Computer Vision 1, Assignment 2

Erik Stammes Max Bos Lars Rigter Jan Zuiderveld

February 27, 2019

1 Introduction

In this assignment we experiment with different low level filters for neighbourhood processing (focusing in particular on Gaussian- and Gabor filters). Different applications in image processing are explored including different denoising techniques (box- / mean-, median- and gaussian filtering), edge detection using first- and second order derivative filters and texture-based foreground-background separation using gabor filtering.

2 Neighborhood Processing

The difference between the correlation and convolution operators is that they both treat the signal $\mathbf{I}$ in the same way, but in the convolution process the mask $\mathbf{h}$ is rotated by 180 degrees.

When the mask $\mathbf{h}$ is symmetrical, the correlation and convolution operators are equivalent, since the rotation by 180 degrees of $\mathbf{h}$ that is performed in the convolution process yields a matrix equivalent to $\mathbf{h}$.

3 Low-level Filters

3.1 Gaussian Filters

3.1.1 1D Gaussian Filter

See the source code for the implementation of `gauss1D.m`.

3.1.2 2D Gaussian Filter

(a) (b) (c) (d)

Figure 1: Results for performing convolution on an original image (a), with a 1-dimension Gaussian kernel in the x- and y-direction (b), and with a 2-dimensional Gaussian kernel (c). (d) shows the difference in image pixel values between the images in (b) and (c).

See the source code for the implementation of `gauss2D.m`. If a filter is separable, as posed in the question, the difference between convolving an image with (1) a 2D Gaussian kernel and (2) a 1D Gaussian kernel in the x- and y-direction is that the former approach is computationally more complex than the the latter. Given an M-by-N matrix and a separable k-by-k filter, the computational complexity of (1) is $O(MNk^2)$, while the computation complexity of (2) is $O(2 * MNk)$. However, the image result will be exactly the same. We have run an experiment where an input image (Figure 1a) is convolved by a 2D Gaussian kernel

(Figure 1c) and a 1D Gaussian kernel in the x- and y- direction (Figure 1b). Figure 1d shows an image of the difference in pixel values between the resulting matrices for the two mentioned methods, from which we can conclude that there exists a minimal total difference of 2.2900e-13 in the results of the two methods, which could be due to rounding differences.

3.1.3 Gaussian Derivatives

It is interesting to design a second order kernel, because applying convolution with a second order Gaussian kernel filter is a process to localize/detect edges within a given image. In specific, the zero-crossings in the result correspond to the position of edges within a given image.

3.2 Gabor filters

3.2.2 Gabor filter parameters

λ corresponds to the wavelength of the sinus function e.g. the wavelength to which the filter responds. θ represents the orientation of the normal to the parallel stripes of a Gabor function e.g. the rotation of the sinus function. ψ is the phase offset e.g. in what sense the sinus/cosinus is shifted. σ is the standard deviation of the Gaussian. γ a parameter for the "elongness" of the ellipses. The effect of the variables can be seen inf figure 2. Most likely, sigma represents the width of the lines, theta represents diagonal lines and gamma represents horizontal lines.

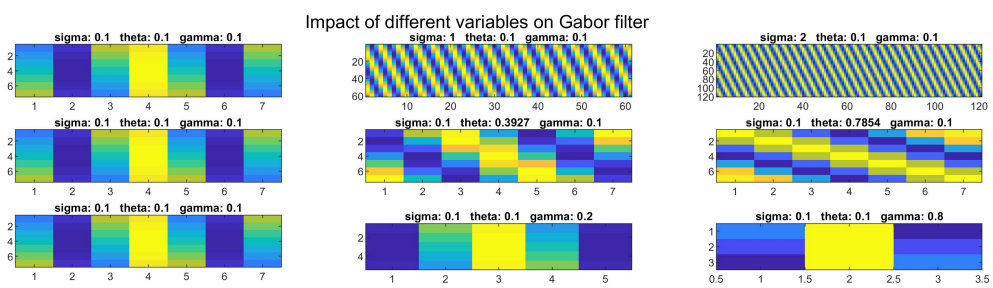


Figure 2: Impact of variables on Gabor filter

4 Applications in image processing

4.2.1 Quantitative evaluation

1. See the source code for the implementation of `myPSNR.m`. The computed PSNR between image1 saltpepper.jpg and image1.jpg is 15.67.

2. The computed PSNR using our implemented function `myPSNR.m` between image1gaussian.jpg and image1.jpg is 14.46.

4.2.2 Neighborhood processing for image denoising

1. The denoised images are shown in Figure 3.

(a) Box filtering of size: 3x3, 5x5, and 7x7 on image1 saltpepper.jpg

(b) Median filtering of size: 3x3, 5x5, and 7x7 on image1 saltpepper.jpg

(c) Box filtering with size: 3x3, 5x5, and 7x7 on image1 gaussian.jpg

(d) Median filtering with size: 3x3, 5x5, and 7x7 on image1 gaussian.jpg

Figure 3: Denoised images

2. For these images in combination with box- and median filters, a larger filter size results in a lower PSNR. So a 3x3 filter does the best job in this case, resulting in the highest PSNR. See Table 1.

3. Median filtering is better for salt and pepper noise, while box filtering is better for gaussian noise. This makes sense because salt and pepper noise is 'extreme' and sparse noise, which has too much influence when convolving an image with a straightforward averaging filter like a box filter. A median filter is better suited for such noise. With gaussian noise a box filter does a better job, since gaussian noise is spread out more evenly.

4. From our experiments the optimal standard deviation seems to be 0.95 in combination wit a window size of 5x5. The optimal parameter values were found using a grid search documented in Table 2.

5. A larger standard deviation makes the filter more evenly weighted, where a smaller standard deviation puts more emphasis on pixels closer to the filter center. As can be seen in Table 2, when a standard deviation is small enough, window size doesn't make a difference for PSNR values, and when the window size is smaller, standard deviation is less of a differentiating factor.

	Salt and Pepper	Gaussian
Box 3x3	15.7513	15.9535
Box 5x5	15.5664	15.6862
Box 7x7	15.3483	15.3926
Median 3x3	16.6560	15.4996
Median 5x5	15.9402	15.4912
Median 7x7	15.4915	15.2544

Table 1: PSNR's calculated using myPSNR.m on images with added salt and pepper noise and gaussian noise after being processed with a box and median filters with different sizes

Standard deviation								
	0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5
3x3 window	14.4613	14.4679	15.1034	15.7711	15.9225	15.9402	15.9331	15.9234
5x5 window	14.4613	14.4679	15.1047	15.8042	15.9686	15.9364	15.8648	15.8012
7x7 window	14.4613	14.4679	15.1047	15.8043	15.9667	15.9163	15.8044	15.6969

Table 2: PSNR's calculated using myPSNR.m on an image with added gaussian noise after being convolved with gaussian filters with window size's of 3x3, 5x5 and 7x7 and standard deviations ranging between 0.1 and 1.5.

6. Median filtering orders all pixel values in the filter window and outputs the middle ranking pixel.

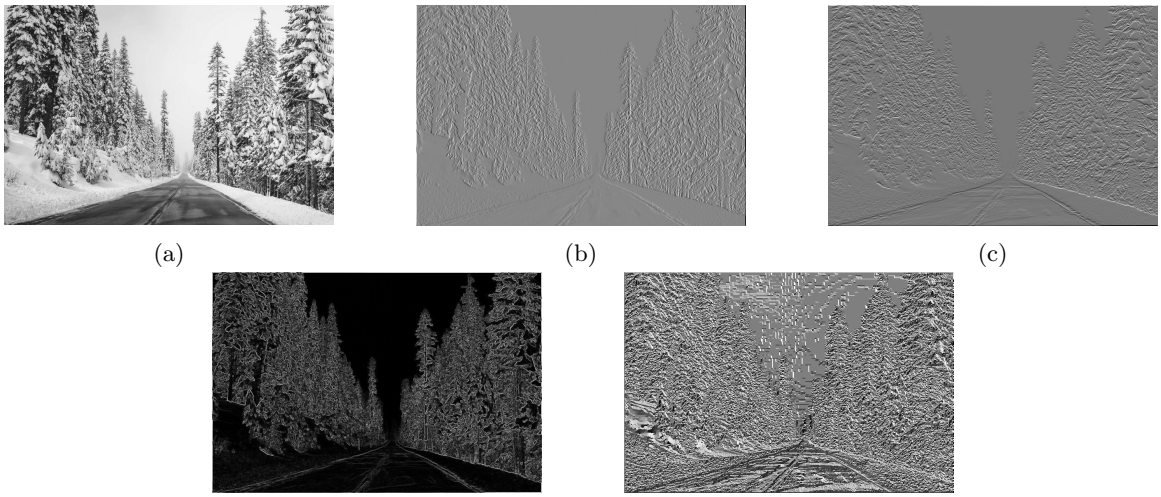
   Box filtering outputs a uniform average of all pixels in the filter.

   Gaussian filtering outputs a weighted average of all pixels in the filter, with centrally located pixels being weighted heavier.

   Median filtering is non-linear. It gives a distinctive look compared to box- and gaussian filtering, because extreme pixel values have practically no influence on the image. This creates a relatively sharp image compared to box- and guassian filtering. Box- and gaussian filters are both linear filters, giving a similar blurring look. When their PSNR values would be close to each other, chances are a human eye won't see much difference between the two denoised images.

4.3 Edge detection

4.3.1 First-order derivative filters

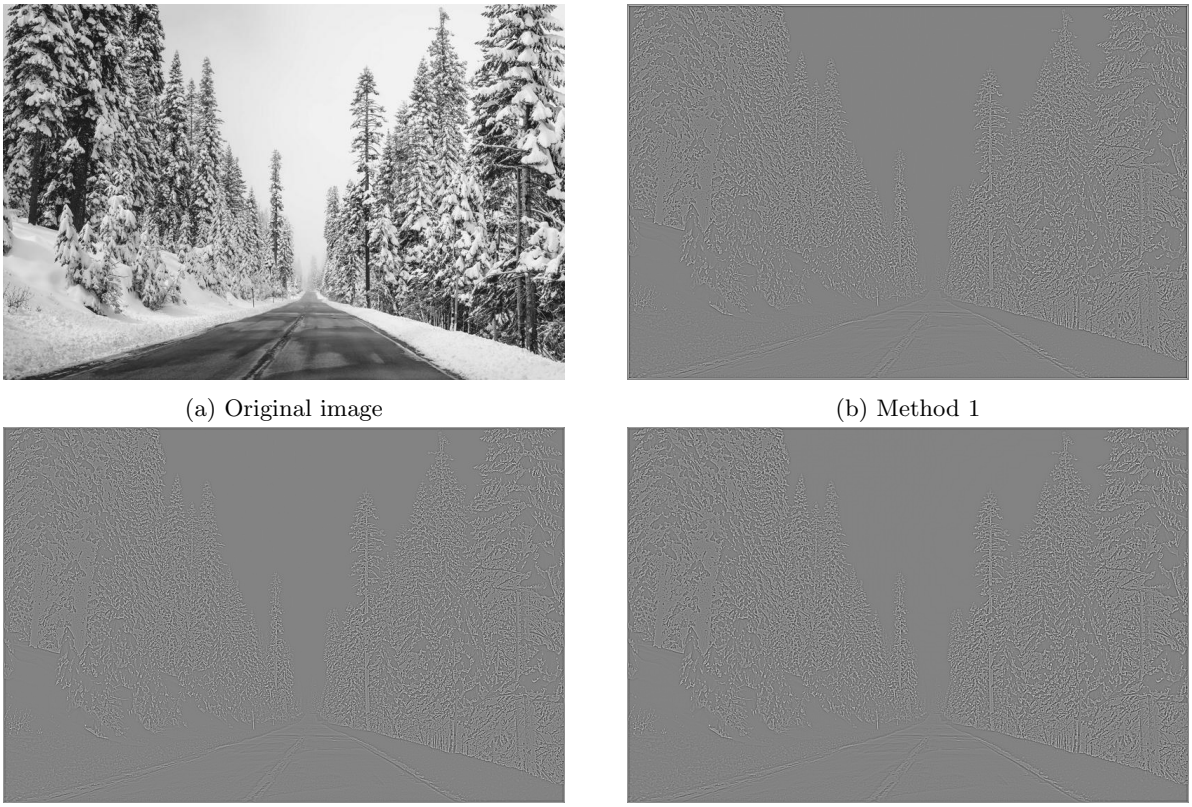


(a) (b) (c) (d) (e)

Figure 4: Results for computing, from (a), (b) the gradient with respect to the x-axis, (c) the gradient with respect to the y-axis, (d) the gradient magnitude, and (e) the gradient direction.

See the source code for the implementation of `compute_gradient.m`. We have run the gradient computation function on an input image shown in Figure 4a. The image for the gradient of the input image in the x-direction is shown in Figure 4b. This image shows the change in image intensity values along the x-direction, which conveys edges with a certain intensity from left to right sides. Similarly, Figure 4c shows the change in image intensity values along the y-direction, which conveys the edges with a certain intensity from the top to bottom sides. Figure 4d depicts the magnitude of the edges found by combining the intensity values for the edges found in the x-direction and y-direction. Figure 4e shows the direction of the gradient within the input image by combining the results for the partial gradients along the x- and y-direction.

4.3.2 Second-order derivative filters



(a) Original image (b) Method 1

(c) Method 2 (d) Method 3

Figure 5: Visualization of the three methods. (a) Original image. (b) Smoothing with Gaussian, then Laplacian filter. (c) Convolution with LoG kernel. (d) Difference of Gaussian.

The three different methods are implemented in `compute_LoG.m`. The results of running the three different methods on `image2.jpg` are in Figure 5. The results of all three methods are very similar, while the computation differs. The first method convolves the image in two steps, first a Gaussian and then a Laplacian convolution, while the second method does these convolutions in one step. The third method approximates the first two by computing the difference of two Gaussian convolutions on the same image, with two different scales σ_1 and σ_2. The image convolved with the smaller sigma is subtracted from the image convolved with the larger sigma value.

In the first method, it is important to first convolve with a Gaussian to reduce the noise in the image, before applying the Laplacian convolution.

For most images, the best ratio between σ_1 and σ_2 is 1.6 [Marr und Hildreth (1980)], which is true for the given image. In our case, that results in the values $\sigma_1 = 0.5$ and $\sigma_2 = 0.8$.

To isolate the road from the trees, one needs to find and mark the zero crossings after applying any of the three aforementioned methods. These zero crossings correspond to edges in the image and can be used to interpolate edges into boundaries, which divide the image into multiple segments.

4.4 Foreground-background separation

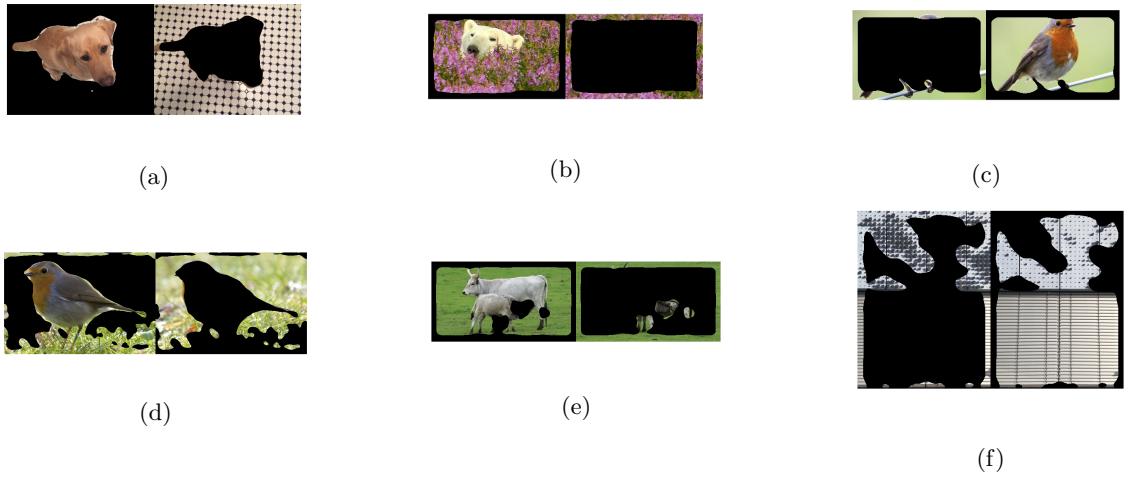


Figure 6: Results for running the Foreground-Background seperation algorithm on all given test images.

Figure 6 shows the results of running the Foreground-Background separation algorithm on the test images. The algorithm with default parameters performs well on some images (Figure 6a and 6d), but does not separate the foreground and background well on others. In some images, the algorithm only separates the border from the images (Figure 6b and 6e), while in others it has difficulty separating two tones of the same color (Figure 6f).

Figure 7: Results for running the Foreground-Background separation algorithm with tuned hyperparameters on images that were not separated well before

Figure 7 shows the results of the Foreground-Background separation algorithm after tuning the parameters, for the four images that did not work well with the default parameters. The used values are as follows:
Figure 7a: $\lambda_{min} = 1.05$, $\sigma = 0.3$
Figure 7b: $\lambda_{min} = 0.3, \sigma = 0.6$
Figure 7c: $\sigma = [0.01, 0.05], \psi = 4$
Figure 7d: $\lambda_{min} = 0.3, \sigma = 0.1665$
All others parameters were kept at their default values. In all cases we decreased the value for σ, because that decreases the standard deviation of the Gaussian, which allows the algorithm to pick up smaller changes between patches of images. Also, λ_{min} was in most cases decreased to regulate which patches would be separated in the algorithm.

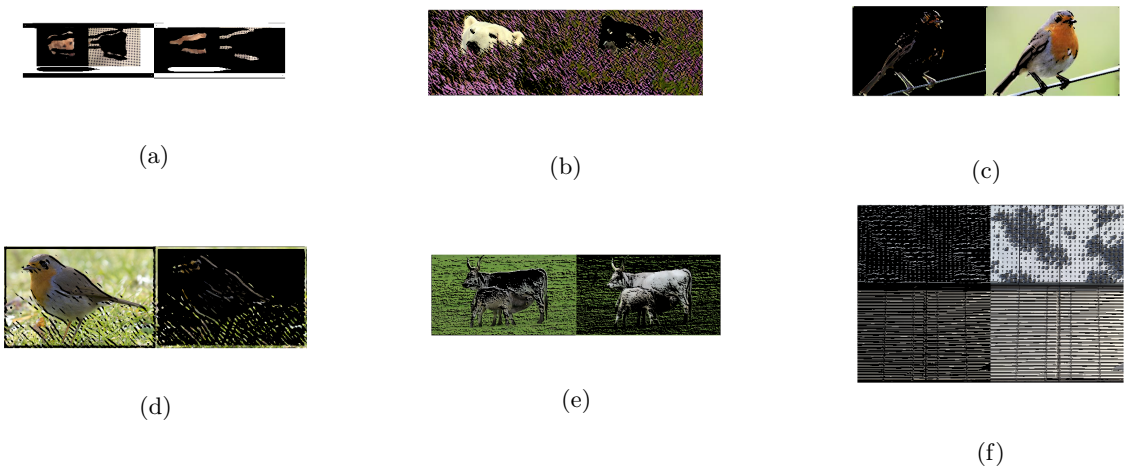


Figure 8: Results for running the Foreground-Background seperation algorithm on all given test images, using tuned parameters, but without smoothing

The images in Figure 8 are barely separated, the algorithm fails to find good clusters in the filterbank and thus cannot separate the foreground from the background. The Gabor filter needs smoothing because than it can isolate narrowband components of a compound waveform, concentrating them for easier detection.

5 Conclusion

In this assignment we experimented with computer vision methods such as edge detection, image denoising and texture-based image segmentation. The obtained results were insightful to us however we realized that certain methods work well in particular for synthetic images. The assignment allowed us to experiment with different methods, parameters, tricks and algorithms, giving us deeper insight into computer vision techniques.

5.1 Contribution

Everyone contributed equally. While the questions were divided amongst the team members, we cooperated on most questions and all came together to discuss the final result and gave feedback where needed. We worked on this together in the same room at the same time at the university. We all agree with the final submission.

References

[Marr und Hildreth 1980] Marr, David ; Hildreth, Ellen: Theory of edge detection. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 207 (1980), Nr. 1167, S. 187–217