Computer Vision and Image Processing
CSE-573
Project-3

Apoorva Biseria
Ubit name- abiseria
UB person Number – 50291145

1. **Morphological Image Processing**

a. Remove noise

Two methods were used while removing noise which are also two image morphological image processing algorithms.

The first method was $(A \bullet B) \circ B$

This was series of morphological operations which are dilation, erosion, erosion and dilation. These morphological operations were done using a 3*3 structuring element.

The second method was $(A \circ B) \bullet B$

This was series of morphological operations which are erosion, dilation, dilation and erosion. These morphological operations are performed using a 3*3 structuring element.

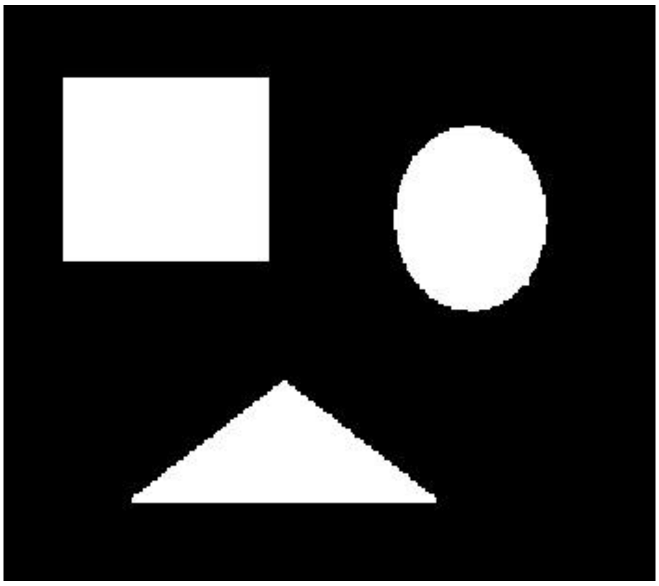


fig 1.1 – res_noise1.jpg

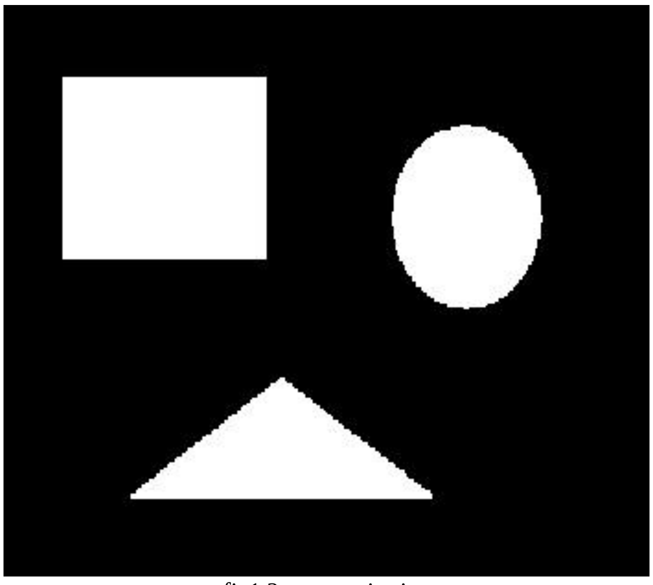


fig1.2 – res_noise.jpg

b. Both the images look exactly similar, both images go through a series of steps to reach the final result in order to remove noise, according to the images, there is no change in sizes of the image.

c. Boundary Extraction- It takes place when you take a image and subtract the eroded version of that image to get boundaries.

Boundary = I – Erosion(I)

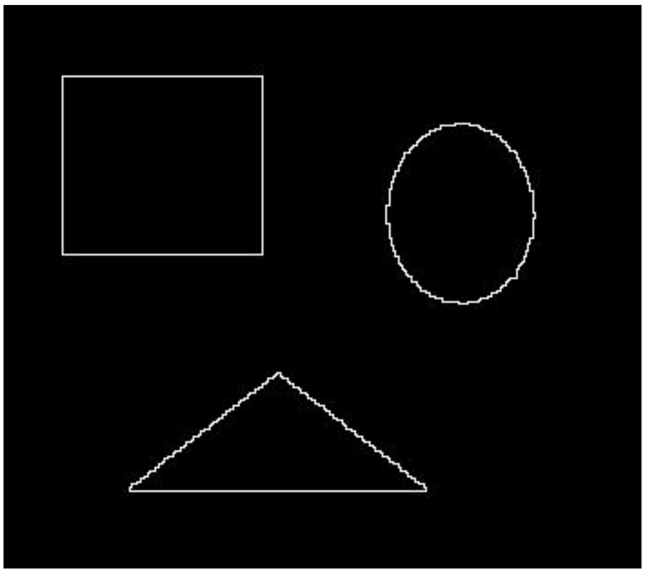


fig 1.3- res_bound1.jpg

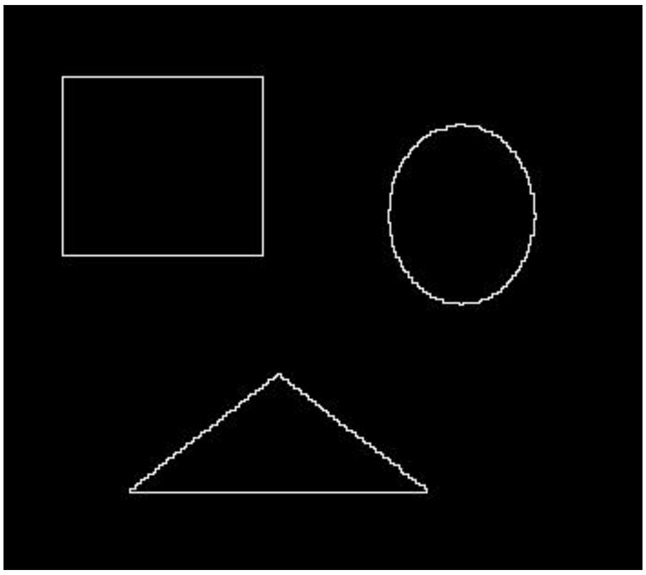


fig 1.4 – res_bound2.jpg

2. **Image Segmentation and point detection.**

a. In this task we were needed to detect porosity of the turbine blade, so I used a 3*3 kernel which was used to find the difference between the mid pixel and all the other 8 pixels and using the absolute value. This masked image was then thresholded to find

out the porosity of the turbine blade. The thresholding was found out manually by observing pixel intensies and the histogram.
The kernel used was kernel = [[-1,-1,-1],[-1,8,-1],[-1,-1,-1]]
Co-ordinates of the point are (445,-249)

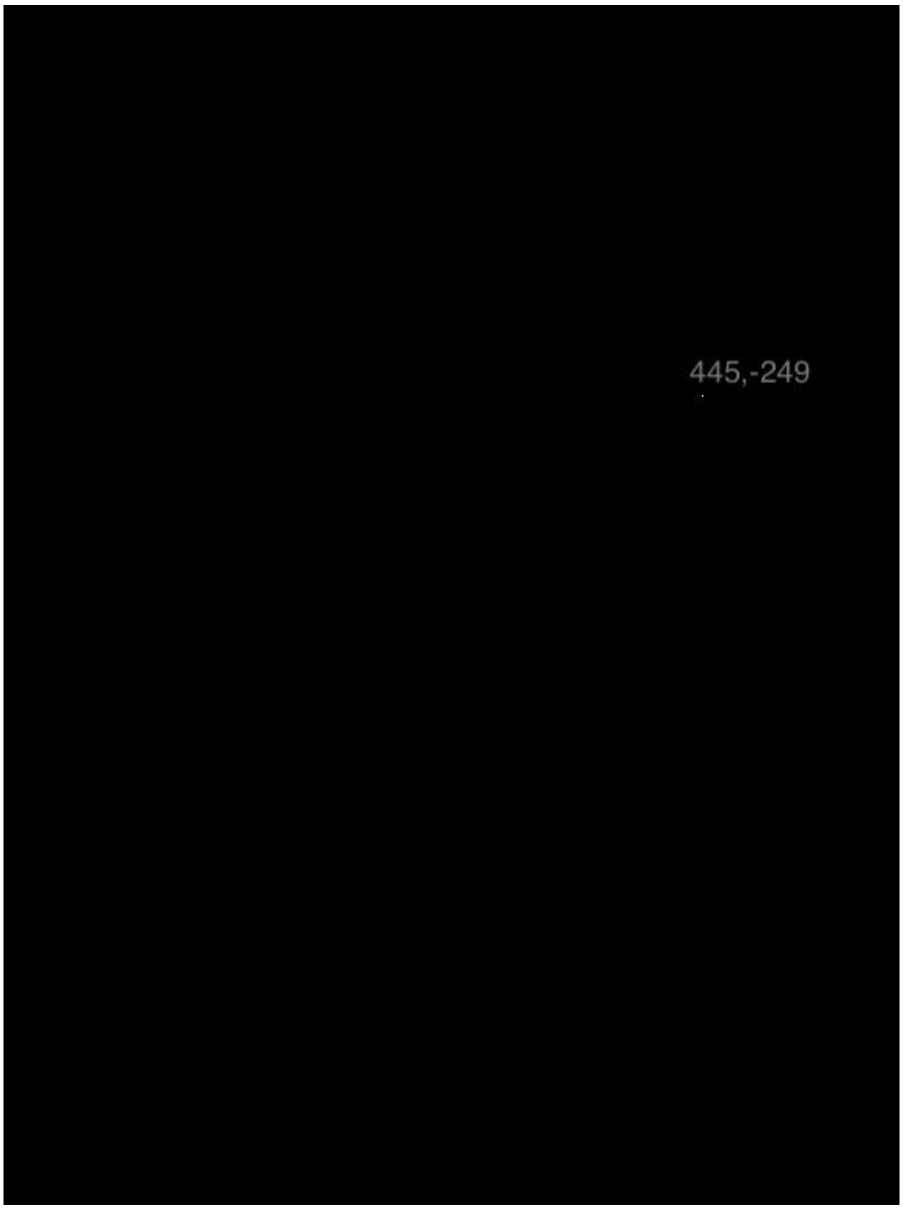
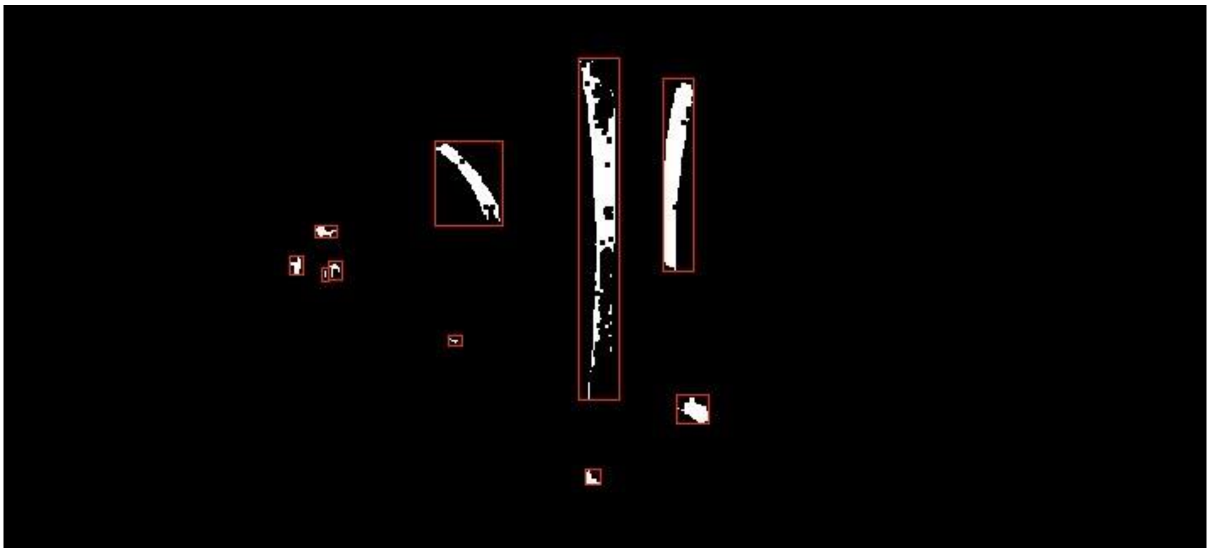


fig 2.1- porosity.jpg

b. We were needed to segment the image into components. Firstly I thresholded the image by observing the pixel intensities and the performed erosion using a 3*3 kernel to get different components.
Threshold was found at 205.



The marked points show the different components.

3. **Hough Transform**

   We were needed to detect the blue as well as red lines in the image.
   Firstly I performed the sobel operator on the image in order to differentiate the edges, then thresholding it and converting to a binary image. So the input for the next step would be a binary image. Next step was creating a [theta,rho] matrix which contained votes of the different edge points for each theta and rho. Then finding the local maxima for the theta for a given ranges of p. So By performing the operations at different angles, I concluded that angles were 2° and 37° and then drew the lines for these angles and maximum p's in order to detect the image.

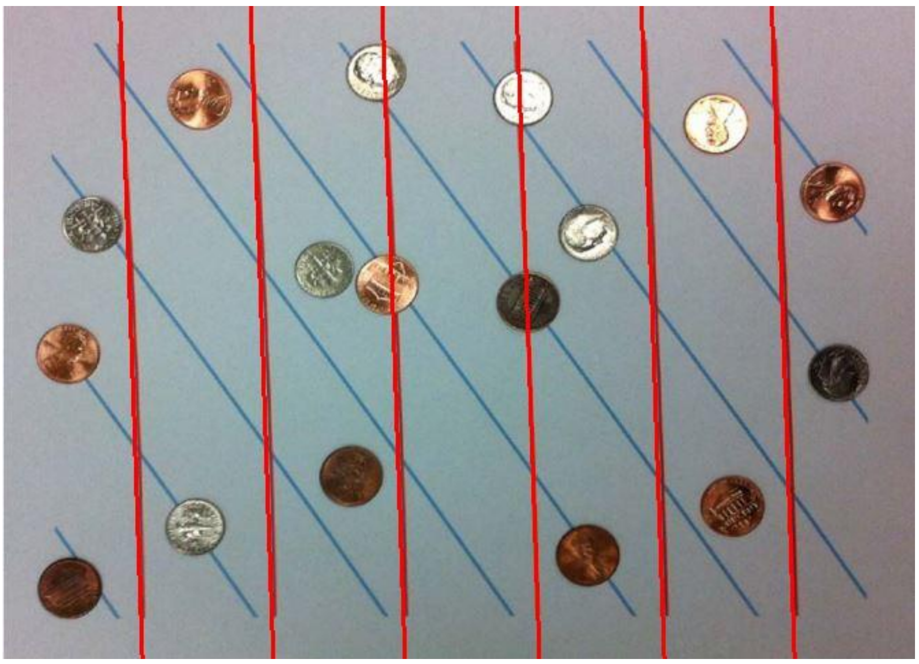


fig 3.1 – red_line.jpg

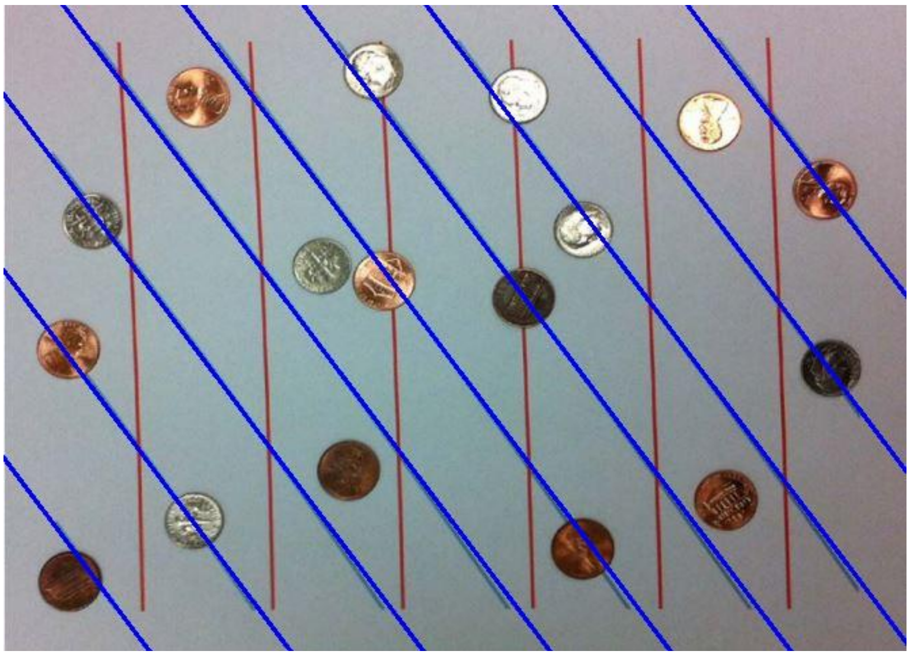


fig 3.2 – red_line.jpg

4. References

[1]"Hough Line Transform — OpenCV-Python Tutorials 1 documentation", *Opencv-python-tutroals.readthedocs.io*, 2018. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html. [Accessed: 03- Dec- 2018].
2]*Crcv.ucf.edu*, 2018. [Online]. Available: http://crcv.ucf.edu/courses/CAP5415/Fall2012/Lecture-18-HoughTransform. [Accessed: 03-Dec- 2018].