# Mini Project 3 Report

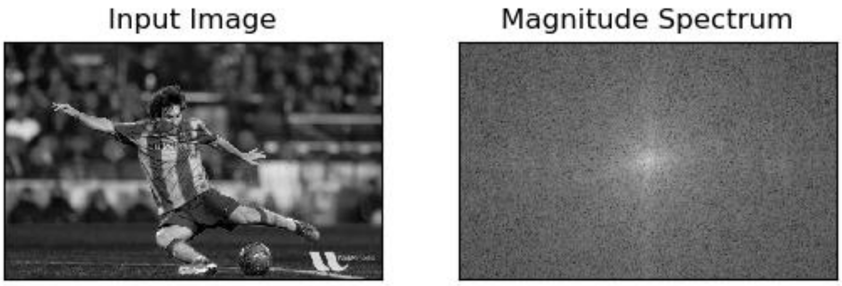
Junyou Chi

## Background

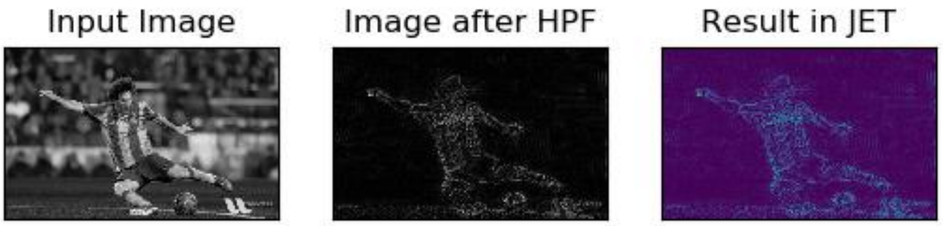
The Fourier transform is used to analyze the frequency characteristics of various filters. For images, 2D Discrete Fourier Transform (DFT) is used to find the frequency domain. Fast Fourier Transform (FFT) fast algorithm is used to calculate DFT. For a sinusoidal signal, x (t) = Asin (2πft), we can say that f is the frequency of the signal. If its frequency domain is accepted, we can see the peak of f. If the signal is sampled to form a discrete signal, we get the same frequency domain, but it is periodic in the range of [−π, π] or [0,2π] (or N-point DFT, [0, N] ). You can think of an image as a signal and sample it from two directions. So Fourier transform in X and Y direction will get the frequency representation of the image.

First, we will see how to find the Fourier transform using Numpy. Numpy has an FFT package to do this job. np.fft.fft2 () gives us the frequency conversion of a complex array. Its first parameter is the input image, which is a grayscale image. The second parameter is optional and determines the size of the output array. If it is larger than the size of the input image, the input image is padded with 0s before calculating the FFT. If it is smaller than the input image, the input image will be cropped. If no parameters are passed, the output array will be the same size as the input.

Once the result is obtained, the zero frequency component (DC component) will be in the upper left corner. If you want to bring it to the center, you need to change the result by passing N / 2 in both directions. This is done by the function np.fft.fftshift (). (It's easier to analyze). Once you find the frequency transform, you can find the size spectrum.

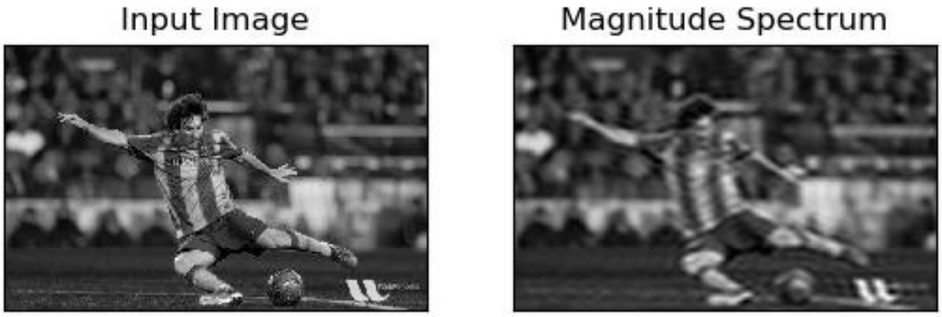


So you find the frequency transform. Now you can do some operations in the frequency domain, such as high-pass filtering and reconstructing the image, that is, finding the inverse DFT. To do this, you simply use a 60x60 rectangular window to remove the low frequency part. Then use np.fft.ifftshift () to apply a reverse shift to make the DC component appear in the upper left corner again. Then use the np.ifft2 () function to find the inverse FFT. Again, the result will be a plural. You can take its absolute value.



You can also use cv.cartToPolar (), which returns both size and phase in one shot.

What we need to do now is inverse DFT. Earlier we created an HPF, this time we will see how to remove high frequency content from the image, that is, we apply LPF to the image. It actually blurs the image. To do this, we first create a mask with a high value (1) and low frequency, that is, we pass low-frequency content and 0 in the high-frequency region.



Reference:

https://blog.csdn.net/JS_XH/article/details/79280519