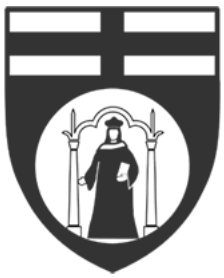

Università degli studi di Genova

Computer Vision

Assignment 2: Image filtering and Fourier Transform .

Aurora Bertino, Chiara Saporetti

March 28,2020

Contents

1. Introduction

In this document are described the steps to perform noise in Image processing such as Gaussian and Salt & Pepper noise and to filter images in order to decrease these disturbances. The filters used are: moving average filters, low-pass Gaussian filters and median filters. The input images are: 'tree.png' and 'i235.png' . Subsequently, the Fourier Transform (FFT) is applied on the provided images.

1.1 Noise

Noise in imaging system is usually present during image acquisition, coding, transmission, and processing steps. This noise, disturbs the original information of the image, for this reason we try to assess how much original signal is corrupted, how we can reconstruct the signal and which noise model is associated to the real noisy image. So, it's clear that noise is a random signal and it is used to destroy most of the part of image information. We often assume that the noise is additive:

I(x,y) = s(x,y) + ni

where s(x,y) is the deterministic intensity signal, and "ni" is a random variable. TRansformed image is due to various types of noise such as Gaussian noise, Poisson noise, Speckle noise and Salt & Pepper noise. These are noise models that we try to reconstruct in analytical way, in order to reduce disturbances present in nature. In the Lab we are going to add Gaussian noise and Salt and Pepper noise to the images: "tree.png" and "i235.png".

1.2 Gaussian noise and Salt & Pepper noise

Gaussian noise is also called electronic noise because it arises in amplifiers or detectors and generally it disturbs the gray values in digital images. The model is designed by a Gaussian function:

$$P(x) = \sqrt{\frac{1}{2\pi\sigma}} e^{-\frac{(x-\mu^2)}{2\sigma^2}}$$

Salt & Pepper noise is seen in data transmission. Image pixel values are replaced by corrupted pixel values, either maximum 'or' minimum pixel value i.e., 255 'or' 0 respectively, if number of bits are 8 for transmission. Let us consider 3x3 image matrices. Suppose the central value of matrices is corrupted by Pepper noise and the central value 212 is replaced by value zero. We can say that dark pixel values are present in bright region and vice versa. We have reviewed noise models available in digital images and they are also designed by probability density function using mean, variance and mainly gray levels in digital images. For this reason the study of noise model is very important part in image processing.

1.3 Median filter

The moving average filter is a simple low-pass filter commonly used in signal and image processing . Given an image and a fixed subset size, the first element in output is obtained by taking the average of the initial fixed subset of pixels. Then the set is modified by "shifting forward" in the image the kernel of the filter. As the length of the filter increases, the smoothness of the output increases.

1.4 Moving average filter

The median Filter is an example of a non-linear spatial filter. It is widely used as it is very effective in removing noise, especially salt and pepper noise, while preserving the edges. It also preserves the discontinuities in the images and only removes the outliers. The main idea of the median filter is to run through the image pixel by pixel, by replacing each pixel's intensity with the median of neighboring pixels. The pattern of neighbors is called the "window", which slides, entry by entry, over the entire figure. The median is calculated by first sorting all the pixel values from the window into an array in numerical order, and then replacing the considered pixel with the middle (median) pixel value. If window elements are even, the middle is computed by using the mean of the two adjacent pixels.

1.5 Low-pass Gaussian filter

The Gaussian low pass filter is a linear filter that performs a weighted average of pixels in a neighborhood by using a Gaussian function. This curve enables to give more weight to the central pixels and less to the periphery ones. Its formula is given by the following:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

The function has finite special support, whereas discrete filters depend on a finite size kernel. The size of the kernel should be set to about 6 times the value of sigma of the function. The standard deviation (sigma) of the gaussian determines the extent of smoothing: as it increases, the smoothing effect increases too.

1.6 Fourier Transform

The Fourier Transform is an important image processing tool which is used to decompose an image into its sinusoidal and cosinusoidal components. The spatial function f(x,y) is decomposed into a weighted sum of 2D orthogonal basis functions in a similar manner to decomposing a vector into a basis using scalar products.

$$f(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(u,v) e^{j2\pi(ux+vy)} dudv$$

The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent. The Fourier Transform is used if we want to access the geometric characteristics of a spatial domain image. Because the image in the Fourier domain is decomposed into its sinusoidal components, it is easy to examine or process certain frequencies of the image, thus influencing the geometric structure in the spatial domain. The Fourier transform consists of a real and a complex component: the magnitude and phase, respectively. The phase information is crucial to reconstruct the correct spatial structure of the image.

2. Matlab Resolution

As previously stated, with our code it is possible to apply, to an input image, different kinds of noises and filters. It also allows the user to compute and plot the Fourier transform to the modified images. We tested our code on two different greyscale pictures. More specifically, in the code we initially add gaussian noise to the input image and then apply two sets of filters to the "noisy" image. The filters of the first set are, in order: Moving average filter, Gaussian filter and Median filter, all of them applied with two different kernel sizes (kSize1=3, kSize2=7). The filters of the second set are and impulse filter, a shifted impulse filter and a sharpening filter. They all have the same kernel size (kSize2=7). Regarding the last filter of this set, our code returns two different images filtered with it, by following, in the first case, slide 44 and, in the second case, slide 45. The resulting images are all shown via the matlab function of imagesc() while the filters are shown with imagesc() and surf(). The same process of filtering is then applied to the input image modified by adding salt and pepper noise.

2.1 Gaussian noise and Salt & Pepper noise

$$(transfImg, hist) = gaussianNoise(Img, stdDev) \tag{2.1}$$

This function takes as a input Img, which is the original image and stdDev which is the standard deviation of the gaussian function. It gives as output the transfImg, hence the Gaussian noisy image.

$$[transfImg] = SPNoise(Img, density) \tag{2.2}$$

This function adds salt and pepper noise to the image. The input values are= the image Img and the density of the noise. It returns the noisy image transfImg.

2.2 Median filter

This function simply applies the median filter to the input image Img by setting the kernel's size to the value passed by the main in kSize. In order to do so, it uses the function medfilt2(). The function returns the filtered image.

$$(transfImg) = medFilter(Img, kSize) \tag{2.3}$$

This function simply applies the median filter to the input image Img by setting the kernel's size to the value passed by the main in kSize. In order to do so, it uses the function medfilt2(). The function returns the filtered image.

2.3 Moving average filter

$$(transfImg, K) = movingAverageFilter(Img, kSize) \tag{2.4}$$

This function applies the moving average filter to the input image Img by setting the kernel's size to the value passed by the main in kSize. The image is modified by adding a "frame" of pixels with intensity set to 1 and width equal to kSize, then the filter is created manually and applied by using the function conv2. Before sending the transformed image in output, it is cut again to its previous size. The function returns both the filtered image and the kernel of the filter.

2.4 Low-pass Gaussian filter

$$(transfImg, h) = gaussFilter(Img, kSize) \tag{2.5}$$

This function applies the gaussian filter to the input image Img by setting the kernel's size to the value passed by the main in kSize. Firstly, the value of sigma (the standard deviation of the gaussian curve) is set to 1/6th of the kernel size, in order to respect the relationship that must occur between the two values. Then the filter is created by using the fspecial() function and applied through the imfilter() function. The function returns both the filtered image and the kernel of the filter.

2.5 Linear filter

$$(transfImg1, K1, transfImg2, K2, transfImg3ver1, transfImg3ver2, K3) = linearFilters(Img, kSize) \tag{2.6}$$

This function applies three different linear filters to the input image Img by setting the kernels sizes to the value passed by the main in kSize. The first filter is a matrix with all values set to zero a part from the central element, which is set to one:

$$K1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.7}$$

The second filter is a is a matrix with all values set to zero a part from the last element of the central row, which is set to one.

$$K2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.8}$$

The third filter is the difference between a matrix equal to the first filter and a matrix of elements all set to . It is applied in two different ways: the first was designed according to slide 44, so by simply calculating the convolution of the image with the kernel:

$$K3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} - \frac{1}{kSize^2} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.9}$$

The second was designed according to slide 45, so by substracting the previous result to the original image, multiplying it for a value "a" and finally summing the result to the original image. We chose the value a=0.1 since it best fits our results.

$$K = image + a(image - res) \tag{2.10}$$

The function returns the transformed images (obtained by applying a convolution between the image and the filter) and the relative kernels.

2.6 Fourier Transform

In the Fourier domain we create a Gaussian low-pass filter with the following function:

$$FG = fspecial('gaussian', pixels, sigma) \tag{2.11}$$

$$FFG = fftshift(fft2(FG)) \tag{2.12}$$

Subsequently, we display the magnitude of the Gaussian filter and the magnitude of sharpening filter that we have already computed.

3. Results

3.1 Test

As result we have noisy images and their histograms. The noises that we added are: Gaussian and Salt & Pepper noise. In this paper we display the results only for the image 'i235.png'. When we apply the two kinds of noise to the images it is evident that the Gaussian noise affects the image less then the salt and pepper one, or at least that is what it appears to the human eye because of the sharp transition in illumination between the pixels. The histograms of the picture with Gaussian noise shows that the pixels appear to form a sort of Gaussian curve, while with salt and pepper noise there are two peaks at the extremities of the illumination values (0 and 256), due to the added black and white pixels. The second test is the comparison between different image filtering. The filter used are: moving average, low-pass Gaussian and median filter. Hence, we analize different type of filters with different size: (3x3),(7x7). The noise applied is Salt & Pepper. In the salt and pepper images, when applying the filters, it is evident that the best one to remove the noise is the median filter. In particular, the one that has a higher K size removes most black and white pixels but also produces a greater blur with respect to the other, so this could be taken into consideration in the choice ok K size. The gaussian filter doesn't work well on the picture, while the moving average eliminates the black and white pixels but produces a lot of blur. In the images affected by gaussian noise, when applying the filters, it seems that the best one to remove the noise is the moving average filter. In general, however, by increasing the size of the kernel also the blurring effect is increased, so that the noise is less evident but information is partially lost. Regarding the three linear filters, the convolution of the image with the first filter results in the same image, since it is an impulse filter. The second result is again the image itself, just shifted of one pixel. The third one is the sharpened version of the image, where the discontinuities in illumination are underlined and increased. Subsequently, we implemented the liner filters described in the slides 41-45, the noise applied is Salt & Pepper. In the last test, we applied the Fourier Transform (FFT) to the images. After that we are going to display the magnitude (log) of the transformed image and we can notice that for the two provided images the magnitude is similar. Then display the magnitude of the transformed low-pass Gaussian filter. Then we conclude showing the magnitude of the transformed sharpening filter, (slide 44). *[6] [4] [5] [2] [1] [3] [7]*

Figure 3.1: Test 1: original image and noisy image (Gaussian noise)

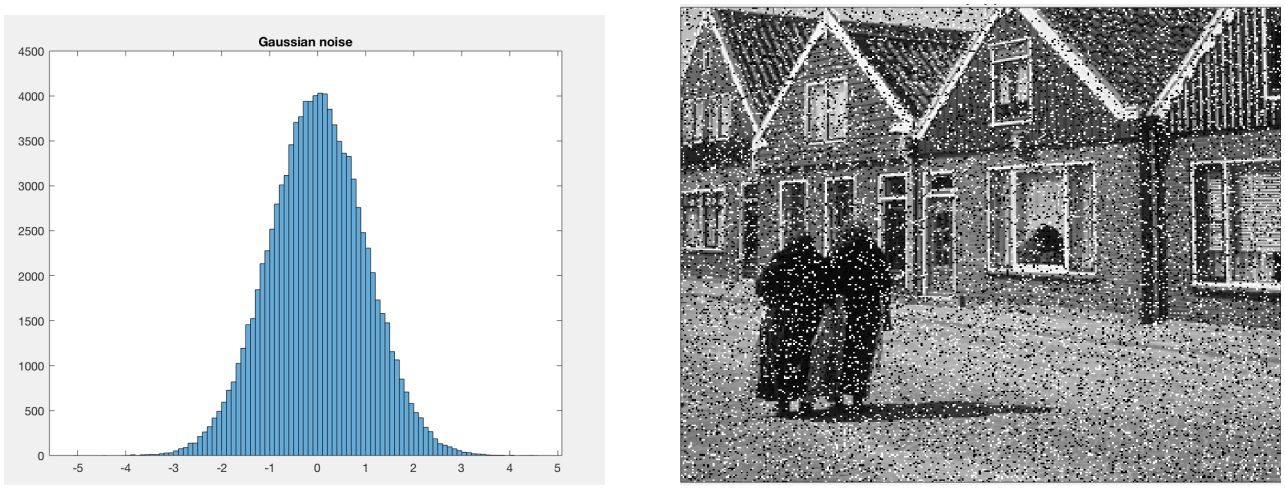

Figure 3.2: Test1: Histogram (Gaussian noise) and image with Salt & Pepper noise.

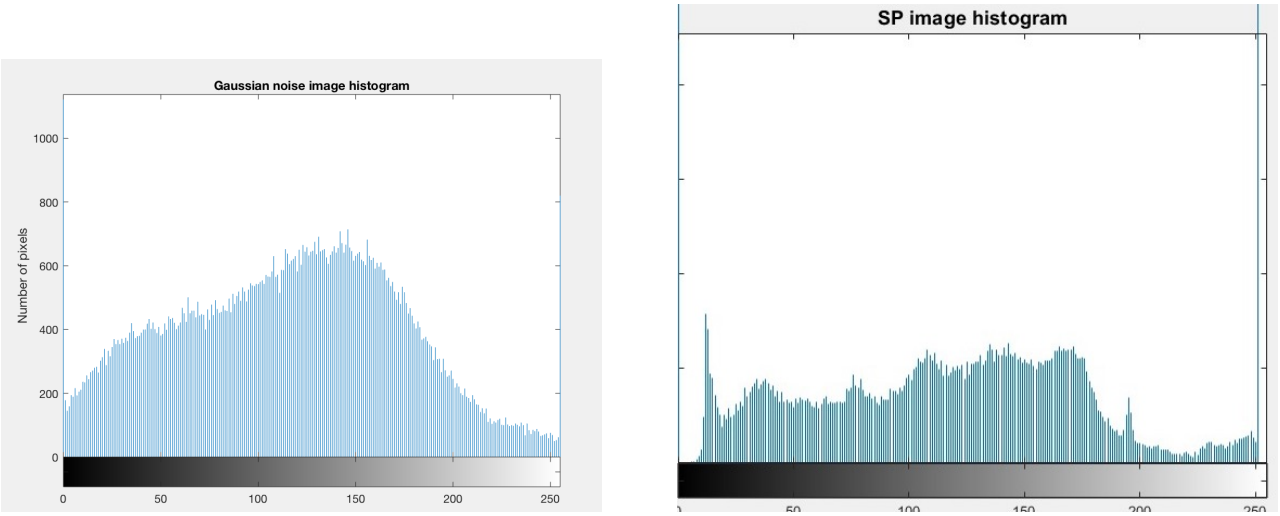

Figure 3.3: Test1: Histogram noisy image (Gaussian noise), histogram noisy image (Salt & Pepper noise).

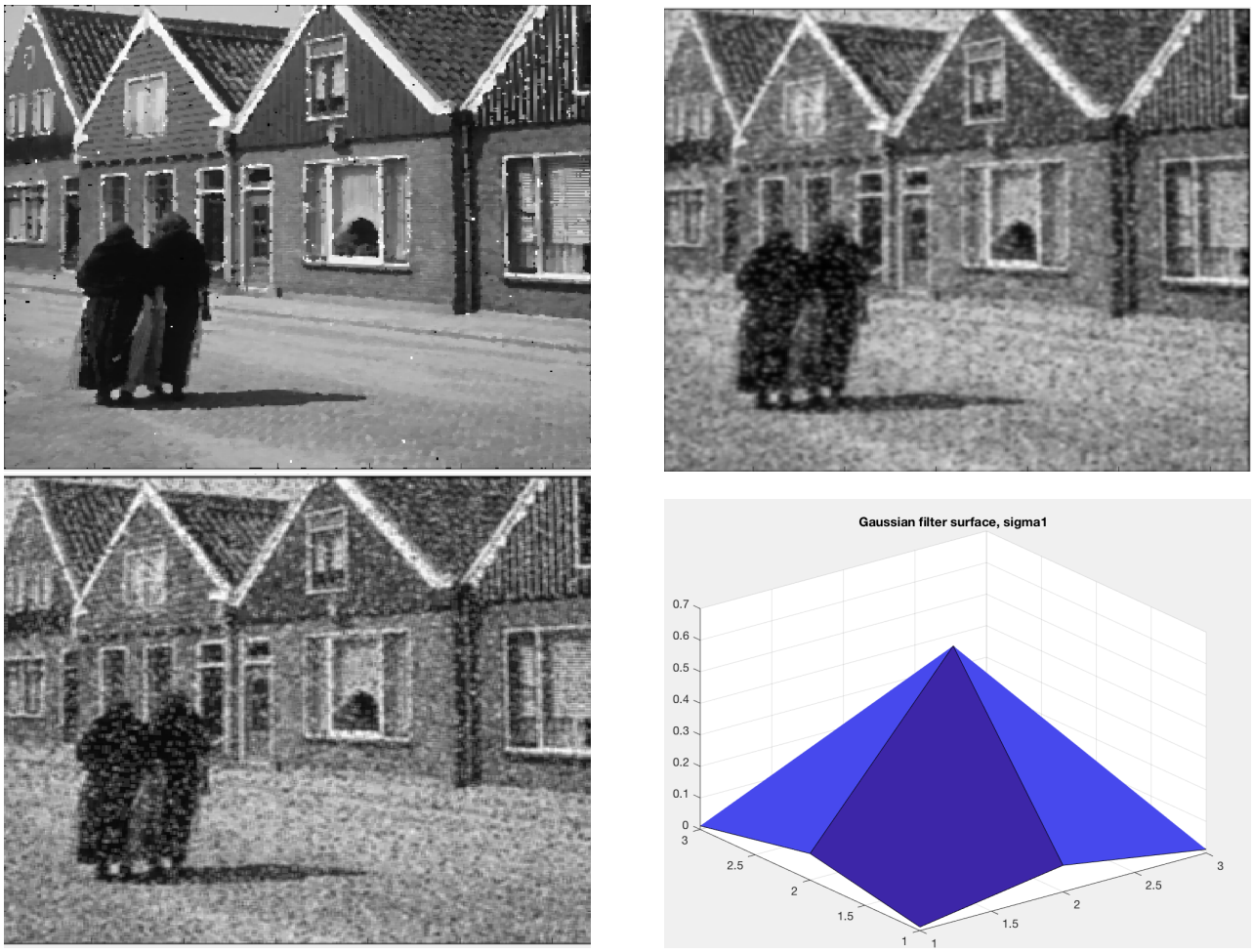

Figure 3.4: Test2: Image filtered with a median filter (kernel k1=[3x3]), image filtered with Gaussian filter (sigma1=0.5), image filtered with moving average filter (kernel k1=[3x3]), 3D plot of Gaussian filter (sigma1) .

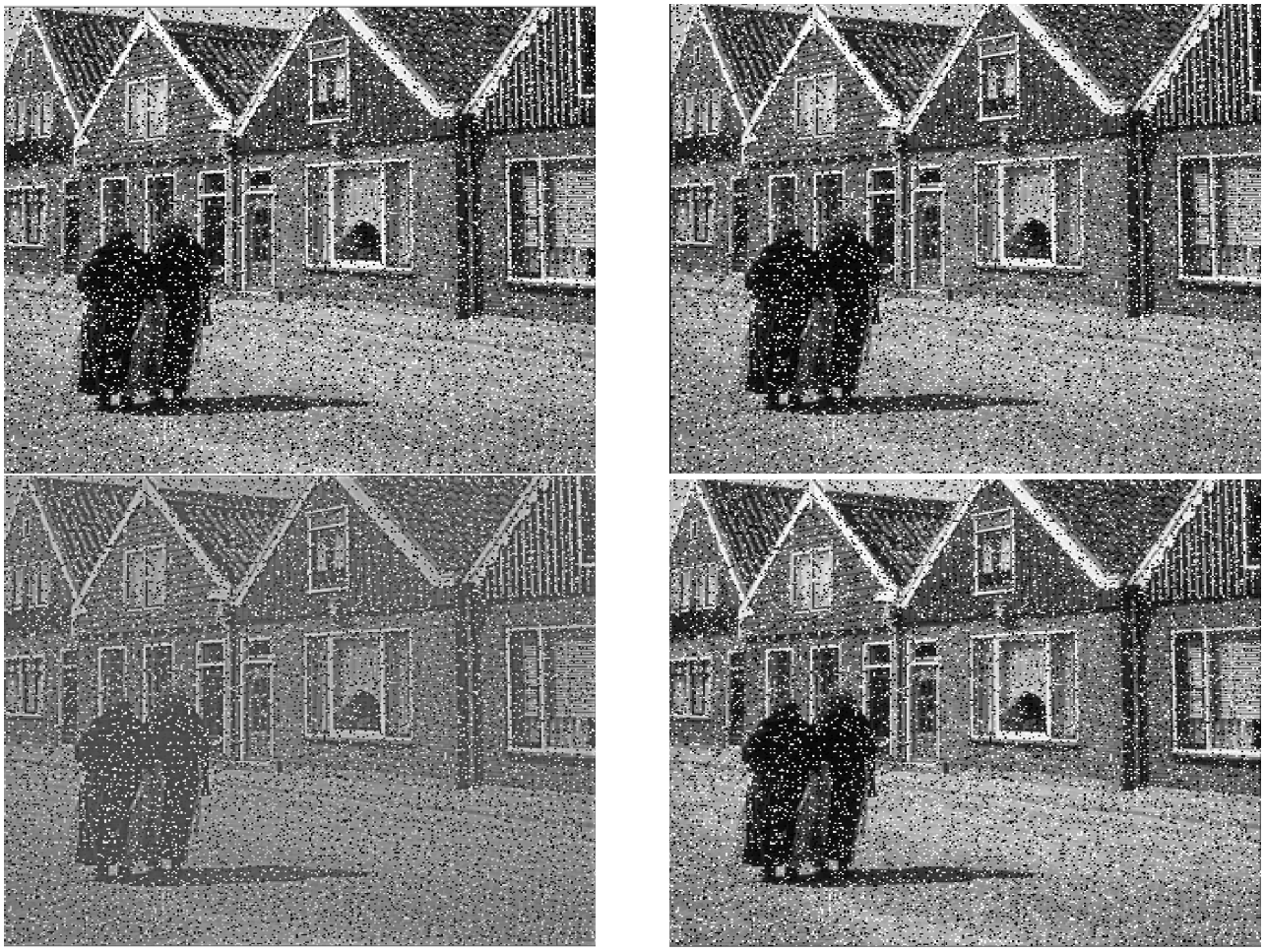

Figure 3.5: Test3: Image filtered with an impulse filter , image filtered with shifted impulse filter, image filtered with sharpening filter (slide 44),image filtered with sharpening filter (slide 45).

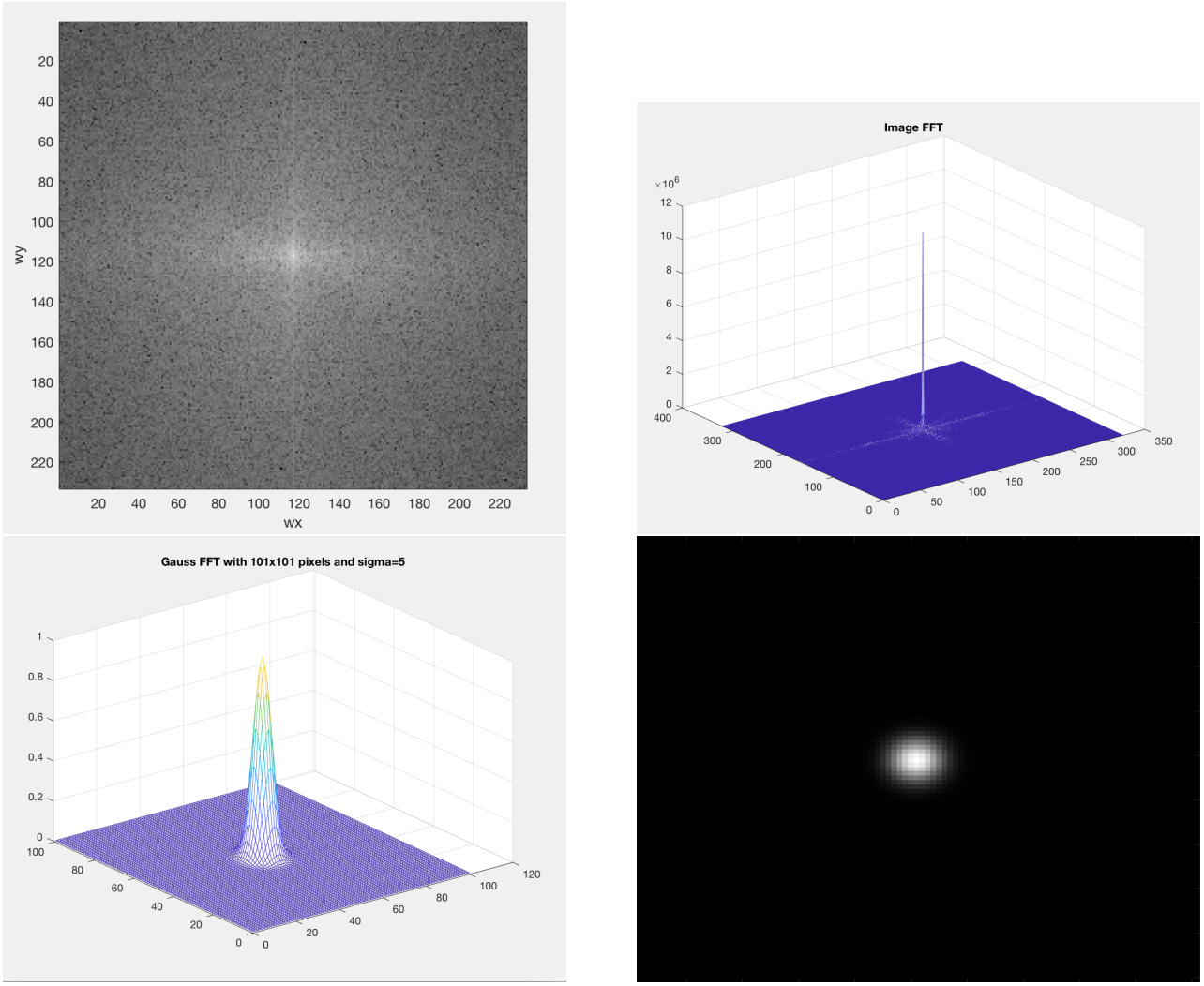

Figure 3.6: Test4: We display the FFT of the images, we also display the magnitude (log) of the the image transformed ("tree.png", the magnitude of the transformed low-pass Gaussian filter (spatial support of 101x101 pixels with sigma=5) in 2D and in 3D.

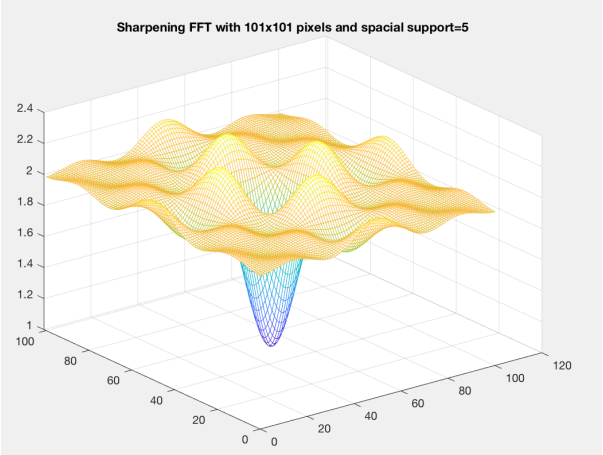

Figure 3.7: Test5: We display magnitude (FFT) of the transformed sharpening filter (slide 44) - the filter has spatial support of 7x7 pixels and it is copied in the middle of a zeros image of 101x101 pixels.

Bibliography

[1] Mathworks documentation on conv2.

[2] Mathworks documentation on fft2.

[3] Mathworks documentation on fftshift2.

[4] Mathworks documentation on fspecial.

[5] Mathworks documentation on medfilt2.

[6] *Brijendra Kumar Joshi Ajay Kumar Boyat.* Signal Image Processing : An International Journal, *volume Vol.6. April 2015.*

[7] *Richard Szeliski.* Computer Vision: Algorithms and Applications. *University of Whashington, 2010.*