CHAPTER 09
변환영역 처리

PART 02 영상 처리와 OpenCV 함수 활용

contents

9.1 공간 주파수의 이해

- 신호의 표현과 주파수 변환
  - 주파수 변환의 물리적 의미
  - 다양한 주파수 변환 방법
  - 디지털 영상의 주파수 변환
- 응용 분야
  - 영상의 주파수 영역 해석
  - 주파수 영역에서의 필터링
  - 화질 개선 등
- 다양한 변환 방법
  - 푸리에 변환(Fourier transform)
  - 이산 여현 변환(DCT : Discrete Cosine Transform)
  - 이산 웨이블릿 변환(DWT : Discrete Wavelet Transform)

9.1 공간 주파수의 이해

- 영상의 표현 및 변환
  - 다양한 공간에서 영상 표현 가능
  - 직교 좌표계
    - 각 화소에 대한 밝기 값의 표현
  - 주파수 영역으로의 변환
    - 공간 (직교 좌표계)상의 영상을 주파수 영역으로 변환

- 신호의 표현
  - 임의의 N차 신호 벡터 $\vec{x}$는 기저 벡터 $\overrightarrow{b_n}$의 선형 결합으로 표현 가능

$$\vec{x} = \sum_{n=1}^{N} c_n \overrightarrow{b_n}$$

  - c_n은 $\vec{x}$ 내에 $\overrightarrow{b_n}$이 내포된 양을 의미
  - N개의 기저벡터 $\overrightarrow{b_n}$는 서로 직교

9.1 공간 주파수의 이해

- 내적을 통한 신호의 포함 관계
  - 임의의 *N*차 신호 벡터 $\vec{x}$ 내에 기저 벡터 $\vec{b}$가 포함된 양은 내적을 통해 계산 가능

$$\vec{x} = \{x_1, x_2, x_3, \dots, x_n\}, \vec{b} = \{b_1, b_2, b_3, \dots, b_n\}$$

$$\vec{x} \cdot \vec{b} = x_1 b_1 + x_2 b_2 + x_3 b_3 + \cdots + x_n b_n = \sum_{i=1}^{n} x_i b_i = |\vec{x}| \left|\vec{b}\right| cos\theta$$

  - 2차원 벡터 $\vec{x}$와 기저 벡터 $\overrightarrow{b_1}, \overrightarrow{b_2}$ 간의 포함 관계

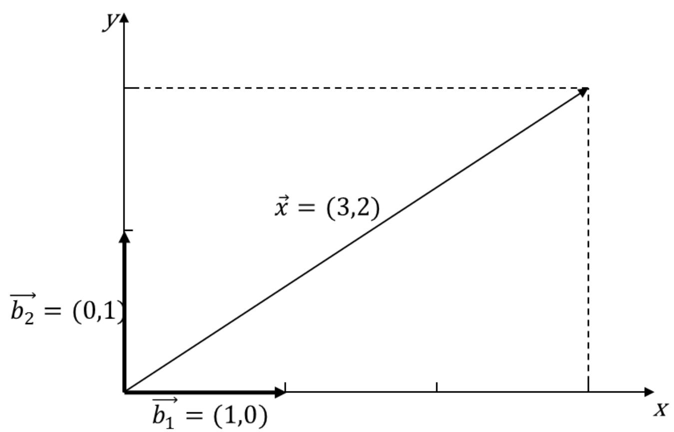


그림 4-1. 2차원 공간에 정의된 임의의 벡터 표현

$$\vec{x} \cdot \overrightarrow{b_1} = 3 \times 1 + 2 \times 0 = 3$$

$$\vec{x} \cdot \overrightarrow{b_2} = 3 \times 0 + 2 \times 1 = 2$$

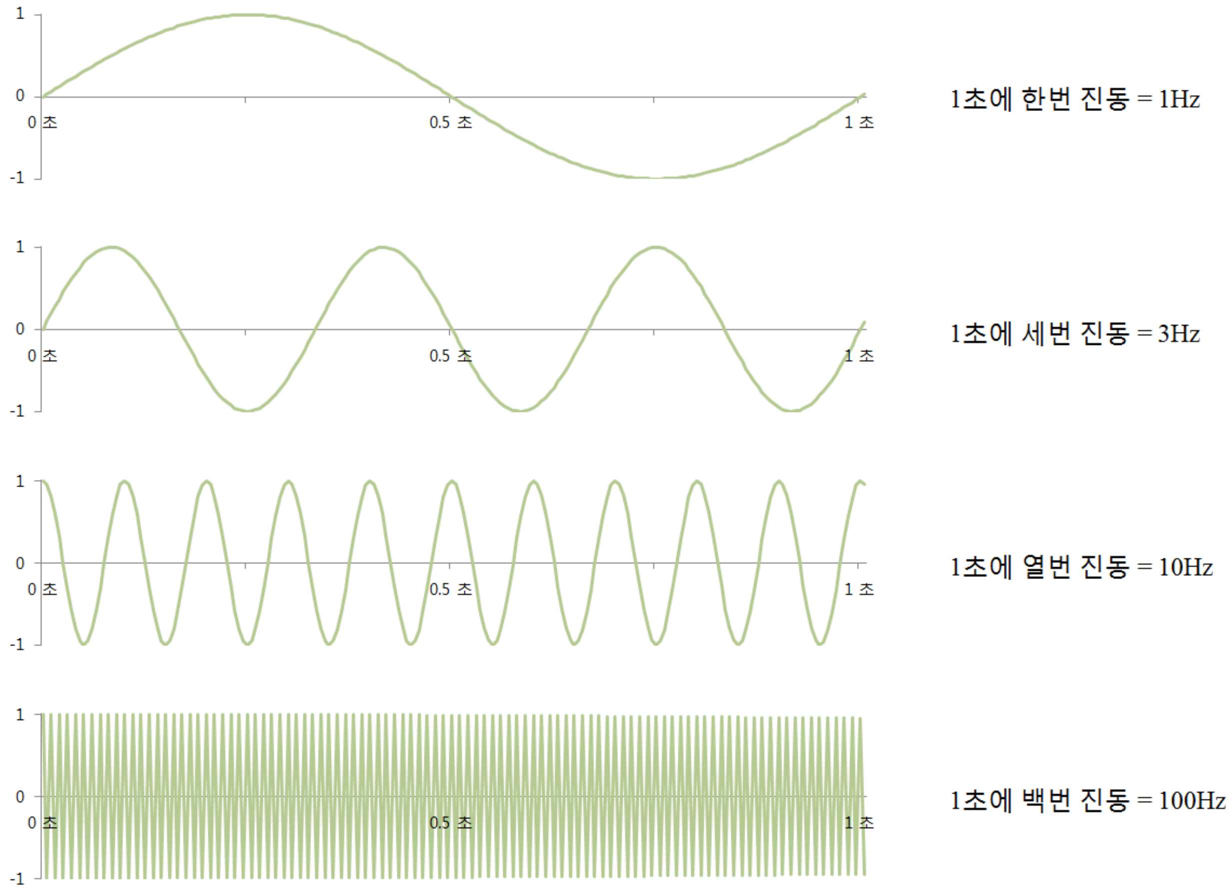
9.1 공간 주파수의 이해

- 기저 벡터의 특징
  - 직교성 (orthogonality)
    - 서로 다른 두 기저 벡터는 직교 → 내적의 결과가 0
  - 정규성 (normality)
    - 각 기저 벡터의 크기는 1
    - $\overrightarrow{b_1} \cdot \overrightarrow{b_2} = (1 \times 0) + (0 \times 1) = 0$
    - $\left\| \overrightarrow{b_1} \right\| = \sqrt{1^2 + 0^2} = 1$
    - $\left\| \overrightarrow{b_2} \right\| = \sqrt{0^2 + 1^2} = 1$

- 2차원 벡터 $\vec{x}$의 복원
  - 기저 벡터 $\overrightarrow{b_1}$, $\overrightarrow{b_2}$ 와의 포함 관계로부터 복원 가능
  - $\vec{x}$ 내에는 $\overrightarrow{b_1}, \overrightarrow{b_2}$가 각각 3, 2만큼 포함되어 있음
    - $\vec{x} = 3 \times \overrightarrow{b_1} + 2 \times \overrightarrow{b_2} = 3 \times \{1, 0\} + 2 \times \{0, 1\} = \{3, 2\}$

9.1 공간 주파수의 이해

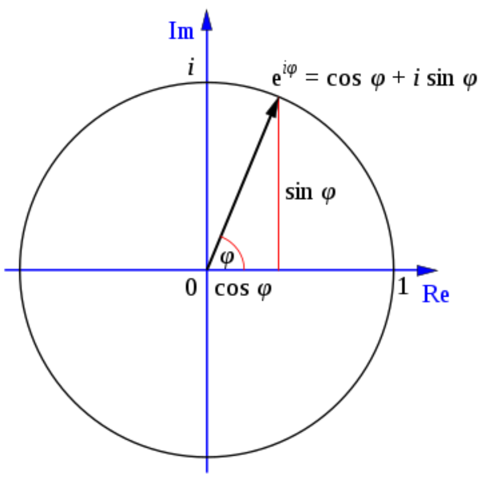
- 헤르츠(Hz)
  - 주파수를 표현하는 단위 , 1초 동안에 진동하는 횟수
  - 전파라는 신호에 국한된 표현

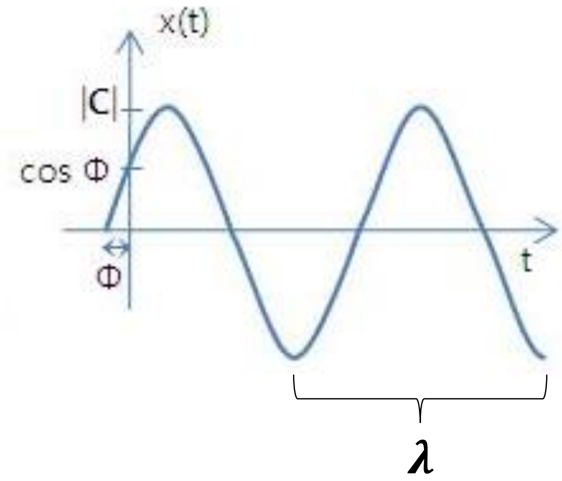


9.1 공간 주파수의 이해

- 코사인 함수 $x(t) = C \cdot cos\,(\,\omega t + \phi\,) = \ C \cdot cos\,(\,2\pi f t + \phi\,)$
  - C : $x(t)$의 진폭
  - ϕ : 위상 (phase)
  - ω : 각 주파수, 혹은 각 속도 (회전의 빠르기 = 주파수 = $2\pi f$)
  - $f = \frac{v}{\lambda}$ (λ : 파장(wavelength), v : 단위속도)
  - 오일러 공식 (Euler's formula)

$$j = \sqrt{-1}, \quad e^{\pm jx} = \cos(x) \pm j\sin(x)$$

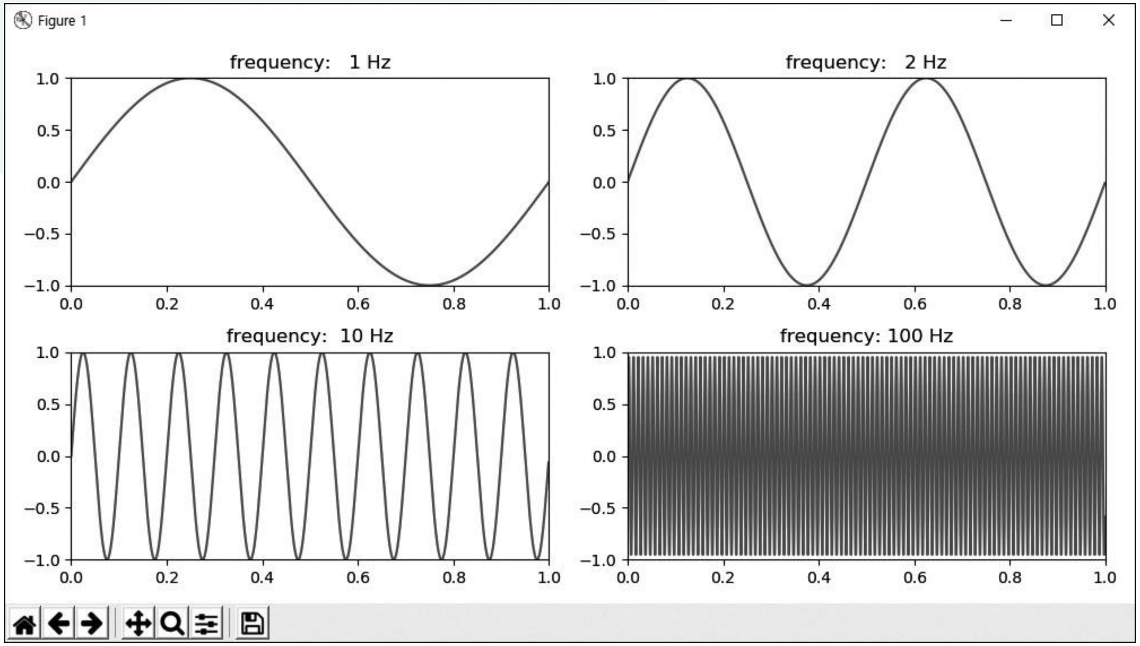




9.1 공간 주파수의 이해

예제 9.1.1 주파수 그리기 |– 01.frequence.py

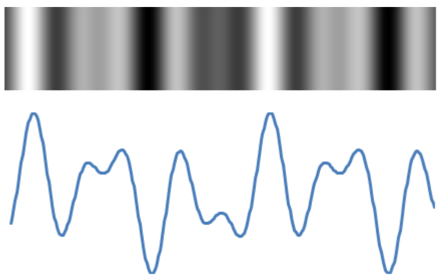
```
import matplotlib.pyplot as plt                      # 그래프그리기 라이브러리 임포트
import numpy as np

t = np.arange(0, 1, 0.001)                           # 샘플링 범위 및 개수
Hz = [1, 2, 10, 100]                                 # 주파수 예시
gs = [np.sin(2 * np.pi * t * h) for h in Hz]         # sin 함수 계산

plt.figure(figsize=(10, 5))
for i, g in enumerate(gs):
    plt.subplot(2, 2, i+1), plt.plot(t, g)           # 그래프 그리기
    plt.xlim(0, 1), plt.ylim(-1, 1 )                 # x, y축 범위 지정
    plt.title("frequency: %3d Hz" % Hz[i])
plt.tight_layout()
plt.show()
```

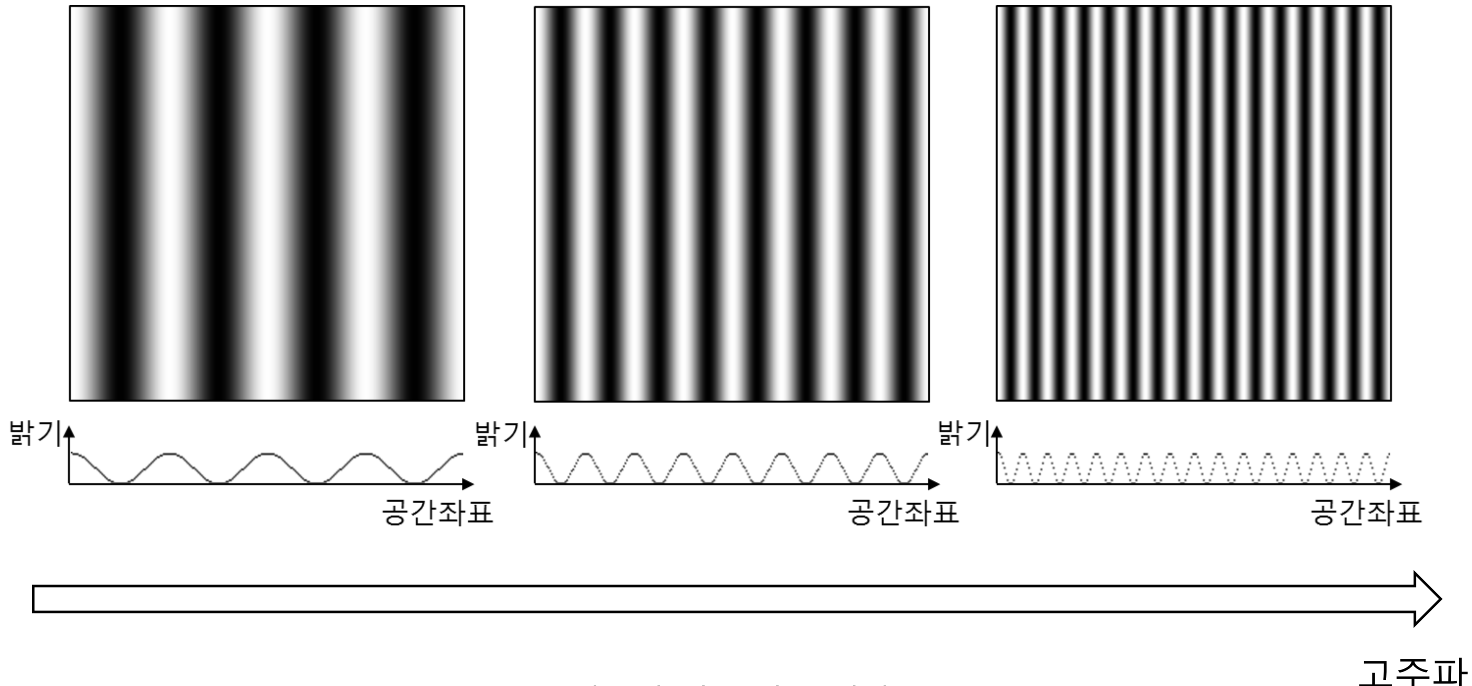


9.1 공간 주파수의 이해

- 영상처리에서는 공간 주파수(spatial frequency) 개념 사용

- 확장된 의미에서 주파수
  - 이벤트가 주기적으로 재발생하는 빈도
    - 예) 화소 밝기의 변화도



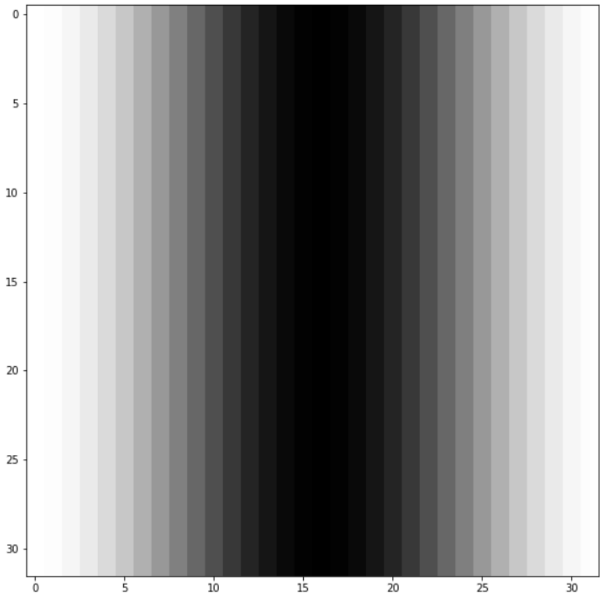


주파수가 서로 다른 영상

9.1 공간 주파수의 이해

- **예제 9.1**

```
import matplotlib.pylab as pylab
import numpy as np
import numpy.fft as fp

im = np.zeros([32,32])
im1 = np.copy(im)

magnitude = 1 # 진폭
phase = 0    # 위상
wavelength = 32 # 파장
frequency = 1/wavelength    # 진동수
omega = 2*np.pi*frequency # 각 주파수 (각 속도)

for n in range(im.shape[1]):
    im1[:, n] += np.cos(omega*n + phase) # 코사인 패턴 이미지 추가


pylab.figure(figsize=(10,10))
pylab.imshow( im1 , cmap='gray')
pylab.show()
```



9.1 공간 주파수의 이해

- 공간 주파수 - 밝기의 변화 정도에 따라서 고주파/저주파 영역으로 분류
  - 저주파 공간 영역
    - 화소 밝기가 거의 변화가 없거나 점진적으로 변화하는 것
    - 영상에서 배경 부분이나 객체의 내부에 많이 있음
  - 고주파 공간 영역
    - 화소 밝기가 급변하는 것
    - 경계부분이나 객체의 모서리 부분

저주파 공간 영역

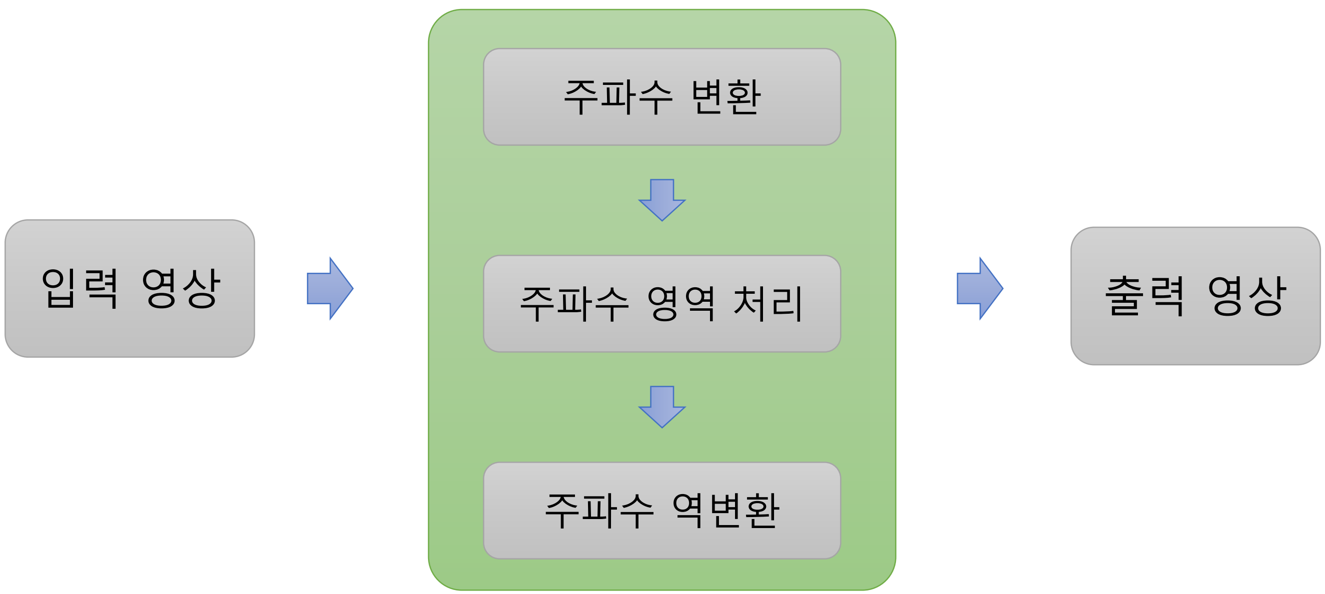
고주파 공간 영역

9.1 공간 주파수의 이해

- 영상을 주파수 영역별로 분리 한다면 ?
  - 경계부분에 많은 고주파 성분 제거한 영상 → 경계 흐려진 영상
  - 고주파 성분만 취한 영상 → 경계나 모서리만 포함하는 영상 즉, 에지 영상

- 공간 영역상에서 저주파 성분과 고주파 성분이 혼합 → 영역 분리해서 선별적 처리 어려움 → 변환영역 처리 필요

- 변환 영역 처리 과정



9.2 이산 푸리에 변환

- 신호의 기본 전제

주기를 가진 신호는 사인파/코사인파의 합으로 표현할 수 있다.

- sin파/cosine파
  - 모든 파형 중에 가장 순수한 파형을 말하는 것으로 사인 또는 코사인 함수로 된 신호

- 주기를 갖는 신호

$$g(t) = 0.3^* g_1(t) - 0.7^* g_2(t) + 0.5^* g_3(t)$$

  - 여러 사인 및 코사인 함수들로 분리 가능
    - 분리된 신호(기저 함수) :

$$g_1(t) = \sin(2\pi * t), \quad g_2(t) = \sin(2\pi * 3t), \quad g_3(t) = \sin(2\pi * 5t)$$

    - 기저 함수에 곱해지는 값(주파수 계수) : 0.3, -0.7 , 0.5

9.2 이산 푸리에 변환

- 주파수 변환 수식

$$g(t)=\int_{-\infty}^{\infty} G(f)\cdot g_f(t)df$$

$g_f(t)$: f 주파수에 대한 기저함수
$G(f)$: 기저함수의 계수

  - 모든 주파수에 대한 기저함수와 그 계수들의 선형 조합
- 연속신호의 주파수 영역 변환
  - 존재하는 모든 기저함수에 대해서 그 계수인 $G(f)$ 를 구하는 것

- 푸리에 변환
  - 사인이나 코사인 함수를 기저함수로 사용

$$g_f(t)=\cos(2\pi ft)+j\cdot\sin(2\pi ft)=e^{j2\pi ft}$$

  - 푸리에 변환 – 원본 신호로 부터 계수 $G(f)$ 를 얻는 것

기저함수로부터 원본 신호를 만드는 것 → 푸리에 역변환

$$G(f)=\int_{-\infty}^{\infty} g(t)\cdot e^{-j2\pi ft}dt$$

$$g(t)=\int_{-\infty}^{\infty} G(f)\cdot e^{j2\pi ft}df$$

9.2 이산 푸리에 변환

- 이산 푸리에 변환(DFT: Discrete Fourier Transform)
  - 디지털 신호(이산신호)에 적용

$$G(k) = \sum_{n=0}^{N-1} g[n] \cdot e^{-j2\pi k \frac{n}{N}}, \qquad k = 0, \cdots, N-1$$

$$g[n] = \frac{1}{N} \sum_{k=0}^{N-1} G(k) \cdot e^{j2\pi k \frac{n}{N}}, \qquad n = 0, \cdots, N-1$$

- $g[n]$: 디지털 신호, $G[k]$: 주파수k에 대한 푸리에 변환 계수
- k와 n은 신호의 원소개수 (유한개)

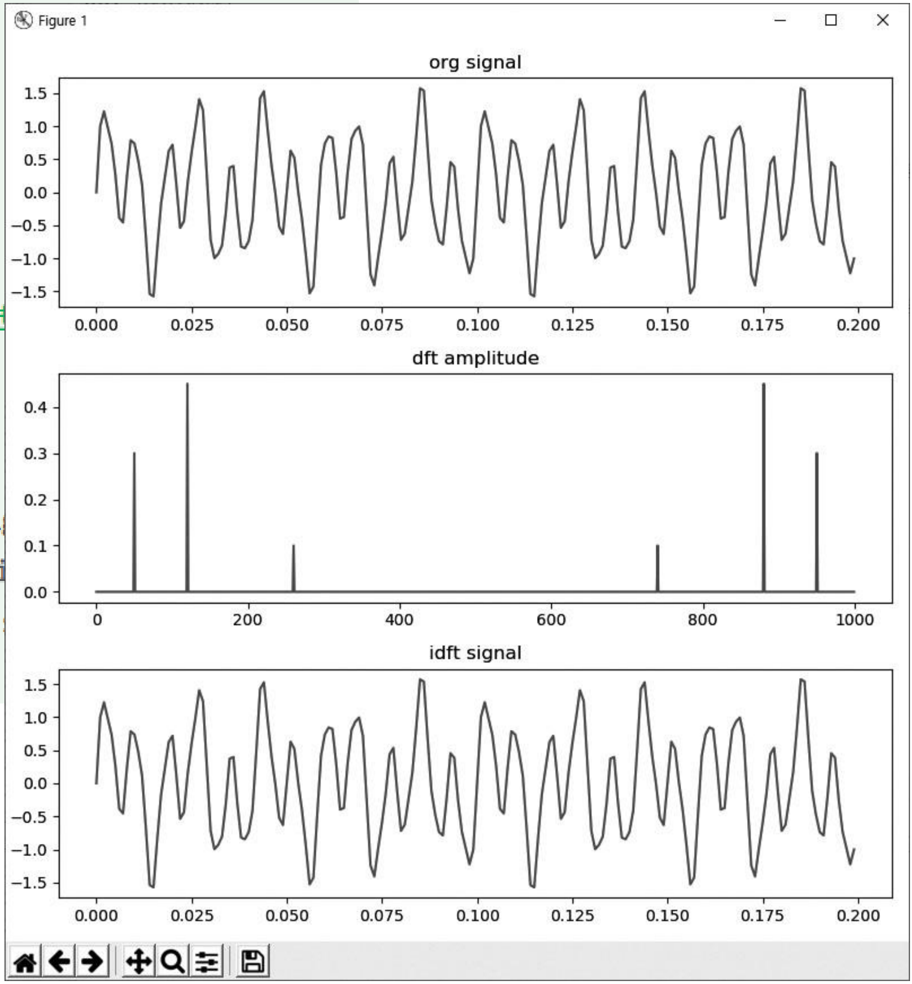
9.2 이산 푸리에 변환

예제 9.2.2 1차원 이산 푸리에 변환 – 03.1d_dft.cpp

```
import numpy as np, math
import matplotlib.pyplot as plt

def exp(knN):
    th = -2 * math.pi * knN                                  # 푸리에 변환 각도값
    return complex(math.cos(th), math.sin(th))               # 복소수 클래스

def dft(g):
    N = len(g)
    dst = [sum(g[n] * exp(k*n/N ) for n in range(N)) for k in range(N) ]
    return np.array(dst)

def idft(g):
    N = len(g)
    dst = [sum(g[n] * exp(-k*n/N) for n in range(N)) for k in range(N) ]
    return np.array(dst) / N

```

9.2 이산 푸리에 변환

```
fmax = 1000                              # 샘플링 주파수 1000Hz: 최대주파수의 2배
dt = 1/fmax                              # 샘플링 간격
t = np.arange(0, 1, dt)                  # Time vector

g1 = np.sin(2 * np.pi * 50 * t)
g2 = np.sin(2 * np.pi * 120 * t)
g3 = np.sin(2 * np.pi * 260 * t)
g = g1 * 0.6 + g2 * 0.9 + g3 * 0.2       # 신호 합성

N = len(g)                               # 신호 길이
df = fmax/N                              # 샘플링 간격
f = np.arange(0, N, df)
xf = dft(g) * dt                         # 저자구현 푸리에 변
g2 = idft(xf) / dt

plt.figure(figsize=(10,10))
plt.subplot(3,1,1), plt.plot(t[0:200], g[0:200]), plt.title("org si
plt.subplot(3,1,2), plt.plot(f, np.abs(xf) ), plt.title("dft ampli
plt.subplot(3,1,3), plt.plot(t[0:200], g2[0:200]), plt.title("idft 
plt.show()
```



9.2 이산 푸리에 변환

- 2차원 이산 푸리에 변환

$$
\begin{aligned}
G(k,l) &= \sum_{m=0}^{M-1}\left(\sum_{n=0}^{N-1} g[n,m]\cdot e^{-j2\pi k\frac{n}{N}}\right)\cdot e^{-j2\pi t\frac{m}{M}}, \qquad \begin{array}{l} k=0,\ \cdots,\ N-1 \\ l=0,\ \cdots,\ M-1 \end{array} \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} g[n,m]\cdot e^{-j2\pi\left(\frac{kn}{N}+\frac{lm}{M}\right)}
\end{aligned}
$$

- 2차원 이산 푸리에 역변환

$$
\begin{aligned}
g[n,m] &= \frac{1}{NM}\cdot\sum_{m=0}^{M-1}\left(\sum_{n=0}^{N-1} G(k,l)\cdot e^{j2\pi k\frac{n}{N}}\right)\cdot e^{j2\pi l\frac{m}{M}}, \qquad \begin{array}{l} k=0,\cdots,N-1 \\ l=0,\cdots,M-1 \end{array} \\
&= \frac{1}{NM}\cdot\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} G(k,l)\cdot e^{j2\pi\left(\frac{kn}{N}+\frac{lm}{M}\right)}
\end{aligned}
$$

9.2 이산 푸리에 변환

- 사인과 코사인 함수로 변경

$$G(k) = \sum_{n=0}^{N-1} g[n] \cdot \left(\cos\left(-2\pi k \frac{n}{N}\right) + j \cdot \sin\left(-2\pi k \frac{n}{N}\right)\right)$$

- 실수부와 허수부를 구분하여 표현 → 복소수 행렬 생성됨

$$G(k)_{Re} = \sum_{n=0}^{N-1} g[n]_{Re} \cdot \cos\left(-2\pi k \frac{n}{N}\right) - g[n]_{Im} \cdot \sin\left(-2\pi k \frac{n}{N}\right)$$

$$G(k)_{Im} = \sum_{n=0}^{N-1} g[n]_{Im} \cdot \cos\left(-2\pi k \frac{n}{N}\right) + g[n]_{Re} \cdot \sin\left(-2\pi k \frac{n}{N}\right)$$

푸리에 변환

$$g[n]_{Re} = \frac{1}{N} \sum_{n=0}^{N-1} G(k)_{Re} \cdot \cos\left(2\pi k \frac{n}{N}\right) - G(k)_{Im} \cdot \sin\left(2\pi k \frac{n}{N}\right)$$

$$g[n]_{Im} = \frac{1}{N} \sum_{k=0}^{N-1} G(k)_{Im} \cdot \cos\left(2\pi k \frac{n}{N}\right) + G(k)_{Re} \cdot \sin\left(2\pi k \frac{n}{N}\right)$$

푸리에 역변환

9.2 이산 푸리에 변환

- 복소수의 행렬을 확인하려면 → 영상으로 표현
  - 복소수의 실수부와 허수부를 벡터로 간주하여 벡터의 크기 계산 → 주파수 스펙트럼
  - 실수부와 허수부 원소의 기울기 계산 → 주파수 위상

$$|G(k,l)| = \sqrt{Re(k,l)^2 + Im(k,l)^2}$$

$$\theta(k,l) = \tan^{-1}\left[\frac{Im(k,l)}{Re(k,l)}\right]$$

9.2 이산 푸리에 변환

- 주파수 스펙트럼 영상
  - 저주파 영역의 계수값이 고주파 영역에 비해 상대적으로 너무 큼
  - 계수값을 정규화해서 영상으로 표현하면 최저주파 영역만 흰색으로 나타나고 나머지 영역은 거의 검은색으로 나타남 → 계수값에 로그함수 적용

(a) 일반적 정규화 (b) 로그함수 후 정규화

9.2 이산 푸리에 변환

• 로그 적용 정규화 함수

```python
def calc_spectrum(complex):
    if complex.ndim==2:
        dst = abs(complex)                          # sqrt(re^2 + im^2) 계산해줌
    else:
        dst = cv2.magnitude(complex[:, :, 0], complex[:, :, 1])  # OpenCV의 경우
    dst = cv2.log(dst + 1)
    cv2.normalize(dst, dst, 0, 255, cv2.NORM_MINMAX)
    return cv2.convertScaleAbs(dst)
```

복소수 객체 행렬 사용 - 2차원 행렬

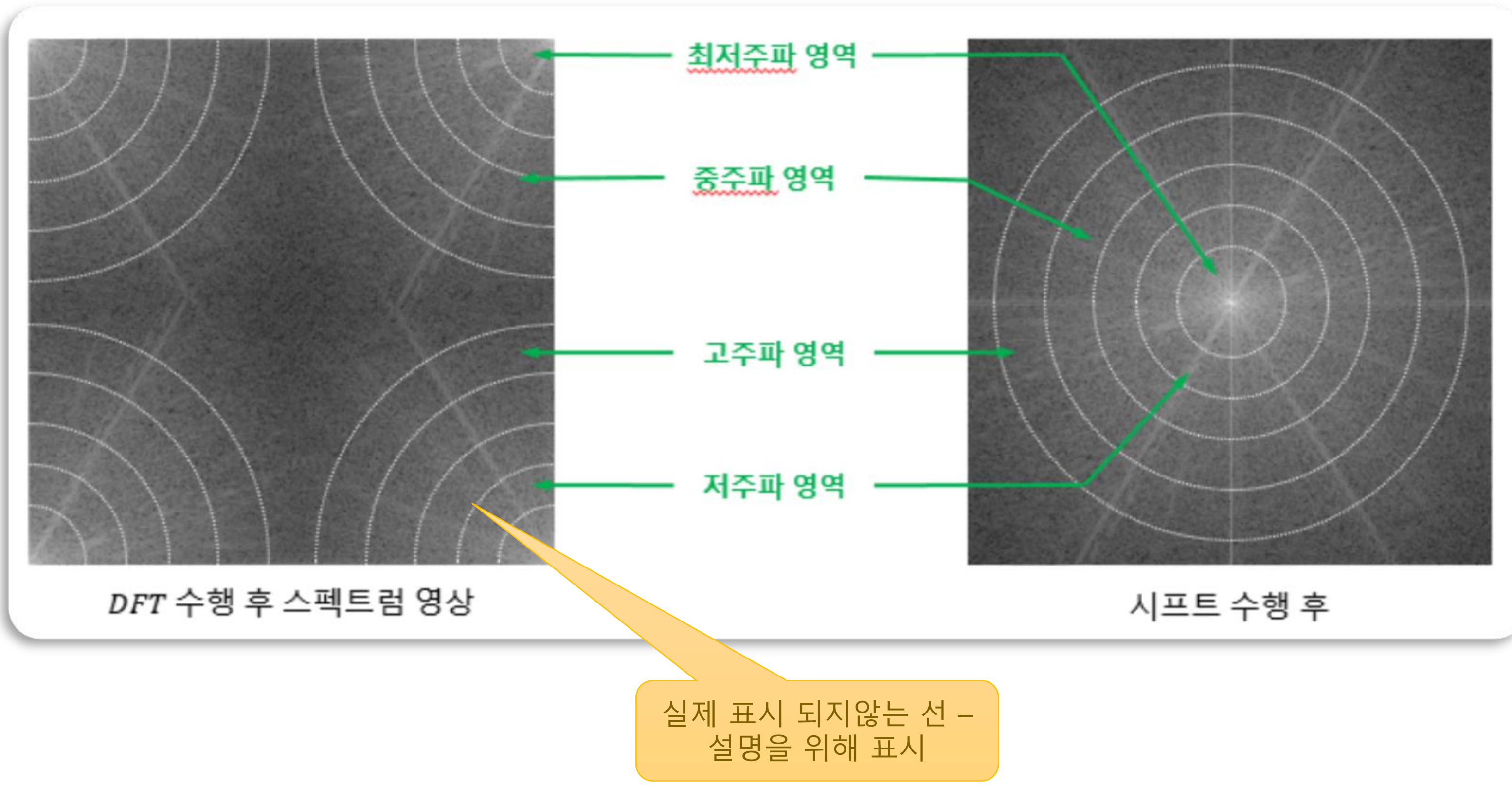
복소수를 2채널로 구성
- 3차원 행렬

log 함수는 0에서
무한대이기 때문에 1을 더함

윈도우 표시 위해 uchar 형변환

9.2 이산 푸리에 변환

- 주파수 스펙트럼 영상
  - 저주파 영역이 모서리 부분에 위치, 고주파 부분이 중심부에 위치
  - 사각형의 각 모서리를 중심으로 원형의 밴드를 형성하여 주파수 영역 분포
    - 해당 주파수 영역 처리시 불편함 → 모양 변경 필요



9.2 이산 푸리에 변환

- 해결 방법 - 셔플링(shuffling)
  - 1사분면과 3사분면의 영상 맞교환, 2사분면과 4사분면의 영상 맞교환
    - 푸리에 변환 주파수 스펙트럼의 주기가 180도이기 때문에 가능
  - 중심이 최저주파 영역, 바깥쪽으로 갈수록 고주파 영역
    - 원형의 밴드로 주파수 영역 쉽게 구분 가능

9.2 이산 푸리에 변환

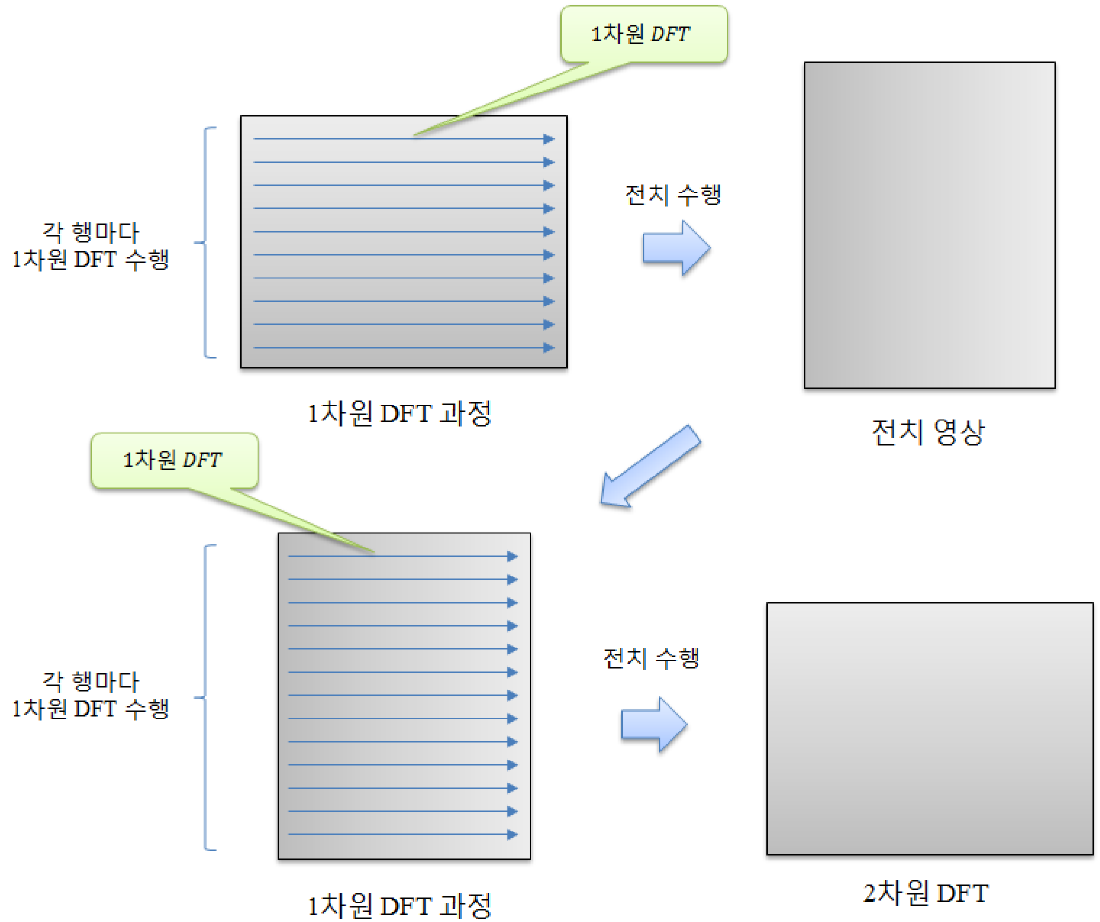
- 셔플링 함수 구현

```
def fftshift(img):
    dst = np.zeros(img.shape, img.dtype)
    h, w = dst.shape[:2]
    cy, cx = h // 2, w // 2                          # 몫연산자
    dst[h-cy:, w-cx:] = np.copy(img[0:cy, 0:cx ])    # 1사분면-> 3사분면
    dst[0:cy, 0:cx ] = np.copy(img[h-cy:, w-cx:])    # 3사분면-> 1사분면
    dst[0:cy, w-cx:] = np.copy(img[h-cy:, 0:cx ])    # 2사분면-> 4사분면
    dst[h-cy:, 0:cx ] = np.copy(img[0:cy, w-cx:])    # 4사분면-> 2사분면
    return dst
```

복사하지 않고 원본 영상을 목적 영상에 할
당하면 참조가 되어 결과에 오류 발생함

9.2 이산 푸리에 변환

- 2차원 DFT 수행 방법
  - 1차원 푸리에 변환 결과 전치 → 다시 1차원 푸리에 변환 수행 → 다시 전치



9.2 이산 푸리에 변환

예제 9.2.3 2차원 이산 푸리에 변환 – 04.2d_dft.py

```
import numpy as np, cv2
from Common.dft2d import dft, idft, calc_spectrum, fftshift    # dft 관련 함수 임포트

def dft2(image):
    tmp = [dft(row) for row in image]
    dst = [dft(row) for row in np.transpose(tmp)]
    return np.transpose(dst)                                   # 전치 환원 후 반환

def idft2(image):
    tmp = [idft(row) for row in image]
    dst = [idft(row) for row in np.transpose(tmp)]
    return np.transpose(dst)                                   # 전치 환원 후 반환

def ck_time(mode=0):                                           # 수행시간 체크 함수
    global stime                                               # 함수 내부에서 값 유지위해
    if (mode ==0 ):
        stime = time.perf_counter()
    elif (mode==1):
        etime = time.perf_counter()
        print(“수행시간 = %.5f sec” % (etime - stime))          # 초 단위 경과 시간

image = cv2.imread(“images/dft_240.jpg”, cv2.IMREAD_GRAYSCALE)
if image is None: raise Exception(“영상파일 읽기 에러”)
```

1차원 DFT 2번 수행

시작 시간 측정

종료 시간 측정 및 수행시간 계산

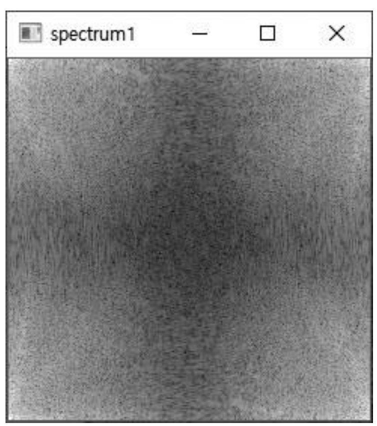
9.2 이산 푸리에 변환

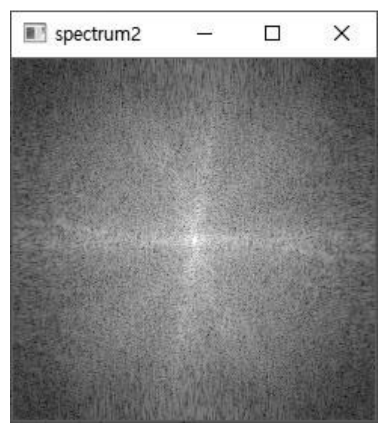
```
ck_time(0)                                   # 시작 시간 체크
dft = dft2(image)                            # 2차원 DFT 수행
spectrum1 = calc_spectrum(dft)               # 주파수 스펙트럼 영상
spectrum2 = fftshift(spectrum1)              # np.fft.fftshift() 사용 가능
idft = idft2(dft).real                       # 2차원 IDFT 수행
ck_time(1)                                   # 종료 시간 체크

cv2.imshow("image", image)
cv2.imshow("spectrum1", spectrum1)
cv2.imshow("spectrum2", spectrum2)
cv2.imshow("idft_img", cv2.convertScaleAbs(idft))
```

복소 객체의 실수부만 가져옴

- 실행결과

```
Run: 04.2d_dft
C:\Python\python.exe D:/source/chap09/04.2d_dft.py
수행시간 = 157.06352 sec
```





단원 요약

- 신호처리에서 주파수라는 말은 1초 동안에 진동하는 횟수라 정의한다. 영상처리에서는 화소 밝기의 변화의 정도를 말한다. 이를 공간 주파수라 하며, 공간 주파수는 밝기가 얼마나 빨리 변화하는가에 따라서 고주파와 저주파 영역으로 분류한다.

- 2. 주기를 갖는 신호는 여러 개의 사인 및 코사인 함수들로 분리할 수 있다. 분리된 신호들을 기저 함수라 하며, 기저 함수에 곱해지는 값을 주파수 계수라 한다.

- 3. 신호를 주파수로 변환하는 것은 각 주파수의 기저 함수들에 대한 계수를 찾는 것이다. 또한, 주파수 영역에서의 역변환은 각 기저 함수와 그 계수들로부터 원본 신호를 재구성하는 것이다.

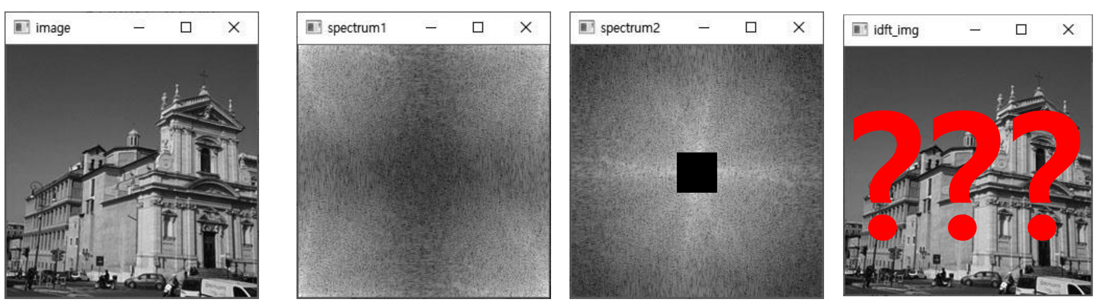
- 4. 푸리에 변환을 수행하면 복소수의 결과가 생성되며, 이것을 영상으로 확인하기 위해서는 복수소의 실수부와 허수부를 벡터로 간주하여 벡터의 크기(magnitude)를 구한다. 이것을 주파수 스펙트럼이라 한다. 복소수 클래스를 이용할 경우, abs() 함수로 절대값을 구하면 크기가 되고, 실수부와 허수부가 2채널 행렬이면, cv2.magnitude() 함수로 구한다.

- 5. DFT 주파수 스펙트럼 영상은 저주파 영역이 영상의 모서리 부분에 위치하고, 고주파 부분이 중심부에 있어서 해당 주파수 영역에서 처리가 어렵다. 이 문제는 시프트(shift) 연산을 통해서 제1 사분면과 제3 사분면의 영상을 맞바꾸고, 제2 사분면과 제4 사분면의영상을 맞바꿈으로써 해결할 수 있다.

9. 실습 과제

- (과제)
  - 1) PPT 11p 예제 9.1을 수정하여 다음 성분을 갖는 이미지를 생성하시오.
    - 성분 1. cosine 신호. 진폭: 0.3, 주파수: 1, 위상: 0
    - 성분 2. cosine 신호. 진폭: 0.5, 주파수: 4, 위상: 0
    - 성분 3. cosine 신호. 진폭: -0.2, 주파수: 16, 위상: 10
    - 성분 4. 학생만의 cosine 신호.
  - 2) 위 3가지 성분을 합친 이미지에 FFT를 가한 결과를 출력 후 논하시오.

- (보너스)
  - 예제 9.2.3의 DFT-shifted 이미지의 중심부 반경 10pixel 만큼의 값을 0으로 바꾸고, 다시 iDFT를 적용하여 복원한 이미지는 어떻게 생겼는가?
    - (비고: 역 shift 연산을 구현해야 한다.)



9. 실습 규칙

- 실습 과제는 실습 시간내로 해결해야 합니다.
  - 해결 못한경우 실습 포인트를 얻지 못합니다.
  - -> 집에서 미리 예습하고 오길 권장합니다.
- 코드 공유/보여주기 금지. 의논 가능.
- 보너스문제까지 해결한 학생은 조기 퇴실 가능