Computer Vision
CS 6476 , Spring 2018

PS 1

Supervisor :
Cedric Pradalier

Author :
Melisande Zonta

February 4, 2018

Contents

1 Creation of the edges image **2**

2 Hough Method applied on the lines **3**

3 Noise effect on the lines detection **4**
3.1 First Step : Smoothing 4
3.2 Second Step : Edges detection 4
3.3 Third Step : Hough Algorithm for lines 5

4 Lines detection on a more complicated image **6**
4.1 First Step : Smoothing 6
4.2 Second Step : Edges detection 6
4.3 Third Step : Hough Algorithm for lines 7

5 Circles Detection **8**

6 Finding lines on a more realistic image **9**
6.1 Application on a line finder 9
6.2 Problems with our line finder 9
6.3 Attempt to find the boundaries lines of the pen 9

7 Finding Circles on the same clutter image **10**

8 Sensitivity to distortion **12**
8.1 Application of the line and circles finder 12
8.2 How to fix the problem ? 12

1 Creation of the edges image

a b

Figure 1: *a.* ps1-1-a : Original Image. *b.* ps1-1-a : Edges Image.

The Figure 1 shows the result of our initial edge detection using the Canny edge detector.

2 Hough Method applied on the lines

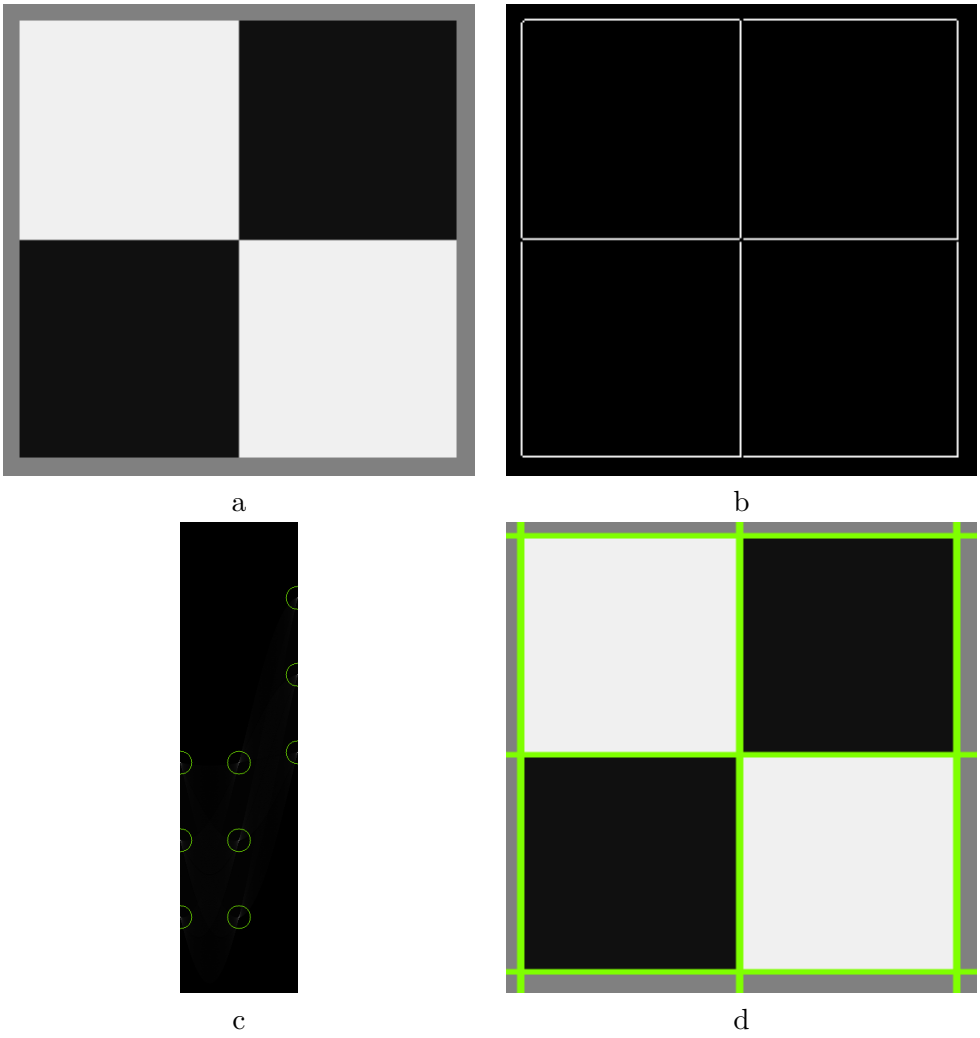


Figure 2: *a.* ps1-2-original : Original Image. *b.* ps1-2-edges : Edges Image.
c. ps1-2-accumulator : Hough accumulator array image. *d.* ps1-2-lines : Intensity Image with lines drawn on them.

The Figure 2 shows the result of the voting in the accumulator and the associated image with the detected lines is shows in Figure 3. Since the computation for this image is rather quick, we choose to keep a bin size of 1 degree on the theta parameter, to get a fine detection.

The image being a 256px square, the maximum value of d is the diagonal, around 136. Thus we choose a bin size of 180 on the d parameter, resulting in a rectangular accumulator.

3 Noise effect on the lines detection

3.1 First Step : Smoothing

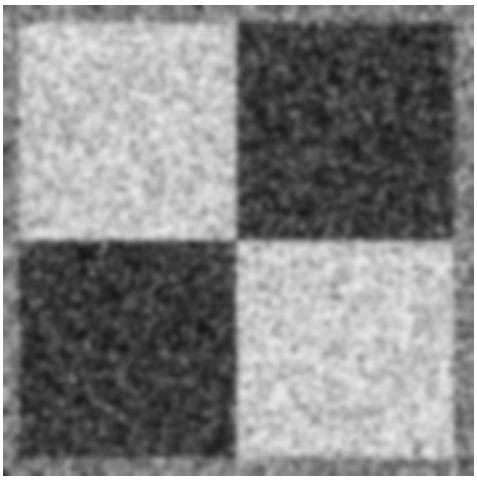
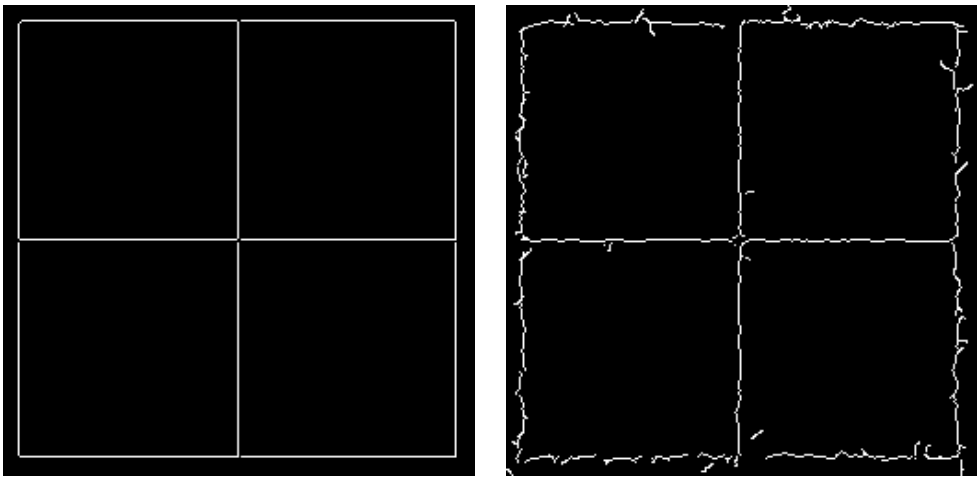


Figure 3: ps1-3-a : Image noisy smoothed

The smoothed version using a Gaussian kernel is visible in Figure 3.

3.2 Second Step : Edges detection

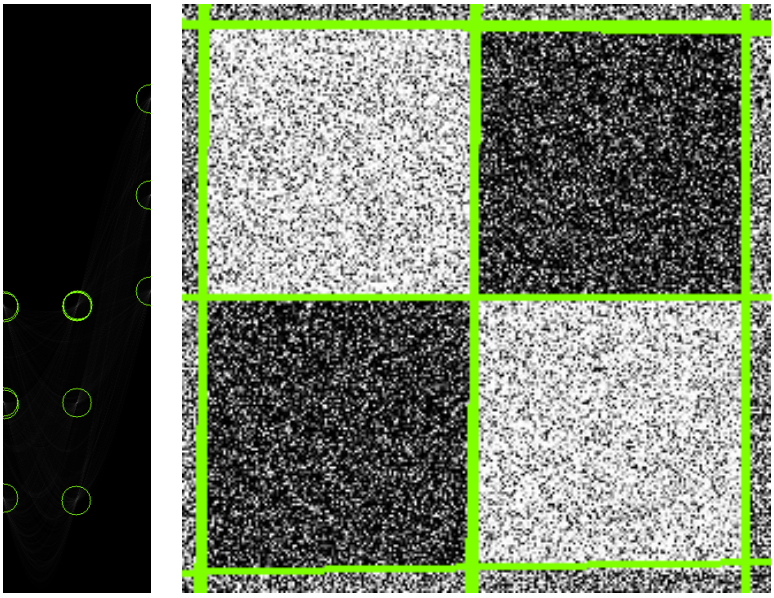


a b

Figure 4: *a.* ps1-3-b-original : Edges Original Image. *b.* ps1-3-b-noisy : Edges Smoothed Image.

The edges resulting from the Canny edge detector on the original image are shown in Figure 4.a, while those detected on the noisy image are visible in Figure 4.b.

3.3 Third Step : Hough Algorithm for lines



a b

Figure 5: *a.* ps1-3-c-accumulator : Hough accumulator array image. *b.* ps1-3-c-lines : Intensity Image with lines drawn on them.

In order to get a good result, we first had to convolve the image with a standard 3x3 Gaussian kernel in order to get an image smoothed with a sigma of a few pixels.

Afterwards, we used the Canny edge detector in order to get a relatively good quality of edges, meaning at least getting each of the 6 edges about halfway visible on the edge image. Even if some artifacts are still visible, in will not matter much since the majority of edges still lie on a line.

I finally used the Hough line detection method to extract the 6 edges, which required lowering the threshold to get all of them. We see on Figure 5.b that vertical edges are represented by multiple lines instead of a single one. This is due to the fact that peaks on the edge of the theta axis get wrapped around the matrix and detected twice). This results in several lines with a little difference in angle.

4 Lines detection on a more complicated image

4.1 First Step : Smoothing

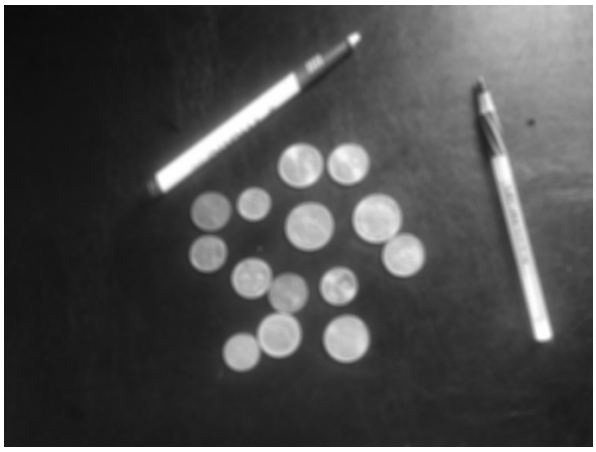


Figure 6: ps1-4-a : Image noisy smoothed

The smoothed version of the pens and coins is visible in Figure 6.

4.2 Second Step : Edges detection

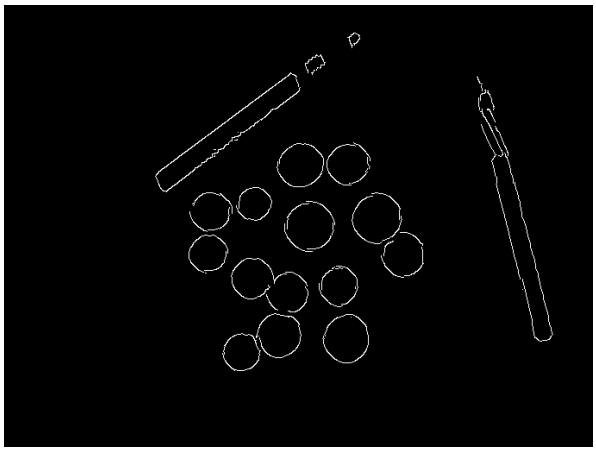
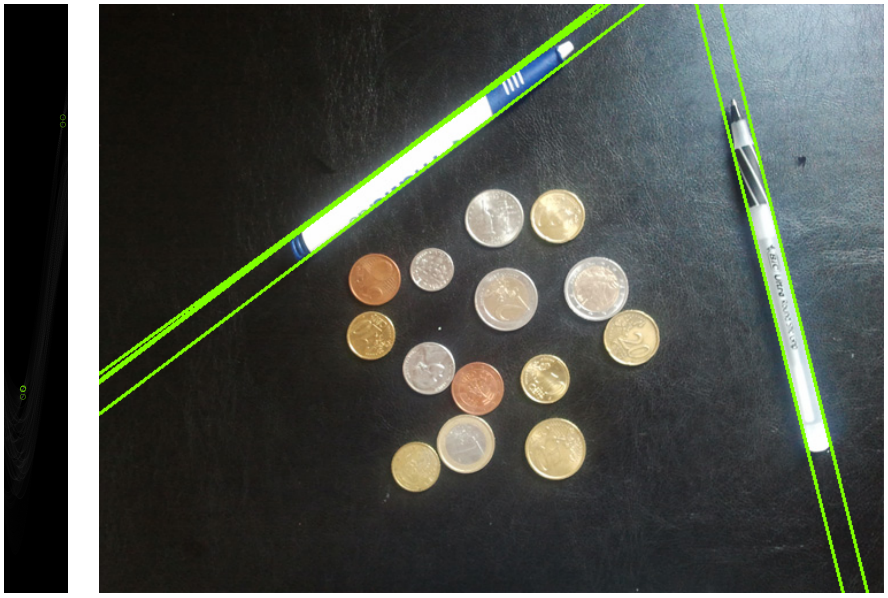


Figure 7: ps1-4-b : Edges Smoothed Image

The Canny edge detection result is given in Figure 7.

4.3 Third Step : Hough Algorithm for lines



a b

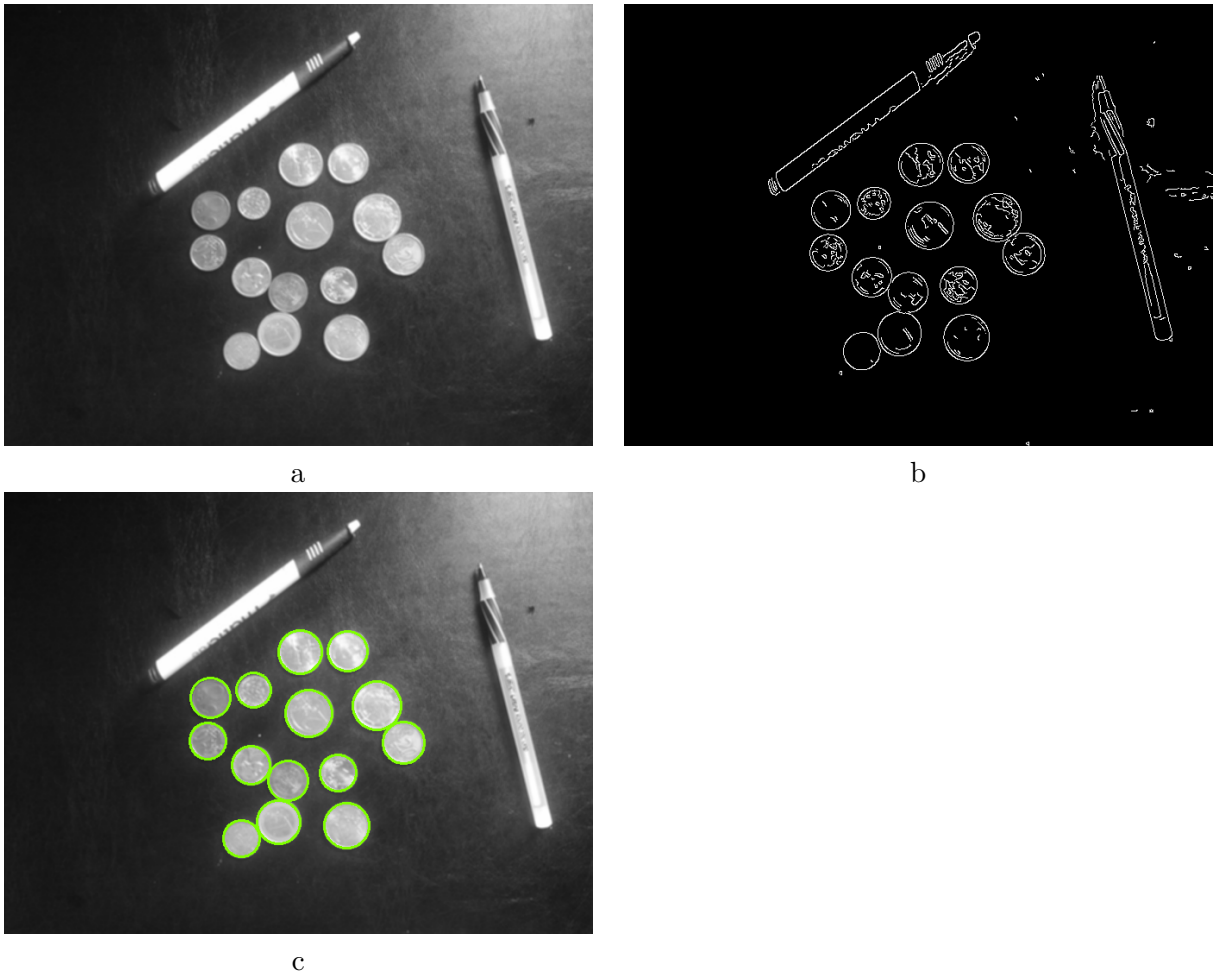
Figure 8: *a.* ps1-4-c-accumulator : Hough accumulator array image. *b.* ps1-4-c-lines : Intensity Image with lines drawn on them.

In order to get to the cleanly detected lines of the pens (Figure 8.b), we had to follow a similar process as previously.

This time we used a larger Gaussian kernel to get a wider extent of smoothing in order to avoid detecting any of the inner edges of the pens. The circles of the coins are less of a problem once we manage to keep them clean, since our Hough algorithm will target specifically the lines.

The resulting accumulator visible in Figure 8.a shows pair of peaks, corresponding to the parallel edges of the pens. We have twice the number of peaks as what is expected, since they are spaced of 180 degrees and result in superposing lines on the final image.

5 Circles Detection



a b c

Figure 9: *a.* ps1-5-smoothed : Image Noisy Smoothed. *b.* ps1-5-edges : Edges Smoothed Image. *c.* ps1-5-circles : Image with circles drawn in color.

In order to detect the coins, we need to adapt the Hough transform to circles. We change our Hough space parameters, from polar coordinates to 3 parameters: the cartesian coordinates of the circle center and the radius of the circle.

We use a slightly lighter smoothing than for the pen detection, since we do not care about their inner edges anymore (none of them is circular). We also use the same Canny edge detector parameters, which gives us the edges visible in Figure 9.b.

We estimate the radius of the coins to be between 15 and 30 pixels, which is necessary in order to limit the computation time of our algorithm. This is also something the standard OpenCV Hough detector offers.

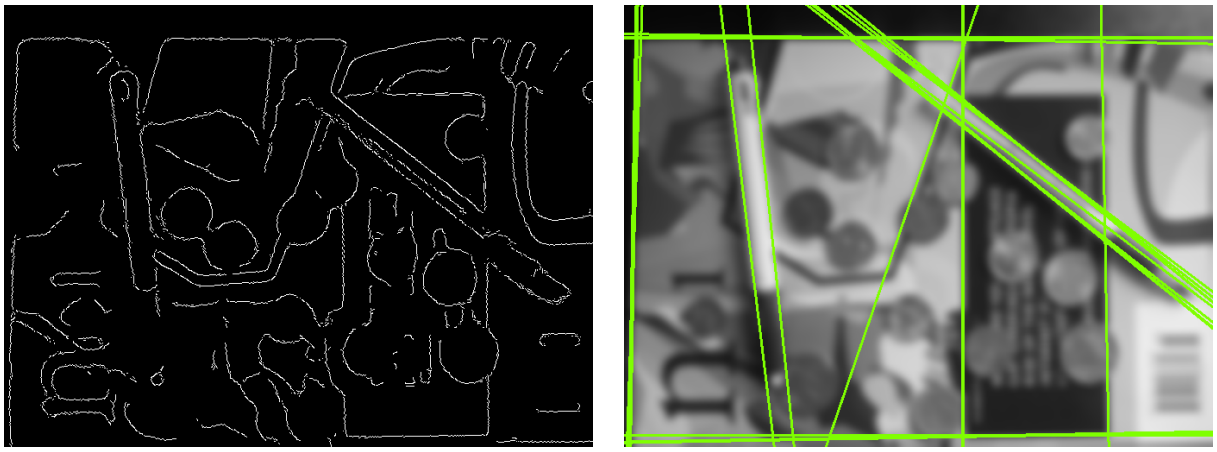
For each edge point and each radius, we first computed the coordinates of the center for 180 values but in order to reduce the length of the computation an improvement of the algorithm was made by using the gradient to ?orient? the search for a potential center. We finally filter

the circles using an extra parameter which gives the minimum distance between two centers, thus avoiding concentric circles (which are not among our targets for this application) and also filtering the closeby maximas. We get the resulting image shown in Figure 9.c.

6 Finding lines on a more realistic image

6.1 Application on a line finder

In the Figure 10.b are displayed the lines initially detected by our Hough transform (after a good smoothing to put aside the numerous edges of the text and barcode).



a b

Figure 10: *a.* ps1-6-a-edges : Edges Image Smoothed. *b.* ps1-6-a-lines : Hough Lines drawn on the smoothed image.

6.2 Problems with our line finder

We note that many lines are rightfully detected by the algorithm because of the magazine cover borders and geometric content. However, we successfully avoided any of the smaller text and barcode edges.

As explained in the previous section, we will rely on the fact that the pens are composed of parallel edges, spaced by a certain amount of pixels, to filter out the undesirable edges.

6.3 Attempt to find the boundaries lines of the pen

Applying a maximum distance and specifying the maximum angle between two lines to be considered parallel, we manage to extract the pens boundaries as visible in Figure 11.

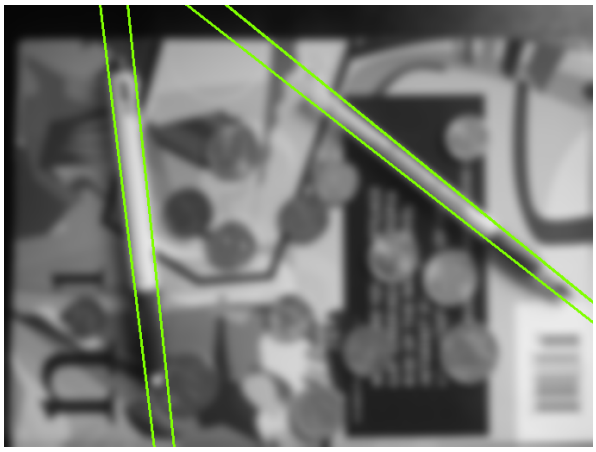
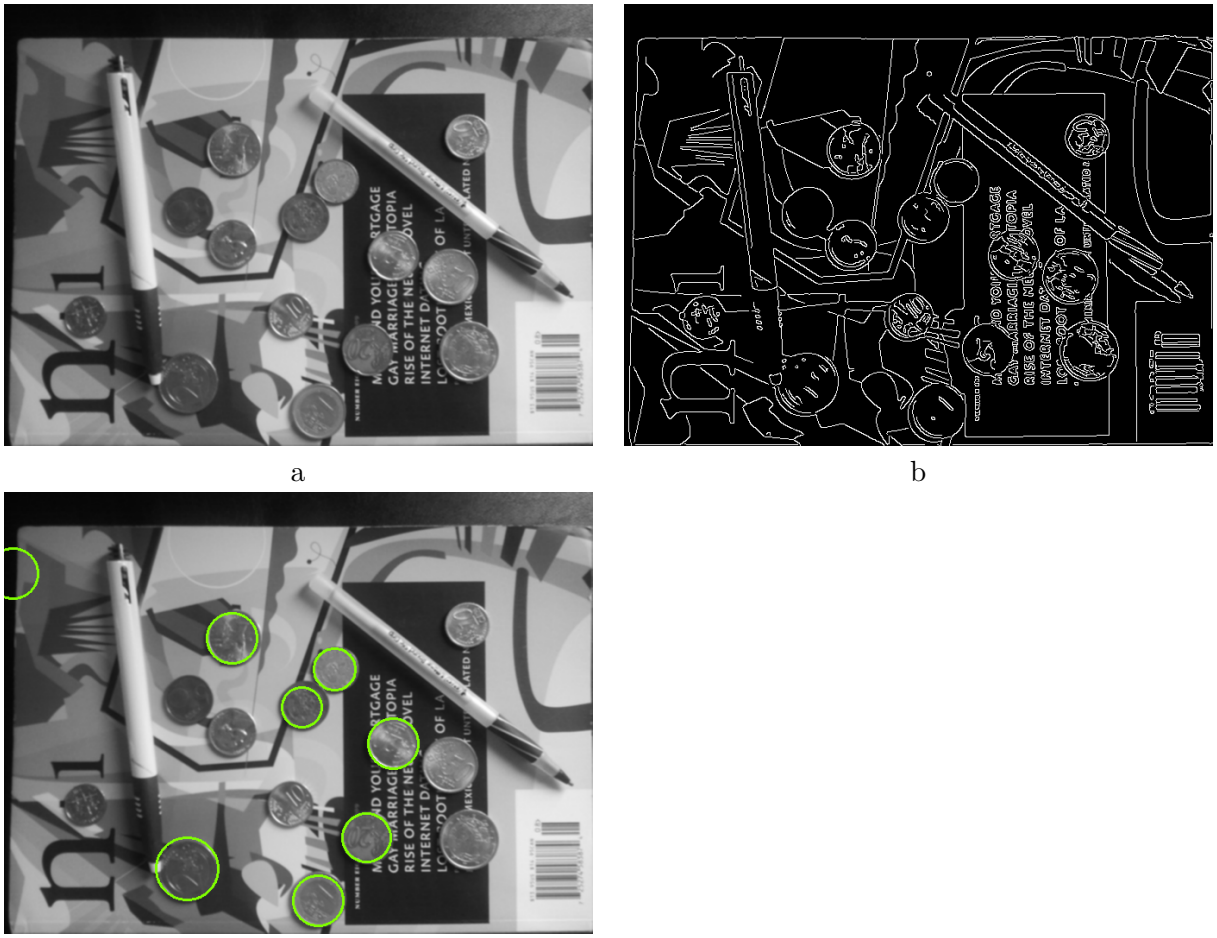


Figure 11: ps1-6-c :New Hough lines drawn on the smoothed image

7 Finding Circles on the same clutter image

Applying the Hough circle transform on the cluttered image, we manage to detect most of the coins, as shown in Figure 12.



a b

Figure 12: *a.* ps1-7-a-smoothed : Smoothed Image. *b.* ps1-7-a-edges : Edges Image Smoothed. *c.* ps1-7-a-circles : Hough Circles drawn on the smoothed image.

8 Sensitivity to distortion

8.1 Application of the line and circles finder

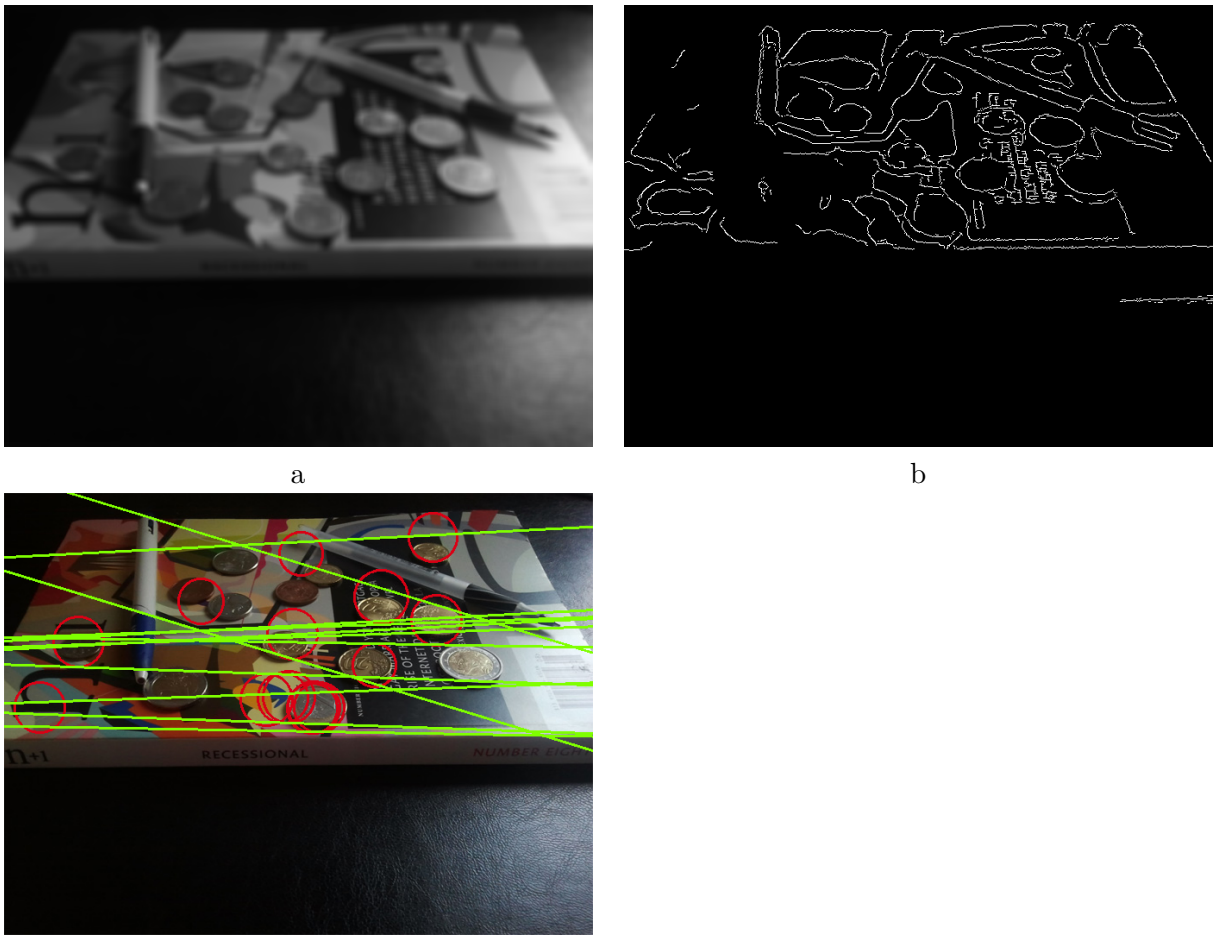


Figure 13: *a.* ps1-7-a-smoothed : Smoothed Image. *b.* ps1-7-a-edges : Edges Image Smoothed. *c.* ps1-7-a-circles : Hough Circles and Hough Lines drawn on the smoothed image.

Our two Hough transforms applied to a distorted image do not perform well (Figure 13.a). Even with a parallel lines filtering, which should help detect the pens even in this setup, we do not manage to extract any of their boundaries but still get unwanted edges.

However, the circles detection seems to perform slightly better, despite plenty of false alarms it seems that we detect several coins, as well as the false circle visible in the "n" letter.

8.2 How to fix the problem ?

In future enhancements of our program we might try to adapt the circle transform to detect ellipses. We would however probably face at least the same amount of false alarms as in the previous section, and would also need to implement further improvements. Regarding the

lines, we would need to adapt the smoothing and edge detection to also take into account the stronger impact of the shadows on this distorted version of the image. We could also add constraints on the filtered lines to look for lines at specific angles and with a specific number of close parallel edges.