Comuputer Vision Lab 2

Maximilian Schlögel UvA
Anca-Diana Vicol VU
Benjamin Kolb UvA
Philipp Lintl UvA

1 Introduction

In this report we will talk about basic applications of neighborhood processing for images. We will start off by discussing operators of linear filters, namely correlation and convolution, where we also discuss their differences. In the next part we explore two well known Gaussian filters and Gabor filter. Especially in the case of the Gabor filters we have many different parameters influencing the final shape and form of the filter. To better understand these, we will visualize changes of three of the five used parameters. After that we will talk about different ways to deal with noise in images. To evaluate those enhancement techniques, we will develop a metric to measure the performance. Using this metric we will compare the before mentioned denoising methods. In the following section we will have a look at edge detection in images and how we can use first-order and second-order derivatives to implement these. Here we focus on the application of Laplacian of Gaussian method. Finally we will apply Gabor filters to implement an image segmentation algorithm and experiment with different parameter settings to achieve good results.

2 Neighborhood Processing

2.1 Question-1

Correlation and convolution are almost identical operators, the main difference is in the direction in which the filter is applied. This results in convolution practically flipping the kernel (h) over before applying it. The significance of their difference is in the fact that convolution is associative while correlation is not always. The associativity is a useful property as multiple kernels can first be convoluted together and then the resulting kernel applied by convolution to the image, simplifying the overall computation. When the kernel is symmetric the two operations become equivalent. [1]

3 Low-level filters

3.1 Gaussian Filters

3.1.1 Question-2

While applying a 2D Gaussian Kernel smooths the image in both directions simultaneously, the consecutive execution of 1D Gaussian filters in x-direction and in y-direction smooths the image first dimension and then smooths the resulting image in the other. We found by computational experimentation that the two procedures do in fact yield the same resulting image.

To show this mathematically let the 1D Gaussian filters in x-direction and in y-direction be defined by

$$\mathbf{G}_\sigma^{\mathbf{x}})(i,j) := \sum_k \mathbf{I}(i+k,j)\mathbf{G}_\sigma(k)\mathbf{G}_\sigma^{\mathbf{y}})(i,j) := \sum_l \mathbf{I}(i,j+l)\mathbf{G}_\sigma(l)$$

Preprint. Work in progress.

for all i, j. Then we have for all i, j:

$$\begin{aligned}((\mathbf{I} \otimes \mathbf{G}_\sigma^{\mathbf{x}}) \otimes \mathbf{G}_\sigma^{\mathbf{y}})(i,j) &= \sum_l (\mathbf{I} \otimes \mathbf{G}_\sigma^{\mathbf{x}})(i, j+l)\mathbf{G}_\sigma(l) \\ &= \sum_l \sum_k \mathbf{I}(i+k, j+l)\mathbf{G}_\sigma(k)\mathbf{G}_\sigma(l) \\ &= \sum_{k,l} \mathbf{I}(i+k, j+l)\mathbf{G}_\sigma(k,l) \\ &= (\mathbf{I} \otimes \mathbf{G}_\sigma))(i,j)\end{aligned}$$

so applying the 1D Gaussian filters in x-direction and in y-direction consecutively is in fact equivalent to applying the 2D Gaussian filter. There is a difference in computational complexity, though. For an image of size $m \times m$ and a kernel size of n in both directions applying the 1D filter twice results in a complexity of $O(2m^2n)$, while the 2D Gaussian filter will require $O(m^2n^2)$.

3.1.2 Question-3

Why would it be interesting to design a second order kernel? As we can see in the figure **??** the filter obtained by taking the second derivative of the 2D Gauss-function, is very sensitive to extreme change in value of the pixels. According to [2] this can be used for either edge detection, stereo matching or image enhancement. You can have a first order derivative with a threshold or a second

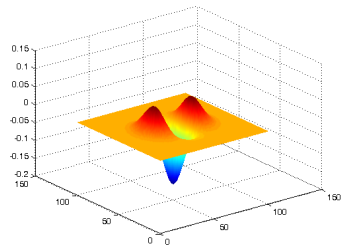

Figure 1: Second-order derivative of the 2D-Gauss function

order derivative with zero-crossing for edge detection.

3.2 Gabor filters

In this section we will discuss the Gabor filters and present the responsibilities of the different parameters that define the filter. Gabor filters are mainly used for texture analysis. Often texture of objects can be represented by a recurrent pattern, that is what makes filters with an inate frequency structure, like the Gabor filters, suitable for such a task. The function for a Gabor filter can be divided into two parts. The first one is the 2-dimensional Gaussian kernel function which determines the size and form of the area of influence of the kernel. Basically a larger spread, given by σ- and γ-values lead to a larger area of influence. The second part is the sinusodial carrier signal. This part is responsible for the frequency structure of the filter. Another important aspect is that the 2-dimensional Gabor function has a real and a imaginary part, where the real part is representing the cosine-modulated and the complex part the sine-modulated part. To discuss the parameter later we will only look at the cosine-modulated part. The 2-dimensional Gabor function is:

$$g_{real}(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \cos(2\pi \frac{x'}{\lambda} + \psi) \tag{1}$$

$$g_{im}(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \sin(2\pi \frac{x'}{\lambda} + \psi) \tag{2}$$

Note that the $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$.

3.2.1 Question-4

Here we will discuss each parameter of the gabor filter and what its respective purpose is. As we have seen above the gabor filter is the multiplication of two function. We will start off by discussing the parameters for the 2-dimensional gaussian function.

- σ
  As in every Gaussian function, the σ-value represents the spread of the function. In the case of the Gabor filter, this parameter decides, how much the influence of pixels decreases with increasing distance to the kernel centre. A larger σ means that pixel further away have larger influence on the value after applying the filter.
- γ
  The γ-value defines the spread along the different (rotated) axis. This may be more obvious when rewriting the term $\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}$:
  $$-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2} = -\frac{x'^2}{2\sigma^2} - \frac{y'^2}{2(\frac{\sigma}{\gamma})^2}$$
  We clearly see now that the γ-value influences only the spread along the rotated y-axis. A larger γ means that we have less variance along this axis. It is noteworthy that this parameter influences rather the ratio between the variances of both axis and not the actual variance, which is done by the σ parameter discussed above.
- θ
  This parameter is an implicit parameter, as it cannot be seen in the actual formula for the Gabor filter. As stated above, θ is used for calculating x' and y' ($x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$). We now see that θ changes the orientation of the coordinates and therefore decides the orientation of the texture we are interested in.

Next we will look at the parameters of the sinusoidal carrier signal:

- λ
  The λ-value in the carrier signal influences the periodicity of the sine and cosine wave, and thus also the frequency of the pattern that is going to match this kernel.
- ϕ
  This parameter describes the phase offset of the sine and cosine wave.

3.2.2 Question-5

After explaining what the different parameters in the Gabor-filter stand for, we will visualize this for the parameters θ, σ and γ. Since in our implementation the kernel size is influenced by the chosen σ, we chose a larger base $\sigma = 5$. By increasing the kernel size, we will be able to see the attributes of the parameter enlarged and more in detail. Additionally we chose $\lambda = 3$ to increase the wave frequency of the carrier signal. This helps us visualizing the influence of the single parameters even better.(The base parameters are chosen as follows: $\sigma = 5$, $\theta = 0$, $\lambda = 3$, $\psi = 0$, $\gamma = 1$). We will only look at parameter changes for the cosine-modulated part of the Gabor filters, as it will suffice to portray the respective changes.

The influence of the θ-parameter:

First we will look at the influence of the θ-parameter. As discussed above we assume that it will influence the orientation of the sine and cosines. We start off with the base parameters and change the rotation by $\pi/8$ twice. It is clearly visible in figures 2, 3, and 4 that changing the θ-parameter also changes the rotation of the filter.

The influence of the σ-parameter:

Now we will look at the influence of the σ-parameter. Above we stated that in theory it must be responsible for the range of the filter. The larger the σ, the larger the area of influence of the neighboring pixel. For the images of figures 5, 6, and 7, we have an increase of σ from 3 to 5 to 7. This change also reflects in the images as expected. An increase of σ also means an increase of

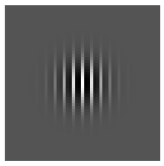

Figure 2: Gabor filter with no rotation

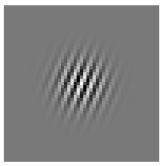

Figure 3: Gabor filter with a rotation by $\pi/8$

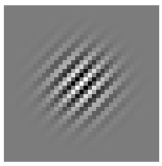

Figure 4: Gabor filter with a rotation by $\pi/4$

Figure 5: Gabor filter with $\sigma = 3$

Figure 6: Gabor filter with $\sigma = 5$

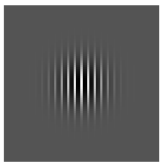

Figure 7: Gabor filter with $\sigma = 7$

the area of influence. Note: As the image size is the same, the increase of the area is visible in the increase of pixels, which also comes with an increase in finer granularity.

The Influence of the γ-parameter:

The last parameter we are going to experiment with is the γ-parameter. We saw in the earlier section that this influences the ratio of spread between the x-, and y-axis. We set θ to 0, to make this clear. In figures 8, 9, and 10. Remember that a smaller γ leads to a higher spread along the y-axis, since we divide σ by γ to get the new standard deviation. In figure 8 we set the σ-parameter to 2.5, so that the relative size of the filter is the same, since the size depends on the maximum-value of x-variance and y-variance. In the other ones we have a maximum variance of 5, and therefore we had to set σ in such a way that $\frac{\sigma}{\gamma} = 5$.

Figure 8: Gabor filter with $\gamma = 0.5$ and $\sigma = 2.5$

Figure 9: Gabor filter with $\gamma = 1$

Figure 10: Gabor filter with $\gamma = 2$

4 Applications in image processing

4.1 Image denoising

4.1.1 Quantitative evaluation

Our `myPSNR` function yielded a peak-signal-to-noise-ratio (PSNR) of 16.1079 between 'image1' and 'image1 saltpepper' and 20.5835 between 'image1' and 'image1 gaussian'.

4.1.2 Neighborhood processing for image denoising

Figures 11 and 15 display the original image with salt pepper noise. Figures 12-14 and 16-18 show the denoised images according to both box and median filtering and for three different kernel sizes.
Salt and Pepper Noise


Figure 11: Original `saltpepper`


Figure 12: box, kernel size 3x3.


Figure 13: box, kernel size 5x5.


Figure 14: box, kernel size 7x7.


Figure 15: Original `saltpepper`


Figure 16: median, kernel size 3x3.


Figure 17: median, kernel size 5x5.


Figure 18: median, kernel size 7x7.

Figure 19-26 correspond to the equivalent images where the original image has a gaussian noise instead if a salt-pepper noise.


Figure 19: Original `gaussian`


Figure 20: box, kernel size 3x3.


Figure 21: box, kernel size 5x5.


Figure 22: box, kernel size 7x7.

As seen in Table 1, for both filters, as well as both images, a larger kernelsize leads to a lower PSNR value. This is caused by the fact that filtering smooths an image, whereas a larger kernel considers more neighboring pixels and thus causes more smoothing. So, larger kernels and thus smoother images imply a smaller signal to noise ratio, as the MSE in the denominator of the formula grows. Considering our evaluation measure, it is thus recommendable to use a smaller kernelsize like 3 x 3. The salt and pepper noise image is characterized by individual pixels that are either of the maximum or minimum intensity value due to corruption caused by a sensor error. Accordingly, white and black spots appear as seen in Figure 11. Considering that the salt and pepper noise image contains a lot of extreme values, the median filter is to be favored. This is the case, as rank ordered filters do not consider the value itself and thus are more robust towards extreme values. The PSNR value corresponds to this notion, as even the largest kernel size in median filters is similar to the smallest in box filters. Looking at the actual Median filtered images, such as Figure 16, reveals that they seem less corrupted than its counter part in Figure 12.
The Gaussian noise image on the other hand is characterized by an added Gaussian noise term, which is why less extreme values than in the salt pepper noise images appear. The box filter yields a higher PSNR value, but rapidly diminishes the quality of the image due to the blurring effect that is introduced. According to the PSNR one would choose the box filter with kernelsize 3.

We justify our choice with Table 2 and the following denoised Figures 27- 41. Choosing by the highest PSNR value leaves us with std of 1 and kernelsize of 3. The images support this decision,


Figure 23: Original `gaussian`


Figure 24: median, kernel size 3x3.


Figure 25: median, kernel size 5x5.


Figure 26: median, kernel size 7x7.

Filter	Box Filter			Median Filter		
Kernel Size	3×3	5×5	7×7	3×3	5×5	7×7
PSNR on $saltpepper.png$	23.3941	22.641	21.422	27.6875	24.4957	22.3722
PSNR on $gaussian.png$	26.2326	23.6610	21.9441	25.4567	23.7983	22.0765

Table 1: PSNR for different filter, kernel sizes on both noisy images: `saltpepper` and `gaussian`.

as mentioned before, higher kernel values result in blurrier images. The same holds for higher std values. That is why the blurriest image is the one with the highest std and kernel size.

As seen in Table 2, a higher standard deviation (std) of the Gaussian filter leads to smaller PSNR values. This stems from the fact that a higher std allows a stronger weighting of not only the nearest neighbors and thus regulates the degree of smoothness. Especially for larger kernelsizes, more and further neighbors are considered leading to a larger decrease in PSNR. The smallest kernelsize is not remarkably influenced by the std, as there are less neighbors and thus more similar weights.

	Standard deviation				
Kernel size	1	2	3	4	5
3	26.602	26.147	26.04	26	25.982
5	26.16	24.215	23.746	23.58	23.504
7	26.098	23.213	22.36	22.054	21.913

Table 2: Influence of standard deviation on the PSNR value of the noisy `gaussian` image.

One main difference is robustness against extreme intensity values which are especially present with salt-pepper noise. Median filtering is rank based and thus does not consider extreme values as highly as the box filter, which averages over the neighboring pixels. The box and Gaussian filter differ in the way that the former weighs all neighbors equally, whereas the Gaussian filter weighs according to the chosen std. So, the Gaussian allows to control the obtained blurriness, whereas the Box filter is an approximation to that and always yields the same image given one kernelsize.

4.2 Edge detection

For finding the edges of an image, we use methods of approximating the first and second derivatives of the image. At the location of the edges, the image brightness changes sharply, thus the first order derivatives will exhibit higher absolute values and the second order derivatives zero-crossing points.

4.2.1 First-order derivative filters - Question-8

Figures 42 and 43 show the derivatives on the x and y axis of the image. The derivative on the x axis expresses the changes in the brightness of the picture from left to right, while the one on the y axes expresses the changes from top to bottom.

Using these results, we calculated the gradient magnitude and gradient direction by the following formulas and the results can be observed in figures 44 and 45.

$$G = \sqrt{(G_x^2 + G_y^2)}$$
$$\theta = tan^{-1}\frac{G_y}{G_x}$$


Figure 27: std = 1, kernel = 3

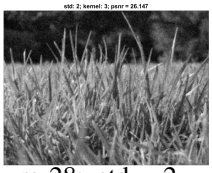
Figure 28: std = 2, kernel = 3

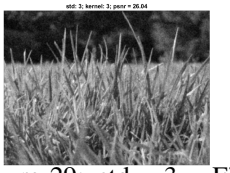
Figure 29: std = 3, kernel = 3

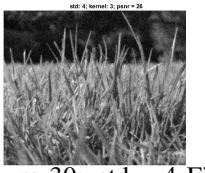
Figure 30: std = 4, kernel = 3


Figure 31: std = 5, kernel = 3

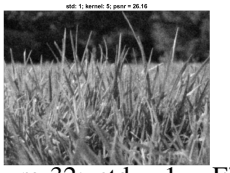
Figure 32: std = 1, kernel = 5


Figure 33: std = 2, kernel = 5


Figure 34: std = 3, kernel = 5


Figure 35: std = 4, kernel = 5


Figure 36: std = 5, kernel = 5

The gradient magnitude is the approximation of the gradient of the image, and, at every pixel it signifies its absolute difference from the pixels exactly above and to its left.

4.2.2 Second-order derivative filters - Question-9

Another way of detecting the edges in an image is by determining the zero-crossing points in the second order derivative of the image. Figures 46, 47, 48 represent the results of applying three different methods of approximating the second order derivative on the same image. The first method consists in applying a smoothing Gaussian kernel, followed by a Laplacian one. The second method consists of applying the Laplacian of Gaussian kernel on the image, and considering that convolution is an associative operation, this is mathematically equivalent to the first method. The last method is calculating the difference between the results of applying two Gaussian filters of different standard deviations. It works by subtracting low frequency changes from high frequency changes, thus creating a band pass filter that singles out the information describing the edges. Thus, the results from the first two methods differ from one another mostly in intensity (which can be minimized by normalization), but they differ in shape from the last method.

In the first method it is important to use a Gaussian kernel before applying the Laplacian in order to smooth out the image. Otherwise the small, insignificant details in the image will result in detecting unwanted edges.

When tuning the third method, we have chosen a quite low standard deviation for the first Gaussian (0.5) and a significantly higher one for the second Gaussian (2.0), having a ratio of 4:1. This ensures that the first result smooths out high frequency brightness changes while the second one contains low frequency changes. Calculating the difference results in smoothing out anything that has a higher frequency than the first one and deleting the changes of lower frequency than the second one, thus extracting the edges in the image.

A possible nest step, after applying any of these methods, is to clean stray pixels that do not take part of a border and to fill in the gaps of borders. This can be done with a heuristic algorithm and can aid in isolating different objects in the image.

4.3 Foreground-background separation

Running the algorithm with the given parameters on the test images yields completely unreasonable separations. This was mainly caused by $\psi = 0$. As the task specifies to look at λ, σ, δ, we tested a few values for ψ and found that changing it to 3 led to useful segmentations. One example shall be given in Figures 59-52, which shows our default segmentation of the `Kobi` image. The other default segmentations can be found in the Appendix. We also found that higher σ result in fewer but less sharp clusters while lower values resulted in more but sharper clusters. Applying no smoothing at all results in a very pixelated and thus not very useful clustering as can be seen in 49. Those results are intuitive, since a wider smoothing results in a wider range of pixels that are likely to be clustered together.

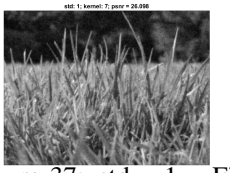

Figure 37: std = 1, kernel = 7

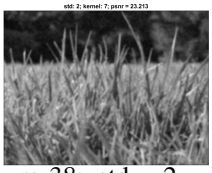

Figure 38: std = 2, kernel = 7

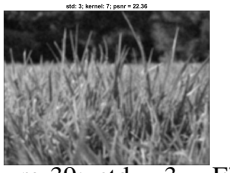

Figure 39: std = 3, kernel = 7

Figure 40: std = 4, kernel = 7

Figure 41: std = 5, kernel = 7

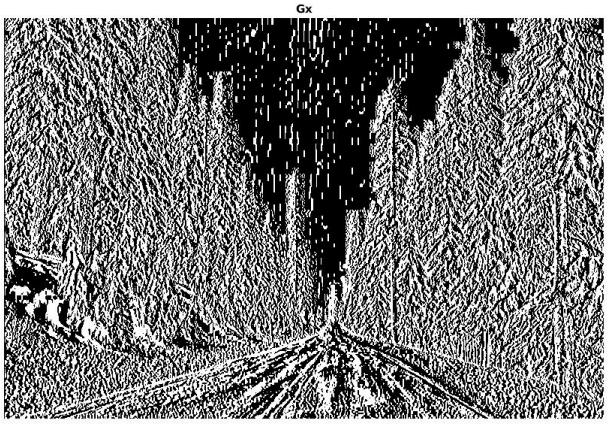

Figure 42: Gradient on the x direction

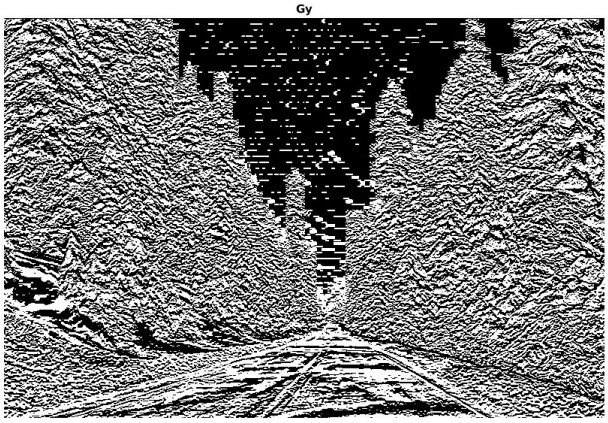

Figure 43: Gradient on the y direction

5 Conclusion

By looking at basic filters and operators we got a good overview over their applications. By inspecting the parameters, we better understand the possible variations of the Gabor filter and how we have to tune it to analyze certain textures.

Looking further into image processing techniques we understood the processes of denoising, edge detection and foreground-background separation. Thus we learned about the sources of noise and came to the conclusion that the median filter is most appropriate for salt and pepper noise. We then compared first and second order derivative methods of detecting edges and applied our acquired nowledge of the Gabor filters to the problem of foreground-background separation.

Appendix Images

References

[1] D. Jacobs. Correlation and convolution. *Class Notes for CMSC*, 426, 2005.

[2] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

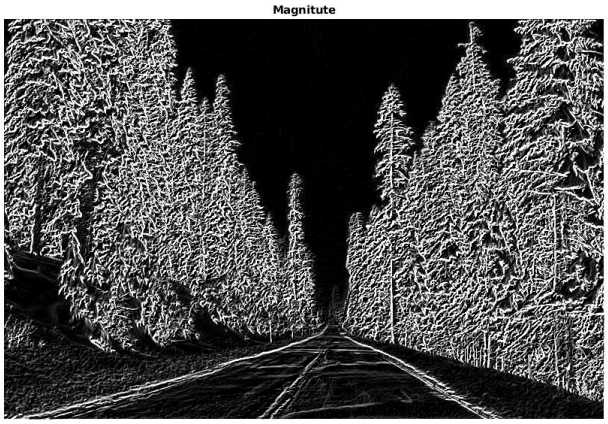

Figure 44: Gradient Magnitude

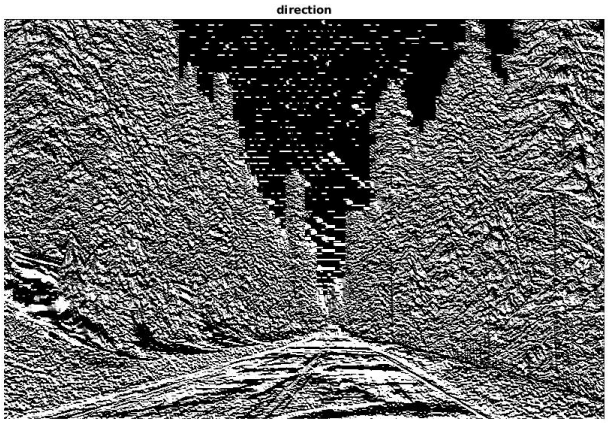

Figure 45: Gradient Direction

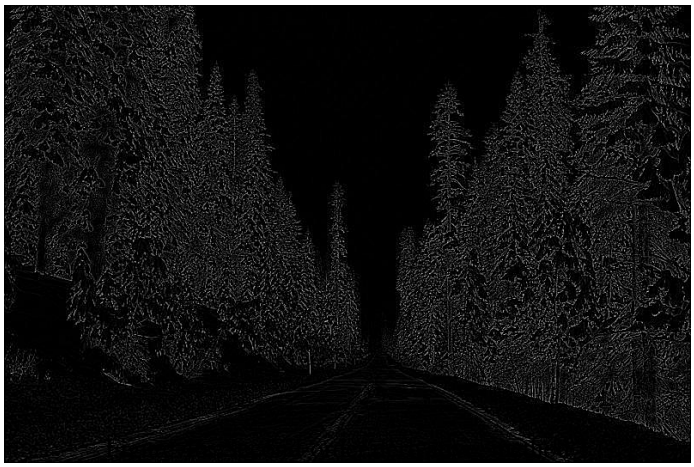

Figure 50: Original `Kobi` image

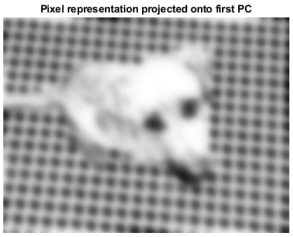

Figure 51: 1st PCA

Figure 52: Separation for default parameters

Figure 53: Original `Robin1` image

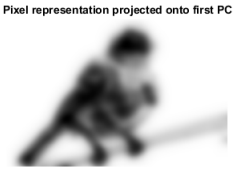

Figure 54: 1st PCA

Figure 55: Separation for default parameters

Figure 56: Original `Robin2` image

Figure 57: 1st PCA

Figure 58: Separation for default parameters

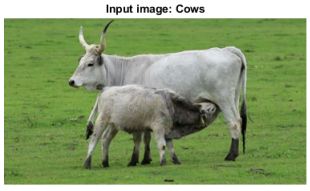

Figure 59: Original `Cows` image

Figure 60: 1st PCA

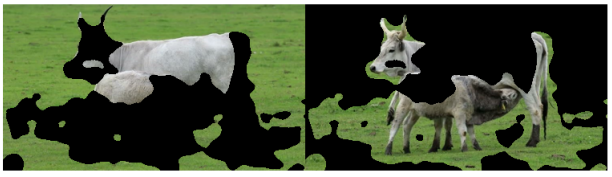

Figure 61: Separation for default parameters

Figure 62: Original `Sciencepark` image

Figure 63: 1st PCA

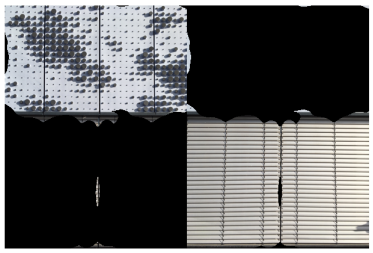

Figure 64: Separation for default parameters