CS371 Digital Image Processing
Exercise 4 part 1
Discrete Space Fourier Transform & Image Segmentation

Alexandros Angelakis
csd4334@csd.uoc.gr

December 14, 2022

1 Discrete Space Fourier Transform

In this section, we are going to look at two different 1D filter, h(m) and g(m), where m is an integer vector from -5 to 5. We will compute the Discrete Space Fourier Transform of the two 1D signals and their product, in the interval $[-\frac{1}{2}, \frac{1}{2}]$, and plot their magnitude.

First of all, let's look at the plots of the 1D signals in the time domain.

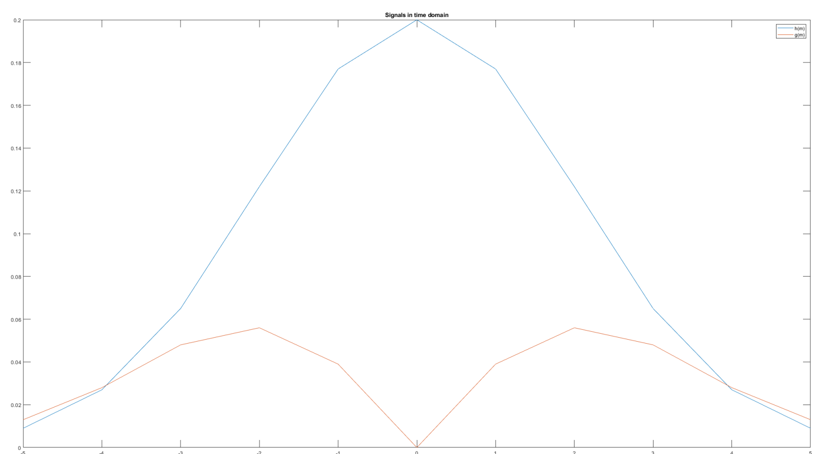

Figure 1: The 1D signals in the time domain

Let's now take a look at the 1D signals in the frequency domain.

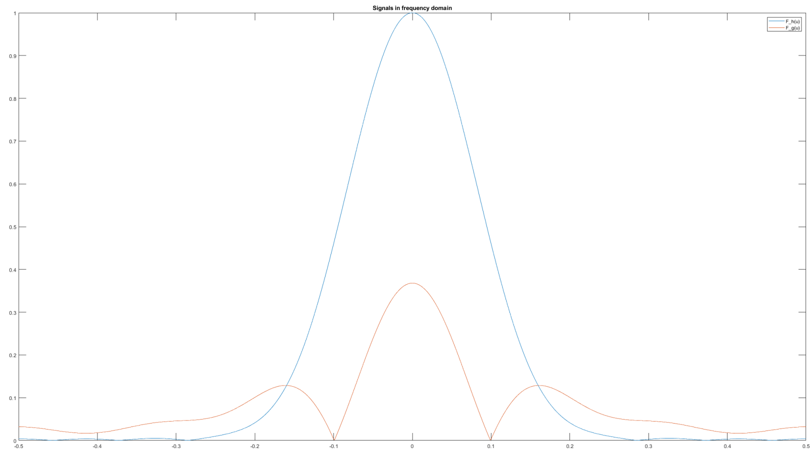

Figure 2: The 1D signals in the frequency domain

As we can see from the figure, the two signals are pretty similar in the frequency domain. They take all the low frequencies of the image and ignore the high frequencies. To be even more clear, the blue filter, $F_h(u)$, is a low pass filter, meaning that it only allows the low frequencies to pass through. It cuts all the frequencies that are greater than $\approx 0.28 = \frac{\pi}{11}$, so the cutoff frequency is $\frac{\pi}{11}$. The red filter, $F_g(u)$, is very similar to the blue one, but we can see some differences. It also allows a lot of high frequencies to pass through and its magnitude is a lot smaller than the blue ones.
To sum up, the blue filter acts like a low pass filter, with cutoff frequency equal to $\frac{\pi}{11}$, and the red filter acts like a low pass filter, but it allows some high frequencies to pass but those frequencies are multiplied by a really low number, so it's a little bit irrelevant. We can also see that it completely cuts off the frequencies -0.1 and 0.1.

Now let's see the 2D filter h(m)g(n), meaning the convolution of those two 1D signals in the time domain and in the frequency domain. I calculated the 2D signal in the time domain with two methods. For the first method, I just did the convolution between the two 1D signals, and for the second method, I computed the Fourier transform for both the 1D signals, multiplied them and then got the inverse Fourier transform.

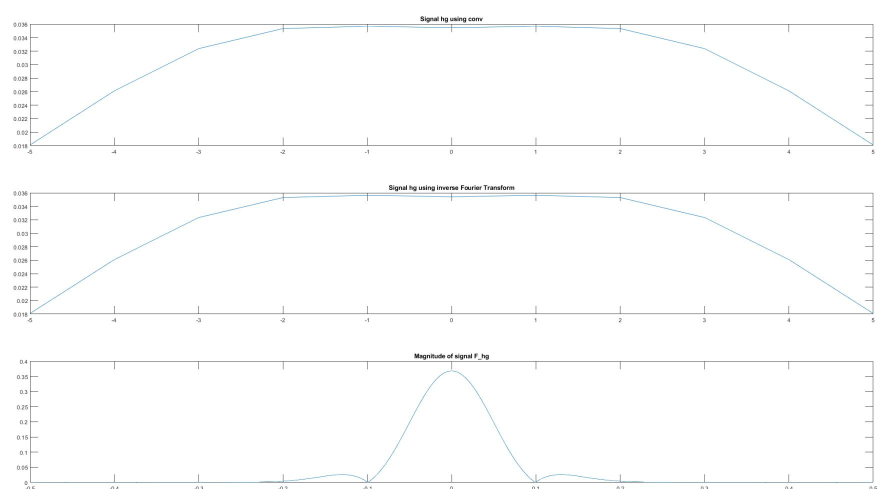

Figure 3: The 2D signal in the time and frequency domain

If we look at the 2D signal in the time domain, we can say that it approaches a square wave filter. That means that in the frequency domain, it cuts off all the frequencies that are outside of the cutoff frequency. Since it's not an ideal square wave, the Fourier transform of the 2D signal will not be a sinc function but an approach of it. From the plot of the Fourier transform, we can see that the cutoff frequency of this filter is 0.1, so all the frequencies beyond this frequency are not passing through.

2 Image Segmentation

In this section we are going to segment our image based on the unsigned direction of the vector θ (we'll talk about it later) using the K-means clustering algorithm. The image we are going to cluster is the following:

Figure 4: The original image

We'll start by utilizing the filter $h_1 = h(m)g(n)$ and the filter $h_2 = g(m)h(n)$, where h, g are the 1D filters from the previous section and m refers to rows, whereas n refers to columns. We then compute the filter responses $y1, y2$ of the image I with the filters h_1, h_2 respectively. Let's take a look at the images after filtering them.

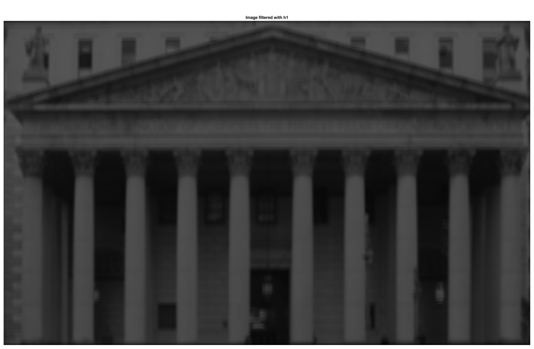

Figure 5: Filter response $y1$ of image I

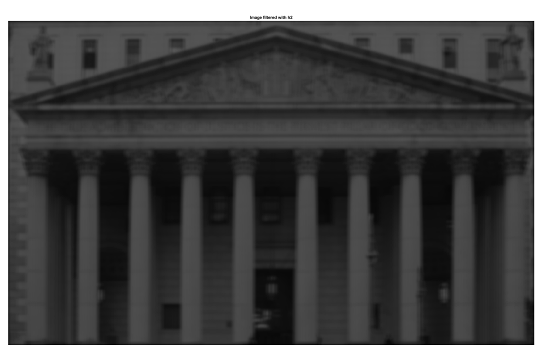

Figure 6: Filter response $y2$ of image I

As we can see, the two images are pretty similar, because the two 1D signals, as we saw from the previous section, are also pretty alike in the frequency domain.
Now let's go to the segmentation part of the image. As I mentioned before, we are going to segment our image using the K-means clustering algorithm with 5 clusters. The first cluster has all the values of $A = y_1^2(m,n) + y_2^2(m,n)$ that satisfy the

relation $A(m,n) \leq \mu$, and the rest clusters will have as initial cluster centers the values $0, 0.15\pi, 0.35\pi, 0.5\pi$. After we run the K-means algorithm for the image I, we get the following result:

Figure 7: The clustered image

First of all, we can see a lot of errors from our grouping, since there are similar part of the image that are not grouped together. For example the bricks on the bottom left of the image, we can see that it was labeled as black but the bricks above them are labeled differently. Despite that, we can see that the wall on the top side of the image and the pillars, since they have the same grey-scale intensity, they have been grouped together, labeled as 'red'. The same goes for all the intensity of the pixels that are close to this grey-scale intensity. We can also observe that all the horizontal and vertical edges of the image are labeled as 'green' or 'yellow'. The 'blue' label I think is useless, it does not give any good information for the image. Finally, we can see that the majority of the image is black. Thus, there are a lot of values of the square magnitude A that we mentioned before, that are less or equal than its mean value. If we increase the number of clusters in our K-means algorithm, the resulting image will slowly approach our original image. So by taking only 5 clusters, the result is mediocre, we can still understand the facade of the building but a lot of details are gone.