UNIVERSITÀ DEGLI STUDI DI GENOVA

Robotics Engineering

Computer vision: first assignment

AY 2022/2023

Lugano Giacomo

S5400573

Tomaiuolo Claudio

S5630055

- **INTRODUCTION:**
  This assignment is devoted to investigating the base concepts of image processing like image noises, different image filtering and fast Fourier transformation for images, all these concepts have been implemented on MATLAB and applied to two different gray scale test images (*Fig.1*).

Fig.1 the two testing images, the right picture is called tree.png (dimension 233x233 pixels), the left one is called i235.png (dimensions 321x321 pixels)

For each assignment question a brief theoretical explanation will be given, followed by the results of such transformation and a brief explanation of the results:

- **1. Add Gaussian (standard deviation = 20) and salt 6 pepper (density=20%) noise to the provided images; display the images; the noisy images and their histograms:**

  - **Theory:** digital images are usually affected by many sources or errors like light fluctuations, sensor noise, quantization effects, finite precision; these errors can affect in different ways the resulting images, more over another great source of error can come from post processing operations actuated on the picture in order to extract information from it.
    We often consider the noise artificially reproducible and additive (*Eq. 1*), meaning that the intensity of light of each pixel $I(x, y)$ is obtained as the real light intensity $s(x, y)$ summed with a certain unknown noise ni:

$$I(x, y) = s(x, y) + ni \qquad \textit{Eq.1}$$

    In particular we will consider two common noises that affect pictures which are:

    - Gaussian noise: variation in the light intensity of the pixels following a Gaussian normal distribution (*Eq. 2*), this is one of the most common noise

types found in digital imaging, it occurs as a results of sensor limitations during image acquisition under low-light conditions, which make it difficult for the visible light sensors to efficiently capture details of the scene [1].

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \qquad \textit{Eq.2}$$

Where:
σ is the standard deviation.
x, y are the position of the pixel along the x and y axes.

- Salt and pepper noise: random occurrence of black (lowest light intensity value) and white (highest light intensity value) pixels in the picture, in this noise type only a few pixels are corrupted, but they are very noisy, it occurs for example in transmitting images over noisy digital links [2], and also by many posts processing algorithms error.

The intensity histogram of an image is a graph showing the number of pixels in it at each different light intensity value, it provides an intuitive tool for evaluating some statistical properties, indeed the left side of the horizontal axis represents the dark areas, the middle the mid-tone values and the right side represent light areas [3]. From it we can distinguish dark or bright and high or low contrast images.

- **Exercise:** in the program the two images are first uploaded thanks to the command `imread()` and translated in a numerical array, then the parameters for the different exercise are required, in particular for this exercise the following parameters are required:
`nstdev`: which is the standard deviation of the Gaussian noise applied to the image.
`nrate`: which is the noise rate in percentual of the salt & pepper noise.
Then both these parameters are passed to the function `noise()` that transform the images applying the noises, that return to the `main` file where their histogram are computed and displayed together with the noised pictures and the original ones to a quick comparison (*Fig.2.1 , Fig.2.2*).

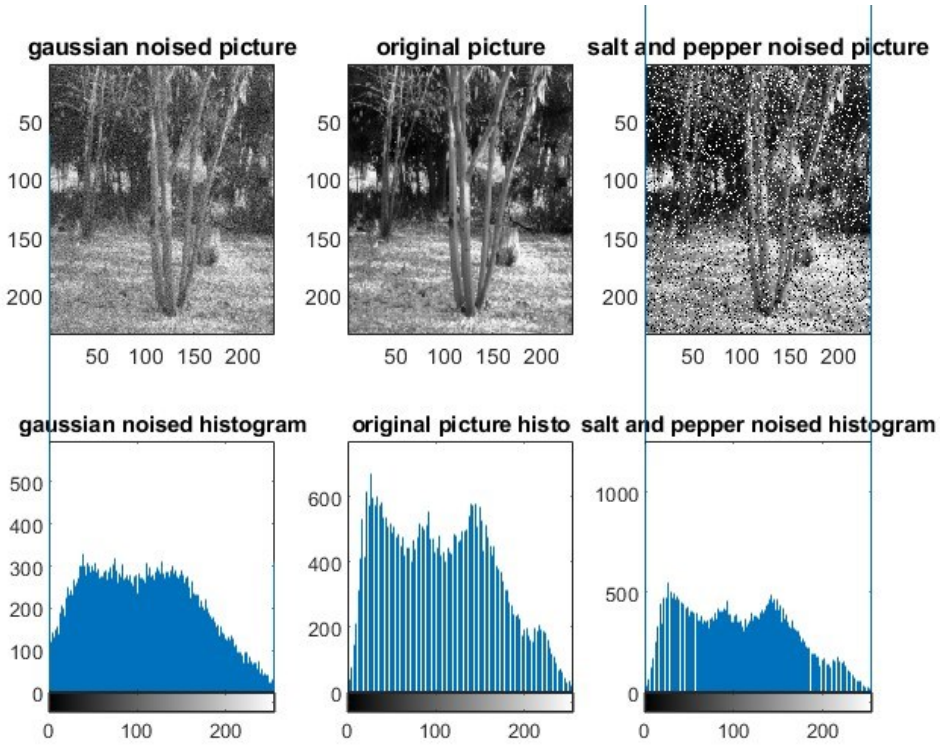

Fig. 2.1 picture tree.png with their histograms, on the right side the gaussian noised picture, on the center the original picture, on the left side the salt & pepper noised picture.

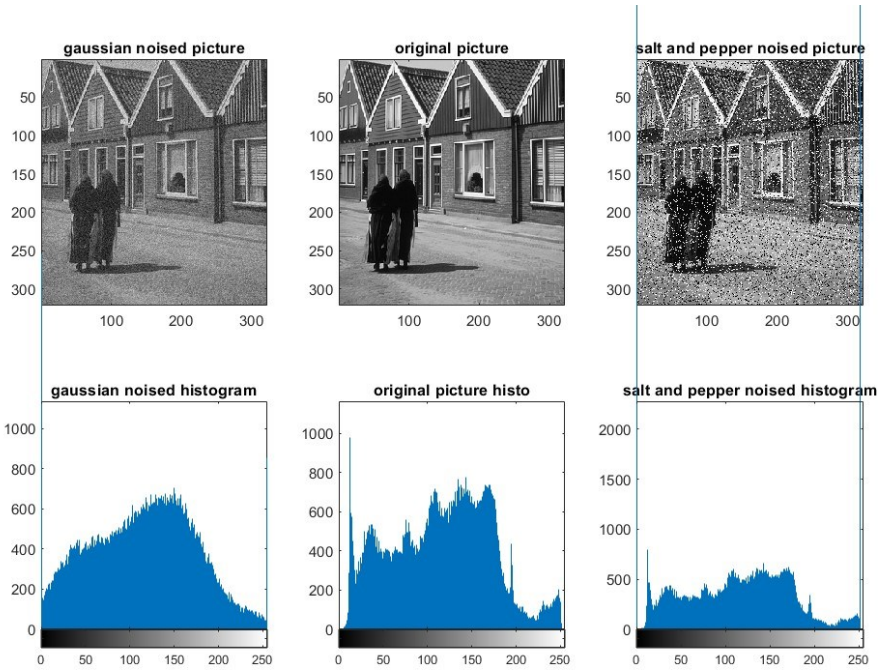

Fig. 2.2 picture i235.png with their histograms, on the right side the gaussian noised picture, on the center the original picture, on the left side the salt & pepper noised picture.

- **2. Remove the noise by using a moving average, a low-pass Gaussian filter and a median filter, use two different spatial supports 3x3 pixels and 7x7 pixels; display the filters by using `imagesc()` and `surf()`; display the resulting images and their histograms:**

  - **Theory:** the operation of filtering is based on different concepts like:
    - Image processing: operations that map pixel values from one image to another one, these operations may be point operators (that depend on the single pixel), neighborhood operators (that depend on the neighborhood pixels) and global operators (that depend on all the pixels of the image).
    - Convolution: used to form new images whose pixels are a weighted sum of original pixel values and its neighborhood ones, in which the weights description in the linear combination is given by the kernel. It express the amount of overlap (similarity) of one function (the kernel) as it is shifted over another function (the image). This operation can be described by the discretized formula (*Eq. 3*):

$$O(i,j) = I * H = \sum_k \sum_l I(k,l)H(i-k, j-l) \qquad \textit{Eq. 3}$$

Between the requested evaluation we've got three filter type:

- Moving average filter: this linear separable filter is applied by the convolution on the image and use a kernel in which any value is equal to 1 divided by the number of cells in it (*Fig.3*), in this way the filtering operation will maintain the light intensity content of the image, while smoothing it by removing the sharp transitions, this can be useful to delete noise and small gaps in contours.
  Practically it produces for each pixel an average of the light intensity of the neighbor pixels contained in an area equal to the same of the kernel.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Fig. 3 example of an average filter kernel (dimensions 3x3)

- Low-pass Gaussian filter: similarly to the moving average filter, this linear separable filter produces a weighted average of pixels in a neighborhood, where the closest pixels to the chosen one have an higher way; this is obtained thanks to a gaussian distribution (*Eq. 2*) for the coefficients of the kernel, and produces results similar to the moving average filter.
  In this case it's important to choose a standard deviation σ of the Gaussian equal at most to one sixth of the kernel dimension in order to obtain a Gaussian and not a paraboloid distribution, then smaller it is smaller will be the contribution of the farer pixels (*Fig. 4*).

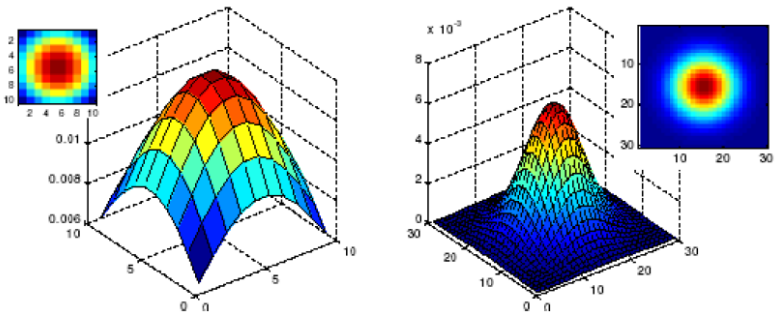

Fig. 4 two Gaussian kernel with different standard deviation, kernel dimension ratio, in particular the left image represents a kernel where the standard deviation is too big compared with the kernel dimension, while the right image represents a correct standard deviation, kernel dimension ratio.

- Median filter: it is a nonlinear filter, in particular it consists in choosing a kernel dimension, then all the neighbor pixels to the chosen one inside the kernel dimension are ordered based for their light intensity and the middle value is selected (*Fig. 5*).
  Obviously this filter doesn't maintain the light content of the image, but it's particularly efficient for removing impulse and salt and pepper noises, while preserving the edges, but it can produces patchy effect.

15	45	68
13	43	56
8	18	42

8 13 15 18 42 43 45 56 68 value assigned to the pixel = 42

Fig. 5 median filter algorithm explanation

- **Exercise:** in the program, after the images upload, the kernel dimension is required `kdim`, since the exercise propose two different spatial support they are stored in a vector; after that the dimension of the kernel together with the noised images are passed to the function `filterim()` which for each one gives as output the average filtered picture `afpicture`, the average kernel `ak`, the Gaussian filtered picture `gfpicture`, the Gaussian kernel `gk` and the median filtered picture `mfpicture`.
  These results are then displayed with their histograms, among them we can observe the most interesting (*Fig. 6,7,8*) on which we can perform some consideration:

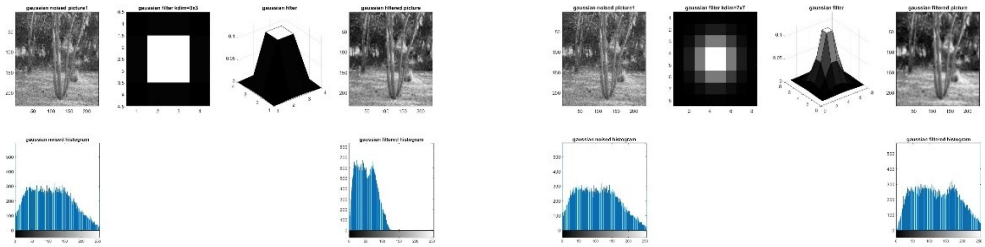

Fig. 6 Gaussian noised picture tree.png filtered by a low-pass Gaussian filter respectively of dimension 3x3 on the left side and 7x7 on the right side.

Form this picture we can observe the filter dimension importance in a filtering operation, indeed it must be tuned to obtain the best effect, in our case we can see how a smaller filter (the 3x3 one) is better for filtering the picture obtaining an image more similar to the original one, the same consideration can be done also for the average filter (*Fig. 7*).

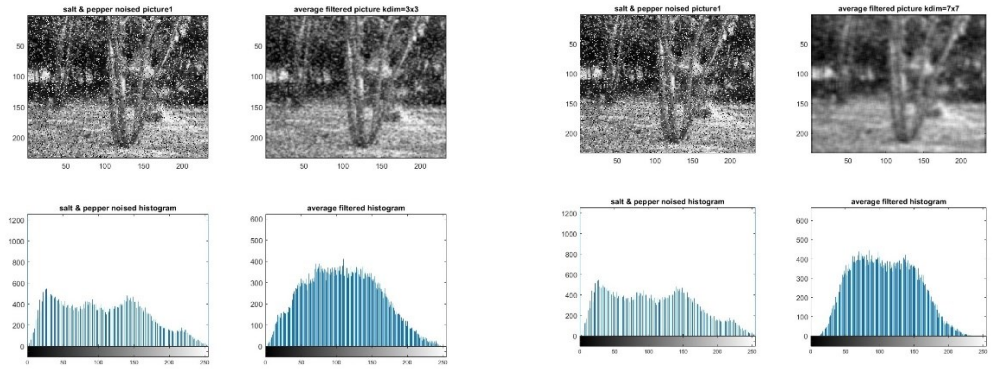

Fig. 7 Salt & pepper noised picture tree.png filtered by an average filter respectively of dimension 3x3 on the left side and 7x7 on the right side.

This fact can be easily explained by considering that a larger filter takes into account more pixels in the neighbor of the considered one, so the result will be more blurred and the sharp transition, introduced not only by the noise but also by the contour of the objects in the scene, will be more smoothed.
This effect is more visible in the average filter since any pixel in the neighbor has the same weight.

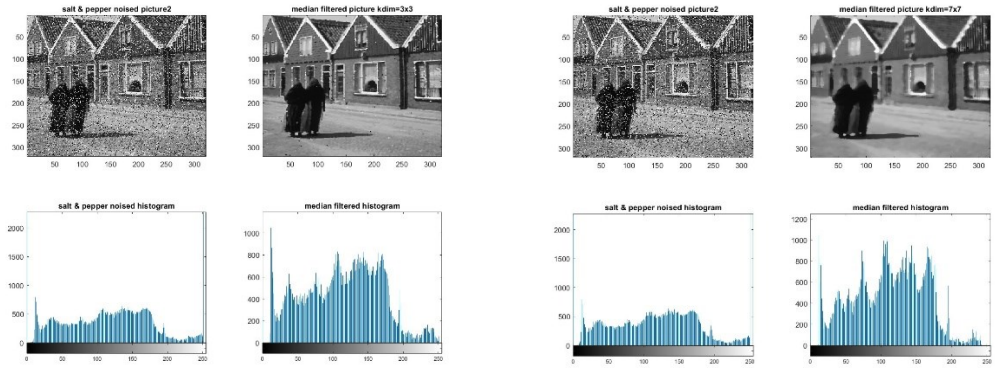

Fig. 8 Salt & pepper noised picture i235.png filtered by a median filter respectively of dimension 3x3 on the left side and 7x7 on the right side.

For the median filter shown in the figure above (*Fig. 8*) the same reasoning holds but is important to notice how the median filter is really efficient in removing the salt and pepper noise, indeed it deletes the impulse noises while preserving the edges but completely discards the spikes.

- **3. Implement the slides 41-45 ”practice with linear filters”, use filters with a spatial support of 7x7 pixels; display the filters by using `imagesc()` and `surf()`; display the resulting images:**

  - **Theory:** the concept explained in the quoted slides are related to the linear filters, in particular we will observe five filters that then we will apply on the images:
    - “Identity filter”: this linear filter has a kernel which is composed by zeros except for the central term that is equal to 1 (*Fig.9*), applied to the picture by the convolution it simply maintains the picture unaltered.

0	0	0
0	1	0
0	0	0

Fig. 9 Example of an “Identity filter” kernel of dimension 3x3.

    - “Shifting filter”: this linear filter has a kernel composed by zeros except for the extreme left term on the central row (*Fig. 10*), applied to the picture by the convolution it shift the whole picture left by a number of pixels equal to half the width of the kernel, indeed in the convolution for each pixel on which the kernel is applied it is replaced with the one at his right at half the kernel dimension distance.

0	0	0
1	0	0
0	0	0

Fig. 10 Example of an “Shifting filter” kernel of dimension 3x3, this filter will shift the picture in the left direction of one pixel.

    - Average filter: as we can see this filter is exactly an average filter, where any element of the kernel is equal to 1 divided the square of the filter dimension (*Fig. 11*), in fact its effect is to smooth the image removing the high frequency content of the image.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Fig. 11 Average filter kernel of dimension 3x3.

    - Sharpening filter: this linear filter can be seen as a linear combination of an “Identity filter” multiplied by a coefficient less an average filter (*Fig. 12*), its effect will be to highlight the component of the pixel on which is applied by subtracting the components of the neighbor pixels increasing the contrast.

$$a^* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Fig. 12 sharping filter linear composition.

A similar result can be obtained using the following filter/technique:

- Sharping and Details: the linear filters seen till now can be linearly composed to create different results, in particular if we subtract pixel by pixel the light intensity of the smoothed picture from the original picture, we will obtain an image where only the details are highlighted.
  If then we add pixel by pixel at the original picture the details one multiplied by a coefficient, we will obtain the same result of the Sharpening filter; we can see this technique in the following exercise explanation.

- **Exercise:** in this case, after the images are uploaded, they are passed to the function `lifilter()` in which the kernel dimension for all the filters is already set to 7, in it the different filter are applied returning as output the "Identity" filter kernel K1 and filtered image `f1picture`, the "shifting" filter kernel K2 and filtered image `f2picture`, the Average filter kernel K3 and filtered image `f3picture`, the Sharpening filter kernel K4 and filtered picture `f4picture`, the Average filter used in the Sharping and Detail technique K5, the detail image `detail` and Sharpened picture `sharpicture`. Then all these outputs are displayed (*Fig. 13, 14, 15, 16, 17*).

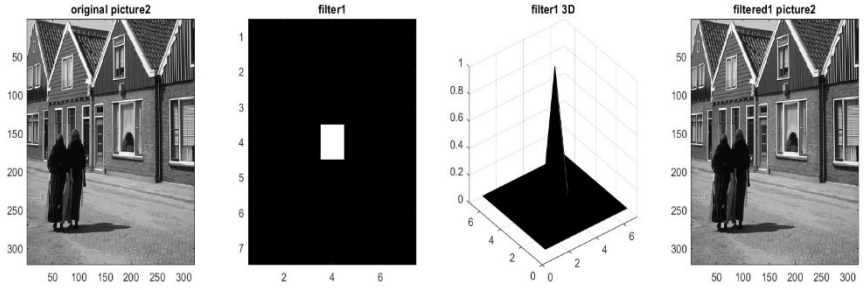

Fig. 13 "identity" filtered picture i235.png

As we can see the effect of this filter is to maintain the picture unaltered.

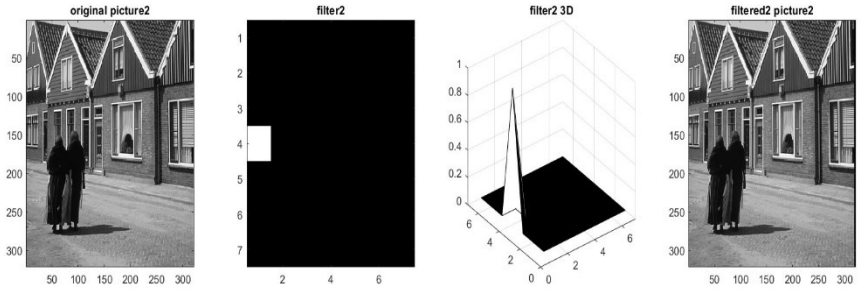

Fig. 14 "shifting" filtered picturei235.png

In this case since the kernel has a dimension of 7x7 and the 1 in the kernel is placed in the central raw and extreme left column the resulting image will be shifted of 4 pixels at its left as we can see from the results `filtered2 picture2`.

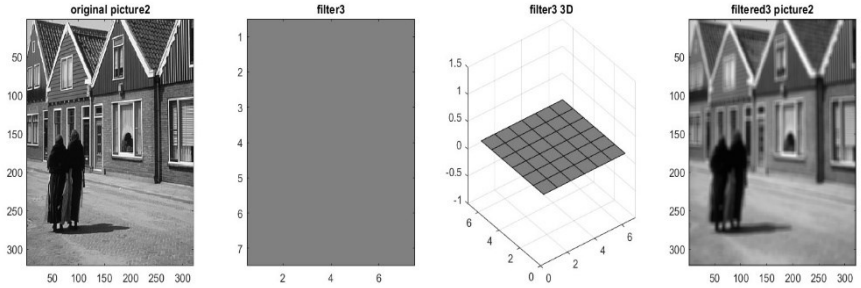

Fig. 15 Average filtered picture i235.png

As we have said this filter correspond to the Average one and so the already done considerations hold.

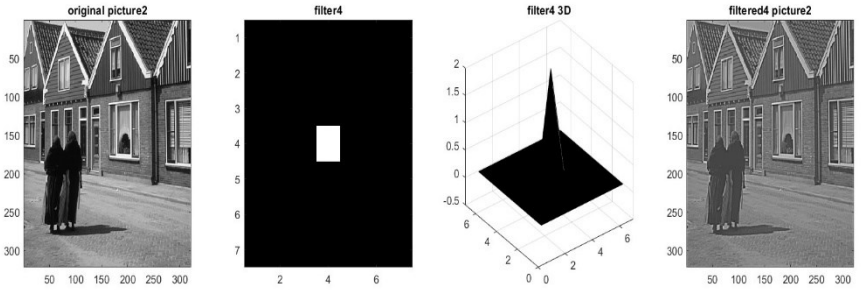

Fig. 16 Sharpened picture i235.png

As we can see the effect of the Sharpening filter is to increase the contrast of the image, using a kernel which decreases the light component of the neighbor pixels and increase the value of the chosen one.

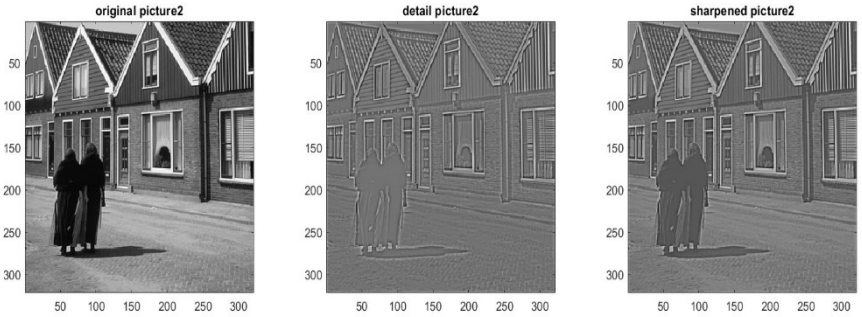

Fig. 17 Detail and sharpened picture of image i235.png

As we can see here the procedure consists of smoothing the original picture and subtract the result at the original picture to obtain the detail image where all the details of the image are highlighted, then we can sum the original picture with the detail one, multiplied by a coefficient (in our case a=2) to obtain the sharpened image.

We can also compare the last obtained sharpened picture with the previously one displayed in figure 16, as we can see the two are really similar, and the difference can be reduced to zero by choosing the right multiplying coefficient a.

- **4. Apply the Fourier transform (FFT) on the provided images, display the magnitude (Log) of the transformed images; display the magnitude of the transformed low-pass Gaussian filter (spatial support of 101x101 pixels with sigma=5); display the magnitude of the transformed sharping filter, slide 44 (the filter has a spatial support of 7x7pixels, copy it in the middle of a zeros image of 101x101 pixels).**

  - **Theory:** the Fourier transform is a powerful tool that allows to translate a signal from the space domain to the frequency one decomposing it into its sine and cosine components [4], the same procedure can be applied on images since they can be seen as 2D signals.
    Let we consider an image $g(x, y)$, then its Fourier transform can be computed by the formula (*Eq. 4*):

    $$\Im\big(g(x,y)\big)(u,v) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(x,y)e^{-i2\pi(ux+vy)}\,dxdy \qquad \textit{Eq. 4}$$

    Where the result is a complex valued function in terms of the frequencies (u, v) and the real component is called magnitude while the complex component is called phase.
    We can also see the discretized Fourier transform which for an image of size NxN is given by (*Eq. 5*):

    $$\Im\big(g(x,y)\big)(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} g(x,y)e^{-i2\pi(\frac{ux}{N}+\frac{vy}{N})} \qquad \textit{Eq. 5}$$

    Where the exponential term is the basis function corresponding to each point $\Im\big(g(x,y)\big)(u,v)$ in the Fourier space; the equation can be interpreted as: the value of each point $\Im\big(g(x,y)\big)(u,v)$ is obtained by multiplying the spatial image with the corresponding base function and summing the result [4].
    The Fourier transform is a global operator since the result depends on the whole image and in it the phase component encloses the spatial structure of the image, however most of the image methods are based on the magnitude.

  - **Exercise:** after the two images are uploaded a dedicated low-pass Gaussian filter and Sharpening filter of the requested dimension are created, then both the images and the filters kernels are passed to the function `FastFourierrT()` , in it we can see how the image is first transformed in a 2D Fourier transform by the function `fft2()`, then the absolute value of it is passed to the function `fftshift()` which rearrange the Fourier transform by shifting the zero-frequency component to the center of the array and finally the natural logarithm of the output of such operation is displayed (*Fig. 18, 19, 20, 21*):

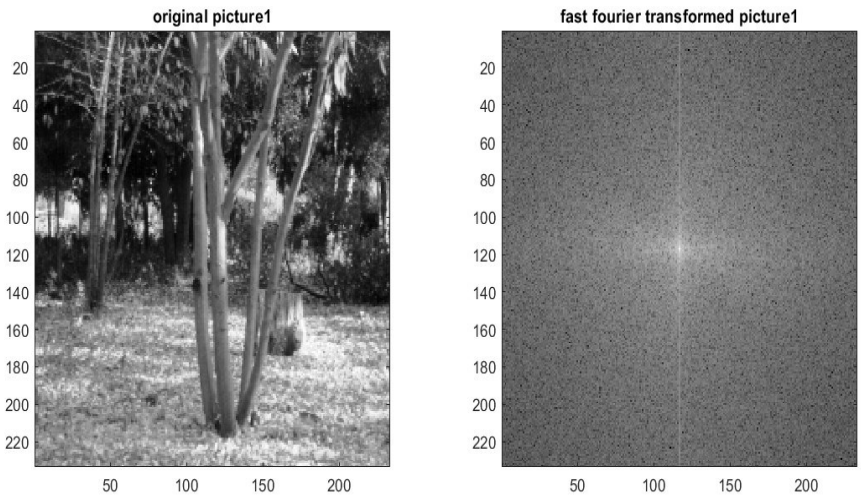

Fig. 18 Fast Fourier transform of image tree.png

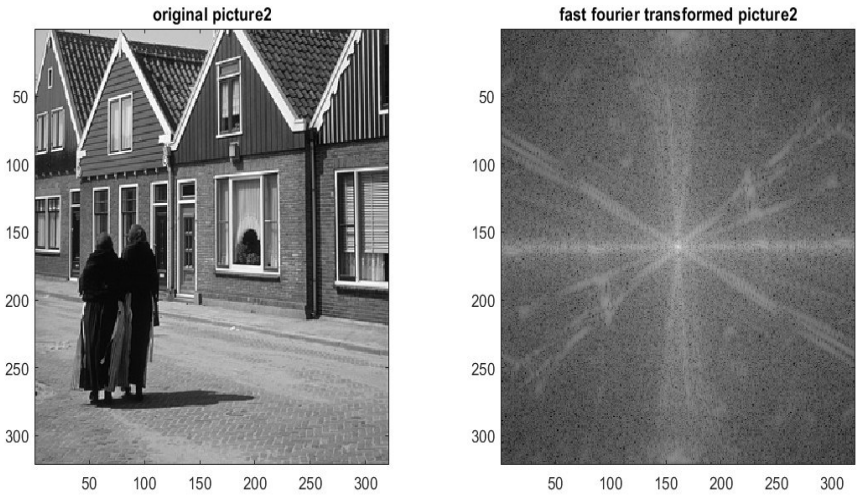

Fig. 19 Fast Fourier transform of image i235.png

As we can see from both fast Fourier transformed images, most of the information aren't clearly displayed even if the result of this transformation contains most of the information of the geometry structure of the spatial domain.

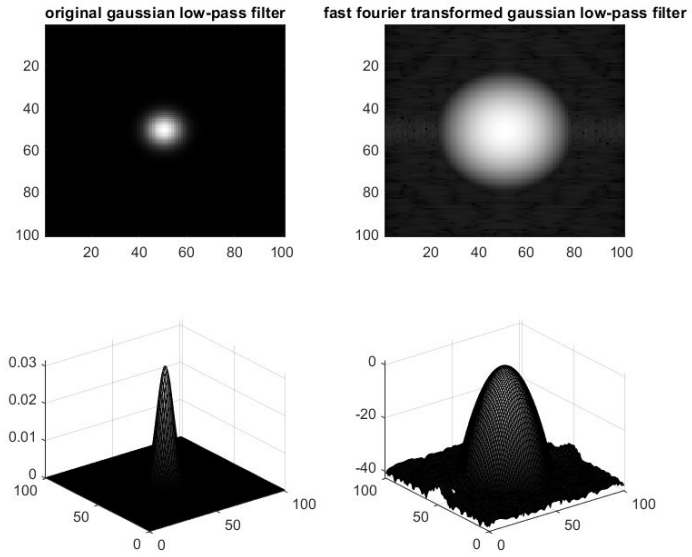

Fig. 20 Gaussian low-pass filter and its fast Fourier transform.

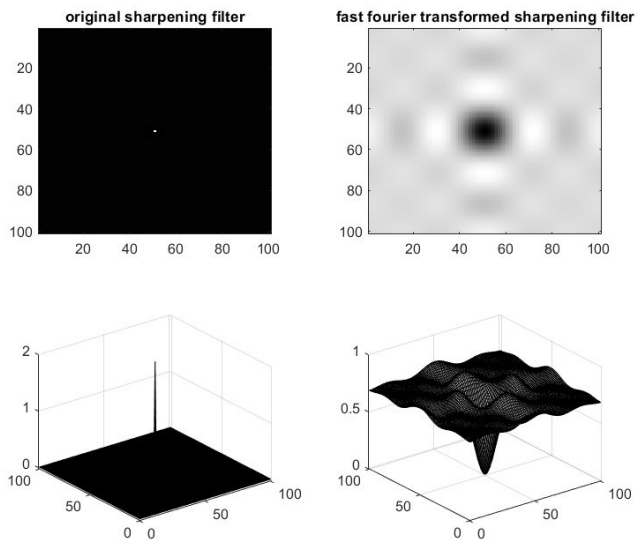

Fig. 21 Sharpening filter and its fast Fourier transform.

The same consideration done for the fast Fourier transformed images of tree.png and i235.png can be done for the fast Fourier transformed filters, in it we can also note how a small surface in the spatial domain becomes a big one in the frequency domain and vice versa, since the relation between space and frequency is inversely proportional.

- **References:**

[1] https://www.vision-systems.com/home/article/14174546/filtering-techniques-eliminate-gaussian-image-noise
[2] https://www.sciencedirect.com/topics/engineering/pepper-noise
[3] https://en.wikipedia.org/wiki/Image_histogram
[4] https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm