Report for the Course Bildverarbeitung 103-0274-01L FS22

Daniel Pfister

May 26, 2022

1 Fouriertransformation

1.1 Transformation

Figure 1 shows the input image on the left and the spectrum after applying `scipy.fft.fft2` and shifting it using `scipy.fft.fftshift`.

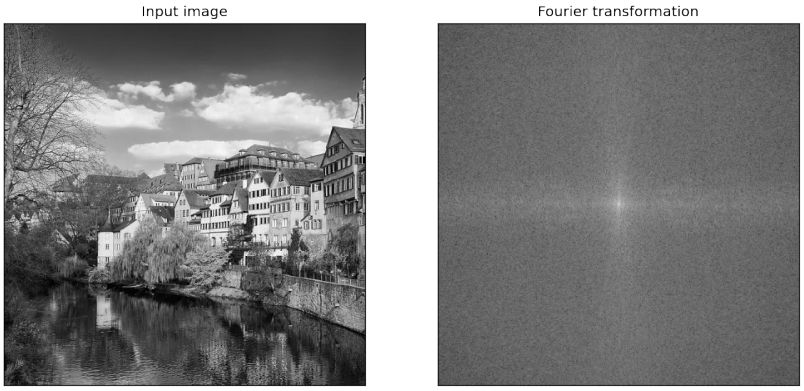

Figure 1: the input image shown in image and fourier space

1.2 High-pass filter

For this task a perfect high-pass filter was constructed and applied in Fourier space. Afterwards the spectrum was transformed back into image space using `scipy.fft.ifft2`

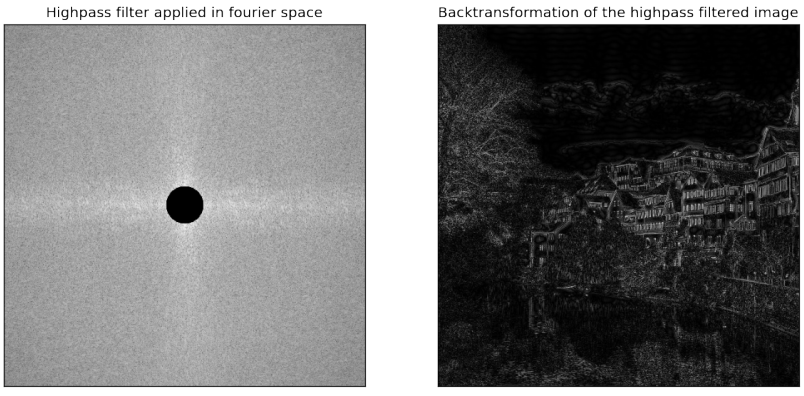

Figure 2: the image after applying a high-pass filter shown in fourier and image space

1.3 Gaussian filter

In this task a Gaussian filter was first constructed in the image space and applied to the image by applying a convolution. Afterwards the filter was transformed to the Fourier space using `scipy.fft.fft2` and applied to the spectrum of the image using multiplication. Additionally, the average numerical difference between the pixel values of the two results was calculated to be 0.073

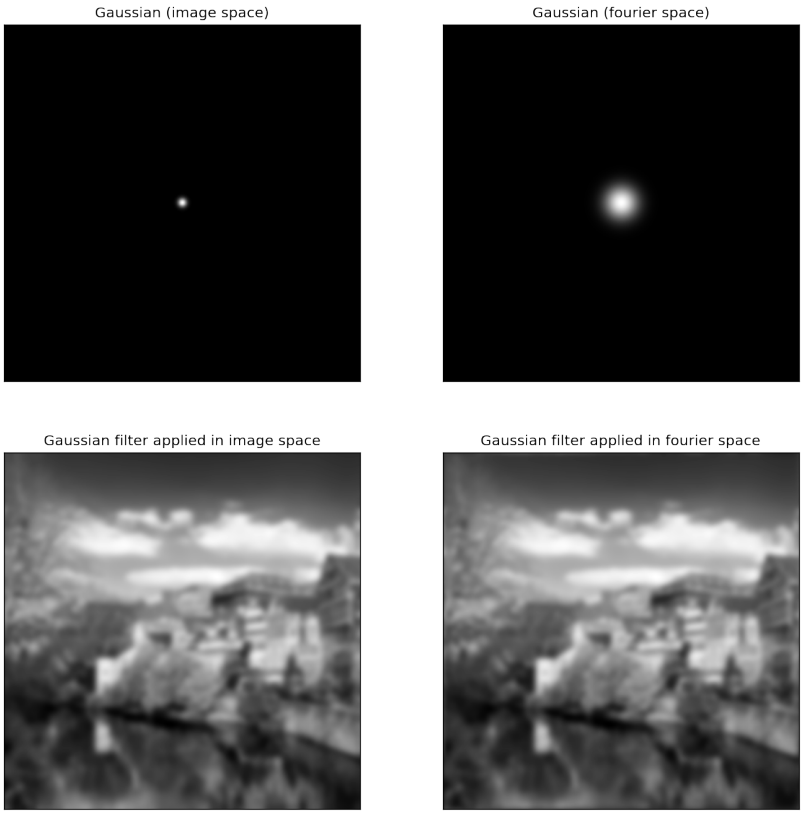

Figure 3: the Gaussian filter in image and fourier space and the images after applying the filter

1.3.6 Computational Difference

The Gaussian filter can either be applied in x and y direction simultaneously using a 2d kernel, or it can be applied seperately for each direction, using a 1d kernel each. The result stays the same. Here, the computational difference between the two methods was explored, by looping each method 20 times and calculating the mean time. The average computation time of the 2d application of the Gaussian filter was measured to be 653.27ms, while the average measured time using the seperate 1d application was 38.13ms.

2 Morphology

2.1 Coins

This is the original image of the coins:

Figure 4: original image of coins

2.1.1 Segmentation

The coins were seperated from the background by transforming the RGB image into HSV image space, where the saturation corresponds to how "colorful" a pixel is. A mask was created at the places where the saturation was in the range of $[0, 25]$. These were the background pixels and they were set to black to create the resulting image.

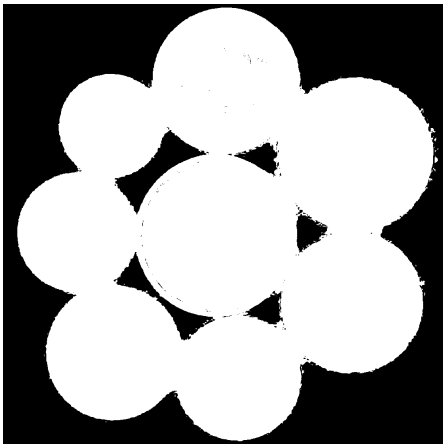

Figure 5: the segemented image of the coins

2.1.2 Closing Holes

The segmented image with the holes within the coins was given. To close the holes, the image was first dilated and then eroded.

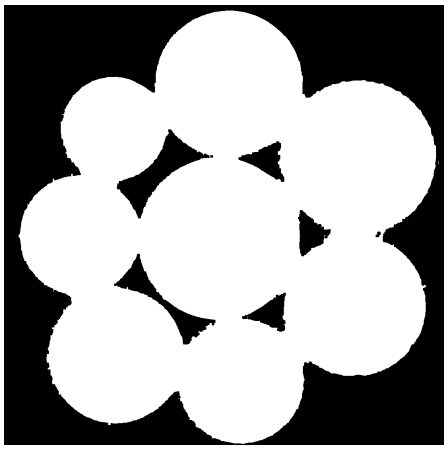

Figure 6: image of the coins after closing the holes

2.1.3 Instance Segmentation

For this task, first the result from the previous image was thresholded to create a binary image. Afterwards the distance transform was applied and the certain foreground and certain background were definied using dilation and erosion. Then, markers were placed on each certain foreground instance and the watershed algorithm was applied. In the last step, each new marker was colored with a random color.

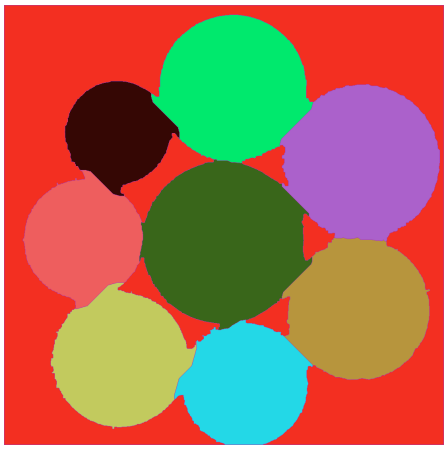

Figure 7: image of the coins after applying the watershed algorithm

2.2 Letter Recognition

Here, the image of the text and the image of letter were first transformed into a binary image, using `cv.THRESH_BINARY_INV` or `cv.THRESH_BINARY`, respectively. Afterwards the binary image of the text was eroded with the binary image of the letter as kernel to find the instances of the letter. Finally, the image was dilated back to have an image constructed of m's.

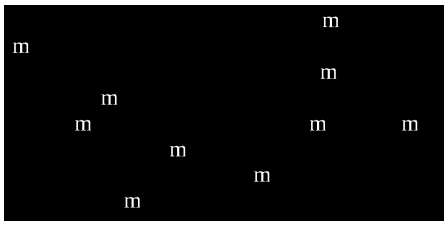

Figure 8: result of finding the letters in the image

3 Corner Detector

In the task the Shi-Tomasi and Harris corner detectors were implemented. For the Harris response, first the gradient along x and y was calculated using a Sobel kernel. Afterwards the structure tensor and the response were calculated and plotted.

Figure 9: Harris response of the input image

Then the response was thresholded and a red circle was placed at every valid pixel.

Figure 10: Harris corner detection

Finally, `scipy.ndimage.maximum_filter` was applied to get the local maxima of the corners.

Figure 11: Harris corner detection after NMS

For the rotated image, the results look the same, because the only difference is that the x and y dimensions were swapped. Below are the three results for the rotated image.

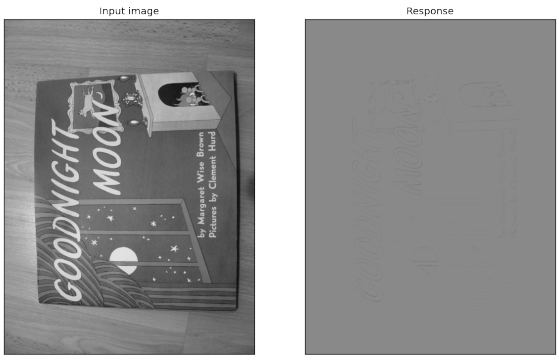

Figure 12: Harris response of the input image (rotated)

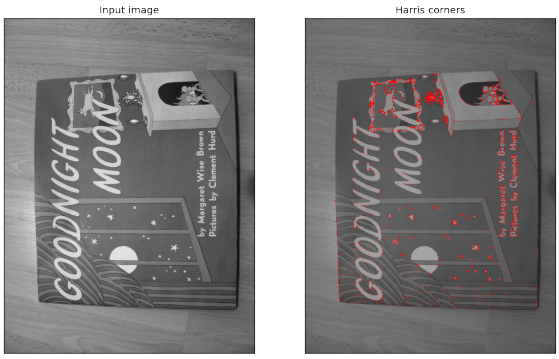

Figure 13: Harris corner detection (rotated)

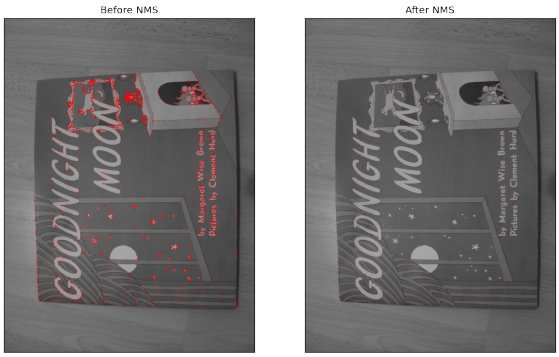

Figure 14: Harris corner detection after NMS (rotated)

4 Canny Edge Detection

Here, the image was first blurred by applying a Gaussian filter.

Figure 15: Image after applying Gaussian blur

Afterwards the gradient was calculated by convolving the image with a Sobel kernel. Using those gradients the magnitude and direction was found and plotted.

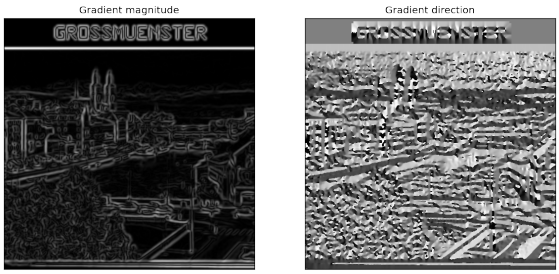

Figure 16: Magnitude and Direction of the blurred image

Finally, the NMS seen in the lecture was applied using code from Github and visualized.

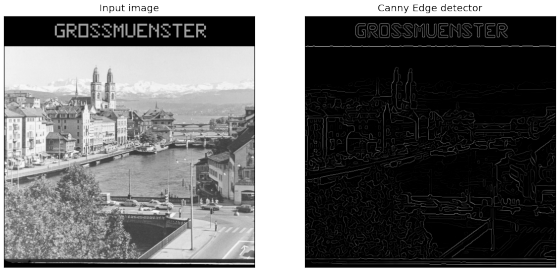

Figure 17: Canny Edge Detection after applying NMS

4.1 Final Words

The code and result images can be found on my personal Github.