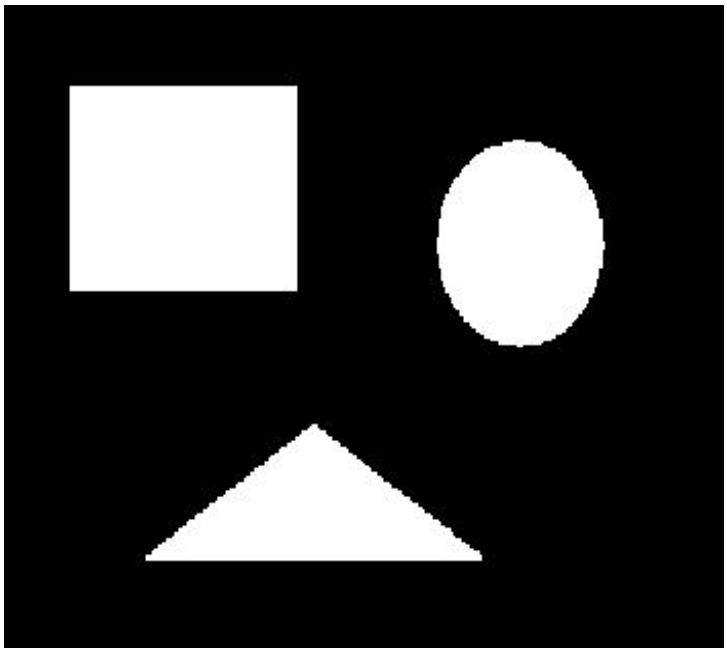
Task1: Morphology Image Processing

1a.

res_noise1.jpg



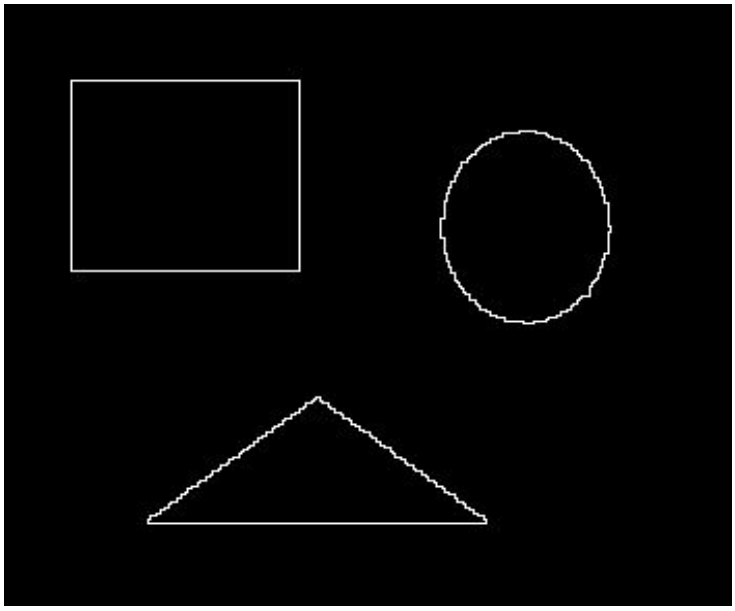
res_noise2.jpg

Two morphology algorithms used to remove noises from the given image were:
i. closing followed by opening.
ii. opening followed by closing.
The output of the two algorithms were saved as ‘res_noise1.jpg’ and ‘res_noise2.jpg’ respectively.

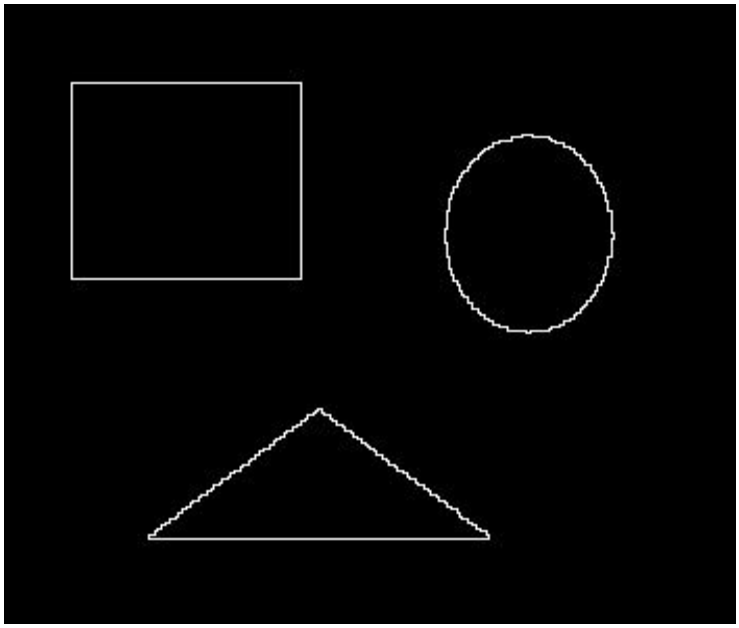
For performing these operations, I used a 3x3 kernel with all 1s, in other words, a square shaped structuring element.
To perform closing, I performed a dilation operation followed by an erosion operation using my 3x3 kernel.
To perform opening, I performed an erosion operation followed by a dilation operation using the same kernel.

1b.
As seen from the resultant images of task1.a, it can be concluded that there is no major change in the sizes and shapes of the objects in the original image. Moreover, in both the output images, the noise has been completely removed. Therefore, both the morphology algorithms are efficient in removing noise.
Though the biggest question still remains- Are the two images completely same?
The answer is- No.
Although the two images are almost similar in the shapes and sizes of their objects, there are some very minor differences in the boundaries of the objects (if you look closely). There are minor distortions in the boundaries of oval-shaped objects and triangle shaped objects specifically.

1c.



res_bound1.jpg



res_bound2.jpg

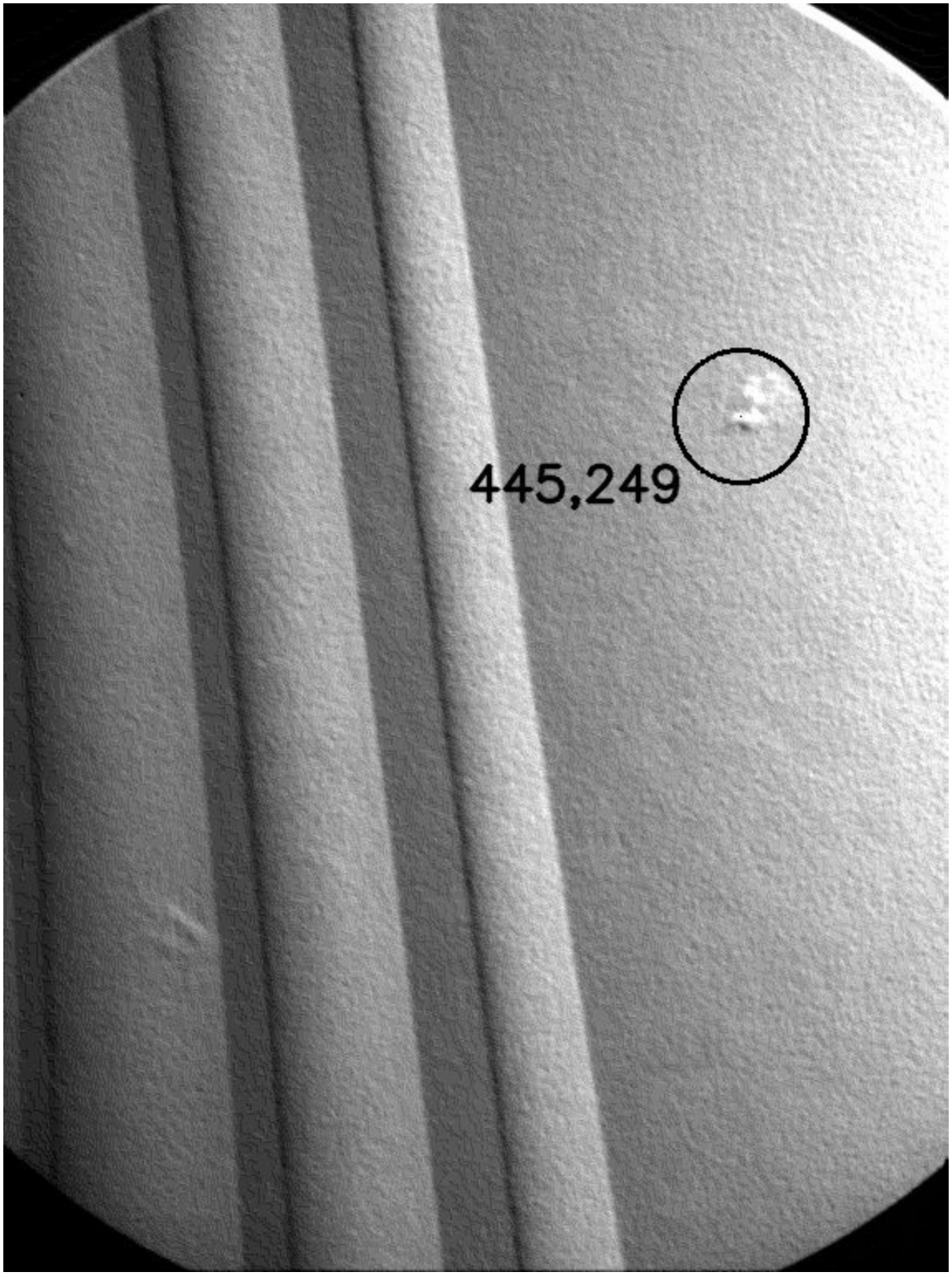
After performing the two morphology algorithms, I got two noiseless images, namely ‘res_noise1.jpg’ and ‘res_noise2.jpg’. I used these two images to carry out boundary extraction.
For extracting boundaries from res_noise1.jpg, I used:
res_bound1=res_noise1- erosion(res_noise1)

For extracting boundaries from res_noise2.jpg, I used:
res_bound2=res_noise2- erosion(res_noise2)
where, res_bound1.jpg and res_bound2.jpg are my output images for this task

Task2: Image Segmentation and Point Detection

2a.

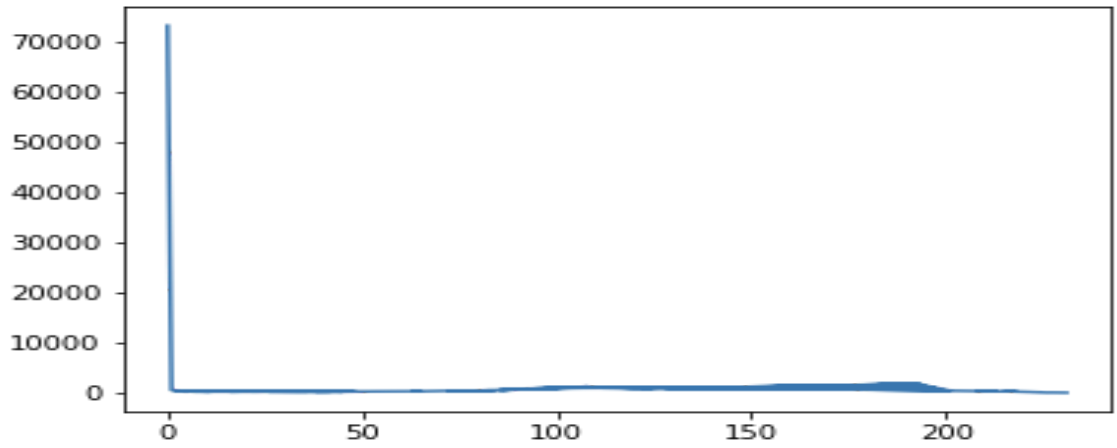
point.jpg



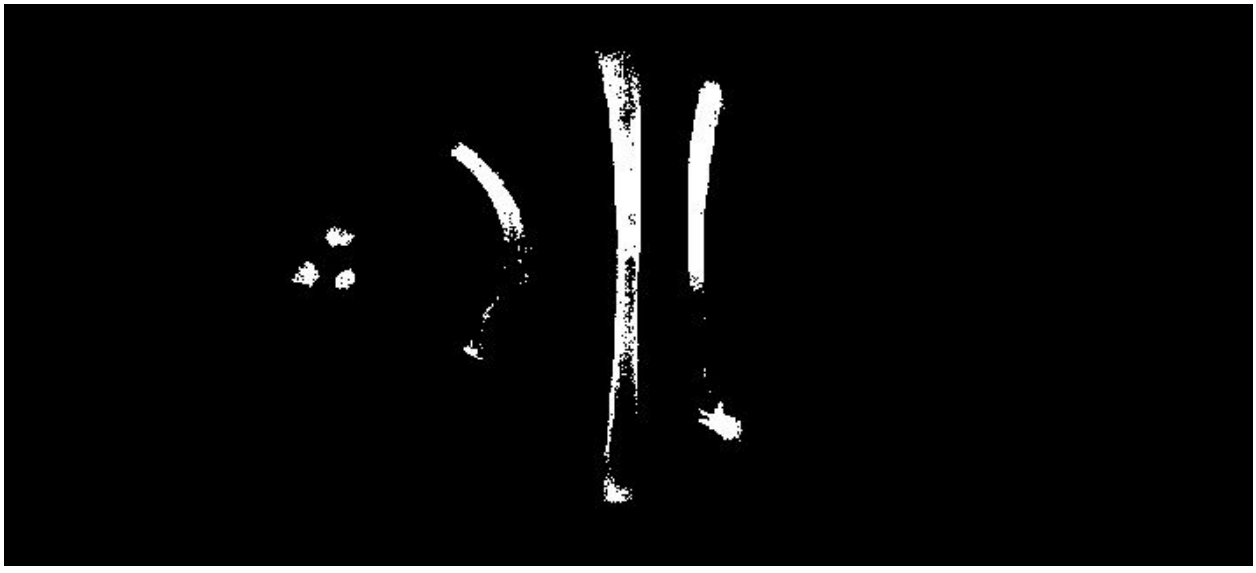
point.jpg(on real image)

For point detection, I performed a correlation on the given image using a 3x3 kernel with values
[[-1,-1,-1],
[-1,8,-1],
[-1,-1,-1]]
After performing correlation, I took absolute value of all the pixel values in the image. Having done that, I performed thresholding of the image, such that all the pixel values above the threshold value are set to 255 and all the pixel values below the threshold get set to 0.
For thresholding, I chose '1100' as the threshold value.
All the points above the threshold value were stored. From the result it can be seen that there was just one such point, which is our detected point.

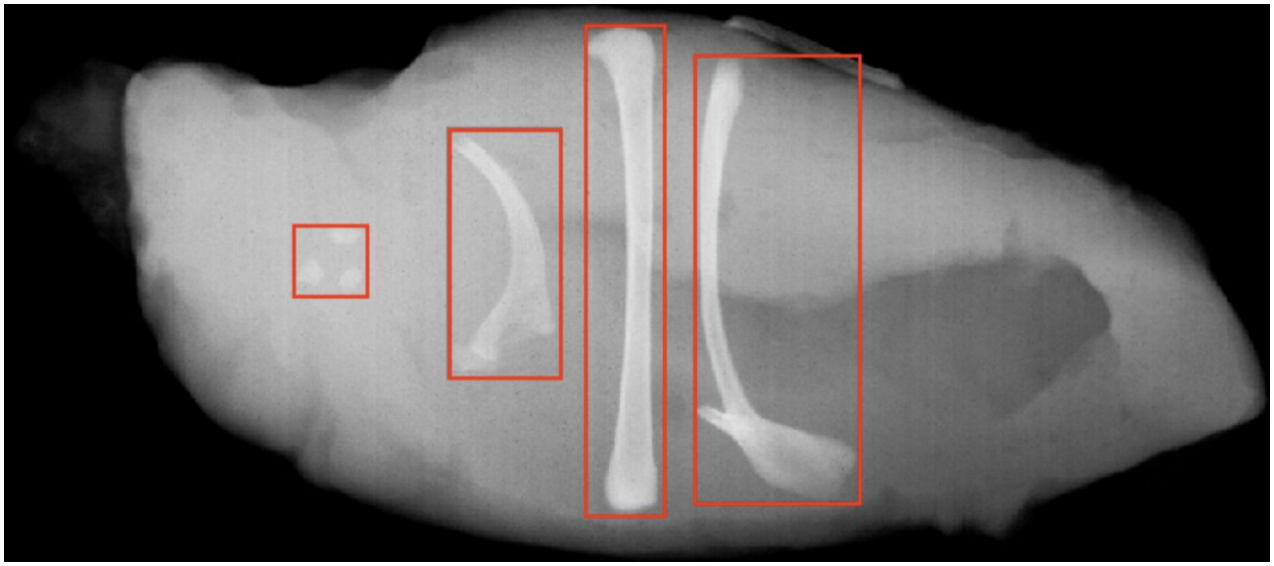
2b.



histogram.png



segment.jpg



boundingbox.jpg

For task 2b, I first plotted the histogram of intensities v/s number of pixel counts.
From the histogram, I concluded that there are two peaks, one at 0 intensity, which has a peak as high as the highest number of pixel counts, i.e., 70000. Another peak is close to 200 intensity value. Therefore, through the method of hit-and-trial, I started putting my threshold value from 195 intensity onwards and found that at intensity value 204, the bones in the object had started to appear. Considering this threshold as my optimum threshold, I generated my segmented image.
For bounding boxes, I drew four bounding boxes for four different objects at following coordinates:
(considering objects from left to right)
Object 1:
((324,447),(405,447),(405,370),(323,370))
Object 2:
((497,259),(497,537),(620,540),(620,262))
Object 3:
((648,144),(648,694),(735,694),(735,142))
Object 4:
((772,177),(772,683),(953,679),(953,176))

Task3: Hough Transform

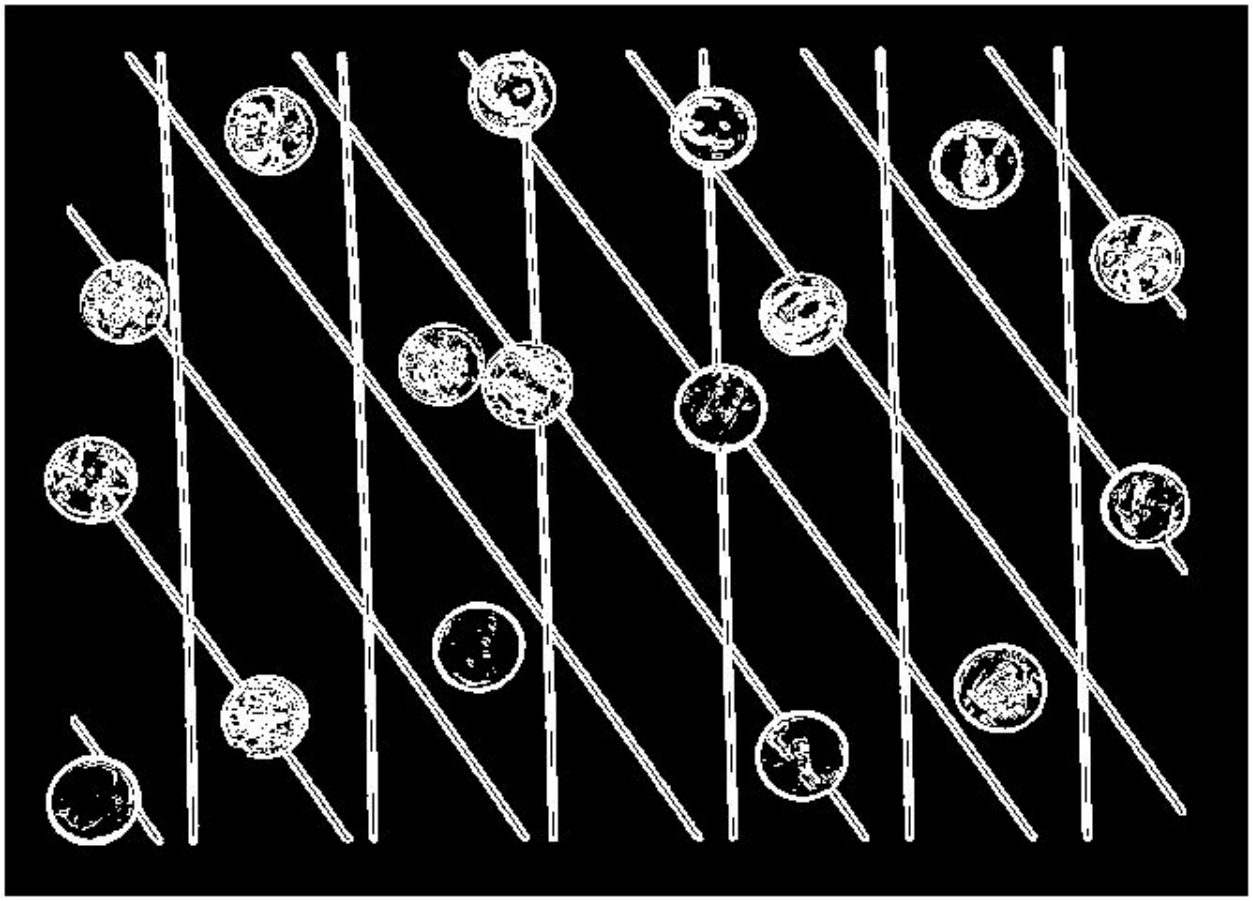
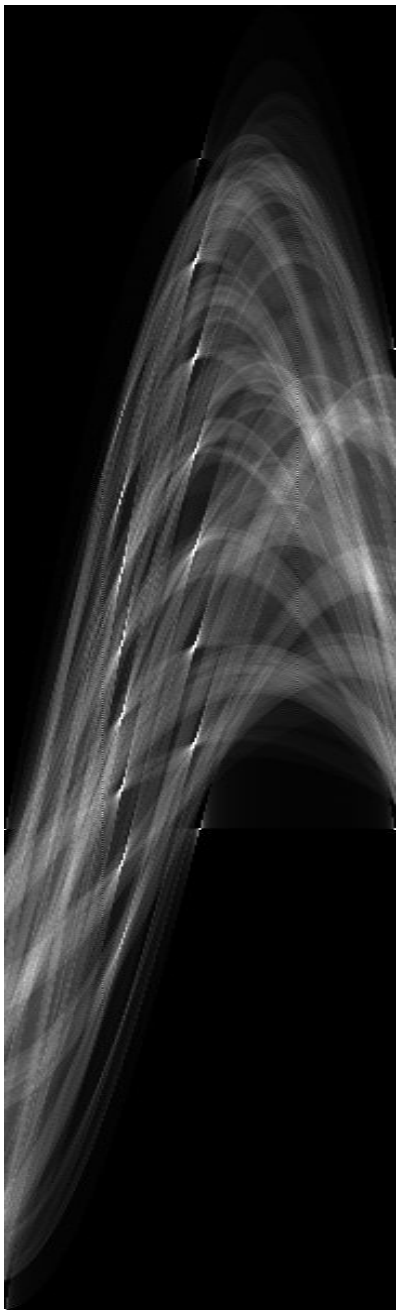


Image after performing edge detection and thresholding

For performing hough transform, my first step was to perform edge detection of the given image using 3x3 sobel kernel. After performing edge detection, I did thresholding of the image such that the the pixel values above the threshold value were set to 255 and the ones below it were set to 0. For this purpose, I chose the threshold value to be 25. Threshold value of 25 gave me the desired image as above.

Next step was to calculate values of d and theta for every pixel (x,y) which has a value of 255,
d= xcos(theta)-ysin(theta) , where theta>=0 and theta<=180
Using the values of d and theta, I create an accumulator array that has the voting value of every (d,theta) pair, which results in the sinusoidal wave as depicted in the image. The bright points in the sinusoidal wave depicts the coordinates with high voting value.



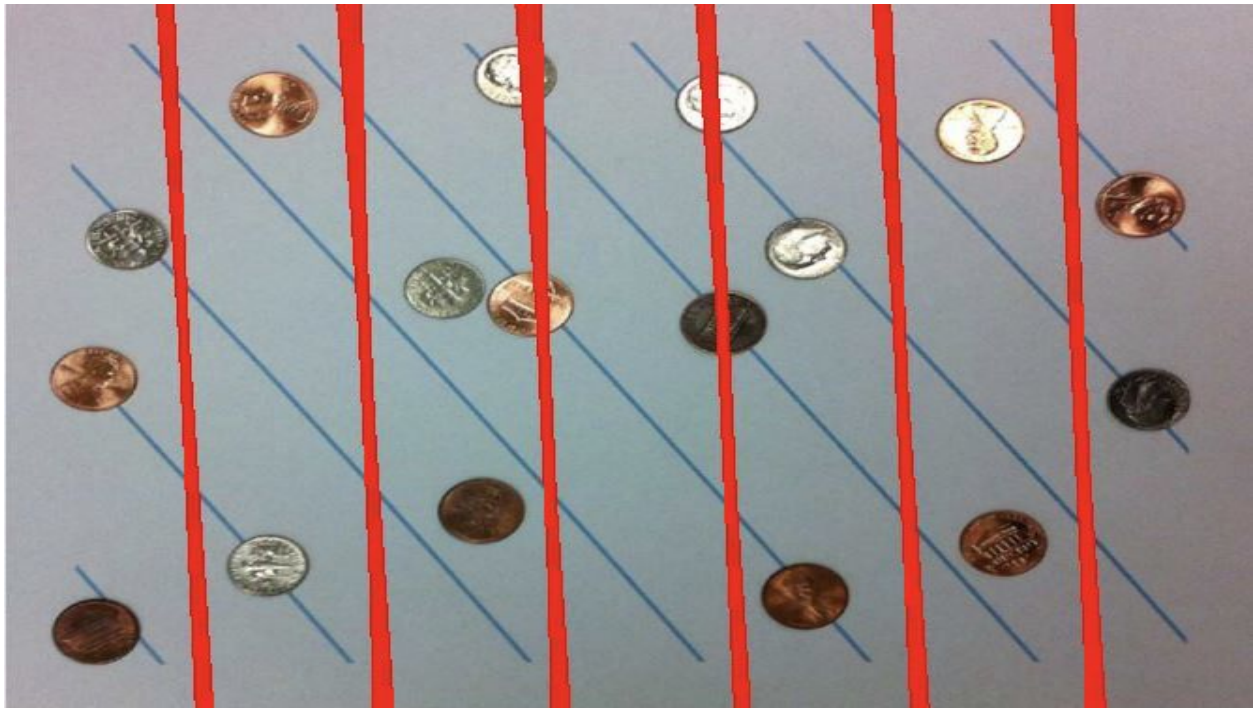
Sinusoidal wave representing voting value of every (d,theta) pair. Bright points represent the points with high voting value.

Next step is performing thresholding of accumulator array. Thresholding is done to distinguish points with low voting value from high voting value. I set the threshold value such that I get the coordinates of the points with high voting value. The coordinates of these points are then used to detect lines on the given image.

I observe the theta values of the coordinates with high voting value. I choose two set of theta values such that one set of theta values has values close to 90 degree (for vertical lines) and

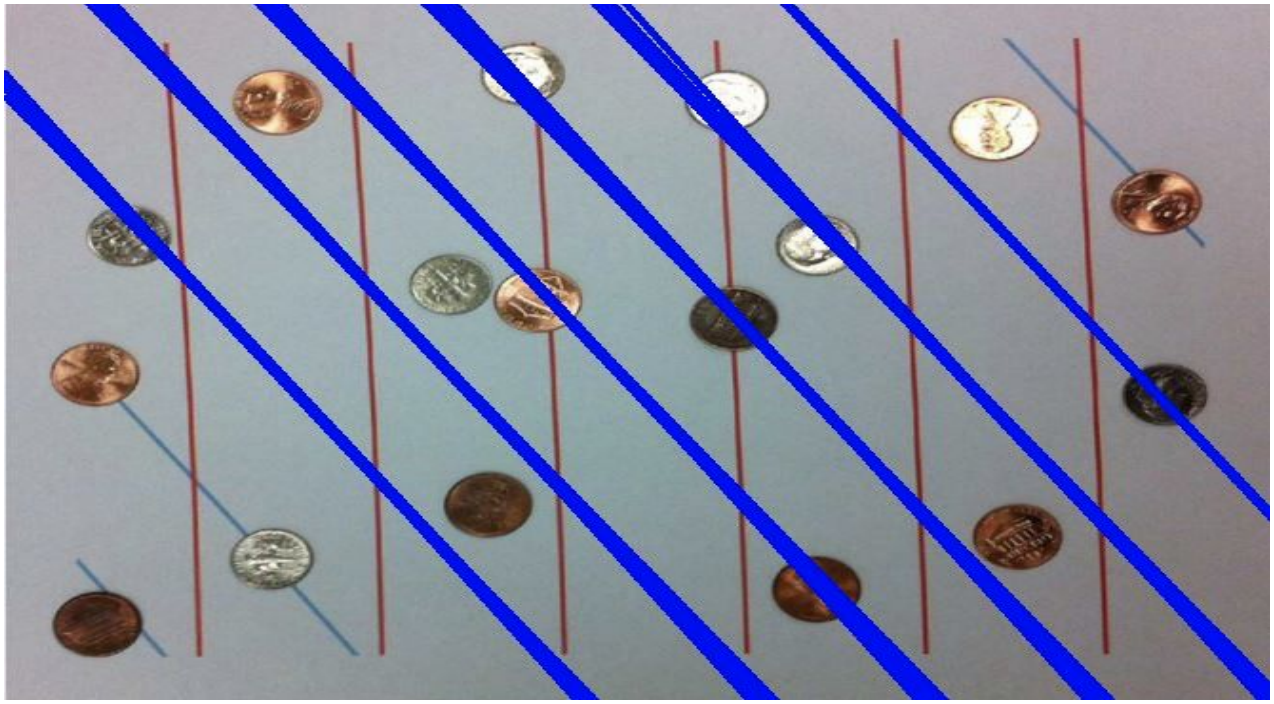
other set of theta values has values lesser than 90 degree(for diagonals). Therefore, I get two images as follows.

3a.



(Total red lines detected - 5/5)

3b.



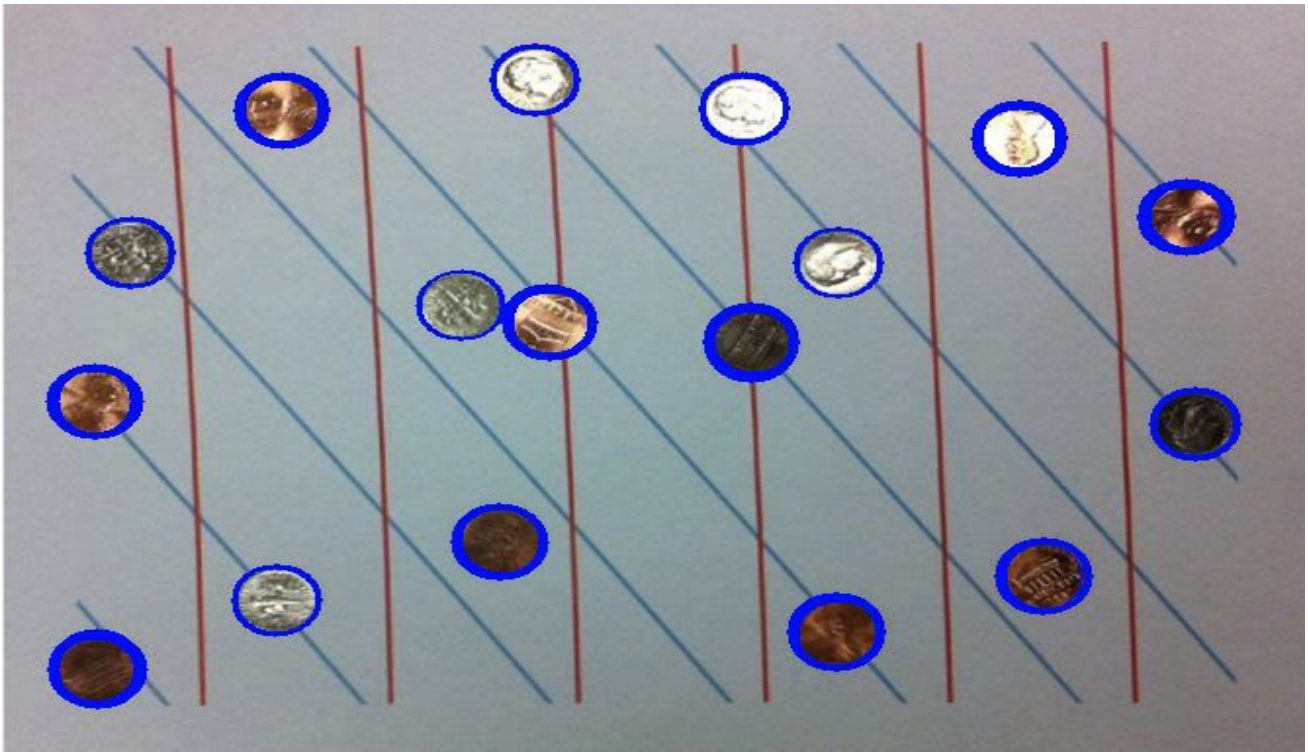
(Total blue lines detected - 6/9)

3c.[BONUS TASK]
For this task, I started with the edge detection of the original image using a 3x3 sobel operator and then thresholding of the output image with the threshold value of 25. (these two steps are the same as 3a and 3b).

Next step is to calculate the value of a, b, r for every pixel (x,y) of the image which has a pixel value of 255, using these formulae:
a = y - (r * (cos(theta)))
b = x + (r * (sin(theta)))
where a, b is the center of the circle, r is the radius.
such that,
r>=22 and r<=25 ,
theta>=0 and theta<=360

Using the values of a,b and r, I create an accumulator array that has the voting value of every (a,b,r) coordinates.

Last step is performing thresholding of accumulator array. Thresholding is done to distinguish points with low voting value from high voting value. I set the threshold value such that I get the coordinates of the points with high voting value. The coordinates of these points are then used to detect coins on the given image.



(Total coins detected - 17/17)