Assignment 2

Nalet Meinen
Finite Element Analysis I

April 10, 2019

Contents

1 Introduction

In this assignment three main methods will be analyzed: linear filtering, edge and corner detection. All of those things base on the same algorithms or functions witch have to be implemented.

2 Methods

2.1 Linear Filtering

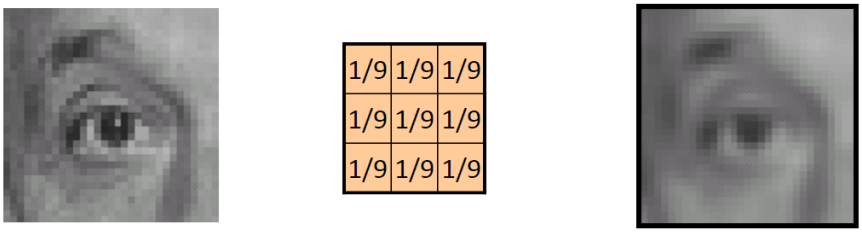

Figure 1: Example of a boxfilter

Linear filterings belongs to the group of neighbour analysis [1]. As the name neighbour analysis says, the goal is to choose a pixel and analyse its neighbours. With the information of the neighbours and depending on our processing strategy it is made possible to modify or enhance images. Applied cases can be smoothing an image or even face recognition. Fig. 1 shows a boxfilter using a 1/9 of all the pixels of the neighbours and the center giving us a blurred image.

2.2 Finding Edges

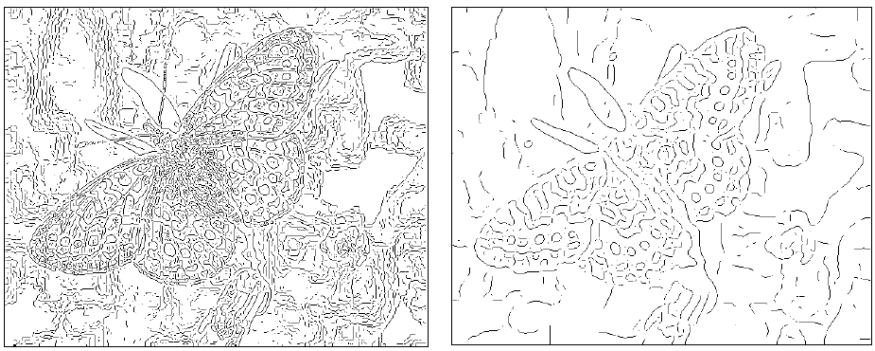

Figure 2: Canny edge detection with (left) high threshold and (right) low threshold

Edges are boundary pixels which correspond to changes in gradients of physical changes in object appearance. The most shapes in an image are consisting of edges. In Fig 3 a so called canny edge algorithm is applied. The high and low threshold determines the density

of edges we would like to have. Edge detection lies the fondation for basic shape detection in signal and image processing.

2.3 Corner Detection

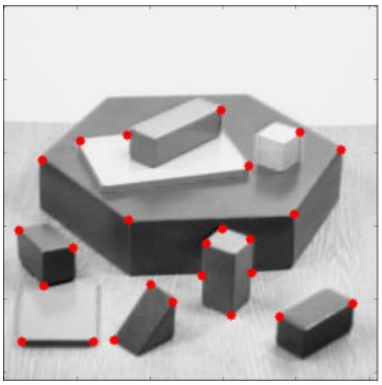

Figure 3: Canny edge detection with (left) high threshold and (right) low threshold

Fig. 3 shows us a image with many corners. The red dots are the corner detected. Using the two methods from above, one can achive such an image output. From a pratical perspective corner detection can be used to detect distortion from a perspective in an image or in medical appliences for calibrating instrumens in the OR[1]

[1]Operation Room

3 Results and Discussion

3.1 Linear Filtering

3.1.1 Boxfilter

The boxfilter is a simple one liner. In the development phase the boxfilter was validated with the debug function of the IDE.

3.1.2 2D Convolution

The algorithm checks the dimensions first before proceeding and if nessecary, reshaping. After preparing the output, a full convolution is made.

3.1.3 Applied Boxfilter

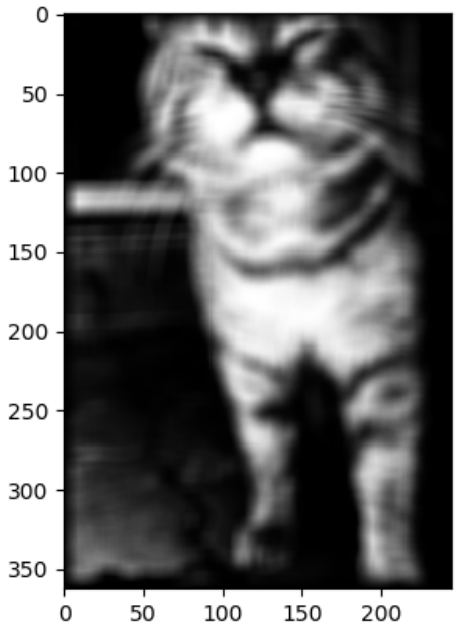

Figure 4: Applied box filter

Result of the applied box filter

3.1.4 1D Gaussian Filter

As described in the task, we using checks for making the filter odd. For performance reasons, the numpy library with array multiplications and division is used.

3.1.5 2D Gaussian Filter

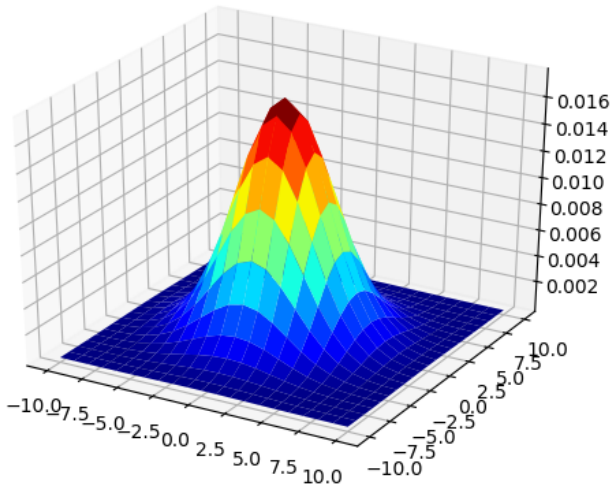

Figure 5: Gaussian kernel in 3D space

Result of the Gaussian 1D and the *myconf* function.

3.1.6 Applied 2D Gaussian Filter

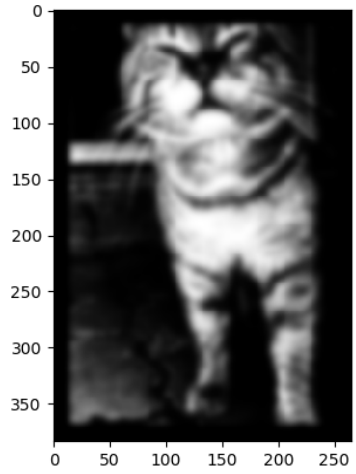

Figure 6: Result of the gaussian on an image

Applied result of our functions.

3.1.7 Convolution with a 2D Gaussien Filter

This text is also in the code.
For each pixel the operations for convolution are n*n, n equals the size, width, and height. Usually, this can be made faster with performing a 1d convolution in both directions, horizontal and vertical, resulting in 2*n operations for a pixel. Convolutions can then be split up. Looking at SVD, with one non-zero value the separations in 1d can be made.

3.1.8 Computation Time

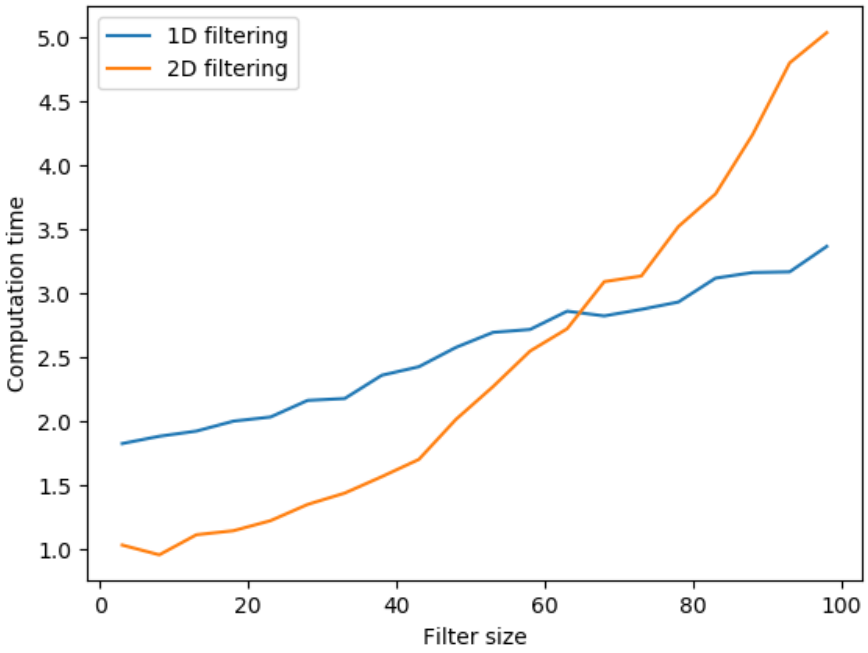

Figure 7: Compairson of 1D and 2D Gaus

We see that the slop of 2D filtering curve is much steeper, because of the factor 2 of the number of dimensions to be calculated.

3.2 Finding Edges

The needed functions are imported via the import statement of python. Measurement has been made, that the code of ex1 does not run on the import. These could be made with $if\_\_name\_\_ ==' \_\_main\_\_' :$

3.2.1 Derivative Operator

Simple implementation of the derivative operator like proposed: $[1, 0, -1]$

3.2.2 Edge Magnitude

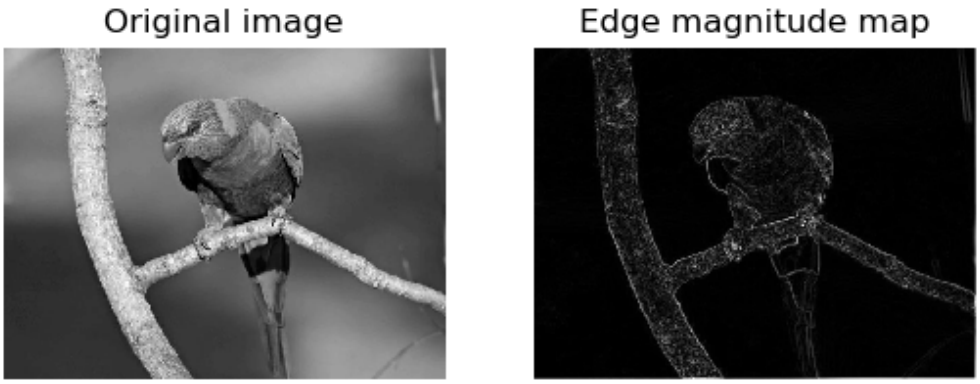

Figure 8: Edge Magnitude

Result of Edge Magnitude

3.2.3 Edge Images of Particular Directions

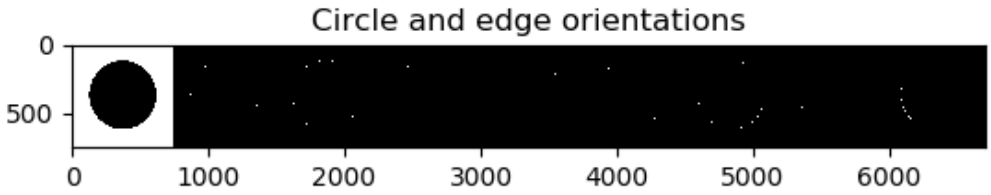

Figure 9: Result of a Circle

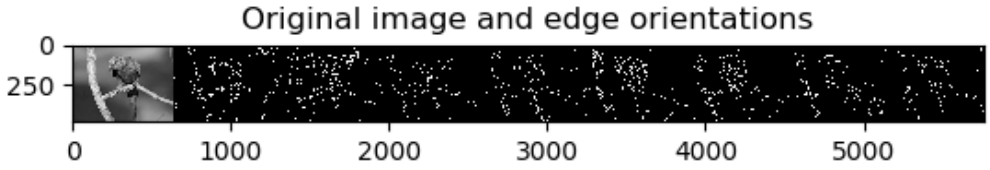

Figure 10: Result of an image

Results of the edge orientations in both images.

3.2.4 Edge Non-Max Suppression

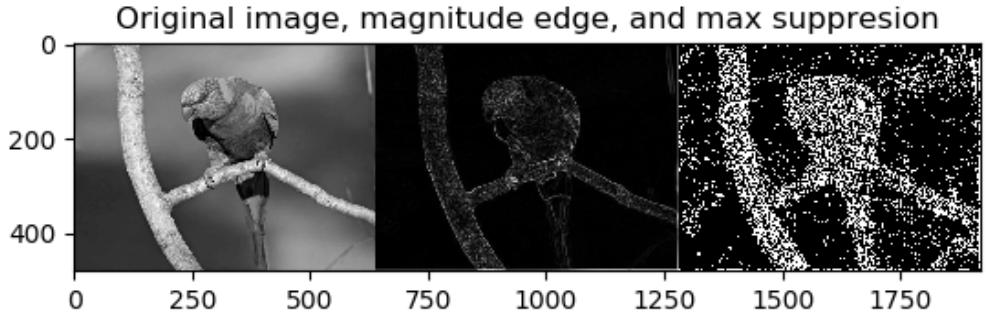

Figure 11: Image output of ex 2.4

Result of the polot of the original image, maginitude edge and the max suppression

3.2.5 Canny Edge

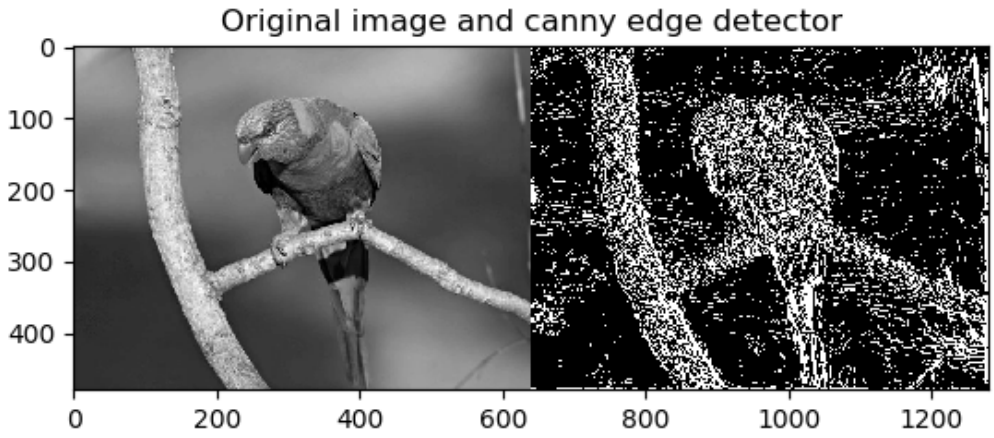

Figure 12: Result canny edge

Here a first result of the canny edge detector, the result. A different approch was taken as in max suppression. The idea was a simpler code, but the result is not quite satifying, as the threshold are not working as expected.

3.3 Corner Detection

3.3.1 Harris Corner

The Harris Corner algorithm was implemented accourding to the steps from the slides ina the lectures.

3.3.2 Evaluation of Harris Corner

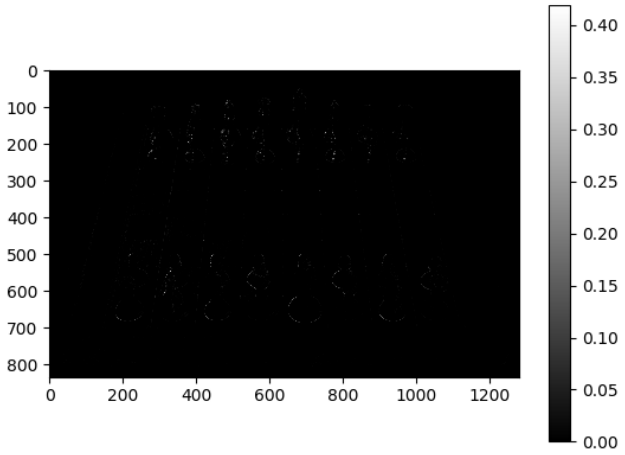

Figure 13: Result Harris Corner

Plot from python program. The plot is quite dark but the result is good, as you can see later.

3.3.3 Rotation

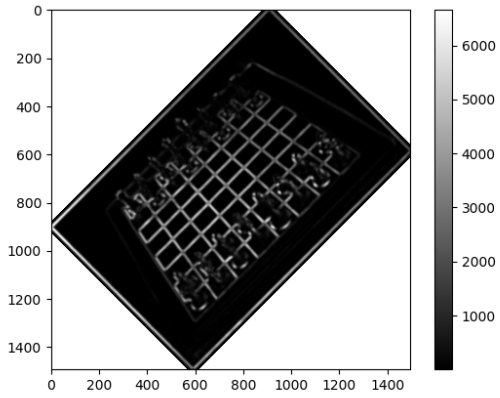

Figure 14: Result Harris Corner after the rotation

Plot from python program after rotation.

3.3.4 Factor of Halv

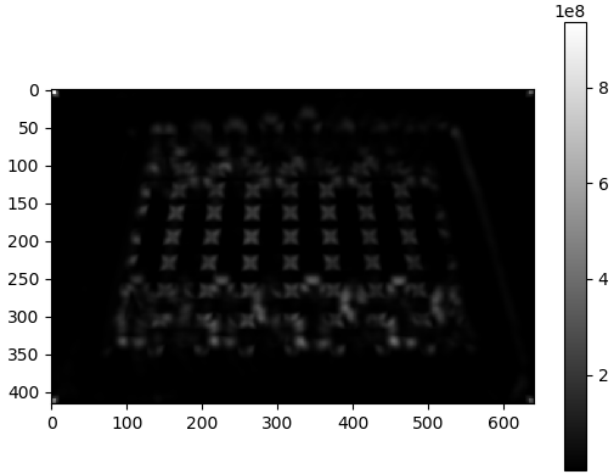

Figure 15: Result canny edge

Plot from python program after scaling. The edges become more visible, larger because auf thebibliography new resolution.

3.3.5 Properties of Harris Corner

The Harris Corner is a region, where a gradient goes in diferent directions. But these regions are difficult to differentiate from edges with a high gradient in only one direction. It is invariant to scaling (as seen above) because the structure matrix can be used for diagonal directions, better as from other gradients in x and y directions.

3.4 Conclusion

In this assigment, many different approches for filtering have been examined. It is interesting that all of those appliences have same origins, in our case, same methods that can be reused. Compared to the default implementation of numpy or cv2, or implementation is quite not optimized. During the development stage, compairsion with the library was made. In the last exercise even the implementation of a library was used. Such basic function, must be robust and performant. So it is important from a mathematical or computer science perspective to be as accurate as possible, regarding to the use in medical appliances.

References

[1] Alasdair McAndrew
A Computational Introduction to Digital Image Processing (2015). (English)

[2] OpenCV Python Tutorials
https://opencv-python-tutroals.readthedocs.io (2019). (English)