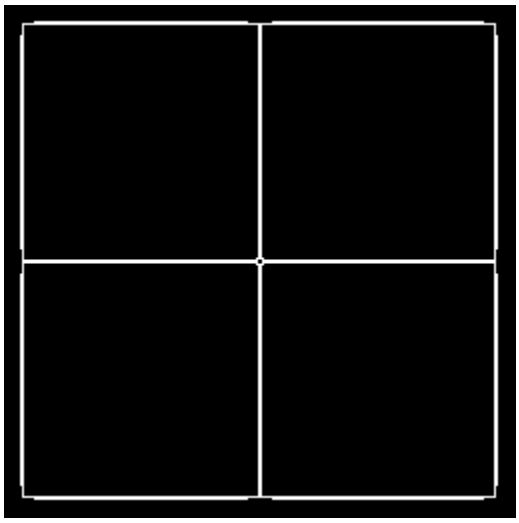
Jacob Kerstetter
4406239

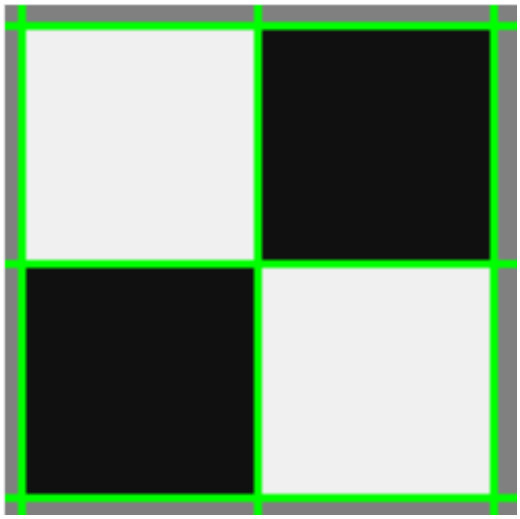
ECE 1390 - PS2 Report



ps2-1-a-1.png

Ps2-2-a-1.png

ps2-2-b-1.png



ps2-2-c-1.png

2d.

To find lines in this image, I used the parameters of RhoResolution, Theta, Threshold, and NHoodSize.

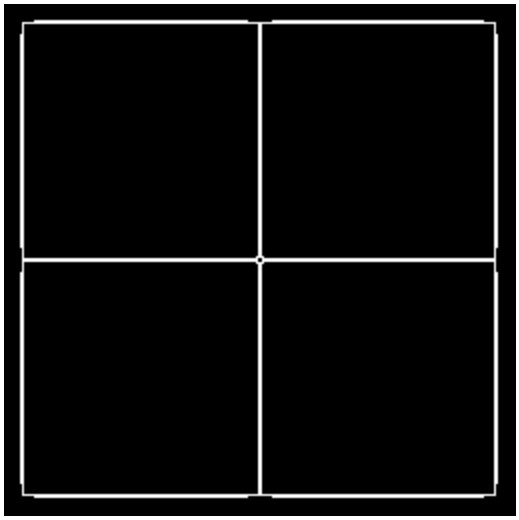
I found that a RhoResolution of greater than 1 affects the accuracy of the lines detected on the image. In this case, they appear at slightly off angles and do not match the true lines to 100% accuracy.

With Theta, as long as the peak values of -90 and 0 degrees were included in the set of parameters, the lines were detected with high accuracy as all of the lines in this image are strictly vertical or horizontal.

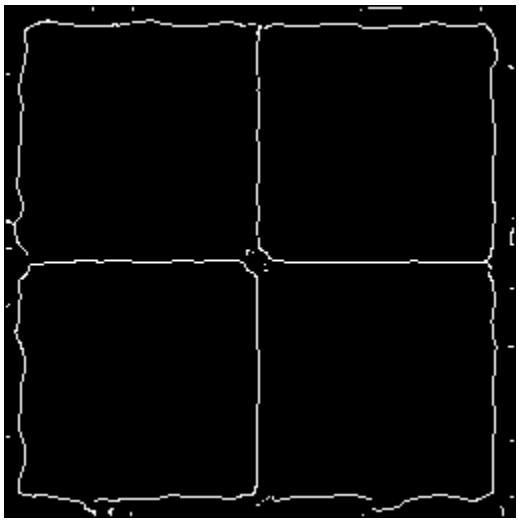
The threshold was less important for this image as all of the lines had strong vote tallies. However, putting the threshold to a value slightly above the default, say 180, makes only the most straight horizontal and vertical lines appear.

Finally, neighborhood size did not make a lot of difference in this image since the lines are fairly spaced out in terms of rho and theta indices.

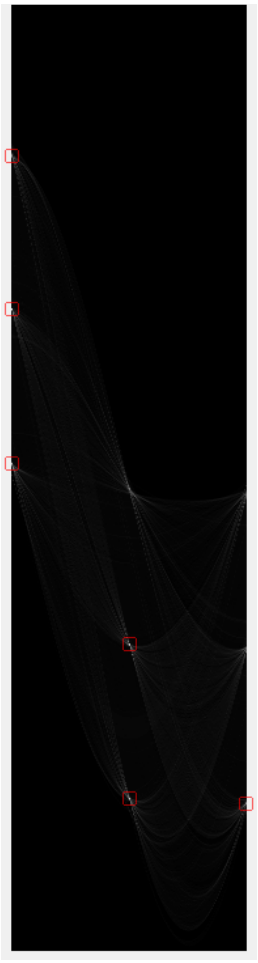
ps2-3-a-1.png



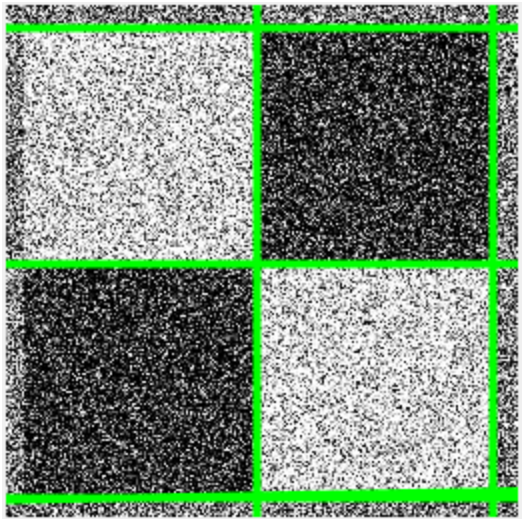
ps2-3-b-1.png



ps2-3-b-2.png



ps2-3-c-1.png

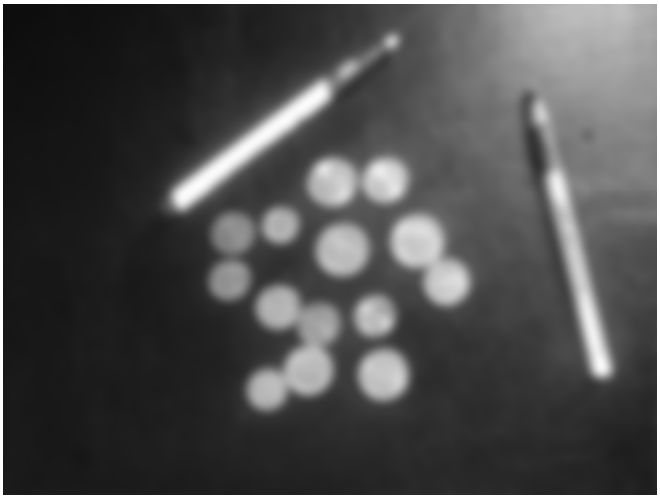


ps2-3-c-2.png

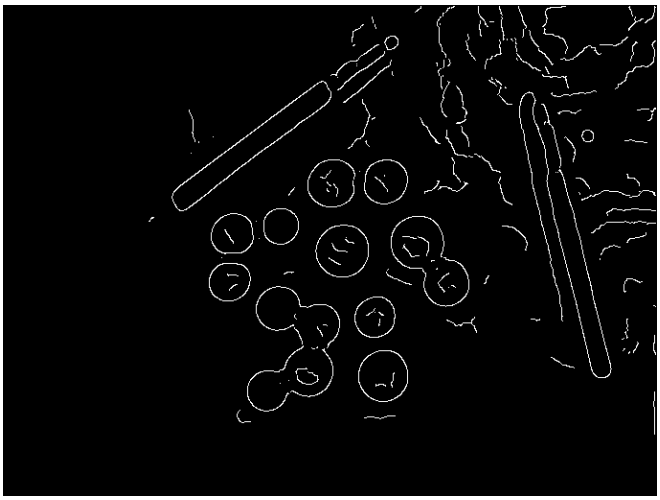
3c.

With my smoothing of sigma set to 5, the most important parameter to change was the threshold. The peaks were much less powerful in this image, so the threshold needed to be lower and specific to get rid of less powerful peaks. The result of this was only the straightest vertical and horizontal lines are shown in the image instead of slightly skewed ones layers on top.

When the RhoResolution was too large and the Theta values too spaced out, the algorithm had a difficult time finding all or precise lines. The algorithm was unable to pick up the leftmost vertical line in the image with a smoothing sigma value of 5 because, as shown in the edge pixel image, the line is very inconsistent.

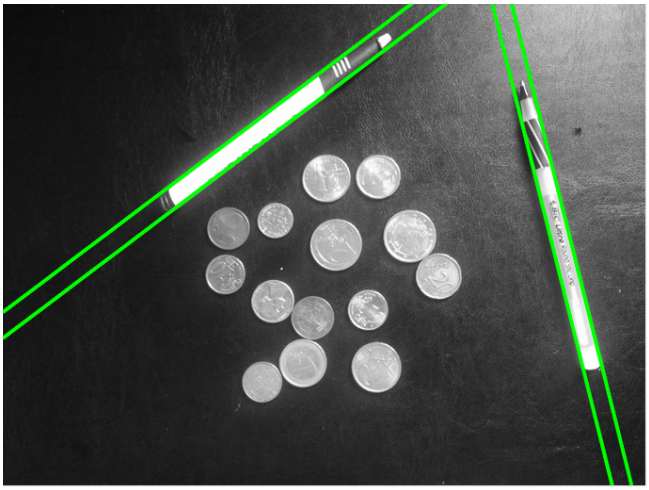


ps2-4-a-1.png



ps2-4-b-1.png

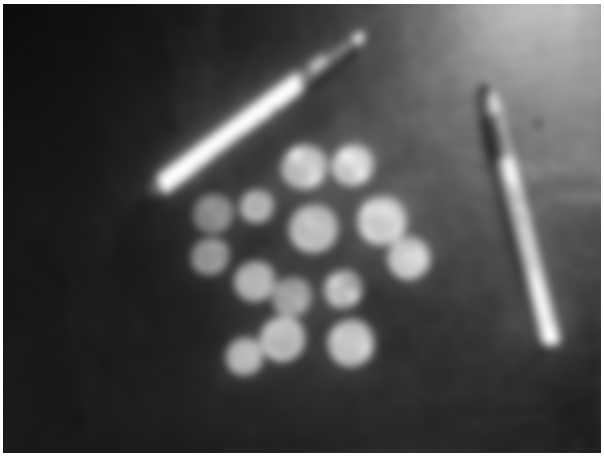
ps2-4-c-1.png



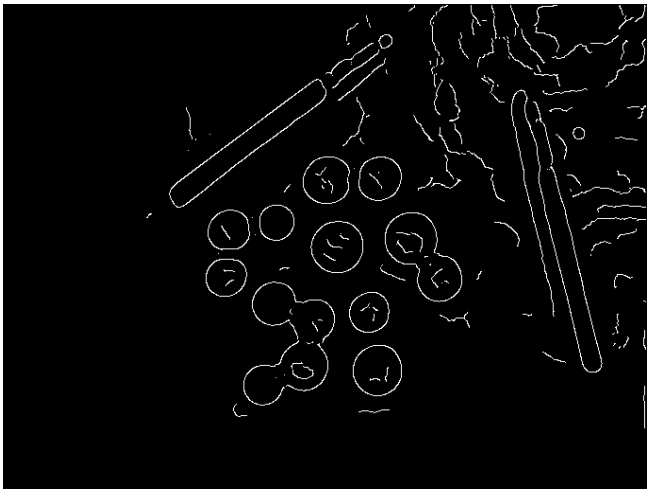
ps2-4-c-2.png

4c.

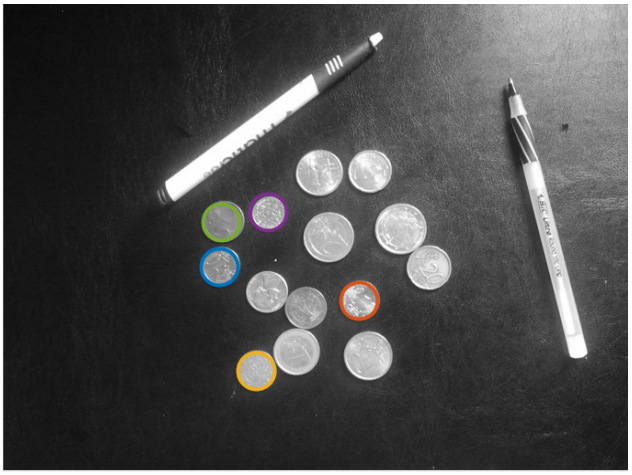
To get the best result for this image, the neighborhood size had to be smaller. With the default neighborhood size, the lines on both sides of each pen would not show up. Rather, only one was detected because the other peaks were ignored. With a smaller neighborhood size, say [5, 5], I was able to detect lines on both sides of the pens.



ps2-5-a-1.png



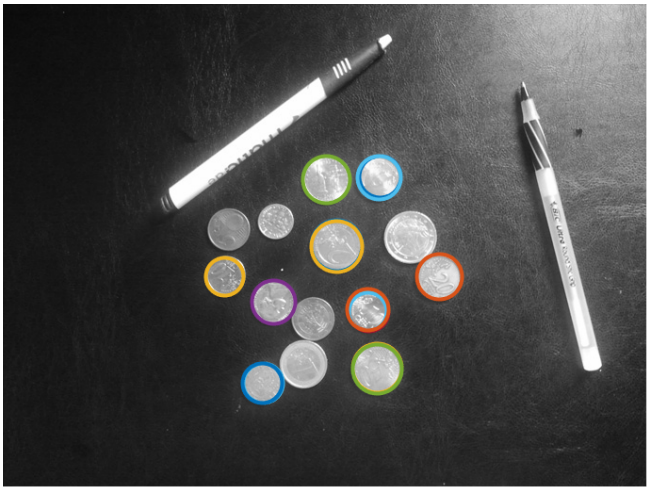
ps2-5-a-2.png



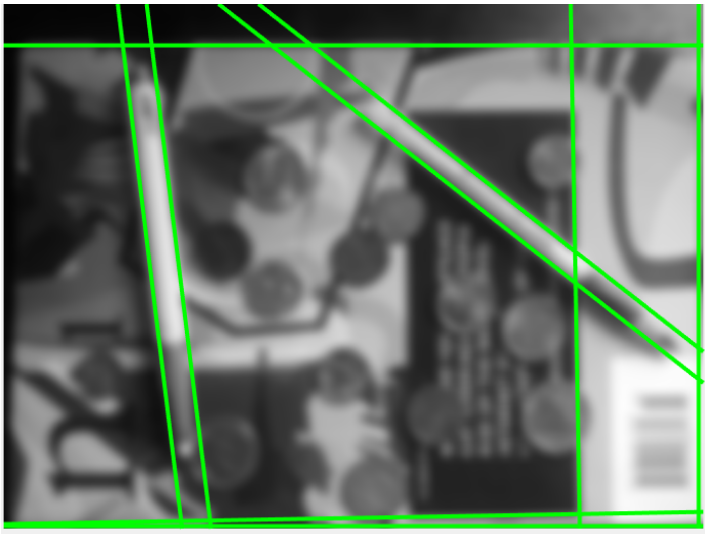
ps2-5-a-3.png

5b.

To find the circles, I had to first write the accumulator. This votes for every circle that each edge point could be a part of. This is run for a range of radii, in this case from 20 to 50. Then, the peak finding function finds the circles that received the most votes. In this case, the top 10 peaks were found. These circles were plotted and are found to match many of the coins in the image.



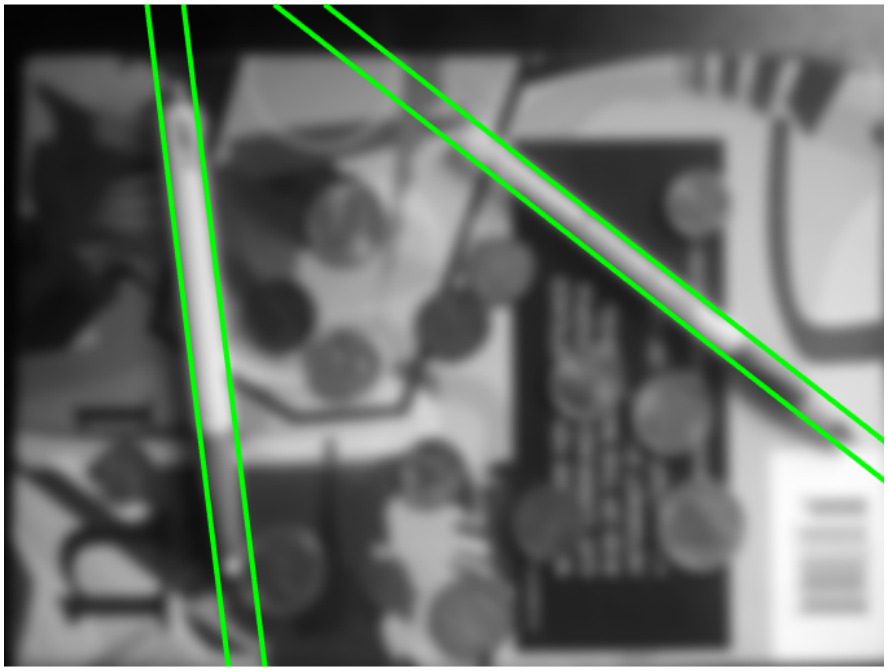
ps2-5-b-1.png



ps2-6-a-1.png

6b.

The problems present are that there are many boundaries between light and dark colors in the background that resemble edges. These are then detected as lines by the Hough algorithm, and may even have higher strength than the pens that are the true objects to be detected.



ps2-6-c-1.png

ps2-7-a-1.png

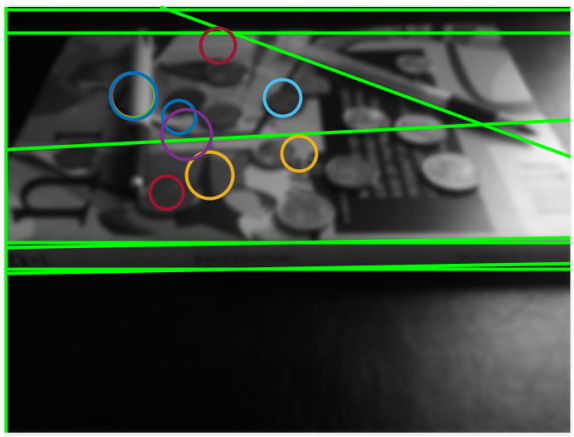
7b.

The biggest false alarms in this image are the rounded edges on the letter in the background. To get the detector to not detect these as circles, I firstly optimized the smoothing of my Gaussian filter by setting the sigma to 5 (line 181).

```
181    img3_smoothed = imgaussfilt(img3_grayscale, 5);
```

Finally, I altered the radius range from 20 to 50, to 30 to 70. This found more accurate radii for the coins (line 187).

```
186    % find circle and draw them
187    [centers, radii] = find_circles(img3_edges, [30 70]);
```



ps2-8-a-1.pn

8b.

To fix this problem of distorted circles, we can alter our algorithm to use the equation for an ellipse. Because of the angle of the image, the circles do not appear as true circles, and have some skew to them. The equation of an ellipse is similar to that of a circle, just adding one extra parameter for each the x and the y coordinate. This would allow us to find the distorted circles as ellipses.