Assignment 2 - Intelligent Multimedia Systems

Rémi de Zoeten (6306894) and Anouk Visser (6277209)

November 21, 2013

1 Introduction

Our assignment was to use Gaussian image filters and Gaussian derivatives to manipulate images. This can be used to find edges in images and an orientation of those edges.

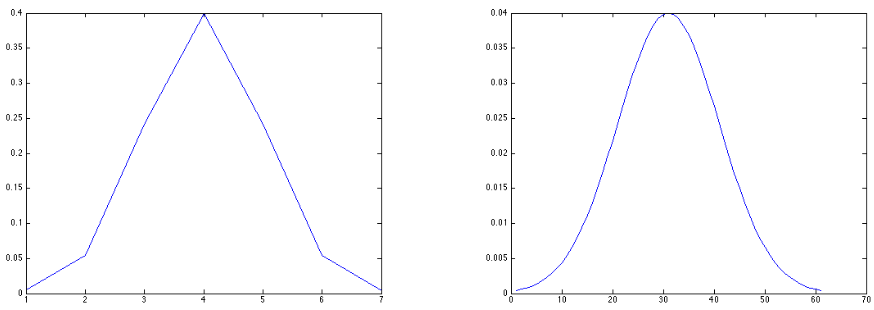
2 Results

2.1 Gaussian Convolutions

We implemented the function *gaussian* that produces a 1D vector that can be used to convolve an image. When called, *gaussian(1)* will produce the following vector:

$$[0.0044, 0.0540, 0.2420, 0.3991, 0.2420, 0.0540, 0.0044]$$

When plotting the resulting vector using MATLAB's *plot* it is clear that the function indeed produces a Gaussian. The plots can be seen in figure 1.



(a) (b)

Figure 1: (a) shows the plot of the result of *gaussian(1)*. (b) shows the result of the same function with $\sigma = 10$.

Using the function *gaussian* we can apply a Gaussian filter in both the x and y direction to an image. In figure 2 we can see the results for two different combinations of sigma.

(a)

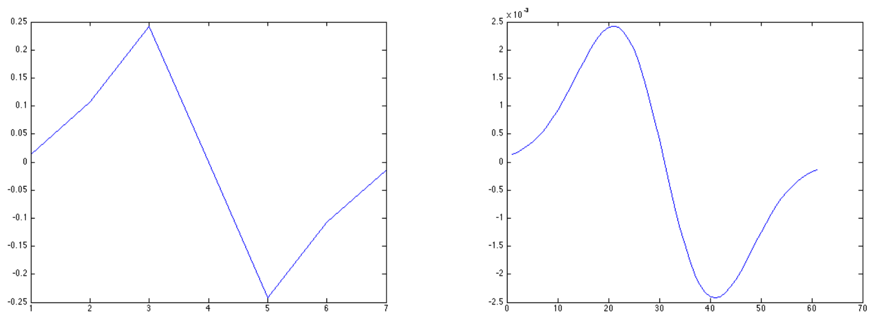
(b)

Figure 2: Results of convolving an image along the x-axis and the y-axis separately with a 1D Gaussian as produced by *gaussian*. (a) shows the result obtained by using a 1D Gaussian with $\sigma = 5$ in both directions. (b) shows the result obtained by using a 1D Gaussian with $\sigma = 10$ in both directions.

When using MATLAB's built-in functions to perform a Gaussian convolution with $\sigma = 10$ (MATLAB convolve the image with a 2D Gaussian filter), then the results are not numerically identical. However, the difference is not visible and the sum of the differences per pixel is 1.7433e-10 and the average difference per pixel is 6.6503e-16.

2.2 Gradient Magnitude and Orientation

We implemented the function *gaussianDer* that produces a 1D vector that can be used to convolve an image. When plotting the resulting vector using MATLAB's *plot* we see the shape of the expected Gaussian derivative. The plots can be seen in figure 3.



(a) (b)

Figure 3: (a) shows the plot of the result of *gaussianDer* with $\sigma = 1$. (b) shows the result of the same function with $\sigma = 10$.

Using the derivative of the Gaussian we can determine the the magnitude and orientation of the gradient for each pixel of an input image. Figure 4 shows the results for three different values of sigma.

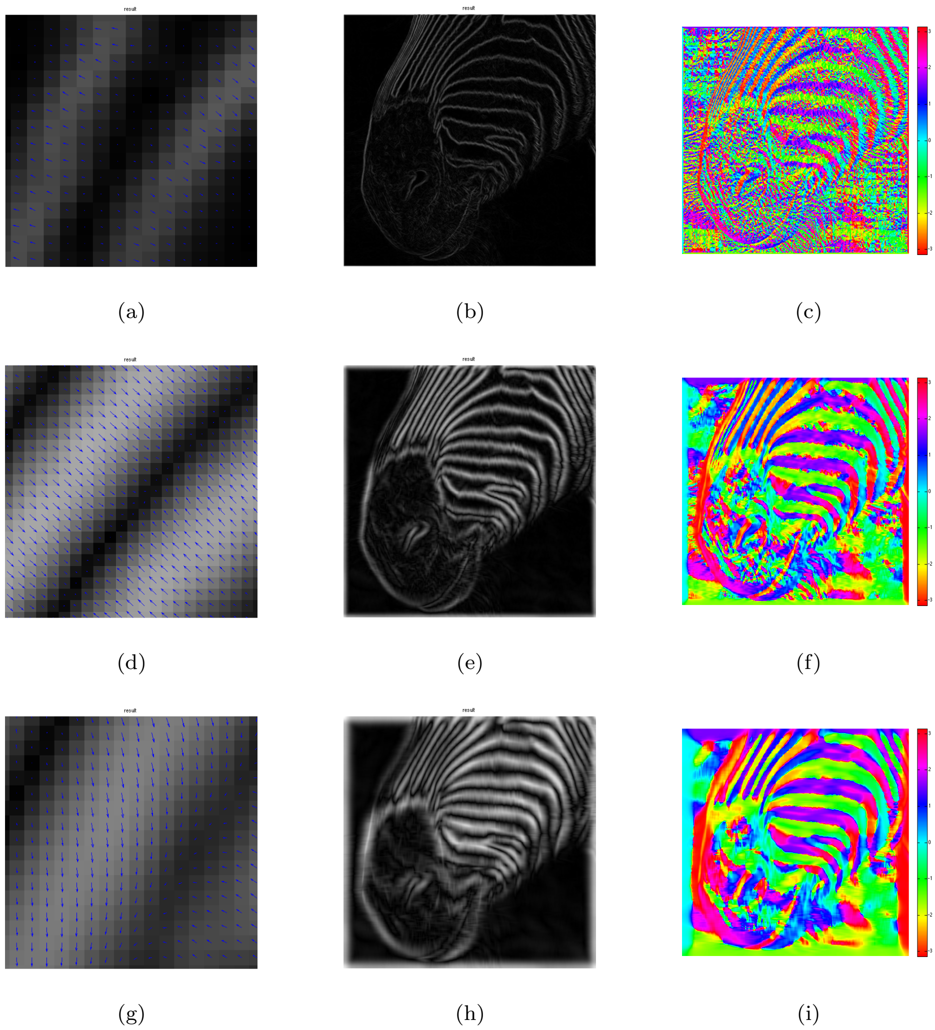


Figure 4: The images were sampled using the following sigma values: a-c: $\sigma = 1$, d - f: $\sigma = 5$, g-i: $\sigma = 10$. The magnitude images become more clear or white as the sigma increases. The colored orientation images become more regular as the sigma increases, because small local variances even out when a larger sigma is used.

We have applied a binary filter to the magnitude images for the sigma's $\{1, 5, 10\}$ and threshold values $\{0.025, 0.02, 0.013\}$. The results of this can be viewed in figure 5.

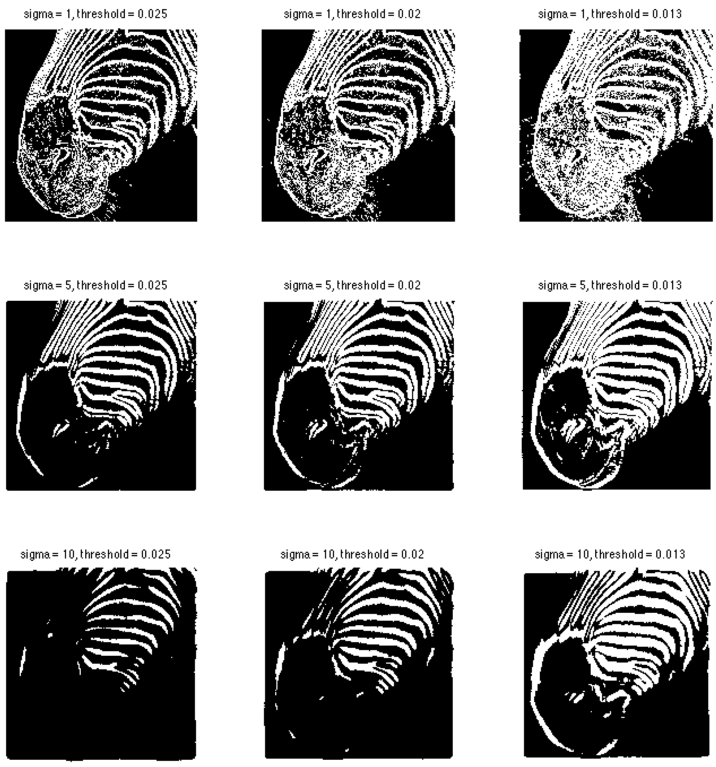


Figure 5: Binariry magnitude images. Choosing an appropriate threshold value is dependent on the sigma that is used. A larger sigma requires a lower threshold.

2.3 Second Order Gaussian Derivative

We implementend the function *gaussianDer(G,sigma)* and used it along with *gaussian(sigma)* to implement the function *ImageDerivatives(img,sigma,type)*. The following figures show images that were created using the *ImageDerivatives* script and $\sigma = 20$. The input image (figure 8) was a black 201×201 pixel image with the center pixel white.

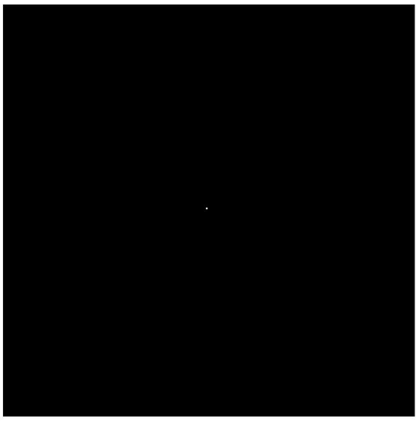


Figure 6: Impulse image.

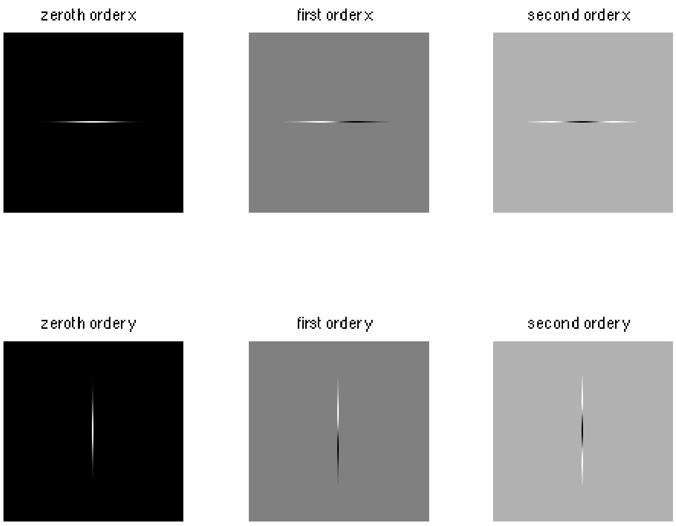


Figure 7: Zeroth, first and second order Gaussian derivatives of the impulse image shown separately in the x and the y direction.

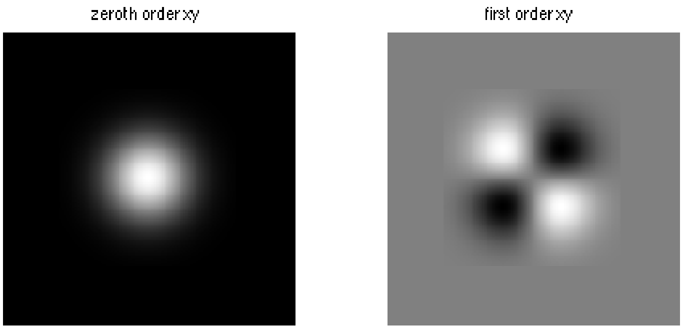


Figure 8: Zeroth and first order Gaussian derivatives of the impulse image applied in both the x and the y direction.

3 Conclusion

In this assignment we have implemented a 1D discrete Gaussian filter and used it to convolve it with an image in the x and y direction separately, resulting in a blurred image. We have shown that using a separate kernel we obtain the same results as the built-in MATLAB functions, which use a 2D Gaussian filter.
Using the first order Gaussian derivative and convolving this with the image, we have created images showing the image's gradient and magnitude for each pixel. When a threshold is applied to the magnitude image, we obtain a binary image showing the edges that depend on the threshold chosen.
Finally, we applied different 1D Gaussian derivative filters on an impulse image.