Rico Mossinkoff - 12805157 *Hannah Lim - 10588973*
Ewoud Vermeij - 11348860 *Yke Rusticus - 11306386*

Computer Vision - Assignment 2

Introduction

In this assignment we will learn more about image processing on the basis of multiple different filters to analyze images. Before working with these filters, it is important to comprehend the basis of the operation of these filters beginning with neighborhood processing. Neighborhood processing is looking at the neighborhood pixels of a certain pixel to assign a new value to it on the hand of a function weighing those neighborhood pixels. Two filters using neighborhood processing are Gaussian and Gabor filters. These are low-level features and can be used for denoising an image or as a part for edge detection. To evaluate how good these filter has eliminated the noise the peak signal-to-noise ratio can be computed, which looks at the denoised image in comparison with the original (non-containing noise) image. Also, these filters can be used for edge detection by computing the first- and second-order derivatives of these. Last in this assignment we cover the topic texture-based image segmentation where we use Gabor filters and clustering to separate an object from the background.

2 Neighborhood Processing

Question 1

1. The kernel (h) for convolution is the correlation kernel rotated 180 degrees. The correlation operator starts in the top left of the kernel and iterates to the right and ends in the bottom right and the convolution operator starts in the bottom right and iterates to the left and ends in the top left.

2. The two operations have the same outcome if the kernel is symmetric around the axes.

3 Low-level Filters

Question 2

The result for the convolution with a 2D gaussian kernel and a 1D gaussian kernel in the x- and y- direction will be the same, because the 2D Gaussian function is separable, meaning that it could be computed by first performing a 1D gaussian in the x-direction followed by a 1D gaussian in a y-direction[1]. Therefore the result will be the same. The computational complexity of the for a 2D gaussian will be K^2 operations (multiply-add) per pixel with K being the size of the kernel. For a 1D gaussian the total of operations will be 2K per pixel. Once performed horizontally and once performed vertically[2].

Question 3

First or second order derivatives of, for example Gaussian filters, are known as band-pass filters, since they filter out both low and high frequencies. A very popular second derivative operator is Laplacian, which is an edge sharpening filter. It highlights regions of rapid intensity change and is very useful for finding small details in an image.

[1] Object Recognition from Local Scale-Invariant Features , David G Lowe
[2] p.115-116, Computer Vision: Algorithms and Applications

Question 4

λ: represents the wavelength of the sinusoidal factor. The wavelength determines the width of the stripes of the Gabor function. Increasing the wavelength results in thicker stripes and decreasing the wavelength in thinner stripes.

θ: represents the orientation of the normal to the parallel stripes of the Gabor function.

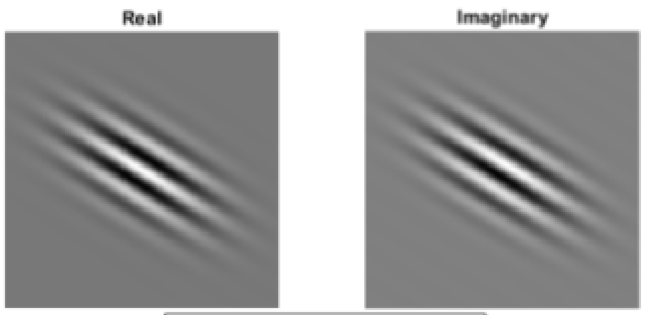
ψ: represents the phase offset sinusoidal function.

σ: represents the standard deviation of the Gaussian.

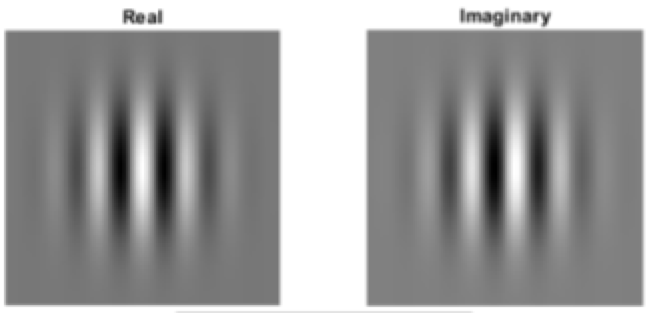
γ: represents the spatial aspect ratio and specifies the ellipticity of the support of the Gabor function.[3][4]

Question 5

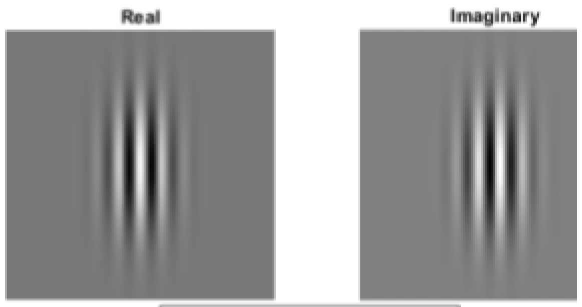
We set λ to 10 and ψ to 0. Setting λ to 10 gives a clear view on the stripes of the Gabor function. Larger values for λ will make the stripes in our visualizations more predominantly present. Adjusting ψ will give the stripes in our visualizations different colors of gray. Setting it to 0 for all images will make the different values for theta, sigma and gamma more clear. θ: Angle of the stripes, σ: number of stripes, γ: width of the stripes.



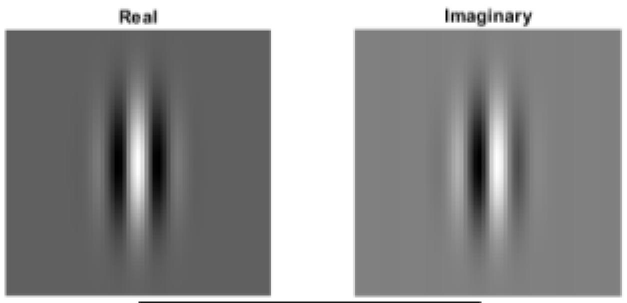
Theta: 45, Sigma: 10, Gamma: 0.5



Theta: 0, Sigma: 10, Gamma: 1



Theta: 0, Sigma: 10, gamma: 0.5



Theta: 0, Sigma: 5, Gamma: 0.5

[3] http://matlabserver.cs.rug.nl/edgedetectionweb/web/edgedetection_params.html
[4] https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97

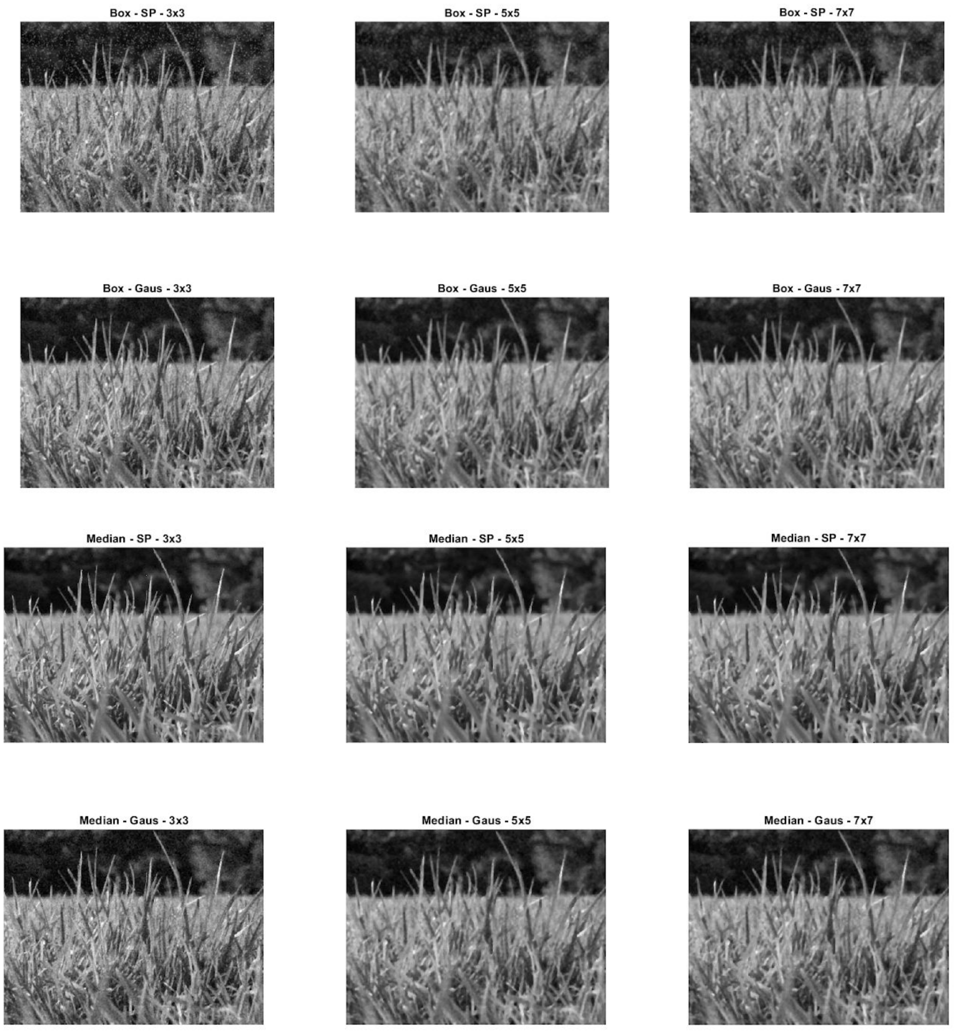
4 Applications in image processing

Question 6

1. PSNR metric is the ratio between the maximum possible power of a signal & the power of corrupting noise that affects the quality of its representation [5]. The PSNR metric is used to compare algorithms. When an algorithm tries to make a noisy image look like the original image, the PSNR measures how well this is done. A higher PSNR score indicates a better reconstructed image. PSNR uses the Mean Squared Error (MSE) in his function to determine the differences between the pixels of the old and reconstructed image. The lower the error, the higher the PSNR.
2. 16.1079 dB
3. 20.5835 dB

Question 7 - (*20pts*)

1.



[5] https://www.ni.com/nl-nl/innovations/white-papers/11/peak-signal-to-noise-ratio-as-an-image-quality-metric.html

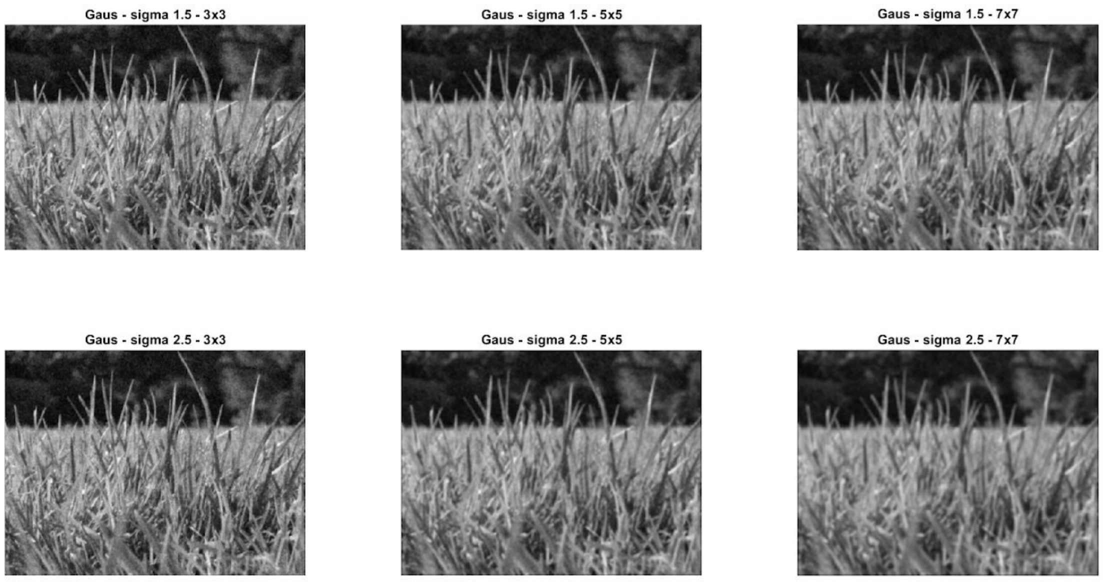
2. The table below shows PSNR values in dB for the different filters and filter sizes.

	Box 3x3	Box 5x5	Box 7x7	Median 3x3	Median 5x5	Median 7x7
Salt pepper	16.4394	15.7926	15.4254	16.1201	15.7417	15.4046
Gaussian	20.3716	19.1919	18.4129	20.2763	19.2558	18.4619

   We can see in the results that a larger filter size does slightly decrease the PSNR score for both box and median filtering in both images. The larger filter size causes the image to be smoother compared to the image filtered with a smaller kernel size, because more pixels will influence the final result in the target pixel. A smoother image will have less similarities with the original image, therefore for larger kernel sizes, the MSE increases. This causes the PSNR to be slightly lower for larger kernel sizes.

3. For the salt-and-pepper noise the median filter seems to filter out the noise best if we look at the images themselves. Even though the PSNR scores of the box filter is slightly higher, the median filter is working better than the box filter since the ‘hot’ pixels will have no influence in the median filter.
   For the gaussian noise the difference between the box and median filter is neglectable. This has to do with how the noise in the image is modelled. The noise is gaussian, so averaging out the noise can give a very similar result compared to the median.

4.



   In the picture above we see six different settings of the gaussian filter on an image with gaussian noise. It is clear that a kernel size of 3x3 has less effect on the image compared to the 5x5 and 7x7 kernel sizes, while the 7x7 kernel size makes the image slightly too blurry. A kernel of size 5x5 with sigma 1.5 seems to be the sweet spot taken only the visuals in the above picture into account. Also, if we take a kernel size larger than twice the size of the sigma, we will prevent the Gaussian filter to become a box filter. Which again is prone to add new noise tot the image while filtering.

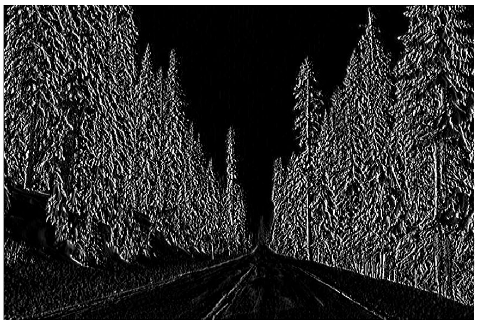
5. To get a more distinctive view on the effect of standard deviation we take a kernel of size 7x7.

Sigma	1	2	3	4	5
PSNR score (dB)	20.7603	19.0526	18.6315	18.4784	18.4067

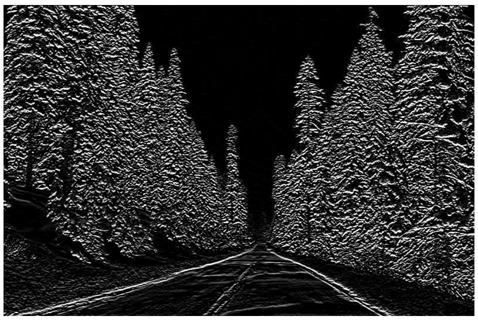
As shown in the table, higher values for sigma will lower the PSNR score. Higher standard deviation will give more influence to pixels representing noise in the neighbourhood of target pixels compared to kernels with lower standard deviation. This causes a higher MSE and thus a lower PSNR score. Also, higher values for sigma will transform the Gaussian into box filtering, which again can add extra noise to the filtered image.

6. Median filtering takes the median of the neighbouring pixels and is working best for filtering hot pixels out, since they are not likely to have influence on the median as a single pixel.
Box filtering simply takes all values in the kernel and give the target pixel the average of all those pixels. This gives all pixels in the kernel the same amount of influence on the target pixel.
Gaussian 2D filtering is similar to box filtering, except that the closer a pixel is to the target pixel, the more influence it will have on the target pixel. This is of course dependent on the standard deviation. Higher values of standard deviation will give more pixels in the area larger influence.
Two filtering methods having similar PSNR scores can have a qualitative difference. This is clearly shown by applying a median filter and a box filter on the salt-and-pepper image. Both are having a similar PSNR for different kernel sizes. However, the median kernel does a better job filtering out the 'hot pixels' from the image.

Question 8



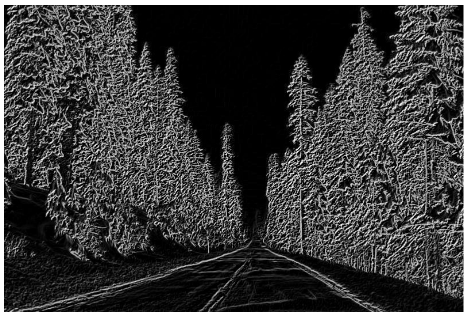
1. The gradient of the image in the x-direction. This shows the horizontal difference in brightness, so the vertical edges (and lines) are detected.



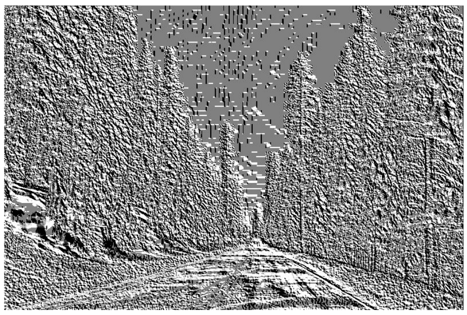
2. The gradient of the image in the y-direction. This shows the vertical difference in brightness, so the horizontal edges (and lines) are detected.

3. The gradient magnitude of each pixel. This shows the difference in brightness in both horizontal and vertical direction by merging the gradient in the x direction with the gradient in the y direction.

   Normalised magnitude of each pixel (i.e. the maximum value is 255)

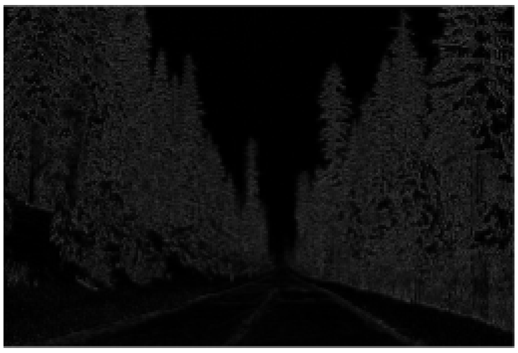


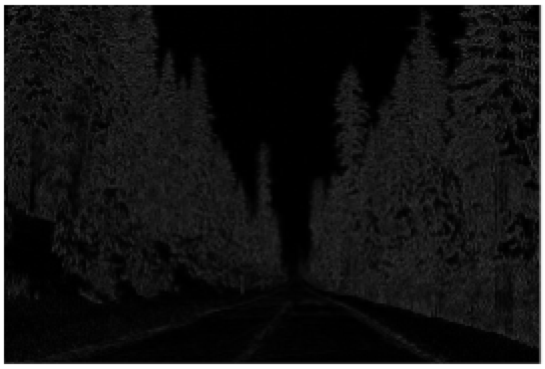
4. The gradient direction of each pixel. This image shows the direction of the gradient of each pixel pointing from dark to white or in between. Black means the gradient is pointing in the x-direction, white means the gradient is pointing in y-direction and gray means the direction is somewhere in between the x- and y-direction.

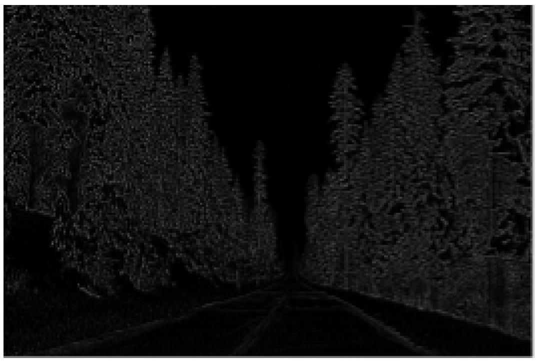


Question 9

1.


Method 1


Method 2

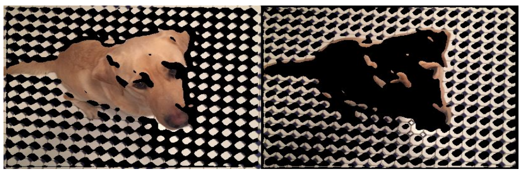

Method 3 (Kernel 1: 5x5, sigma=1, Kernel 2: 5x5, sigma=0.91)

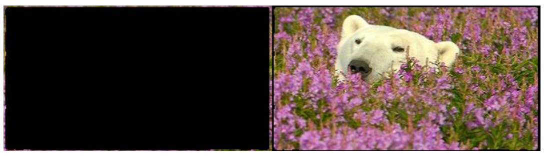
2. The first method first does a convolution with the Gaussian kernel on the image and after that does a convolution with the Laplacian. The second method does a convolution of the Gaussian with the Laplacian and convolves the image with this new LoG kernel (AxA). The second method is therefore less costly efficient, because it has to do A^2 operations, where as the first method does $a1^2 + a2^2$ operations. With $A = a1 + a2 - 1$ and $a1$ = kernel size Gaussian1, $a2$ = kernel size Gaussian 2As convolution is commutative, the order does not matter. So the first and the second method should return the same result. In method 3 the algorithm looks at two images smoothened with two Gaussian filters

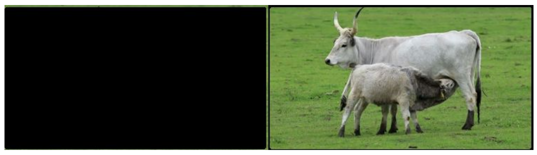
with different standard deviations. The smaller the ratio between the standard deviations, the better the approximation is.

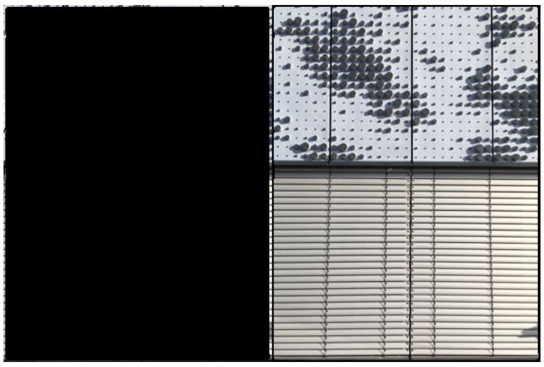
3. A Gaussian filters out high frequencies, which are often considered to be small or noisy features in an image. When taking second order derivatives with the Laplacian, we are generally not interested in the second order derivatives of or around these noisy features, but we are rather interested in changes at the boundaries of larger structures. So applying a Gaussian filter before taking the laplacian makes sure we filter out noise before we look for edges in our image. In addition second order derivatives are very sensitive to noise, so before using the LoG we want to filter out the noise.
4. By the method of trial and error we found that a ratio of around 1.6 gives a result similar to the LoG (sigma1 = 1, sigma2 = 0.625). One Gaussian is used to smooth out edges in the image, which we want to detect by comparing the smoothed image with the original. However, since this Gaussian also removes high frequencies from the image, which we do not want to detect, so we need to denoise the original image before comparison. For this reason we need another Gaussian to denoise the original image, with a smaller standard deviation than the first Gaussian.
5. Line recognition is needed to determine which edges belong to the road, and which don't. Then the other edges in the image can be filtered. To accomplish this, find lines that are (nearly) not interrupted and use vanishing points to determine if it is a road.[6] Lines can be seen as neighbour pixels with roughly the same value, going into one direction.

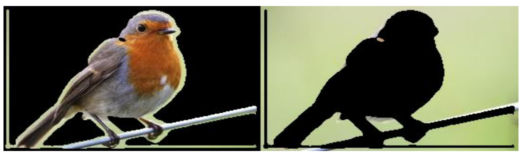
Question 10

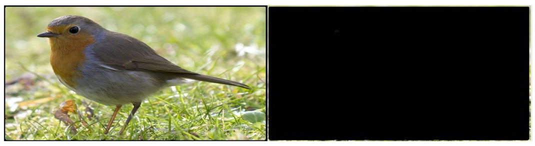












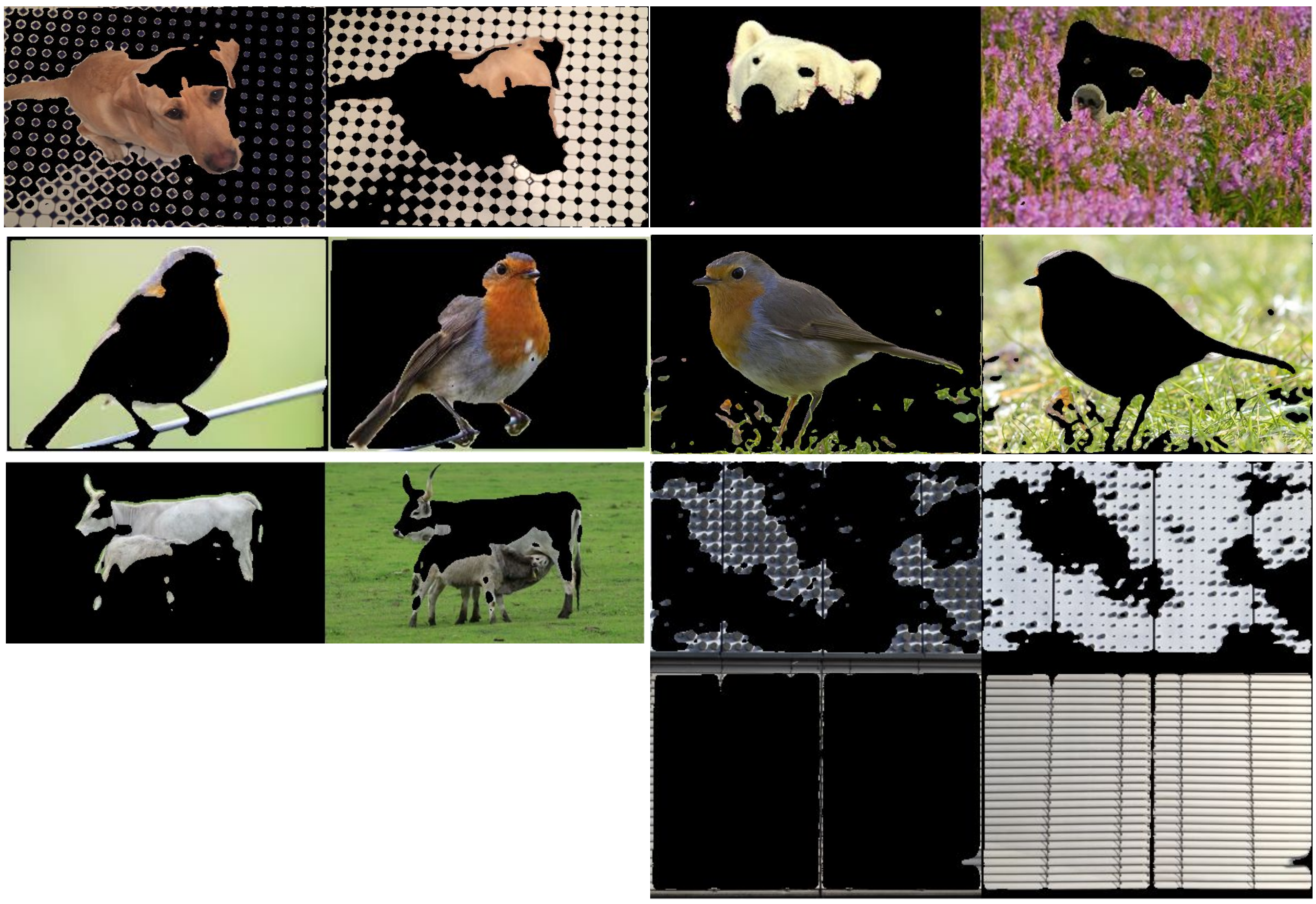
1. The provided parameter settings work better for some images than for other, although for none of the images it works the way we hoped for. For 'Kobi' mostly the floor is cut out of the image, and some of the edges of the dog. For 'Robin-1'' the algorithm works better with the provided parameters, although the twig on which the

[6] https://hal-enpc.archives-ouvertes.fr/hal-00834892/file/CVPR09c.pdf

Rico Mossinkoff - 12805157
Ewoud Vermeij - 11348860
Hannah Lim - 10588973
Yke Rusticus - 11306386

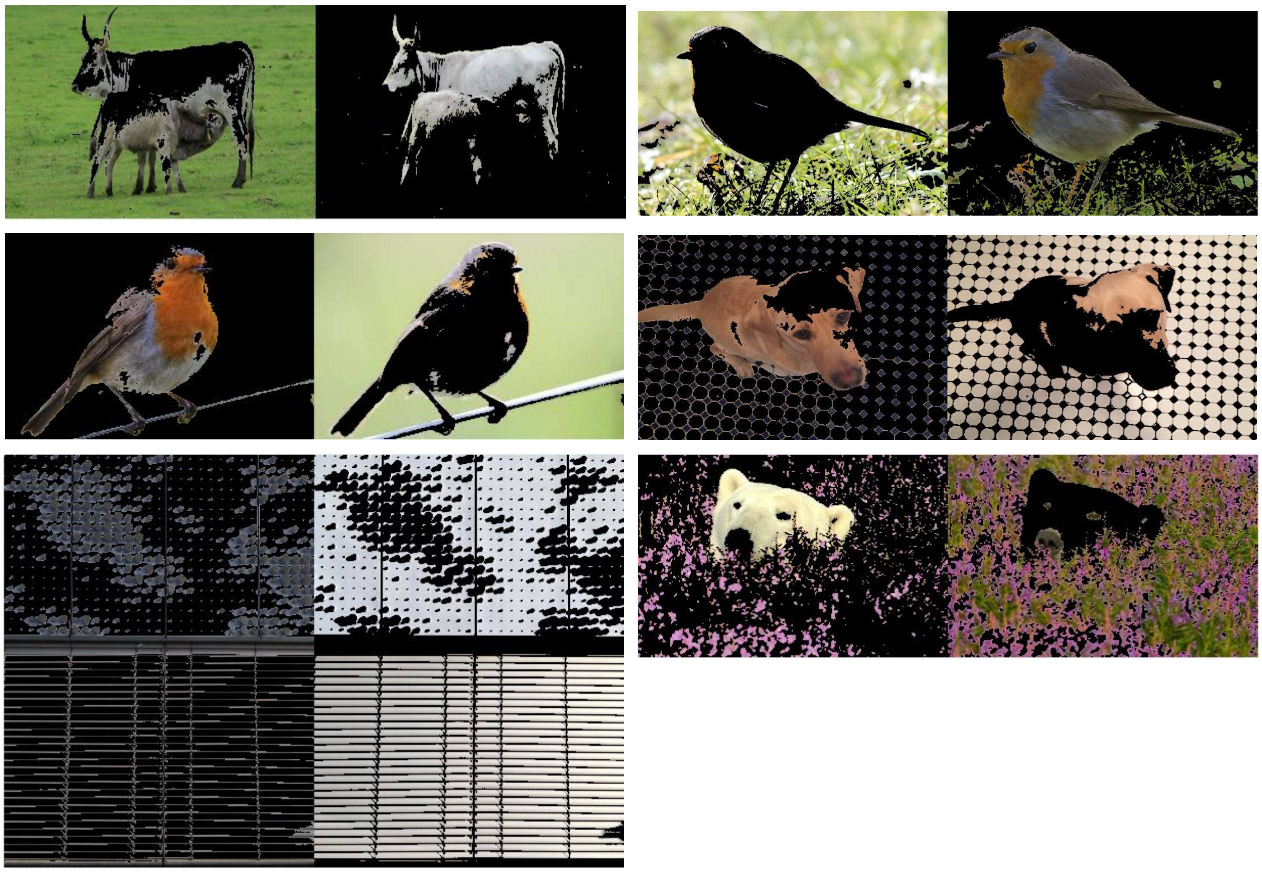
birds stands is also separated from the background. For four of the six images, the resulting selection is a (mostly) black image, which is not at all what we want.

2. We have experimented with different sets of parameters for each of the images. The animals in the images all have different sizes and appearances, so we would expect different parameter sets to work best for each individual image. However, we have found a single parameter set that works best for every image. The results are shown below. The parameter set is the following: the unchanged values for lambda and theta, and for sigma we chose the single value 0.2. The reason why this value for sigma works so well is because it is relatively small, suited for separating the (often more) blurry background from a more detailed foreground. In the case of SciencePark, a single value for theta of pi/2 could be considered to separate the top half of the image from the lower half, since the lower half is characterized by its horizontal lines. In that case a larger value for sigma is preferred. However, we chose to separate the dots on the wall from the rest of the image, which works well for sigma = 0.2. In all cases, the standard deviation of the smoothing kernel was 3.



3. Without smoothing the magnitude images, the selected patches in the images have more edges, i.e. the patches become more detailed. In general more detailed parts of the objects (or animals) are separated. This step could be useful to further tune your

parameters in a more detailed manner to find the optimal set of values to select the preferred object.



Conclusion

The goal of this assignment was to get acquainted with the fundamentals of image processing. We learned that different image filters each perform best at different filtering goals. Median filters perform best at filtering out 'hot' pixels in an image, and Gaussian filters are great for smoothing the image. The performance of these filters can be measured with the Peak Signal to Noise Ratio (PSNR), which uses the Mean Squared Error (MSE). However, the PSNR does not necessarily indicate a good quality image for the human eye. The PSNR score for the Gaussian filter performed on the salt and pepper image is almost equal to the PSNR score of the median filter performed on the salt and pepper image. However, for the human eye, it is clear that the median filter performs a better job at filtering the 'hot' pixels out of the image.
We have seen that determining the gradient of an image could be a useful tool to detect edges or lines. This is important for applications such as road detection because there are often clear distinguishable lines on each side of the road. However, there are again several methods which could be used to do so, and each of the methods give a slightly different result. We tested several methods in this work and we compared their results. It has become clear that any edge could be detected by simple Sobel filters, but in order to detect sharp, fast changing edges, the preferred method is to take second derivatives of the image. In the last part of this work, we have seen that multiple Gabor filters could be used to separate objects from their background. However, in order to so, a well suited set of parameters must be found depending on the given input image.