Fourier Transform

2017135002/최성윤

기본원리

Fast Fourier Transform(Danielson-Lanczos Algorith

$$H_n \equiv \sum_{k=0}^{N-1} h_k \, e^{-2\pi i k n/N}$$

해당 식을 DFT 하면 N^2 의 연산을 한다.
Time Complexity O(n^2)

Danielson-Lanczos Algorithm

$$F_k = \sum_{j=0}^{N-1} e^{-2\pi i j k/N} f_j$$

$$= \sum_{j=0}^{N/2-1} e^{-2\pi i k(2j)/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{-2\pi i k(2j+1)/N} f_{2j+1}$$

$$= \sum_{j=0}^{N/2-1} e^{-2\pi i k j/(N/2)} f_{2j} + W^k \sum_{j=0}^{N/2-1} e^{-2\pi i k j/(N/2)} f_{2j+1}$$

$$W \equiv e^{-2\pi i/N}$$

$$= F_k^e + W^k \, F_k^o$$

We can repeat the above division until the number of Fourier elements becomes one.

That is, $F_k^{eoeeoeo\cdots oee} = f_n$ for some n

Divide and Conquer Algortihm (점점 반복해나가며 쪼개간다
을 사용하여 시간 단축
Time Complexity O(nlogn)

기본원리

FFT in Two-Dimenson

$$H_n \equiv \sum_{k=0}^{N-1} h_k \, e^{-2\pi i k n/N}$$

1차원

$$H(n_1, n_2) \equiv \sum_{k_2=0}^{N_2-1} \sum_{k_1=0}^{N_1-1} \exp(-2\pi i k_2 n_2/N_2)\ \exp(-2\pi i k_1 n_1/N_1)\ h(k_1, k_2)$$

$$\begin{aligned} H(n_1, n_2) &= \text{FFT-on-index-1}\left(\text{FFT-on-index-2}\left[h(k_1, k_2)\right]\right) \\ &= \text{FFT-on-index-2}\left(\text{FFT-on-index-1}\left[h(k_1, k_2)\right]\right) \end{aligned}$$

2차원

L 차원에 대하여 Fourier Transform

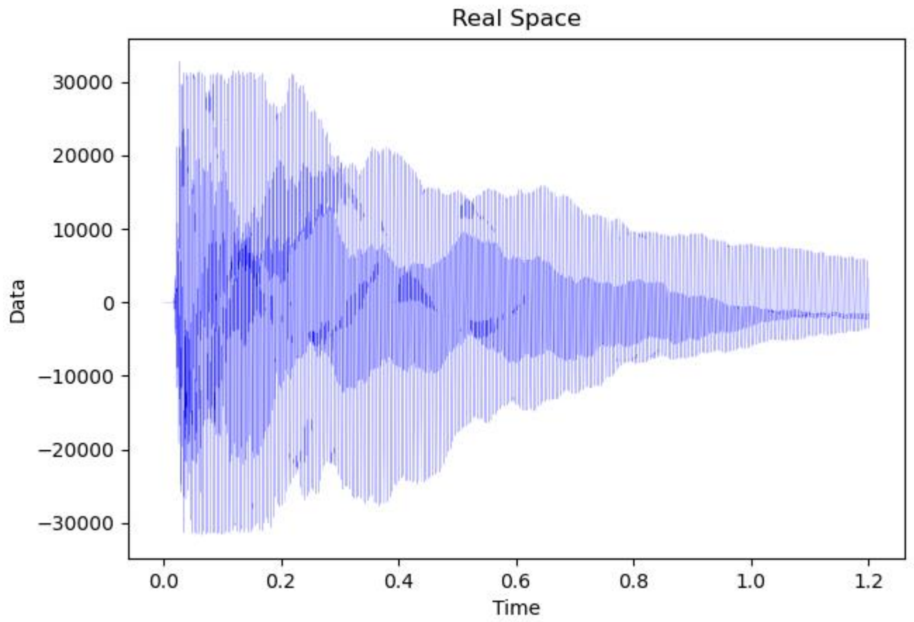
$$\begin{aligned} H(n_1, \ldots, n_L) \equiv \sum_{k_L=0}^{N_L-1} \cdots \sum_{k_1=0}^{N_1-1} & \exp(-2\pi i k_L n_L/N_L) \times \cdots \\ & \times \exp(-2\pi i k_1 n_1/N_1)\ h(k_1, \ldots, k_L) \end{aligned}$$

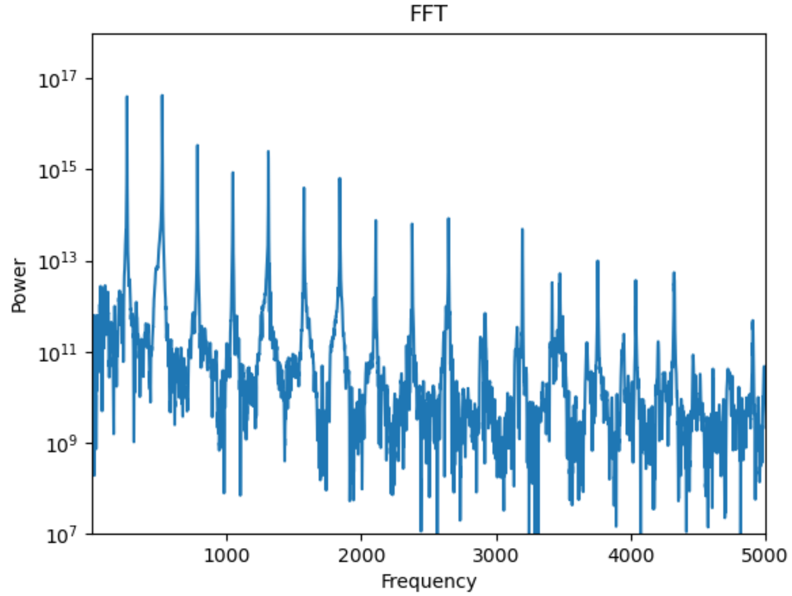
실행

FFT wav file

```
#피아노 도 추출
sound=wavio.read('Do262.wav')
duration=sound.data.shape[0]*1.0/sound.rate
x=sound.data.flatten()*1.0
y=np.fft.fft(x)
p=np.abs(y)**2
freq=np.fft.fftfreq(sound.data.shape[0])*sound.data.shape[0]/duration
time=np.linspace(0,duration,sound.data.shape[0])
```

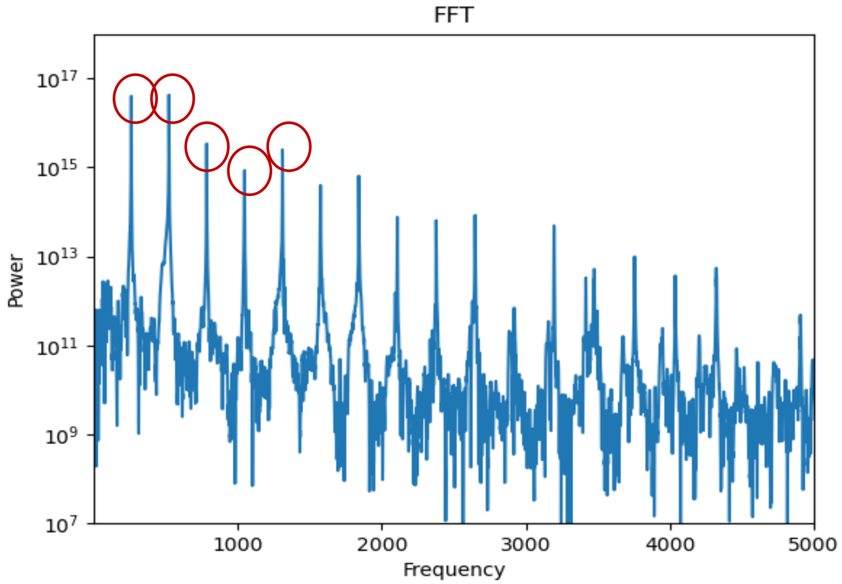
Wavio 를 통하여 wav file의
Sound data를 읽은후
FFT 해주었다





토의사항

1.FFT 에서의 특징

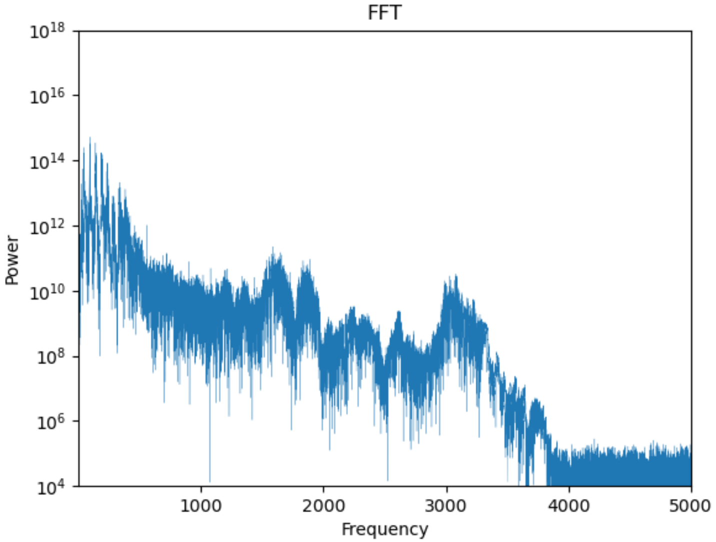


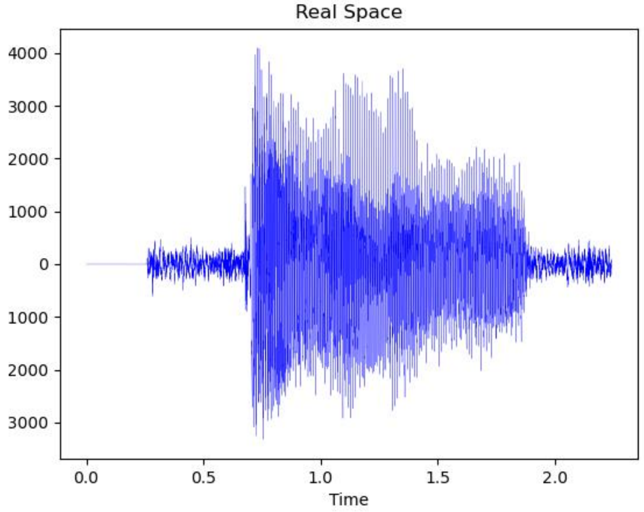
옆 그래프와 같이 특정 주파수 지점에서 Power 매우 커진다.
이때의 주파수를 확인해보면

258, 516, 783, 1047, 1310, 1575, 1842로 커짐을 알 수 있다.
이는 C4의 주파수인 261.63의 정수배로 근사 될수 있다

2.자신의 목소리와 비교

My C



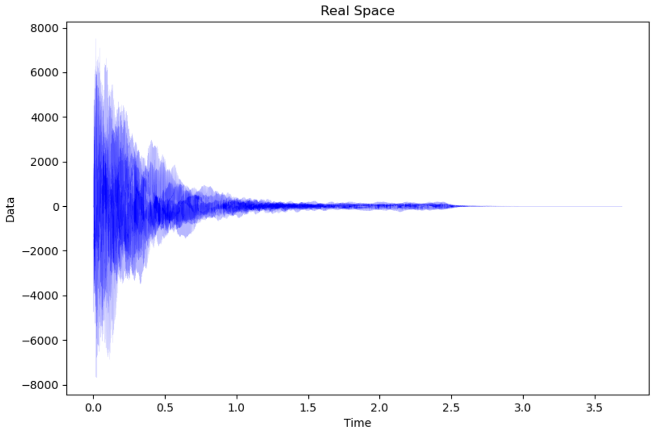


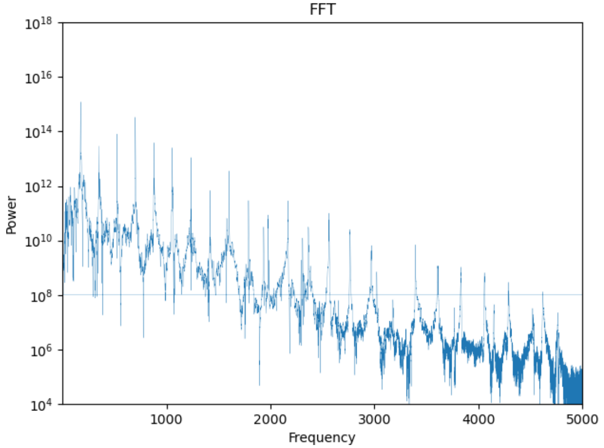
Piano 인 경우와 유사하게 주파수가 증가할수록 power가 감소함을 보인다. 하지만 Piano 인 경우와 달리 뚜렷한 peak를 가지는 주파수가 보이지 않는다. 이는 사람의 C는 일정하지 않고 주위의 환경이나 녹음의 품질에 따라 영향을 많이 받기 때문이라고 추측할수 있다.

토의사항

3.Piano의 다른 음정

Piano F



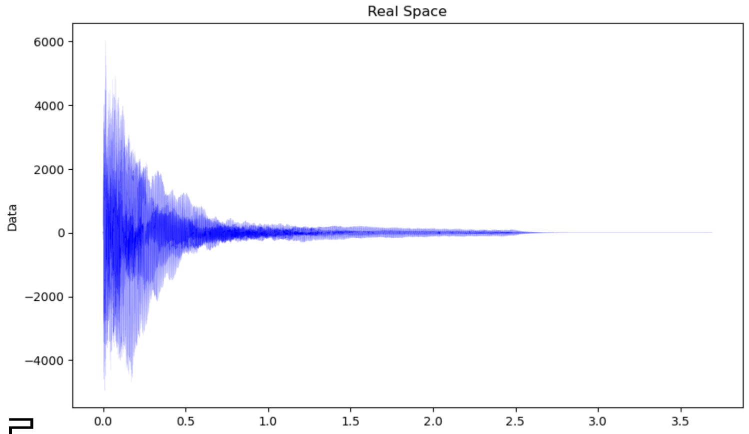


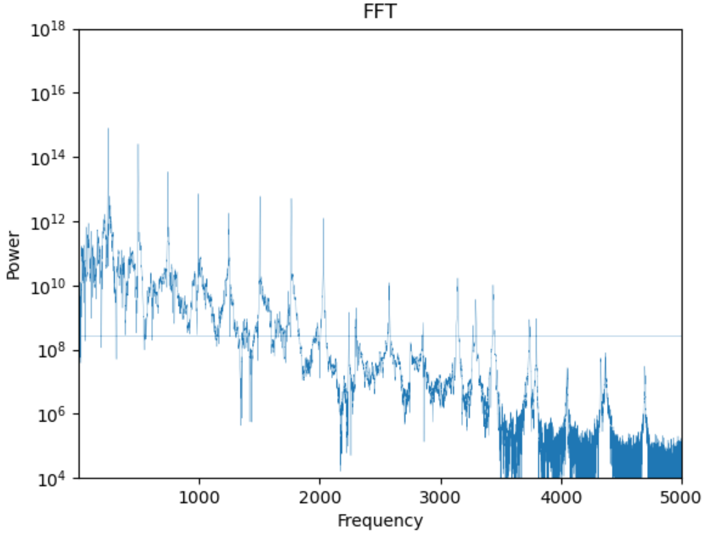
C 인경우와 비슷하게 FFT 경우 특정 주파수 지점에서 peak을 가짐을 확인할수 있다.

F의 경우 대략 174, 349, 524, 701,.....
B의 경우 대략 247, 495, 743, 995,.... 로 증가한다

이는 F3의 주파수는 174.61 B3의 주파수는 246.94로 이것의 정수배와 근사함을 알 수 있다.

Piano B



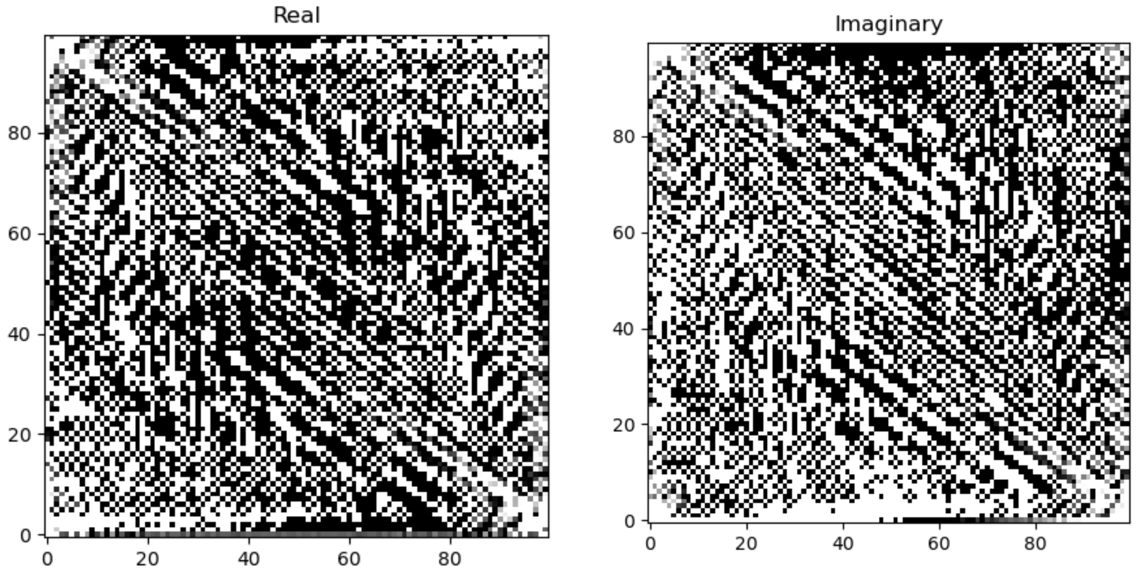


실행

```
image=fits.open('image1.fits')
data=image[0].data
H=np.fft.fft2(data)
Hreal=H.real
Himag=H.imag
power=abs(H)**2
h=np.fft.ifft2(H)
f=np.fft.fftfreq(100)
fx,fy=np.meshgrid(f,f)
F=(fx**2+fy**2)**0.5
```

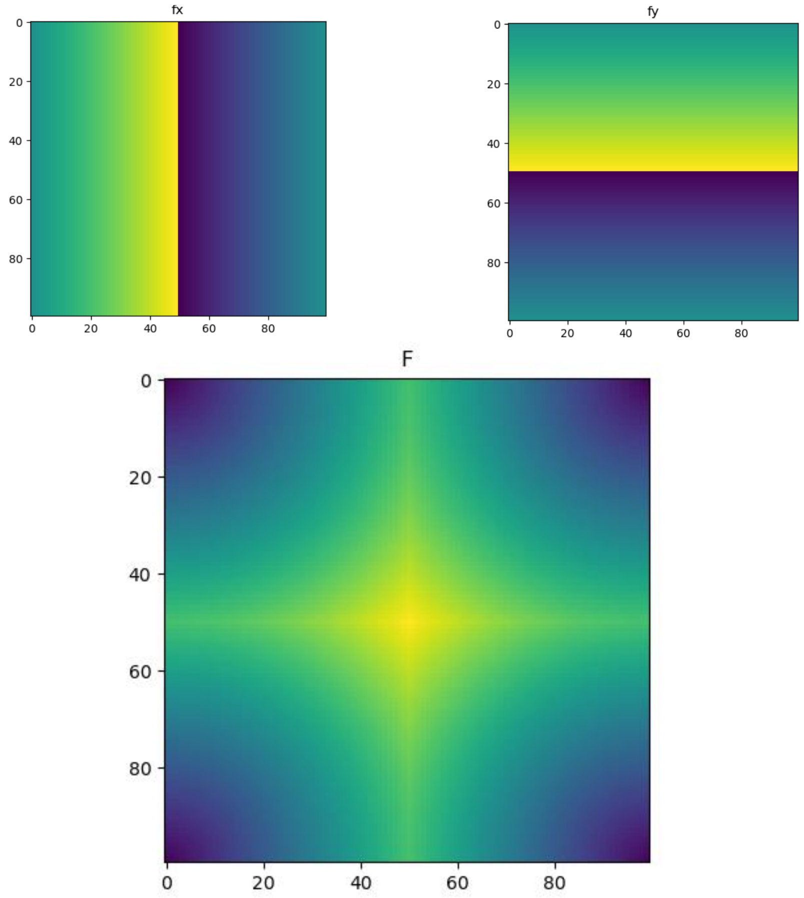
Image의 data 값을 추출한후 실수 부분과 허수 부분으로 분리후 FFT를 통하여 Power 값을 구해주었다.

FFT 한 값의 실수 부분과 허수 부분

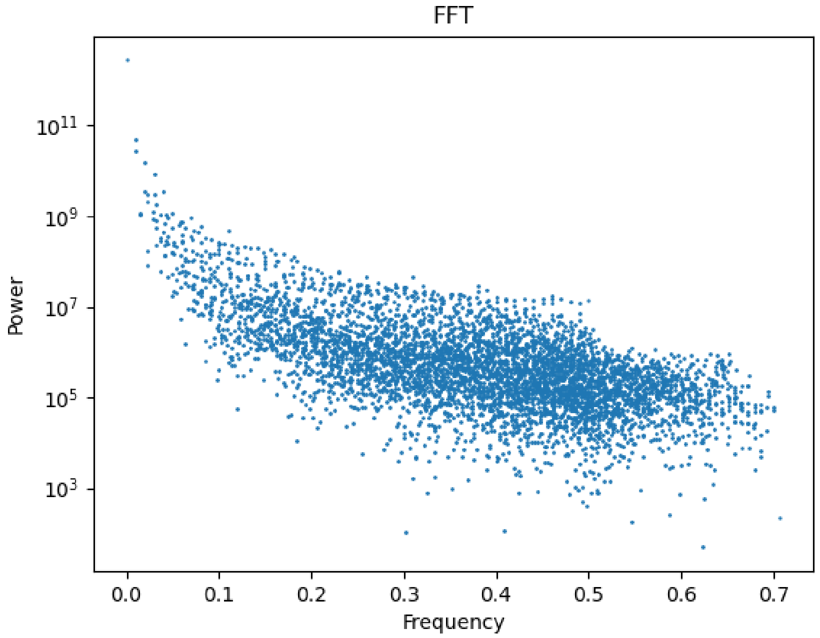


결과

2D Frequency map 생성

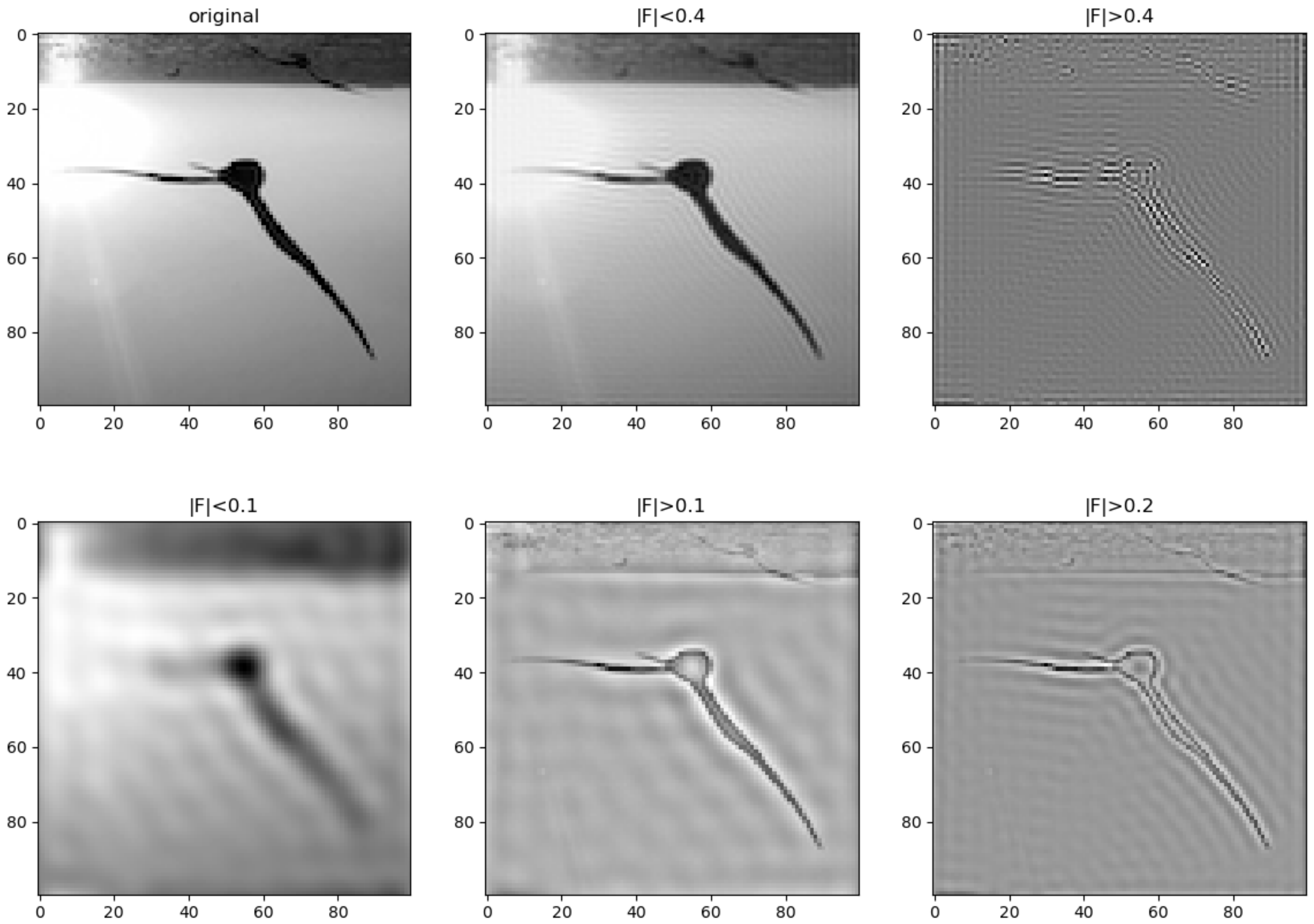


Power Spectrum



주파수가 작아질수록 Power가 커지는 경향이 있지만 어느정도 power 값이 균등하게 퍼져있다

주파수 범위에 따른 IFFT 결과



부등호 방향이 > 이면 image가 그림의 경계, 윤곽선을 구성하고 있는 것으로 보이며

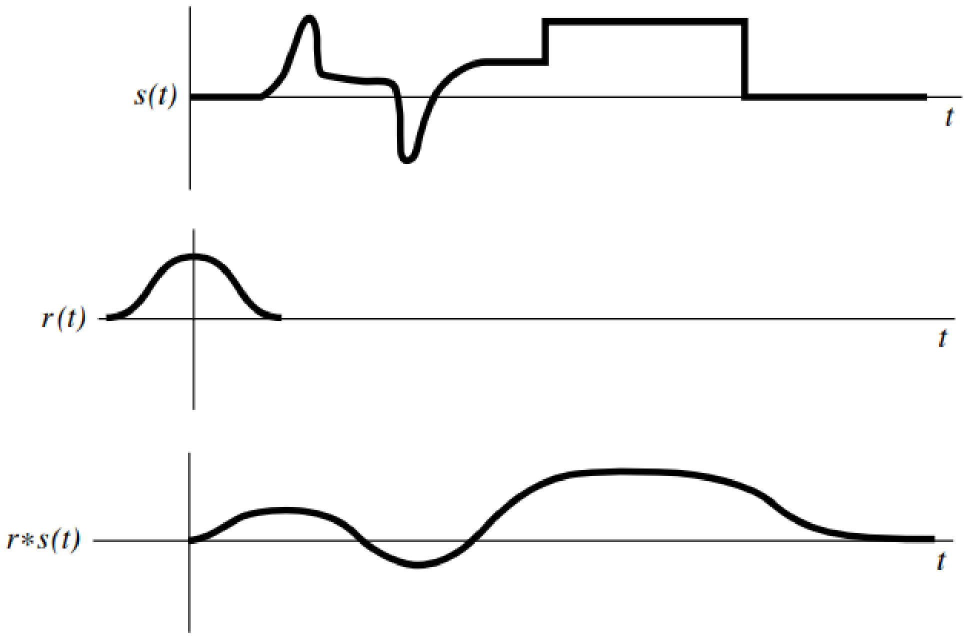
< 이면 image의 색상을 결정하고 있는것으로 보인다.

이는 주파수가 클수록 data 즉 색상의 변화가 큼을 의미한다.
따라서 주파수가 클수록 색상이 변화하는 지점 즉 경계선을 나타내고,
주파수가 낮을수록 동일한 생각을 나타내기 때문에 image의 한부분의 색상을 결정한다.

기본원리

Convolution

$$s * r \equiv \int_{-\infty}^{\infty} s(\tau) r(t-\tau) d\tau$$



Kernel을 통하여 원래 함수를 Smoothing 한다

Smoothing 정도는 kernel의 폭으로 결정

$$\sum_{k=-N/2+1}^{N/2} s_{j-k} \, r_k \quad \Longleftrightarrow \quad S_n R_n$$

FFT후 Convolution 하는것과
바로 Convolution 하는 2가지 방법 존재

실행

```python
def cg(sigma,type):
    x = np.linspace(0, 999, 1000)
    G=(np.exp(-((x-500)**2)/(2*sigma**2))/(np.sqrt(2*np.pi)*sigma))
    CG=np.array(list(G[500:])+list(G[:500]))
    if type==1:
        return CG
    elif type==0:
        return G
```

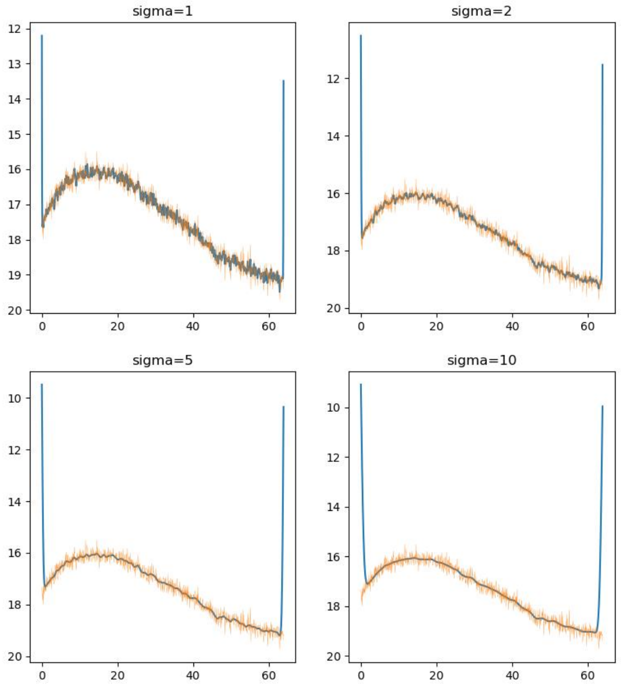
Gaussian Kernel을 sigma에 따라, FFT를 이용할지
Real Space에서 바로 할지에 따라 return 값 조정

```python
zerom=np.array(list(np.zeros(500))+list(m)+list(np.zeros(500)))
endm=np.array(list(np.tile(m[0],500))+list(m)+list(np.tile(m[-1],500)))
periodicm=np.array(mirror2+list(m)+mirror1)
mirror1.reverse()
mirror2.reverse()
mirrorm=np.array(mirror1+list(m)+mirror2)
```

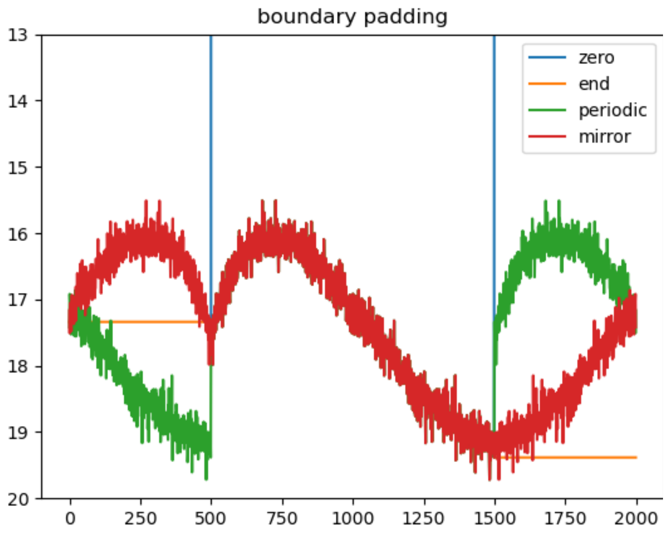
Padding 방법에 따른 array 생성

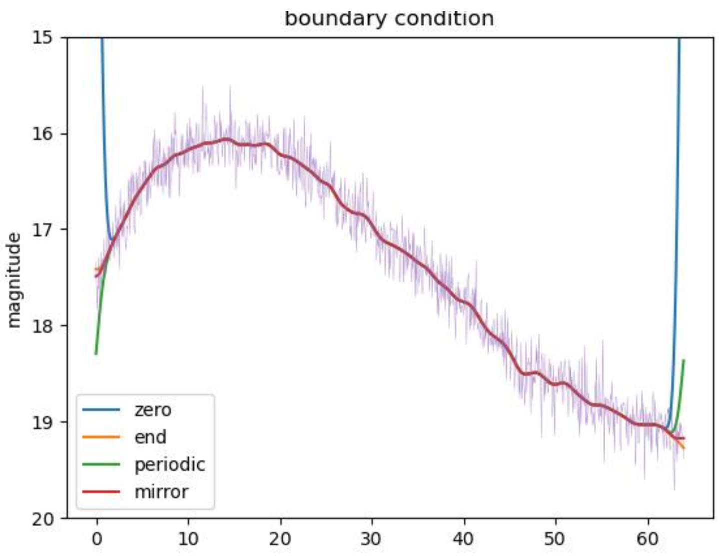
결과

Zero padding with direct convolution



Zero padding을 사용하여 convolution 하였을 경우에는 boundary에서 심한 오차가 발생함을 확인할수 있다.



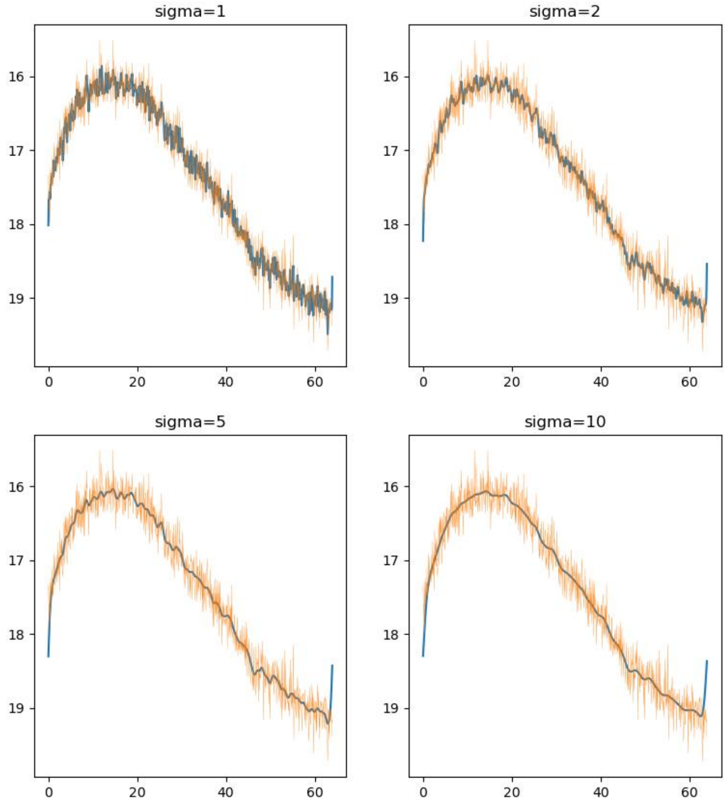


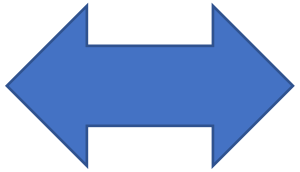
위는 boundary를 zero 이외에도 다양한 조건을 사용하여 convolution을 하였다.
왼쪽은 조건에 따른 index 위치에서 magnitude 값
오른쪽은 convolution 하였을때 결과를 나타내었다.

Zero-periodic-mirror-end 순으로 양 끝에서 변화가 적음을 확인할수 있다.

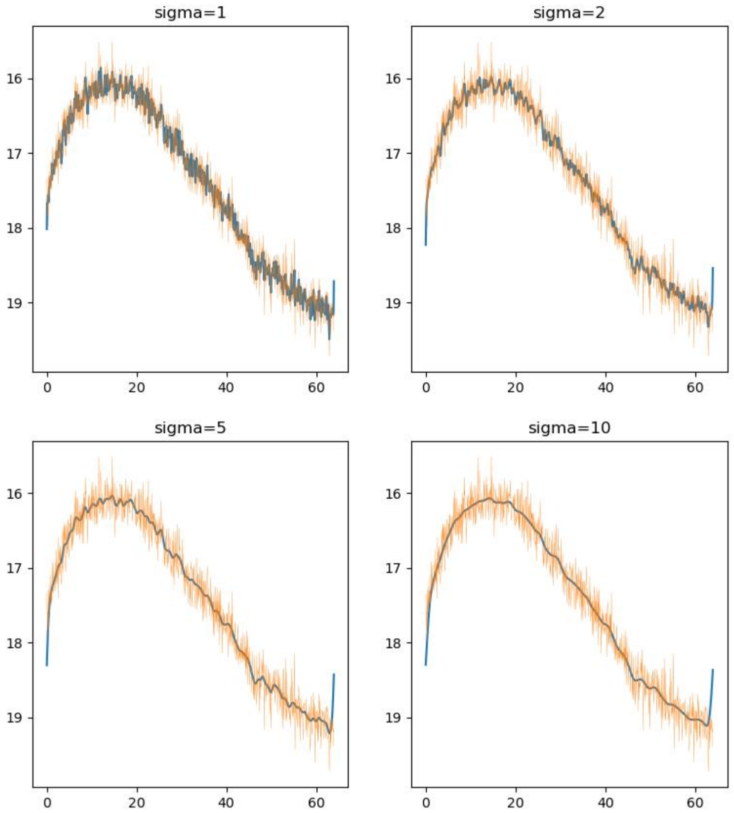
결과

FFT Convolution





Periodic padding convolution



FFT를 사용하여 convolution을 하였을 경우에는 periodic padding 사용했을때와 유사하다

Reference

Fast_Fourier_transform,en-wikipedia

https://en.wikipedia.org/wiki/Fast_Fourier_transform

Free piano samples, samplefocus

https://samplefocus.com/categories/piano

Frequencies for equal-tempered scale, A_4 = 440 Hz

https://pages.mtu.edu/~suits/notefreqs.html