Computer Vision 1: Assignment 2 Report

Petru Neague
(11844523)
petru.neague@student.uva.nl

Tomas Fabry
(11868414)
tomas.fabry@student.uva.nl

1 Introduction

In this project report we describe the following tasks:

- To implement a set of filters and run a series of images through the filters with the purpose of understanding how they work and apply them in noise reduction
- to implement first-order and second-order derivative methods for detecting edges in an image
- To implement a Gabor Filter used to separate the foreground from the background of an image

We adhere to the structure given by the assignment instructions and provide questions with the respective answers right below accordingly.

2 Neighbourhood Processing

2.1 Correlation and Convolution

Question 1. What is the difference between correlation and convolution operators? How do they treat the signals I and h?
A: Convolution are linear operations on the image, whereas correlation is a measure of similarity. In practice however, they treat **h** and **I** the same way, except that the matrix is rotated by 180 degrees. This also means that the difference only comes from how we use the filter, since we can use both operators for the same functionality by choosing different values for the kernel.
Correlation and convolution operators are equivalent when we make an assumption on the form of the mask h. Can you identify the case?
Since the difference between the correlation and convolution operators is just the flip of the filter, they are equivalent when the **h** is symmetric.

3 Gaussian Filters

In this section, the task was to implement a bi-dimensional Gaussian symmetric filter obeying the formula:

$$G_\sigma(x,y) = G_\sigma(x) * G_\sigma(y) = \frac{1}{\sigma^2 * 2 * \pi} * exp(-\frac{x+y}{2*\sigma^2}) \quad (1)$$

Where $G_\sigma(x,y)$ is the term which describes the distortion of pixel placed at coordinates (x,y); $G_\sigma(x)$ and $G_\sigma(y)$ are the individual components on the axes x and y and σ is the standard deviation of the symmetric Gaussian function.

Question 2. What is the difference between convolving an image with (1) a 2D Gaussian kernel and (2) a 1D Gaussian kernel in the x- and y-direction? Will the result be the same? What is their computational complexity?

Qualitatively there is no difference between a 2D Gaussian kernel and 2 1D Gaussian kernels for the x and y axes. Thus, the result should be the same. In terms of complexity however, there is a difference: Suppose we have an image of size MxN and a filter of size PxQ; for a 2D convolution, you need approximately M*P*N*Q multiplications and additions. For two 1D filters (of size P and Q), you only need MNP+MNQ = MN(P+Q) operations. Therefore, you get a speedup of the order of PQ/(P+Q).

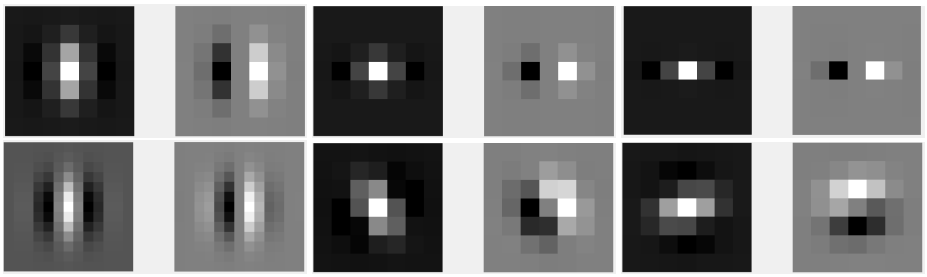
3.1 Gaussian Derivatives

Question 3. A second order derivative of the Gaussian kernel can also be computed. Why is it interesting to design a second order kernel?
A: As the first order derivative of the Gaussian kernel is used for edge detection on the smaller scale, a second derivative can be used for finding the larger-scale gradients in the image.

4 Gabor Filters

Question 4. Conduct a self-study on the Gabor filters. Explain shortly what the parameters λ, θ, ψ, σ, γ control.
A: The Gabor filter has an impulse response of the form of a Gaussian function multiplied by a sine / cosine function. Of these, θ controls the rotation angle signal expected to be measured, a θ = 0 indicates no rotation, i.e. vertical signal measuring. λ and ψ are part of the sine / cosine part of the formula and control the periodicity and offset of this oscillation. γ and σ are part of the Gaussian envelope and control the aspect ratio of the expected signal (ratio of the x axis to the y axis), and the standard deviation of the Gaussian function.

Question 5. Visualize how the parameters θ, σ and γ affect the filter in spatial domain.
It can be seen that when γ is changed, the alignment of the filter changes from being uniform on



The images in this figure are arranged in groups of 2: the real part of the Gabor filter and the imaginary part of the filter. From left to right we have the default Gabor filter: with σ = 1, θ = 0, λ = 5, ψ = 0, γ = 1, all the other sets of images are compared to this one (i.e. only one variable is changed). From there, the next set of images represent the data from $\gamma = 2$, the next set one represents $\gamma = 3$; the 4'th group represents $\sigma = 2$; the 5'th and 6'th groups represent $\theta = 0.5$ and $\theta = 1.8$

the x and y axes to being elongated along the y axis. σ changes the distribution of the data, and θ changes the angle of the measured pixels.

5 Image Denoising

Question 6.
Using your implemented function myPSNR, compute the PSNR between image1 saltpepper.jpg and image1.jpg. Report the PSNR.
A: The PSNR between saltpepper.jpg and image1.jpg is calculated to be 16.1079.

Using your implemented function myPSNR, compute the PSNR between image1 gaussian.jpg and image1.jpg.
A: The PSNR between the image1Gaussian.jpg and image1.jpg is calculated to be 20.5835.

Question 7.
Using your implemented function denoise, try denoising image1_saltpepper.jpg and image1_gaussian.jpg by applying the following filters:
(a) Box filtering of size: 3x3, 5x5, and 7x7.
(b) Median filtering with size: 3x3, 5x5 and 7x7.
Show the denoised images in your report.
A: See Figure 1 and 2

Figure 1: Denoising image1_saltpepper.jpg

Figure 2: Denoising image1_gaussian.jpg

Using your implemented function myPSNR, compute the PSNR for every denoised image (12 in total). What is the effect of the filter size on the PSNR? Report the results in a table and discuss.
A: See Table 1 and 2.

	3x3	5x5	7x7
Box filtering	23.40	22.64	21.42
Median filtering	27.69	24.50	22.37

Table 1: PSNR results for image1_saltpepper.jpg

	3x3	5x5	7x7
Box filtering	26.24	23.66	21.94
Median filtering	25.46	23.80	22.08

Table 2: PSNR results for image1_gaussian.jpg

Which is better for the salt-and-pepper noise, box or median filters? Why? What about the Gaussian noise?
A: Median filtering is better for salt and pepper noise. This is due to the fact that median filtering is less sensitive to outliers, which in our case are the salt and pepper pixels. The salt and pepper pixels have maximum or minimum values, therefore their color will be averaged over in the rest of the pixels when used box filtering, consequently creating too much of a blur. Median filtering uses the median, which is closer to what we want the pixels to look like. For the gaussian noise, both filtering methods perform similarly.

Try denoising image1 gaussian noise.jpg using a Gaussian filtering with a standard deviation of 0.5, 1 and 2 (i.e. the sigma parameter in your gauss2D function. Choose an appropriate window size and justify your choice. Show the denoised images in your report.
A: We choose kernel size 3 based on the results of the previous filtering methods showing that bigger kernel size performs worse in general when applied to this kind of noise. The images can be seen in Figure 3

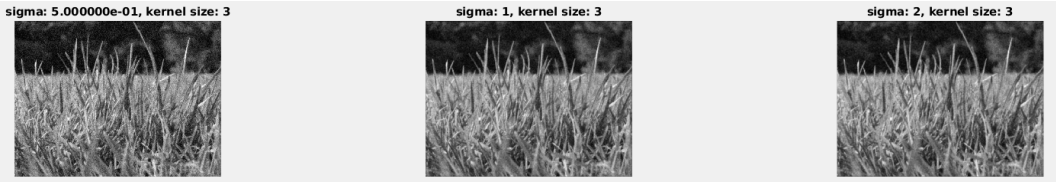


Figure 3: Denoising image1_gaussian.jpg using gaussian filtering

What is the effect of the standard deviation on the PSNR? Report the results in a table and discuss.
A: The results can be seen in Table 3. The best standard deviation option is 1. The intuition behind these results is that if we choose a sigma that is too small, the filter will focus on the center pixel too much and therefore not change the image enough, while setting it too high will result in too much blur. However it is worth it to note that this is somewhat an image-specific and noise-specific issue.

What is the difference among median filtering, box filtering and Gaussian filtering? If two filtering methods give a PSNR in the same ballpark, can you see a qualitative difference?
A: Box filtering simply averages the pixel values of the neighborhood given a pixel and update it,

Sigma	0.5	1	2
PSNR	24.29	26.60	26.15

Table 3: PSNR results for gaussian filtering

median filter does the same thing except it uses the median, not the mean, while gaussian filtering weighs pixels of the neighbourhood differently, giving more 'influence' to the center pixel and less to the pixels that are further.

For the case when PSNR measures give similar results, but the visual quality of the image is different: this is possible because the PSNR measure considers mean squared error, which takes the average error over all the pixels. Due to the inherent way averaging works, it can be the case that we have many pixels with a small error or few pixels with very big error, therefore getting two similar PSNR results but different image results. We can see difference in images when using gaussian filering, while the PSNR remains in the same ballpark. The higher the sigma, the more blurred the image becomes, which means many pixels have small error, whereas higher sigma has fewer pixels with higher error.

6 Edge Detection

Question 8.
Using your implemented function compute gradient on image2.jpg, display the following figures:
1. The gradient of the image in the x-direction.

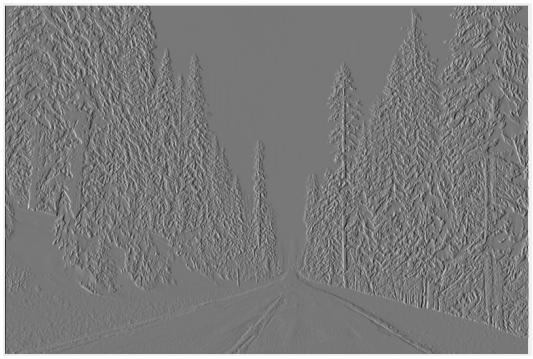


Figure 4: Gradient of the image in the x-direction

2. The gradient of the image in the y-direction.

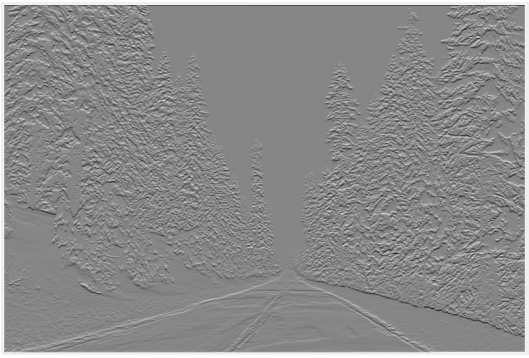


Figure 5: Gradient of the image in the y-direction

3. The gradient magnitude of each pixel.

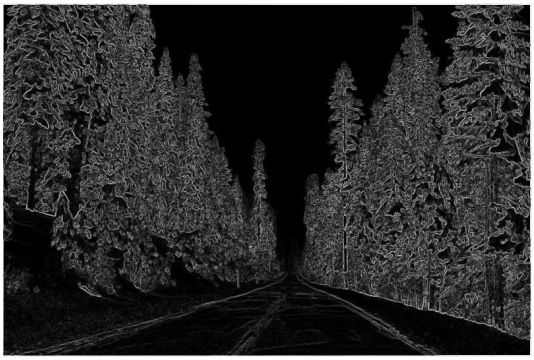


Figure 6: Gradient magnitude of each pixel

4. The gradient direction of each pixel.

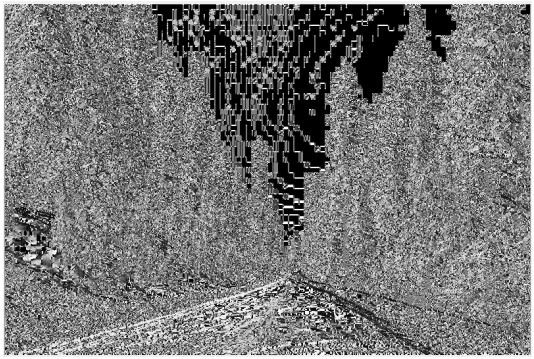


Figure 7: Gradient direction of each pixel

Question 9.
1. Test your function using image2.jpg and visualize your results using the three methods.

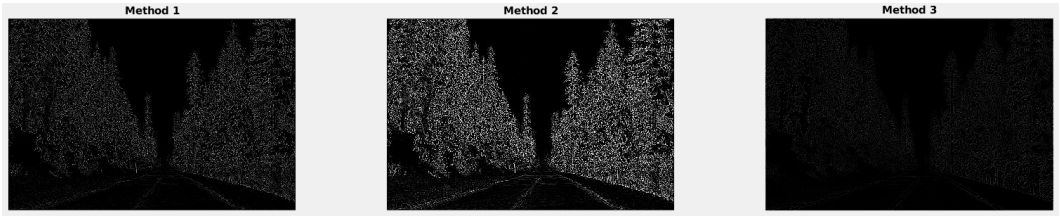


Figure 8: LoG Methods

2. Discuss the difference between applying the three methods.
A: The first method is applying a gaussian filter in the image first and taking the second derivative of the image. In the function, this creates a spike around the center, enveloped by negative values as illustrated in Figure 9. This is useful for example for detecting blops in an image. The second method differs in that the second derivative is taken on the gaussian kernel, which is then applied on the image, consequently creating a very similar effect. The reason why the images are different in brightness(although the pattern remains the same) is that the matlab fspecial function uses kernel size of 3 when using the 'laplacian' parameter, while with the second method we can specify kernel size to be 5. The third method approximates the discussed function by applying two gaussians on the same image separately and taking their difference. If the two gaussians have different standard

deviations, their difference causes the function to be similar in shape. The values tend to be lower and could also be approximated to be the same, given the perfect ratio between standard deviations.

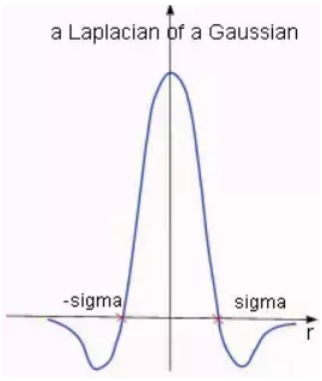


Figure 9: Laplacian of Gaussian illustration

3. In the first method, why is it important to convolve an image with a Gaussian before convolving with a Laplacian?
A: We do that for the purpose of approximating the Laplacian of Gaussian, which has been shown to be possible by blurring(or smoothing) the image first with a gaussian filter. It also helps to eliminate noise before taking the second order derivatives.

4. In the third method, what is the best ratio between $\sigma 1$ and $\sigma 2$ to achieve the best approximation of the LoG?
A: Ratio of 1.6 tends to produce the best approximation of the LoG. [1]

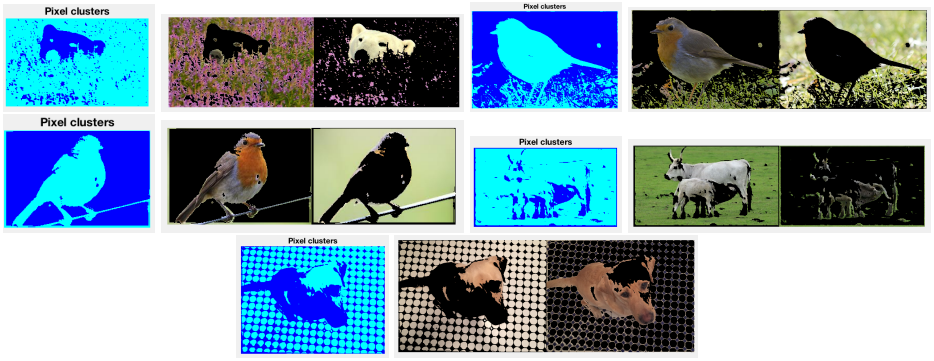
5. Do you notice a visible difference with the gradient magnitude of the first-order method?
A: Because Laplacian of Gaussian relies on the second derivative of a Gaussian filter, it will focus on large gradients in the image, as opposed to the first-order method, which better captures more local dependencies.

6. What else is needed to improve the performance and isolate the road?
A: Increasing the kernel size will focus on large gradients even more and might help detecting long lines, such as road lines.

7 Foreground-Background Separation

Question 10.
1. Run the algorithm on all test images with the provided parameter set- tings. What do you observe? Explain shortly in the report. A: The default values for θ ranged from 0 to $\frac{\pi}{2}$ in measures of $\frac{\pi}{8}$; λ took the values 2.8284, 5.6569, 11.3137, 22.6274, 45.2548, 90.5097; and σ was 1, 2, 3. The images from this setup are shown in Figure 7.

Figure 10: Segmentation results for the given images



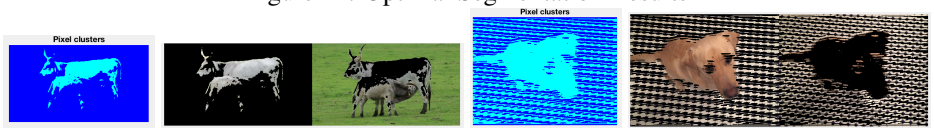
For each pair of images in this figure, the first one represents the abstract clusters of pixels belonging to the different clusters and the second is a comparison between the cropped out foreground and background. These sets of images from left to right and top to bottom: a polar bear in a field of flowers, a bird in a pose, a bird in another pose, 2 cows on a field and a dog on a patterned floor.

It seems the bear and the birds are well distinguished, even though there are still background elements around in all of them, it is safe to say that the algorithm works well for these images. For the dog and the cows however there are big issues. In the case of the dog the shade of the part of the head is confused (because of similar color, texture and lighting) with the floor is standing on, leading to poor results. As for the cows, their texture is the same with the ground they're on, and even though the color is different, the confusion still breaks the quality of the rendering.

2. Experiment with different λ, σ and θ settings until you get reasonable outputs. Report what parameter settings work better for each input image and try to explain why. Hint: Don't change multiple variables at once. You might not need to change some at all.

A: The first image that needed to be optimized was the dog. For this, after experimenting with many sets of coefficients of λ, σ and θ, it was decided that better alternatives of θ and λ were too hard to find. Larger wavelengths split the dog since his color gradient is significant on larger lengths; default θ was well chosen, after trying many intervals ($\pi/6$,$\pi/8$,$\pi/12$, $\pi/24$ and $\pi 36$) start points (0, $\pi/3$, $\pi/4$) and end points($\pi/2$, π, π and even 2π and 4π), it was decided that the default angle was the best option possible. The standard deviation was chosen to be 1, 3 here: 1 because it can get the small deviations in texture and 3 because it can get the part of the head that $\sigma = 1$ could not do. In the case of the cows, the same situation occurred where the default values for the θ and λ were the best ones that we could find. The standard deviation here was set to be 0.05, 1.5 because the small σ could get the small deviations and the large one could get the bigger ones. The result is still very inadequate, but after trying the same combinations as above, this was deemed to be the most efficient segmentation. The rest of the images were segmentated correctly and so they did not need additional interference

Figure 11: Optimal Segmentation Results



The best found segmentation of the cows and the dog.

3. After you achieve good separation on all test images, run the script again with corresponding parameters but this time with smoothingFlag = false; Describe what you observe at the output when smoothing is not applied on the magnitude images. Explain why it happens and try to reason about the motivation behind this step.

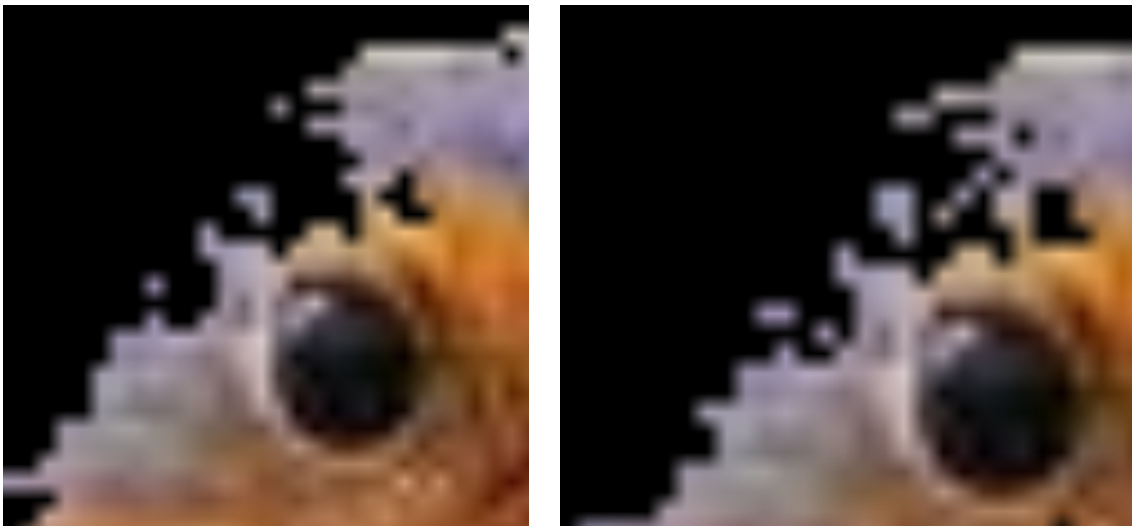


Figure 12: Picture on the left is with enabled smoothing, picture on the right with smoothing disabled

A: Although the difference is very tiny in this case, zooming in on specific parts of the image we can see on Figure 12 that disabling the smoothing(image on the right) causes some of the pixels that belong to the bird be misclassified as the background and vice-versa, although the former being more prevalent. When the image is smoothed first, it removes outliers to certain extent, consequently helping the separation. However, with σ set too high when doing the prior smoothing, it might cause, in particular, the edges of the main object to become too blurred with its environment, which decreases the 'accuracy' of the separation. Furthermore, it can cause the texture features to not be recognized by the filters in the filterbank. Again, it comes down to choosing the right parameters for smoothing to optimize this method.

8 Conclusion

We have implemented and experimented with multiple filtering methods to denoise images, find edges in them and separate the background from the foreground. The problems to solve are often problem-specific. As an example can serve background-foreground separation: different parameter values are to be used for different images. If the texture features on the image are very frequent and small, yet still important to capture, we might want to use smaller σ. Similarly, there is a difference for which type of filter to use with which parameters, depending on the type of noise. In this assignment, we have developed understanding and intuition for how various convolution filters and methods work and to which problems to apply them to, as described in the chapters above.

References

[1] Peter Kovesi. *Fast Almost-Gaussian Filtering*, volume E4271, pages 121–125. IEEE, Institute of Electrical and Electronics Engineers, United States, sydney, australia edition, 2010.