Computer Vision Assignment 1 (PS1)

By Dheeraj Ramchandani

September 15, 2019

1

instruction to run the code All the files for question 1to 6 are self named and for question 8 for circle detection use testing.py in which advance features are coded.

1.1

In this question To find the edges sobel operator is used and gradient in the x direction and y direction ;is calculated separately and then gradient magnitude is calculated. Input image is

1.2

The Lines are detected in the image using hough transform.
The theta angle is varied from 0 to 180 and in steps of 5 as the image is quite simple and lines are also horizontal and vertical. There are also few lines so the theta precision is kept low.

1.3

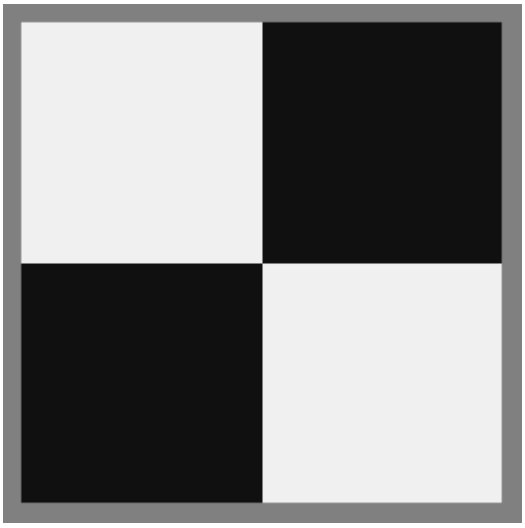
The adjustments done in this images is to firs smooth the image in order to remove the noise and adjust the parameters of the canny edge detector in order to remove the noise edges. The lower threshold is increased which leads to rejection of most of the noise pixels and only the stronger pixels related to the image are retained. The values for the canny edge detector set for this question is 140 and 210.

1.4

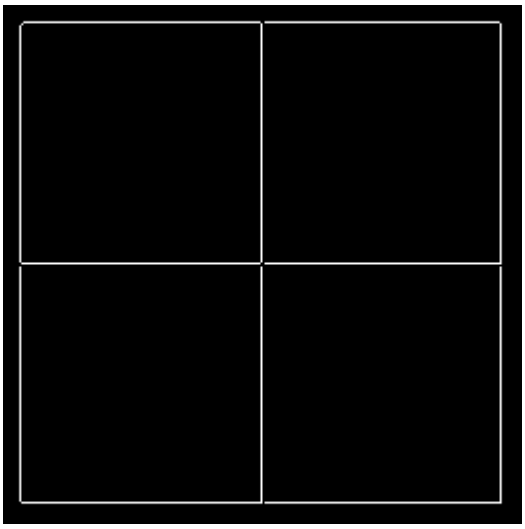
The code which worked for question 3 also worked for question 4 nothing much.

1.5

The main thing is to adjust the radius parameter in the function. Because when the radius is too small if will detect all the shapes and draw circles around it. This. happens when the radius is too small.



(a) Input image



(b) output image.

Figure 1: Edge finding using canny operator

1.6

- The Hough transform found lines in the background and also the pen boundary is not detected perfectly because the first pen boundary is not that strong so in order to detect it if we lower the threshold value of the canny edge operator then all other lines also get detected as well. The other parameter to consider is the hough accumulator threshold value which can be adjusted to detect the object of interest.

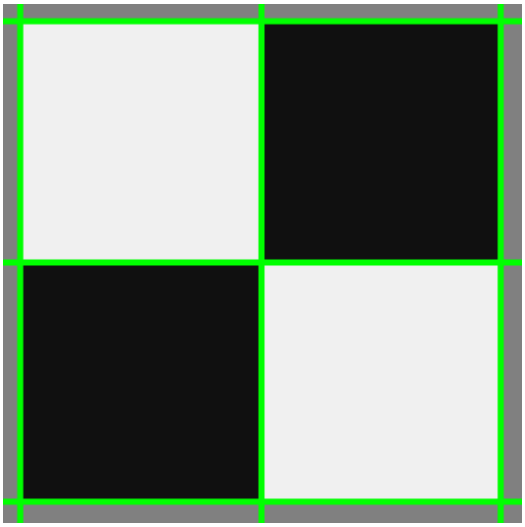


Figure 2: Line Detection using hough transform

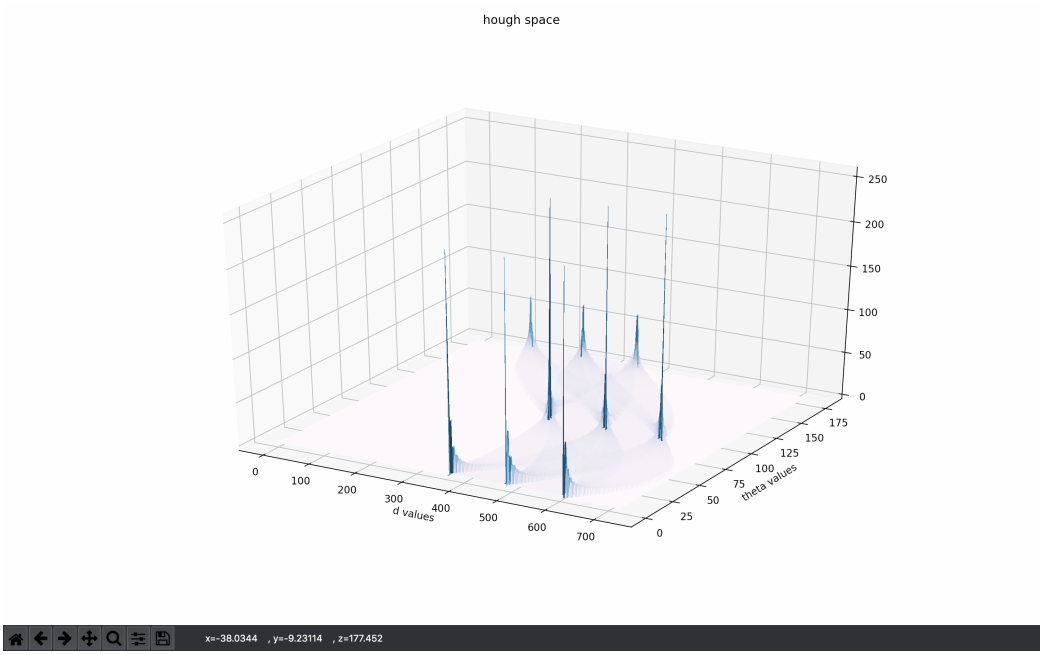


Figure 3: PS1-2 Hough Space Plot in 3D

- After making the required adjustments for the canny edge operator and the hough accumulator value the lines the pen boundaries are detect but two more lines are also coming as they are also strong. To particularly detect the pen boundaries also after detecting the line parameters (r,theta) x and y values are calculated and then they are

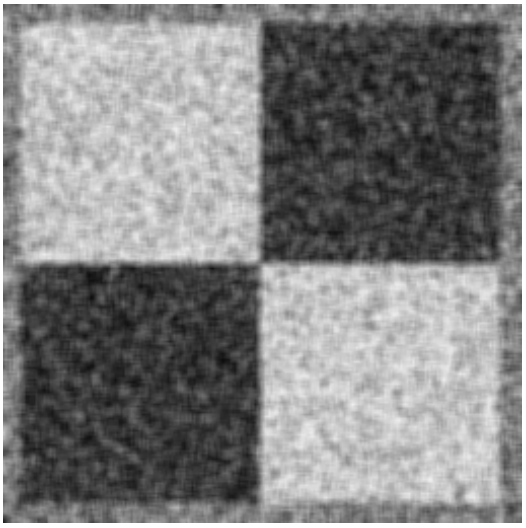
cross checked to see whether there is an edge or not.

1.7

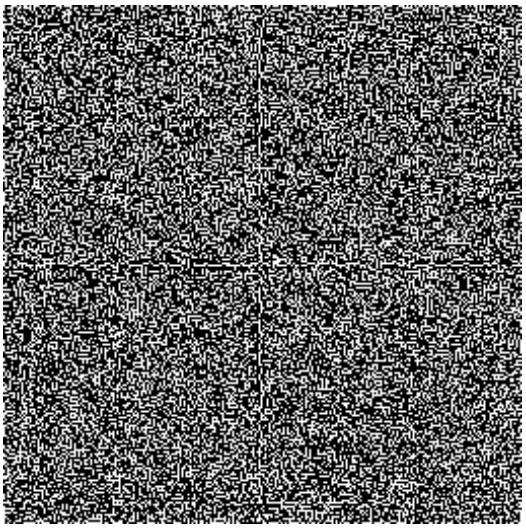
- The false alarms are same as detected in the problem 5 if the starting radius is too small then the method treats everything as a circle and draws around it. This can be removed by selecting the appropriate radius values and detecting the object boundary and post hough transform checking the values which we got from the hough transform with the image.
- To reduce the clutter meaning drawing of the same circle again and again the results are sorted with respect to the radius and the minimum distance between the centres of the circles is also considered as a factor while detecting the circles.
- The threshold value above which the centres is to be considered is also adjusted to detect appropriate circles.

1.8

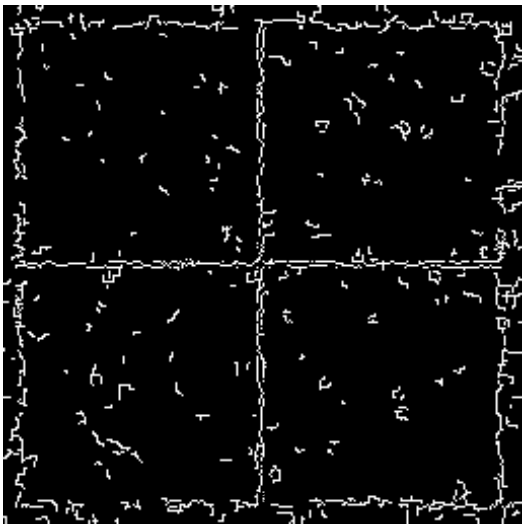
For circle detection the clutter of the circle is tried to be reduced by taking in consideration minimum distance between two circles and then in the result of the two circles are near by then they are removed.



(a) PS1-3 Smoothed image



(b) Edges in the original image



(c) Edges in the smoothed image

Figure 4: PS1-3 Edge detection in the Noisy image

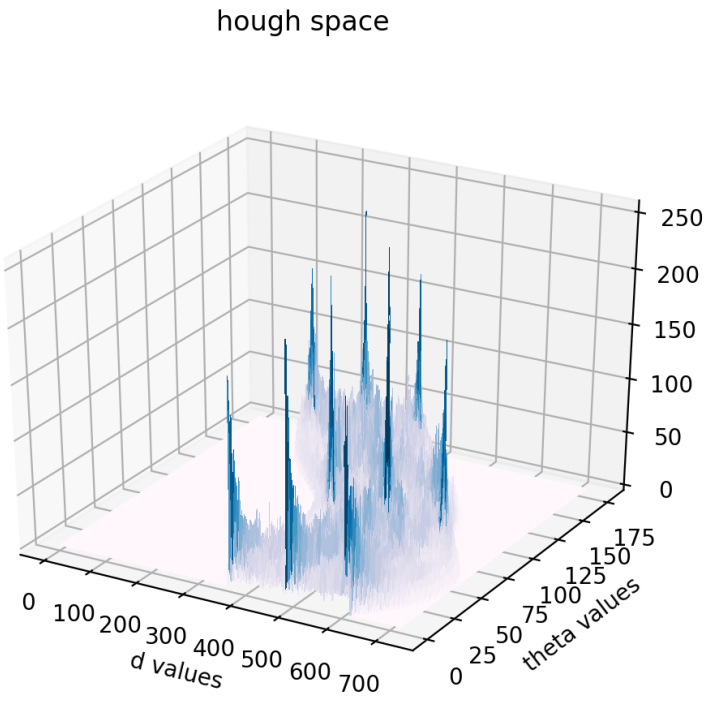


Figure 5: PS1-3 Hough Space Plot in 3D

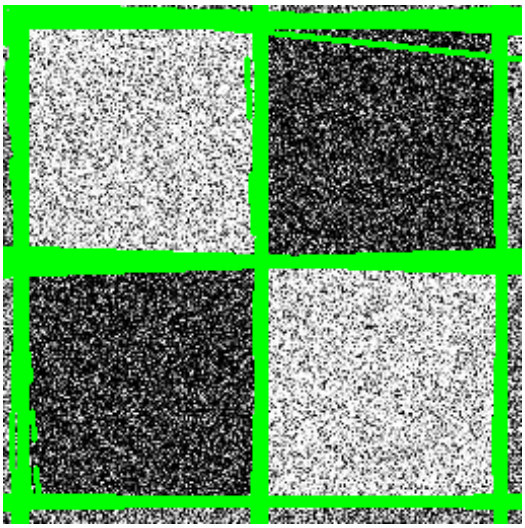
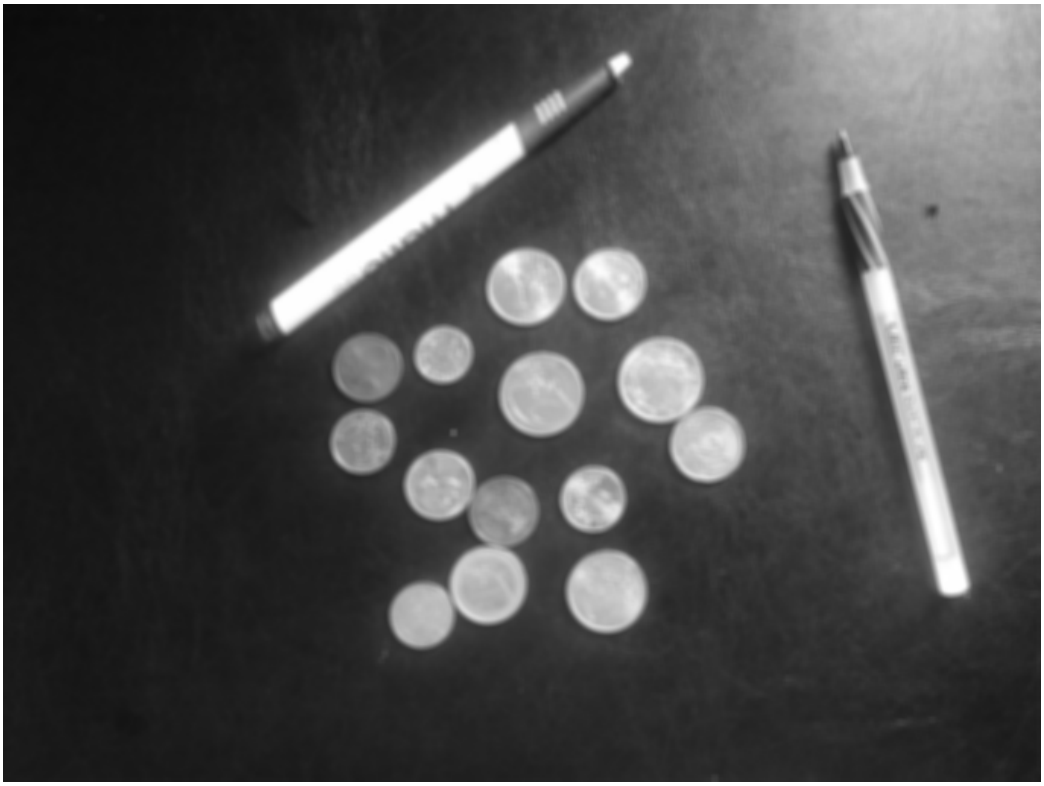
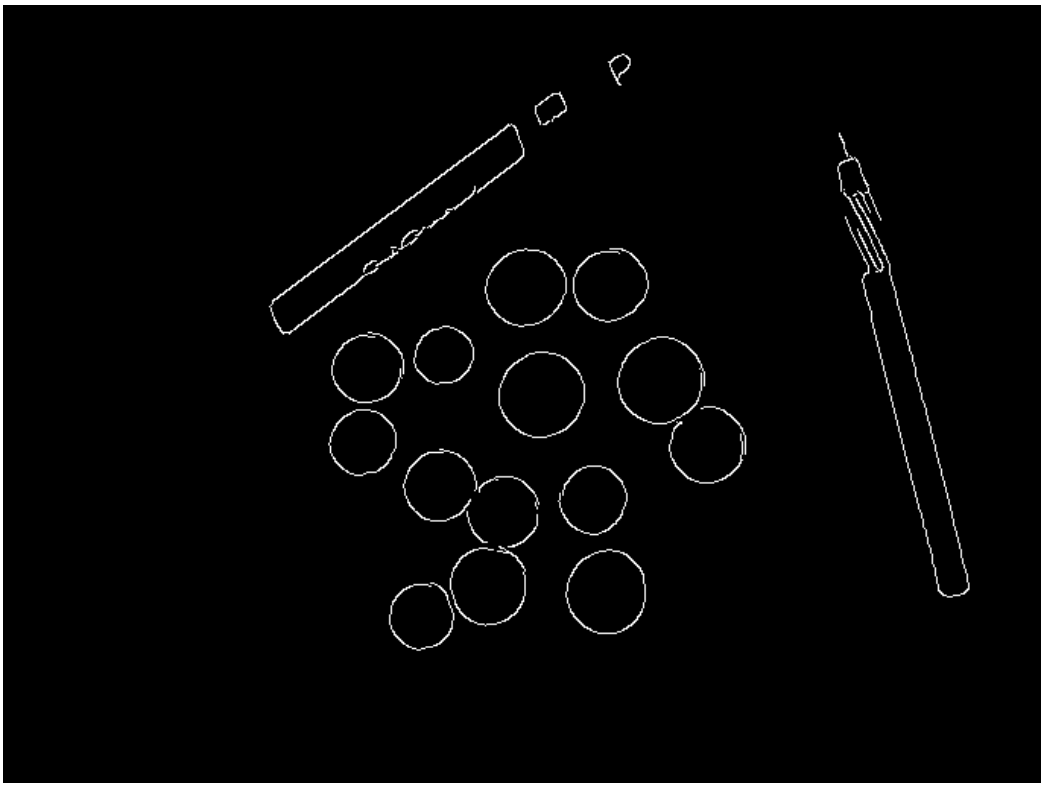


Figure 6: Image with lines drawn

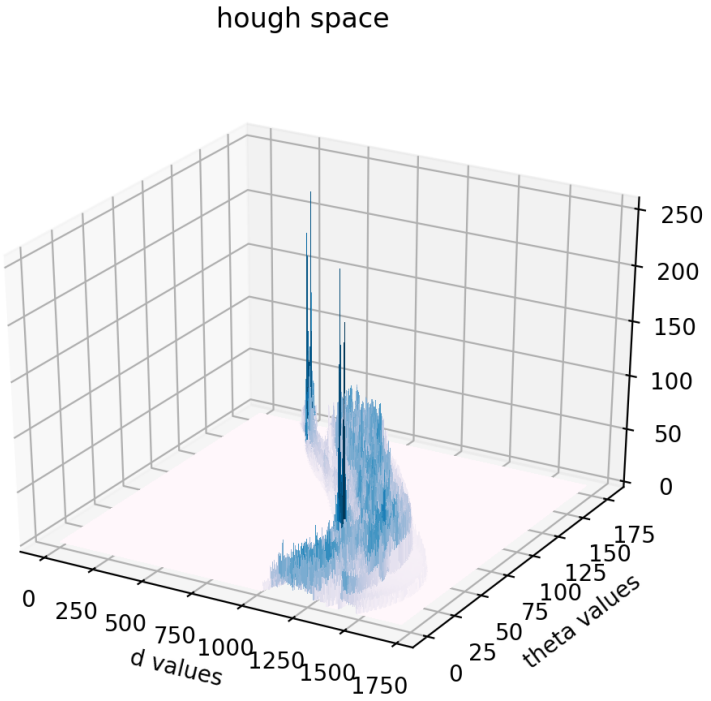


(a) Smoothed Monochrome image

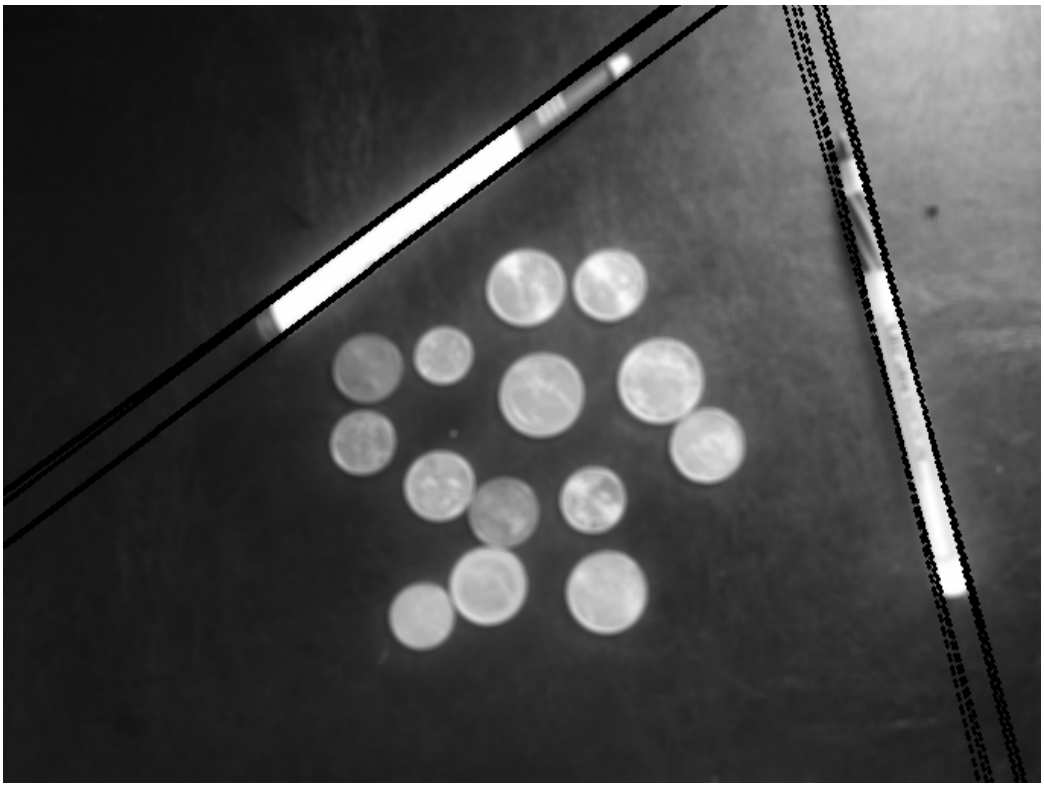


(b) Edges in the smoothed image

Figure 7: PS1-4 images

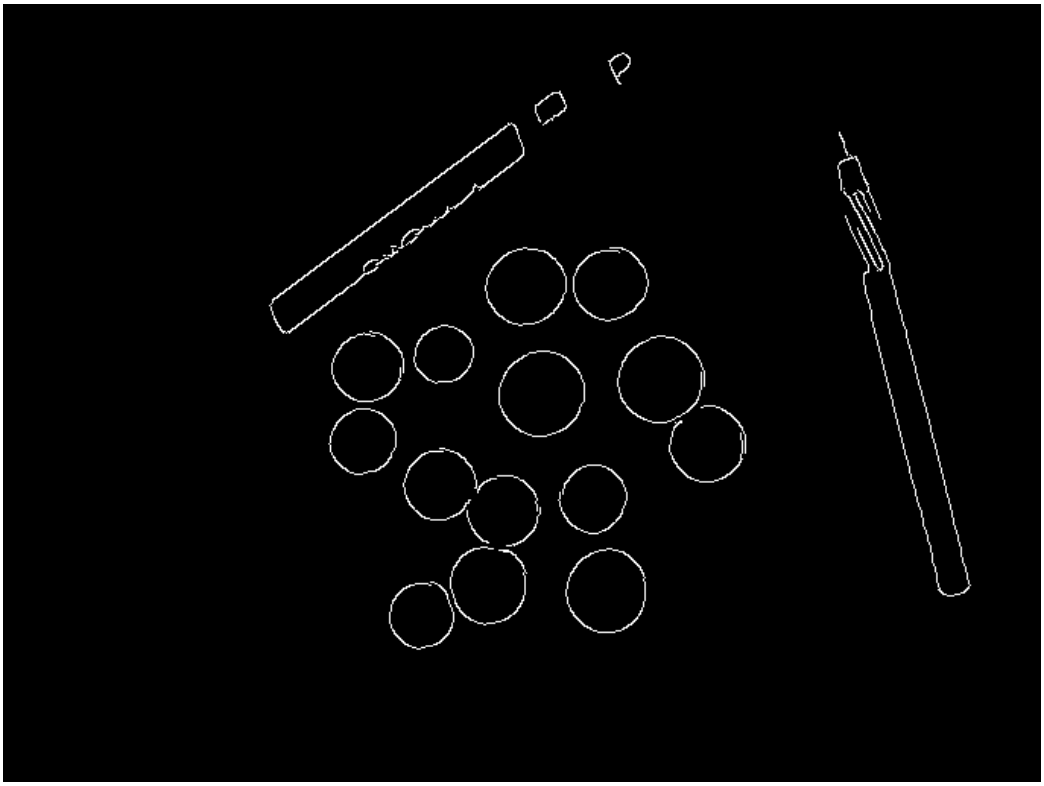


(a) Smoothed Monochrome image

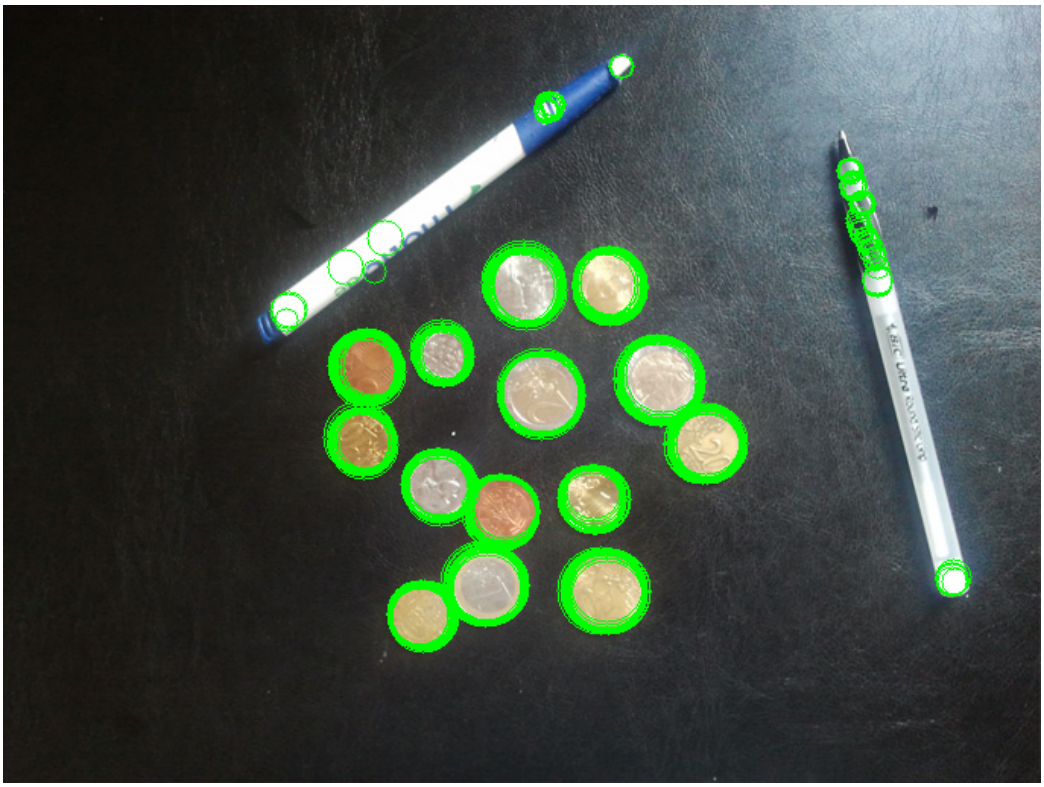


(b) Edges in the smoothed image

Figure 8: Hough space PS1-4



(a) Edge image



(b) circles detected in the image

Figure 9: PS 1-5

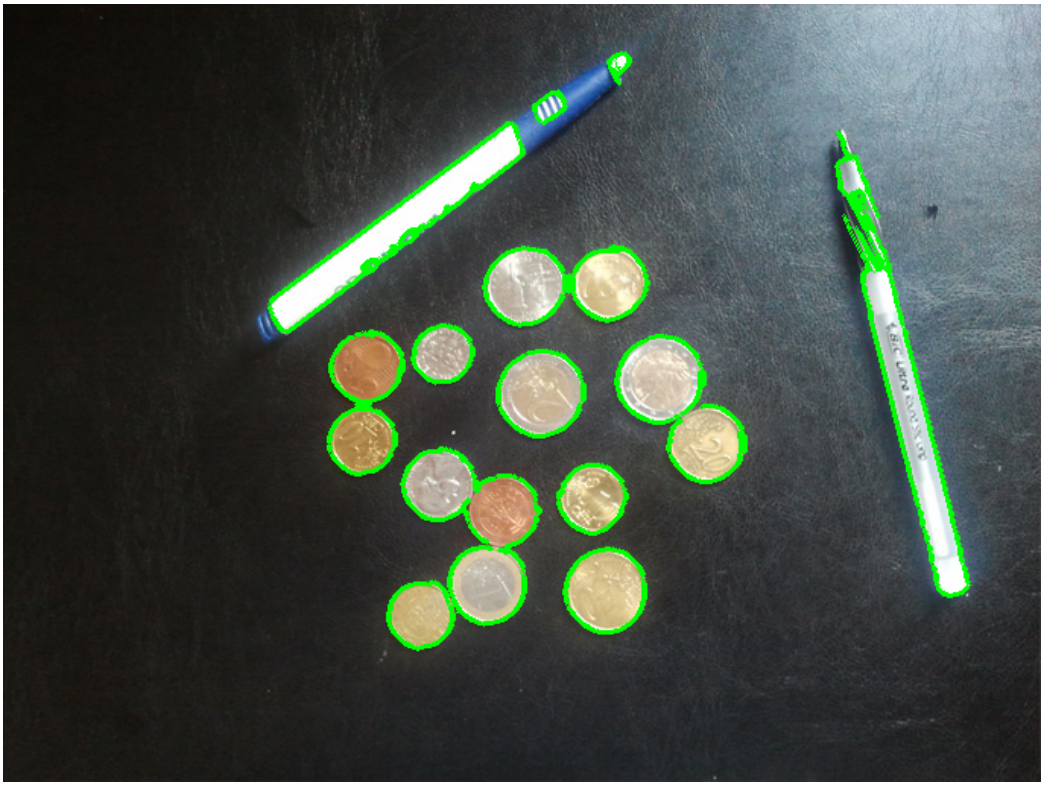
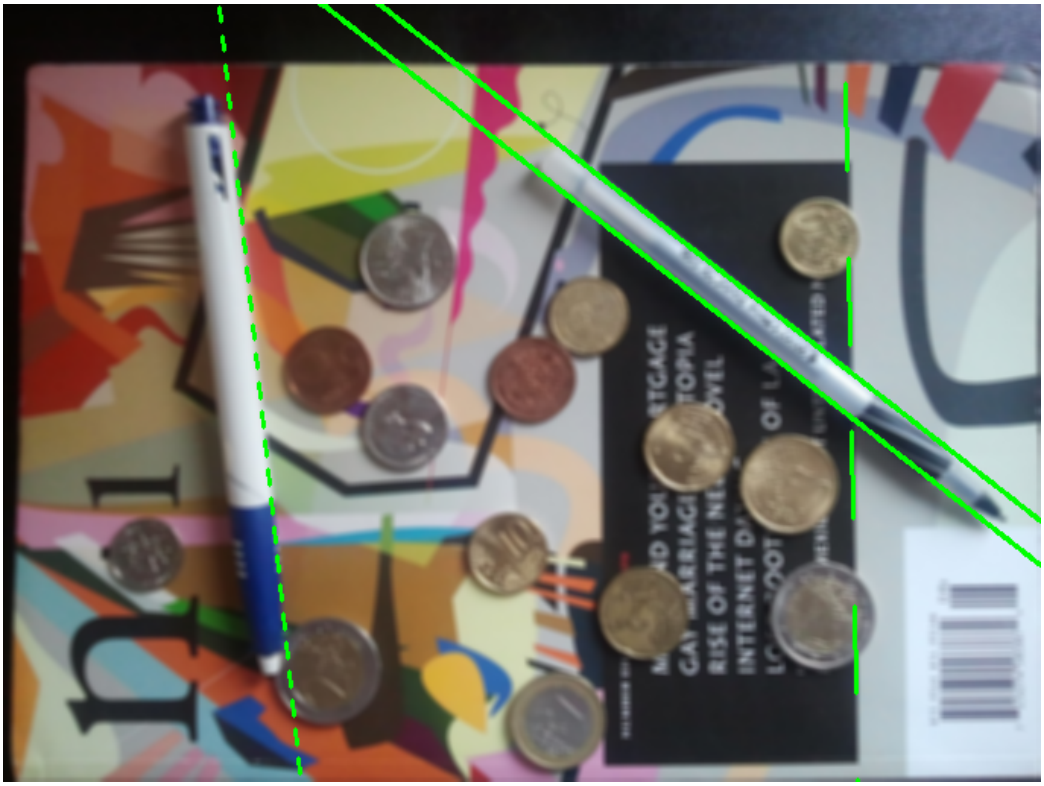
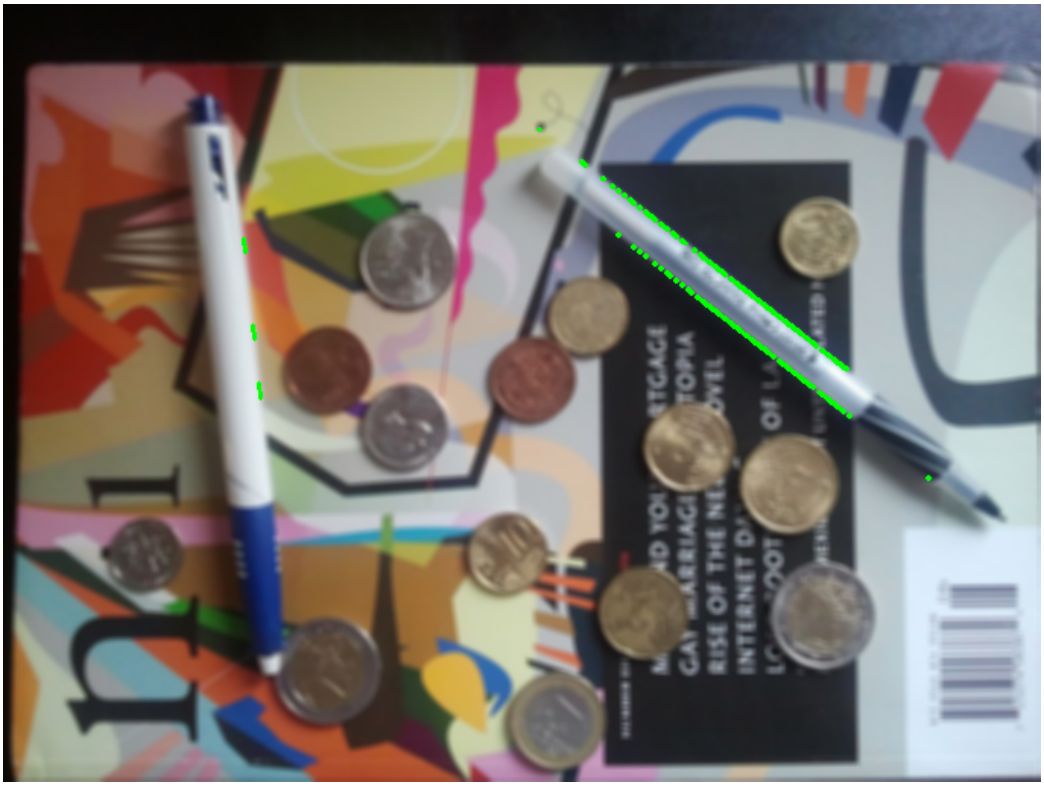


Figure 10: Faulty Detection PS1-5

Figure 11: Line detection PS1-6



(a) Refined lines



(b) Lines Refined More

Figure 12: PS 1-5

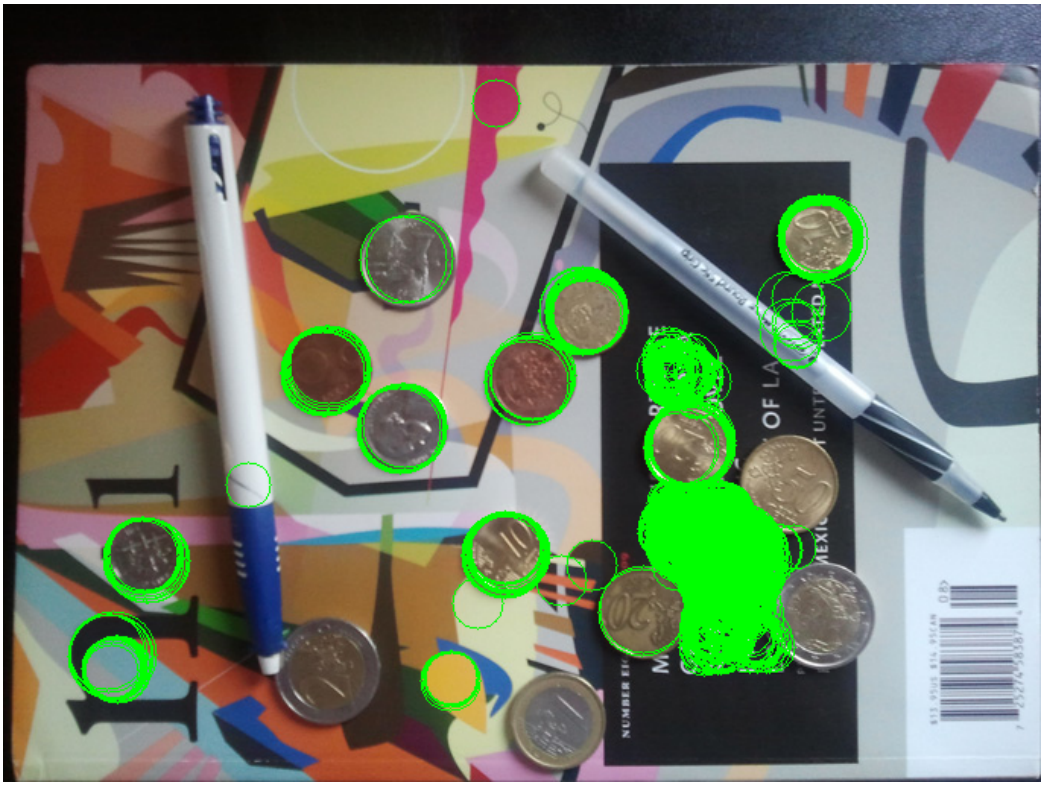


Figure 13: Circle detection PS1-7

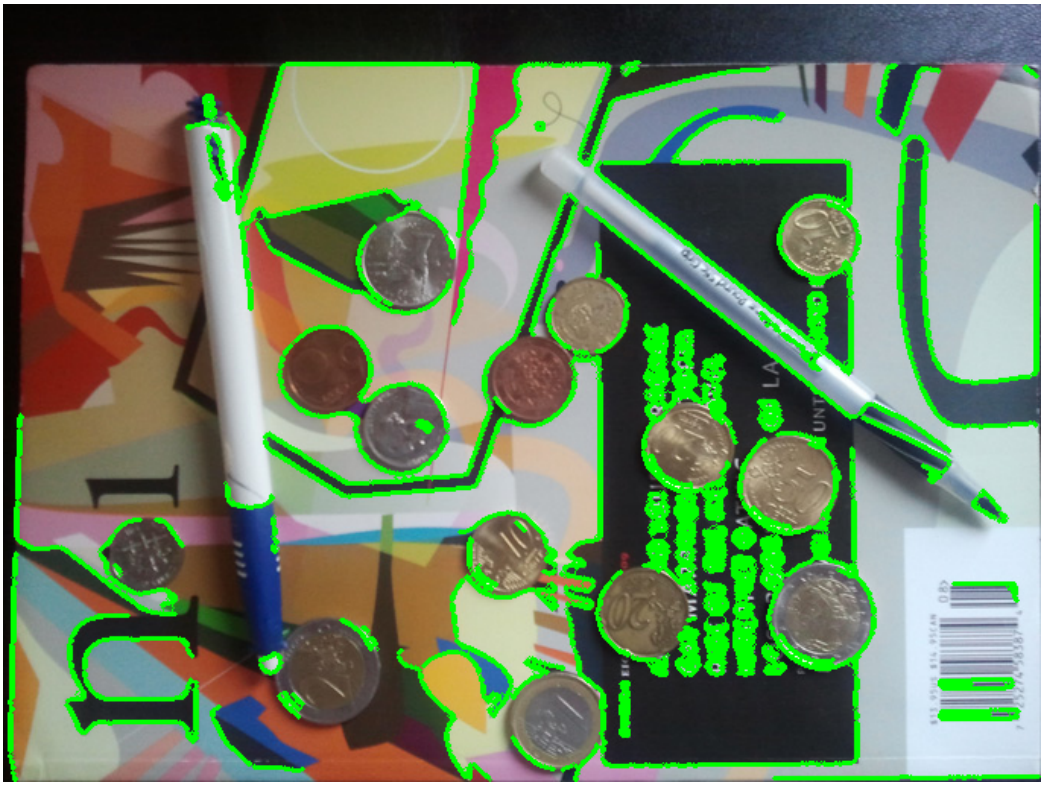


Figure 14: Faulty detection PS1-7

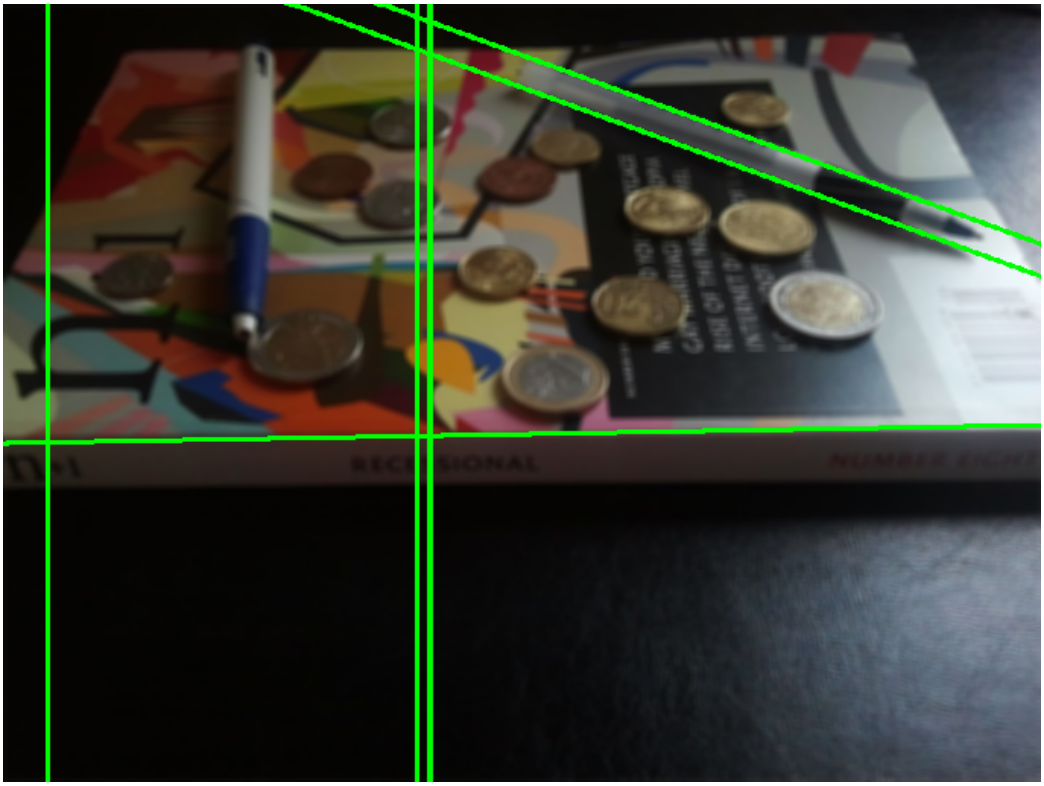


Figure 15: Line detection PS1-8

Figure 16: Line detection PS1-8