Computer Vision: Lab 2

Thomas de Groot 11320303
Selina Blijleven 10574689

February 2017

Todo list

2 More on Filters

2.1 Gaussian Versus Box

A **denoise** function in MATLAB was implemented to denoise a given input image. It uses either box filtering or median filtering.

(a) Box filtering was done with kernel sizes 3x3 (Figure 1), 5x5 (Figure 2), 7x7 (Figure 3) and 9x9 (Figure 4). Median filtering was done with kernel sizes 3x3 (Figure 5), 5x5 (Figure 6) and 7x7 (Figure 7).

(b) Smoothing uses the values of neighbours to smoothe an image. The amount of neighbours can be influenced by altering the filter size. A small filter size blurs mostly the pixels around itself and is a very local smoothing method. This might cause very small details, or noise, to disappear. A big filter size smoothes over the full image and causes more local information to disappear. The focus will be on the global structure in the image.

(c) Box filtering uses the average of the neighbourhood, while median filtering uses the median. When de-noising an image it is important to preserve the edges, which are very important for the visual appearance of an image. Because box filtering uses the average, both sides of the edge are factored in, as well as noise. This causes the edges to blur and very light-coloured noise, for example, might alter the colour of the image as a whole. Median filtering performs better because it uses the median, which may be either side of the edge. Very contrasting noise is not factored in either and this makes the algorithm very efficient at removing speckle noise.

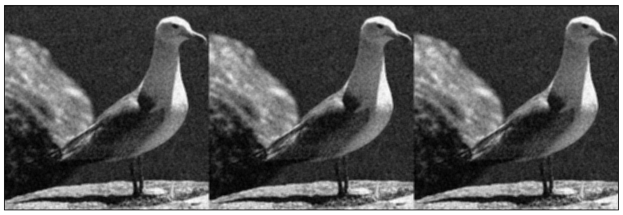


Figure 1: Box filtering 3x3

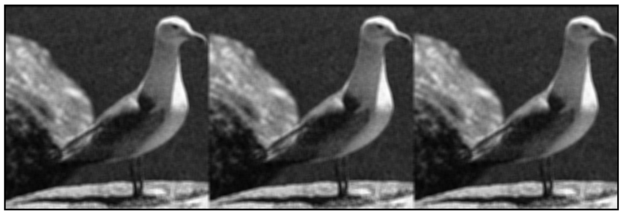


Figure 2: Box filtering 5x5

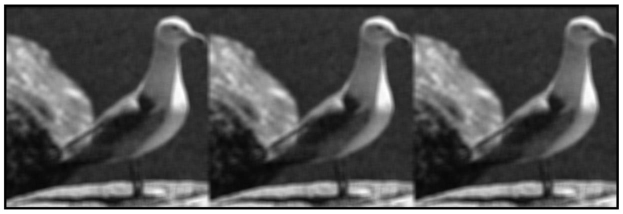


Figure 3: Box filtering 7x7

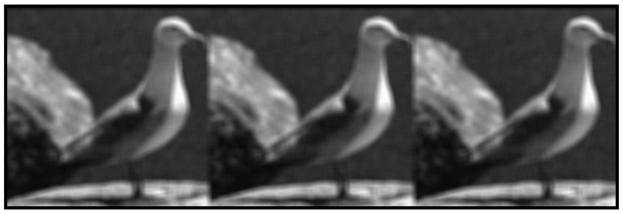


Figure 4: Box filtering 9x9

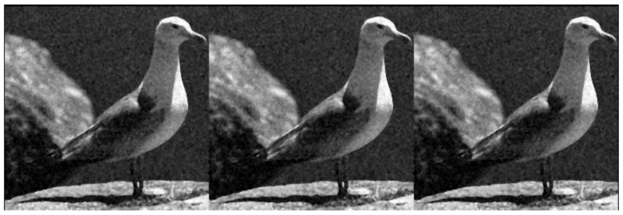


Figure 5: Median filtering 3x3

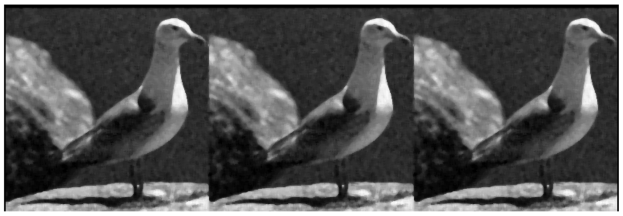


Figure 6: Median filtering 5x5

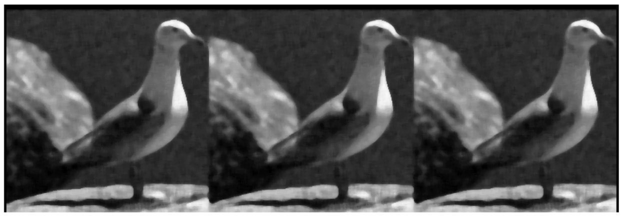
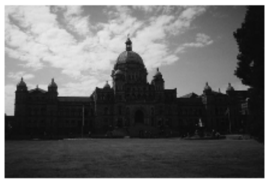


Figure 7: Median filtering 7x7



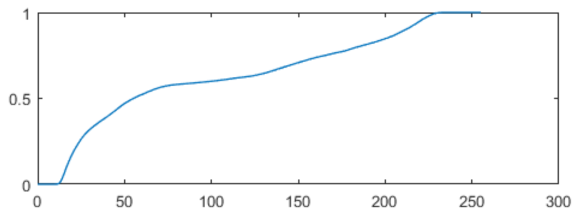


Figure 8: Input image and corresponding histogram

2.2 Histogram Matching

A **myHistMatching** function was implemented to transform an input image (input.png) such that it has the same histogram as the reference image (reference.png). It outputs 3 figures:

(1) Input image and corresponding histogram. (Figure 8)

(2) Reference image and its corresponding histogram. (Figure 9)

(3) Transformed input image and corresponding histogram. (Figure 10)

2.3 Gradient Magnitude and Direction

A **compute_gradient** function was implemented in MATLAB that computes the magintude and the direction of the gradient vector corresponding to a given image (image_3.jpeg in this case). It uses the Sobel kernel to approximate the 2D derivative of an image. It outputs 4 figures:

(1) The gradient of the image in the x-direction. (Figure 11)

(2) The gradient of the image in the y-direction. (Figure 12)

(3) The gradient magnitude of each pixel. (Figure 13)

(4) The gradient direction of each pixel. (Figure 14)

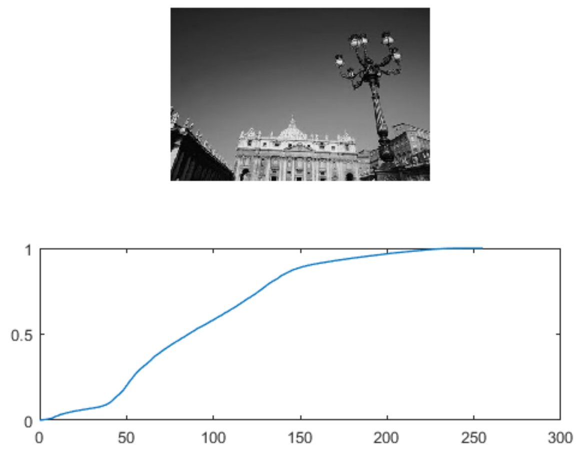


Figure 9: reference image and corresponding histogram

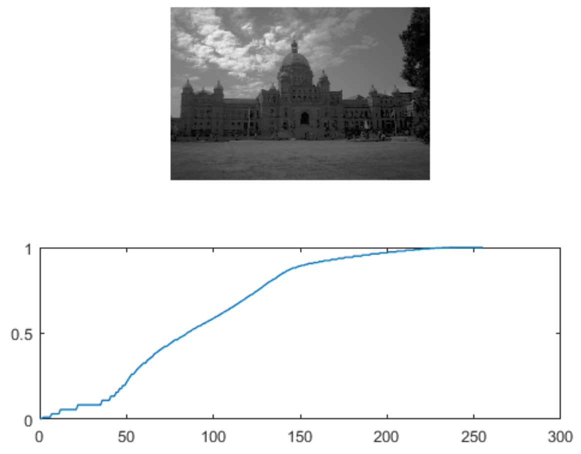


Figure 10: output image and corresponding histogram

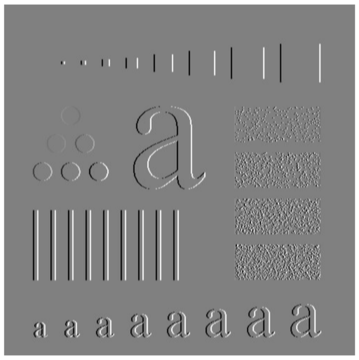


Figure 11: Gradient of the image in the x-direction.

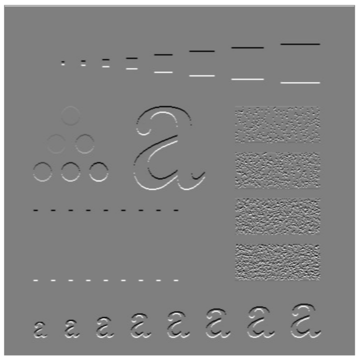


Figure 12: Gradient of the image in the y-direction.

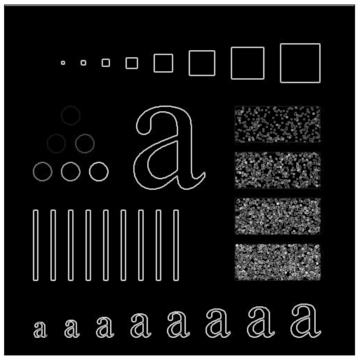


Figure 13: Gradient magnitude of each pixel in image3.jpeg

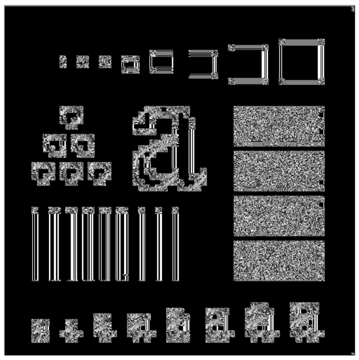


Figure 14: Gradient direction of each pixel in image3.jpeg

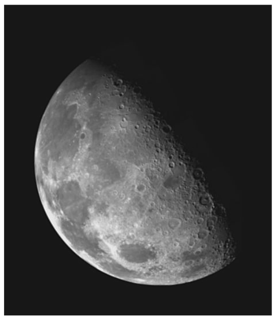


Figure 15: Unsharp masking with sigma = 1 , kernel size = 5 and k = 1

2.4 Unsharp Masking

A function named **unsharp** was implemented that highlights the image's fine details. It smoothes the original image with a Gaussian kernel of a given size and Sigma. By subtracting the smoothed image from the original image it obtains a high-passed version, which should be strengthened k (weighting factor) times and adding it to the original image. All of this will be done on image4.jpeg.

(a) Testing the function resulted in Figure 15.

(b) The weighting factor k influences the amount of contrast that is added. Using k = 1 adds contrast to the edge borders from the mask. Using k >1 amplifies this effect and makes the edge borders lighter and darker to emphasize the difference even more.

Possibly add more figures to strengthen explanation.

(c) The kernel size affects the radius of the pixels used in smoothing. A small radius emphasizes smaller scale detail while a high radius emphasizes larger structures and may cause halos at the edges. In this case a high radius causes a large ring to appear under the moon. The intensity of the edges is then affected by applying the weighting factor.

Possibly add more figures to strengthen explanation.

(d) Unsharpened masking is a way to seemingly improve the resolution of an image. This is done by using a smoothed image to create a mask, which can be applied a given amount of times to make the image less blurry. This, however, does not necessarily create a more accurate picture. Because it ignores low frequency information the contrast within the image becomes higher. It can be used to make a picture seem of higher quality or to discover small or large patterns that were hard to see in the original.

2.5 Laplacian of Gaussian

A function named **compute_LoG** was implemented that performs the Laplacian of Gaussian following three possible methods:

- Smoothing the image with a Gaussian operator, then taking the Laplacian of the smoothed image.
- Convolving the image directly with LoG operator.
- Taking the difference between two Gaussian (DoG) computed at different scales σ_1 and σ_2.

(a) Testing the function with all three methods resulted in Figures 16, 17 and 18 in the same order as described above.

(b) The first method uses the Gaussian operator and takes the Laplacian of the smoothed image. The second method is a hybrid filter which exploits the associative property of convolution. By pre-calculating the LoG kernel only one convolution is required to transform the full image and is therefore the least costly. The third method, DoG, uses the Gaussian operator twice to approximate the LoG. This method is more costly than the direct approach, but very similar.

(c) Images often contain noise, which is blurred with the Gaussian. Applying the Laplacian before the Gaussian would result in a blurred version of the intensity changes from a noisy image. However, in this case, the noise from the image is reduced with the Gaussian and intensity changes are detected in the filtered image.

(d) The higher the standard deviation, the more blurred an image will be. The approach in this method is to subtract a blurred image from a less blurred image. The goal is to identify the spatial information that lies between the frequencies of the images. Since the first image using σ_1 should be less blurred the ratio σ_1:σ_2 should be <1.

(e) This method uses smoothing to reduce noise and detect rapid intensity changes. These intensity changes are useful for edge detection, which is useful for identifying objects through the computer. A robot could use edge detection to find objects or avoid collision, for example.

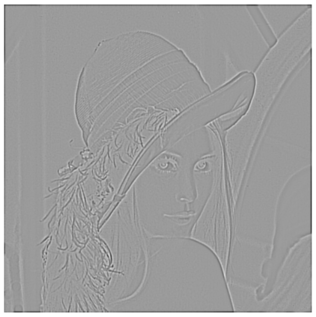


Figure 16: LoG with Laplacian of image smoothed by Gaussian.

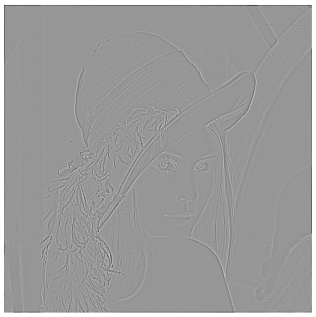


Figure 17: LoG with direct LoG operator.

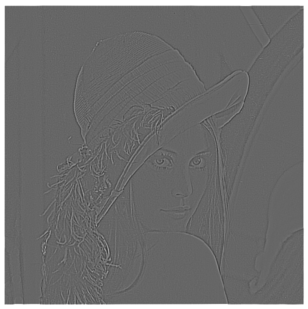


Figure 18: LoG with difference between two Gaussians at different scales σ_1 and σ_2.