Assignment 2
Neighbourhood Processing & Filters

Maartje de Jonge
0194107
University of Amsterdam
maartjedejonge@gmail.com

Lea van den Brink
10909419
University of Amsterdam
leavdb@hotmail.com

1 Introduction

This second assignment focuses on the implementation and application of Gaussian filters and Gabor filters. We discuss the implementation of gaussian filters in Section 3, and the implementation of Gabor filters in Section 4. Next in Section 5 we apply box, median and Gaussian filters for image denoising. We then apply first and second order derivatives of the Gaussian filter for edge detection in Section 6. Finally, in Section 7 we apply gabor filtering to separate the foreground of an image from its background. Our results show that these methods can be applied successfully for these important computer vision tasks.

2 Neighbourhood Processing

- **Question 1.1**

  Both convolution and correlation operate on an image by applying a kernel locally using a sliding window to pass the full image [4]. The difference between correlation and convolution operators is that convolution is a flipped version of correlation. The filter is upside down.

  A key property that only holds for convolution is associativity, that is: $(F*G)*I = F*(G*I)$. This property allows for precomputing composed filters which benefit performance since we only have to convolve a given image once. In general, convolution is used for image processing operations such as smoothing, while correlation is used to match a template to an image in which case associativity is not important [4].

- **Question 1.2** The difference between correlation and convolution is none, however, when the filter is symmetric, since flipping it will not change a thing in that case.

3 Gaussian Filters

We implemented the 1D Gaussian filter by calculating the values for $G_\sigma(x) = exp(-\frac{x^2}{2\sigma^2})$ for x in the interval $-\lfloor \frac{ksize}{2} \rfloor <= x <= \lfloor \frac{ksize}{2} \rfloor$. We then normalised the values so that they sum up to one. Notice that the term $\frac{1}{\sigma\sqrt{2\pi}}$ can be ignored when calculating the values because the values are normalized in the end.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

We implemented the 2D Gaussian filter by taking the matrix product of two 1D Gaussian filters, the first along the y-axis (column vector) and the second along the x-axis (row vector).

- **Question 2** The result of passing an image with a 2D Gaussian kernel will be the same as passing the image with a 1D Gaussian kernel twice, once in the x direction and once in the y direction. The reason is that the convolution operator is associative.

  However, the performance of passing the image twice with a 1D Gaussian filter will be better [3]. Applying the 2D Gaussian kernel with size KxK on an image of size MxN requires approximately $MNKK$ calculations. On the other hand, passing the same image twice with a 1D kernel of size $1xK$ requires only $2MNK$ calculations. Thus, the computational advantage of using the 1D kernel twice is roughly $\frac{K}{2}$.

- **Question 3** The second order derivative (Laplacian) can be used for edge detection since edges correspond to the zero-crossings.

4 Gabor Filters

We implemented a 2D Gabor filter using the following equations for the real and imaginary parts (with $x' = x \cdot cos(\theta) - y \cdot sin(\theta)$ and $y' = x \cdot sin(\theta) + y \cdot cos(\theta)$):

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

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

- **Question 4**

  As stated in [1], Gabor filters are very interesting since they are assumed to be similar to visual systems that can be found in the visual cortex of the human brain. Gabor filters are used to detect local textures, which can be helpful in many situations. Below we list the parameters and their meaning as stated in [1]:

  - λ: the wavelength of the sinusoidal factor.
  - θ: The orientation of the Gaussian. Shows how the normal and parallel stripes are orientated.
  - ψ: offset of the phase.
  - σ: the standard deviation of the Gaussian.
  - γ: the spatial aspect ratio of the Gaussian, i.e. the ellipticity.

- **Question 5** Figure 1, 2 en 3 visualize the the effect of the parameters θ, σ and γ on the gabor filtering. The θ parameter controls the orientation (Figure 1), the σ parameter controls the spread of the gaussian (Figure 2), and the γ parameter controls the aspect ratio, i.e. the ratio between height and width (Figure 3).

Figure 1: **Effect of the parameter θ on the Gabor filter. The left image shows the real and imaginary parts for $\theta = 0$, the right image for $\theta = \frac{\pi}{4}$.**

Figure 2: **Effect of the parameter σ on the Gabor filter. The left image shows the real and imaginary parts for $\sigma = 1$, the right image for $\sigma = 3$.**

Figure 3: **Effect of the parameter γ on the Gabor filter. The left image shows the real and imaginary parts for $\gamma = 0.2$, the right image for $\gamma = 1.5$.**

5 Image Denoising

The peak signal-to-noise ratio (PSNR) is a metric that is commonly used to evaluate the performance of image enhancement algorithms. The metric quantifies the likeliness between the original, noise-free image and the result of the enhancement technique applied to a version of the original image with added noise. A high PSNR indicates a high-quality image, i.e. an image that is more similar to the original image.

We implemented the PSNR using the formula $PSNR = 20 * log_{10}(I_{max}/RMSE)$ with I_{max} the maximum pixel value of the original image and $RMSE$ the root of the mean squared error between the original image and the denoised image.

- **Question 6.1 and 6.2** First we calculate the PSNR of an image with added salt-and-pepper noise (Figure 4, second) and an image with added Gaussian noise (Figure 4, third). The PSNR of the salt and pepper image is 16.1 dB. The PSNR of the Gaussian image is 20.6 dB.

Figure 4: Original image **(First)**, Salt and Pepper noise **(Second)** and Gaussian noise **(Third)**.

5.0.1 Neighbourhood Processing for Image Denoising

- **Question 7.1**
  - **(a)** See Figure 5 for box filtering applied to image1.saltpepper.jpg, and Figure 6 for box filtering applied to image1.gaussian.jpg.

Figure 5: Salt Pepper Image, Box filtering: **(First)** 3x3, **(Second)** 5x5 and **(Third)** 7x7.

Figure 6: Gaussian Image, Box filtering: **(First)** 3x3, **(Second)** 5x5 and **(Third)** 7x7.

– **(b)** See Figure 7 for median filtering applied to image1.saltpepper.jpg (Figure 4, second), and Figure 8 for median filtering on image1.gaussian.jpg (Figure 4, third).

Figure 7: Salt Pepper Image, Median filtering: **(First)** 3x3, **(Second)** 5x5 and **(Third)** 7x7.

Figure 8: Gaussian Image, Median filtering: **(First)** 3x3, **(Second)** 5x5 and **(Third)** 7x7.

- **Question 7.2**

  We computed the PSNR values for all denoised images, the results are summarized in Table 1. We see that both box filtering and median filtering with given filter sizes results in higher PSNR values than the images with added noise (16.1 for salt-and-pepper and 20.6 for Gaussian noise), which suggests that these techniques improved the quality of the noisy images.

  The values in the table suggest that the PSNR decreases when the filter size increases. A possible explanation is that a small filter size is already effective for noise reduction, while a large filter size has the risk that small details in the original image get lost with the noise removal.

- **Question 7.3** The results in Table 1 suggest that Median filtering is more effective against salt-and-pepper noise than Box filtering (27.7 vs 23.4 for the optimal 3 x 3 filter size). At the other hand, for Gaussian noise, the difference is less clear (26.2 for box filtering versus 25.5 for median filtering for the optimal 3 x 3 filter size).

  A visual comparison between 3 x 3 box filtering and 3 x 3 median filtering on the salt and pepper image (Figure 5 respectively 7, first) exposes two problems of box filtering on salt-and-pepper noise. First, the edged become blurry since the box filter interpolates new values for pixels on the edges. Secondly, the black and white noise pixels with their unrepresentative values affect the mean values of all the pixels in their neighbourhood. This has the visual effect that the noise becomes more blurry but does not disappear. The median filter does not suffer from these shortcomings, that is, the noise disappears while the edges stay sharp [5].

  We also performed a visual comparison between 3 x 3 box filtering and 3 x 3 median filtering on the Gaussian noise image (Figure 6 respectively 8, first). However, in this case, we could not identify a clear difference in the quality of both denoising techniques. Th denoised images look more smooth than the Gaussian noise image, but they both look a bit blurry; which is explainable by the fact that the gaussian noise image also looks a bit blurry (contrary to the salt-and-pepper image).

Noise	Filter Type	Filter Size	PSNR (dB)
Salt Pepper Noise	Box	3x3	23.4
		5x5	22.6
		7x7	21.4
	Median	3x3	27.7
		5x5	24.5
		7x7	22.4
Gaussian Noise	Box	3x3	26.2
		5x5	23.7
		7x7	21.9
	Median	3x3	25.5
		5x5	23.8
		7x7	22.1

Table 1: PSNR values for applying box filtering and median filtering to an image polluted with, respectively, salt-and-pepper noise and Gaussian noise.

Image	σ	PSNR Value
Gaussian image	0.5	24.3
	1.0	26.2
	2.0	22.8

Table 2: PSNR values for denoising the image with added gaussian noise, using a gaussian filter with different σ values.

- **Question 7.4** We applied our 2D Gaussian filter to denoise the image with added Gaussian noise. We used a standard deviation of 0.5, 1, and 2. For these standard deviations we used a kernel size given by the formula $2 * \lceil (2 * \sigma) \rceil + 1$. This kernel size is chosen in such a way that it covers at least two standard deviations to both sides, and so that it is an odd number. The results are shown in Figure 9.

Figure 9: Gaussian Image, Gaussian filtering: **(First)** $\sigma = 0.5$, **(Second)** $\sigma = 1$ and **(Third)** $\sigma = 2$.

- **Question 7.5**

  Table 2 contains the PSNR values for denoising the image with added Gaussian noise using a Gaussian filter with different standard deviations.

  The standard deviation determines the distance weights of the filter and hence the amount of smoothing. Sigma 1.0 gives better results than both sigma 0.5 and sigma 2.0. A possible explanation is that a sigma that is chosen too low does not remove noise effectively, resulting in a lower PSNR and a lower quality of the image. On the other hand, a sigma chosen too high results in a blurry image and the loose of small details during denoising; also resulting in a lower PSNR and a lower quality of the image. A visual comparison of the images in Figure 2 confirms this idea.

- **Question 7.6** Median filtering replaces each pixel value with the median value of its local neighbourhood; Box filtering replaces the value of each pixel by the average of all the pixel values of its local neighbourhood; gaussian filtering differs from box filtering in that it replaces each pixel with a weighted average of the neighbouring pixels. The weights are set by the 2D Gaussian function parameterized by the sigma parameter which determines the degree of smoothing.

  Box filtering has the disadvantage that unrepresentative salt-and-pepper noise pixels affect the values of their neighbours and that edges become blurry. The Gaussian filter is better in

retaining the edges because of the lower weights for far distance pixels. The Gaussian filter is suitable for removing Gaussian noise. The median filter keeps sharp edges and is optimal for removing salt-and-pepper noise [5].

PSNR measures the likeliness of a denoised image with the original noise-free reference image. The PSNR is intended to correlate with quality as perceived by humans, however, in practice two images with the same PSNR value may have different quality according to humans. For example, one image may have a relative low PSNR value due to remaining noise (under smoothing) while the other image may have a relative low PSNR value because of blurry edges and lost details (over smoothing). These images are different despite their similar PSNR value and as a consequence, the quality judgement of humans may also be different.

6 Edge Detection

6.1 First-Order Derivative Filters

- **Question 8**
  We computed the gradients for the image showing the road. See figure 10. We see that uniform areas are coloured black, while changes in intensity, i.e. edges, are coloured white in the gradient magnitude image (Figure 10, third).

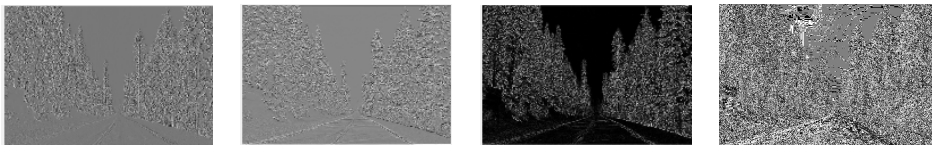

Figure 10: **(First)** Directional Gradient: X axis, **(Second)** Directional Gradient: Y axis, **(Third)** Gradient Magnitude and **(Fourth)** Gradient Direction.

6.2 Second-Order Derivative filters

We implemented the Laplacian of Gaussian using three different methods. First method: we smooth the image using a Gaussian and then take the Laplacian of the smoothed image. Second method: we convolve the image directly with a LoG kernel. Third method: we approximate the LoG by taking the difference of two Gaussians computed at different scales. For visualization purpose, we convert the resulting matrices to a greyscale image.

- **Question 9.1**
  See implementation and figure 11

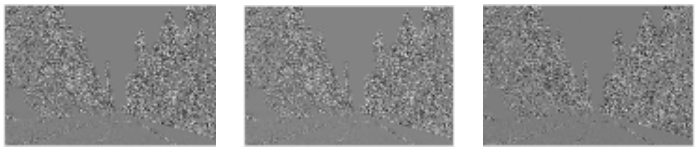

Figure 11: **(First)** Laplace after Gaussian, **(Second)** LoG filter and **(Third)** Dog filter.

- **Question 9.2**

  The first method first smooths the image with a Gaussian kernel and then applies the Laplacian. The second method is similar to the first given that the convolution operator is associative. However, the second method only needs one convolution pass where the first method needs two. This results in a faster processing time for the second method. The third method is an approximation of the LoG method that gives a faster processing time [2].

  Visual inspection of the results of the three methods applied to the road image in Figure 11 shows that these methods indeed lead to similar outcomes.

- **Question 9.3**
  The first method first smoothes the image with a Gaussian kernel. This will decrease the noise which is helpful for the Laplacian since the Laplacian reacts to rapid changes in the intensity of the pixels. Using a Gaussian kernel beforehand makes the results after the Laplacian less noisy.
- **Question 9.4** For the choice of the values for $\sigma 1$ and $\sigma 2$ we followed the recommendation given in [2]. That is, we set $\sigma 1 = \frac{\sigma}{\sqrt{(2)}}$ and $\sigma 2 = \sqrt{(2)} * \sigma$. The resulting ratio is therefore $\sigma 1 : \sigma 2 = 1 : 2$. For the value of σ we use 0.5 since this is the σ used by the other two methods. This settings for the σ gave a good result as shown in Figure 11 at the right.
- **Question 9.5** Both the gradient magnitude in Figure 10 (third) and the different versions of the LoG filter in Figure 11 show the edges of the road and multiple detected edges for the trees on both sides of the road. The edges of the road seem to be a bit thicker in the gradient magnitude image, also the higher contrast makes the contours of the road a bit more pronounced. Apart from these small differences, the images look quite similar.
- **Question 9.6** The next step in road detection is to implement a higher level feature that looks for the specific characteristics of a road. That is, a (striped) line in the middle, two lines at the left and right marking the borders of the road, and relatively uniform areas in between those lines. The road detector feature should also take into account the effect of 'depth', that is the lines meet each other at some point at the horizon as seen in the given image.

7 Foreground-Background Separation

We implemented functionality to separate foreground and background using a collection of Gabor filters with varying scale and orientations.

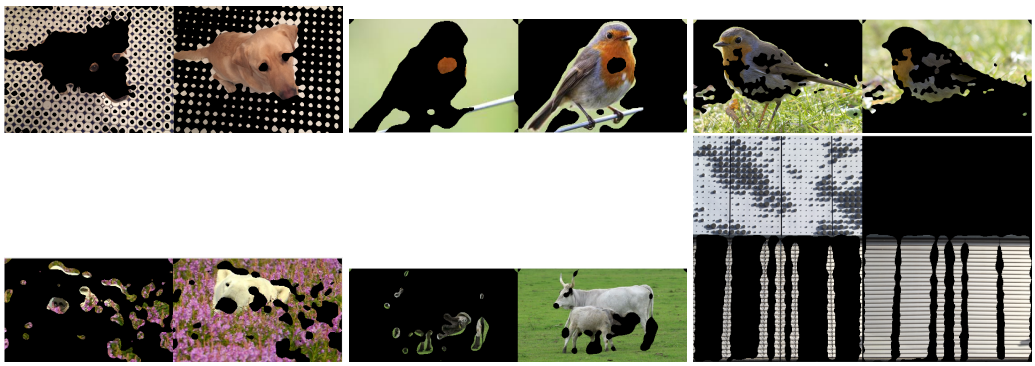

Figure 12: **Foreground background separation with gabor filtering using the predefined parameter settings.**

- **Question 10.1** We run the background/foreground separation function with the predefined settings on a set of images. The results are shown in Figure 12. We see that the segmentation with the given parameters works reasonably well for some images, while it fails completely for other images.

  The segmentation is successful for one of the birds (top, middle) and the science park blinds (bottom, right). We think that these images are relatively easy because of the flat uniform background of the bird and the clear texture patterns of the curtains. The segmentation also performs reasonably for the dog (top, left) and the other bird (top right). The dog is separated but the floor pattern is not correctly recognized as background. We expect that the dog segmentation can be improved with parameter tuning, since the background has a clear texture. The top-right bird is separated from the background but the grass that is in focus is also marked as foreground. Separating the focused grass seems hard since it has a different texture compared to the vague grass in the background. The segmentation fails for the polar bear (bottom left) and the cows (bottom middle). The polar bear seems separatable

based on texture but this requires some parameter tuning, the cows, on the other hand, may be difficult since the cows and the grass have a similar texture on first sight.

Figure 13: **Foreground background separation with Gabor filtering using tuned parameter settings. The bottom-right image shows pixel clusters for the bird top-left using three clusters instead of two.**

- **Question 10.2** We tried to improve the quality of the segmentation by tuning the parameters λ, σ and θ until we got reasonable outputs for all images. The results are in Figure 13 (we left out the science park blinds image because the given parameters already worked well and we did not manage to improve much with parameter tuning.).

  We made the following changes: for Kobi (top, left) we changed the sigma values from [1,2] to [2,3,4]; for Polar () we changed the sigma values to [3,4]; for Robin-1 () we changed the sigma values into [3]; for Robin-2 () we changed the sigma values into [2.8]; for Cows () we changed the sigma values into [2,3] and we decreased the lambda values by multiplying them with 0.1. We changed the sigma values in order to better adapt to the local size of the texture characteristics. In most cases, tuning the sigma parameter was sufficient. For the cows image we also adapted the lambda values in order to match more subtle structures. This improved the quality slightly.

  With Robin-2 (top, right) we identified the problem of separating the bird from the focused grass as well as the blurry background. To solve this issue we experimented with creating three clusters instead of two, i.e. setting the parameter $k = 3$. The result is shown in the bottom right as pixel clusters. The image shows that it is indeed possible to separate the background, the focused grass and the bird using three clusters.

- **Question 10.3**

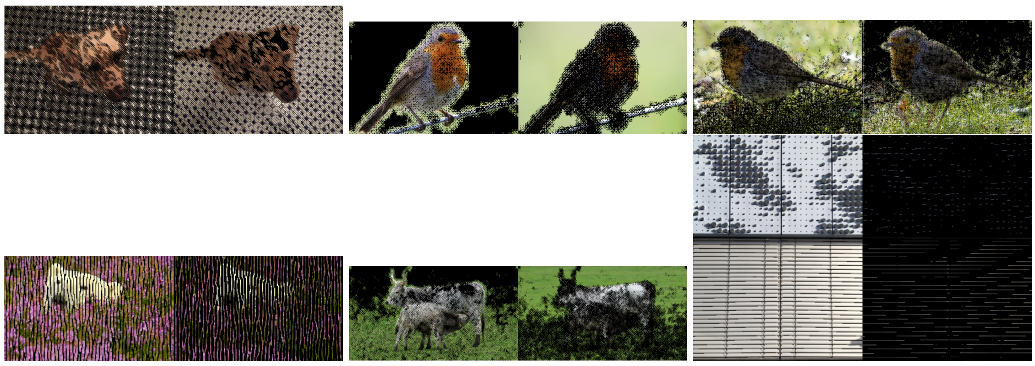

Figure 14: **Foreground background separation with Gabor filtering without smoothing the magnitude images. We used tuned parameters.**

We ran the segmentation function with the tuned parameters again, but this time without smoothing the magnitude images. The results are in Figure 14. We see that the segmentation is much worse for all images. We see some cases were the segmentation fails: the dog (top, left), the polar bear (bottom, left) and the science park blanks (bottom, right). We also see some cases in which the segmentation becomes noisy: the two bird images (top, middle and right) and the cows (bottom, middle).

The low-quality results of Figure 14 is caused by local accidental variations in the magnitude image. These local variations disturb the clustering step, leading to failures in segmentation or noisy results. The smoothing step smooths the noise in the magnitude image using a Gaussian filter. As a result, the magnitude image is cleaner and the segmentation leads to better quality results.

8 Conclusion

In this assignment, we investigated the application of different filters to important computer vision tasks.

We successfully applied box, median and Gaussian filters for image denoising; the success of these methods was measured quantitatively by means of the PSNR score, and qualitatively by visual inspection. Median filtering appeared to be the best method for salt-and-pepper noise. While all methods scored reasonably well on Gaussian noise, we could not identify a method that significantly out-performed the others on this type of noise.

We applied the first and second order derivative of the Gaussian filter to detect the edges of a road in an image. Both the gradient magnitude image as the image created by applying three different versions of the second order derivative filter showed the contours of the road in the resulting image. This illustrates the applicability of these filters for the task of edge detection.

Finally, we applied Gabor filtering for the task of foreground-background separation. This task was complicated because it required heavy parameter tuning. After tuning the parameters we managed to get reasonable results for most of the test images. The separation algorithm worked best for images in which the foreground had a combination of textures that was clearly distinct from the background.

References

[1] Gabor filter, https://en.wikipedia.org/wiki/gabor_filter, Feb 2018.

[2] Robert Collins. Log and dog filters, http://www.cse.psu.edu/ rtc12/cse486/lecture11_6pp.pdf, Feb 2018.

[3] Steve Eddings. Separable convolution, https://blogs.mathworks.com/steve/2006/10/04/separable-convolution/, 2006.

[4] David Jacobs. Correlation and convolution, http://www.cs.umd.edu/ djacobs/cmsc426/convolution.pdf, 2005.

[5] Rizzo Rosetta. Image noise removal, http://www.dmi.unict.it/b̃attiato/ei_mobile0708/noiseremoval (rizzo).pdf, Feb 2018.