UNIVERSITY OF AMSTERDAM

Assignment 2 Report

Luca Simonetto - 11413522
Heng Lin - 11392533
Computer Vision 1

February 27, 2017

Median versus Box filter

Having implemented both filters, the first using the median of the surrounding pixels and the second calculating their average value, we used them to produce the results shown in Figure 1 and 2.

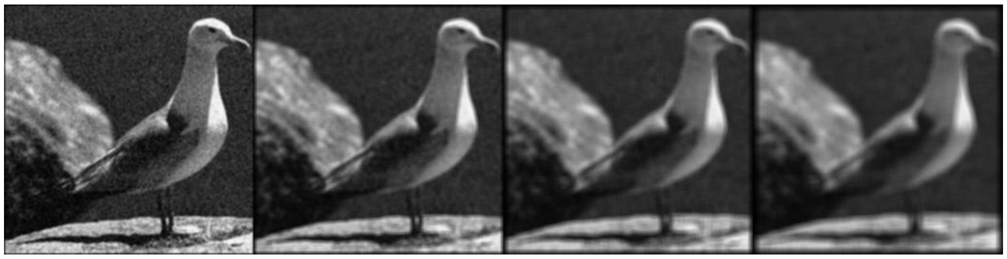

(a) 3x3 (b) 5x5 (c) 7x7 (d) 9x9

Figure 1: Box filter with different sizes

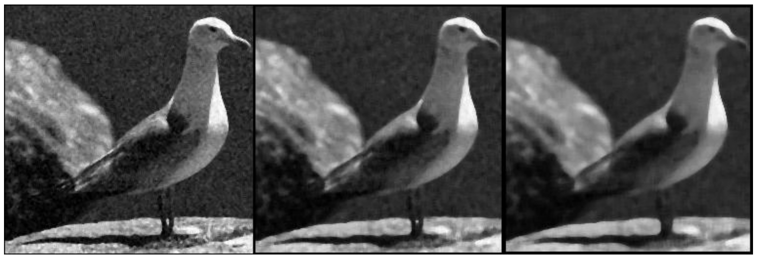

(a) 3x3 (b) 5x5 (c) 7x7

Figure 2: Median filter with different sizes

We can see that each filter behaves differently when the kernel size is increased: in the case of box filter we have an uniform blur that decreases the quality of the image as the filter is enlarged, while with the median filter this effect is more contained, and if the scope of the application is noise reduction then the median filter is to be considered superior. Even with a 7x7 median filter, we can see that the edges are kept sharp enough and other details in the image are still visible, whereas box filtering degrades the image even at 5x5 pixel.

Histogram matching

This method tries to match the histogram of the input image with the one of a reference image, by analyzing and altering the shape of the cumulative distribution function of the histogram. In this exercised we tested the results with two images, the input shown in Figure 3 and the reference shown in Figure 4.

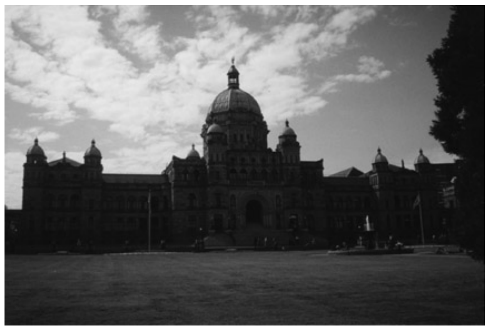


Figure 3: Input image

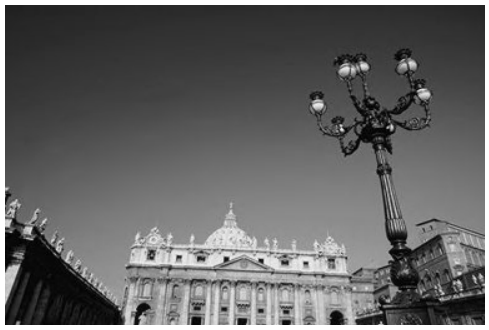


Figure 4: Reference image

The first step is to extract the histogram of each image (Figure 5 and 6), and calculate the cumulative distribution function as shown in Figure 7 and Figure 8.

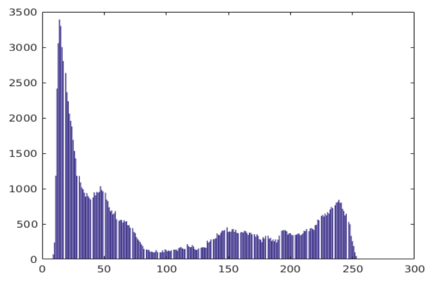


Figure 5: Input image histogram

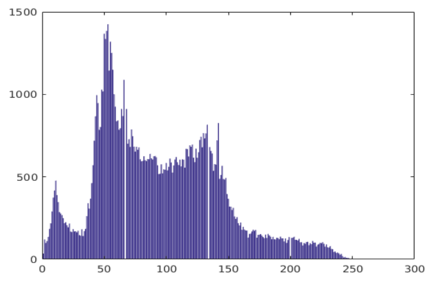


Figure 6: Reference image histogram

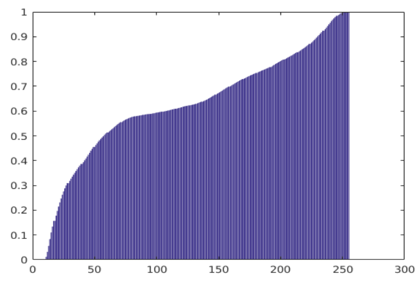


Figure 7: Input image cdf

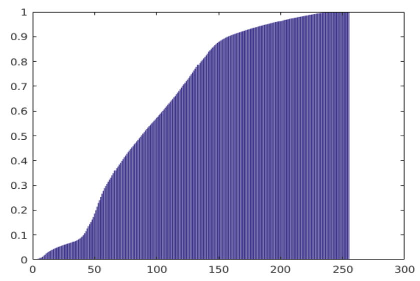


Figure 8: Reference image cdf

The objective of the algorithm is to match the input cumulative distribution function as close as possible to the one in reference image, by moving (not swapping) the bins of the histogram to the best position. Mathematically, this procedure approximates a function of the reference image with another function that, when applied to the first image histogram, produces the effect of matching the brightness of each pixel to the desired value. The result of this application is shown below, in Figure 9, 10 and 11.

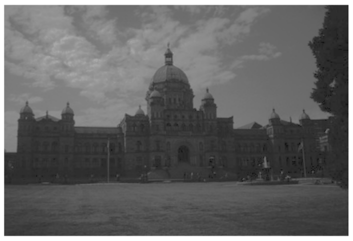


Figure 9: Result image

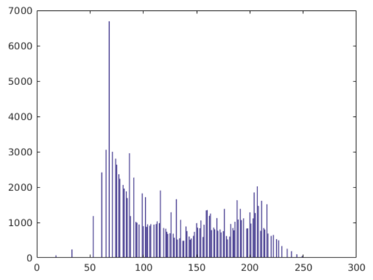


Figure 10: Result image histogram

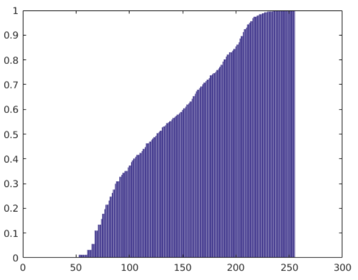


Figure 11: Result image cdf

Gradient magnitude and direction

In this exercise we used the Sobel kernel for edge detection on a given image. The kernel has the property of approximating the horizontal and vertical first derivatives of an image G_x and Gy, by convolving the kernel to the pixel matrix A.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} A$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} A$$

In order to compute the gradient magnitude G and direction θ for each pixel, having calculated both G_x and G_y, we used the following formulas

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

Given as input the image shown in Figure 12, the values for G_x, G_y, G and θ can be seem in Figure 13 to 16. Some of the results present a gray color, indicating that the pixel values have been normalized in order to take into account out of bounds values.

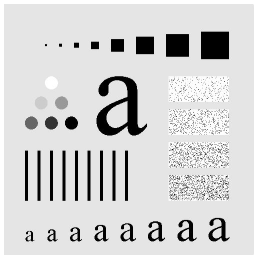
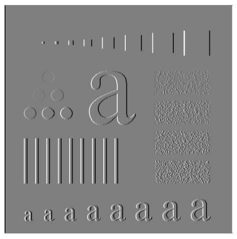
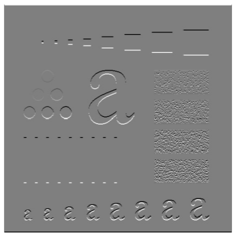
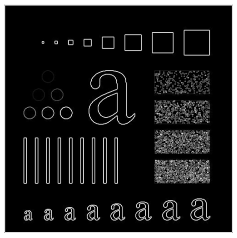


Figure 12: Input image







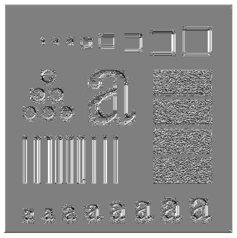


Figure 13: Gx Figure 14: Gy Figure 15: magnitude Figure 16: direction

Unsharp masking

This task revolves around the sharpening of an input image edges, by first convolving it with a Gaussian filter and then using the high pass version of the input in order to remove low frequencies from around the edges. This has the effect of effectively sharpening the image, as shown below. First, chosen the parameters for the Gaussian kernel (in our case sigma = 1 and kernel_size = 5), we applied a 2D convolution to get a blurred version of the input image. In order to get the high pass version of the image we then subtract the blurred image from the original one: the resulting picture shows only the most prominent edges of the image, and is enhanced by multiplying it by a factor k. Adding the resulting image to the original input results in a sharpened picture, as shown in Figure 17, 18, 19 and 20.

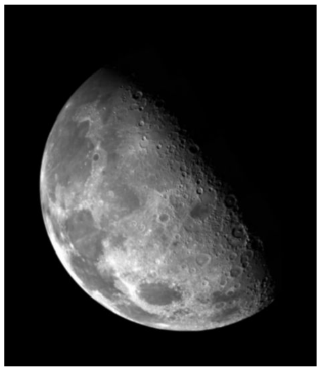


Figure 17: Input image

The difference between k = 1 and k > 1 can be seen in Figure 21 and Figure 22. The value of k alters the intensity of the high-passed image, which is the image containing features like edges, therefore higher value of k result in sharper edges in the image.

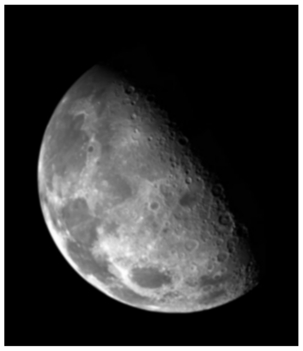


Figure 18: Gaussian filtering

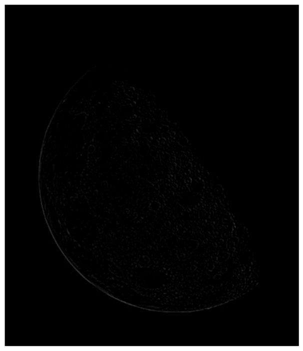


Figure 19: High pass

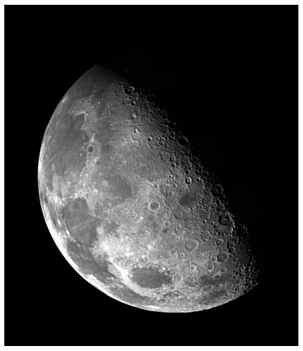


Figure 20: sharpened result

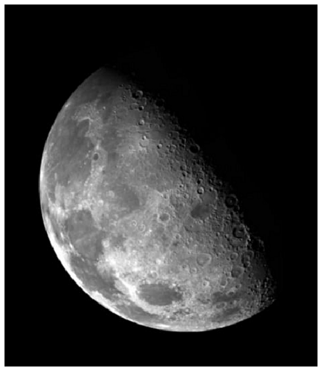


Figure 21: k = 1

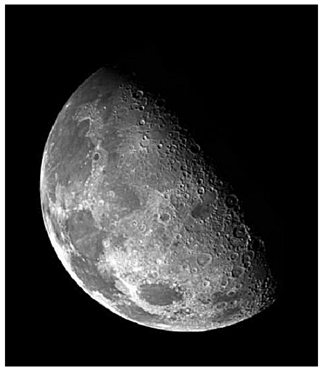


Figure 22: k = 5

The effect of changing the kernel size of the Gaussian filter can be seen in Figure 23 and Figure 24, where kernel size of 3 and 9 were applied. It can be seen that the kernel size influence the resulting image in a way that the larger the kernel size, greater range in the high frequency component will be removed from the image. Therefore, we end up removing less low frequency component when subtracting the low-passed Gaussian blurred image from the original image, which means greater high frequency pixels remains and makes the edges not as obvious as the case of using smaller kernel size. As a result, the image used kernel size of 9 is more blurred when comparing to the image used kernel size of 3.

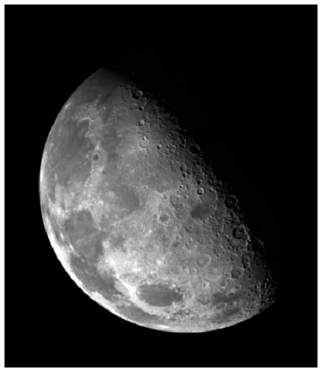


Figure 23: kernel size = 3

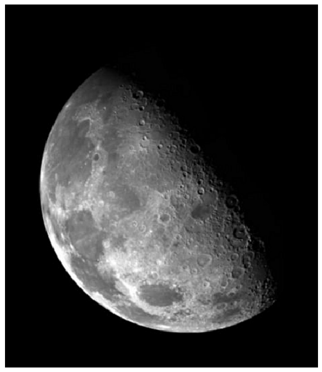


Figure 24: kernel size = 9

The most common application of unsharp masking is to sharpen an image. It is often used to correct the sharpness of a digital image that is out of focus.

Laplacian of Gaussian

In this exercise, we applied three different methods to experiment with the Laplacian of Gaussian filters. We used a kernel size of 3 for all three different methods. The first method is applying a Gaussian filter with sigma of 0.8 prior to the Laplacian filter, and the result is shown in Figure 25. The second method is convolving the image directly with the Laplacian operator and the result is shown in Figure 26. The third method is applying Gaussian filter of different variance to the same image, such that the sigmas were 0.75 and 1.2, then take the difference between them, and the result is shown in Figure 27.

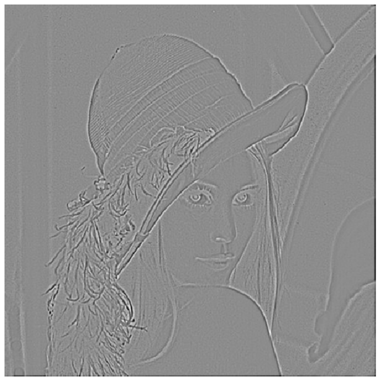


Figure 25: Gaussian and Laplacian

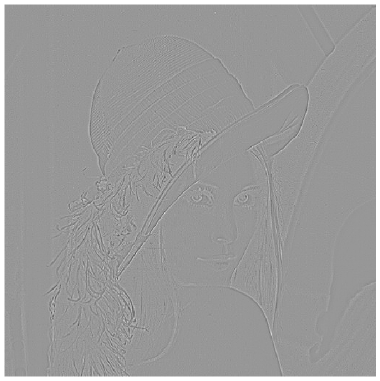


Figure 26: Laplacian

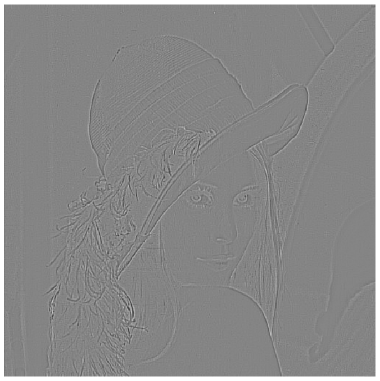


Figure 27: Difference of Gaussian

The difference between the first two methods is the amount of noise in the image. It can be seen that noises are more obvious in the image filtered directly with Laplacian operator when comparing to image firstly filtered with a Gaussian filter. The differences between Laplacian of Gaussian and Difference of Gaussian is not very obvious in this case.

It is important to apply a Gaussian filter to smoothen the image prior to applying the Laplacian filter when considering the Laplacian equation, $L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$, where I is the intensity. It can be seen that Laplacian equation is a second order differential equation, and this implies that any noises in the image will be leveraged as a result of the derivatives.

The best ratio for the Difference of Gaussian is determined according to Figure 28. The value of sigma for each of the two Gaussian is obtained by multiplying and dividing the sigma used in the Laplacian, by the value of k. It can be seen the Difference of Gaussian approximates better when the value of k is closer to 1.

One of the common application of Laplacian of Gaussian is edge detection in an image. The further application edge detection includes image enhancement, and this is done by applying a Laplacian of Gaussian to an image to acquire the edges, then combine with the original image to obtain the enhanced image. The original image can be seen Figure 29, and the image sharpened with application of Gaussian filter can be seen in Figure 30, and the image sharpened with direct application of Laplacian in Figure 31, and image sharpened with Difference of Gaussian in Figure 32. This also shows that the sensitivity to noise is significantly larger in the direct of application of Laplacian in Figure 31 in comparison to the image smoothened with Gaussian in Figure 30.

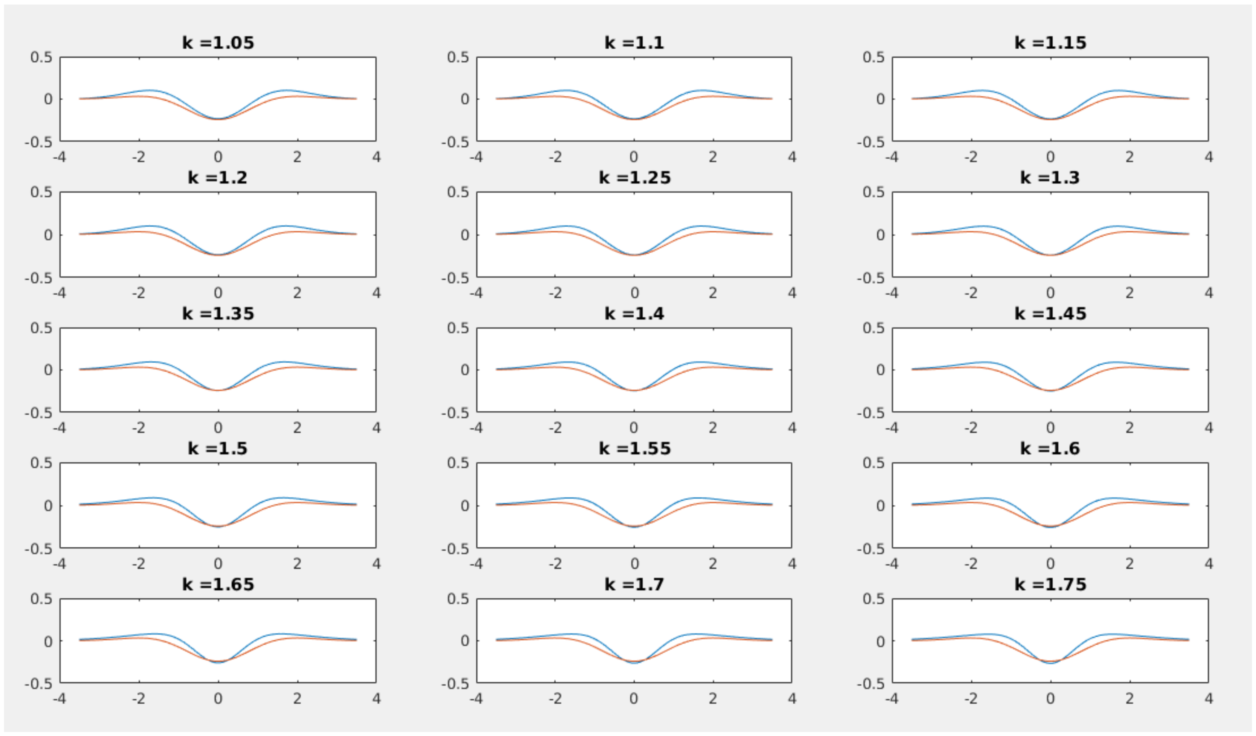


Figure 28: Approximating DoG to Laplacian

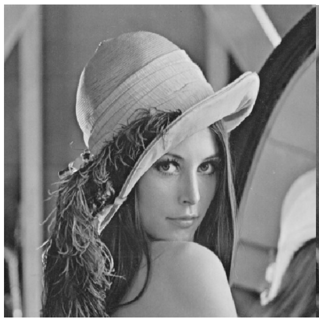


Figure 29: Original

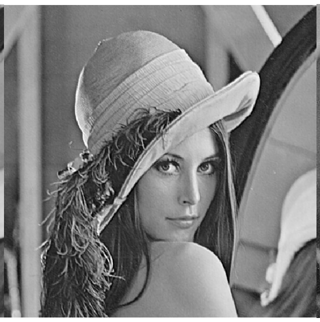


Figure 30: LoG

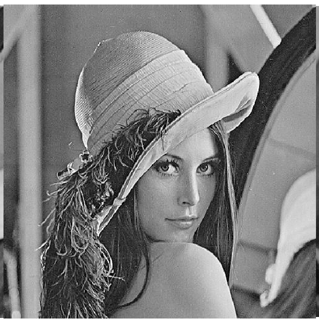


Figure 31: Only Laplacian

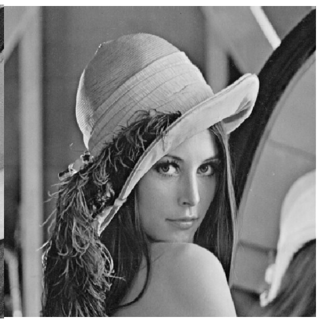


Figure 32: DoG