# Fast Fourier Transformation of an image

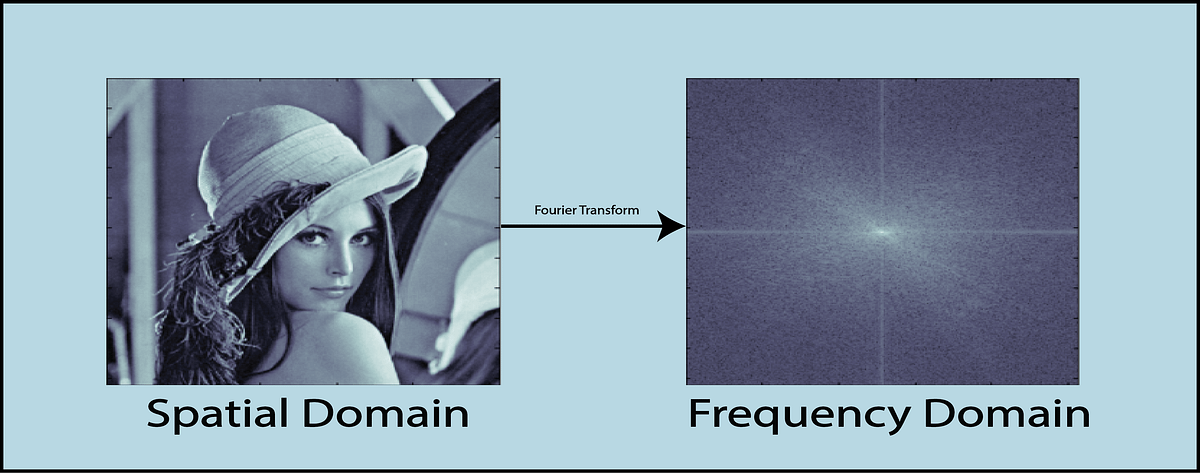
**Ivona Nencheva**

**Sami Fetyani**

**What Does the FFT Compute?**

The FFT converts an image in the spatial domain (x and y) to the frequency domain. In the spatial domain each point represents a pixel and its magnitude represents the color of the pixel. Whereas, in the frequency domain each point represents a frequency and its magnitude is the contribution of that frequency to the image. The strength of the magnitude determines the strength of the contribution of that frequency.

Another way of looking at the FFT is that it decomposes an image into sine and cosine components of varying frequencies.



Result of **fftw\_plan\_dft\_2d()**

When you apply the 2D FFT to an image with fftw\_plan\_dft\_2d() and fftw\_execute() the resulting output will be the frequency spectrum of the image. The DC component corresponding to 0Hz will be present in out[0] whilst the high frequency component will be present in out[N-1] where N = n x m and n is the number of pixels in the x-direction and m is the number of pixels in the y-direction.

This is the so-called **Forward FFT**.

The **Inverse or Backward FFT** is when the image is returned back to the spatial domain after it was manipulated.

**Frequency points in the domain and high-pass/low-pass filtering**

Exampe of frequency points in **DC-centered** array:

Image 44x40

Has lowest distance of 0.0 for [22,20]

And highest of ~29 for [0,0]

The center is [22,20] so those pixels are closest to the center and therefore have the **lowest frequency.**

• Low frequencies correspond to the smooth variations in an image, such as large uniform areas or soft gradients.

• High frequencies correspond to abrupt changes in pixel values, such as edges, textures, or noise.

• By applying a **high-pass filter**, we aim to enhance the high frequencies (edges, details) while attenuating the low frequencies. The cutoff value determines which frequencies are considered "high" and which are considered "low."

• A smaller cutoff value will preserve more high-frequency components, leading to stronger sharpening but also potentially amplifying noise. A larger cutoff value will attenuate more high-frequency components, leading to milder sharpening.

• For a **low-pass filter**, frequencies below the cutoff value are allowed to pass through (i.e., remain unaltered or attenuated less), while frequencies above the cutoff are attenuated (i.e., reduced in magnitude, potentially to zero).

• For a high-pass filter, frequencies above the cutoff value pass through, while frequencies below the cutoff are attenuated.

In our code we are not using DC centered arrays, our DC component is at the top-left –

**double dist = sqrt(x \* x + y \* y);**

* The resulting **dist** gives the distance of the point **(x, y)** from the top-left corner **(0, 0)**. This distance is used in the context of filtering to determine if a point in the frequency domain is within a certain radius (like **cutoff**)

**double cutoff = 200.0;**

This line defines the cutoff value. Any frequency component within this radius from the center of the spectrum will be considered "low frequency" and will be suppressed (set to zero).

if (dist < cutoff) {

out[y \* width + x][0] = 0.0;

out[y \* width + x][1] = 0.0;

}

If the distance *dist* is less than the cutoff, it means the frequency component is in the low-frequency region. As such, both the real ([0]) and imaginary ([1]) parts of that frequency component are set to zero, effectively suppressing it, therefore enhancing the high frequency parts and applying a high-pass filter.

**blur() and sharpen()**

In essence, these functions split the input image into its color channels, blur/sharpen each channel independently, and then merge the manipulated channels back together to produce the final filtered image. By going through each and every pixel, the QColor gets each value for each R, G and B values and sets them to each QImage– rChannel, gChannel and bChannel.

**sharpenChannel() and blurChannel()**

In short, both functions apply firstly the Forward FFT, then the respective low or high-pass filters based on a radius or cutoff, and lastly call the Inverse FFT where the normal visualization of the image is returned back(spatial domain).