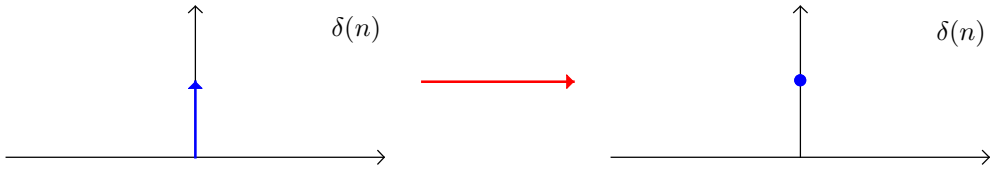
Spatial Filtering

BY MARTINO FERRARI

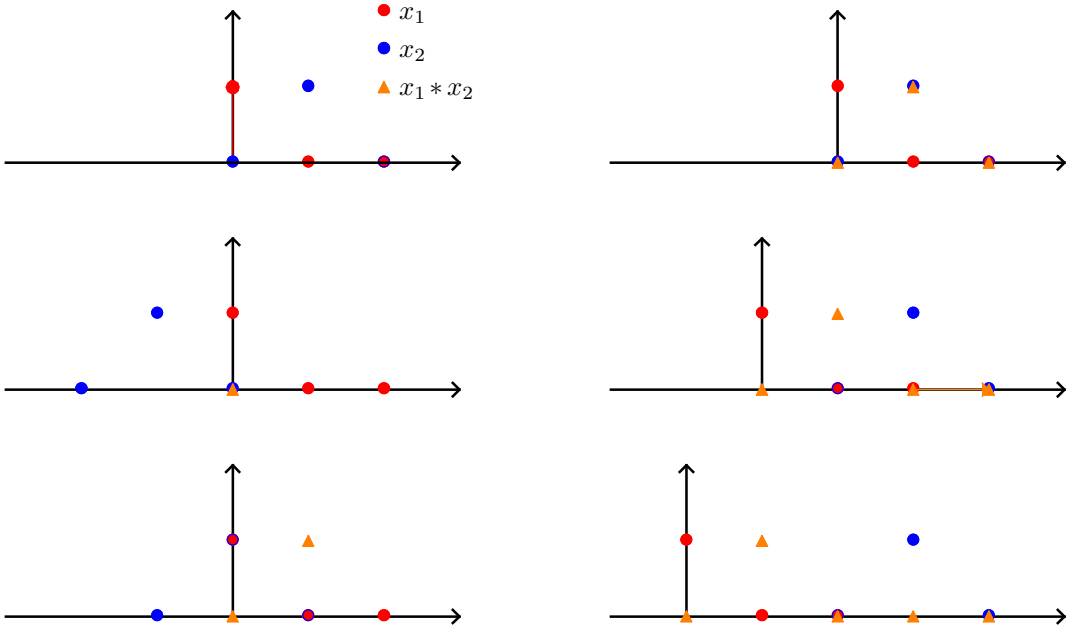
1 On Spatial Filtering

In the following exerscies I will use a disk instead of the classical Dirac delta graphism beacouse is much simpler.

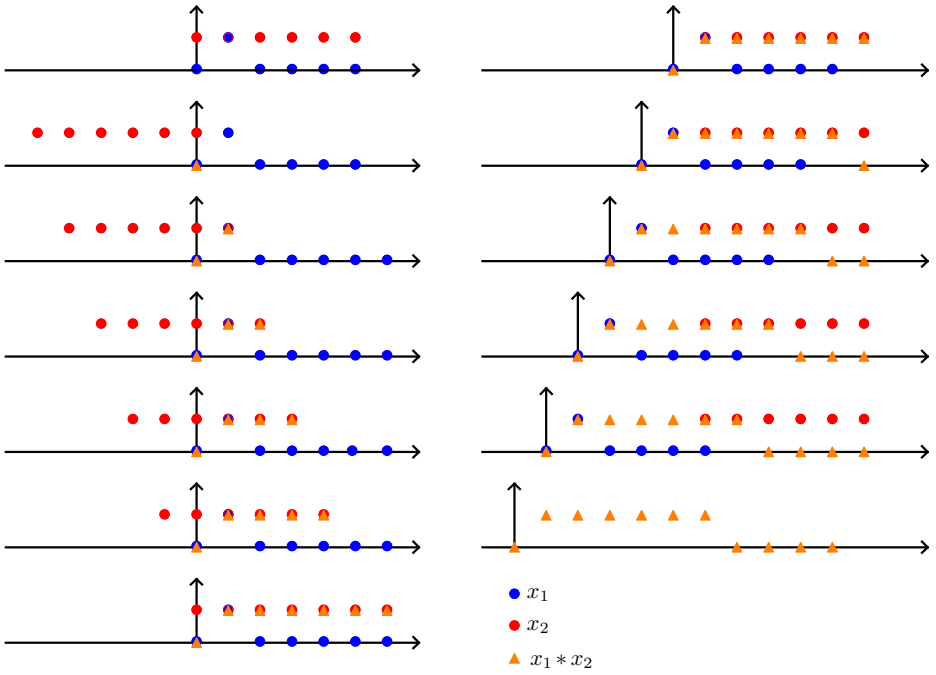


Exercise 1

Manually convolve the two signal $x_1 = [1, 0, 0]$ and $x_2 = [0, 1, 0]$:

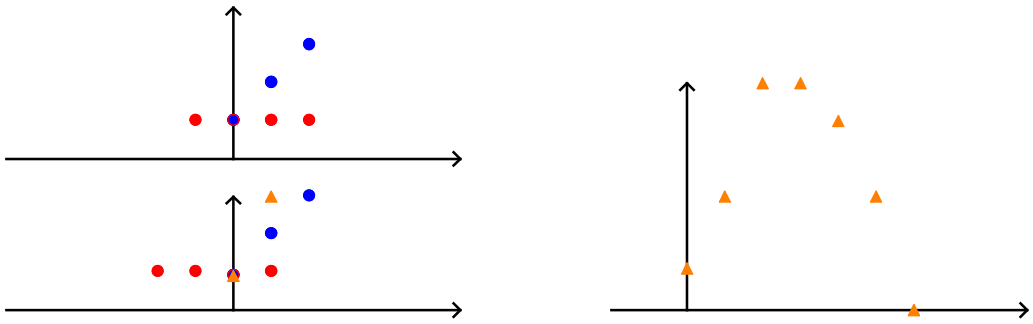


Manually convolve the two signal $x_1 = [0, 1, 0, 0, 0, 0]$ and $x_2 = [1, 1, 1, 1, 1, 1]$:



Exercise 2

Manually convolve the functions $h(n) = \delta(n) + 2\delta(n-1) + 3\delta(n-2)$ with the function $x(n) = \delta(n) + \delta(n+1) + \delta(n-1) + \delta(n-2)$:



2 Spatial Filtering

Exercise 1: Low Pass Filtering

After implementing the *averaging filter*:

$$h = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

I tested it on the image *peppers.png*:

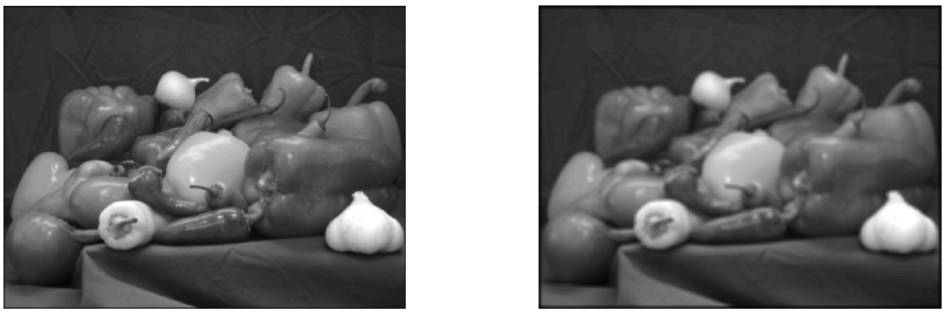


Figure 1. *peppers.png* convolved with averaging filter (size 3×3 and 6×6)

The result is a blurred image of the original version where the high frequency components of the original image are removed from the results. More the filter is big more the result is accentuate (more blur and less high frequency components preserved).

Similary for the gaussian filter:

$$h = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Figure 2. *peppers.png* convolved with gauss filter size 3×3

Similarly to increase the effect both in the spatial and frequency domain it's enaugh to increase the filter kernel size.

Exercise 2: Order-Statistics Filtering

In this second exercise we will try to use different filters to suppress some *salt and pepper* noise.

Figure 3. *peppers.png* with *salt and pepper* noise

The first filter used is the averaging filter with both size 3×3 and 5×5.

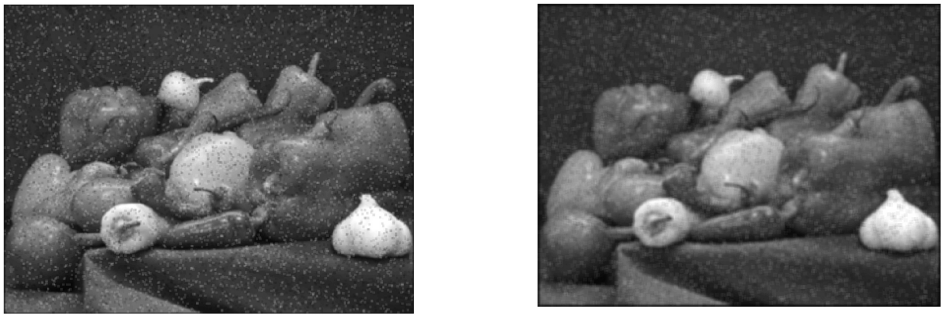


Figure 4. Convulition of the averaging filter (3×3 and 5×5) with the noisy image

The quality of the image after an avereaging filter doesn't improve much, event if the local effect of the noise is less important (now there are no more pixels at 255 or at 0) the noise has been distributed to the neighborhood. With bigger windows the image becomes to blur to be used.

The second filter used is the median filter:

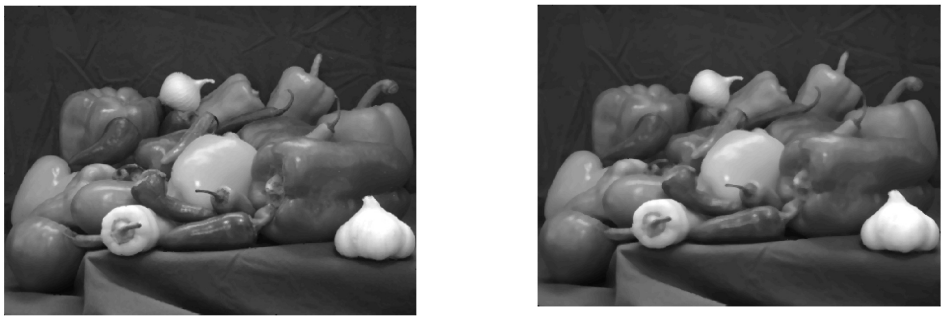


Figure 5. onvulition of the median filter (3×3 and 5×5) with the noisy image

In this case the result is muche better and the filtered image with the 3×3 window size give very good results, however increasing the size f the windows results in some color artifict and loss in the sharpness. This high quality results are due the fact that the filter is taking in account the ranking of the neighnorhood pixels avoiding the distribution of the original noise.

Exercise 3: High Pass Filtering

Before starting to really use the high pass filter we were asked to determine the first and second derivative of the discrete signal:

$$f = [5, 7, 6, 5, 4, 3, 2, 1, 0, 5, 4, 3, 2, 1, 0, 0, 0, 6, 0, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0, 0, 7, 7, 7, 7]$$

The first and second discrete derivate are defined as:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

So in our case the first derivate is:

$$\frac{\partial f}{\partial x} = 2, -1, -1, -1, -1, -1, -1, -1, 5, -1, -1, -1, -1, -1, 0, 0, 6, -6, 0, 0, 0, 1, 2, -2, -1, 0, 0, 0, 7, 0, 0, 0$$

$$\frac{\partial^2 f}{\partial x^2} = -3, 0, 0, 0, 0, 0, 0, 6, -9, 0, 0, 0, 0, 1, 0, 6, -12, 6, 0, 0, 1, 2, -4, 1, 1, 0, 0, 7, -7, 0, 0$$

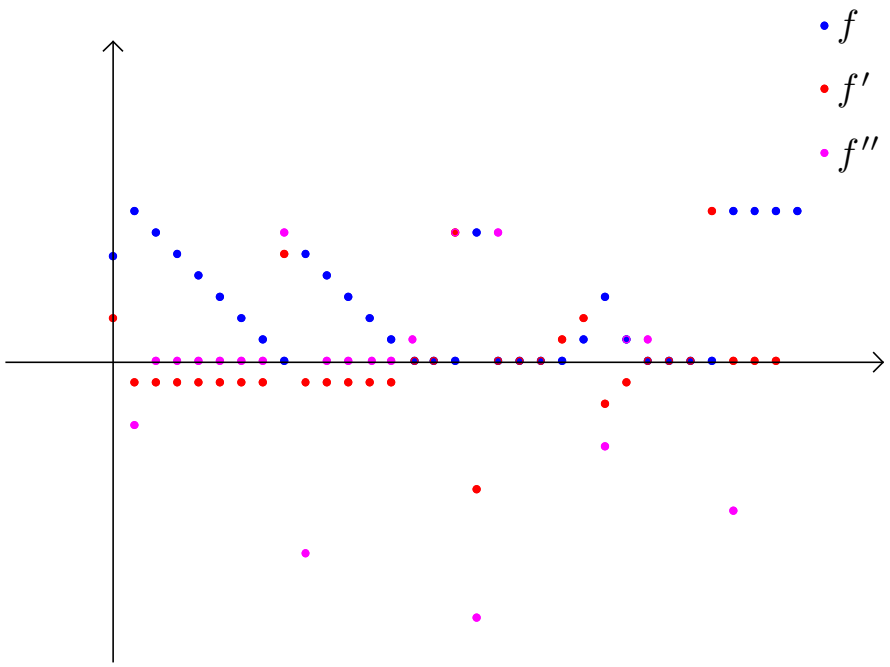


Figure 6. f, f' and f'' displayed on single plot

As previsible the first derivative shows the direction of the trend (ex: if it decrease of one every step it will have value -1) while the second derivative shows the changment in direction (ex: $[...,0,0,6,0,0,...] \rightarrow [...,6,-6,0,...]$). The second derivative can be especially usefull when looking for borders as detect discontiuities.

Exercise 4: The Laplacian

A first implementation of the laplacian based on the equation:

$$\nabla^2 f = (f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1))-4f(x,y)$$

that has kernel matrix:

$$h = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

The results on the image *peppers.png* is the following:

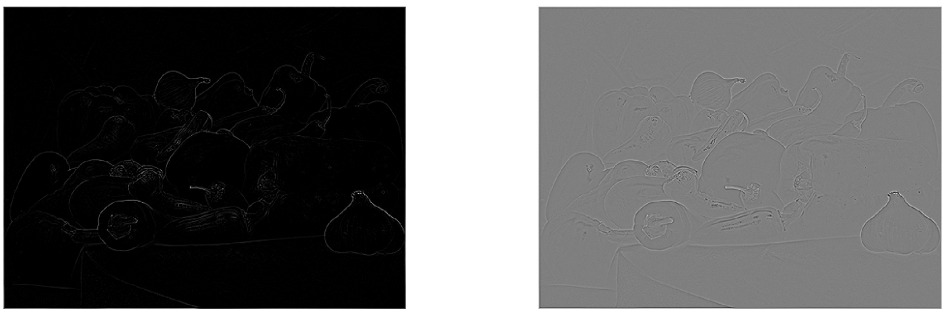


Figure 7. first implementation of the Laplacian on the image (without and with negative components)

As expected the borders of the objects are well evidencied.

Then after implementing the others 3 possible variants, I get the following results:

Figure 8. alternative implementations of the Laplacian filter

The first alternative give similar results of the previus method (as the kernel matrix is the same but negative) the third and forth method (as well very similar togheter) shows more details and borders than the other two methods as they take account as well of the diagonal neighboors.

The difference of the sign of the kernel impact only as 1 pixel shift in the visualization.

We can use the Laplacian filter to sharpen the image:

$$g(x,y) = f(x,y) \pm \nabla^2 f(x,y) \quad (+ \text{ if the kernel is positive, } - \text{ otherwise})$$

Figure 9. *peppers.png* with 2 version of Laplacian filter as sharpener filter

Has expected the diagonal version have a very strong effect on the image (image on the rigth) while the first filter is more soft and has more realistic results (image on the left).

The kernels of the used sharpering filters are:

$$h_1 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \qquad h_2 = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Exercise 5: Unsharp masking

The *Unsharp masking* is a process used to enhance the quality of the picture and increasing the sharpness of it. The filter is:

$$f_{\text{um}}(x, y) = f(x, y) - \bar{f}(x, y)$$

where $\bar{f}$ is a blurred version of the original image.

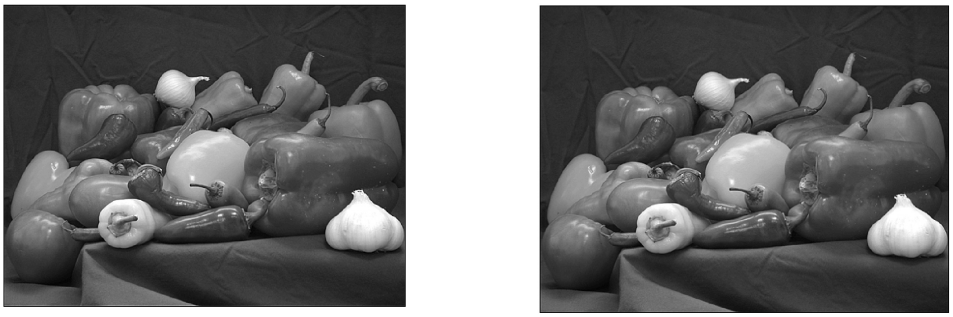


Figure 10. *peppers.png* after unsharping mask with averaging and gaussian filter

As it possible to see the resulting images have a much more natural look than the previous one (figure 9), and the differences between the two version (using averaging or gaussian) is very small.

3 The Covariance Matrix and the Eigenvectors

In this exercise we were asked to generate 100 realization of two random variable with pdf $\mathcal{N}(0, 1)$:

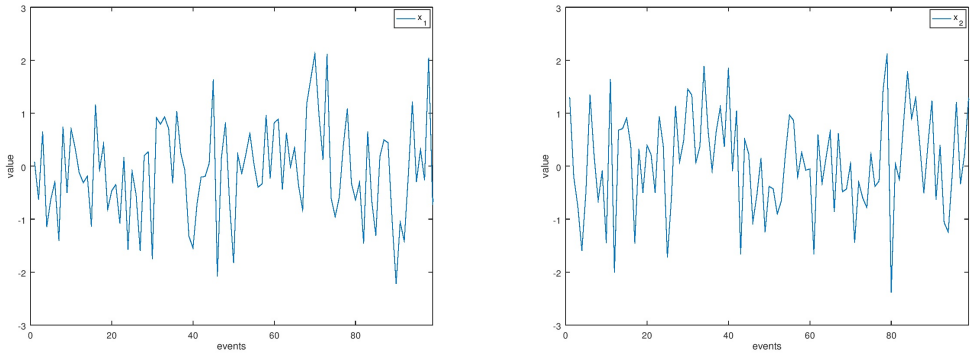


Figure 11. Realization of two random variable X_1 and X_2

Its expectations are:

$$E[X_1] = 0.0081620 \qquad E[X_2] = 0.063321$$

and its variances are:

$$\text{var}[X_1] = 0.91647 \qquad \text{var}[X_2] = 1.0173$$

This results will become more close to the real mean and variance of the pdf with more realizations of the two variables, with infinite samples we will have the exact values.

The *covariance* matrix of this to variables is:

$$C_Y = \begin{pmatrix} 0.9165 & -0.1587 \\ -0.1587 & 1.0173 \end{pmatrix}$$

And the plot of it will result in:

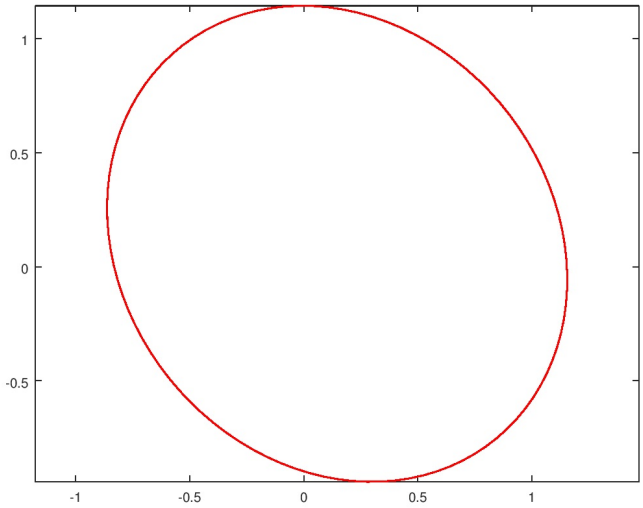


Figure 12. *plotcov* of C_Y

As expected the covariance between X_1 and X_2 is close to 0 as they are indipendent.

After applying a linear transformation $A = \begin{pmatrix} \sqrt{2} & 0 \\ 0 & 1/\sqrt{3} \end{pmatrix}$ to $Y = [X_1; X_2]$ we obtain the following results:

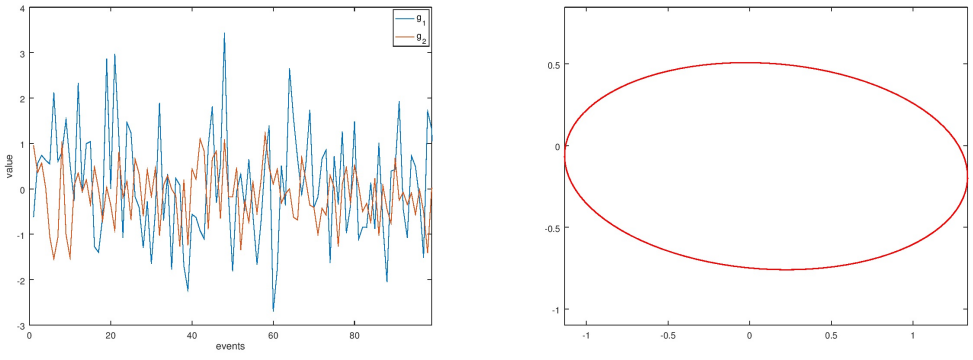


Figure 13. G_1, G_2 and its *plotcov*

As expected both the signal and the covariance matrix has been deformed by the matrix A, now the ellipse has x axis bigger than the y axis as the transformation has increased the amplitude of X_1 and decreased the one of X_2. This means that the expected values have not changed while the variances have been multiplied by the scale factor.

We were asked to applay a second linear transformation:

$$A = \begin{pmatrix} \cos\left(\frac{\pi}{6}\right) & -\sin\left(\frac{\pi}{6}\right) \\ \sin\left(\frac{\pi}{6}\right) & \cos\left(\frac{\pi}{6}\right) \end{pmatrix}$$

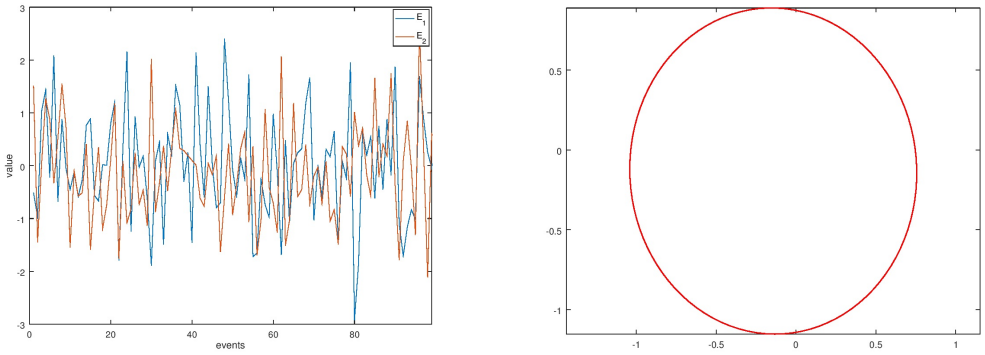


Figure 14. E_1, E_2 and its *plotcov*

This second linear transformation has different results as is mixing the values of the two signal (as the matrix is not diagonal).

The eigen vectors and eigen values of G are:

$$V_G = \begin{pmatrix} -0.998 & 0.059 \\ -0.059 & -0.998 \end{pmatrix} \qquad D_G = \begin{pmatrix} 1.850 & 0 \\ 0 & 0.427 \end{pmatrix}$$

In this case we have quite high separation between the two components.

While for E are:

$$V_E = \begin{pmatrix} -0.269 & -0.963 \\ 0.963 & -0.269 \end{pmatrix} \qquad D_E = \begin{pmatrix} 1.323 & 0 \\ 0 & 0.986 \end{pmatrix}$$

The eigen values in this case are more similar meaning that the information is more uniformly distribuited, meaning it would be harder to extract.

4 Feature Detector

In this last part we were asked to implement a simple corner detector based on the derivative filters:

$$A_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \qquad A_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

I tested this filter first on the image *im02.jpg* (shown in figure 15) provided.

Figure 15. *im02.jpg*

In particular we were supposed to compute the corner response matrix r as:

$$r = \frac{L_{xx} . * L_{yy} - (L_{xy})^2}{L_{xx} + L_{yy}}$$

Where the matrix L_{xx}, L_{yy} and L_{xy} are the blurred version (using a gaussian filter) of the derivative products (I_{xx}, I_{yy} and I_{xy}):

$$\begin{aligned} L_{xx} &= G_{\sigma^w} * I_{xx} \\ L_{yy} &= G_{\sigma^w} * I_{yy} \\ L_{xy} &= G_{\sigma^w} * I_{xy} \end{aligned}$$

Where G_{σ^w} is the kernel of a gaussian filter with $\sigma = 3$ for this first part of the exercise.

The results of such operation is the following:

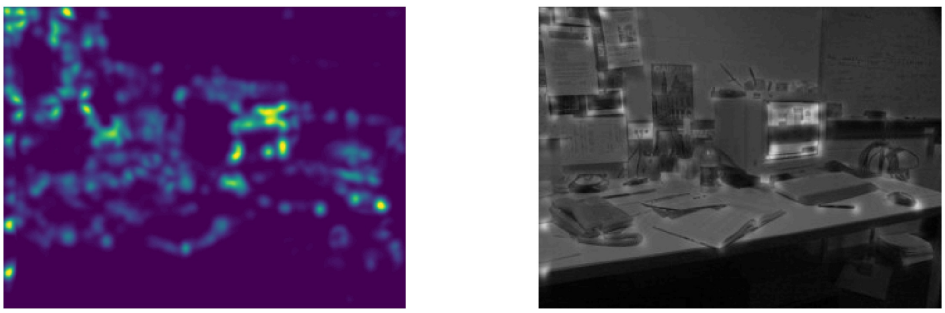


Figure 16. r of the image *im02.jpg* and r overposed to the original image

A we can see the matrix r evedinciate the reagion of the image where a corner is present, with a simple algorithm (segmenting the image in smaller reagion and finding the local max) it is possible to find this corners:

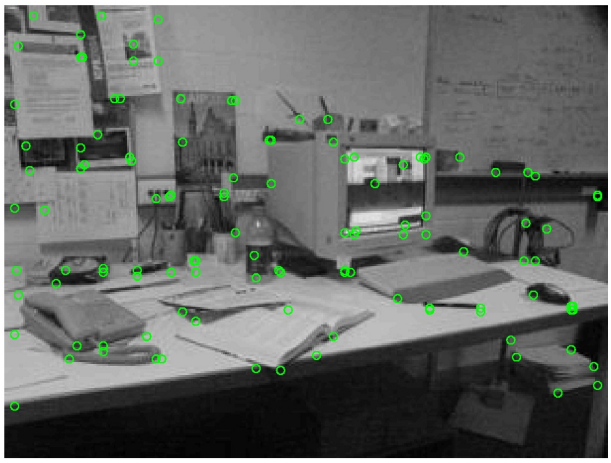


Figure 17. Simple corner detector

The corner identifier algorithm is very simple and I'm sure that can be vastly improved in the future.

In the second part of the exercise we were asked to try different values of σ on the image *square.jpg*:

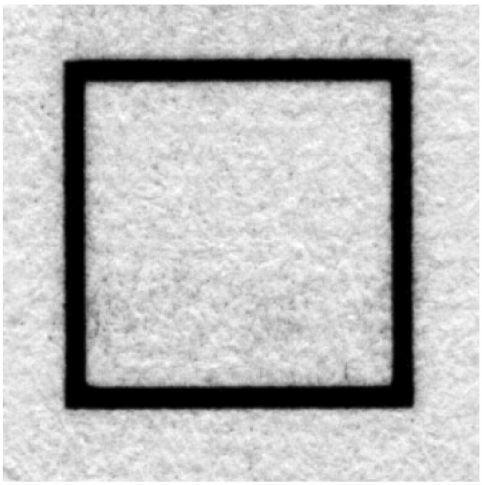


Figure 18. *square.jpg*

The results are the follwing:

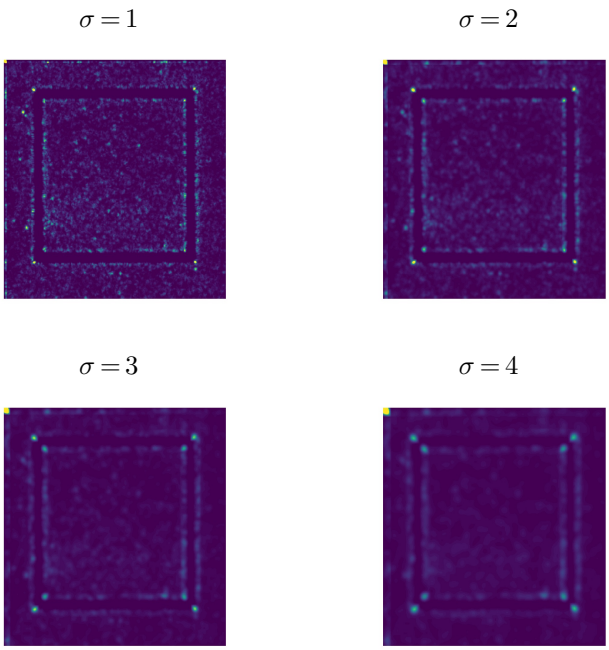


Figure 19. Matrix r for different values of σ

As expected with bigger values of σ the results is more pricse as the high frequency components (the paper structure in this case) is attenuated by the gaussian filter.