Discrete Fourier Transform for Dummies

Alex Frasson and Tiago Augusto Engel[1]

[1]Universidade Federal de Santa Maria (UFSM)

{afrasson,tengel}@inf.ufsm.br

1. Introduction

2. The Real DFT

nesse link explica

http://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch31.pdf

While historically most theorists engineers find easier to work with the complex form, the Fourier transform can be calculated using real numbers as well.

$$ReX(k) = \frac{2}{N}\sum_{n=0}^{N-1} f(n)\cos(2\pi kn/N) \tag{1}$$

$$ImX(k) = \frac{-2}{N}\sum_{n=0}^{N-1} f(n)\sin(2\pi kn/N) \tag{2}$$

The N sample time domain signal $f(n)$ is decomposed into a set of N/2+1 cosine, and N/2+1 sine waves, with frequencies given by the index k. The amplitudes of the cosine waves are contained in $ReX(k)$, while the amplitudes of the sine waves are contained in $ImX(n)$. These equations operate by correlating the respective cosine or sine wave with the time domain signal. In spite of using the names: real part and imaginary part, there are no complex numbers in these equations[ANALOG 2016].

3. 1D DFT Definition

$$F(u) = \sum_{x=0}^{M-1} f(x)e^{-j2\pi ux/N} \tag{3}$$

where $u = 0,1,2,\ldots,M-1$

$$f(x) = \frac{1}{M}\sum_{u=0}^{M-1} F(u)e^{j2\pi ux/N} \tag{4}$$

where $x = 0,1,2,\ldots,M-1$

4. 2D DFT Definition

The 2D transform can be obtained by calculating the 1D of the rows of the image, following by the same function applied to the columns.

```
def DFT_slow(x):
    x = np.asarray(x, dtype=complex)
    N = len(x)
    n = np.arange(N)
    k = n.reshape((N, 1))
    M = np.exp(-2j * np.pi * k * n / N)
    return np.dot(M, x)
```

Figure 1. 1D DFT python implementation.

$$F(k,l) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-j2\pi(\frac{km}{M}+\frac{ln}{N})} \tag{5}$$

where $m = 0,1,2,\ldots,M-1$ and $n = 0,1,2,\ldots,N-1$

$$f(m,n) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} F(k,l) e^{j2\pi(\frac{km}{M}+\frac{ln}{N})} \tag{6}$$

where $k = 0,1,2,\ldots,M-1$ and $l = 0,1,2,\ldots,N-1$

```
def DFT2D_slow(x):
    x = np.asarray(x, dtype=complex)

    M = len(x)
    N = len(x[0])
    mat = np.zeros((M, N), np.complex)

    for row in range(M):
        mat[row] = DFT_slow(x[row])

    for col in range(N):
        mat[:,col] = DFT_slow(mat[:,col])

    return mat
```

Figure 2. 2D DFT python implementation.

4.1. The spectrum

The norm of the amplitude, or Fourier spectrum is defined as the image corresponding to the length of each complex number. Indeed, we can calculate the length of the complex number as:

$$|F(u,v)| = (R^2(u,v) + I^2(u,v))^{1/2} \tag{7}$$

The spectrum can be visualized by calculating the aforementioned formula. However, in order to properly visualize it, we need to translate the top-left corner to the center of the image. This can be done by applying the transform from -M/2 to M/2 or simply multiplying the input image by $(-1)^{x+y}$ [Gonzalez and Woods 2008].

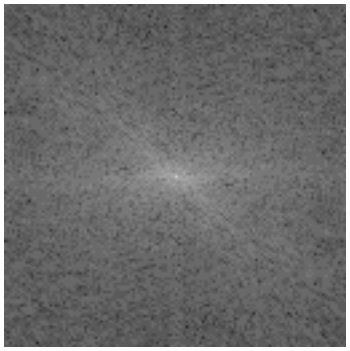

Figure 3. Fourier spectrum example.

Finally, as the frequency components u and v are zero at the origin, it is often referred to as DC component. This component corresponds to the highest magnitude in the image, and is the average of all other samples.

4.2. Phase angle

The phase angle is defined as:

$$\phi(u,v) = \arctan(\frac{I(u,v)}{R(u,v)}) \tag{8}$$

Even though phase information is not used as much as the spectrum information, there are specific applications where it is useful. If we construct synthetic images made from the amplitude information of one image and the phase information of another, it is the image corresponding to the phase data that we perceive, if somewhat degraded. We can see this effect in the following figure.

References

ANALOG (2016). The Complex Fourier Transform.

Gonzalez, R. C. and Woods, R. E. (2008). Digital image processing. *Nueva Jersey*.

Figure 4. Fourier spectrum example.