Computer Vision Lab 2

Maria Diea - 12593125 Diana Epureanu - 12710199
Teodora Stoenescu - 12598291 Razvan Tomescu - 12277584

September 2019

1 Introduction

In this report we start by looking at neighborhood processing used in image processing. The focus is on linear filters which are used in low-level image understanding. Gaussian and Gabor filters allow us to extract structural patterns or to analyse textures. Such techniques are very important building blocks in studying more complex procedures used in convolutional neural networks. Afterwards we shift our attention on simple algorithms used to correct noise in digital images. Lastly, we look at texture-based image segmentation where we use several Gabor filters to perform foreground-background separation.

2 Neighborhood Processing

Question 1. The difference between the correlation and convolution operators is that the convolution kernel h is rotated by 180 degrees around its center during the computation. Correlation is a similarity measure between h and I while convolution measures the effect of the signal h on I. In the case of h being symmetric, the output would be the same.

3 Low-level filters

3.1 Gaussian Filters

3.1.1 1D Gaussian Filter

This filter was implemented using the definition of a Gaussian.

3.1.2 2D Gaussian Filter

The 2D Gaussian filter was implemented by taking the product of 2 1D Gaussian filters, one on the x axis and the other on the y axis.
Question 2. The Gaussian filter is a separable filter, hence applying the 2D Gaussian filter has the same effect as applying two independent 1D Gaussian filters, on x and y axis. The resulting image will be the same. However, if we investigate the difference in computational complexity, we see that applying a 2D Gaussian filter requires $O(area_{kernel} \cdot area_{image})$ while applying two 1D Gaussian filter gives us $O(width_{kernel} \cdot area_{image}) + O(height_{kernel} \cdot area_{image})$. So for a square kernel we obtain $O(width^2_{kernel} \cdot area_{image})$ and $O(2 \cdot width_{kernel} \cdot area_{image})$. Thus, it is clear that using two 1D Gaussian filters is more advantageous from the computational complexity perspective.

3.1.3 Gaussian Derivatives

Question 3. It is interesting to design a second order kernel because it can also help in detecting edges in images. As Figure 1 shows, edges in images can be detected using the local minimum and maximum of the first order derivative, but also the zero-crossings of the second order derivative. The second derivative of Gaussian is a form of laplacian operator, where the laplacian has properties such as sensitivity to noise (because it is a second derivative in comparison to a first derivative) and also cheaper cost because it is only one mask. We can detect a lot of small unstable contours and because of the shape of the second derivative, the contour of the edges is well-defined, making them appear as 'framed'/ closed contours.

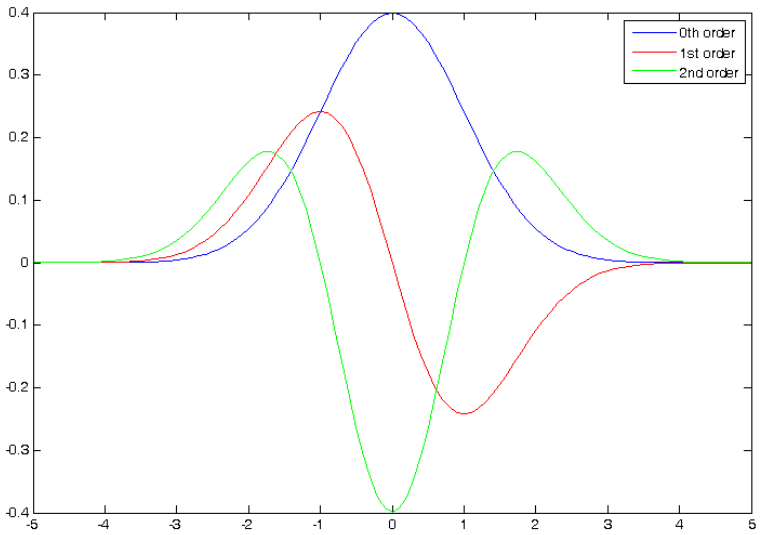


Figure 1: Gaussian curve

3.2 Gabor filters

3.2.1 1D Gabor Filters

3.2.2 2D Gabor Filers

Question 4. The 2D Gabor function then takes the forms:

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

in the real and complex parts, where:

- λ represents the wavelength of the sinusoidal factor and is measured in terms of pixels;
- θ represents the orientation of the normal to the parallel stripes of a Gabor function;
- ψ is the phase offset, it takes values between $-\pi$ and π;
- σ is the standard deviation of the Gaussian envelope, which controls the scaling;
- γ is the spatial aspect ratio, and specifies the ellipticity of the support of the Gabor function;

and lastly, x' and y' are the rotation by an angle of θ of x and y [1].

Question 5. Figure 2 shows the effects of different values for σ, θ and γ, with specific values as indicated in the figures.

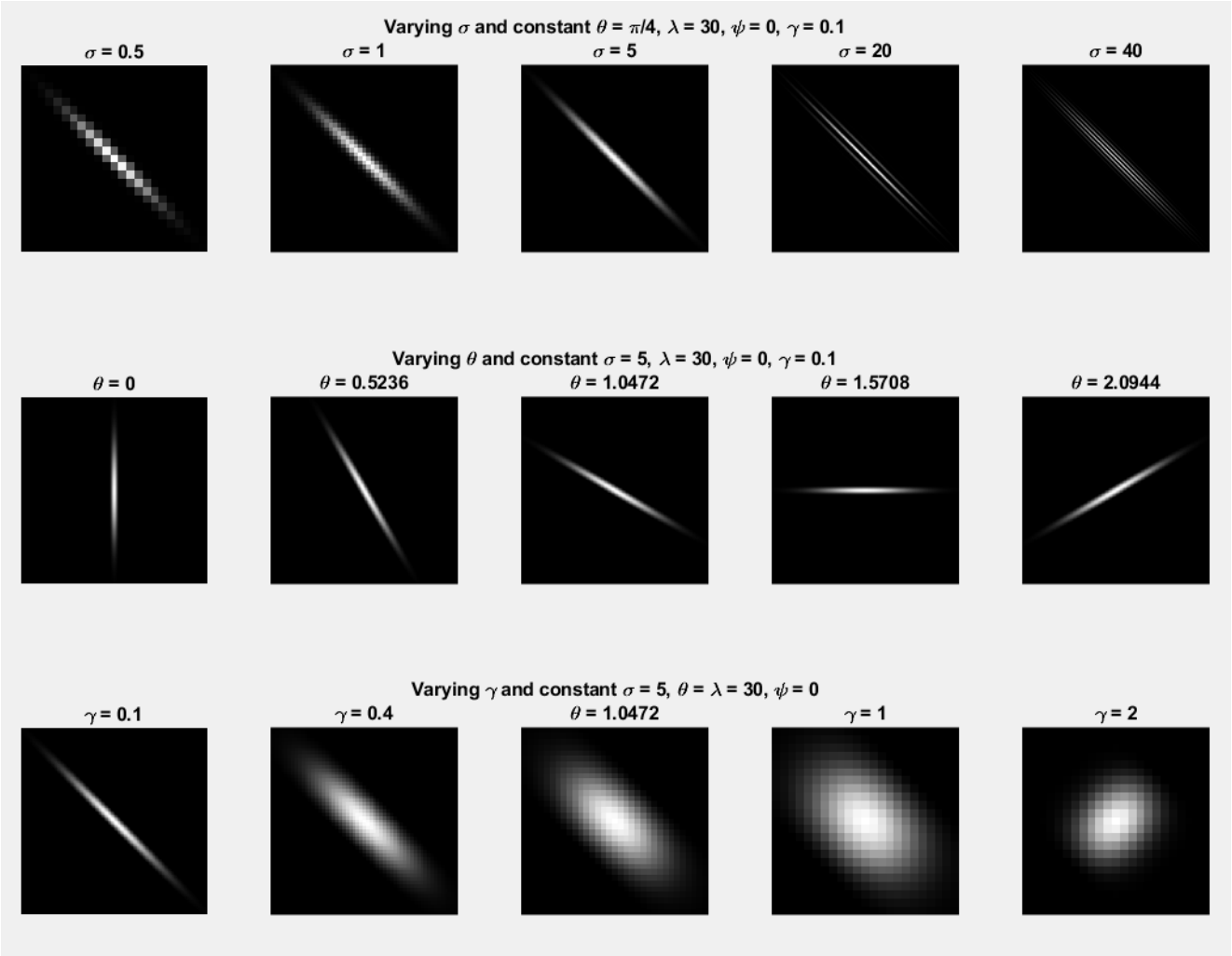


Figure 2: Different parameter values for the Gabor function

Looking at the different values of σ in the first row of images in the figure, it can be seen that it controls the size of the Gabor envelope. For larger sizes, the envelope increase which leads to the appearance of more stripes. θ controls the orientation of the Gabor function as it is clear to see from the second row of images in the figure. The γ values control the width of the Gabor stripes. The larger γ is, the thicker the stripes as easily observable from the last row of images from the figure.

4 Applications in image processing

4.2 Image denoising

Question 6 PSNR measures the quality of image enhancement. A high-valued PSNR implies less noise, therefore when comparing different methods for the same original image, a higher PSNR can imply better quality.

`myPSNR`*(image1.jpg, image1_saltpepper.jpg)*	16.10
`myPSNR`*(image1.jpg, image1_gaussian.jpg)*	20.58

Table 1: Reported PSNR between an image and two different corrupted enchantments

Question 7 Consider the denoised images presented in figure 3 and figure 4. The results of running `myPNSR` between *image1.jpg* and the two different noisy images are presented in table 2. It can be observed that for every

combination of images and filters, the value of the PNSR decreases when increasing the filter size and thus the best size is 3×3.

		3×3	5×5	7×7
image1_saltpepper.jpg	Box	23.39	22.64	21.42
	Median	**27.68**	24.49	22.37
image1_gaussian.jpg	Box	**26.23**	23.66	21.94
	Median	25.45	23.79	22.07

Table 2: PNSR for various denoised images

When denoising *image1_saltpepper.jpg* it can be observed that the 3×3 median filter leads to the best qualitative result. This is due to the random distribution of defective pixels as one bad pixel in a neighborhood does not affect the median result significantly.

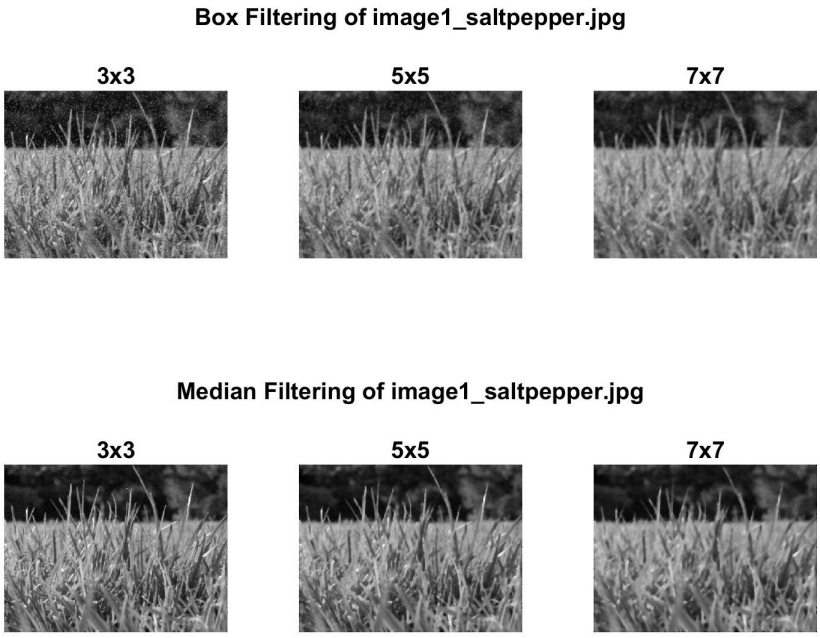


Figure 3: Denoised *image1_saltpepper.jpg*

In contrast, *image1_gaussian.jpg* reacts better to the box filter as the Gaussian distribution has a mean of 0. The difference between the 3×3 median filter and the 3×3 box filter is not that distinctive, but the box filtered result has a higher PNSR value than the median one.

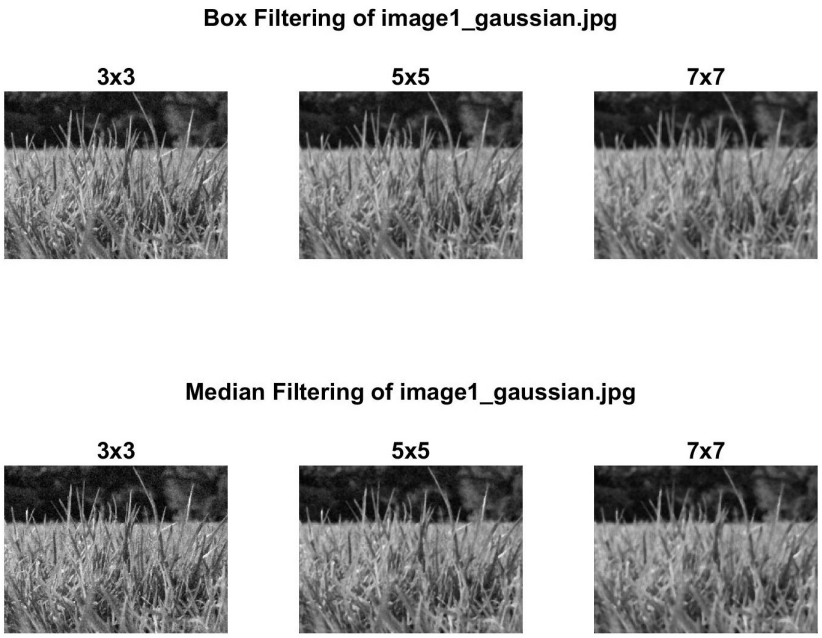


Figure 4: Denoised *image1_gaussian.jpg*

To apply an appropriate Gaussian filter to *image1_gaussian.jpg*, it is necessary to analyze different standard deviations and window sizes. To that end, consider $\sigma = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$ and the standard window-sizes 3×3, 5×5 and 7×7.

$\sigma =$	3×3	5×5	7×7
0.1	20.58	20.58	20.58
0.2	20.58	20.58	20.58
0.3	20.71	20.71	20.71
0.4	22.02	22.02	22.02
0.5	24.28	24.29	24.29
0.6	25.89	25.93	25.93
0.7	26.54	26.58	26.58
0.8	**26.69**	26.66	26.65
0.9	26.67	26.46	26.44
1.0	26.60	26.15	26.09

Table 3: PNSR for a gaussian filter with various sigmas and window sizes

According to the PNSR values from table 3, an appropriate choice of parameter is $\sigma = 0.8$ with a window-size of 3×3. Figure 5 presents the resulting image which is clear and detail-preserving, proving that the standard deviation and the window-size have indeed been properly chosen.

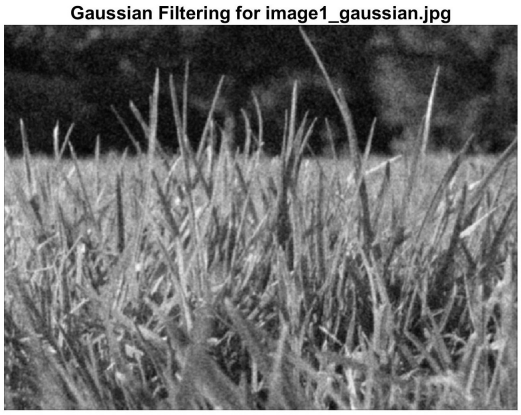


Figure 5: 3×3 Gaussian filtering with $\sigma = 0.8$

Analyzing the PNSR values from table 3, it can be observed that when the standard deviation is small, between 0.1 and 0.4, the Gaussian filter does not have an obvious effect. Because the standard deviation is small, the neighboring pixels around a point are assigned weights ≈ 0. When the standard deviation increases, the Gaussian filter assigns non-uniform weights and thus the effect of the filter can be noticed. The PNSR increases along with the standard deviation until it peaks at $\sigma = 0.8$. As we are only considering three window-sizes, the PNSR will decrease if $\sigma > 0.8$ and the result will be similar to that of the box filtering. If we considered larger window-sizes, the PNSR would have peaked at a greater standard deviation value.

The **median** filter replaces the value of a pixel with the median of neighboring pixel values and it is *non-linear*. The **box** filter is *linear*, *uniform*, replacing the value of a pixel with the mean of the neighborhood. The **gaussian** filter is *linear*, but *non-uniform* as it uses the weighted average of neighboring pixels. Similar PNSR values of different filtering methods do not imply the same quality due to the human eye and how humans perceive images.

4.3 Edge detection

4.3.1 First-order derivative filters

Question 8. Implementing the `compute_gradient` function, the following results are obtained:

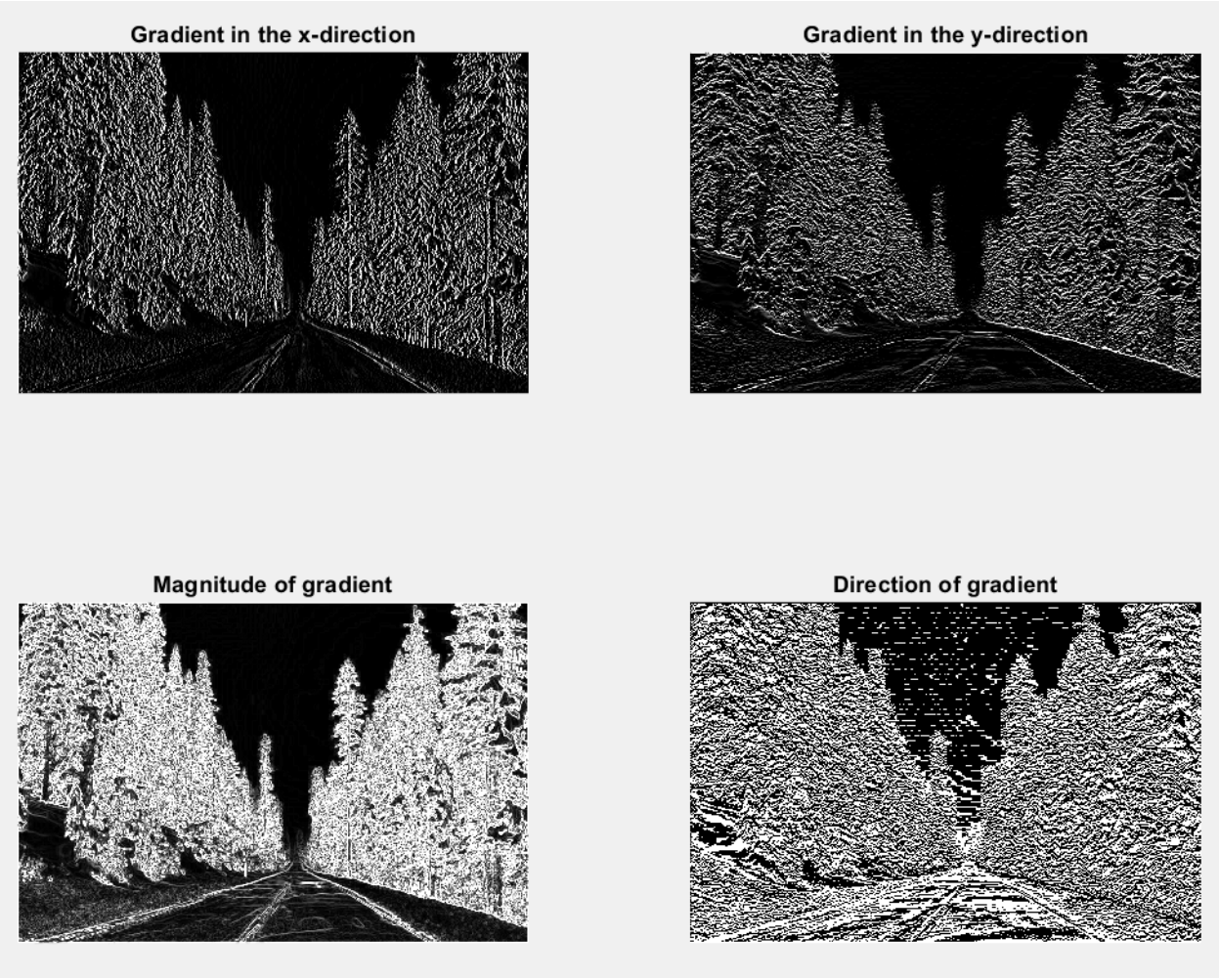


Figure 6: Caption

From the computed figures, the following information can be derived:

- The gradient of the x-direction, or G_x, holds information about the vertical edges of the objects in pictures,
- The gradient of the y-direction, or G_y, holds information about the horizontal edges of objects,
- The magnitude of the gradient is approximately the sum of the magnitudes of G_x and G_y and corresponds to the rate of change in the direction of the gradient vector G at each image point (the height of the edges) and
- The direction of the gradient holds information about the orientation of image points.

4.3.2 Second-order derivative filters

Question 9 1. Figure 9 shows the results after applying methods 1, 2 and 3 on the original image.

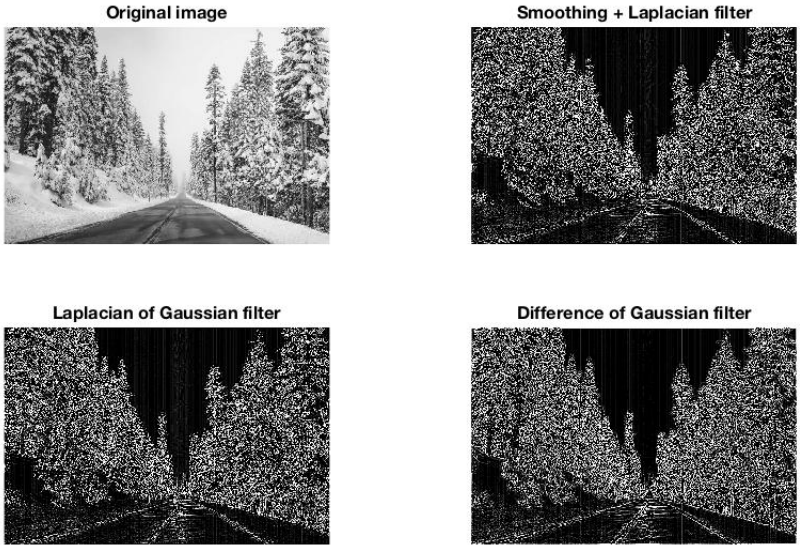


Figure 7: Second-order derivative filters

2. In the first method we firstly smooth the image with a Gaussian then take the Laplacian of the Gaussian-filtered image. In the second method we take the Laplacian of a Gaussian and apply it on the original image. By the associativity property of these filters, we know that these 2 cases are identical. The difference of Gaussians appied in the third method represents an approximation of the Laplacian of Gaussian. Because it consists of a difference of two filters, DoG is tunable by adjusting the ratio between the two sigmas used in the computation.
3. When we take the derivatives of images, the high frequencies and the noise are amplified, this is why smoothing the image with a symmetric, circular kernel prevents this from happening. Gaussian is the only separable such filter, thus it is used very often in the process of edge detection.
4.The best ration between σ_2 and σ_1 to achieve the best approximation of the LoG is theoretically around 1.6 [2]. The purpose of having 2 standard deviations is to obtain two images with a different level of blur. Then, when they are subtracted, the spacial information found between the range of frequencies of the two blurred images is preserved. Thus the details of the image are preserved, while the unnecessary noise is removed. We have used the mean squared error to compute the best approximation for the LoG. We have taken $\sigma_1 = 0.5$ and $\sigma_2 = 0.5 \cdot i$, where $i \in [1 : 0.1 : 8]$. As we can see in Figure 8, the MSE starts to flatten around 1.6.

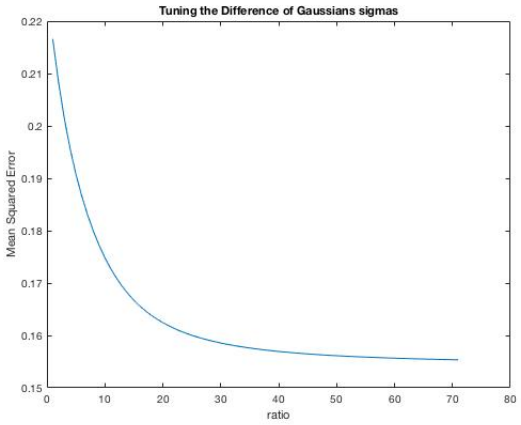


Figure 8: Tunning the sd ratio in DoG

5.In order to improve the performance and isolate the road, we can try to use the Sobel filter, because this

focuses on smaller gradients in the image, hence it may detect the difference the difference in brightness on the sides of the road. We could also try to use a mask for the road(based on the results of a image segmentation algorithm and applying), before applying any of the three second-order derivative filter described in 4.3.2

4.4 Foreground-background separation

1. We observe that almost all the dog is separated from its background. Only a small part of the dog's tail and foot is left out as we can see in left picture. Furthermore a small area of the background is kept together with the dog. The only part that is a bit unexpected is a very small part in the top right corner is considered to be in the same cluster as the dog. Overall, we think this is a good segmentation.

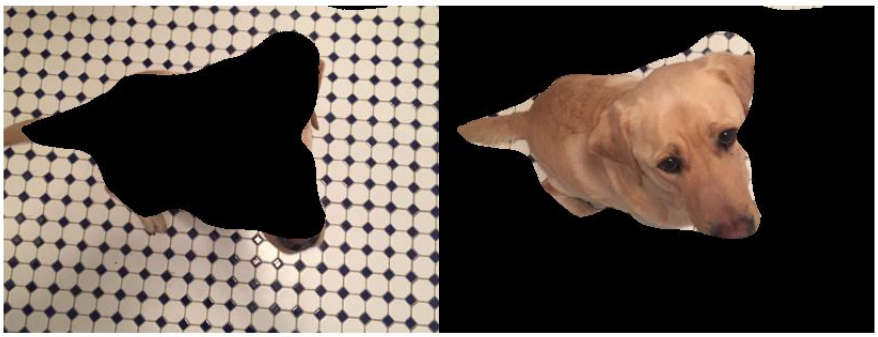


Figure 9: Results for the first picture

2.In table 4(on the following page) are in our opinion the best results for each picture that we observed. Compared to the defaul parameters, we only changed the set of sigmas for the Gaussian envelope to obtain better results, since we observed that if we change λ or θ we don't get considerably better results. We think that changing the set of sigmas for the Gaussian envelope for each picture helps, because each picture has different variations of brightness on the edges of desired segmentation.

3.If we set smoothingFlag to false, then we observe we have to change the parameters again(now we observed greater variation while also changing θ. But, the main observation is that the pixel cluster are not a continuous area of the picture. For example, in the picture with the bird we can observe the most of pixels that form the bird are in the cluster, but divided in many smaller clusters. Since the effect of changing the value of this variable is that now the smoothing is not applied to the magnitude of the picture, also the segmentation suffers since it focus on a smaller scale due this smoothing.

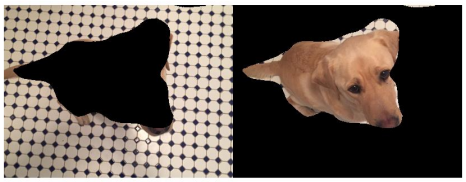
5 Conclusion

We have analysed interesting properties of images in each discussed section, compared some popular algorithms and investigated the most important applications that result from the image processing. We also applied different algorithms in order to smooth, filter or denoise the images. In the last section, we also used a clustering algorithm together with image processing algorithms to realize foreground-background separation.

References

[1] `https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97`

[2] Singh, S. (2005). Pattern recognition and image analysis proceedings. Berlin: Springer.



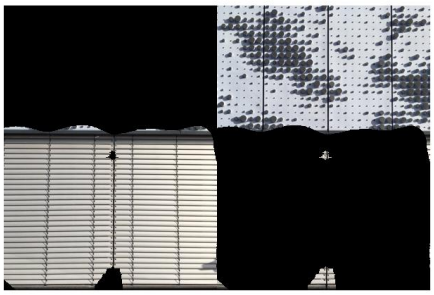
(a) Figure 1 $sigmas = [1, 1.4]$

(b) Figure 2 $sigmas = [1, 1.4]$

(c) Figure 3 $sigmas = [1, 1.35]$

(d) Figure 4 $sigmas = [1, 3]$

(e) Figure 5 $sigmas = [1, 4.5]$



(f) Figure 6 $sigmas = [1, 5]$

Table 4: Best result for each picture