CVIP PROJECT 3

Name : Kothapally srujan

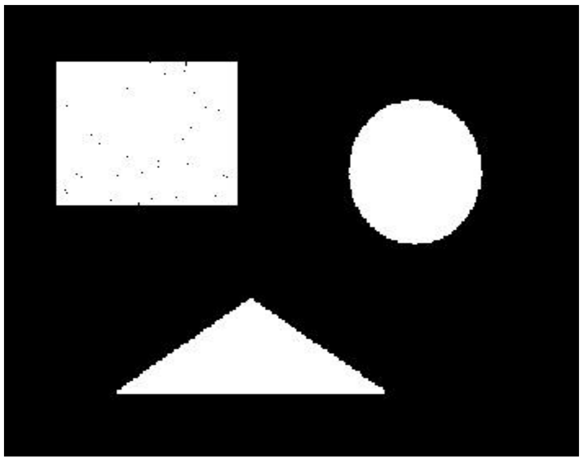
UB#:50291486

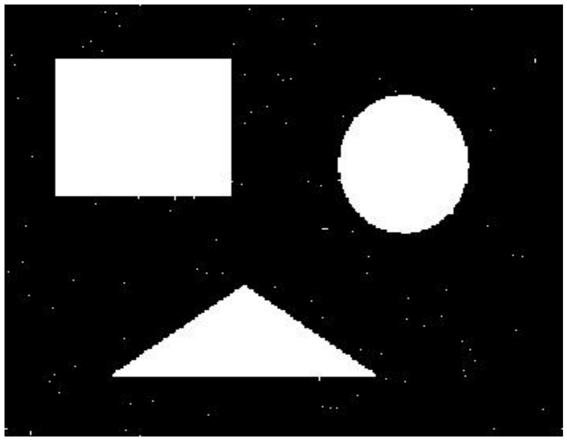
UBID: srujanko

1.(a)

For the image **Res_noise2.jpg** we use the morphological operation “opening” which is done by doing the dilation of the erosion of the input image.

For the image **Res_noise1.jpg** we use the morphological operation “closing” which is done by doing the erosion of the dilation of the input image.

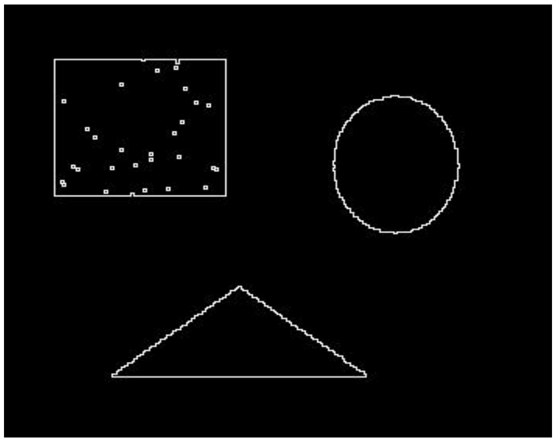

I.Res_noise2.jpg

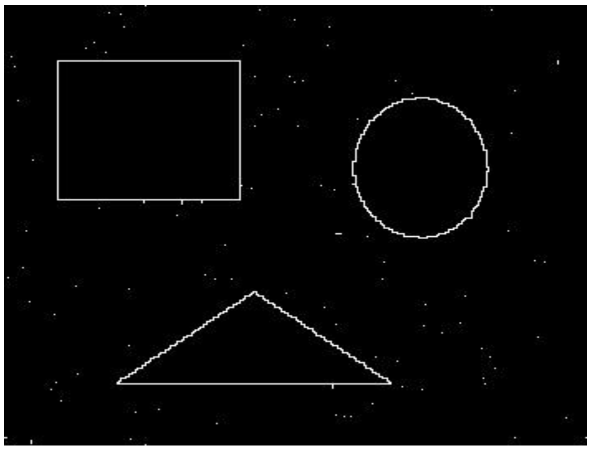

II.Res_noise1.jpg

For I. image which is opening the noise present the the background is removed.
For II. Image which is closing the noise present in the foreground(object) is removed.

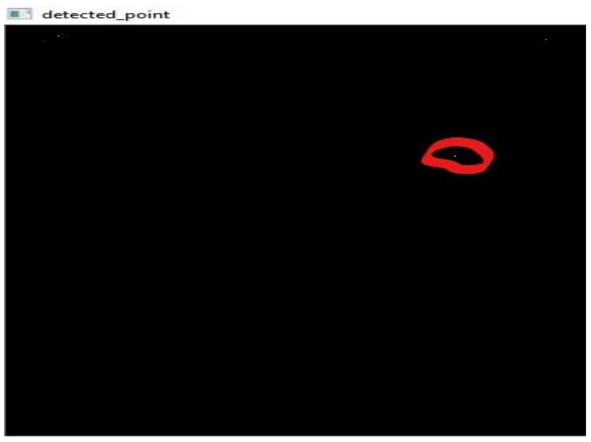
1.(b)
We can say that the both the images which are obtained by doing opening and closing respectively are **not same**.

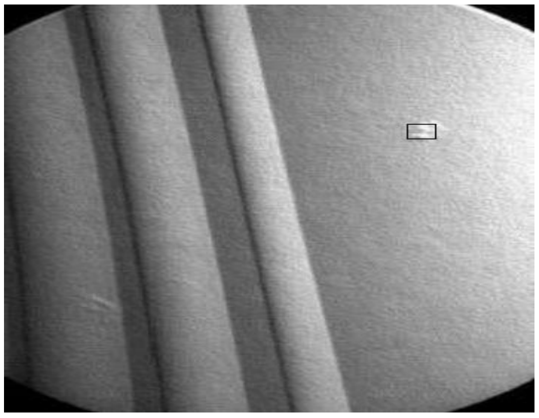
1.(c)
The boundaries for the images Res_noise2.jpg and the Res_noise1.jpg are obtained and are shown below with names Res_bound2.jpg and Res_bound1.jpg respectively.


I.Res_bound2.jpg


II.Res_bound1.jpg

2.(a)
For the point detection we blur the image and then we convolve the image using the laplacian filter.
The the threshold is give to the image so that the point of the porus is found.
The point that Is detected is circled.
The co-ordinates of the detected point in the original image is: **(276,151)**



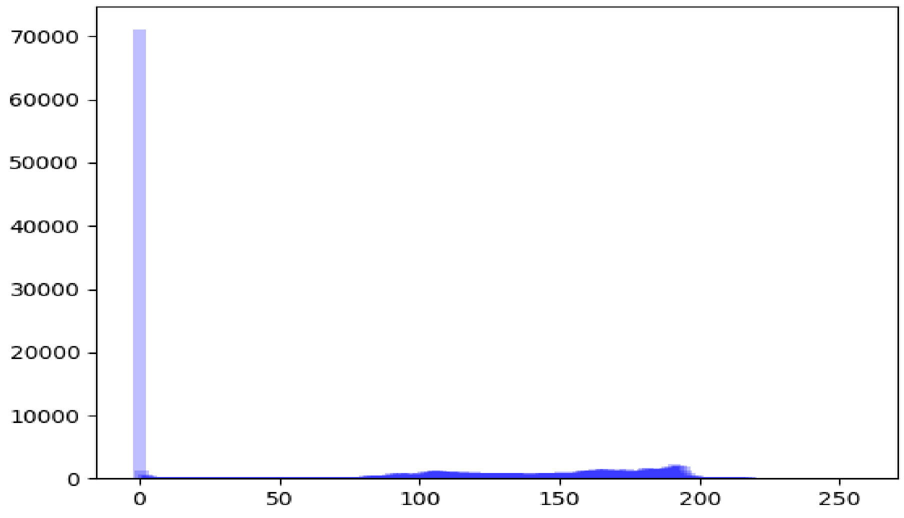


Detected point

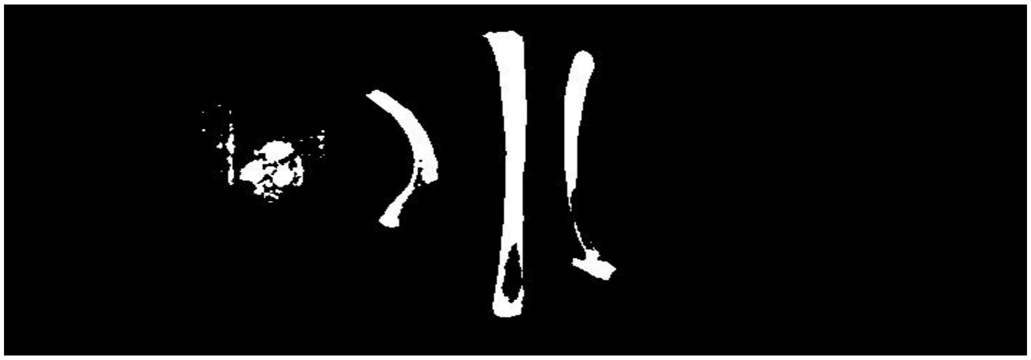
Point in the corresponding original image.

2.(b)

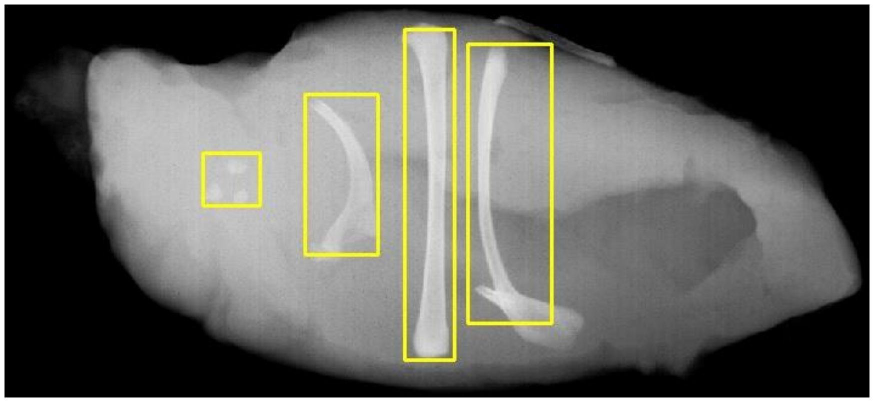
In the image segmentation we first find out the histogram for the image as shown below



For this histogram we set the threshold as 195 to segment the foreground image from the background. Then after thresholding the image the image is :



We use the connected components concept for the detection of the objects in the foreground and thus the output obtained is:



The coordinates represent the left-top corner and the right bottom corner of the bounding box.

The co-ordinates for 1st bounding box (left to right): (left top corner of rectangle, right bottom corner of rectangle):
((163,121),(210,164))

The co-ordinates for 2nd bounding box (left to right): (left top corner of rectangle, right bottom corner of rectangle):
((247,73),(307,204))

The co-ordinates for 3rd bounding box (left to right): (left top corner of rectangle, right bottom corner of rectangle):
((329,20),(370,290))

The co-ordinates for 4rd bounding box (left to right) : (left top corner of rectangle, right bottom corner of rectangle):   **((381,32),(450,260))**

TASK3: HOUGH TRANSFORM:

Corner or edge detection.

We give the edge-detected images as input to the Hough transform.
For the edge detection we use sobel edge detection.

For finding out the vertical line we give the vertical edge detection image using sobel to the hough function so that all the vertical(Red) lines are clearly visible.

For the Blue lines we use the combination of the horizontal edge detection and the vertical edge detection so that all lines blues lines are clearly visible in the edge detected images.

The noise in the edge detected images is removed using the eliminate2 method.

The resultant binary/grey image will have 0s indicating non-edges and 1s or above indicating edges. This is our input image.

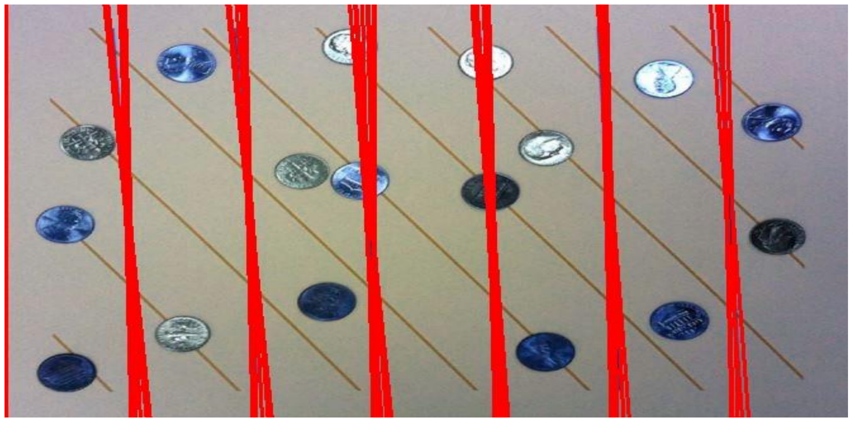
rho range and Theta range: ρ ranges from -max_dist to max_dist where max_dist is the diagonal length of the input image. θ ranges from −90 deg to 90 deg.

accumulator of θ vs ρ. It is a 2D array with the number of rows equal to the number of ρ values and the number of columns equal to the number of θ values.

Voting in the accumulator: For each edge point and for each θ value, the nearest ρ value is founded and increment that index in the accumulator. Each element gives how many points/pixels contributed "votes" for line candidates with parameters (ρ,θ).

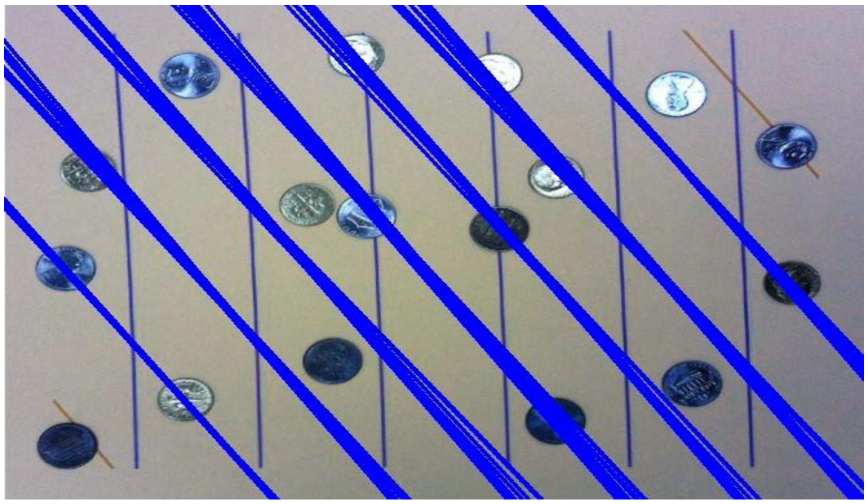
Peak finding: Local maxima in the accumulator indicates the parameters of the most prominent lines in the input image. Peaks can be found most easily by applying a threshold or a relative threshold (values equal to or greater than some fixed percentage of the global maximum value). For our image threshold value is 195.

3.(a):


Red_line.jpg

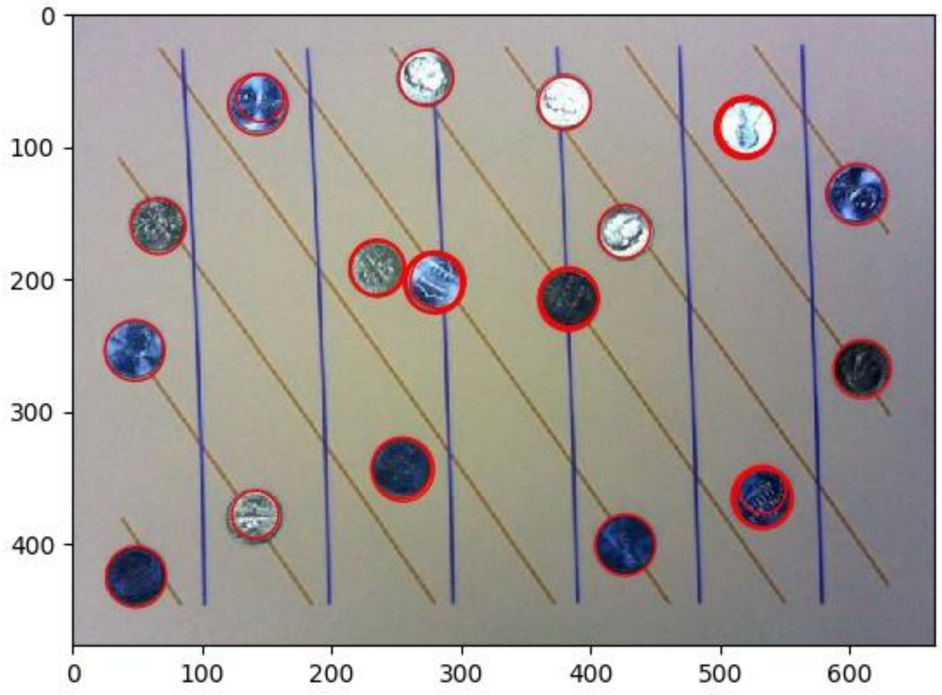
The number of Red lines detected are 6

3.(b):



The number of Blue_Lines detected are 7

3.(c) (BONUS): COINS DETECTION



EDGE DETECTION: For the circle detection we use the sobel filter to detect the edges from the image and we combine the horizontal edge detection and the vertical edge detection to make the coins clearly visible and we convert this image to the binary image and give this as the input to the Hough circles function.

R value: Now for the memory constraints and the time complexity we assume that we know the radius value R.
We assume the range of R values from 1 to the length of the diagonal. And we run the algorithm for each and every value in this range. And the maximum value of R is the diagonal because . No possible circle on the image can have a radius greater than or equal to the diagonal.

The equation of circle is given by:
$x = a + R\cos\theta$
$y = b + R\sin\theta$

So, every point in the xy space will be equivalent to a circle in the ab space R isn't a parameter, we already know it. This is because on rearranging the equations, we get:
$a = x_1 - R\cos\theta$
$b = y_1 - R\sin\theta$

for a particular point (x_1, y_1). And θ sweeps from 0 to 360 degrees.

The algorithm used :

```
For each pixel(x,y) in the image:
   For each radius min(r) = 1 to max(r) = length of diagonal   // the possible radius
     For each theta t = 0 to 360                                // the possible  theta 0 to 360
        a = x – r * cos(t * PI / 180);                         //polar coordinate for center
        b = y – r * sin(t * PI / 180);                         //polar coordinate for center
        A[a,b,r] +=1; //voting
     end
   end
 end
```

The list A is the accumulator and the it takes the voting for the circles detected.

The threshold for the radius of the detected circles is in range(17,25) so the accurate circles are detected.