Computer Vision
CS 6476 , Spring 2018

PS 1

Professor :
Cedric Pradalier

Author :
Melisande Zonta

February 5, 2018

Contents

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 is the result of the Canny edge detector to perform the original edge detection.

2 Hough Method applied on the lines

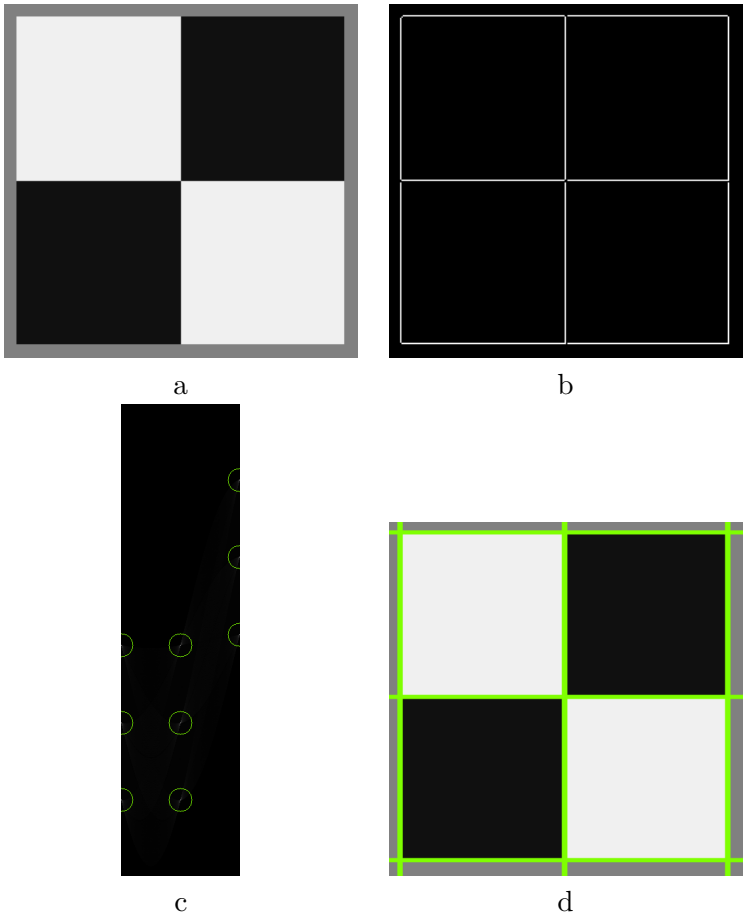


Figure 2: *a.* ps1-2-original : Original Image. *b.* ps1-2-edges : Edges Image.
c. ps1-2-accumulator : Hough accumulator array image. *d.* ps1-2-lines : Intensity Image with lines drawn on them.

The Figure 2-c is the result of the voting in the accumulator and the figure 2-d is the associated image with the detected lines. In order to get the finest detection possible, a choice was made on the bins size.

Concerning θ the length of the bin is

$$\frac{180}{grid_{size}} + 1$$

. Concerning d, the range for values varies from $-d_{max}tod_{m}axwithd_{m}ax = \sqrt{m^2 + n^2}$, m the number of rows of the image and the number of columns. Hence the length of the bin is $2 \times d_{max} + 1$.

The image being a 256px square, the maximum value of d is the diagonal, around 362. Thus the choices we made for the bin result in a rectangular accumulator.

3 Noise effect on the lines detection

3.1 First Step : Smoothing

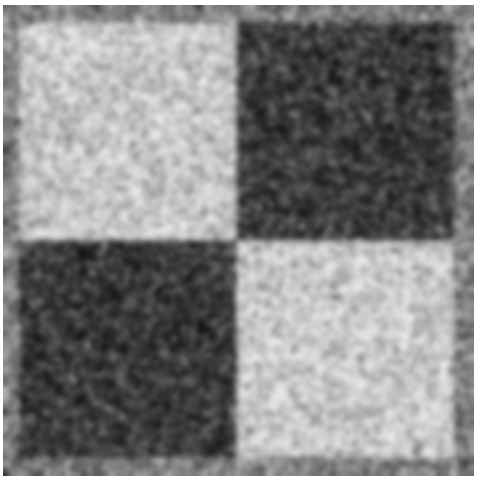
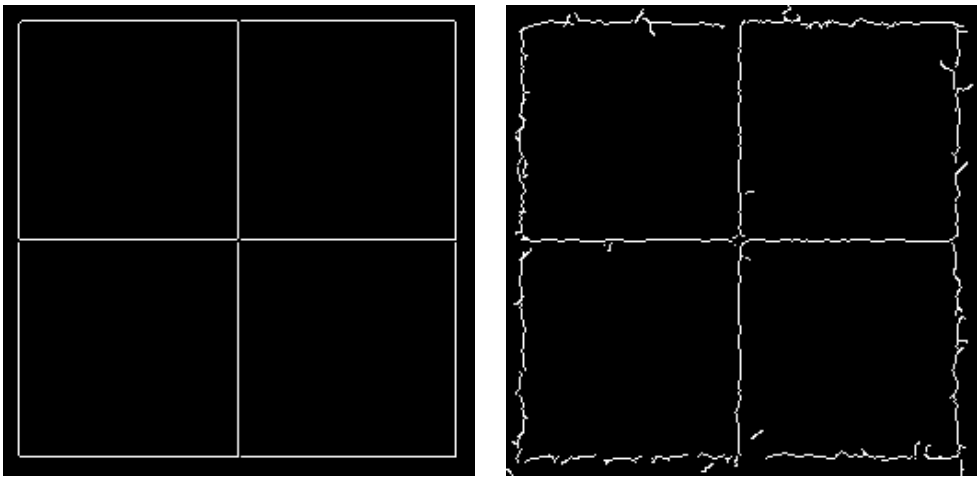


Figure 3: ps1-3-a : Image noisy smoothed

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

3.2 Second Step : Edges detection

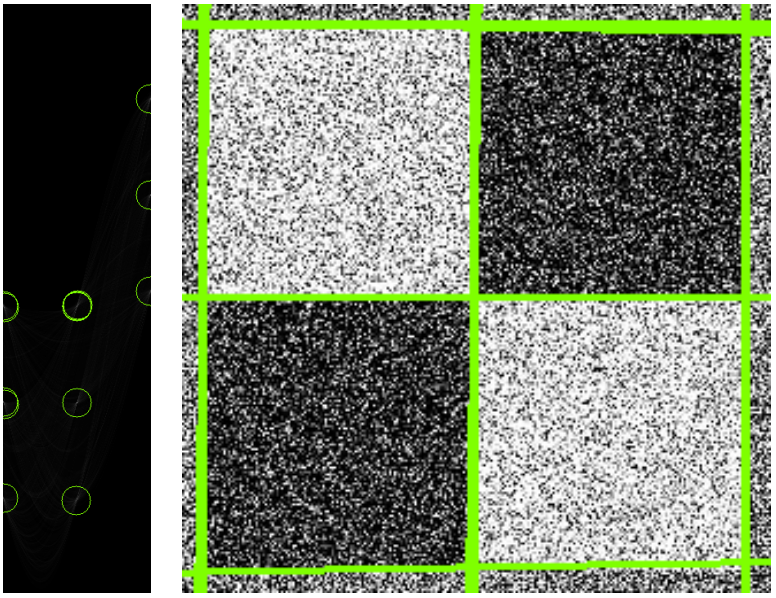


a b

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

The figure 4.a show the edges resulting from the Canny edge detector on the original image. The figure 4.b show the edges detected on the noisy image.

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 on the noisy image, we first had to get an image smoothed with a sigma of a few pixels meaning we first had to convolve the image with a standard Gaussian kernel.

Afterwards, we used the Canny edge detector in order to get a relatively good quality of edges, meaning getting each of the 9 edges as in the previous part.

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

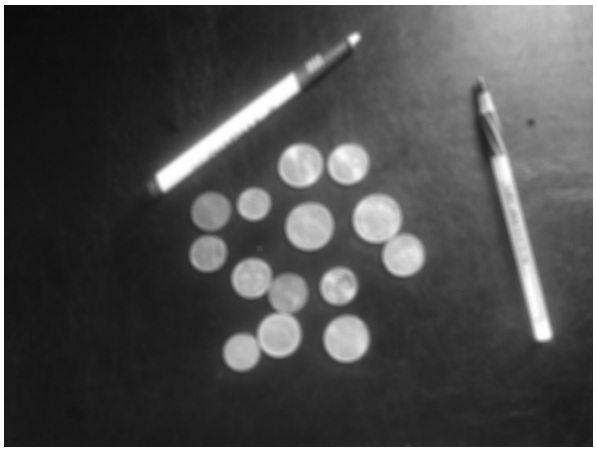


Figure 6: ps1-4-a : Image noisy smoothed

The Figure 6 shows the smoothed version of the pens and coins.

4.2 Second Step : Edges detection

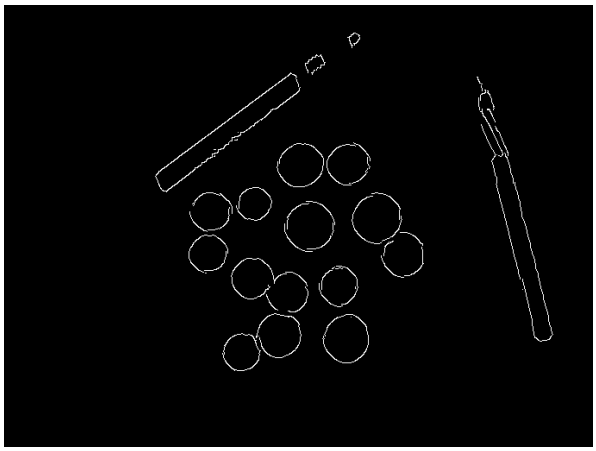
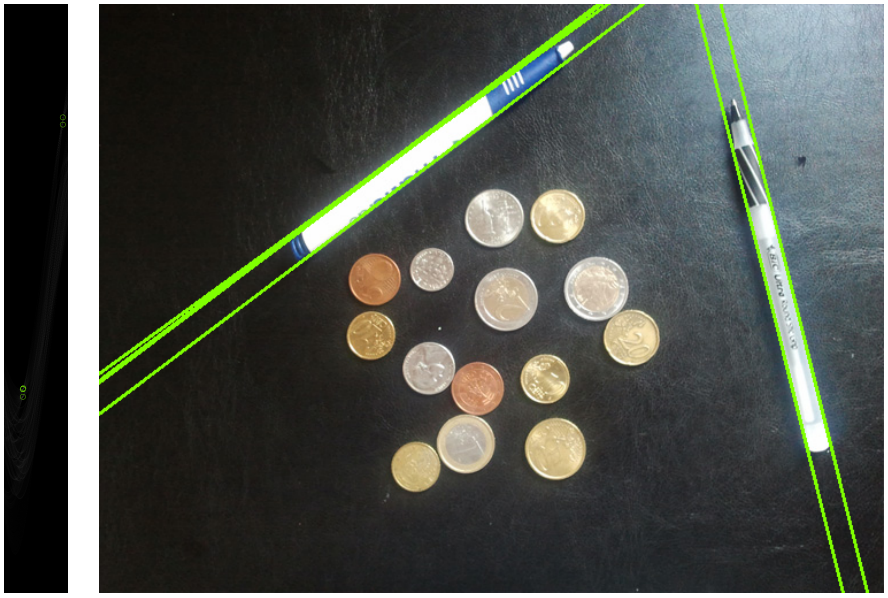


Figure 7: ps1-4-b : Edges Smoothed Image

The Figure 7 shows the Canny edge detection result.

4.3 Third Step : Hough Algorithm for lines



a b

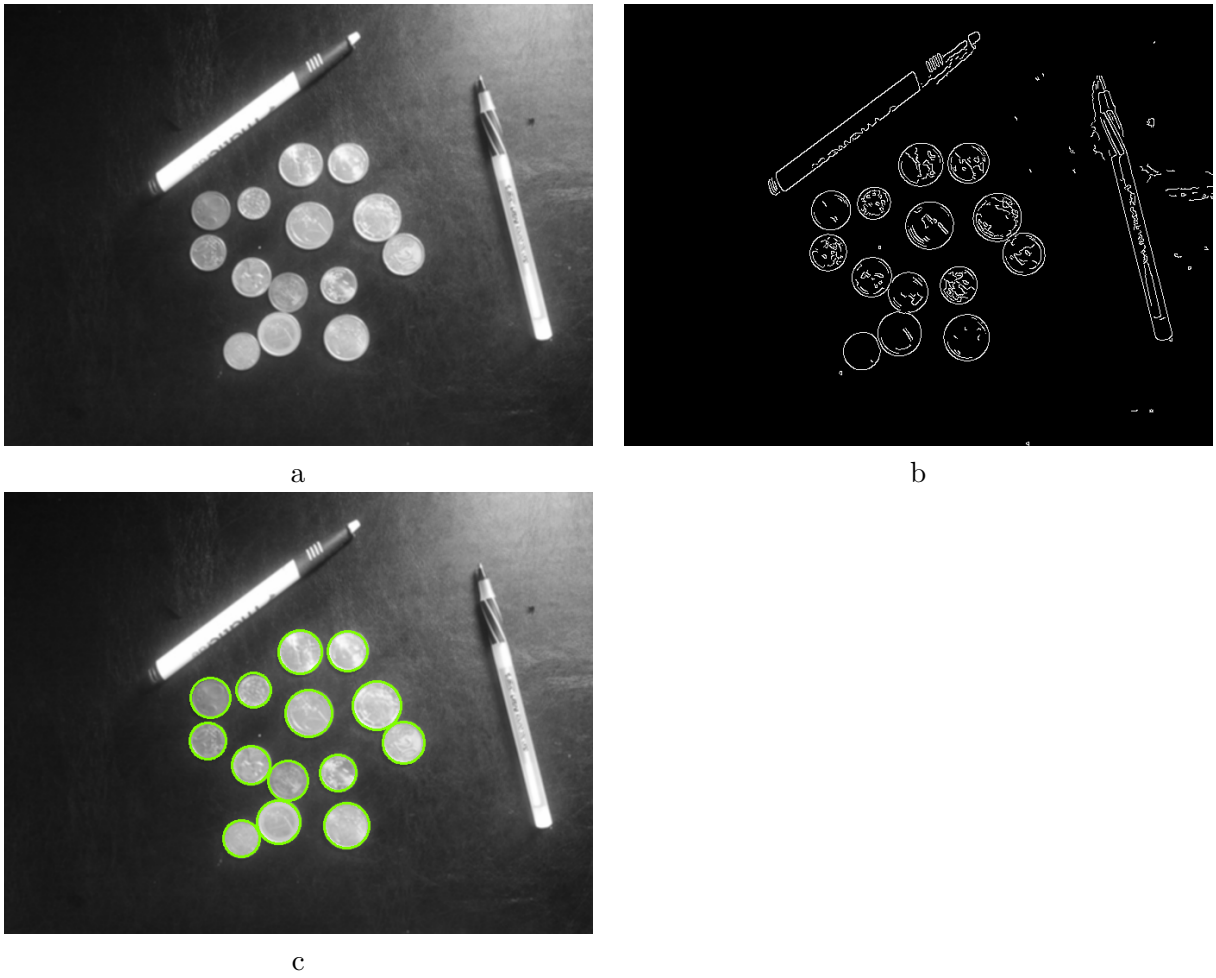
Figure 8: *a.* ps1-4-c-accumulator : Hough accumulator array image. *b.* ps1-4-c-lines : Intensity Image with lines drawn on them.

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.

Now our aim is to detect the coins on the image, we have to change the Hough transform to adapt it for circles meaning we have to take in account 3 parameters which are the cartesian coordinates of the circle center and the radius of the circle.

We first have to operate a light smoothing indeed it's not as important as in the previous question since the inner edges are not circular. The Figure 9.b shows the result of the Canny edge detector.

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.

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.

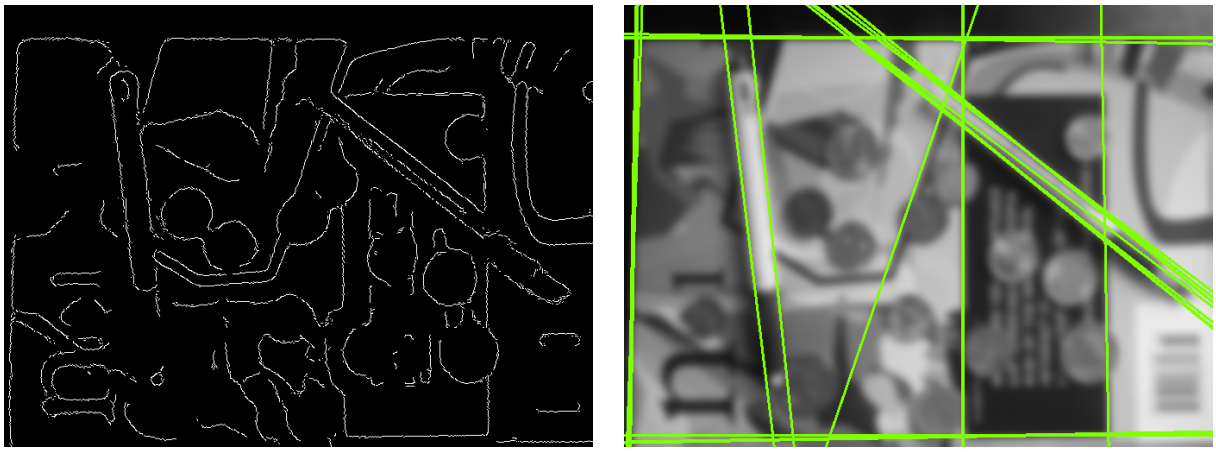
Finally since the result gave some concentric circles superposed instead of a single, a

"filtering" was made by taking into account the distance between the centers and by applying a threshold on it. A good solution seemed to take the mean circle for those really closed circles. We save then the radius and the center coordinate for that mean circle. It allows us to obtain the Figure 9.c.

6 Finding lines on a more realistic image

6.1 Application on a line finder

The Figure 10.b is the result of the lines detection made by the Hough transform for lines after having tried to smoothe to put aside the 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

Many lines are rightfully detected by the algorithm because of the magazine cover borders and geometric content.

Now since we want to focus on the boundaries of the pen, we will rely on the fact that those lines are parallel edges, space by a minimal distance, in order to filter out the undesirable edges.

6.3 Attempt to find the boundaries lines of the pen

Applying a maximum distance (to avoid the magazine cover borders which are parallels too), a minimum distance (to avoid the superposed lines)and specifying the maximum and minimum angle between two lines to be considered parallel, we manage to extract the pens boundaries as visible in Figure 11.

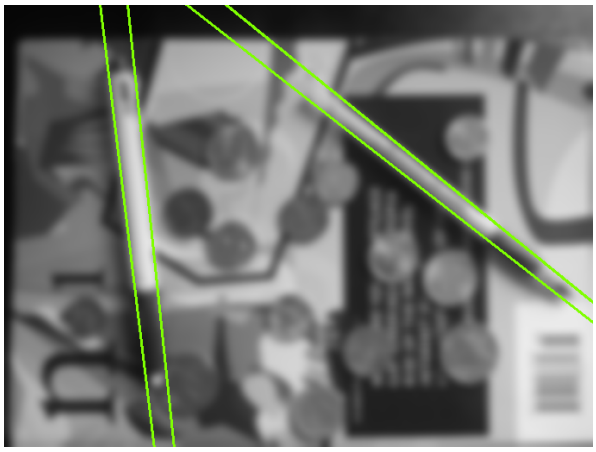
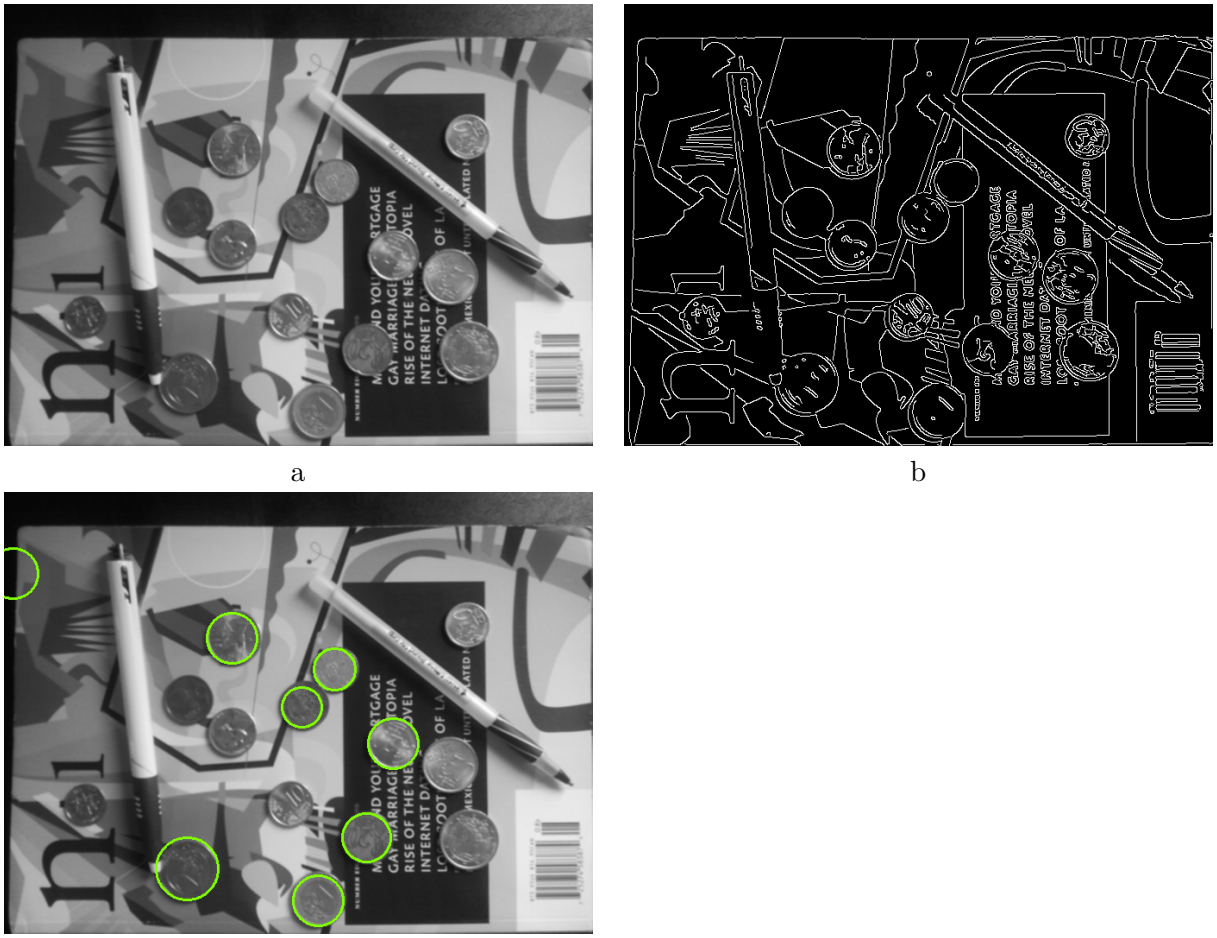


Figure 11: ps1-6-c :New Hough lines drawn on the smoothed image

7 Finding Circles on the same clutter image

By applying the Hough circle transform on the cluttered image, we manage to detect some of the coins but not all. A tradeoff has to be made on the threshold. A lower threshold, even after having realized a really effective smoothing (as shown on Figure 12.b), was providing a lot of false alarms. Here this threshold avoids the false alarms but does not allow to obtain all the circles. The result is presented on Figure 12.c.



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

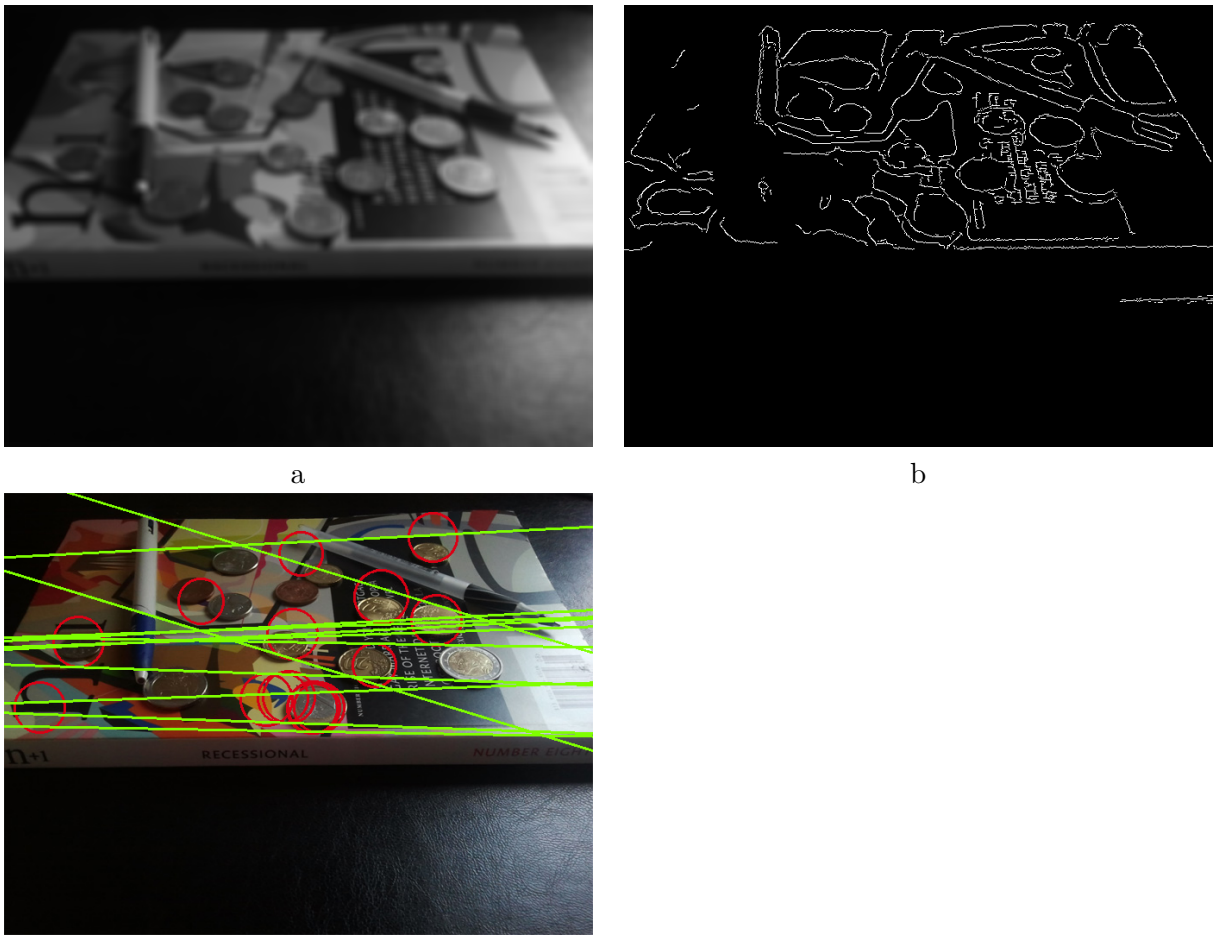


Figure 13: *a.* ps1-7-a-smoothed : Smoothed Image. *b.* ps1-7-a-edges : Edges Image Smoothed. *c.* ps1-7-a-circles : Hough Circles and Hough Lines drawn on the smoothed image.

Our two Hough transforms applied to a distorted image do not perform well (Figure 13.c). 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 ?

Future enhancements of our code would be to adapt the circle transform to an ellipse one. Indeed with the distortion, the circles become 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.