UNIVERSITEIT VAN AMSTERDAM

COMPUTER VISION

ASSIGNMENT 2

NEIGHBOURHOOD PROCESSING & FILTERS

Ahmet Taskale
Student Number:12344087
ahmet.taskale@student.uva.nl

Andreea Teodora Patra
Student Number:13365169
andreea.patra@student.uva.nl

September 21, 2020

ABSTRACT

.

1 Introduction

Note: This assignment was done by team A37 with only two group members. Please take this into consideration when grading the report.

This assignment includes image processing techniques, feature detection and image segmentation. Section 2 is devoted to neighbourhood processing, which is a method for the purpose of measuring relationships between neighbouring pixels, where a certain function h(k, l) is applied to a pixel I(x, y) in the given image I. This low-level technique will be put to use for image denoising with linear filters like the Gaussian filter and the Gabor filter. Furthermore, the term neighbourhood operator is explained and discussed by taking two neighbourhood linear filter operators as examples. Neighbourhood processing, along with its filters are mainly used for image enhancement. Section 3 includes the principles of the linear filters known as the Gaussian and the Gabor filters. These low-level filters are, as mentioned above, mostly used for image enhancement. Different purposes of the 1D and 2D Gaussian filters are discussed, which are used for image denoising and more complicated processes like edge detection. Gabor filters are Gaussian functions, integrated with sinusoidal carrier signals. For these type of filters, the 1D and 2D functions will also be discussed along with the parameters that defines them. Finally, section 4 will demonstrate the mathematical concepts by applying them to images for the purpose of denoising and segmentation.

2 Neighbourhood Processing

Neighbourhood processing is a method for the purpose of measuring relationships between neighbouring pixels, in which a certain function h(k, l) is applied to a pixel I(x, y) in the given image I. The function h(k, l), also known as the neighbourhood or local operator, takes various forms like the popular linear filter. The principle of a linear filter lies in computing the sum of pixel intensities of neighbouring pixels in a specified window, assigning the output to the pixel of interest. This section covers two types of linear filter operators known as correlation and convolution.

Question 1

1. Correlation and convolution are both processes in which a kernel, also called a filter mask, is moved over the whole image while the sum of the products are computed for each location . The difference in the processes is that the filter used for convolution is flipped by an angle of 180 degrees, being applied to the image afterwards. Correlation computes element wise values of the same neighbouring coordinates in the image as the coordinates of the kernel, summing the values of the pixel multiplied by the value of the kernel at the same coordinates, as mentioned before. In this case the center of the kernel is aligned with the image cell of interest. Convolution, also as mentioned before, uses the same process however the starting point of the element wise multiplication is mirrored (180 degrees flipped).

2. Correlation and convolution operators have the same effect whenever the values of the mask are symmetrical. In this case, it doesn't matter which filtering operator is applied to the image, since the resulting effect is the same. For example, imagine a kernel of 3 by 3 is used consisting of ones. Regardless of the starting point of the element wise multiplication (Top left corner or bottom right), the resulting weighted sum of neighbouring pixel intensities will be the same.

3 Low level filters

This section covers the design and information about so called low-level filters in neighbourhood processing, such as the Gaussian and Gabor filters. Both filters are classified as linear filters for the purpose of image enhancement (e.g. denoising).

Question 2

There is no difference between convolving an image with a 2D Gaussian kernel and a 1D Gaussian kernel twice (x and y), since the 2D Gaussian kernel also convolves in the x and y direction. However, in the latter case, the convolution process in the x and y direction is initialized simultaneously.

Question 3

As mentioned in the question, Gaussian derivatives serve as building blocks for edge detection or segmentation. Usually edge detection methods are classified into two categories. The first derivative of a Gaussian function is a search-based method, which relies on pointing out the maximum first derivative values of pixels in order to detect edges. The second derivative of the Gaussian function is a zero crossing method. which implies that the method is pointing out regions where the value of the Laplacian changes sign and passes through zero. The second derivative of pixel values in an image detects regions of drastic change in intensity.

Question 4

γ (aspect ratio): Gamma defines the elipticity. For example, $x^2 + y^2$ is a circle and when gamma = 1 this results in a circle. Lowering the value for gamma close to 0, the output results in a more eliptical shape in the y direction.

λ (wavelength): This parameter controls the wavelength. High values of lambda result in wider images, whereas lower values for lambda has the opposite effect.

ψ (phase offset): This parameters defines the offset of the sine and cosine factor, which is specified in degrees.

θ (orientation): Theta controls the rotation of the filter, which is also defined in degrees. A value of θ = 0 indicates no rotation and would make the ellipses vertical.

σ (standard deviation): Sigma defines the standard deviation/spread of the Gaussian distribution.

Question 5

As explained in question 4, the parameters θ, σ and γ control the orientation, the spread and the elipticity, respectively. To demonstrate the effect of these parameters on the filter, the figures below show the changes in θ and the resulting image of the filter.

Figure 1. Image of the Gabor filter kernel with values set at 5, 0, 10, 0 and 0.5 corresponding to σ, θ, λ, ψ and γ.

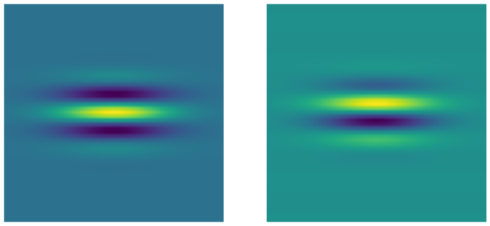

Figure 2. Image of the Gabor filter kernel with values set at 5, 90, 10, 0 and 0.5 corresponding to σ, θ, λ, ψ and γ.

Figures 1 and 2 show the difference in orientation when θ is set to 0 and 90 degrees. Naturally, if the image in figure 1 is representing 0 degrees, the image representing 90 degrees should be quarter turned. As for σ, figures 3 and 4 show the resulting differences when σ is set to 0.5 and 2.

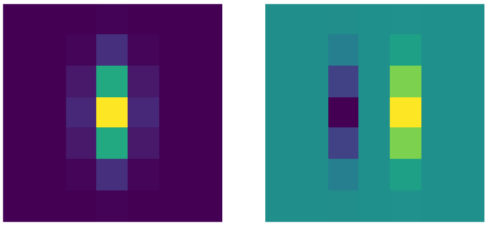

Figure 3. Image of the Gabor filter kernel with values set at 0.5, 0, 10, 0 and 0.5 corresponding to σ, θ, λ, ψ and γ.

Figure 4. Image of the Gabor filter kernel with values set at 2, 0, 10, 0 and 0.5 corresponding to σ, θ, λ, ψ and γ.

The parameter γ, also known as the aspect ratio, defines the elipticity of the Gabor filter. When this value is set to 1, the Gabor filter is circular. For values below 1, the filter is elongated in the y direction as shown in figures 5 and 6.

Figure 5. Image of the Gabor filter kernel with values set at 5, 0, 10, 0 and 0.1 corresponding to σ, θ, λ, ψ and γ.

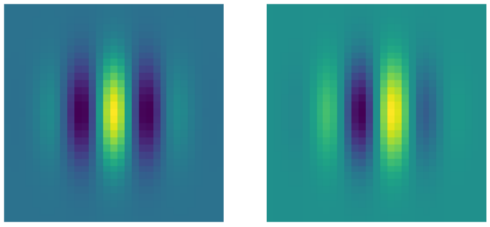

Figure 6. Image of the Gabor filter kernel with values set at 5, 0, 10, 0 and 1 corresponding to σ, θ, λ, ψ and γ.

4 Applications in image processing

Question 6

In this section, we evaluate a technique used for calculate the performance of image enhancement algorithms.

1. The mean squared error measures the departure of the approximation from the original image. Therefore, we aim for a small MSE value, as we want the enhanced corrupted image to be as close as possible to the original image. The PSNR uses the squared root of the MSE and when wanting to make MSE smaller and smaller, the PSNR gets higher and higher. Therefore, we would usually seek a higher value for PSNR.

2. We are using an implemented function of PSNR to calculate signal-to-noise ratio of an image corrupted with salt-and-pepper noise.

Figure 7. The original image

Figure 8. The corrupted image with salt-and-pepper noise

After the calculation, we get a value of 16.107930272610393 dB.

3. We are using an implemented function of PSNR to calculate signal-to-noise ratio of the same image corrupted with additive gaussian noise.

Figure 9. Image corrupted with additive gaussian noise

After the calculation, we get a value of 20.58354550645258 dB.

Question 7

In this section, we evaluate techniques used to remove the salt-and-pepper noise or the additive Gaussian one. In the salt-and-pepper noise, random pixels are replaced by either a white or black pixel. In the additive Gaussian noise, every pixel has a noise component sample from the same Gaussian probability distribution.

1. To remove noise, we implement 3 methods: box filtering, median filtering and Gaussian filtering. The box filters averages the pixel values in a KxK window. We convolve the image with a kernel of all ones and then scaling. The median filter computes the median of all values under the kernel window and the central pixel is replaced with this median value. The third method we implement is the Gaussian filter, discussed in the previous section.

We apply median and box filtering to the corrupted images presented in the previous Question.
Gaussian noise

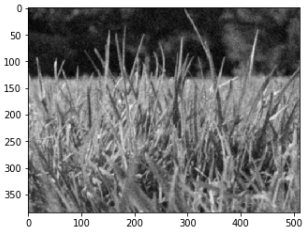

Figure 10. Box filter with 3x3 window

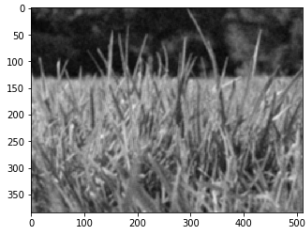

Figure 11. Box filter with 5x5 window

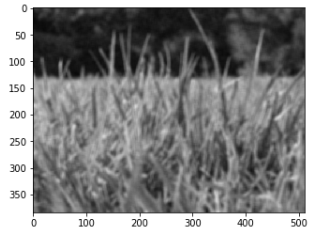

Figure 12. Box filter with 7x7 window

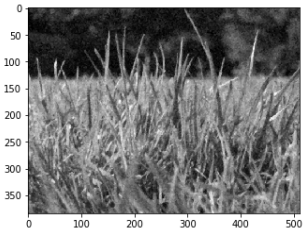

Figure 13. Median filter with 3x3 window

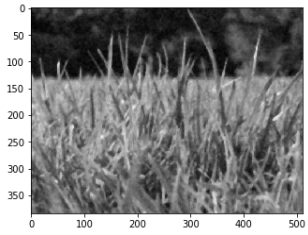

Figure 14. Median filter with 5x5 window

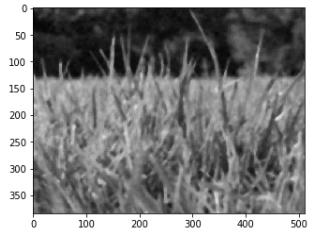

Figure 15. Median filter with 7x7 window

Salt and pepper noise

In the salt-and-pepper noise image, random images are either set to zero or a maximum value, thus creating that white and black dot pattern. When using the box filter, we get blurrier images as we increase the kernel size and the dot pattern is still slightly kept. Therefore, the box filter did not manage to get rid of the noise completely. On the other hand, the median filter manages to get rid completely of the noise as we can see in Figure 13. But as well, when increasing the kernel size, finer details are lost and the image becomes blurrier.

In the additive Gaussian noise image case, we can see that there is not a significant difference between the 2 methods as it can be further seen from the PSNR method as there is only a difference of 1 dB. The median filter shows a small improvement from the box filter. The image becomes again blurrier when we increase the kernel size.

2. Calculating the PSNR for the 12 images, we get the following results.

	Salt and Pepper Noise image	Additive Gaussian Noise image
Box filter 3x3	23.386104619727245	26.215422372069767
Box filter 5x5	22.630763131733033	23.648511343727314
Box filter 7x7	21.413595533347895	21.933532786837098
Median filter 3x3	27.85672121727408	25.528240927439963
Median filter 5x5	24.667103719735493	23.94164897106323
Median filter 7x7	22.545577896160125	22.243094251189298

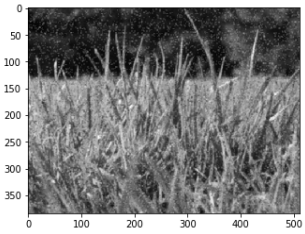

Figure 16. Box filter with 3x3 window

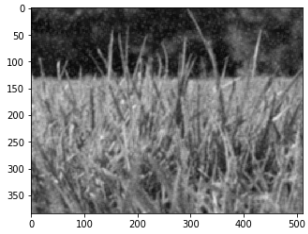

Figure 17. Box filter with 5x5 window

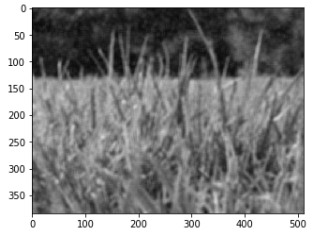

Figure 18. Box filter with 7x7 window

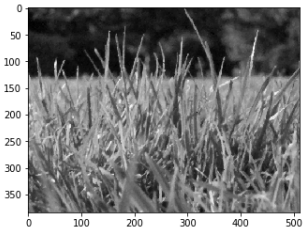

Figure 19. Median filter with 3x3 window

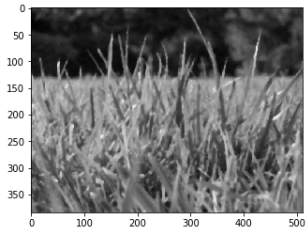

Figure 20. Median filter with 5x5 window

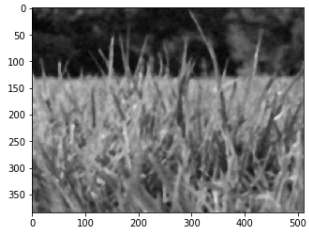

Figure 21. Median filter with 7x7 window

We have established that we are looking for higher values of the PSNR. Overall, we can deduce that increasing the kernel size, lowers the PSNR. Both of these methods are lowpass frequency filters which means that they remove the high spatial frequency components of an image. Therefore, when we raise the kernel size, we end up with less noise, but also with less frequency details. Reducing the fine-scaled image details means that our PSNR will become smaller and smaller. Looking at the values, we can see that for the salt-and-pepper noise image, the box filter with 3x3 kernel window performs better. On the other hand for the additive gaussian noise image, the box filter with 3x3 kernel window performs better.

3. For the salt-and-pepper image, the median filter outperforms the box filter. The box filter does a mean of the neighbourhood points and in our image, the corrupted values vary significantly from the original and as a result the mean can be different from the true value. Median does a better job preserving the edges and the small details, by taking the median of the neighbours.

For the gaussian image, the box filter outperforms the median filter. The median filter allows to pass more high frequency details, therefore is more useful in images that have less than a half of pixels in the smoothing neighbourhood affected. This will not help us as our image has every pixel slightly modified. Moreover, using the Central Limit Theorem, we know that by applying the box filter multiple times we get closer and closer to the true mean values, thus the central pixel true value.

4. The standard deviation in the Gaussian filter specified the degree of smoothing applied to the image. We are going to use a kernel window of 5 so that we can reduce most of the noise produced. The standard deviation chosen is 1. In practice, for a standard deviation of σ, we need at least $6\sigma - 1$ for the kernel size. We wanted to preserved the finer details, therefore we have chose a smaller value for σ. Using the formula, we needed a kernel size of 5. We also present denoised images using a kernel of 3x3 and standard deviation of 1 and a kernel of 11x11 and standard deviation of 2.

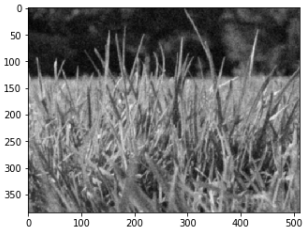

Figure 22. Gaussian filter with 5x5 window and σ of 1

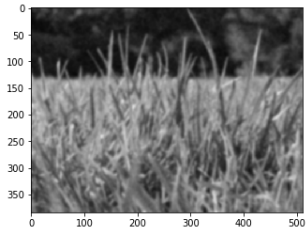

Figure 23. Gaussian filter with 11x11 window and σ of 2

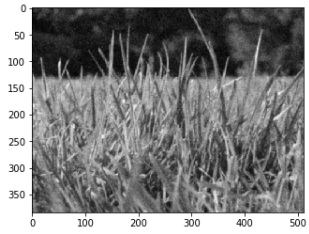

Figure 24. Gaussian filter with 3x3 window and σ of 1

We can see that the most blurred image is the one with $\sigma = 2$. Getting rid of all the noise comes at the expense of the finer details of the image. We can see that using a kernel of 5x5, we can get rid of the noise better than using a kernel of 3x3.

5. We are going to test the influence of the standard deviation on the PSNR. We are going to keep the kernel fixed of 5x5 and vary the standard deviation.

	PSNR
$\sigma = 1$	26.40808100214439
$\sigma = 2$	24.485731170541342
$\sigma = 3$	23.98765519921461
$\sigma = 5$	23.805188385789773
$\sigma = 7$	23.71989871814015

As expected, with $\sigma = 1$, we get the highest value for the PSNR. Choosing a higher value reduces the PSNR. Therefore, they are in an inverse relationship. We can also observe that the PSNR converges to a value of 23 dB.

5. The median filter is more robust compared to the box filter as single unrepresentative value pixel value in the smoothing neighborhood will not affect the central pixel value. On the other hand, the box filter performs better where all the pixels in the neighbourhood suffered a transformation. The median filter is also more computational expensive than the box filter as it requires sorting the values first, before selecting the median. The Gaussian filter provides a weighted average of each pixel's neighborhood, therefore it performs a gentler smoothing. The Gaussian filter is used more due to its frequency response. Mean filter has more oscillations in its frequency response than Gaussian filter. If 2 methods, give a PSNR in the same ballpark, we need to be aware of the computational costs as well. Moreover, there are situations where we are interested to eliminate all the noise completely for edge detection for example and we would not mind to have a blurrier image.

Question 8

In the next parts, we will focus on edge detection. Edges appear where there is a sharp change in brightness. For the convolution operation, we are going to use the mode='same' which applies padding so that we keep the size of the initial image and to preserve the edges.

In this question, we focus on the first order derivative filters, the Sobel kernels. The sobel kernels are described using a convolution operation. We are going to compute the kernels in the x and y direction, as well as the gradient magnitude and the gradient direction.

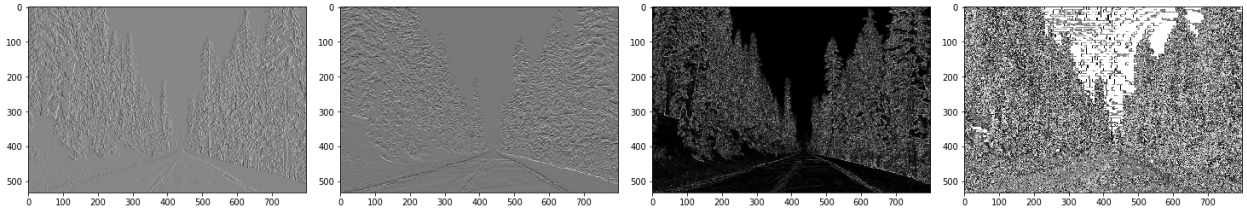

Figure 25. The gradient of the image in the x direction

Figure 26. The gradient of the image in the y direction

Figure 27. The gradient magnitude of each pixel

Figure 28. The gradient direction of each pixel

The gradient in the x direction focuses on the changes in the brightness on the x axis and as well, the gradient in the y direction focuses on the changes in the brightness in the y axis. Therefore, we can see that we can capture the road in the y direction. For the x direction, we can visualise as an edge detection in one sweep on the x axis. It shows the changes of the gray levels in the y and x direction. The bright spots mean a change from dark value to bright and on the other hand, the darker spots mean a change from bright to dark. We have a change from bright to dark in the road adn ca change from dark to bright in the trees. The gradient magnitude combines both the x direction and the y direction and outputs all the edges detected. The gradient direction shows the degree changes of going from G_x to G_y

Question 9

In this question, we focus on the second-order derivative filters. For the convolution operation, we are going to use the mode='same' which applies padding so that we keep the size of the initial image and to preserve the edges. We implement the Laplacian of Gaussian using 3 methods:

- smooth the image using a Gaussian kernel and then taking the Laplacian if the smoothed image
- convolve the image using a LoG kernel (kernel size of 5 and standard deviation of 0.5)
- take the difference of 2 Gaussian computed a different scales

1.

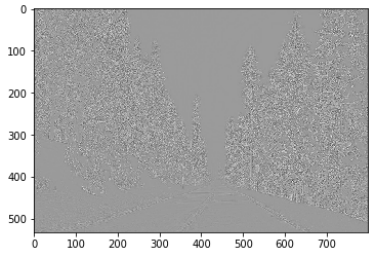

Figure 29. First method of LoG

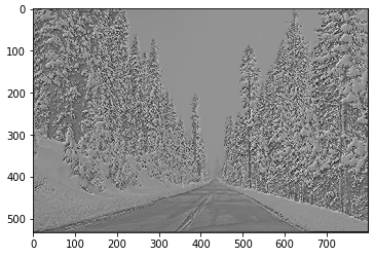

Figure 30. Second method of LoG

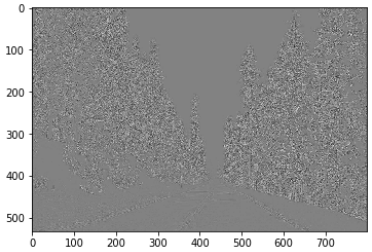

Figure 31. Third method of LoG using $\sigma_1 = 0.5$ and $\sigma_2 = 0.2$

2. In the areas where there is a constant intesity, the LoG response will be zero. Where there is a change in intensity, the LoG will be positive on the darker side and negative on the lighter side. In the second method we are calculating the kernel before hand and as a result we need to apply only one single convolution operation on the image. For the other methods, we apply 2 convolution methods. Using just one convolution operation gives us a more sharp image in the end.

3. We are approximating a second derivative measurement of the image which is very sensitive to noise, therefore we need to apply a filter so we can smooth the image and get rid of the noise.

4. We are using 2 different standard deviations because if we would have been using a single deviation, the filters would have captured the same edges and when we do the difference, they would have canceled each other out, leaving a black image. In theory, the smaller the ratio between the standard deviation, the better the approximation of the LoG curve we get. We would want at most 1.6 ratio between them.

5. The next steps that we would need to take is to perform zero crossing. We look for places in the image where the Laplacian changes sign as this is the main idea behind the algorithm. Edges occur at places in the image changes rapidly, therefore going from + to - . We need to pay attention to the size of the Gaussian, as if we have a higher standard deviation, then fewer and fewer zero crossings are going to be detected. However, a zero crossing can also happen at places where there is no edge. That is why we would need to introduce a threshold of the LoG output at zero.

4.1 Question 10

1. In Robin-2, we can see that some part of the leaves were considered to have similar texture with the feathers of the bird, probably due to the similar color. In the image with Cows, as there are 2 animals really close to each other, the algorithm has trouble differentiating between them. In image Robin-1, the algorithm performs as expected except for a dark patch from the head of the bird. In the Polar image, we get a good segmentation, apart from the nose which has a very different texture and color from its fur and therefore is considered from a different cluster. In the Science Park image, we manage to segment all the dots that appear clustered together from the image, apart from the isolated ones.

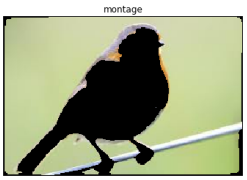

Figure 32. Segmentation of Robin-1

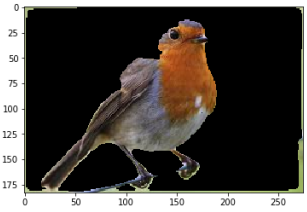

Figure 33

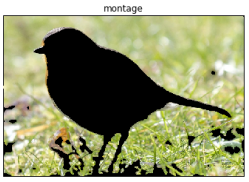

Figure 34. Segmentation of Robin-2

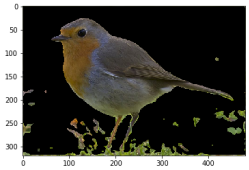

Figure 35

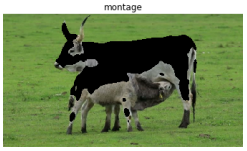

Figure 36. Segmentation of Cows

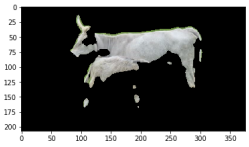

Figure 37

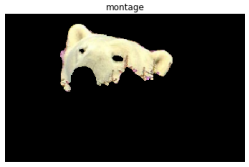

Figure 38. Segmentation of Polar

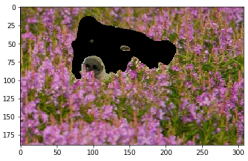

Figure 39

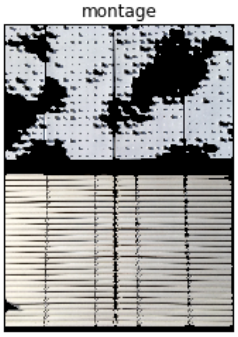

Figure 40. Segmentation of Science Park

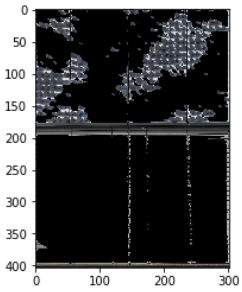

Figure 41

2. We are mainly focusing on θ and σ because they have the most impact on the Gabor Filters as explained in the previous sections.

The best parametrisation for the Polar image, which is close to the initial one is to keep λ unchanged,$\sigma = 0.5, 1, 2$ and θ from 0 to 150 using a step size of 30.

The best parametrisation for Robin-1 image, is θ from 0 to 180 with a stept size of 25, $\sigma = 0.5, 1, 2$ and λ unchanged.

The best parametrisation for Cows image, is θ from 0 to 150 using a step size of 30 and $\sigma = 1, 2, 3$. For the Robin-2 image, we keep the same parametrisation as in the Cows image.

Overall, we see small improvements when changing the parameters such as small patches from the background being removed from the main cluster of the segmentation.

5 Conclusion

The goal of this assignment was to have an overall understanding of Gaussian and Gabor filters, the implementation of edge detection and image denoising and performing texture-based image segmentation. We have stated that the neighbourhood processing method is for the purpose of measuring relationships between neighbouring pixels with a function h(k, l) serving as the neighbourhood or local operator. Two type of linear filters were summarized and and the differences were mentioned. Correlation is moved over the whole image while the sum of the products are computed for each location. Convolution is basically the same process, however in this case the kernel is flipped by 180 degrees.

As for the Gaussian filters, we concluded that there was no difference in convolving an image with a 2D Gaussian kernel and a 1D Gaussian kernel twice in the x and y direction. Furthermore, Gaussian derivatives are classified into two groups. The first derivative is a search-based method which localizes maximum first derivative values of pixels to detect edges. The second derivative is a zero crossing method which points out regions where drastic changes in

intensity occurs. The Gabor filter is characterized by the parameters σ, θ, λ, ψ and γ. Each parameter controls the shape of the filter to a certain degree.

Last but not least, we followed the necessary steps that we need to take in order to perform image segmentation. We evaluated techniques useful for denoising individual types of noise and how that affects the PSNR value. We conclude that salt-and-pepper noise is better removed with a median filter and for additive Gaussian filter we should use box filter. Moving on, we focused on various techniques for edge detection that incorporate Gaussian filters. As a result, we looked into Sobel and Laplacian of Gaussian kernels and how they handle the changes in brightness when detecting edges. For the Laplacian of Gaussian, we conclude that is better to directly convolve an image using the LoG kernel. Finally, we implement Gabor segmentation and evaluate the impact of different parameters on the clusters.