Laboratory Experiments 4 - Fourier Transform and Frequency Filtering

1 Objectives
- Understand the Fourier transform and the frequency domain.
- Become familiar with some important signals in 1D and 2D.
- Study the properties of Fourier Transform.
- Understand the basis of frequency filtering.

2 Background: 1D Fourier Transform

Fourier Transform is a mathematical operation that translates a signal from the spatial (or time) domain to the frequency domain. The basis of transform is the analysis of a signal in terms of exponential Fourier series, that is, represent a signal as a sum of exponential signals that are orthogonal to each other. The exponential function $e^{jx}$ can form a family of orthogonal functions within a certain interval $(t_0, t_0 + 2\pi/\omega_0)$ with the functions $e^{jn\omega_0 t}$, $n = 0, \pm 1, \pm 2, ...$ where $e$ is the complex Euler identity:

$$e^{jn\omega_0 t} = \cos n\omega_0 t + jsin n\omega_0 t$$

This function can be formed by a pair of sinusoidal signals, by which any signal can be expressed:

$$f(t) = F_0 + F_1 e^{j\omega_0 t} + F_2 e^{j2\omega_0 t} + F_3 e^{j3\omega_0 t} + ... + F_{-1} e^{-j\omega_0 t} + F_{-2} e^{-j2\omega_0 t} + F_{-3} e^{-j3\omega_0 t} + ...$$

$$f(t) = \sum_{n=-\infty}^{+\infty} F_n e^{jn\omega_0 t} \quad (t_0 < t < t_0 + T) \quad T = 2\pi/\omega_0$$

The coefficients $F_n$, can be obtained by computing how similar a certain signal is to the exponential:

$$F_n = \frac{\int_{t_0}^{t_0+T} f(t)(e^{jn\omega_0 t})^* dt}{\int_{t_0}^{t_0+T} e^{jn\omega_0 t}(e^{jn\omega_0 t})^* dt} = \frac{\int_{t_0}^{t_0+T} f(t)(e^{-jn\omega_0 t}) dt}{\int_{t_0}^{t_0+T} e^{jn\omega_0 t}(e^{-jn\omega_0 t}) dt}$$
It is important to notice that the previous analysis is limited to a certain interval in time (or space). For an interval \((-\infty, \infty)\), it is necessary thus to transform the whole signal and not just to express it in terms of component series. To do this, the exponential Fourier series of a signal \(f(t)\) can be calculated by transforming the signal into a periodic signal where \(f(t)\) is the first period. This implies that the signal is thought to be repeating until infinity. The limit for the period \(T \to \infty\) is evaluated and then the signal has only one cycle in the range of \((-\infty < t < +\infty)\). By taking the limit (and some mathematical manipulations) we can reach the Fourier Transform pair:

\[
F(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t)e^{-j\omega t} dt
\]

which sometimes is represented by:

\[
F(\omega) = Ff(t) \leftrightarrow f(t) = F^{-1}F(\omega)
\]

The previous expressions can be complex and therefore two planes can be used to show it: (real/imaginary) or (magnitude/phase).

In Matlab the commands `fft` and `ifft` perform the direct and inverse transformations (`fft2`, `ifft2`, `fftn`, `ifftn` for higher dimensions).

**Exercises**

1.1 Generate the signal \(G(t)\) and obtain its Fourier transform.

\[
G(t) = \begin{cases} 
0 & -\infty < t < -\frac{t_0}{2} \\
A & -\frac{t_0}{2} < t < \frac{t_0}{2} \\
0 & \frac{t_0}{2} < t < -\infty
\end{cases}
\]

Gate Function

**Hint:** For say \(t=20\). The maximum value for \(t\) may be 100 and minimum may be -100;

\[
G_{w0} = \text{fft}(G,t);
\]
Remember that $G(w)$ is a complex function, and therefore to plot it you can use the functions: abs, angle, real and imag. Try

```matlab
plot(abs(G_w));
plot(angle(G_w));
plot(real(G_w));
plot(imag(G_w));
```

In most cases, only the magnitude is used, still in the following examples try to observe the angle, and the real and imaginary parts.

1.2 Generate a sampling function and transform it with `fft`. What can you observe? Are the results what you expected? Perhaps the command `fftshift` could help. Hint: A sampling function is a function which has a periodic high and low value. A simple sampling function may be given by:

```
Func=[0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0]
```

Try to plot this function.

1.3 Generate the following signals in Matlab, transform them and study the results. Keep the length of the vectors constant.
1.4 Repeat step 1.3 for the following functions in Figure 1. Hint: These functions are essentially sinusoids of different frequencies. Please refer to previous labs inorder to understand how to generate these functions.

Figure 1. Signals to be generated.

3  2D Fourier Transform

Fourier Transform in two dimensions can be expressed as:

\[ F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)e^{-j2\pi(ux+vy)}dxdy \]
\[ f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v)e^{j2\pi(ux+vy)}dxdy \]

Exercises

2.1 Generate the 2D signal analogous to exercises 1.1 and 1.3, transform and observe:

Hint: Refer to previous labs for help in generating these functions.

2.2 Generate the 2D signal analogous to exercise 1.4 (as shown in figure 1), transform and observe:

2.3 What happens if you rotate (for 90° you can do this by transposing the matrix y = x;) the image before transforming?
2.4 Observe and compare the difference between the product of the two images and the sum of them ($y_1 \times y_2$ vs. $y_1 + y_2$)

2.5 What are the most important properties of the Fourier Transform? (Separability, translation, periodicity, rotation, ...). Check them in one of the references.

2.6 Extend the previous examples to understand the properties of the Fourier transform.

2.7 Generate the following signals: random noise (with \texttt{randn}), a delta (all zeros and one non-zero value), a checker board, and transform them, what do they look like?

4 Filtering

Frequency filtering is a very important subject in image processing. We know that the product of two signals in the frequency domain is equivalent to the convolution of their inverse transforms:

$$h(x, y) \ast f(x, y) \Leftrightarrow H(u, v)F(u, v)$$

Therefore if we multiply a suitable function with our image, we can filter the signal. First load one image and transform it:

```matlab
>> image1=double(imread('tire','tif'));
>> imw=fftshift(fft2(image1));
>> surfImage(log(1+abs(imw)))
```

Now, generate two Low Pass filters, one ideal filter (similar to the gate functions of section 2.1) and a Gaussian. The filtering of the image is done by a simple element-wise product:

```matlab
>> idealLP = imw.*filt1;
>> gaussLP = imw.*filt2;
```

Be careful to use pixel to pixel multiplication .* and not matrix multiplication *!.
The filters should look something like:

![Ideal and Gaussian Low pass filters](image)

**Exercises**

3.1 What differences can you see in the frequency domain filtered images?
3.2 Inverse transform the images and observe.
3.3 Generate two high pass filters (this can be done in just one line based on the previous filters) and repeat the filtering procedure.

![Ideal and Gaussian high pass filters](image)

3.4 Generate now a band pass and stop band filters and repeat the previous steps.

3.5 What happens if the filters are not centred in the origin? Load an image with many textures (Barbara image is excellent for this) and try to filter it with filters of different size and position. This is the basis of sub-band filtering, multi-channel filtering, and to some extent wavelets. Through this filtering, texture in images can be easily analysed.
5 Block Transformations

In order to generate block transformations like the Multi-Resolution Fourier Transform (MFT), you need to partition the image into several sub-images and transform each region separately. How can you do this? (Remember, the less loops the better, but you can still use them!).

Exercises

4.1 Can you identify some elements from the MFTs?