Homework 2

Thomas Buchegger
Introduction to Signal and Image Processing

April 10, 2019

Contents

1 Used Methods

1.1 Linear Filtering

Linear filtering is used to modify a picture. This includes enhancing or removing features. Filtering is a so called neighbourhood operation, which means that the value of a certain pixel is based on its neighbourhood pixels.
An example for a linear filter would be a box filter. It calculates the arithmetic means based on its neighbourhood pixels for the new pixel. This leads to a blurring effect.

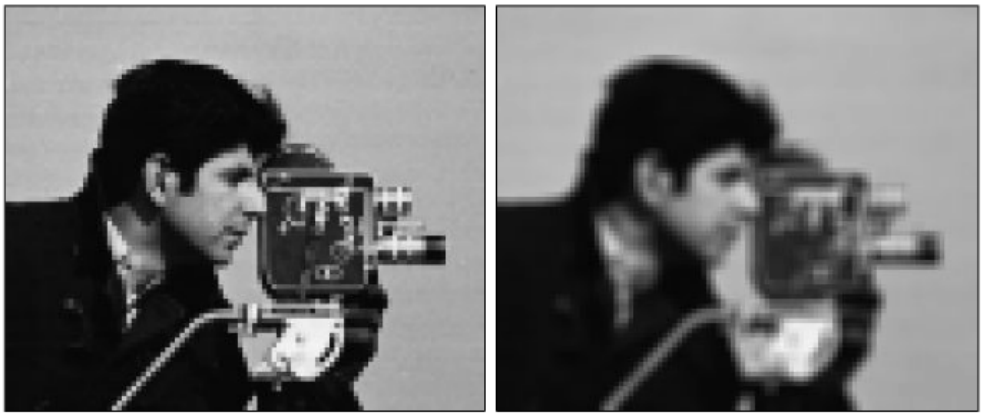

Figure 1: Filtering with a box-filter of size 3×3.

1.2 Finding edges

As the name suggests, the goal is to detect edges in images. One of the most popular edge detection algorithm is canny edge detection.
It consists of four steps:

1. Preprocessing with gauss for example

Figure 2: Before and after low pass filter

2. Gradient calculation

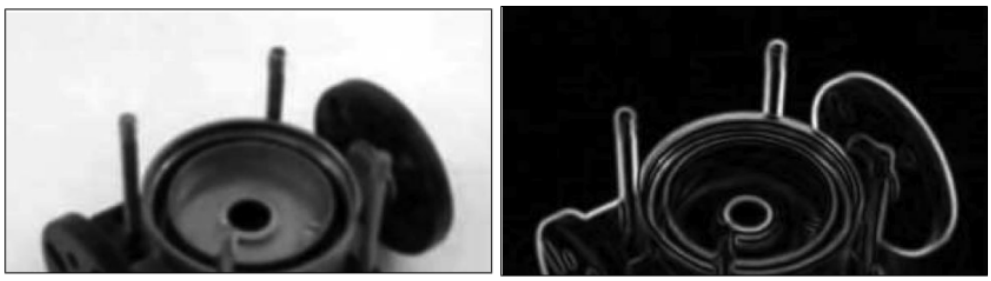

Figure 3: Before and after gradient calculation

3. Nonmax suppression

Figure 4: Before and after non max suppression

4. Hysteresis thresholding

Figure 5: Before and after thresholding

1.3 Corner detection

The goal is to detect corners in images. A corner is defined as an intersection between two edges, as seen in figure 6.

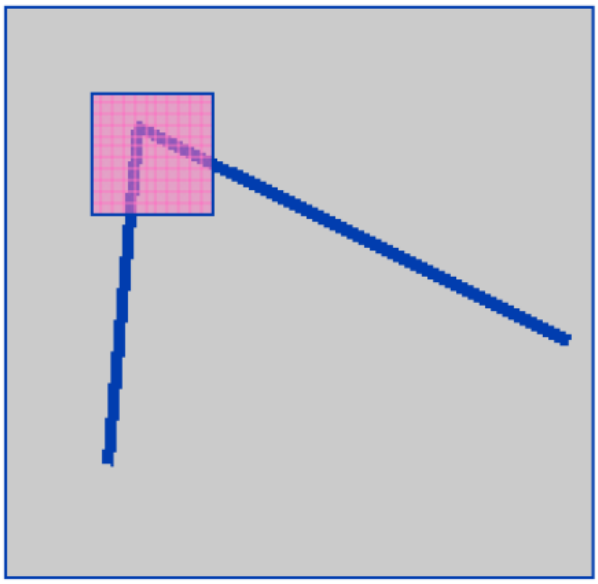

Figure 6: Corner as an intersection of two edges.

2 Exercises

2.1 Linear Filtering

2.1.1 Boxfilter

The function boxfilter(n) can be summarized as $np.ones((n, n))/(nxn)$.

2.1.2 myconv2

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

2.1.3 boxfilter with myconv2

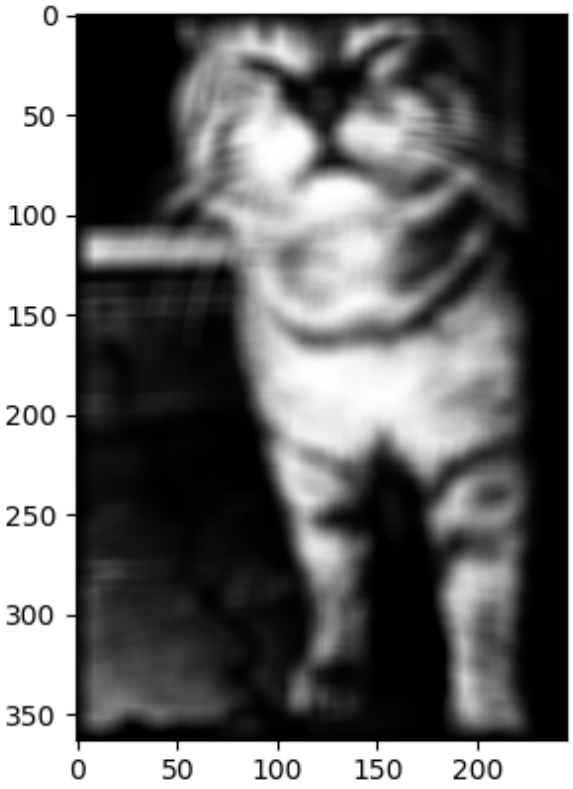

Figure 7: Applied box filter size 10

2.1.4 gauss1d

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.

2.1.5 gauss2d

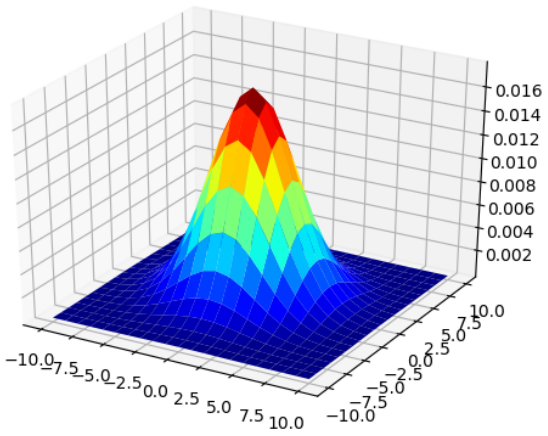

Figure 8: Gaussian kernel in 3D space

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

2.1.6 gconv

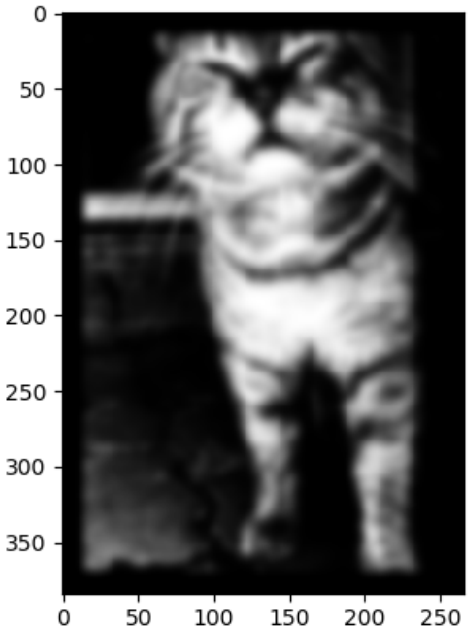

Figure 9: Result of the gaussian with sigma 3 on an image

2.1.7 More efficient convolution

The process of performing a convolution requires K^2 operations per pixel, where K is the size ($width == height == K$) of the convolution kernel.In many cases, this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring 2*K operations per pixel.If this is possible, then the convolution kernel is called separable! Look at the singular value decomposition (SVD) of the kernel, and if only one singular value is non-zero, then it is separable.

2.1.8 Computation time vs filter size experiment

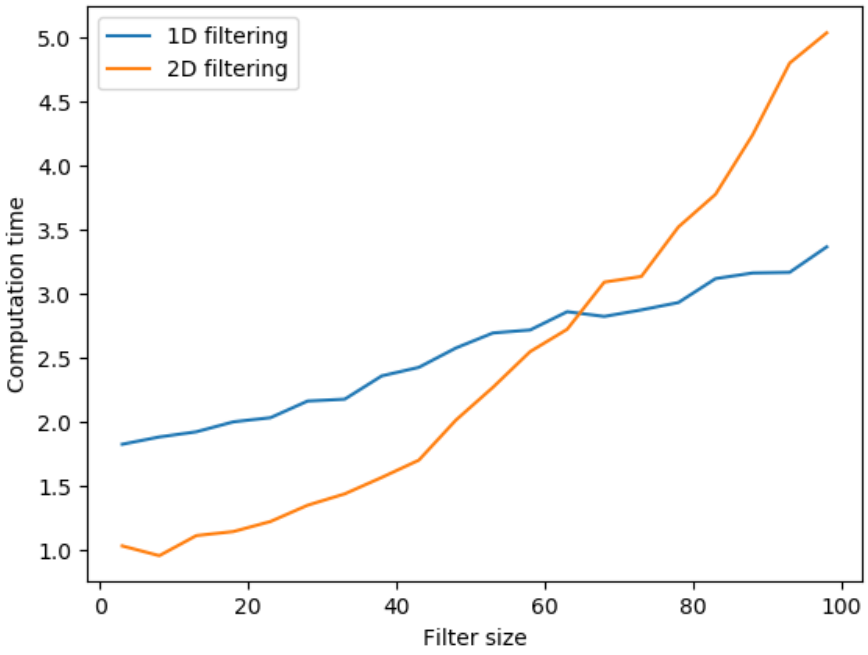

Figure 10: Compairson of 1D and 2D Gaus

The 2D filtering curve is much steeper, because of the factor 2 of the number of dimensions to be calculated.

2.2 Finding edges

2.2.1 gradients

Gradient detection is done with a high pass filter. In our excercise this is done with once in x and once in y direction with the help of the $myconv2$ and $gauss1d$ functions. The value for sigma is 1.

2.2.2 create edge magn image

We only want the strongest edges in all directions, therefore we first define $direction_ranges$ in all eight directions. After that we iterate through the picture eight times - once in each direction. We then get the strongest and weakest gray value and set only the strongest to 255, the rest to 0.

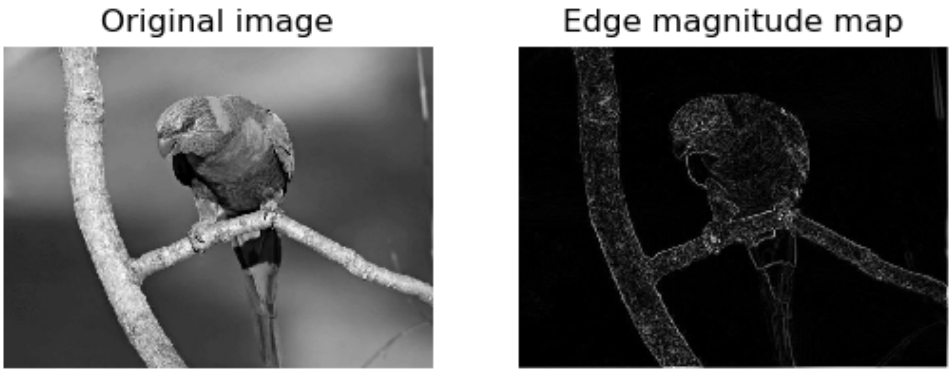

Figure 11: Original image and its magnitude map

2.2.3 make edge map

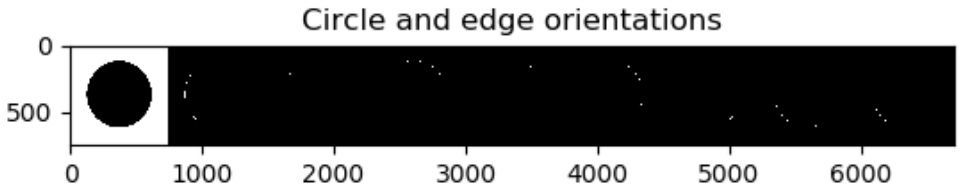

Figure 12: Edge orienation of a circle in all eight directions

2.2.4 edge non max suppression

The non max suppression is the last step of the canny edge detector algorithm. This step is needed, bceause edges are usually larger than one pixel, so they get reduced to the pixel with the max. gradient.

This is done by iterating through the picture four times, once in each direction. The currently looked at pixel is then compared to its neighbours. If it has the strongest gradient in [x,y], then it survives. Else it's set to 0.

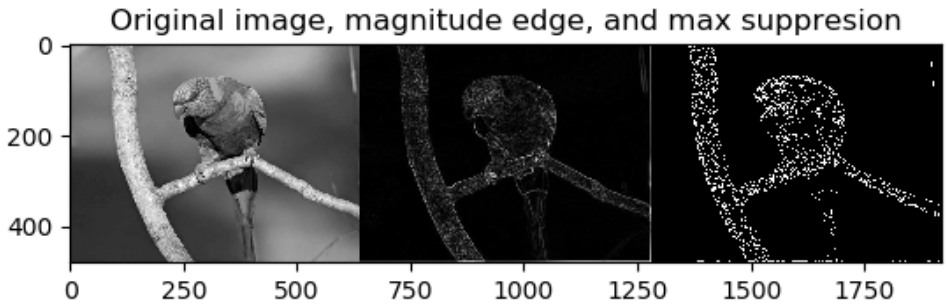

Figure 13: Comparison of original image, image with gradients and image with non max suppression

2.3 Corner detection

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

2.3.1 myharris

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

2.3.2 Evaluate myharris

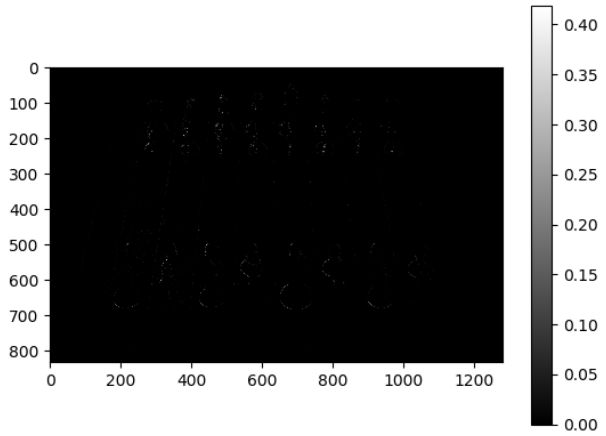

Figure 14: Output with low sigma of 0.2

2.3.3 Evaluate myharris rot 45°

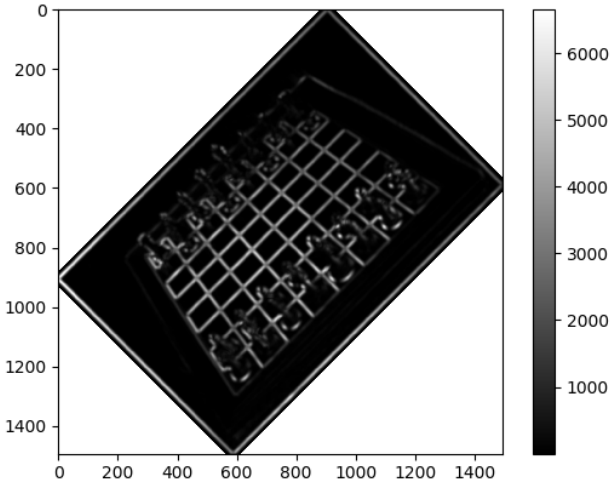

Figure 15: Output with simga of 6 and rotated 45°

2.3.4 Evaluate myharris downscaled

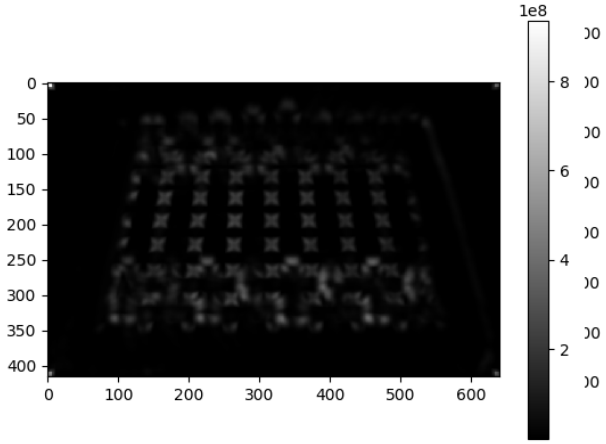

Figure 16: Output with simga of 6 and downscaled by 0.5

2.3.5 Harris Corner Properties

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 Conclusion

It was nice and a good training to implement all these algorithms by hand for once. Usually one would only use the from different frameworks provided algorithms without having the actual knowledge how they really work.