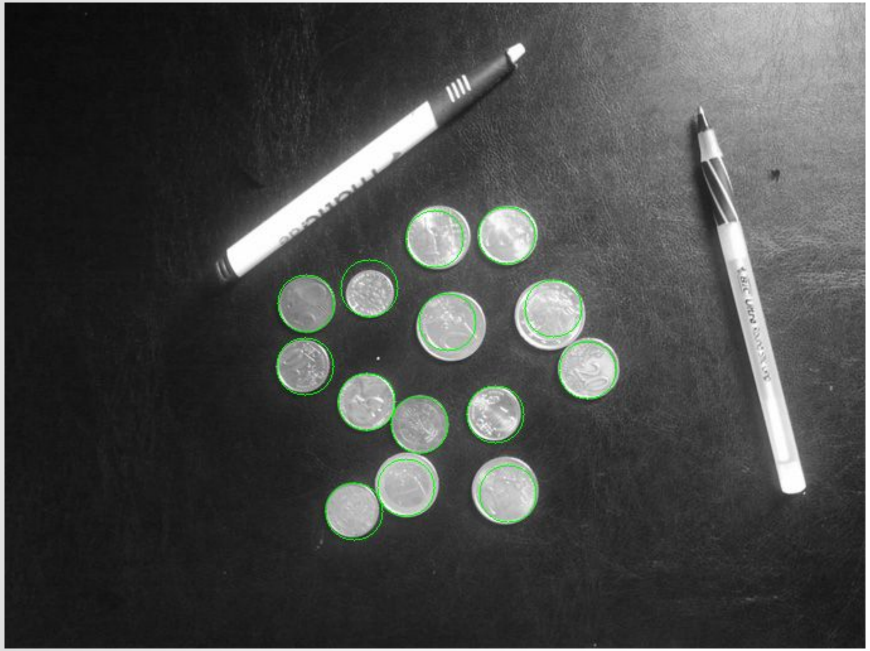
original monochrome image with circles drawn in color- ps2-4-b-1.png

5a: Coins and Pens

Accumulator Array with peaks highlighted - ps2-5-a-1.png

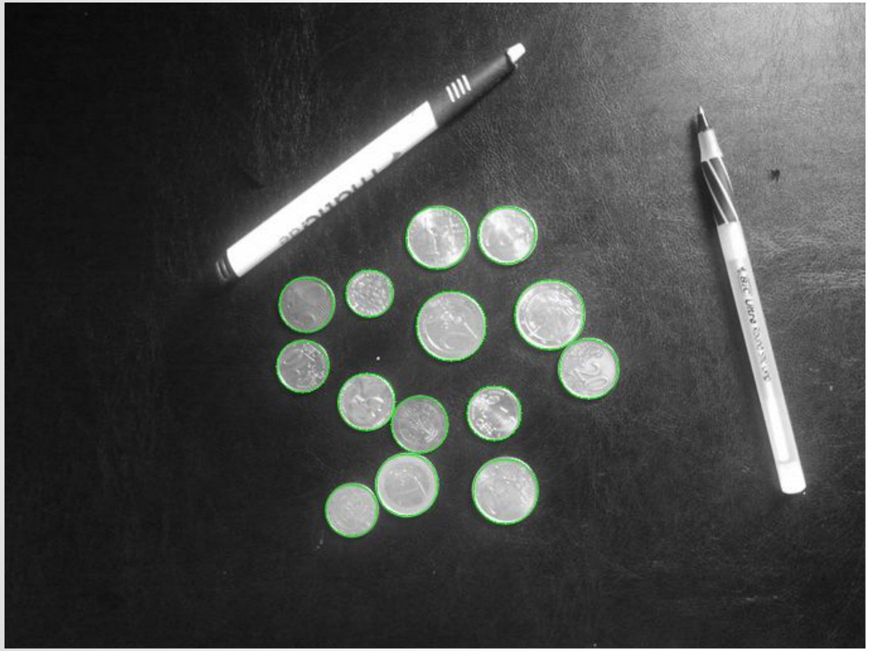

Monochrome image with lines drawn - ps2-5-a-2.png

5b: Coins and Pens: Circles



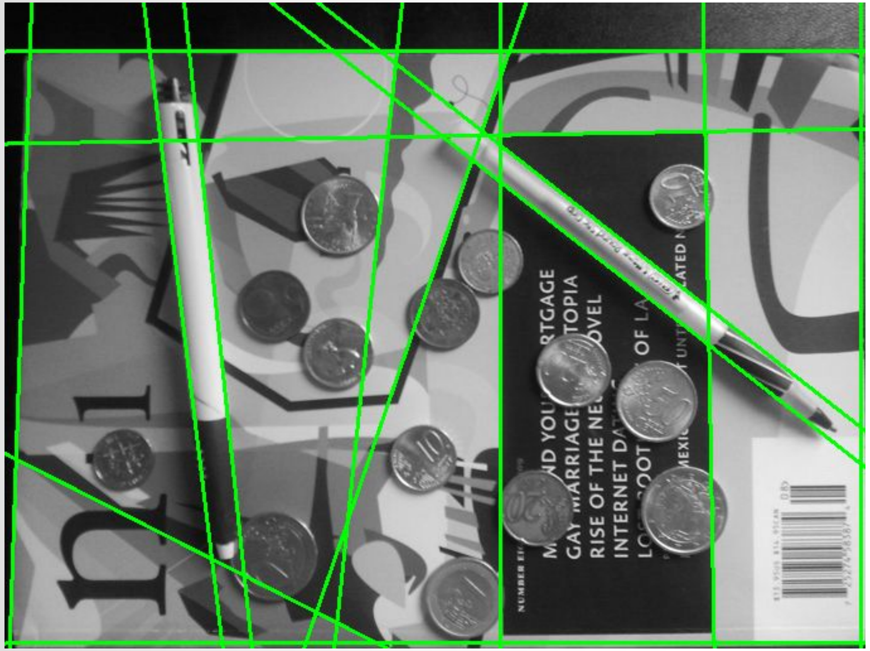
Original monochrome image with circles drawn - ps2-5-b-1.png

5c: Coins and Pens: Circles (cont.)



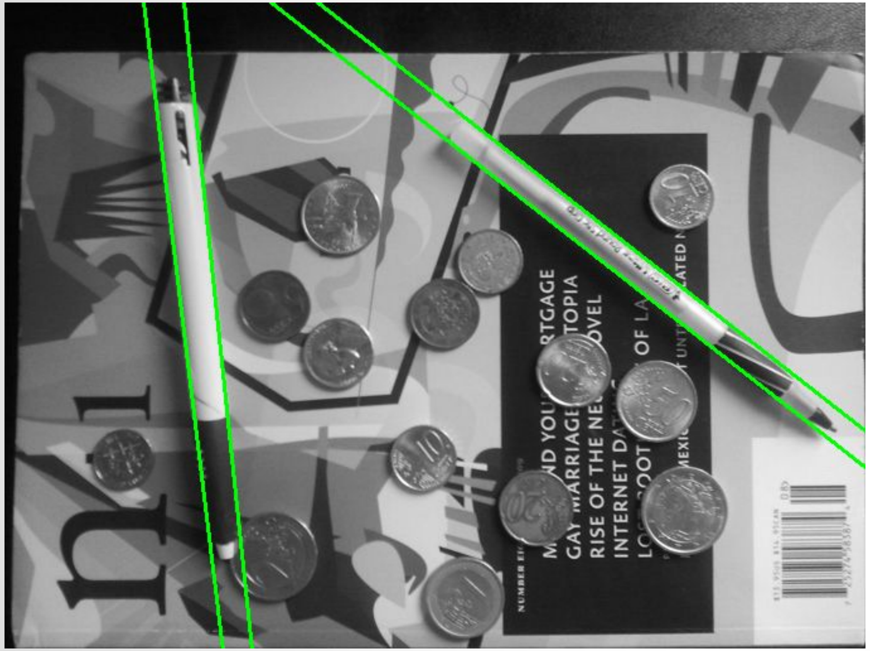
Original monochrome image with circles drawn - ps2-5-c-1.png

6a: Images with Clutter (pens)



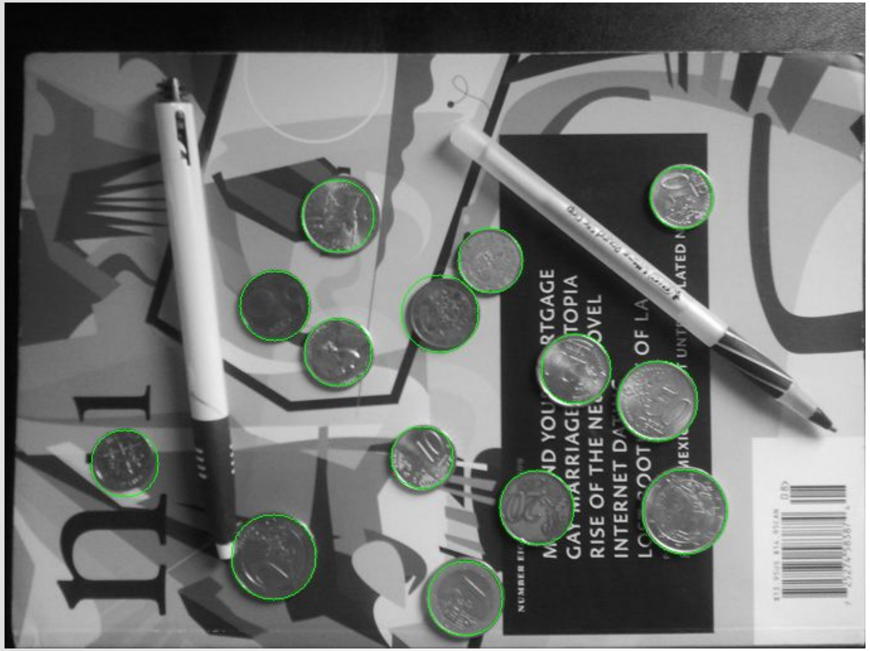
Smoothed image with lines drawn - ps2-6-a-1.png

6b: Images with Clutter (pens), part 2



Smoothed image with lines drawn - ps2-6-b-1.png

6c: Images with Clutter (coins)



Smoothed image with circles drawn - ps2-6-c-1.png

7: Discussion

Answer the questions below:

a. For each of the methods, what sorts of parameters did you use for finding lines in an image? Circles? Did any of the parameters radically change? (Describe your accumulator bin sizes, threshold, and neighborhood size parameters for finding peaks, and why/how you picked those.)
   i. I use GaussianBlur with filter size of 13 and canny edge detection with threshold between 28-115. The same parameter is used throughout the sample images.
   ii. To find lines, I use accumulator of 2*rho for the row size and 180 degrees for the col size. The bin size is 1 pixel * 1 degree (0.01745 radian).
   iii. To find circles, I use accumulator the same size of the image. For the single point, the bin size is 1* 1 pixel and for the point plus, the bin size is 3 * 3 pixels.

7: Discussion

Answer the questions below:

b. What differences do you see when finding the edges in a noisy vs. regular image?

   i. To find edges in a noisy image, we have to apply gaussian filter to blend the noise so that we can find the actual edges. In regular image, we normally do not need to apply the gaussian filter before finding the edges.

7: Discussion

Answer the questions below:

c. Is there any perceived difference in time or computational cost between the single-point and the point-plus gradient method implementations?

    i. For this specific implementation, the point-plus has less cost than the single-point (given that we are comparing when finding the same circle). This is because for the single point, we have to iterate for every possible theta while in point-plus, we use the gradient x and y to compute the theta.

7: Discussion

Answer the questions below:

d. For question 5, were you able to find all the circles? Describe what you had to do to find circles.

   i. Yes, I was able to find all the circles. To find those circles, I still use the same gaussian blur with filter size of 13 and canny edge threshold between 28-115. However, I change the ksize for sobel operator to 11. Then, I set the hough_threshold to 100 and neighborhood to 20x20 so that it does not have extra circles.

7: Discussion

Answer the questions below:

e. In question 6, for a cluttered image, you likely found lines that are not the boundaries of the pen. What problems did you face to overcome this?

   i. In the cluttered image, there are many lines found that are not boundaries of the pen. First, I tweak the edge detection to get the minimum number of lines including the pen edges. Next, Lines with same theta are kept (with margin +/- 1 deg). This will keep all the parallel lines. Finally, lines with difference in rho less than 40 pixels are kept. This results in the 4 lines representing the edges of the pens.

7: Discussion

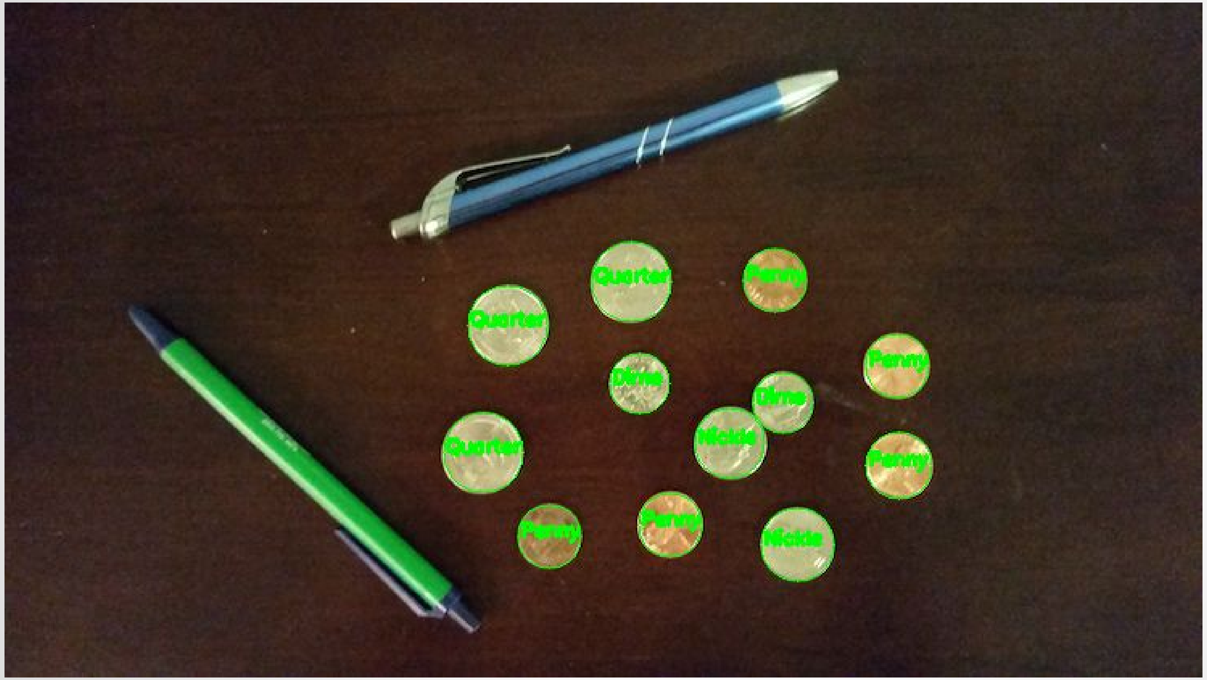
Answer the questions below:

f. Likewise in question 6, there may have been false positives for circles. Did you find such false positives? How would/did you get rid of them? If you did these steps, mention where they are in the code by file, line no., and also include brief snippets

   i. I managed to remove the false positives by increasing the hough_threshold and nhood_radii. I also provide the radius in .5 interval to try to get more precise radius but unsuccessful to do so. I think this is more to do with my canny edge and the gradient x/y.

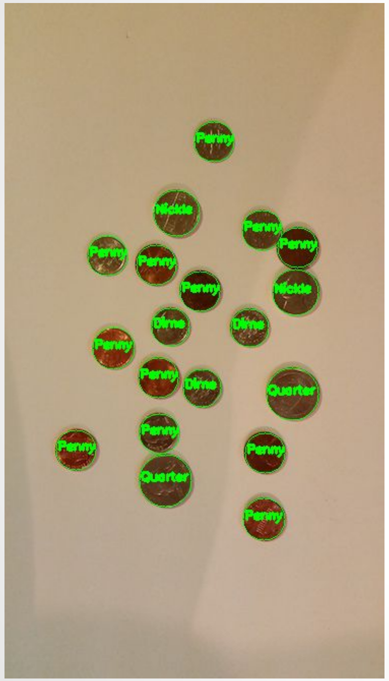
```
cluttered_image_grad_x = cv2.Sobel(cluttered_image, cv2.CV_64F, 1, 0, ksize=3)
cluttered_image_grad_y = cv2.Sobel(cluttered_image, cv2.CV_64F, 0, 1, ksize=3)
radii_range = np.linspace(25, 35, 21)
circles = find_circles(cluttered_image_edges, cluttered_image_grad_x,
cluttered_image_grad_y,
                    radii=radii_range, hough_threshold=156, nhood_radii=(20, 20))
```

8: CHALLENGE PROBLEM



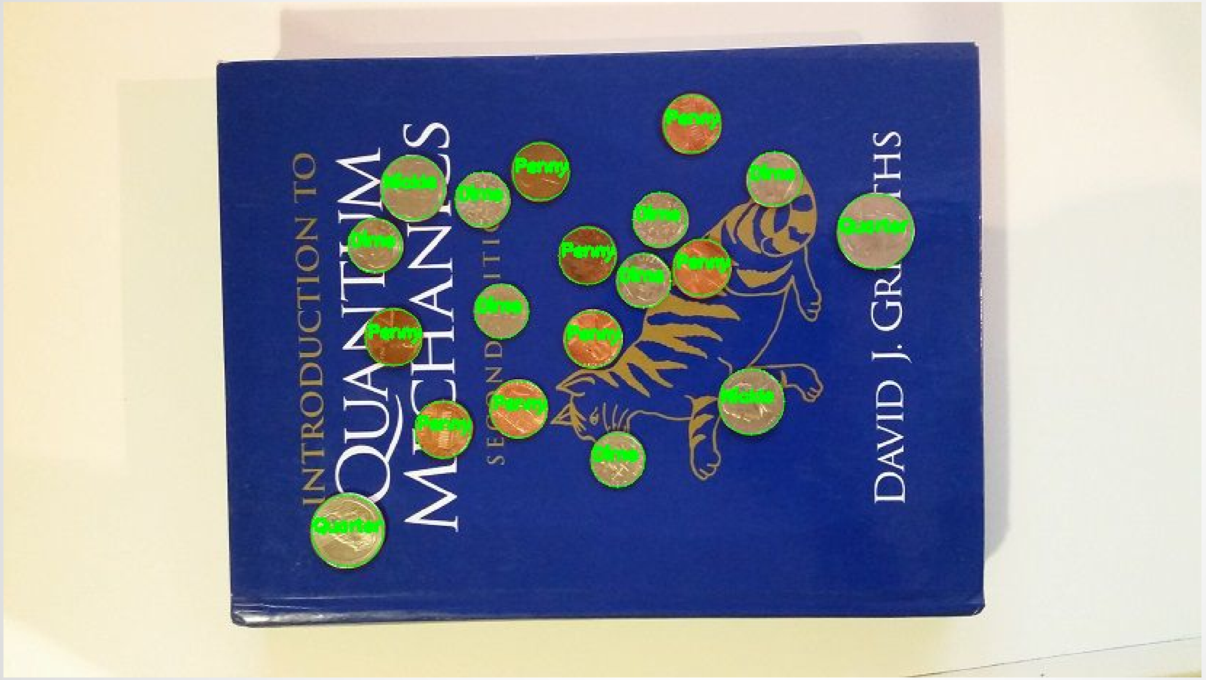
Results of Challenge Problem a - ps2-8-a-1.png

8: CHALLENGE PROBLEM



Results of Challenge Problem a - ps2-8-a-2.png

8: CHALLENGE PROBLEM



Results of Challenge Problem a - ps2-8-a-3.png

8: CHALLENGE PROBLEM

For the challenge problem, I was able to successfully recognize all coins and their value by using the radius. First, I apply gaussian blur with filter size of 7x7 then extract the canny edge from the smoothed image with threshold 28-115. To be able to successfully recognize the coin size, I needed to set the radii range in .5 degree interval. In addition, I extract gradient x and y with ksize of 7. By setting the threshold large enough, I was able to exclude uninteresting points and only include the points where the center of the circles are located.

8: CHALLENGE PROBLEM

Snippet code:

```
challenge_image_orig = cv2.imread(os.path.join(input_dir, 'challenge_image1.jpg'), 1)
challenge_image = cv2.cvtColor(challenge_image_orig, cv2.COLOR_BGR2GRAY)
challenge_image_smoothed = get_smoothed_image(challenge_image, (3, 3), sigmaX=0, sigmaY=0)
challenge_image_edges = get_edge_image(challenge_image_smoothed, 28, 115)
cv2.imwrite(os.path.join(output_dir, 'ps2-8-a-1-challenge_image1_edges2.png'), challenge_image_edges)

radii_range = np.linspace(19, 26, 15)
circles = find_circles(challenge_image, challenge_image_edges, radii=radii_range, hough_threshold=150, nhood_delta=(10, 10))

circles = circles.astype(np.int)
output_image = draw_circles_color(challenge_image_orig, circles)
for circle in circles:
    text = ''
    if circle[2] in [20, 21]:
        text = 'Penny'
    elif circle[2] in [22, 23]:
        text = 'Nickle'
    elif circle[2] in [19]:
        text = 'Dime'
    elif circle[2] in [25, 26]:
        text = 'Quarter'
    cv2.putText(output_image, text, (circle[1]-circle[2]+2, circle[0]), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 255, 0), 2)
cv2.imwrite(os.path.join(output_dir, 'ps2-8-a-1.png'), output_image)
```