Computer Vision Laboratory Report n.2

IMAGE FILTERING AND FOURIER TRANSFORM

April 1, 2019

Claudio Curti ID: 4216203
Nicola De Carli ID: 4198668
Alberto Ghiotto ID: 4225586
University of Genoa

Contents

1 Introduction **2**
1.1 Defining the objective 2

2 Theoretical Background **3**
2.1 Image noise 3
2.2 Noise reduction 3
2.3 Linear filtering 4
2.4 Fast Fourier Transform 4

3 Implementation and Results **5**
3.1 Noise Adding 6
3.2 Noise Filtering 7
3.3 Linear filter examination 18
3.4 Fast Fourier Transform Evaluation 20

Chapter 1

Introduction

1.1 Defining the objective

This project aims to get a solid grasp of the effects produced by two different types of noise and the possible countermeasures that the image filtering procedure provides. In addition, in order to gain a broad perspective of the various existing types of filters, some experimentation with linear filters was conducted. Eventually the Fourier Transform was examined by evaluating the magnitude of the transformed images and the magnitude of a transformed low pass Gaussian filter.

Figure 1.1: Image considered for the experimentations

Chapter 2

Theoretical Background

2.1 Image noise

Image noise is a random variation of brightness or color information in images, usually generated during image formation, generally due to one of the following causes: light fluctuations, sensor noise, quantization effects, finite precision. The ways in which it could affect the quality of the images are various, in this case it has been considered an additive noise, independent at each pixel, independent of the signal intensity, of one of the following types:

- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution.
- **Salt and pepper noise**: random occurrences of black and white pixels.

2.2 Noise reduction

Since, as said before, usually noise is independent at each pixel, to reduce its effect it is useful to apply a smoothing filter, whose effect is that of blurring the image, removing great variations in the intensity of the signal (low-pass effect). There are different image filtering techniques, both linear and non linear. Linear filters works by applying a weighted average over an area of a certain dimension (the sum of the coefficients of the average has to be 1 to preserve the average intensity of the image). An important parameter of these filters is the size of the kernel, which determines the amount of smoothing, the bigger the size the bigger the effect of smoothing.
In this experiment the focus was given to the following filter types:

- **Moving average filter**: it is a linear filter that works by applying a uniform average over a pixel area
- **Gaussian filter**: it is a linear filter that works by applying a weighted average over a pixel area, giving more weight to the central pixels. An important parameter

for the Gaussian filter is the standard deviation, which determines the extent of smoothing. In fact, its shape is given by the following formula: $G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$. It's important for the filter to have half-width to about 3σ, since discrete filters use finite kernels but Gaussian function has infinite support.

- **Median filter**: it is a non linear filter that works by ordering the pixel's values contained in the dimension of the kernel and taking the middle value. It is particularly efficient to remove spikes and therefore very useful to reduce the effect of noise like impulse and salt & pepper noise. Other very important features are that it is edge preserving and it doesn't introduce new pixel values.

2.3 Linear filtering

Linear filters can be used to achieve various results other than smoothing. Their effect is obtained by the usage of the 2D-convolution, which only differs from the classical 1D-convolution in the way it compute the values of border pixels (various ways exist), therefore some of the result achieved shouldn't surprise. The linear filters examined in this experiment are the following:

- **Identity filter**: it is the 2D equivalent of the Dirac delta function, therefore the result of the function is the original image, as to be expected.
- **Shifting filter**: it corresponds to a shifted Dirac delta function, therefore the result of the function is the original image translated of the same quantity as the delta. Again it is the same as in the 1D analysis.
- **Sharpening filter**: this filter works by multiplying the intensity value of each pixel for a constant (the effect of the filtering using a Dirac delta function multiplied by the constant) and then subtracting the original image filtered by a moving average filter (therefore smoothed) in which the sum of the weights is equal to the constant value minus one, this because the final result of the filtering should preserve the average intensity of the image (actually in the implementation at first the difference of the two filter is computed and then the convolution to the original image is applied). The final effect is that of sharpening the image.

2.4 Fast Fourier Transform

Finally, examples of the result of the Fourier Transform of images are examined. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image. The number of frequencies corresponds to the number of pixels in the spatial domain image, i.e. the image in the spatial and Fourier domain are of the same size. The formula for the Fourier Transform in the discrete domain is the following: $F(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) e^{-j(\frac{2\pi}{M} + \frac{2\pi}{N})}$
In MATLAB it is done using the functions fftshift and fft2.

Chapter 3

Implementation and Results

The code is organized using a script, which loads the grayscale image in the workspace and then calls all the developed functions. In order to carry out all the required manipulation the following functions were developed:

- **GaussianNoise**: which simply adds a Gaussian noise to the image and displays the resultant image along with its histograms.
- **SPNoise**: which simply adds a salt & pepper noise to the image and displays the resultant image along with its histograms.
- **MAFilter**: which implements a moving average filter with a support of the dimension indicated by one of the parameters and displays the filter support, the pre and post filter image along with the relative histograms.
- **GaussianFilter**: which implements a Gaussian filter with a support of the dimension indicated by one of the parameters and displays the filter support, the pre and post filter image along with the relative histograms.
- **MedianFilter**: which implements a median filter with a support of the dimension indicated by one of the parameters and displays the filter support, the pre and post filter image along with the relative histograms.
- **NoChangeFilter**: which simply implements the identity filter and plot the resultant image.
- **TranslateFilter**: which simply implements the shifting filter and plot the resultant image.
- **SharpeningFilter**: which simply implements the sharpening filter and plot the resultant image.
- **FFTtranformShow**: which applies the Fast Fourier transform to the original image and displays its magnitude.

- **FFTGauss**: which applies the Fast Fourier transform to a Gaussian filter and displays its magnitude.

3.1 Noise Adding

In order to follow the path outlined in the Introduction section, the first thing to do was to add the two different types of noise separately to the provided image, producing a couple of the same image, one affected by Gaussian noise and one affected by salt & pepper noise. For the Gaussian noise, a matrix in which each pixel is randomly generated with normal distribution using a standard deviation of 20, is simply added to the original image, the result can be seen in figure 1.

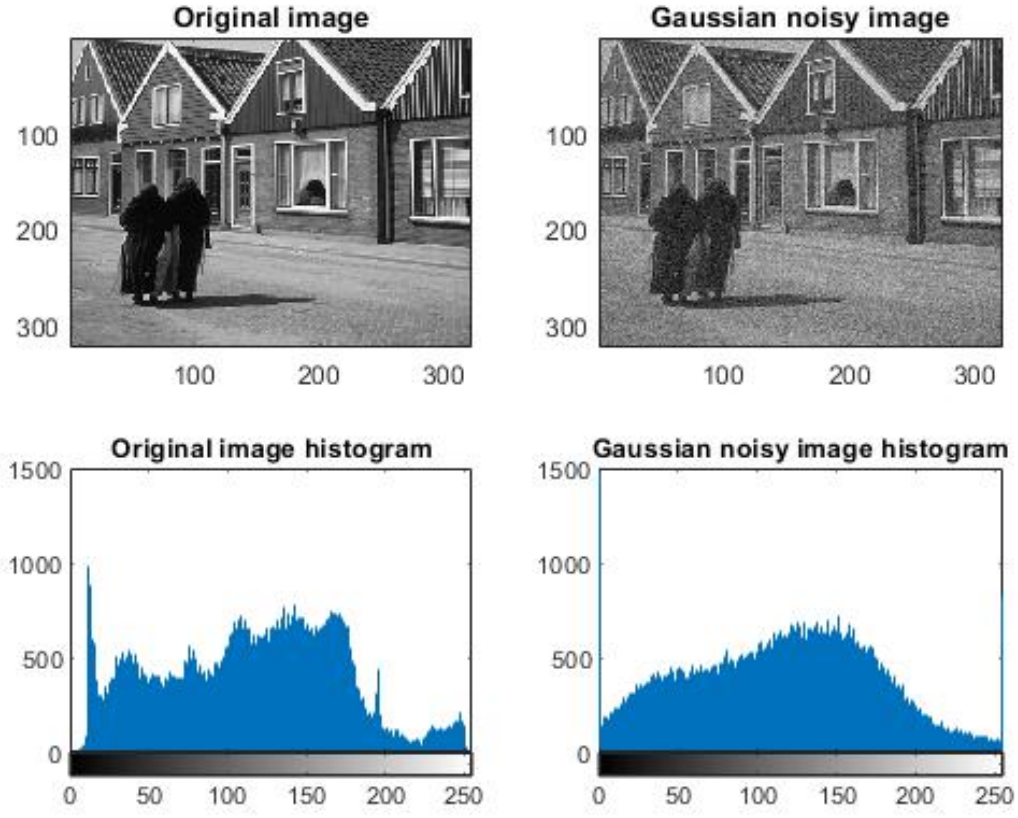

Figure 3.1: Original image compared with gaussian noisy image

Regarding the salt and pepper noise, the procedure is a little bit less straightforward. First it is generated a random uniformly distributed matrix of the same dimension as the image with all zeros but some elements in random positions with the density required, whose values are between zero and one. Then from this matrix two masks are generated,

one in which all the elements whose value is between 0 and 0.5 (extremes excluded) are set to one and to zero otherwise, the other in which all the elements whose value is between 0.5 and 1 are set to one and to zero otherwise. Eventually, these masks are applied by bitwise multiplication to the original image in different ways. The final result is the original image with some pixels set to the maximum value of intensity and some other set to the minimum value of intensity. The result can be seen in figure 2 below.

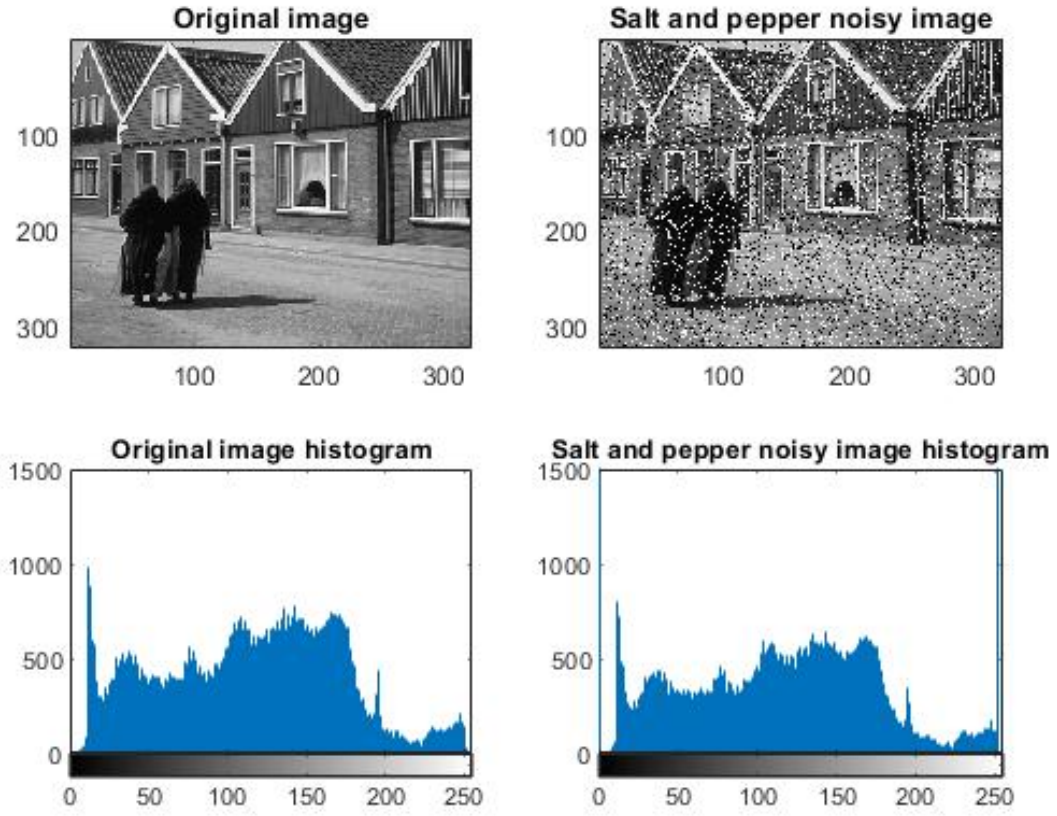

Figure 3.2: Original image compared with salt & pepper noisy image

3.2 Noise Filtering

This experiment examined the countermeasures provided by the three different type of filters described in the previous chapter. All these filters were applied to the noise affected images in two different ways by using two different spatial supports, one of 3x3 pixels and one of 7x7 pixels. For the moving average filter a matrix of ones divided by the square of the support dimension is created and then the final image is the result of the convolution of this matrix and the source image. For the Gaussian filter there is a special function in

MATLAB to create this type of filter with the desired support and standard deviation. It is sufficient to apply the created filter to the image by convolution. It is important that the support's dimension is at least six times the standard deviation to avoid a distorted resulting image. The median filter in MATLAB is applied by using a specific function. Below are included all the resulting images of each case, from figure 3 to figure 12 and it is possible to notice how effective the median filter is in the case of the salt and pepper noise, instead the moving average filter and the gaussian filter seem to be more effective in the case of gaussian noise, rather than in the salt and pepper case.

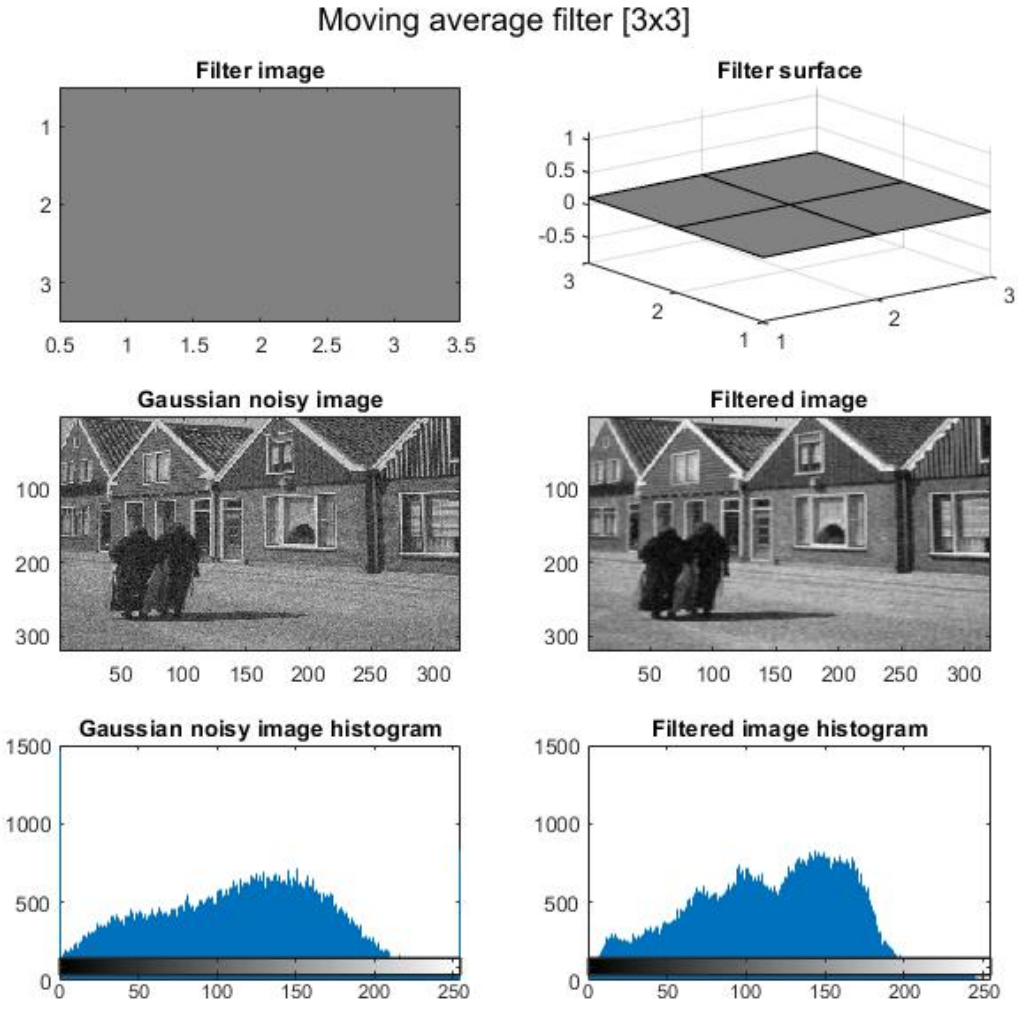

Figure 3.3: Moving average filter applied on gaussian noisy image

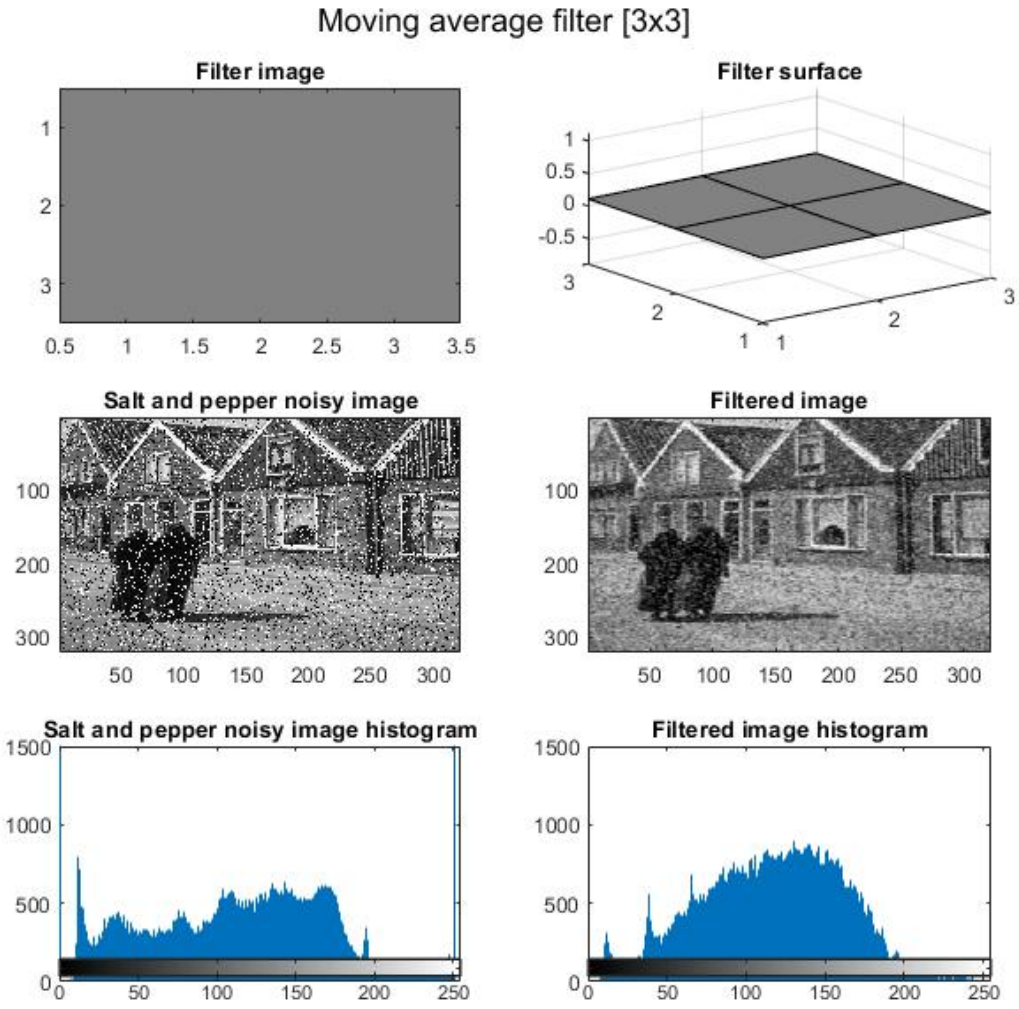

Figure 3.4: Moving average filter applied on salt and pepper noisy image

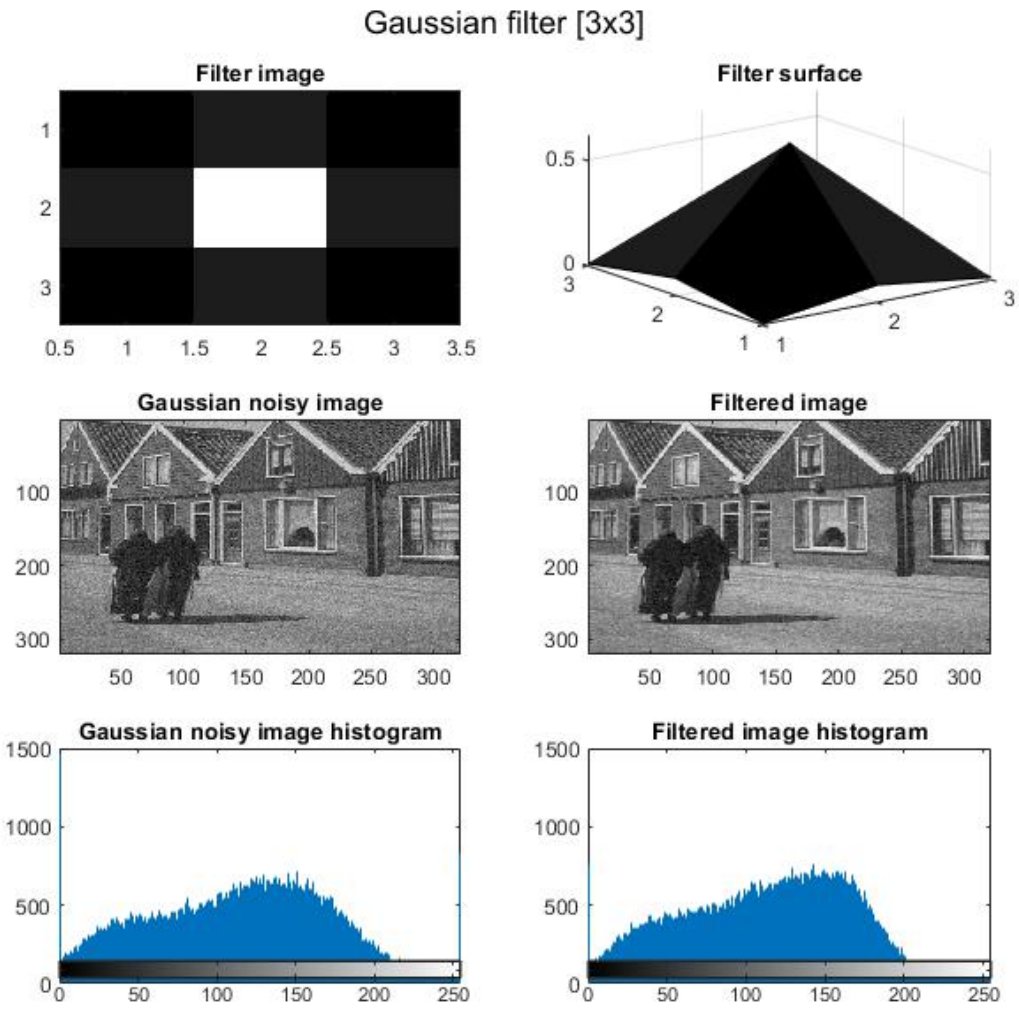

Figure 3.5: Gaussian filter applied on gaussian noisy image

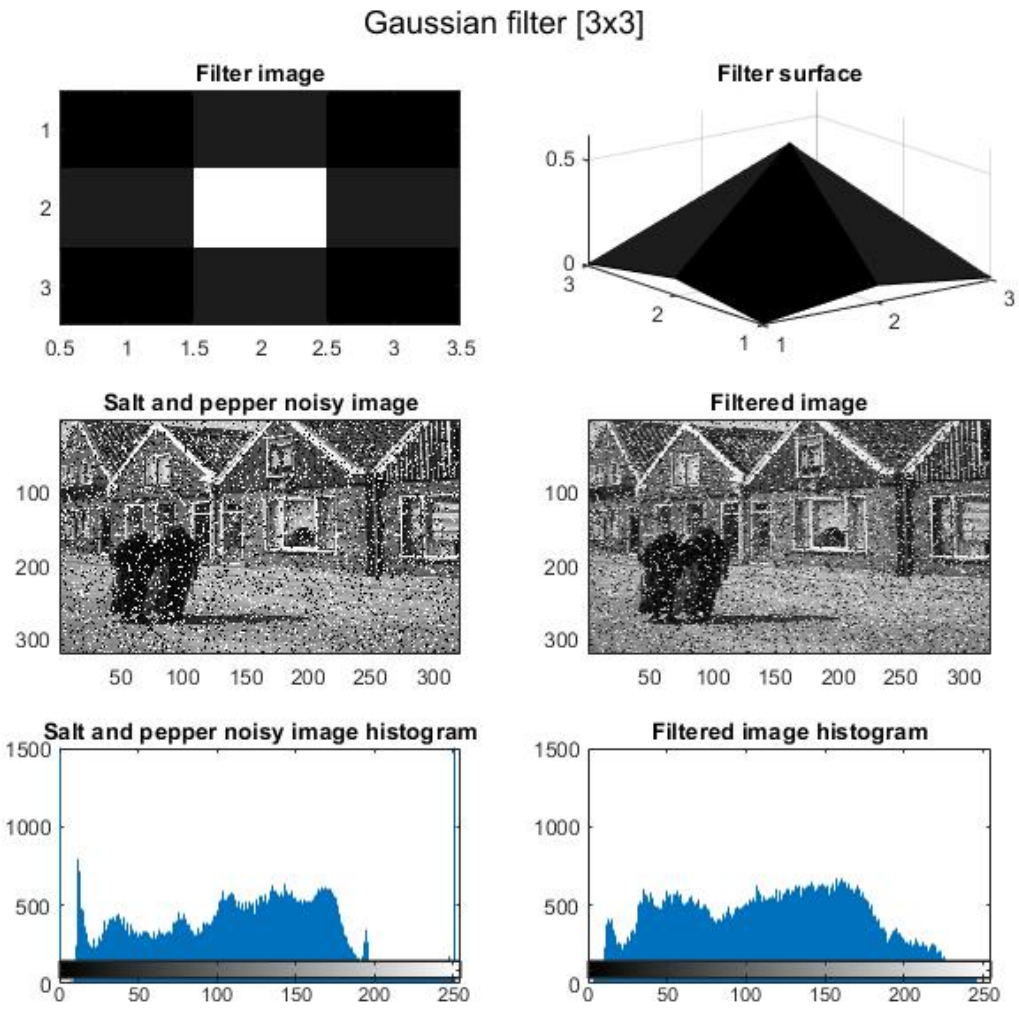

Figure 3.6: Gaussian filter applied on salt and pepper noisy image

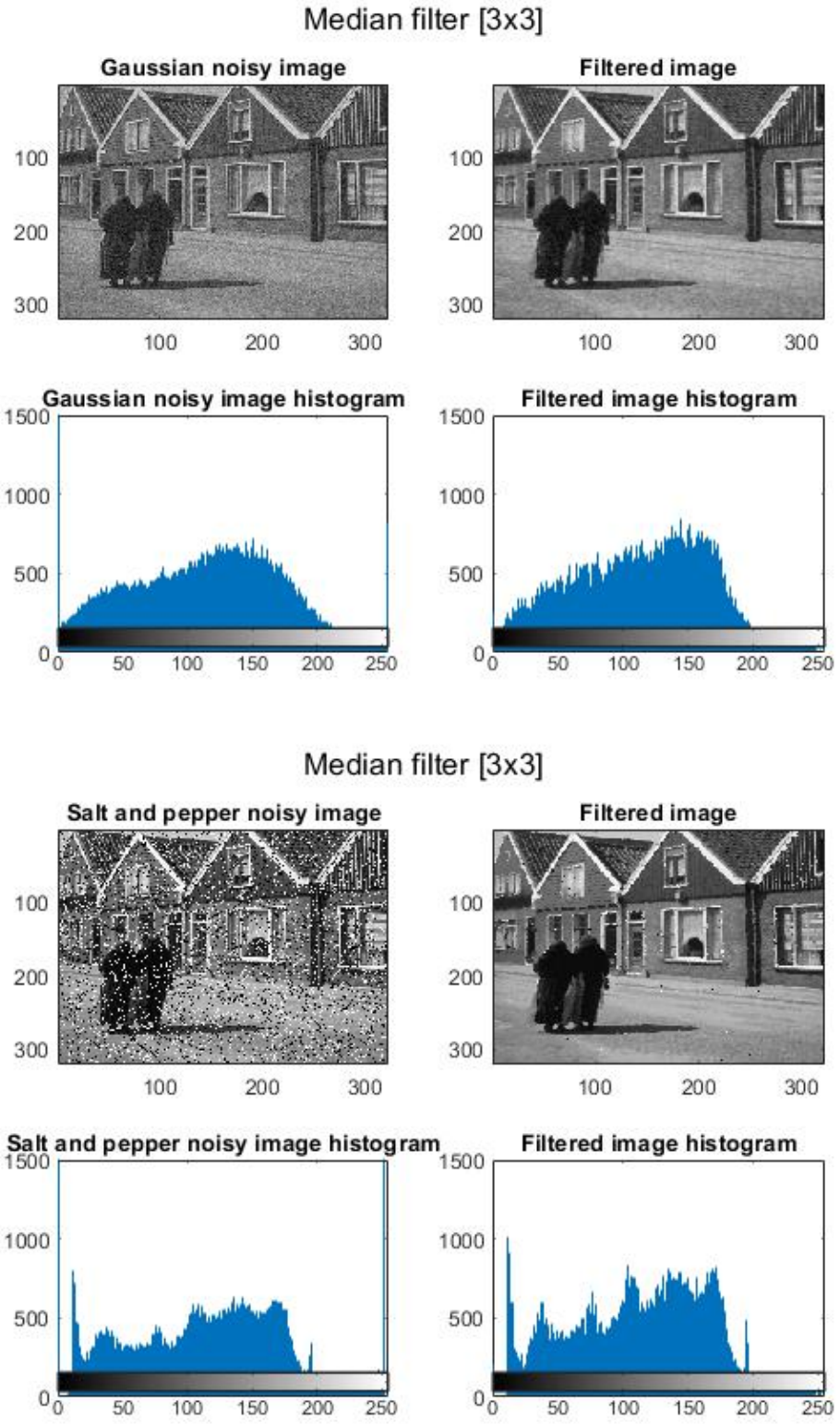

Figure 3.7: Median filter applied on gaussian and salt and pepper noisy images

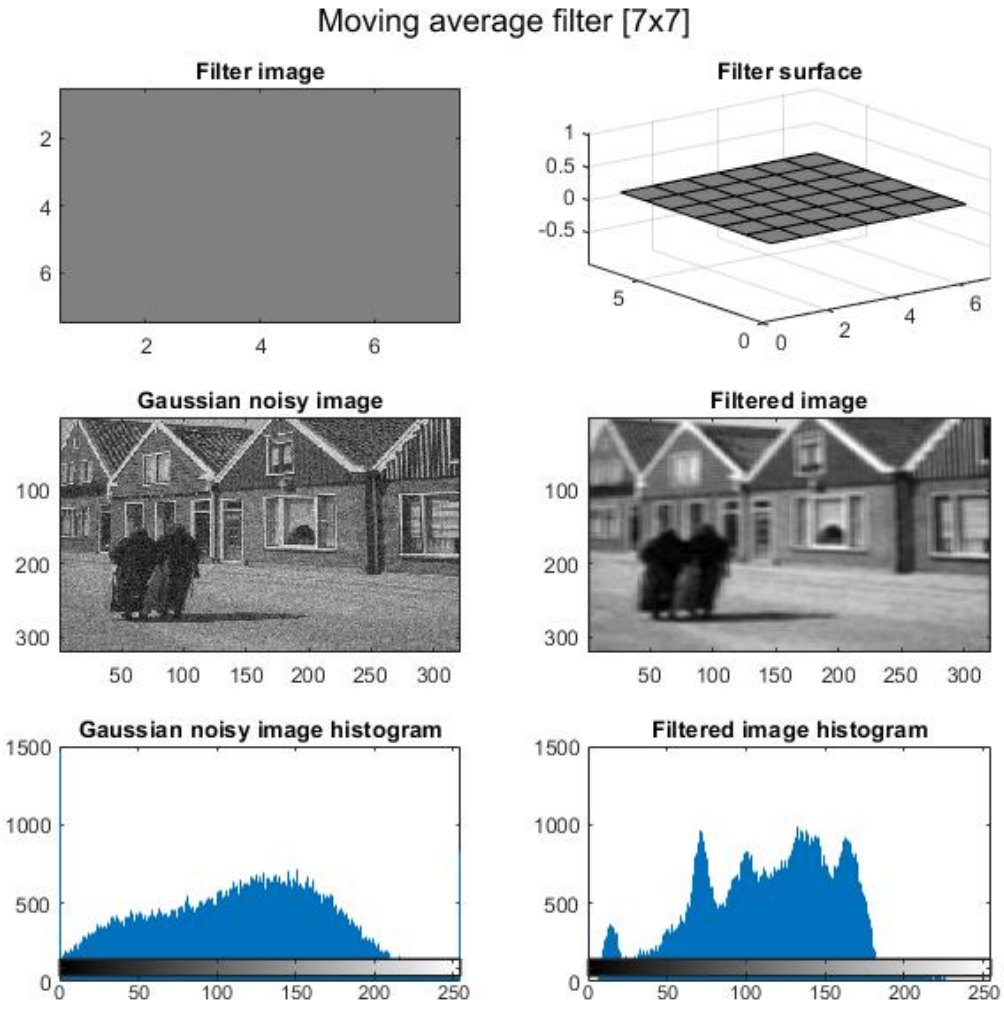

Figure 3.8: Moving average filter applied on gaussian noisy image

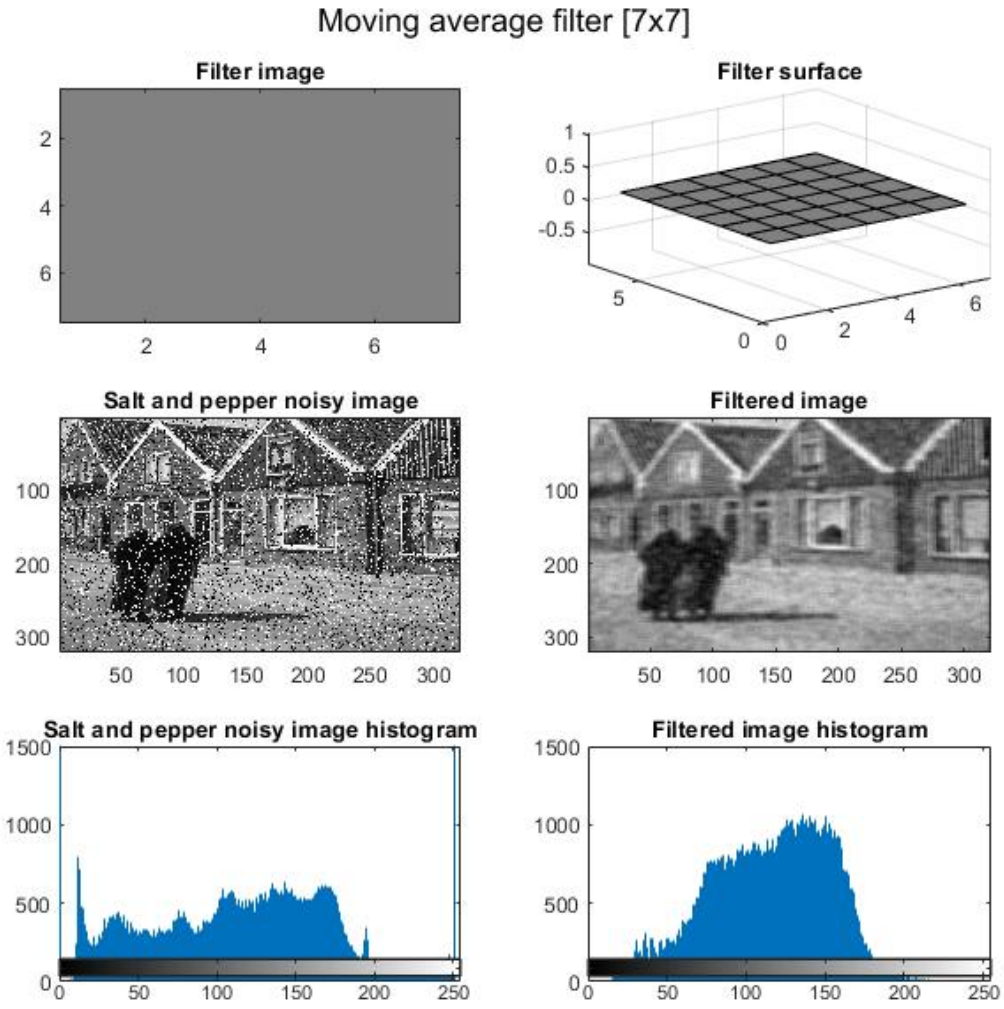

Figure 3.9: Moving average filter applied on salt and pepper noisy image

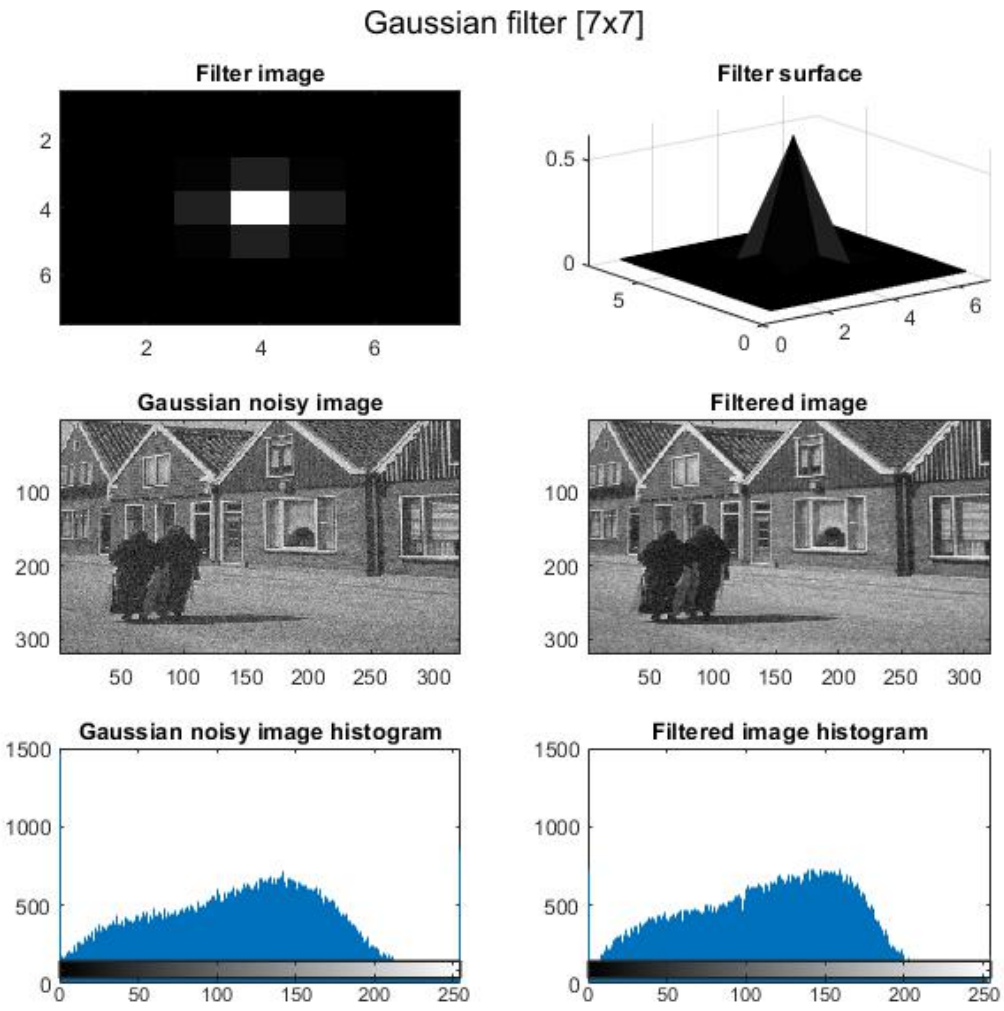

Figure 3.10: Gaussian filter applied on gaussian noisy image

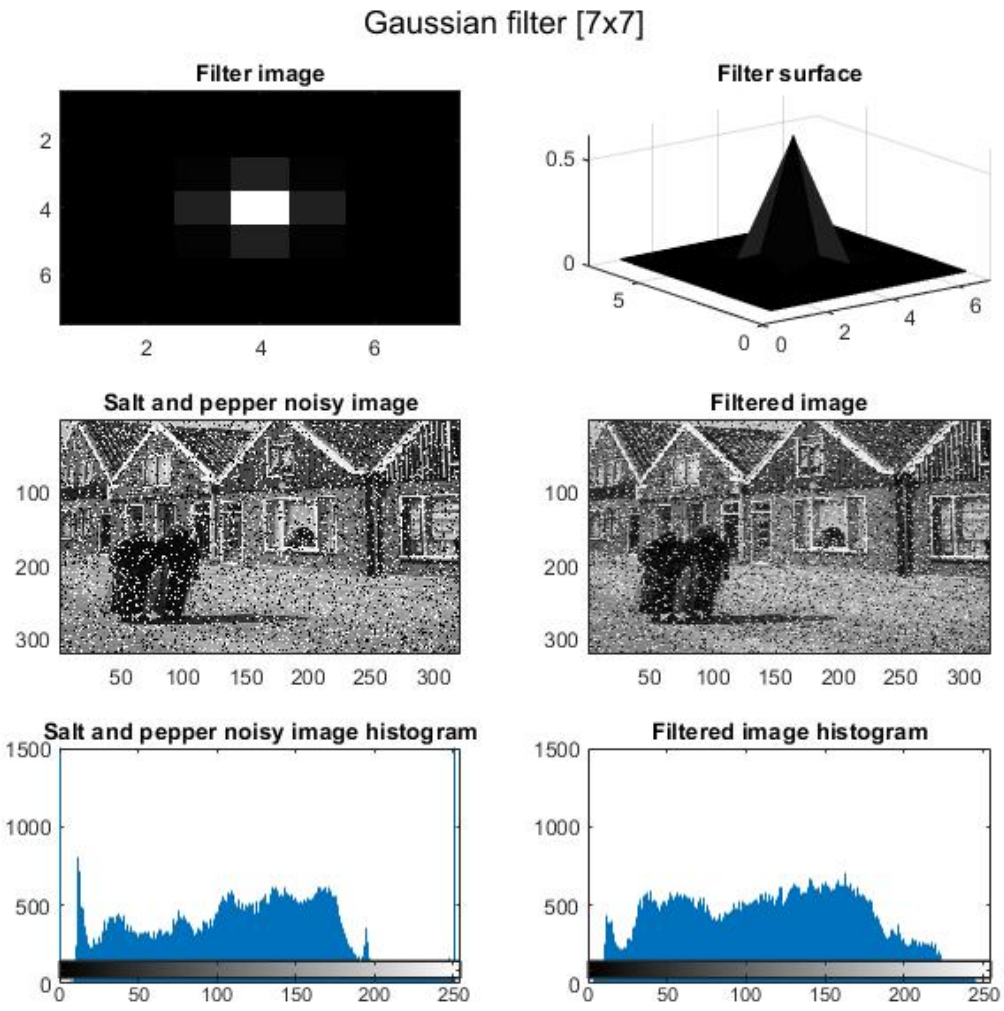

Figure 3.11: Gaussian filter applied on salt and pepper noisy image

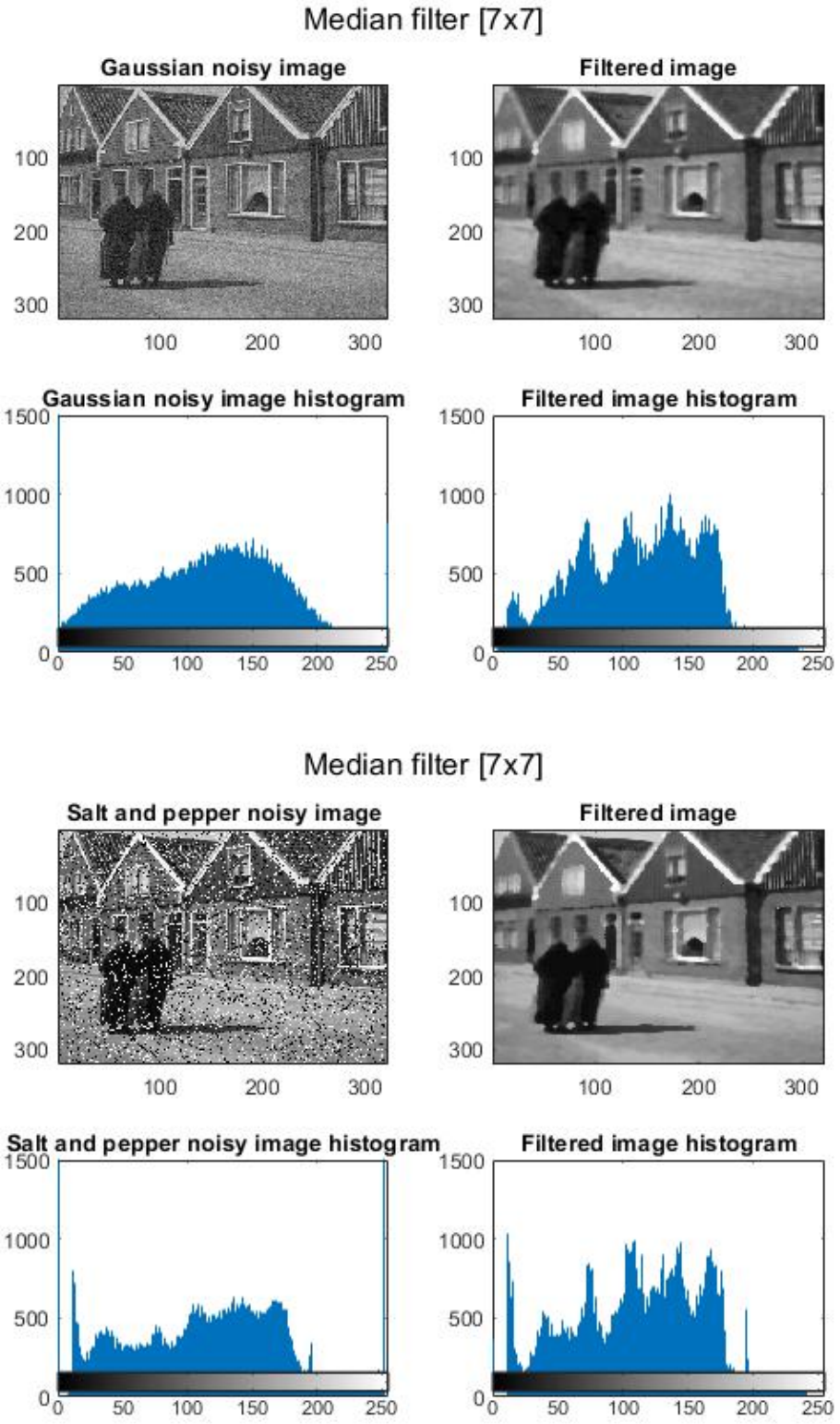

Figure 3.12: Median filter applied on gaussian and salt and pepper noisy images

3.3 Linear filter examination

Between Noise Filtering and Fourier Transform examination, some experimentation with linear filters were conducted. In particular the three types of linear filters listed in the previous section were applied to the original image, not affected by noise, with a spatial support of 7x7 pixels. The result are shown in the images below, from figure 13 to figure 15 and it is possible to notice that in the first case the image remain the same, in the second it is translated of 3 pixel toward right and in the third details are emphasized and the edges enhanced.

Figure 3.13: Unitary/identity filtering

Figure 3.14: Translation filtering

Figure 3.15: Sharpening filtering

3.4 Fast Fourier Transform Evaluation

As anticipated in the introductory section the last part of the experiment focuses on evaluating the Fast Fourier Transform by examining the magnitude of the transformed image, shown in figure 18 and the magnitude of the transformed low pass Gaussian filter having a size of 101x101 pixels and $\sigma = 5$ as parameters, shown in figure 19.

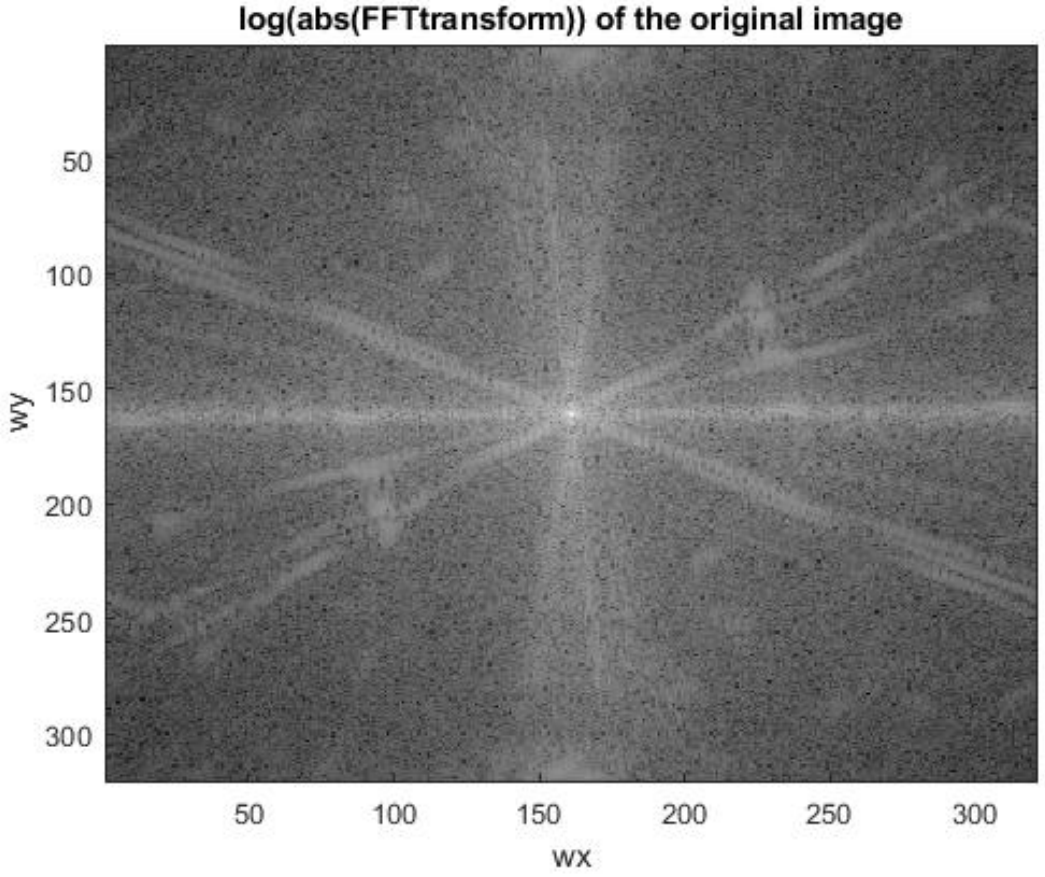

Figure 3.16: Magnitude of the transformed original image

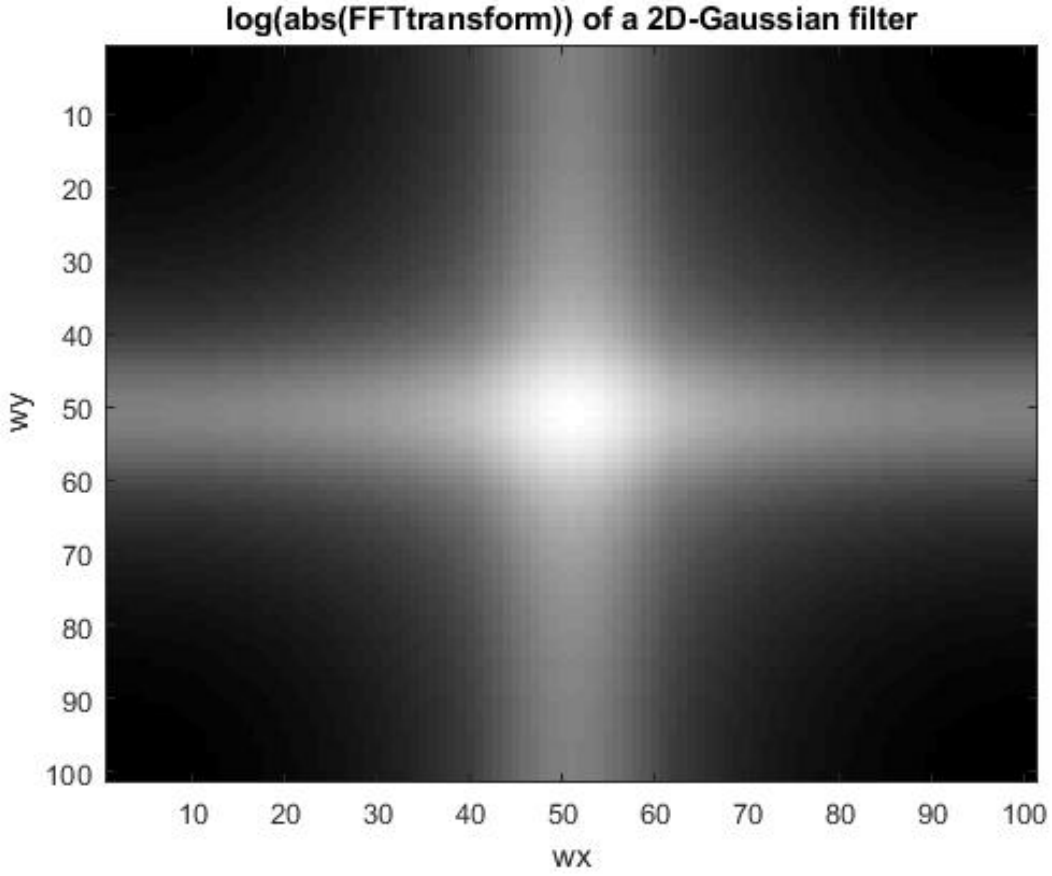

Figure 3.17: Magnitude of the transformed low-pass gaussian filter