PROJECT 3

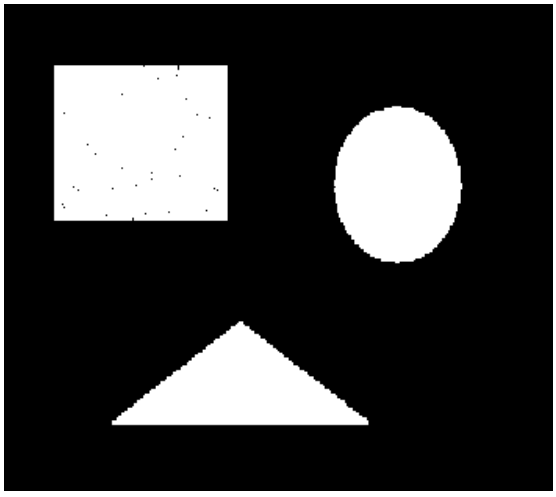
IMAGE PROCESSING

ADITYA VIKRAM PARAKALA| 50289171 | 12/03/2018

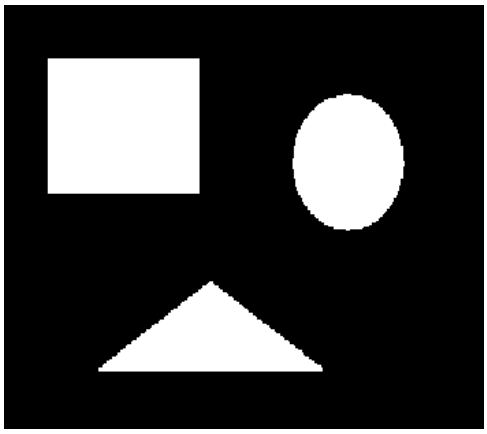
TASK 1: -

- **DENOISING:**

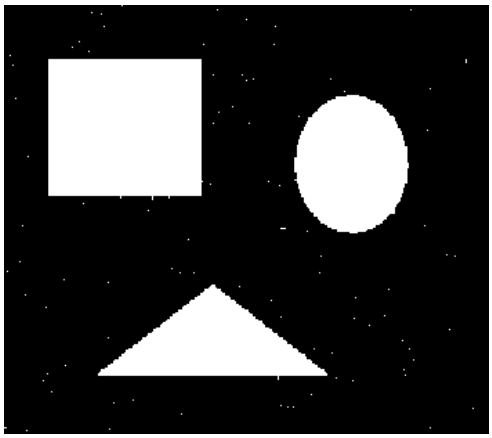
1. Implemented basic morphological operations i.e. erosion, dilation and the combination of both the operation which gives us two different algorithms for noise removal they are opening and closing.
2. The two key things for our algorithm are image and a structuring element.
3. The two algorithms are opening- erosion followed by dilation, closing- dilation followed by erosion, which are used in succession i.e. first we do opening followed by closing and then we do closing followed by opening.
4. By using erosion, the noise in the background is removed, size of the object in foreground is reduced and then followed by dilation which will restore the lost structure of the foreground object.
5. By applying dilation, the holes are covered, and the overall object size is increased, then by applying erosion the original size of the image is restored.
6. Noise pixels outside the object area are removed by opening and the noise pixels inside are removed by closing with the structuring element.



Opening on the input image (first step of algorithm 1)



(1.a(1)) res_noise1



Closing on the input image (first step of algorithm2)

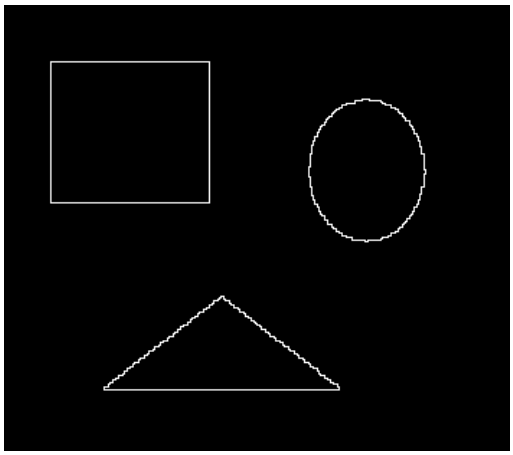
(1.a(2)) res_noise2

➢ ARE THE TWO IMAGES SAME?

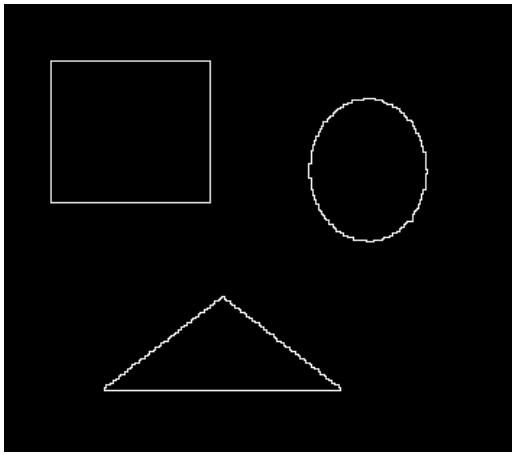
By the observing the above two images, we can clearly see that they are the same. The holes are filled in both the images due to dilation and the background noise is removed completely. Due to the compound operations that we performed on succession the original image size is preserved and the noise is completely removed.

➢ BOUNDARY EXTRACTION: -

1. The boundary of an image is the difference between the image and its erosion.
2. Boundary of first image is the difference between res_noise1 and its erosion.
3. Boundary of the second image is the difference between res_noise2 and its erosion.



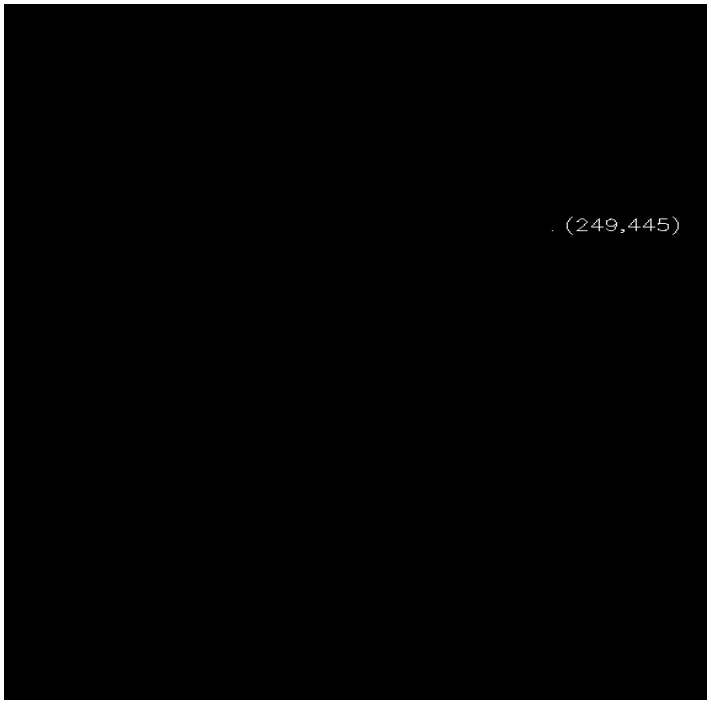
(1.c(1)) res_bound1



(1.c(2)) res_bound2

Task 2: -

- The given image has a porous point which we detect using the point detection algorithm.
- The image is convolved with a 5x5 kernel which is an eight-neighbor Laplacian mask with 24 as a middle pixel value and rest all are -1.
- By applying convolution, it removes all the unwanted pixel values which are not of much importance.
- We compute the weighted difference between center pixel and its neighbors.
- In this process the brightest point would be preserved, all other pixels would be made zero.
- We detect the porous point to know the defective point in the image.
- The thresholding technique used is global thresholding to perform thresholding on the convolved image.
- Compute the absolute values of the convolved image and we take optimal threshold value as 90% of the maximum absolute value of the convolved image.
- The resultant image after thresholding would contain the point.



(2a) Location of the detected point is (249,445)

Task2b: -

- The input image is read as a gray-scale image and all the pixel values are between 0 and 255.
- We plot a histogram to know the frequency of each pixel value in the entire image.
- The last maximum peak value is chosen as the optimal threshold for the given problem.
- The optimal threshold value is 208 according to the histogram peak value.
- The technique used for thresholding is global thresholding.
- After applying threshold, the resultant image consists of 4 objects that are the bones in chicken fillet image.
- The bounding boxes are drawn around the detected object in the image.

The below is the histogram generated from the image pixel values to manually threshold the image by considering the last highest peak value from the below histogram.

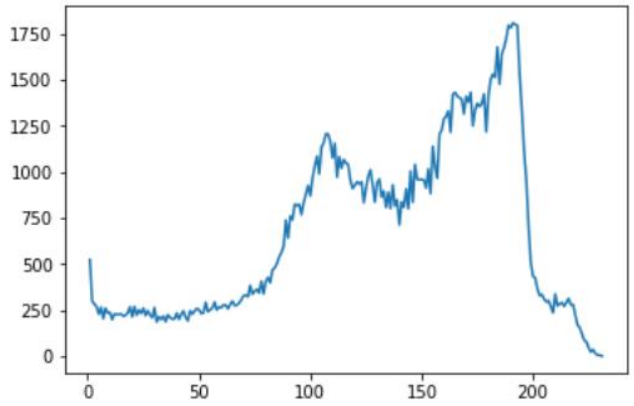
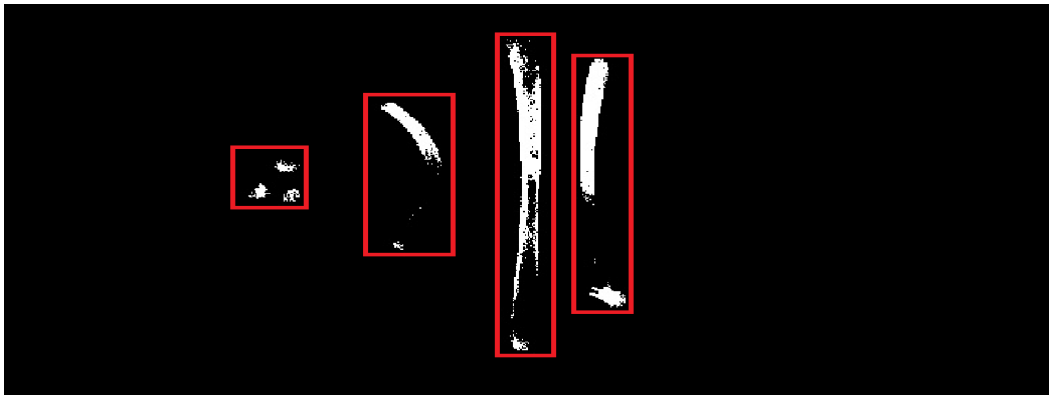


Image after applying optimal global threshold by manual tuning.

(2b) Resultant image with bounding boxes.



- The bounding boxes are drawn with help of MS-Paint and the coordinates of the bounding boxes are as follows:
- [(159,124), (199,164)]
- [(245,76), (299,203)]
- [(328,23), (364,286)]
- [(386,40), (422,250)]
- The above coordinates are the top-left and bottom-right corners of the bounding rectangles.

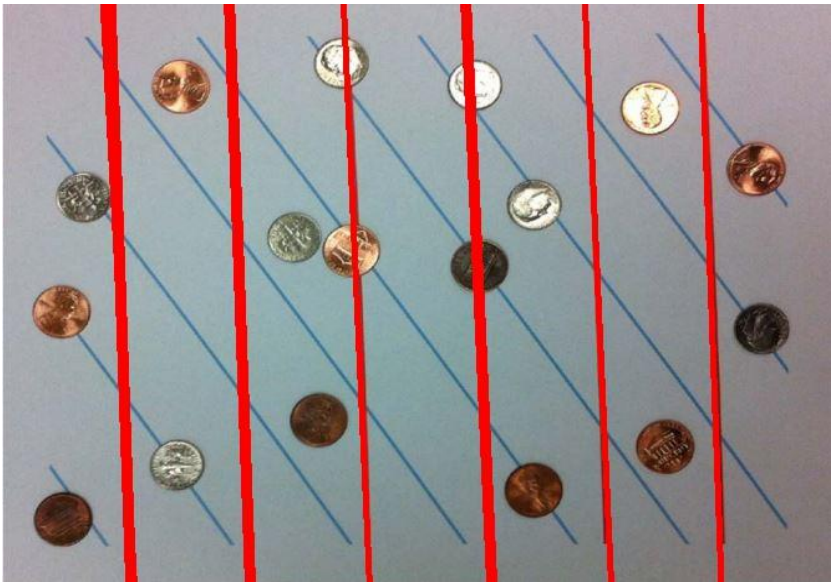
Task 3: -

- The given input image consists of vertical lines, diagonal which are inclined and coins.
- First, we apply the gaussian blur with 5x5 mask to remove unnecessary noise from the input image.
- Then apply sobel edge detection along x and y axis and calculate the magnitude which we use to detect the edges and the important features.
- Perform thresholding on the magnitude image to get the bright points which belong to the lines i.e. blue lines and red lines.
- Next, construct an accumulator matrix which is a rho x theta matrix. The range of rho is from 0 to length of image diagonal (maximum value)

- For each thresholded image pixel value we go iterate through all the theta values from smaller to larger magnitude and the rho values range from 0 to maximum of the length of the image diagonal.
- For all the non-zero values in the image we calculate the pho value which is given by the formula:

  rho = (x * cos(theta) + y * sin(theta))

- Now we have a complete accumulator with only the bright points which have the highest votes in the matrix that specifies which points to use for calculating the highest peak values.
- From the above step we get the indices of the bright points and construct another matrix which would have all the neighborhood pixels as 255 i.e. white, which indicates that there is a possibility that a line is detected.
- Accordingly, from after all the above steps are done, we get the points that are essential for drawing the detected lines using the Hough transform.
- The theta value we check are from -10 to 20 for vertical line detection and -40 to -20 for diagonal line detection.
- Resulting image would be all the 6 red lines detected successfully.
- Similarly, by the above process detected 8 blue lines i.e. the diagonal lines.



(3a) The detected output image of the red lines.

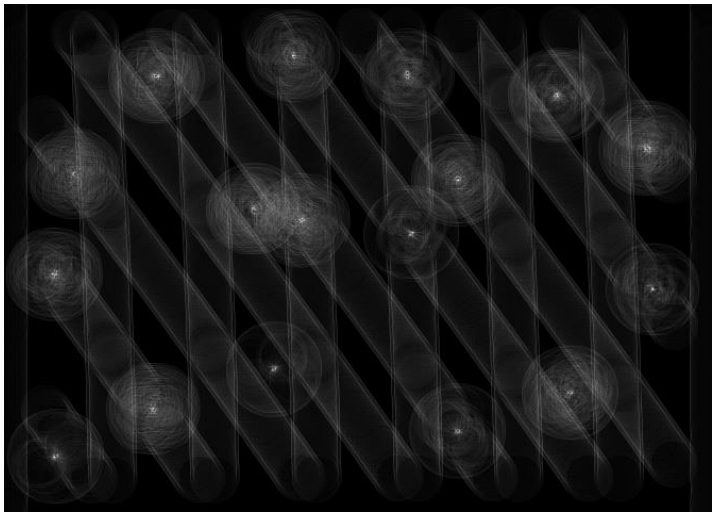
(3b) The detected image for the blue lines i.e. the inclined lines. All the lines except the last one is detected.

BONUS TASK:

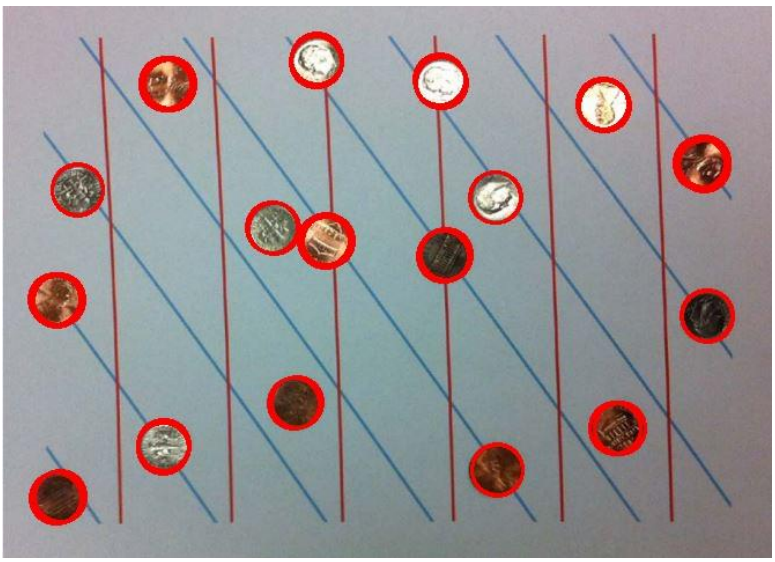
- The accumulator which is a matrix of pho * theta dimensions is constructed from by looping through the rho values from 0 to maximum and the theta values range from 0 to 360 degrees as we should detect a circle
- As the initial processing the original image is applied sobel filter across the x and y directions, the resultant image is thresholded to remove the unwanted noise.
- For all the possible theta values the values of center pixel are calculated as follows:

$$a = (x - radius*\cos(theta))$$
$$b = (y- radius*\sin(theta))$$

- A Hough space matrix is created which stores the votes of all the pixels, at the end this matrix indicated the bright points which are used for detecting the circle.
- The accumulator values of the image looks like the following images where only the bright circle centers are spotted, using which we detect the circle in the original image.



- Appropriate thresholding is applied on the voting matrix to extract the best intensity values, which are the centers of the detected circles in the image.
- The coordinates of the points which survive thresholding are stored in a list.
- Used cv2.drawCircle() method of the OpenCV library to draw the circle with radius value of 22 around all the detected center pixels which are accessed from the list in the above step.
- The final resultant image after implementing Hough transform for circle we get



(3c) Coin detected image.