Introduction to Image and Video Processing

Coronaproject 1: spatial filtering, processing

Selim Gilon* - i6192074

*Maastricht University - Department of Data Science and Knowledge Engineering

April 19^{th}, 2020

Here are the two chosen images for this assignment:

Figure 1: Image 1 - with strong edge directionality

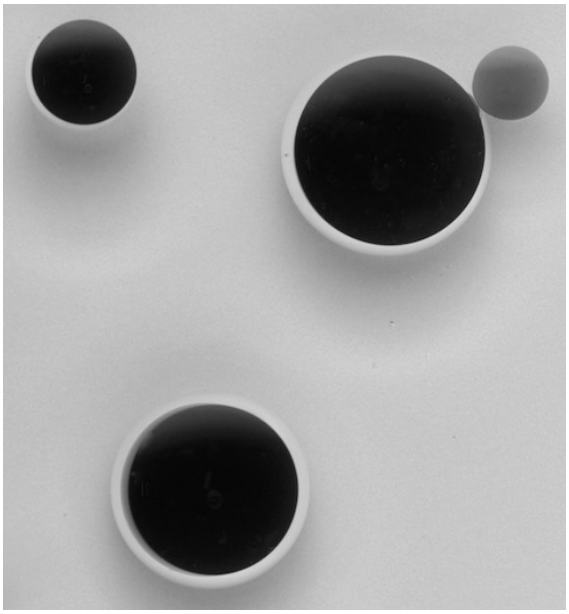

Figure 2: Image 2 - with strong circular features

1 Spatial Filtering - edges

1. Apply 1st order gradient edge detection to your two images with:

   (a) A first gradient spatial mask (Prewitt, Sobel).

   I started by applying the Sobel edge detection using [-1 0 1;-2 0 2;-1 0 1], [-1 -2 -1;0 0 0;1 2 1], [-2 -1 0; -1 0 1; 0 1 2] and the function imgfilter in order to respectively detect the vertical, horizontal and the diagonal edges.

Figure 3: Sobel Edge Detection using matrices and imgfilter (from L to R: vertical, horizontal and diagonal edge)

Figure 4: Sobel Edge Detection using matrices and imgfilter (from L to R: vertical, horizontal and diagonal edge)

I then used the edge detection Sobel from Matlab (edge(xd,'sobel')).

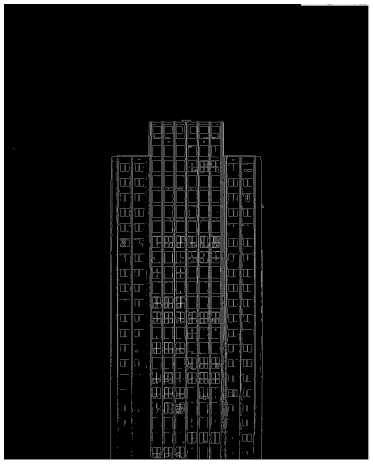

Figure 5: Sobel Edge Detection from Matlab

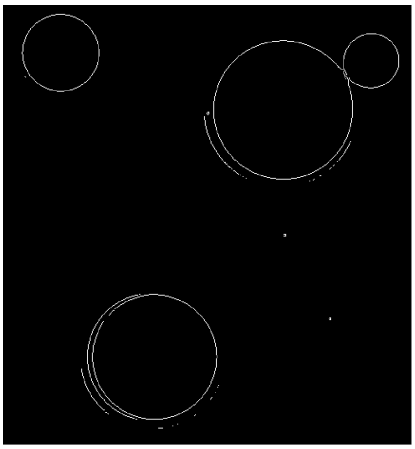

Figure 6: Sobel Edge Detection from Matlab

Finally, I implemented my own function for the Sobel edge detection so I neither use the imgfilter function nor the edge detection function.

Figure 7: Sobel Edge Detection using my implementation (horizontal - vertical - both)

Figure 8: Sobel Edge Detection using my implementation (vertical - horizontal - both)

(b) Canny edge detector.

I used the edge Canny edge detection function from matlab.

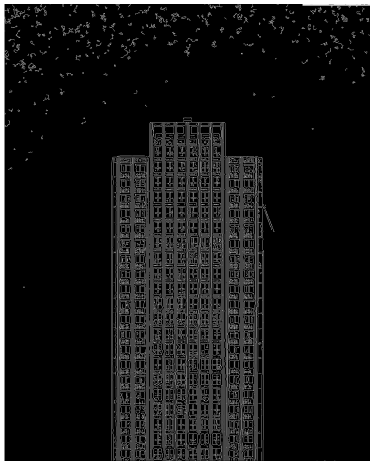

Figure 9: Canny Edge Detection from Matlab

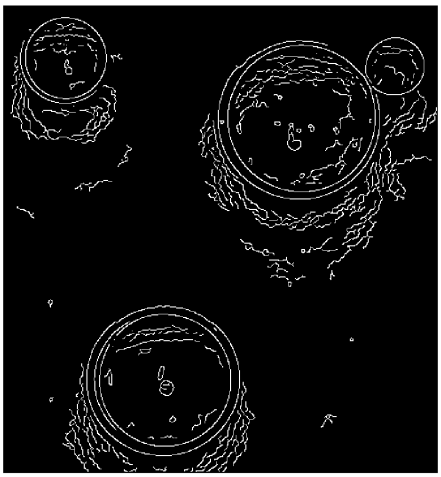

Figure 10: Canny Edge Detection from Matlab

Comparing the results from the Sobel edge detection and from the Canny edge detection, we can see that the Canny method provides more detailed edges, it seems to perform better. I would say that this is logic since Canny is an optimized edge detector since it first removes the noise from the image and then suppress the edges that are considered not dominant compared to the ones next to them. From my experiments, I see Canny as an improved version of Sobel. Still, we can see that Sobel is doing very well since it detects most of the edges of both images. However, when using the function imgfilter and the different kernels, it is performing better on the building image than on the circles ones. The results from the built in Sobel edge detection and from my own implementation of the same technique are a little bit different. Indeed, if you look at the building on figure 5 and the third image from figure 8, we can see on the result of my function that there are different "types" of edges while there is only one "type" on the result from the built-in function. Also, the edges are a bit less thicker on this one.

2. Add noise to the images and apply your favorite 1st and 2nd order gradient edge detectors to them.

   (a) Show and discuss your results of 1st and 2nd order gradient edge detection on the noisy images. I used the Sobel edge detection and the Laplace edge detection on images with Gaussian noise.

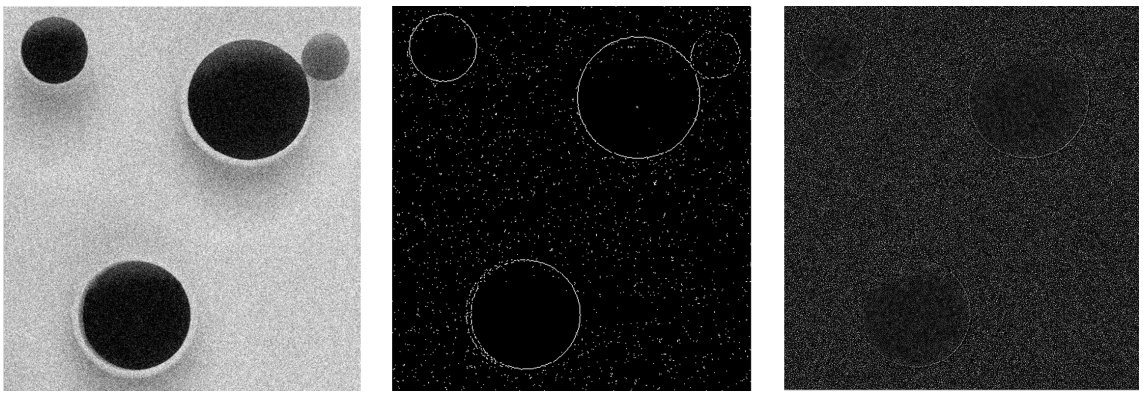

Figure 11: L to R: Image with Gaussian noise - Sobel edge detection - Laplace edge detection

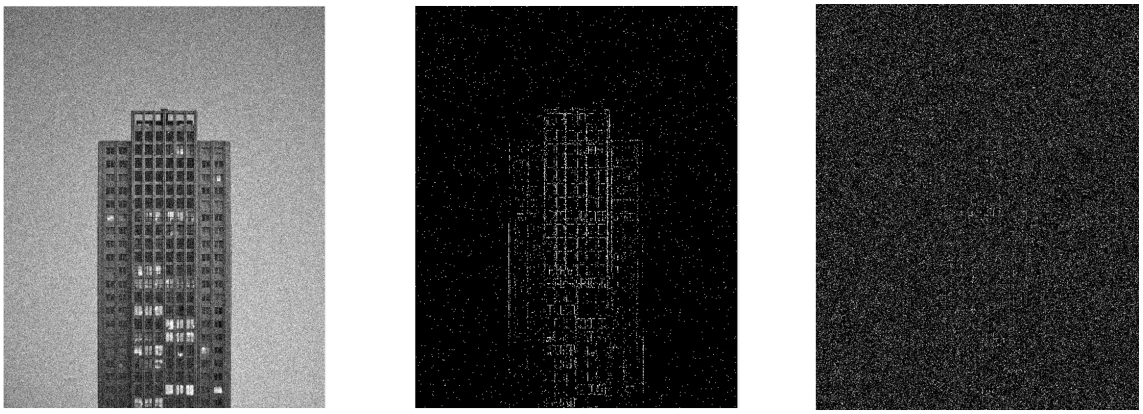

Figure 12: L to R: Image with Gaussian noise - Sobel edge detection - Laplace edge detection

We can see that with noisy images, the 1st order gradient edge detection is performing better than the 2nd order one. We can barely see any edges on the results of the Laplace gradient. Those results are pretty logic since as we studied in class (see figure below), 2nd order gradient are very sensitive to noise (even more than 1st order ones). As said in the book Gonzalez chapter 10.1.3: "... fairly little noise can have such a significant impact on the 2 key derivatives used for edge detection in images ...".

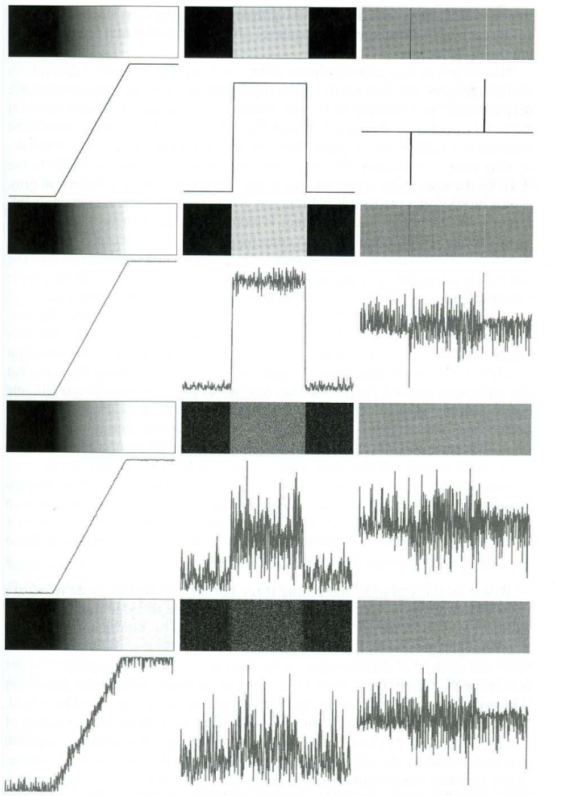

FIGURE 10.7 First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and $\sigma = 0.0, 0.1, 1.0$, and 10.0, respectively. Second column: first-derivative images and gray-level profiles. Third column: second-derivative images and gray-level profiles. a b c d

Figure 13: Graph from the lecture

(b) How can you improve the results of the 2nd order gradient edge detection? Apply this, show and discuss your results.

I believe I could improve the result of the 2nd order gradient edge detection by first, blurring/smoothing the image with a Gaussian filter and then apply the Laplacian. In this way, I would reduce the noise on the image (reduce the enhancement of the noise) and that would improve the results since the 2nd order gradient are very sensitive to noise. Let's try it (with the Laplacian of Gaussian) and see the results.

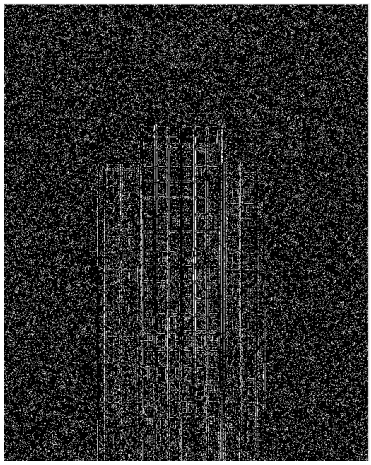

Figure 14: Laplacian of Gaussian

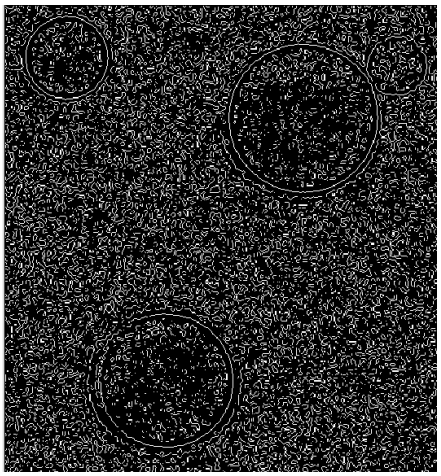

Figure 15: Laplacian of Gaussian

As expected, the result are better.

2 Geometric transformations - image warping

1. Transform the Cartesian coordinates to polar. Warp the previous two images to the polar coordinates system.

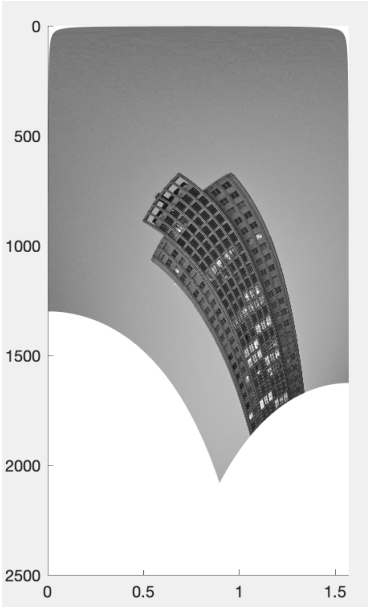

Figure 16: polar coordinates

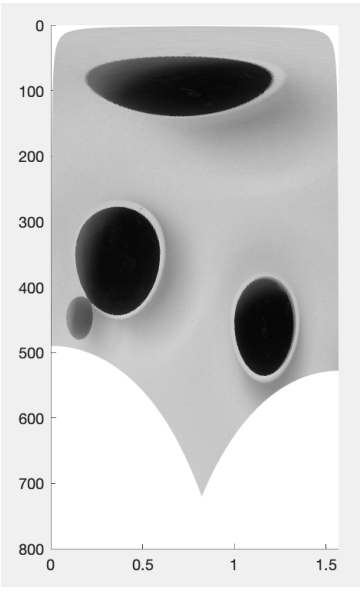

Figure 17: polar coordinates

Note that the x axis corresponds to the angle theta (in radians) and the y axis corresponds to rho.

2. Show and discuss your results, especially with respect to the appearance of strong linear edges and circular features of each image in each coordinate system.

   The smaller rho is, the bigger the deformation is. Also, it seems like the circular patterns are less deformed than the linear edges. Another observation is that the linear edges in cartesian coordinates tend to become more circular in the polar form while the circular patterns in the cartesian coordinates tend to become more linear in the polar coordinates. In order to give you a feedback: unfortunately, I feel like we didn't discuss enough about those materials during the lecture and it is very unpractical to not be able to meet in person to ask you questions. We can email you but still, it is not the same because it takes a lot more time and it is way harder to communicate.

3. Choose an image that will apply special effects to. It can be your face if you want, or any photo you took etc. Apply two image warping transformations of your liking, such as the glass effect.

   Here is the original image:

Figure 18: Original image of me

Here is a wrapped image using a sinus transformation. The coordinates change with the sinus function. We can see the horizontal waves on the image.

On the right is a wrapped image using a sinus and cosinus transformation. The coordinates change with the sinus and the cosinus function. We can see the horizontal and vertical waves on the image.

Figure 19: Wrapped image of me with horizontal wave (sinus)

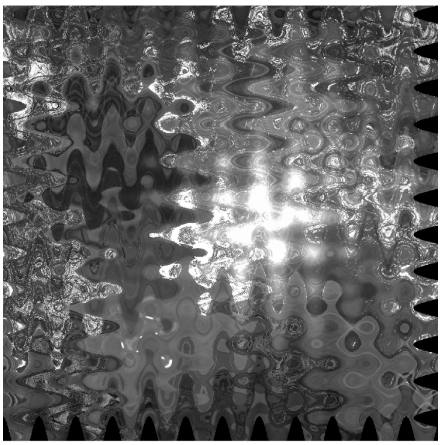

Figure 20: Wrapped image of me with vertical and horizontal waves

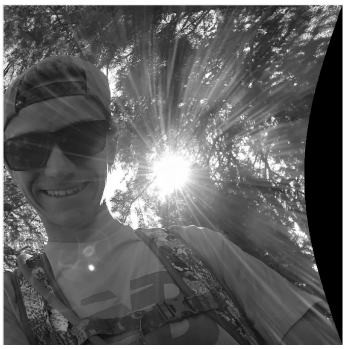

Figure 21: Image of me with a concave effect