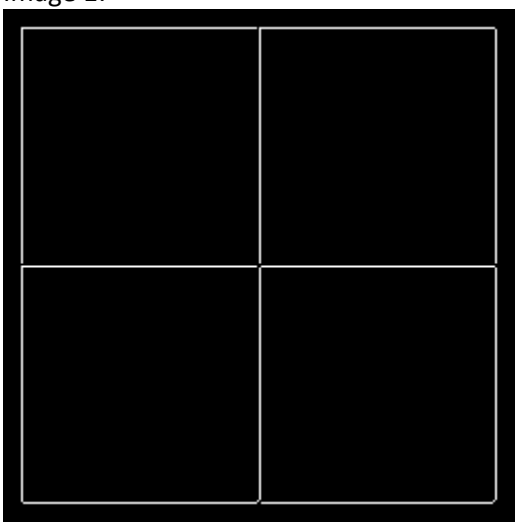
James Liu
CS 4495, Fall 2014
Problem Set 1

Problem Set 1, Question 1, Part A.
Image 1:

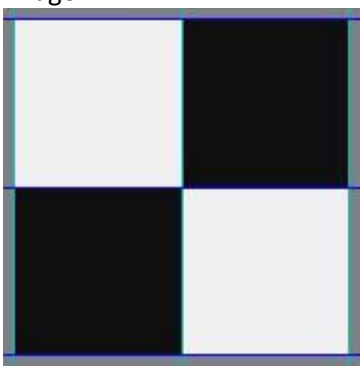


Edge Image, generated by the Canny edge operator

Problem Set 1, Question 2, Part A.
Image 1:

Hough accumulator array of the edge image, peaks marked in red.

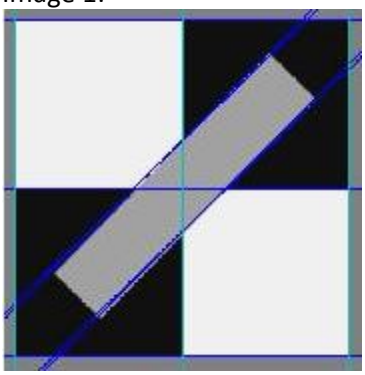
Image 2:



Original grayscale image with detected lines marked in green and blue.

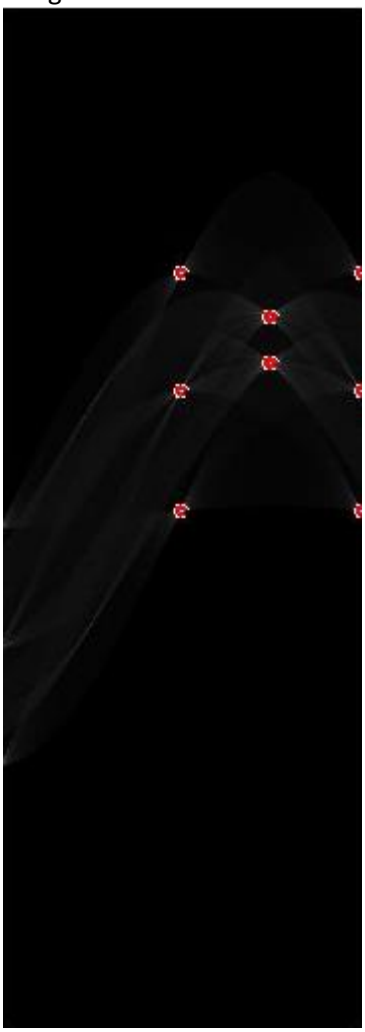
I decided to use a bin size of d=2, theta=1 degree. Theta I chose on a whim, expecting a 1 degree difference to be enough to capture perfectly horizontal/vertical lines. The d bin size was found through trial and error. d< didn't catch more than one signifigant line, and d>=4 started generating excess number of lines adjacent to the correct ones.

Problem Set 1, Question 3, Part A.
Image 1:



Original grayscale image with detected lines marked in green and blue.

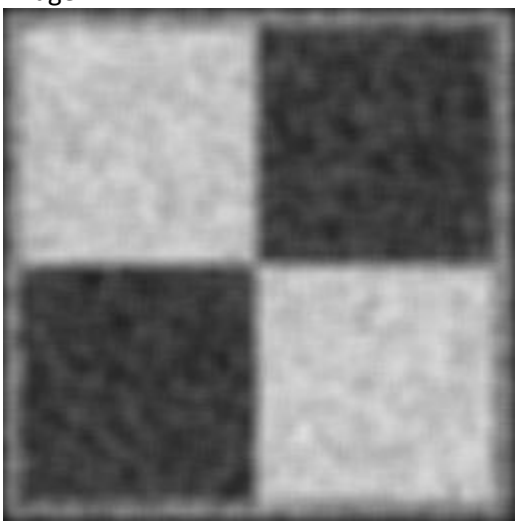
Image 2:



Hough accumulator array of the edge image, peaks marked in red.

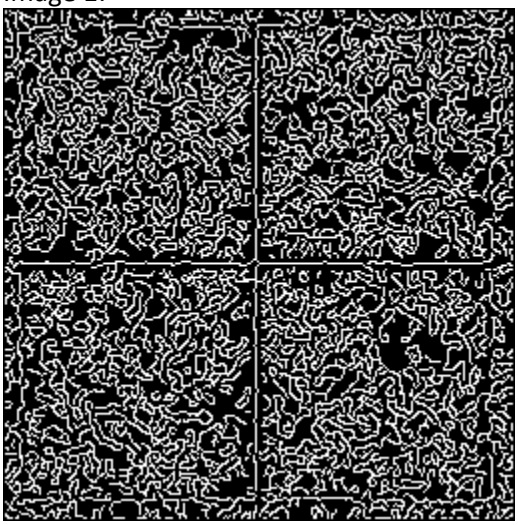
The previous settings worked in finding the diagonal lines, so I kept them.

Problem Set 1, Question 4, Part A.
Image 1:



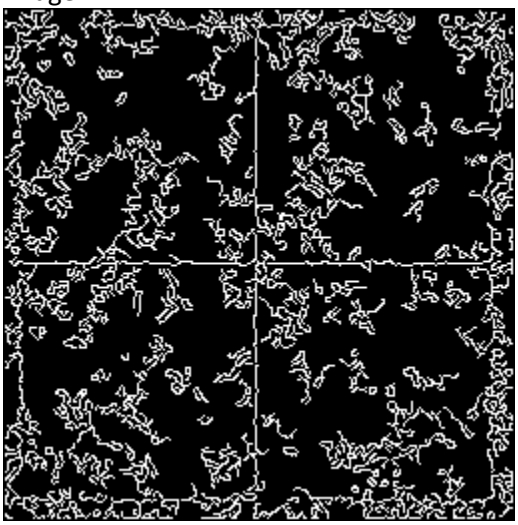
Smoothed grayscale image

Problem Set 1, Question 4, Part B.
Image 1:



Edge image generated by the Canny edge operator on the raw grayscale image

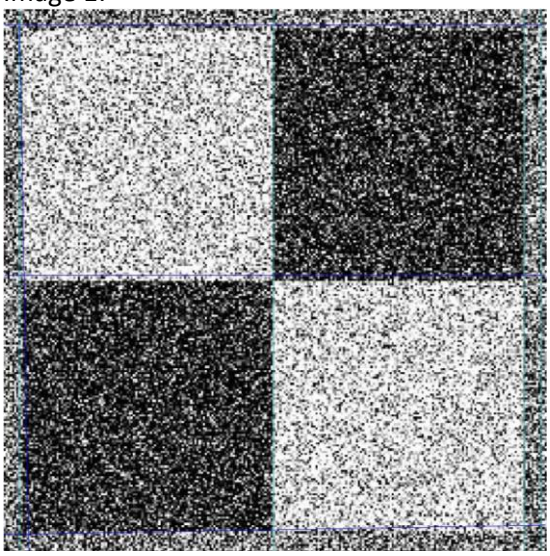
Image 2:



Edge image generated by the canny edge operator on the smoothed image

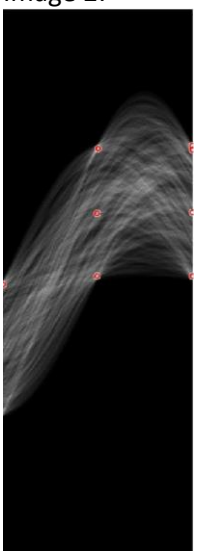
Problem Set 1, Question 4, Part C.

Image 1:



Detected lines plotted on in green and blue on top of the original image

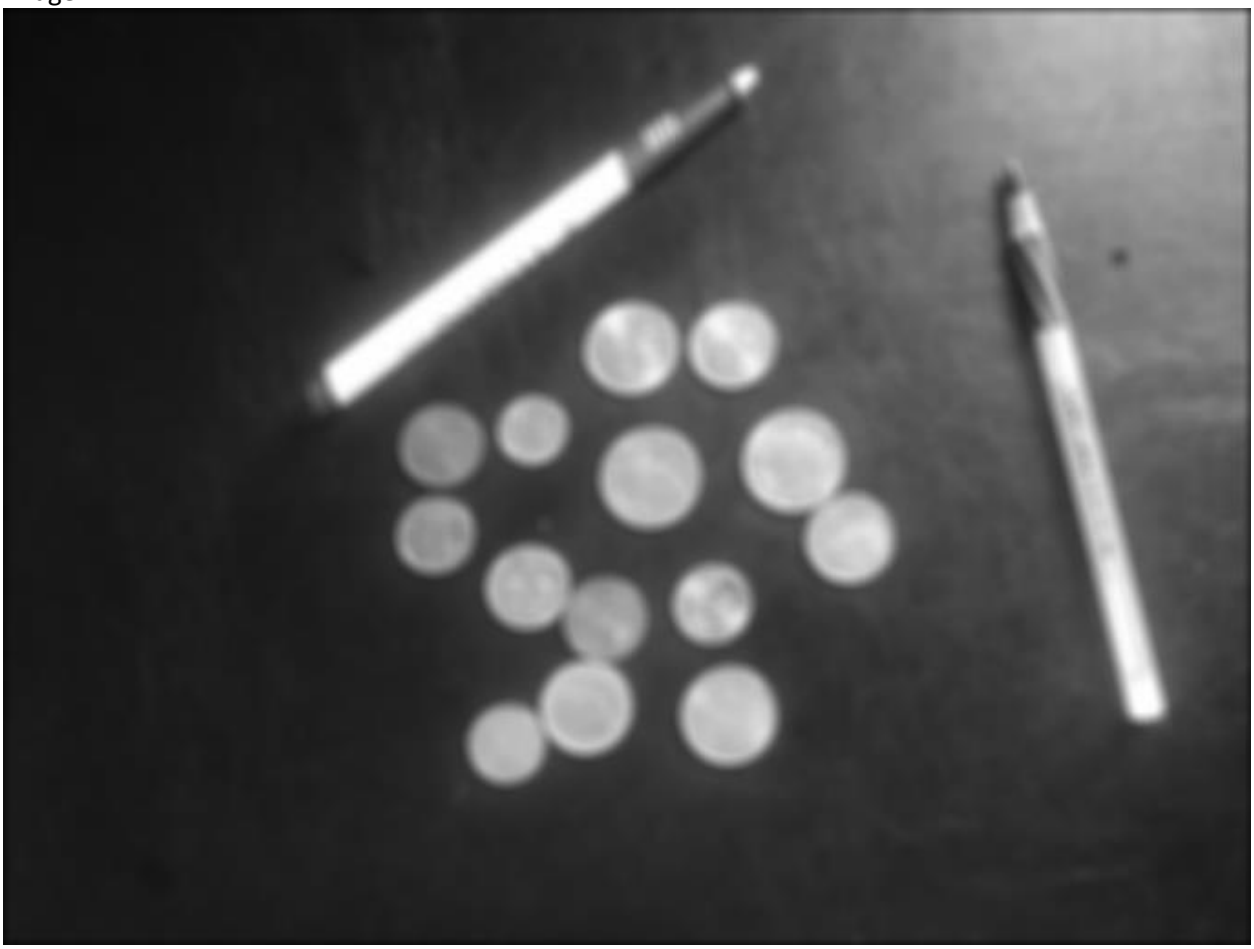
Image 2:



Hough accumulator array of the edge image, peaks marked in red.

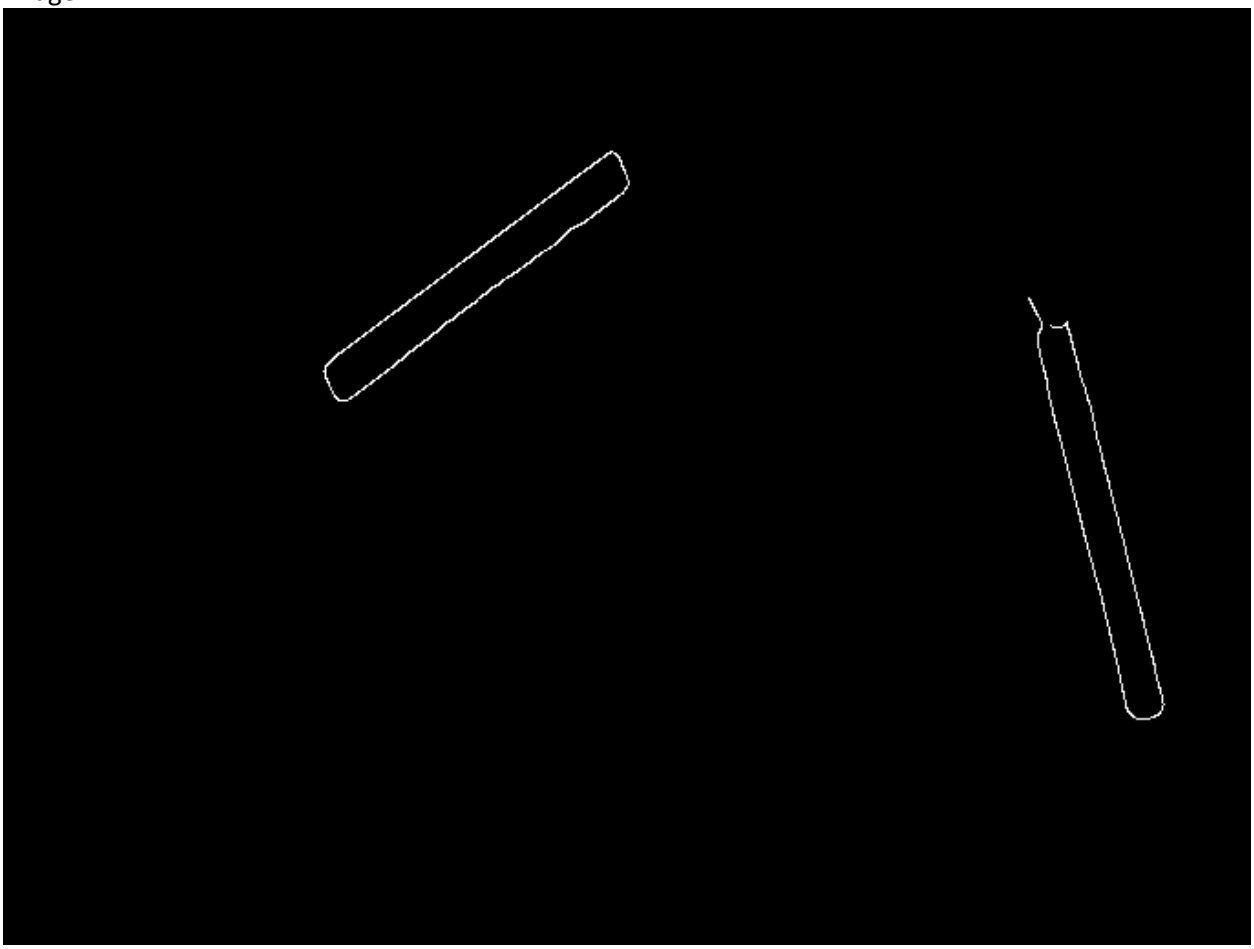
I had to tweak the smoothing filter's size and sigma, the canny edge operator's threshold, the accumulator bin sizes, and the peak threshold for which points are considered lines until a valid result appeared.

Problem Set 1, Question 5, Part A.
Image 1:



The grayscale smoothed image

Problem Set 1, Question 5, Part B.
Image 1:

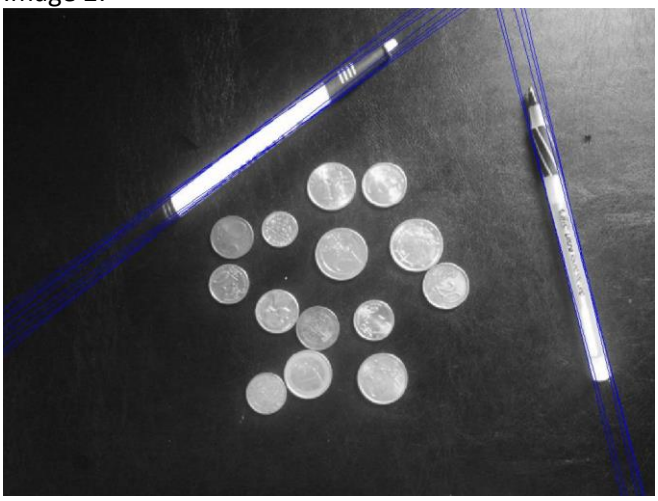


The edge image produced by the canny edge operator on the grayscale image.

Problem Set 1, Question 5, Part C.
Image 1:

Hough accumulator array of the edge image, peaks marked in red.

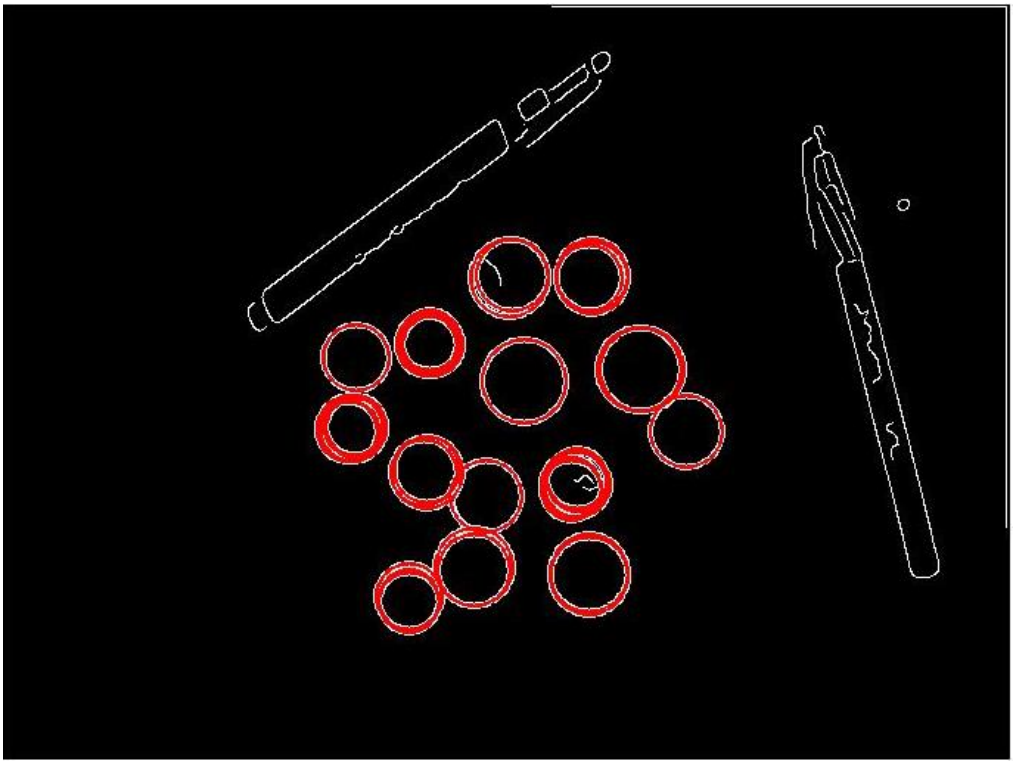
Image 2:



Detected lines plotted on in green and blue on top of the intensity image.

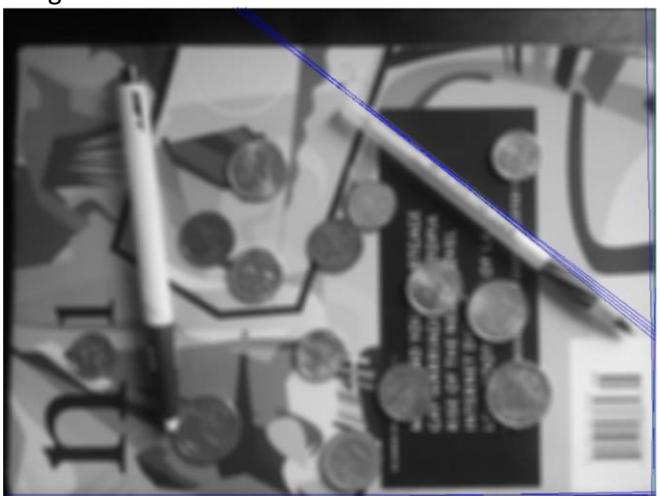
The detector was having trouble detecting the pens with the coins in the way, so I increased the smoothing filter and edge operator's settings until the coins were removed from the edge image. I knew the pens would remain with most of the line intact, as, even when smoothed heavily, the difference in intensity was great enough to be not be removed before the coins. Certain features of the pen, such as the actual writing point and the black upper sections were removed but the bright white/gray areas remained.

Problem Set 1, Question 5, Part A.
Image 1:



I had to alter the way it was parameterized to allow storage of a X,Y coordinate and a radius. Then alter the main detection loop to use a circular voting method. (I didn't opt for the gradient method). I also had to increase the bin sizes signifigantly, as it low bin sizes increased run time and occasionally made me run out of memory. Finally, I needed to swap the coordinates {Y,X} instead of {X,Y} due to the way the plotting function works.

Problem Set 1, Question 7, Part A.
Image 1:



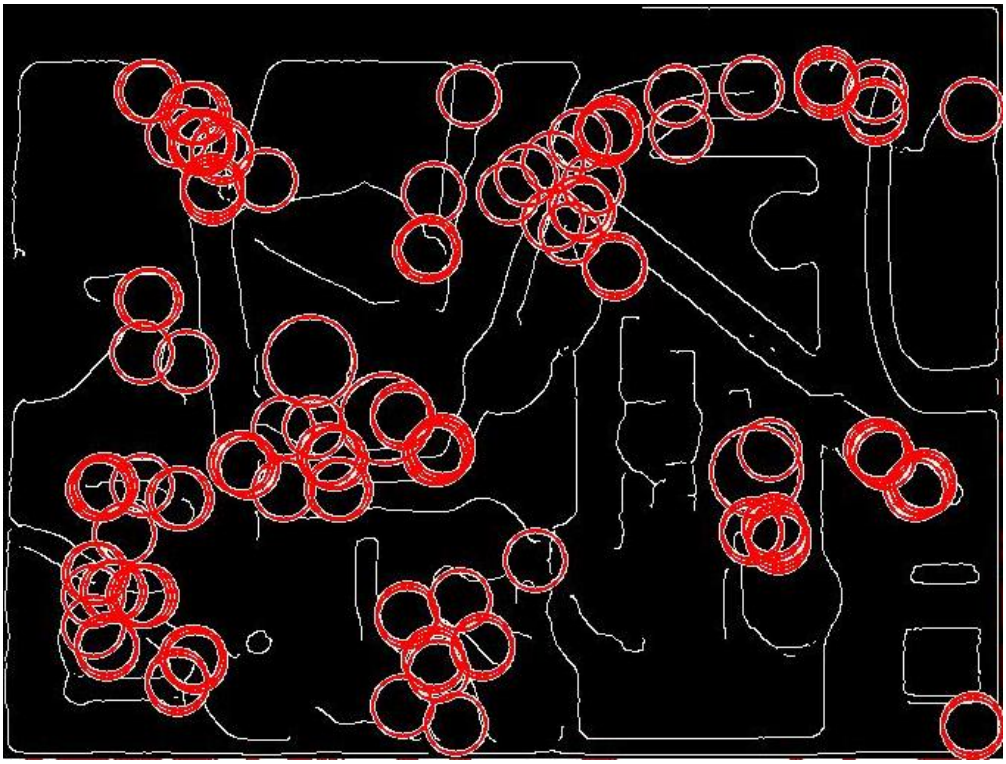
Smooth image with detected Hough lines drawn on top.

Problem Set 1, Question 7, Part B.
With the clutter added in there are other lines that, even after smoothing and edge detection, that present more evidence to the Hough line detector than the pen does, especially edges that are inherently longer than the pen's. With the white background, parts of the pen do not appear to the edge detector: there's not enough change in intensity, which makes the pen seem shorter than it really is.

Problem Set 1, Question 7, Part C.
Image 1:

Better adjusted smoothed image with hough lines drawn on top.

Problem Set 1, Question 8, Part A.
Image 1:



My best attempt at finding the coins in the image. 3 of the 14 coins were found.

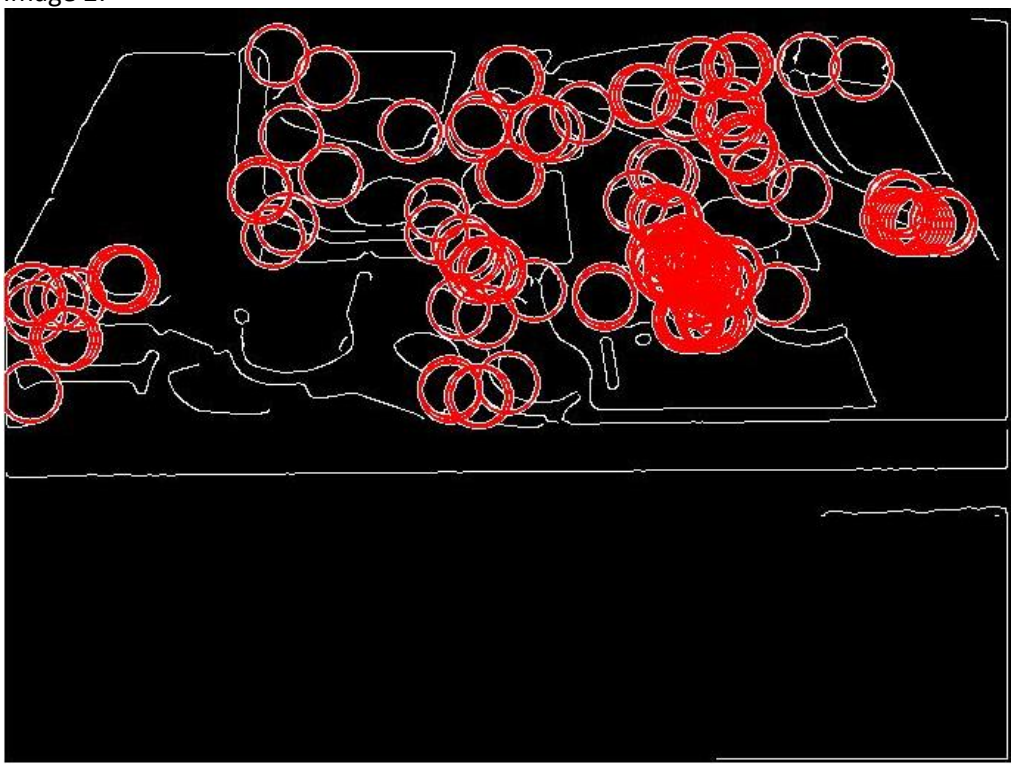
There are a lot of false positives. In previous attempts, the edge detection kept detecting the text on the card below, which concentrated over 80% of the detected circles. Decreasing the threshold would keep the blob of detected circles there, and increasing the smoothing or the threshold on the edge detector removed parts of coins. The latter proved a bit more successful than the former.

Problem Set 1, Question 9, Part A.

Image 1:

Attempted line detection on the distortion image

Image 2:



Attempted circle detection on the distortion image

Problem Set 1, Question 9, Part B

Add an additional parameter to the circle detection algorithm, turning the circle detection algorithm into an ellipse detection algorithm. An ellipse requires x, y, a, b parameter, instead of the circle's x, y, and r. This would allow for the algorithm to account for circles rotated in 3D space, as their projection onto the camera is simply an ellipse or a line in 2D space. This would increase the runtime significantly, however.