CSE 573 - Project 3

Amlan Gupta (#50288686)

December 2018

1 Morphology image processing

a **Using two morphology image processing algorithms to remove noises of the image below, save the results and name as 'res_noise1.jpg' and 'res_noise2.jpg'**

Figure 1: res_noise1.jpg using Opening

Figure 2: res_noise2.jpg using Closing

What is Opening?
Opening is the dilation of the erosion of image A by a structuring element B.
$A \circ B=(A \ominus B) \oplus B$, where $\ominus$ and $\oplus$ denote erosion and dilation, respectively

What is Closing?
Closing is the erosion of the dilation of image A by a structuring element B.
$A \bullet B = (A \oplus B) \ominus B$, where $\ominus$ and $\oplus$ denote erosion and dilation, respectively

For res_noise1.jpg, first closing and then opening was performed.

For res_noise2.jpg, first opening and then closing was performed.

Following Structuring element was used:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

b **Please compare the two results, and indicate are they the same in your report.**

As we can from figure 1 and figure 2 that, both the images are same.

Opening removed small noises from the foreground of the image, placing them in the background, while closing removes small holes in the foreground, changing small islands of background into foreground.

Since opening is the dual of closing, performing both the operation though in different order produced the same result.

c **Using morphology image processing algorithms to extract the boundary 'res_noise1.jpg' and 'res_noise2.jpg', save the results and name as 'res_bound1.jpg' and 'res_bound2.jpg'.**

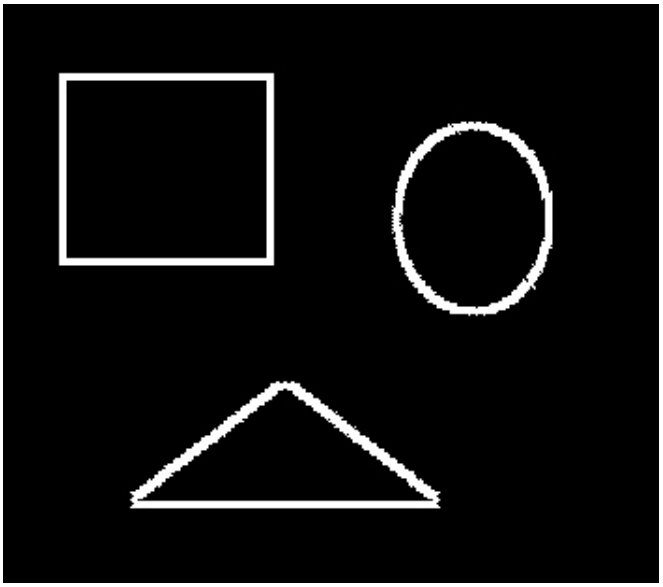

Figure 3: res_bound1.jpg

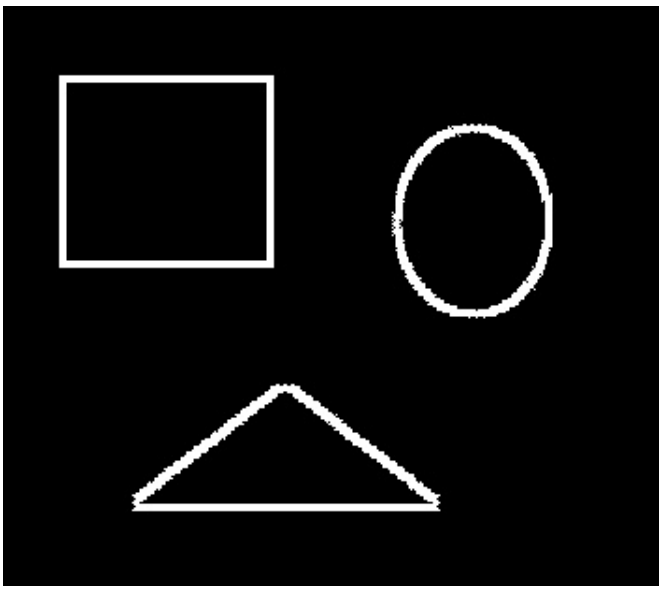

Figure 4: res_bound2.jpg

Boundary can be extracted with many methods, for example:

(a) Dilated image - Original image

(b) Original image - Eroded image

(c) Dilated image - Eroded image

In this case, we have first created two images by dilating and eroding, then subtracted between them. This was the border stays thick and easily recognizable.

2 Image segmentation and point detection

a **The 'point.jpg' is a X-ray image of a turbine blade with a porosity. Using the point detection algorithm, you learned to detect the porosity. Label the points and give its (if more than one their) coordinates on the image.**

Note: The below image has been used instead of point.jpg, as there might have been some changes due to lossy image compression.

http://sse.tongji.edu.cn/linzhang/DIP/demoCodes/ch8/pointDetec/turbine-blade.jpg

For point detection first we have applied Laplacian operator on the image first. Since this kernel is approximating a second derivative measurement on the image, it is very sensitive to noise.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

As the central position is heavily weighted compared to it's neighbors, if any noise is present in the center, it will be readily detected if we perform this mask.

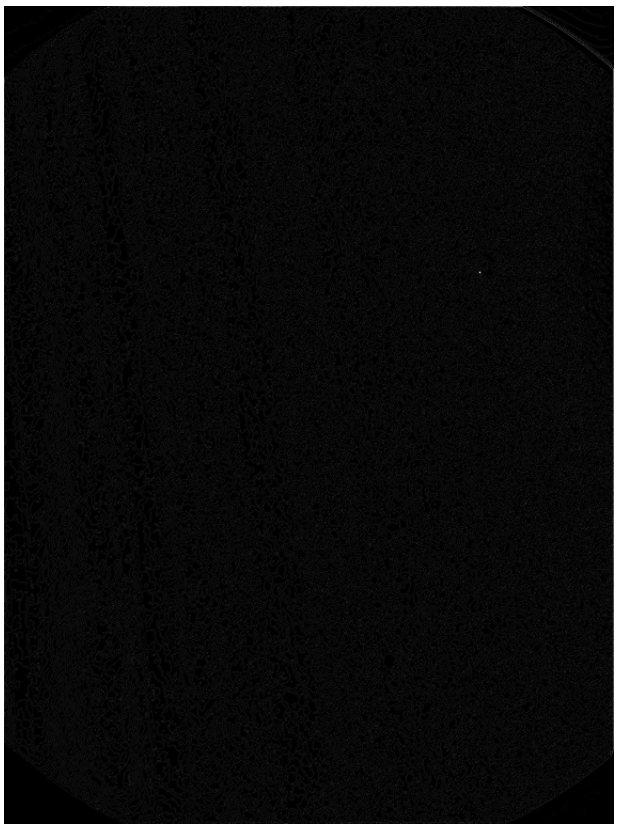

Figure 5: Result of Mask output

After applying the mask and normalizing we can see, the location of the porosity becomes the brightest pixel i.e 255 and the second brightest pixel is 113. So by using a threshold of 120 we can easily extract the location as evident by the below image. The co-ordinate of the detected point is (445, 249)

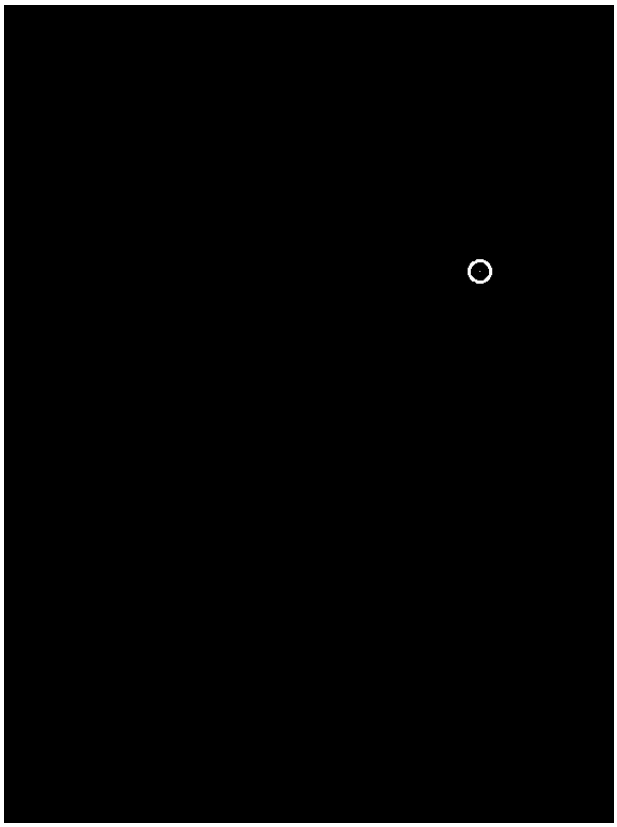

Figure 6: Detected Porosity

b **Choose an optimal threshold to segment the object from background by thresholding on 'segment.jpg'. Show the segmentation results, and draw a rectangle bounding box on the segmented results (bounding box is the minimum box that covers all of pixels of the object within it, an example (bounding box) is shown below for you to refer). Give the coordination's for four corner points of the bounding box you drew.**

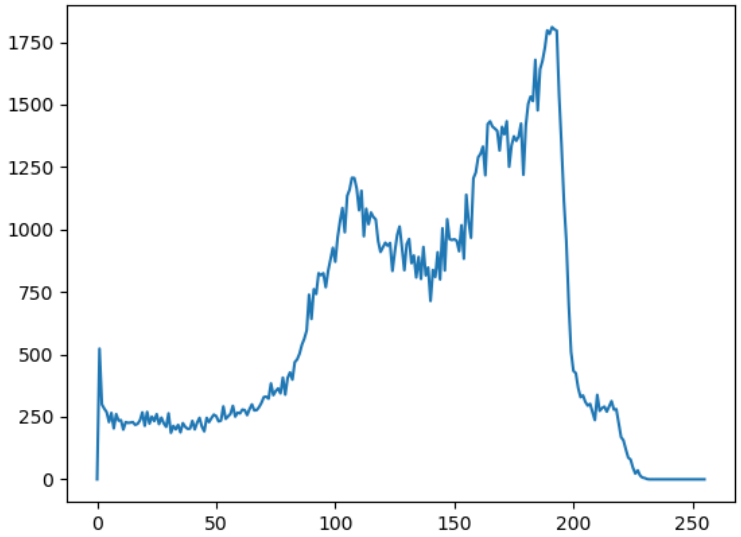

Figure 7: Histogram of pixel intensity of segment.jpg

To find the threshold effectively we need to create a histogram of the image which will plot the pixel intensity on x-axis and number of occurrence in the y-axis. It is important to ignore the occurrences of pixels with zero intensity since it is both inconsequential as well as it will change the scale of the histogram rendering the graph futile.

Since the bones have the brightest pixels in the image we can guess that threshold should be around 200.

However we will test this hypothesis with Heuristic approach. We will only apply on the pixels with intensity of more than 185. We can understand from the histogram that, the chicken fillet's intensity should be around 185, so we will be considering the chicken filet as the background and the bones as foreground, which are creating two Gaussian curves in the histogram.

(a) Lets set initial estimate of threshold T to an arbitrary number 220

(b) If we segment the image using T this will produce two groups of pixels: G1 consisting of all pixels with gray level values $>$ T and G2 consisting of pixels with gray level values $\leq$ T

(c) Then compute the average gray level values μ_1 and μ_2 for the pixels in regions G1 and G2

(d) Compute a new threshold value T = $0.5(\mu_1 + \mu_2)$

(e) Repeat steps (b) through (d) until the difference in T in successive iterations until they converge.

Executing the above steps will set the final threshold to 204.

Using 204 as threshold will extract bone segments from chicken filet.

Segments	Top Left	Bottom Left	Top Right	Bottom Right
1st bones (yellow box)	(162, 124)	(162, 164)	(202, 124)	(202, 164)
2nd bone (red box)	(253, 75)	(253, 207)	(304, 75)	(304, 207)
3rd bone (green box)	(334, 25)	(334, 287)	(366, 25)	(366, 287)
4th bone (blue box)	(386, 38)	(386, 254)	(421, 38)	(421, 254)

Table 1: Co-ordinates of the bounding boxes

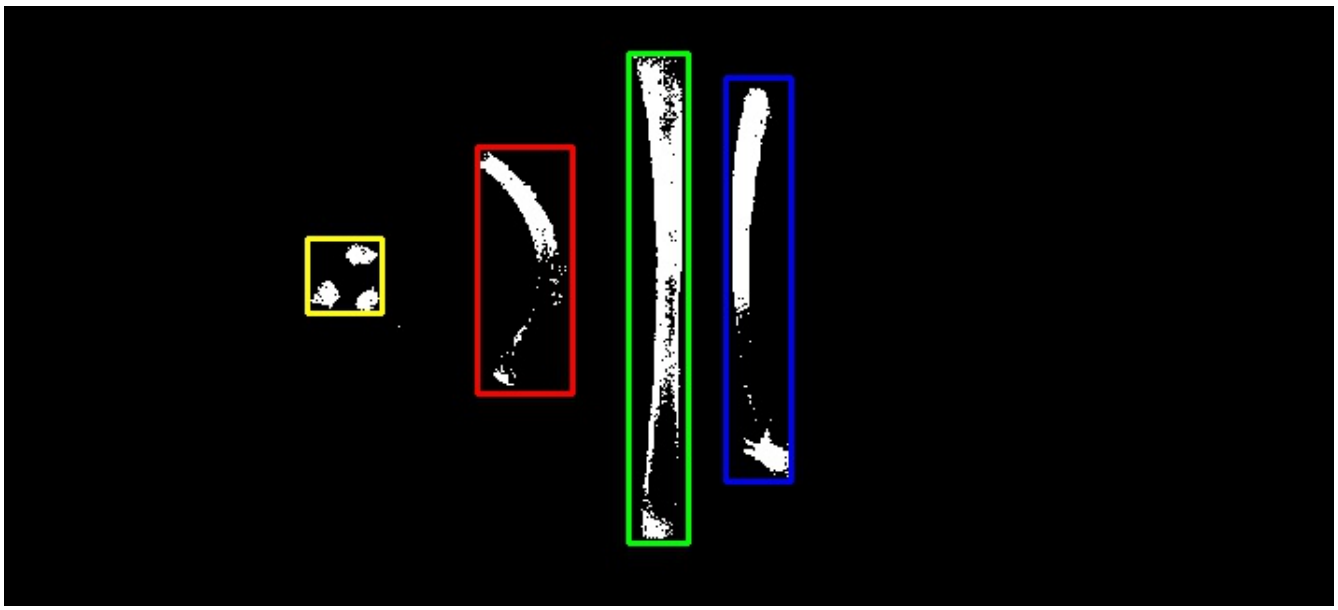

Figure 8: Segmented results

3 Hough transform

a **Utilizing the Hough transformation, design and implement an algorithm to detect all vertical (red lines), save the result image name as 'red_line.jpg' and indicate how many red lines you have detected.**

The Hough transform is a feature extraction technique used to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

Extracting straight line is a classical Hough transform problem, in which we will be using the Hesse normal form of straight line:
$r = x \cos\theta + y \sin\theta$

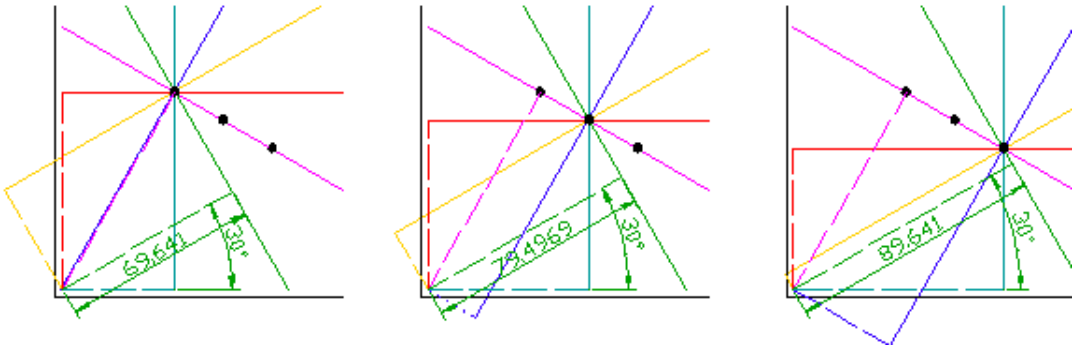

Figure 9: Straight lines in normal form

For each data point, a number of lines are plotted going through it, all at different angles.

For each solid line a line is plotted which is perpendicular to it and which intersects the origin.

The length (i.e. perpendicular distance to the origin) r and angle θ of each dashed line is calculated.

This is repeated for each data point.

A graph of the line lengths for each angle, known as a Hough space graph, is then created.

The point where the curves intersect gives a distance and angle. This distance and angle indicate the line which intersects the points being tested. In the graph shown the lines intersect at the pink point; this corresponds to the solid pink line in figure 9, which passes through all three points.

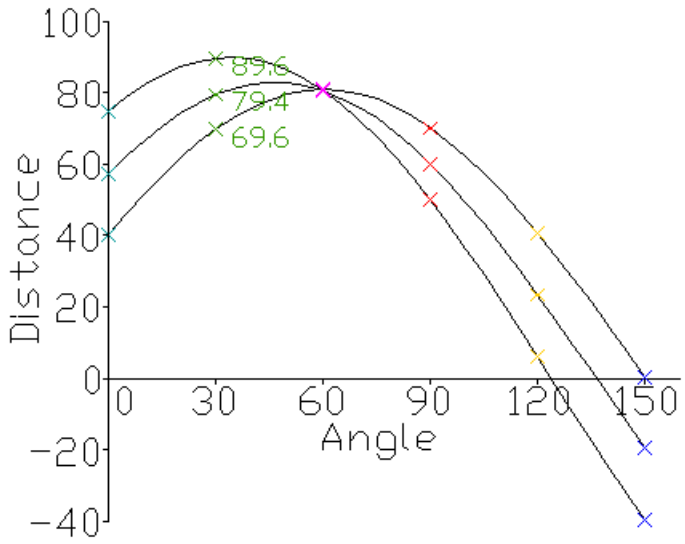

Figure 10: Points mapped in Hough Space

This way we can vote in Hough spaces, the points which get the most votes should create the most obvious straight lines.

In case of **hough.jpg**, first we will detect edges using sobel operator, then convert all the points to hough space. In hough space the value of r has been plotted against values of θ from -90 to 90 degree.

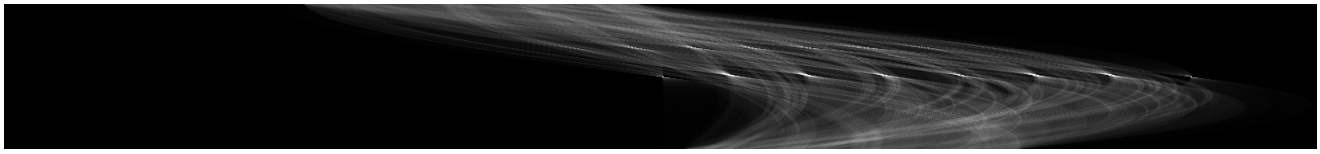

Figure 11: Accumulator for lines in hough.jpg

The brightest points got the most votes, so it is evident that, those are the straight lines represented in hough space. Extracting the peaks we convert it to x-y space again using the normal line representation formula and overlay them on the original image.

Figure 12: red_line.jpg detecting all 5 red lines

Since the peaks will include location for other lines also, we filter the red line from them. As we can see, the angle θ for red lines are -2 degree, we will use this as the filter criteria and discard other detected lines. This way we were able detect all 5 lines from the image.

b **Detect all the diagonal (blue lines), save the result image as 'blue_lines.jpg' and indicate how many blue lines you detected.**

From the same accumulator we will now extract the blue lines. We will be using the value of θ which is -36 degree as the filter criteria again.

Using this algorithm, we were able to extract 8 blue lines.

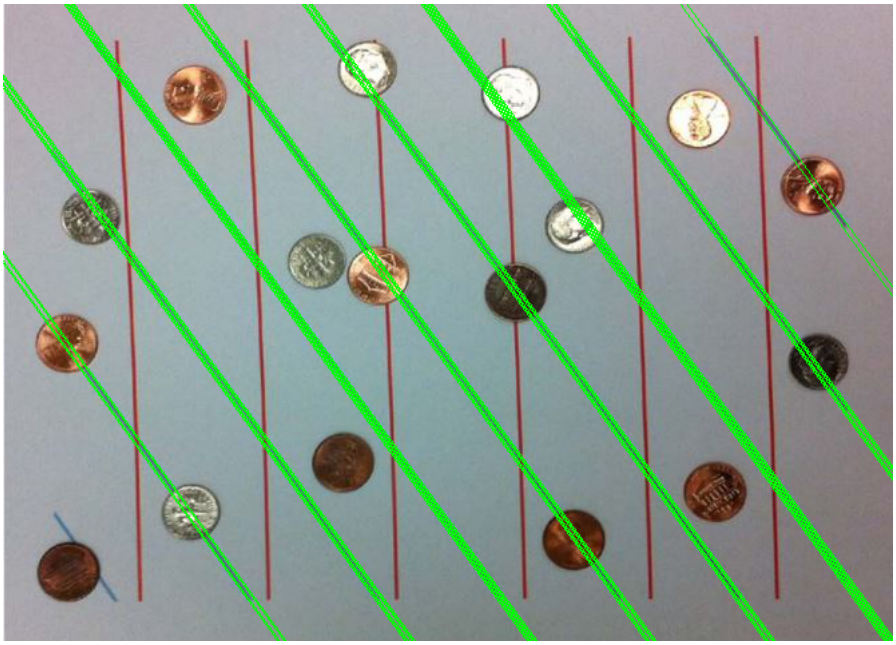

Figure 13: blue_lines.jpg detecting 8 blue lines

c **Detect all the coins in the image, save the result image as 'coin.jpg' and indicated how many coins you have detected.**

We will following the same method for circle detection. Though naturally we will be using the formula for circle.

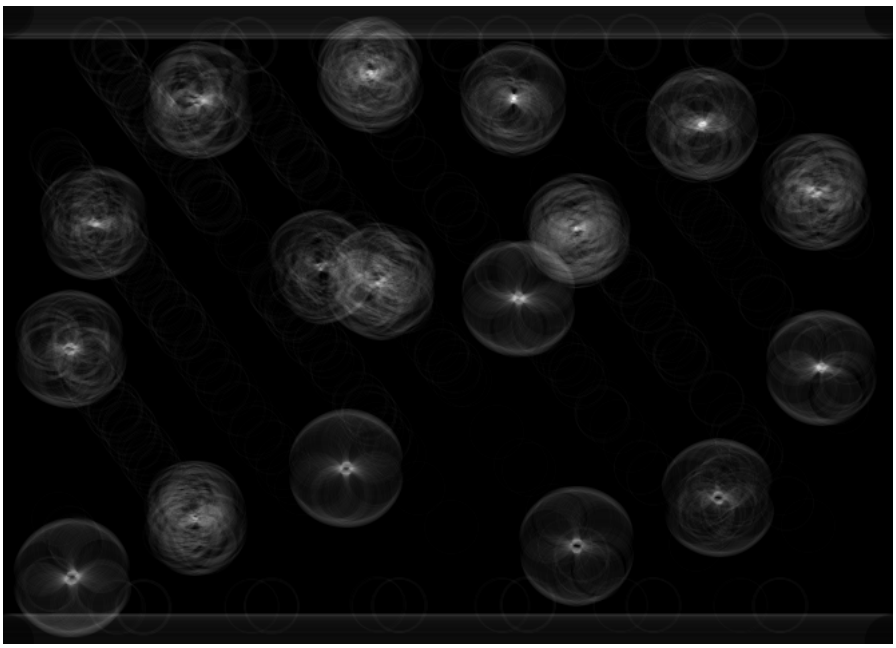

Figure 14: Accumulator for circles

Gaussian filter was used at the beginning to smoothen the image and then sobel filter in y direction was used. This way the vertical lines were removed and using threshold we were mostly able to eliminate slanting lines. By trial and error we were able to pinpoint the radius of circles which is 20. Then we transferred the remaining points in hough space and generated accumulator can be viewed above.

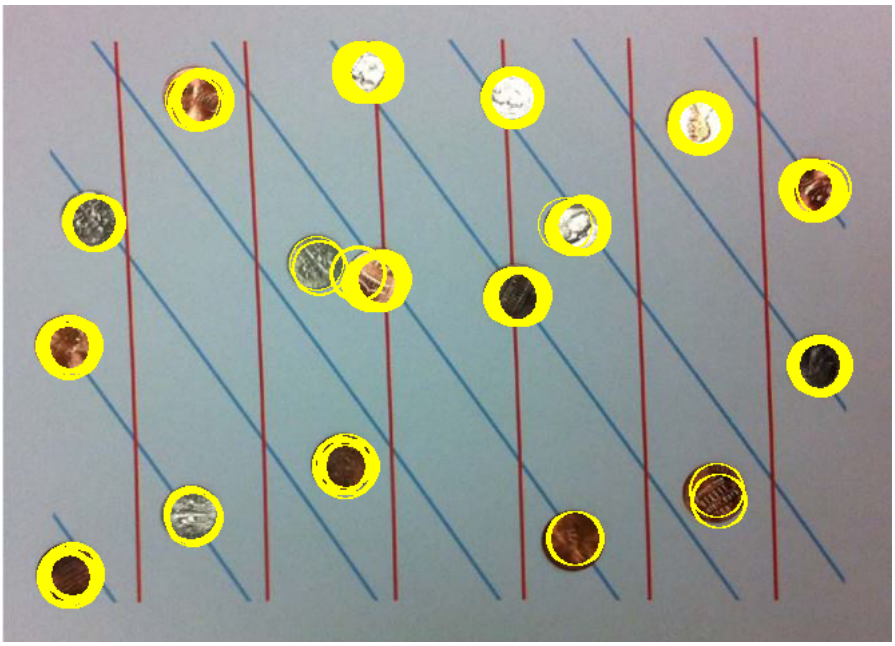

Figure 15: coin.jpg detecting all 17 coins

Then the highest peaks were converted back to x-y space and as we can see in figure 15, we were able to locate 17 coins.

References

[1] wikipedia. Hough transform, 2014.