Discrete Fourier Transform
INF01213 Computational Photography – 2020/1

Lucas N. Alegre

June, 2020

1 Part 1 - Real and Imaginary Reconstruction

For this and the following tasks, I used the commands *fft2* and *ifft2* from python's *scipy* package, which are equivalent to the homonymous commands from MATLAB.

As both the Discrete Fourier Transform (DFT) and the Inverse Discrete Fourier Transform (IDFT) are linear operators, we can reconstruct an image by summing the IDFT of the real and imaginary components of its DFT. In Figure 1b and Figure 1c are shown the reconstruction of the cameraman image from the real and imaginary components of the DFT, respectively.

Both images contain a mirrored copy because of the complex conjugate symmetry. This means that for each cosine (real components) and sine (imaginary components) there is a symmetric sine/cosine with same magnitude but opposed phase (negative frequency). When both components are summed, the imaginary components are cancelled and the mirrored effect disappears. Moreover, the image in Figure 1c is darker because it contains many negative values, which were clipped to zero in order to generate the image. These negative values compensate for the extra brightness present in Figure 1b.

(a) Cameraman.

(b) Reconstruction from the real components of the DFT.

(c) Reconstruction from the imaginary components of the DFT.

Figure 1: Real and imaginary reconstruction.

2 Part 2 - Implementing the DFT and IDFT

In the second part of the assignment we should implement our own versions of the DFT and IDFT. If we exactly follow the equations that define them, we produce an algorithm which complexity is $O(N^2)$, where N is the number of pixels in the image. For this reason, this task was done with a reduced version of the cameraman image (Figure 2a). Alternatively, the Fast Fourier Transform implementations of MATLAB and scipy have complexity $O(N \log N)$.

In Figure 2 both my (DFT and IDFT) and scipy's ($fft2$ and $ifft2$) implementations are compared. As expected, despite the difference in computation time, both implementations produce the same results.

(a) Cameraman (100x100).

(b) $fft2 \longrightarrow IDFT$.

(c) $DFT \longrightarrow ifft2$.

(d) $DFT \longrightarrow IDFT$.

Figure 2: DFT and IDFT versus fft2 and ifft2.

3 Part 3 - The DC Term

In the third part of the assignment we should explore the direct current (DC) term of the DFT. The coefficients of the harmonic waves that constitute the DFT depend only on the image structure, but not on the image brightness. On the other hand, the DC term $F[0, 0]$ controls only the image brightness.

Consequently, if we reconstruct an image by applying the IDFT after zeroing the DC term of the DFT (Figure 3b), we obtain the same image as if we subtracted from the image the average intensity pixel level (Figure 3c).

(a) Cameraman.

(b) Reconstruction zeroing the DC term.

(c) Image after subtracting the average intensity level from all pixels.

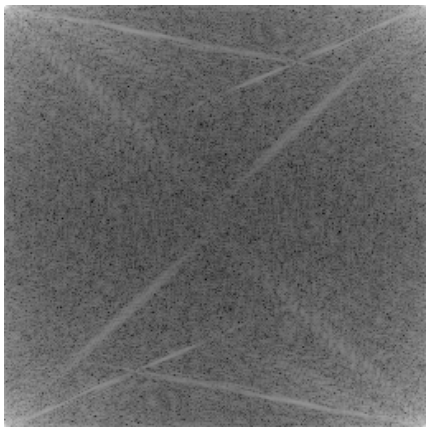
Figure 3: The DC term.

4 Part 4 - The fftshift command

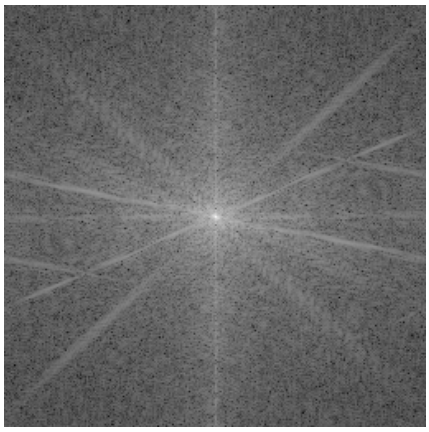
In this task, we explore the effects of the *fftshift* command. In Figure 4c and Figure 4d it is depicted the frequency spectrum of the cameraman before and after the *fftshift* command, respectively. Note that the DC term appears in the center of the image in the shifted spectrum. In order to show the frequency spectrum, I computed the log of the magnitude of the coefficients ($\log(abs(f) + 1)$) and then I scaled the resulting values to the $[0, 1]$ interval.

(a) Cameraman.

(b) Image reconstructed from the shifted spectrum.



(c) Frequency spectrum.



(d) Frequency spectrum after *fftshift*.

Figure 4: *fftshift* on frequency domain.

In Figure 5 we can observe that applying *fftshift* in the frequency domain is equivalent to multiplying the image by $-1^{(x+y)}$ in the space domain. The resulting images present a checkerboard pattern because the negative values were clipped to zero.

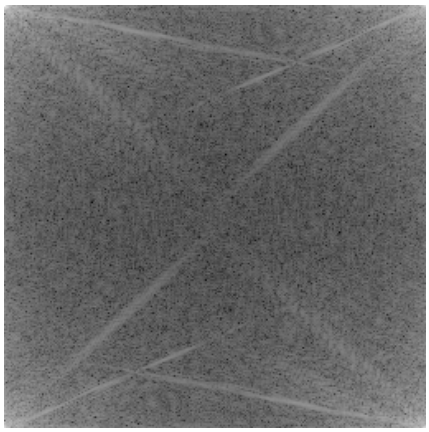
(a) $g(x, y) =$ *ifft2*(*fftshift*(*fft2*(f(x,y)))).

(b) $g(x, y) = -1^{(x+y)} f(x, y)$.

Figure 5: Frequency spectrum shift effect.

Alternatively, applying *fftshift* on space domain is equivalent to multiplying the frequency spectrum by $-1^{(u+v)}$. The spectrum on Figure 6b, however, looks the same as the frequency spectrum of the original image (Figure 4c). This is due to the fact that these images are only depicting the magnitude of the coefficients, which are the same despite some waves having its sign changed.

(a) Cameraman after *fftshift*.



(b) Frequency spectrum of the image shifted on space domain.

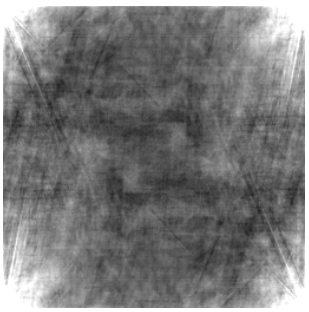
Figure 6: *fftshift* on space domain.

5 Part 5 - Amplitude versus Phase Spectra

Finally, in the last part of the assignment we study the role of the amplitude and the phase of the coefficients of the DFT. Each DFT coefficient can be seen as a vector in the complex plane, i.e., $F(u,v) = |F(u,v)|e^{i\theta(u,v)}$. Therefore, we can retrieve only the amplitude information by setting $\theta = 0$, which can be achieved using the *abs* command. In Figure 7b it is shown the resulting reconstructed image after applying the *abs* command to the DFT. On the other hand, if we retrieve only the phase information from the DFT ($F(u,v) = e^{i\theta(u,v)}$) before applying the IDFT and then scale the reconstructed image to the interval $[0,1]$, we obtain the image in Figure 7c. The edges and lines represent coordinates where many waves have the same phase.

Observing these images we can conclude that the amplitude holds the information about the image's contrast, whereas the phase holds the information about the image's structure. In other words, the amplitude modulates the harmonic waves, while the image structure results from waves and their phases.

(a) Cameraman.



(b) Reconstruction using only the amplitude information.

(c) Reconstruction using only phase information.

Figure 7: Amplitude versus Phase reconstruction.

6 Conclusions and considerations

This assignment helped me to better understand many concepts behind the Discrete Fourier Transform, as well as how they can be applied to 2D images.

It took me some time to realize how could I create the images of the frequency spectrum after applying the DFT. Additionally, I had some bugs related to type conversion between complex, float and unsigned integers in python, which were solved in the end.