Fourier

BY MARTINO FERRARI

Exercise 1

Show that the DC component of a signal $f(x)$ with zero mean is 0:

$$E[f(x)] = 0 = \frac{1}{M}\sum_{x=0}^{M-1} f(x) = 0$$

$$\mathcal{F}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)e^{-\frac{i2\pi ux}{M}}$$

The DC component stands for the continuous components (or with frequency 0) and is equivalent to $\mathcal{F}(0)$:

$$\mathcal{F}(0) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)e^{-\frac{i2\pi 0x}{M}} = \frac{1}{M}\sum_{x=0}^{M-1} f(x)\cdot 1 \equiv E[f(x)] = 0$$

Exercise 2

Given $f(x) = [5, 4, 3, 2, 1]$:

- manually compute the DFT of *f(x)*:

$$\mathcal{F}(u) = \frac{1}{5}\left(5\cdot e^{0} + 4\cdot e^{-\frac{2i\pi u}{5}} + 3\cdot e^{-\frac{4i\pi u}{5}} + 2\cdot e^{-\frac{6i\pi u}{5}} + 1\cdot e^{-\frac{8i\pi u}{5}}\right)$$

$$\mathcal{F}([0,1,2,3,4]) = [3, 0.5 - 0.69i, 0.5 - 0.16i, 0.5 + 0.16i, 0.5 + 0.69i]$$

- this gave me the same result that using the *fft* function of *numpy*, however I had to normalize the results dividing the output for the number of samples.

- if $f(x) = [5, 4, 3, 2, 1, 5, 4, 3, 2, 1]$ the DFT becomes:

$$\mathcal{F}(u) = \frac{1}{10}\left(5\cdot e^{0} + 4\cdot e^{-\frac{2i\pi u}{10}} + 3\cdot e^{-\frac{4i\pi u}{10}} + 2\cdot e^{-\frac{6i\pi u}{10}} + 1\cdot e^{-\frac{8i\pi u}{10}} + 5e^{-\frac{10i\pi u}{10}} + ...\right)$$

$$\mathcal{F}([0,1,2,...,9]) = [3, 0, 0.5 + 0.69i, 0, 0.5 + 0.16i, 0, 0.5 + 0.16i, 0, 0.5 + 0.69i, 0]$$

  As the frequence of each signal doubled the DFT has now 0 between each component.

$$\mathcal{F}(u) = \frac{1}{10}\left(5\cdot e^{0} + 4\cdot e^{-\frac{2i\pi u}{10}} + 3\cdot e^{-\frac{4i\pi u}{10}} + 2\cdot e^{-\frac{6i\pi u}{10}} + 1\cdot e^{-\frac{8i\pi u}{10}} + 5e^{-\frac{10i\pi u}{10}} + ...\right)$$

Exercise 3

In this exercise we were asked to analyze the magnitude and the phase of an image, I chose to use *lena.png* as source image:

Figure 1. source image: *lena.png*

After converting it in gray scale I transformed the image in the frequency domain using the function *fft2*:

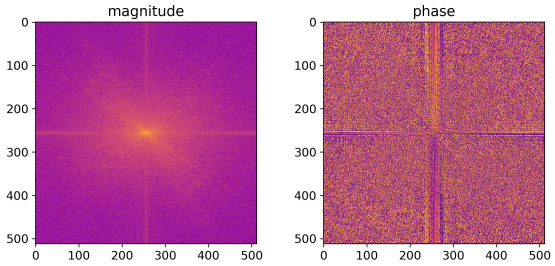

Figure 2. plot of magnitude and phase in the frequency domain

Then we were asked to transform back to the spatial domain using the magnitude and the phase independently:

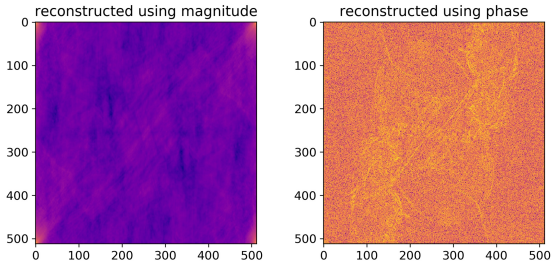

Figure 3. plot of the reconstructed images

From the magnitude reconstruction is impossible see any element of the original images, has the energy are not distribute ed in the right positions. From the phase one is possible to distinguish the borders of *Lena*, this because in the borders also the phase changes.

Exercise 4

After implementing the correlation function in the frequency domain:

$$f \star g = \mathcal{F}^{-1}(\mathcal{F}(f)^* \cdot \mathcal{F}(g))$$

I used it to compute the auto-correlation of the *cross.png* image:

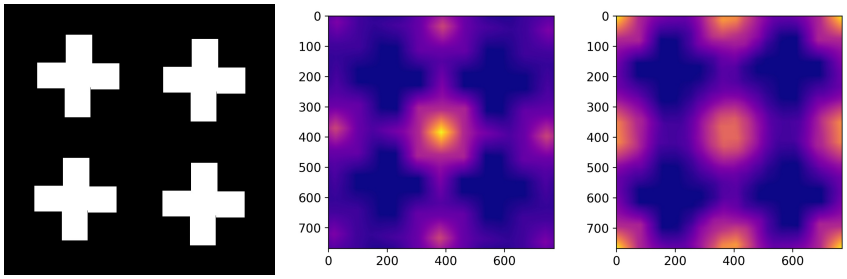

Figure 4. Results of the different correlation methods

The results are quite similar however the performance of the correlation in the frequence domain is around 10 times the one of *xcorr2*.

I tested the same code on *cameraman.bmp* to see the results:

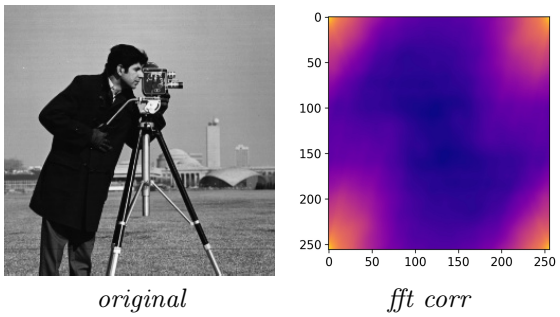

Figure 5. autocorrelation of *cameraman.png*

The autocorrelation in the center is higher as the the cameraman pixels are highly correlated.

Exercise 5

In this exercise we were asked to implement the *gaussian low pass filter* and the *difference of gaussian high pass filter* in the frequency domain:

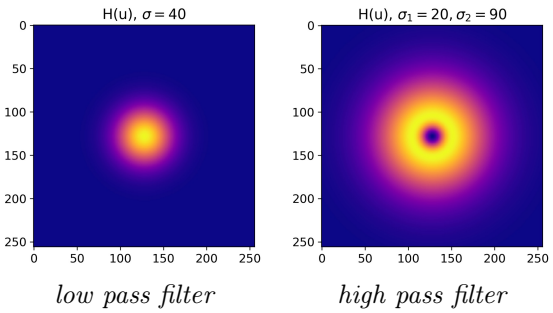

Figure 6. filters in the frequency domain

Then I chose to app lay it on the image *bird.png*:

Figure 7. source image: *bird.png*

Applying the low pass filter will result as following:

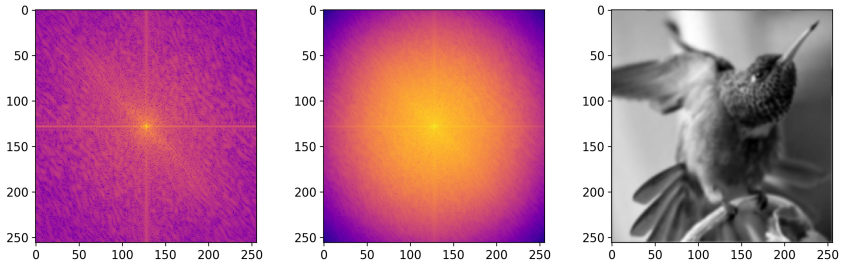

Figure 8. low pass filter applied to *bird.png*

Instead the high pass filter gave this results:

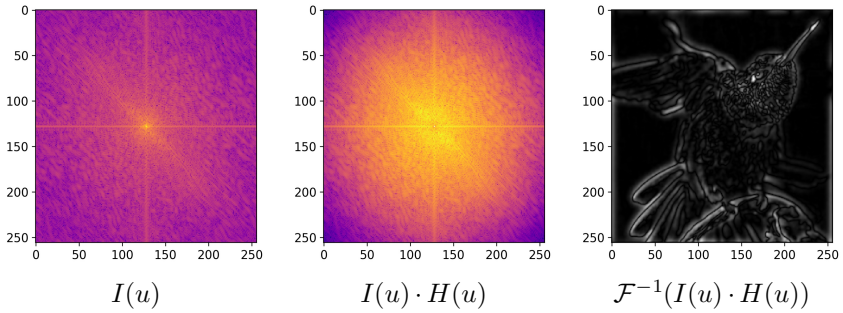

Figure 9. high pass filter applied to *bird.png*

Exercise 4

In this exercise we were asked to recover one image of choice. In my case I chose the image *csi_ 1.png*:

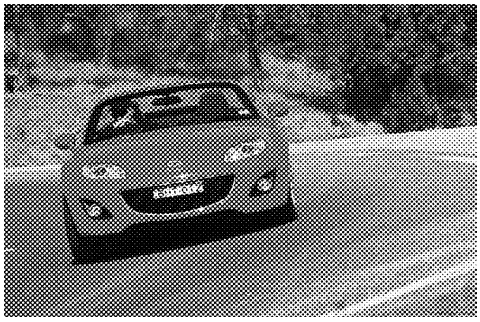

Figure 10. source image: *csi_ 1.png*

The first step is to transform it and display the magnitude in the frequency domain as there is clearly a pattern in the printing of the image that should be evident in the frequency domain:

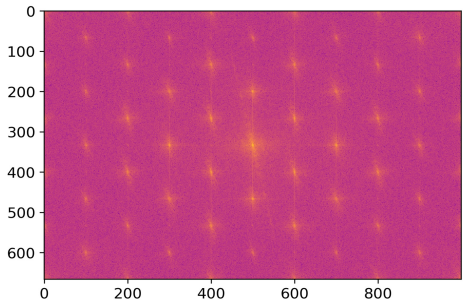

Figure 11. magnitude of *csi_ 1.png* in the frequency domain

As supposed the pattern is very clear in the frequency domain however to identify better the frequency to filter I chose to project the magnitude on the x and y axis:

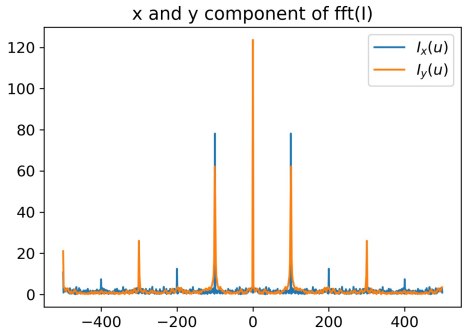

Figure 12. projection of the magnitude

The axis normalization is done using the function *np.fft.freq()* to be able to display correctly the two dimension that have difference sizes. Now is much more easy to identify the paras site frequency and those are multiply of $100u$.

To improve the image I chose to use a combination of filters: a *low pass filter* and a *notch filter*.

The first one is the same used in the previous exercise, the second one is constructed similarly:

$$H(u) = (1+i) - (1+i)A^{\left(-\frac{u^2}{2\sigma^2}\right)}$$

The resulting filters are shown in the following figure:

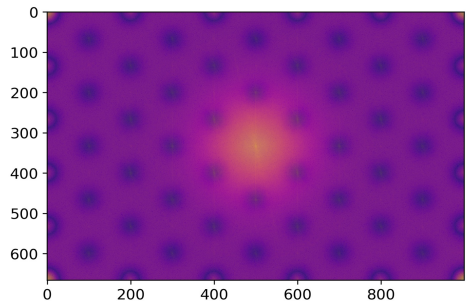

Figure 13. *low pass filter* and *notch filter* and magnetude of *I(u)*

Now that the filters are defined we have to combine it together, to do so in the frequency domain is very simple, given $I(y)$, the image transformed, $H'(u)$, the gaussian low pass filter and $H''(u)$, the notch filter, the reconstructed image in the frequency domain is computed as:

$$I_r(u) = I(u) \cdot H'(u) \cdot H''(u)$$

The reconstructed image is shown in figure 14.

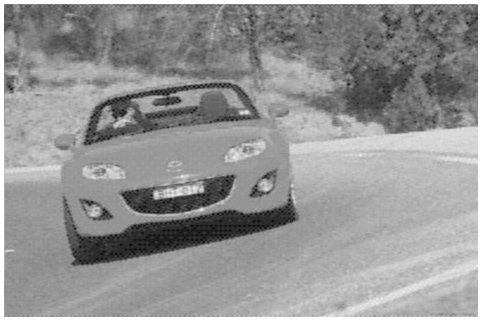

Figure 14. result of the reconstruction

The improvement is huge and now is even possible to read the car plate:

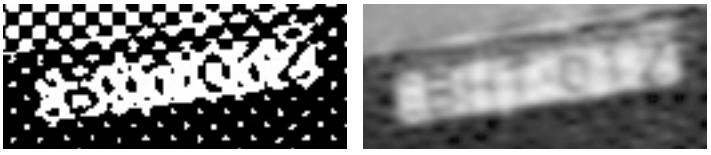

Figure 15. details differences between *csi_ 1.png* and the restored one

Code

The code can be found in the folder `code/` and has been written in *python 3* using *numpy, scipy* and *matplotlib* libraries, using *Jupyter notebook*, as this tools are both perform ant, powerful, free and open-source.