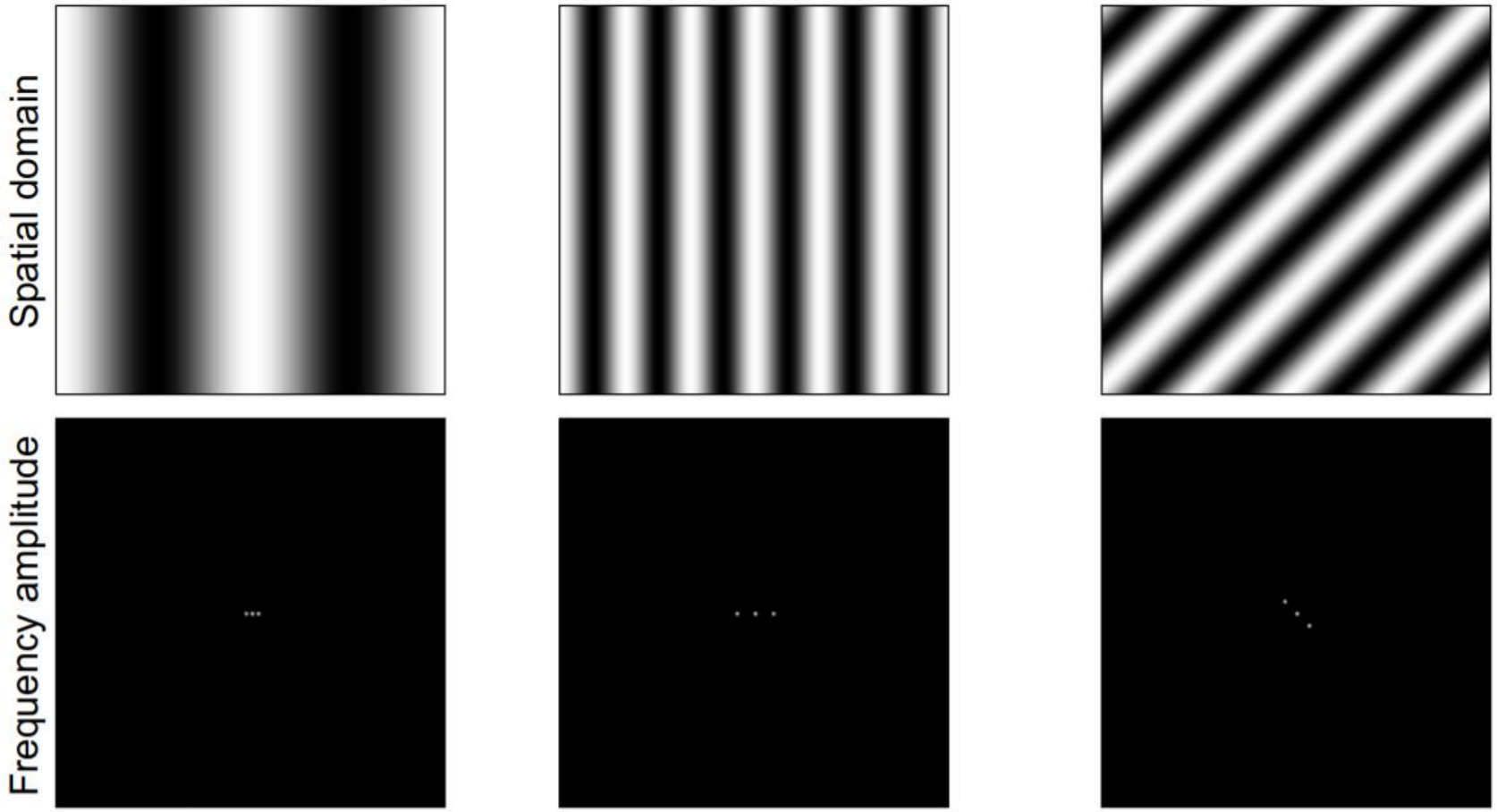
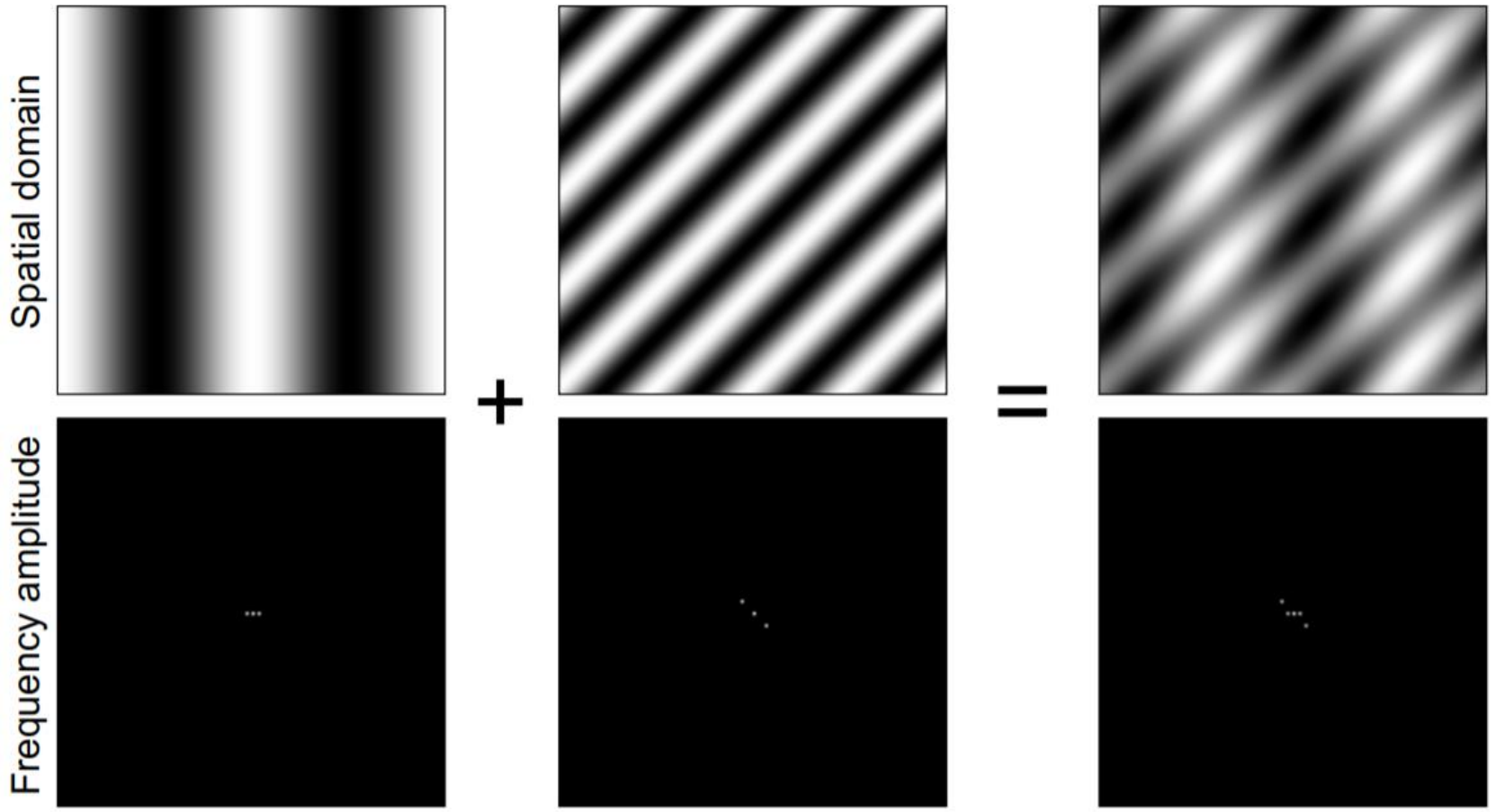
CHAPTER 09
변환영역 처리 (2)

PART 02 영상 처리와 OpenCV 함수 활용

Fourier transform 보강

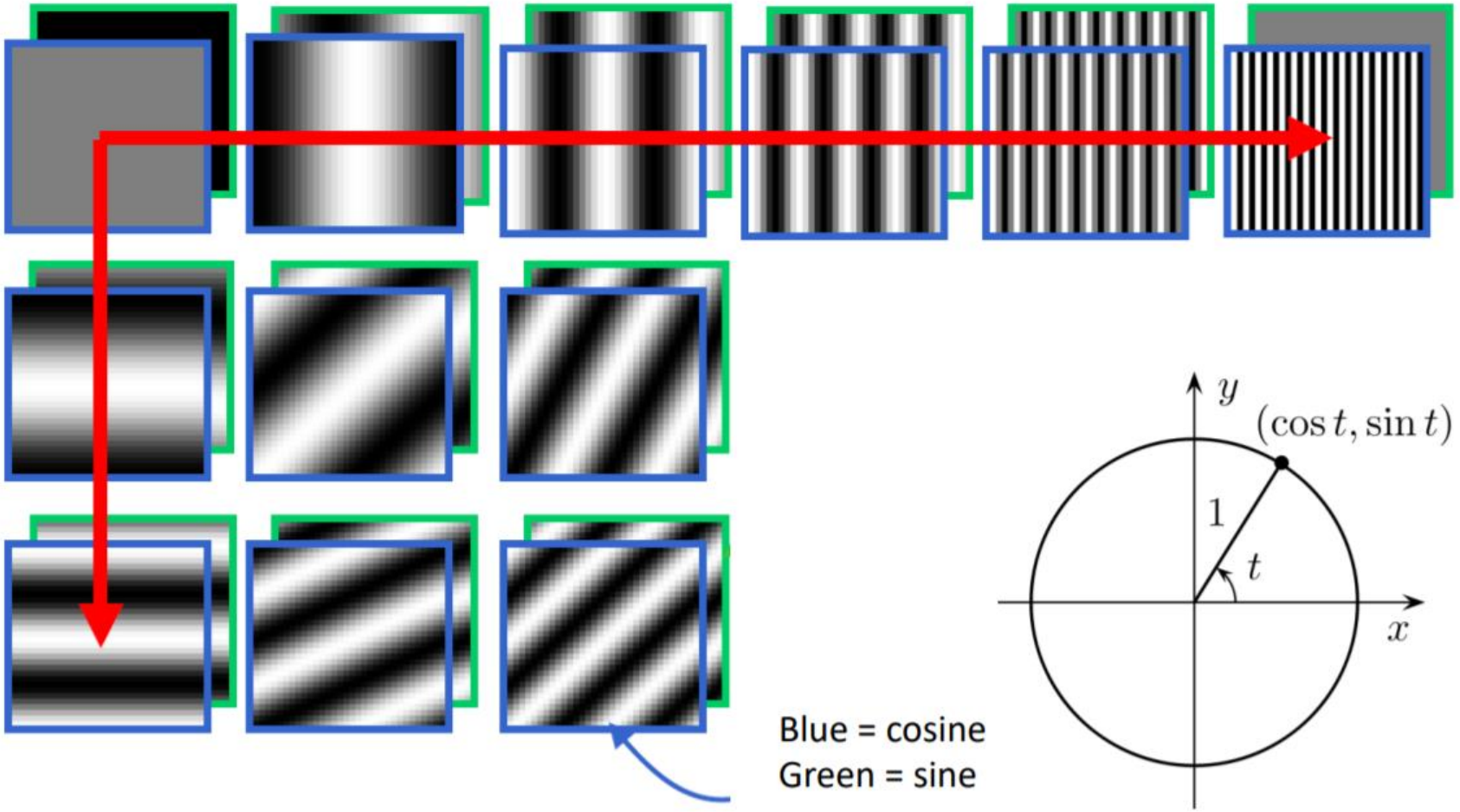


Fourier transform 보강



Fourier Bases

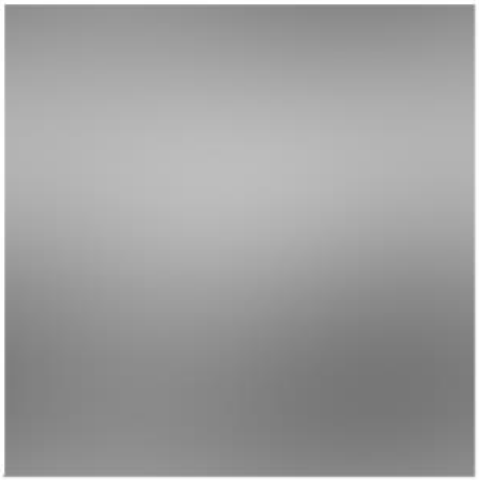
■ 이미지를 ‘fast vs. slow’한 변화를 가진 basis로 쪼갠다



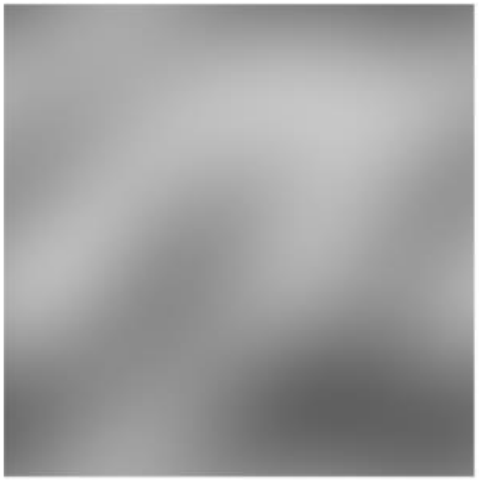
Basis reconstruction

Full image

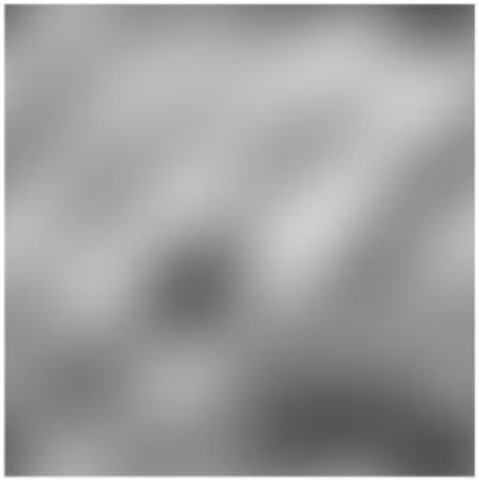
First 1 basis fn



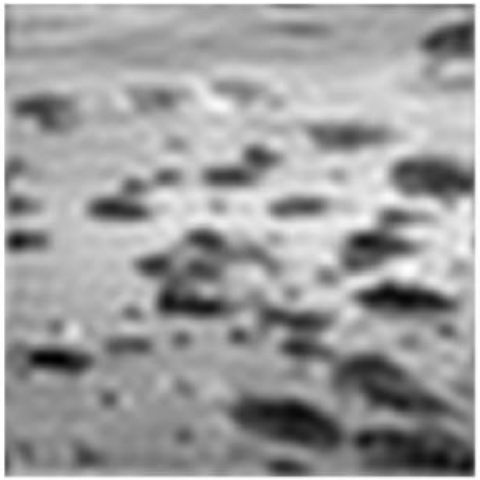
First 4 basis fns



First 9 basis fns



First 16 basis fns

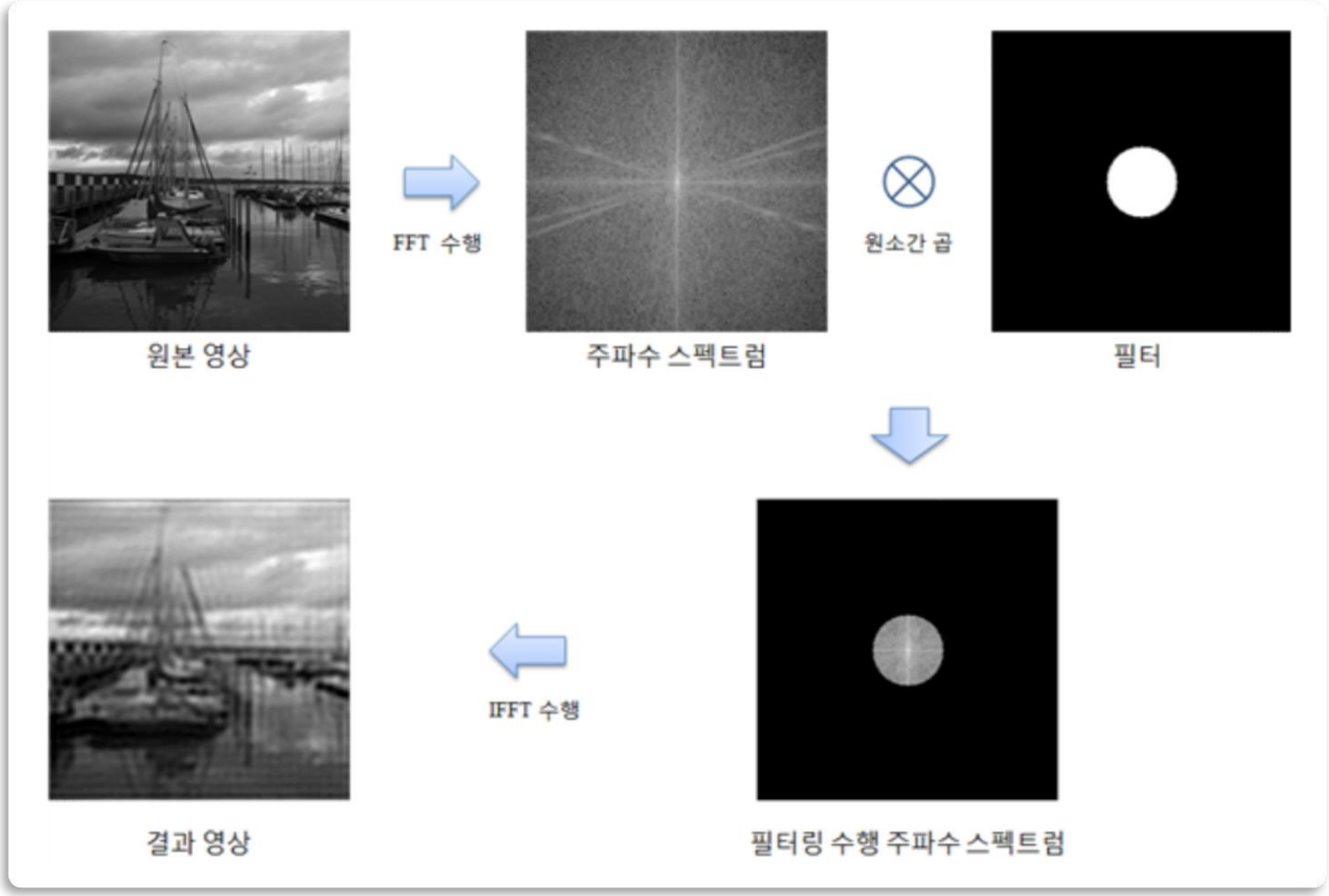


First 400 basis fns

9.4 FFT를 이용한 주파수 영역 필터링

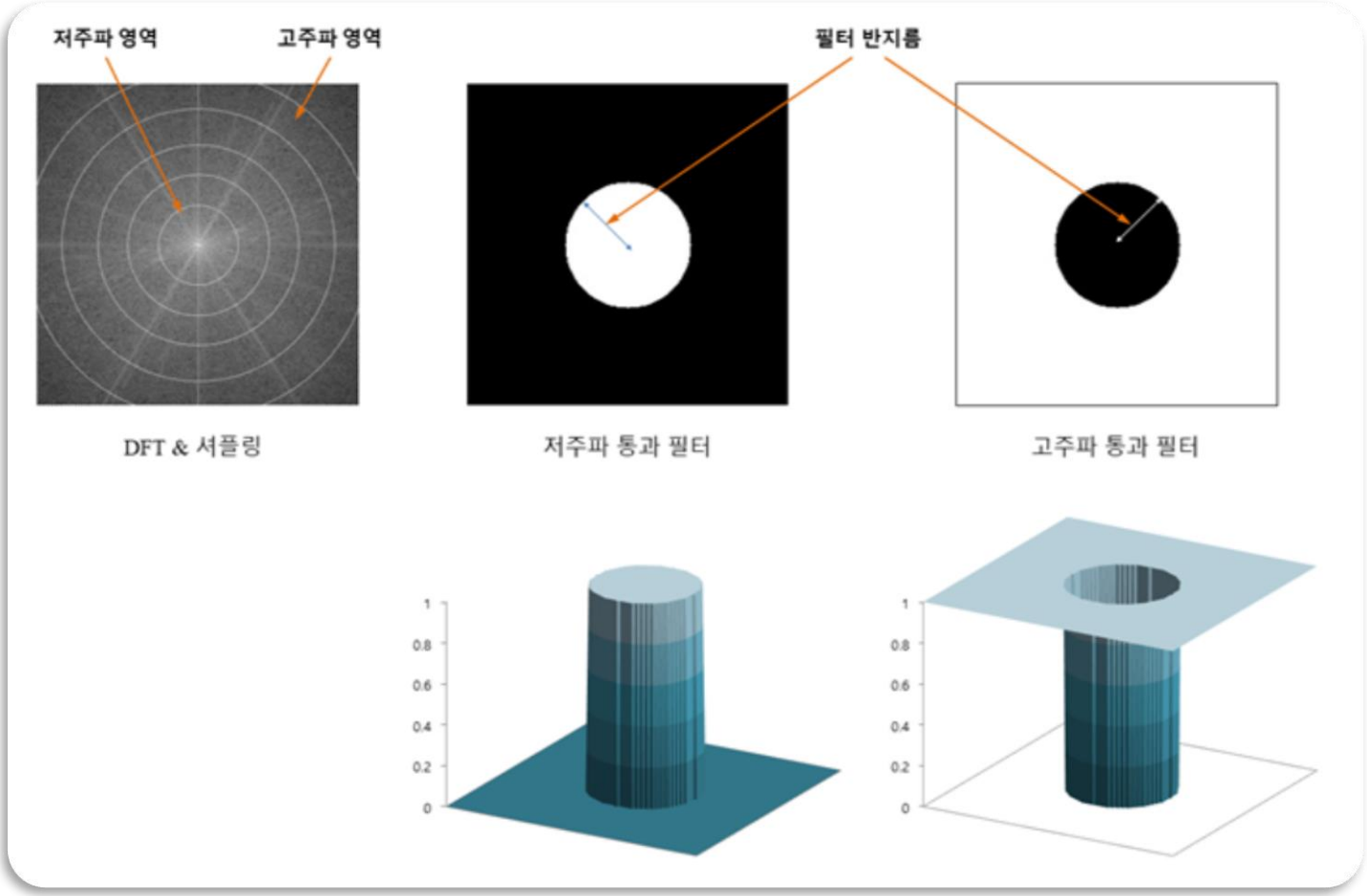
9.4.1 주파수 영역 필터링의 과정

- 영상의 주파수 영역 변환
  - 저주파 영역과 고주파 영역 분리 → 특정 주파수 영역 강화, 약화, 제거 가능
- 주파수 영역 필터링 과정



9.4.2 저주파 및 고주파 통과 필터링

- 저주파 통과 필터링
  - DFT 변환 영역에서 저주파 영역의 계수들은 통과, 고주파 영역의 계수 차단
- 고주파 통과 필터링
  - 고주파 영역의 계수들을 통과시키고 저주파 영역의 계수들 차단



9.4.2 저주파 및 고주파 통과 필터링

예제 9.4.1 주파수 영역 필터링1 – 06.FFT_filtering1.py

```
01  import numpy as np, cv2
02  from Common.fft2d import fft2, ifft2, calc_spectrum, fftshift # fft 관련 함수 임포트
03
04  def FFT(image, mode=2):
05      if mode == 1: dft = fft2(image)                          # 저자 구현 함수
06      elif mode==2: dft = np.fft.fft2(image)                   # 넘파이 함수
07      elif mode==3: dft = cv2.dft(np.float32(image), flags=cv2.DFT_COMPLEX_OUTPUT)
08      dft = fftshift(dft)                                      # 주파수 시프트
09      spectrum = calc_spectrum(dft)                            # 주파수 스펙트럼 영상
10      return dft, spectrum
11
12  def IFFT(dft, shape, mode=2):
13      dft = fftshift(dft)                                      # 역 시프트
14      if mode == 1: img = ifft2(dft).real
15      if mode == 2: img = np.fft.ifft2(dft).real
16      if mode ==3:  img = cv2.idft(dft, flags=cv2.DFT_SCALE)[:,:,0]
17      img = img[:shape[0], :shape[1]]                          # 영삽입 부분 제거
18      return cv2.convertScaleAbs(img)                  # 절대값 및 uint8 스케일링
19
```

fft 수행 + 스프트 + 스팩트럼 영상 생성

역스프트+ 역fft 수행

9.4.2 저주파 및 고주파 통과 필터링

```
20  image = cv2.imread("images/filter.jpg", cv2.IMREAD_GRAYSCALE)
21  if image is None: raise Exception("영상파일 읽기 에러")
22  cy, cx = np.divmod(image.shape, 2)[0]                    # 행렬 중심점 구하기
23  mode = 3                                                 # FFT 방법 선택
24
25  dft, spectrum = FFT(image, mode)                         # FFT 수행 및 셔플링
26  lowpass = np.zeros(dft.shape, np.float32)                # 저주파 통과 필터
27  highpass = np.ones(dft.shape, np.float32)                # 고주파 통과 필터
28  cv2.circle(lowpass , (cx, cy), 30, (1, 1), -1)           # 2개 채널로 값 지정
29  cv2.circle(highpass, (cx, cy), 30, (0, 0), -1)
30
31  lowpassed_dft = dft * lowpass                            # 주파수 필터링
32  highpassed_dft = dft * highpass
33  lowpassed_img = IFFT(lowpassed_dft, image.shape, mode)   # 푸리에 역변환
34  highpased_img = IFFT(highpassed_dft, image.shape, mode)
35
36  cv2.imshow("image", image)
37  cv2.imshow("lowpassed_img", lowpassed_img)               # 역푸리에 변환 영상
38  cv2.imshow("highpased_img", highpased_img)
39  cv2.imshow("spectrum_img", spectrum)
40  cv2.imshow("lowpass_spect", calc_spectrum(lowpassed_dft))  # 필터링 후 스펙트럼
41  cv2.imshow("highpass_spect", calc_spectrum(highpassed_dft))
42  cv2.waitKey(0)
```

OpenCV 함수는 필터 행렬 2개 채널로 구성

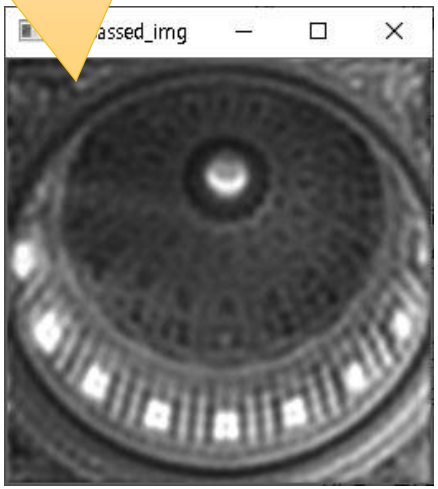
필터링 - 주파수 계수와 필터행렬의 곱

9.4.2 저주파 및 고주파 통과 필터링

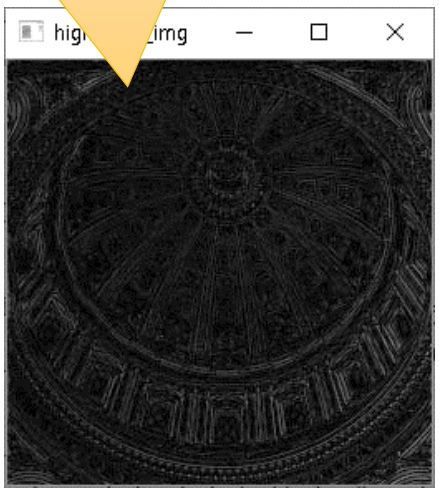
- 실행결과

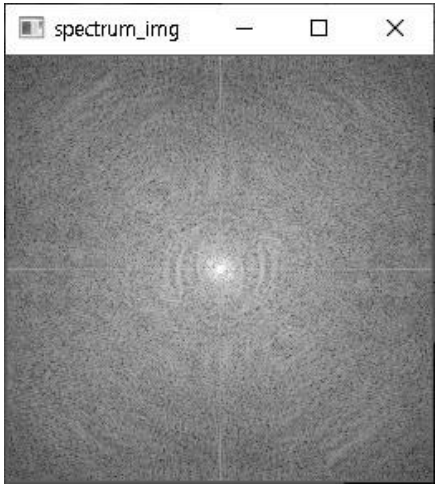
입력영상

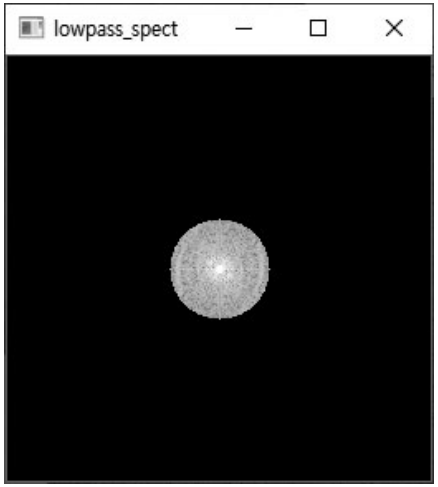
저주파 통과결과 – 흐림 현상



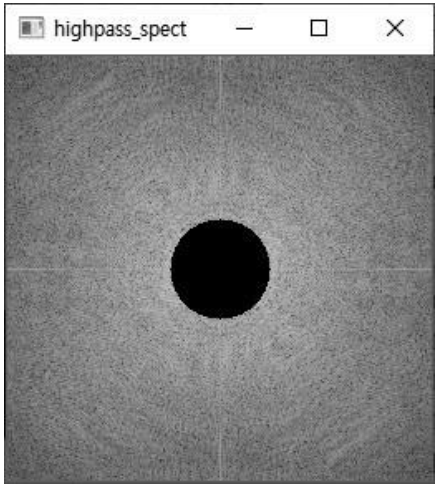
고주파 통과 결과 – 에지 영상







저주파 통과 후
주파수 스펙트럼 영상



고주파 통과 후
주파수 스펙트럼 영상

9.4.3 버터워스, 가우시안 필터링

- 대역 통과 필터
  - 특정한 대역에서 급격하게 값을 제거하기 때문에 결과 영상의 화질 저하
  - 객체의 경계부분 주위로 잔물결 같은 무늬(ringing pattern) 나타남

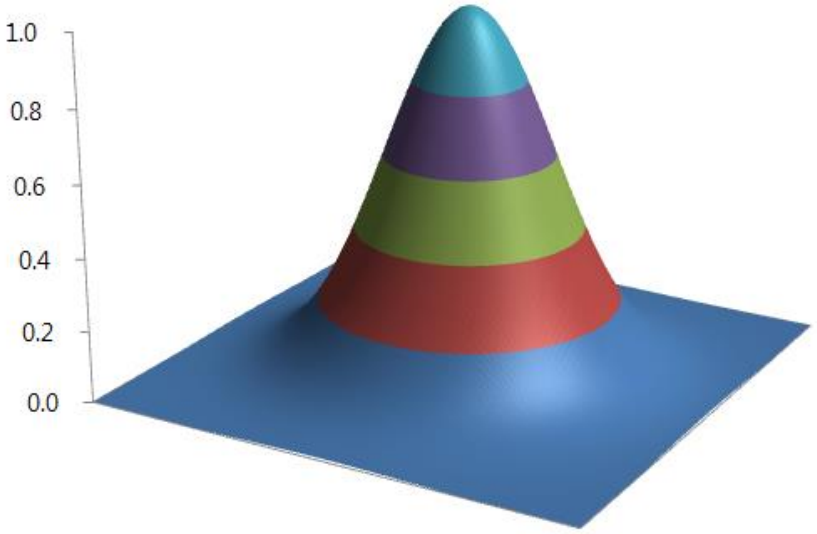
- 해결방법
  - 필터 원소값을 차단 주파수에서 급격하게 0으로하지 않고 완만한 경사 이루도록 구성
  - 버터워즈 필터(Butterworth filter)나 가우시안 필터(Gaussian filter)

9.4.3 버터워스, 가우시안 필터링

- 가우시안 필터
  - 필터 원소의 구성을 가우시안 함수의 수식 분포를 갖게 함으로써 차단 주파수 부분을 점진적으로 구성한 것

$$f(x, y) = exp\left(-\frac{dx^2 + dy^2}{2R^2}\right), \quad \begin{matrix} dx = x - center.x \\ dy = y - center.y \\ R : \text{주파수 차단 반지름} \end{matrix}$$

필터 계수를 밝기로 표현



필터 계수를 3차원 표현

9.4.3 버터워스, 가우시안 필터링

- 버터워즈 필터
  - 차단 주파수 반지름 위치(R)와 지수의 승수인 n 값에 따라서 차단 필터의 반지름과 포물선의 곡률 결정

$$f(x, y) = -\frac{1}{1+\left(\frac{\sqrt{dx^2+dy^2}}{R}\right)^{2n}}$$

$$dx = x - center.x$$
$$dy = y - center.y$$
$$R : \text{주파수 차단 반지름}$$

필터 계수를 밝기로 표현

필터 계수를 3차원 표현

9.4.3 버터워스, 가우시안 필터링

예제 9.4.2 주파수 영역 필터링2 – FFT_filtering2.py

```
01  import numpy as np, cv2
02  from Common.fft2d import FFT, IFFT, calc_spectrum      # 2차원 푸리에 변환 함수 임포트
03  import matplotlib.pyplot as plt                        # 그래프 그리기 임포트
04  from mpl_toolkits.mplot3d import Axes3D                # 3차원 그래프 라이브러리 임포트
05
06  def get_gaussianFilter(shape, R):                      # 가우시안 필터 생성 함수
07      u = np.array(shape)//2
08      y = np.arange(-u[0], u[0], 1)                      # x축 범위 및 간격 지정
09      x = np.arange(-u[1], u[1], 1)                      # y축 범위 및 간격 지정
10      x, y = np.meshgrid(x, y)                           # x, y 좌표 정방행렬 생성
11      filter = np.exp(-(x**2 + y**2)/ (2 * R**2))
12      return x, y, filter if len(shape) < 3 else cv2.merge([filter, filter])
13
14  def get_butterworthFilter(shape, R, n):                # 버터워스 필터 생성 함수
15      u = np.array(shape)//2
16      y = np.arange(-u[0], u[0], 1)
17      x = np.arange(-u[1], u[1], 1)
18      x, y = np.meshgrid(x, y)
19      dist = np.sqrt(x**2 + y**2)
20      filter = 1 / (1 + np.power(dist / R, 2 * n))
21      return x, y, filter if len(shape) < 3 else cv2.merge([filter, filter])
```

OpenCV 함수는
필터 2채널로 구성

9.4.3 버터워스, 가우시안 필터링

```
23  image = cv2.imread("images/filter1.jpg", cv2.IMREAD_GRAYSCALE)
24  if image is None: raise Exception("영상파일 읽기 에러")
25
26  mode = 2
27  dft, spectrum = FFT(image, mode)                          # FFT 수행 및 셔플링
28  x1, y1, gauss_filter = get_gaussianFilter(dft.shape, 30)  # 필터 생성
29  x2, y2, butter_filter = get_butterworthFilter(dft.shape, 30, 10)
30
31  filtered_dft1 = dft * gauss_filter                        # 주파수 공간 필터링– 원소 간 곱셈
32  filtered_dft2 = dft * butter_filter
33  gauss_img = IFFT(filtered_dft1, image.shape, mode)        # 역푸리에 변환
34  butter_img= IFFT(filtered_dft2, image.shape, mode)
35  spectrum1= calc_spectrum(filtered_dft1)                   # 필터링 후, 주파수 스펙트럼 영상
36  spectrum2= calc_spectrum(filtered_dft2)
37
```

가우시안 마스크 생성위한 표준편차

차단 반지름 및 곡률

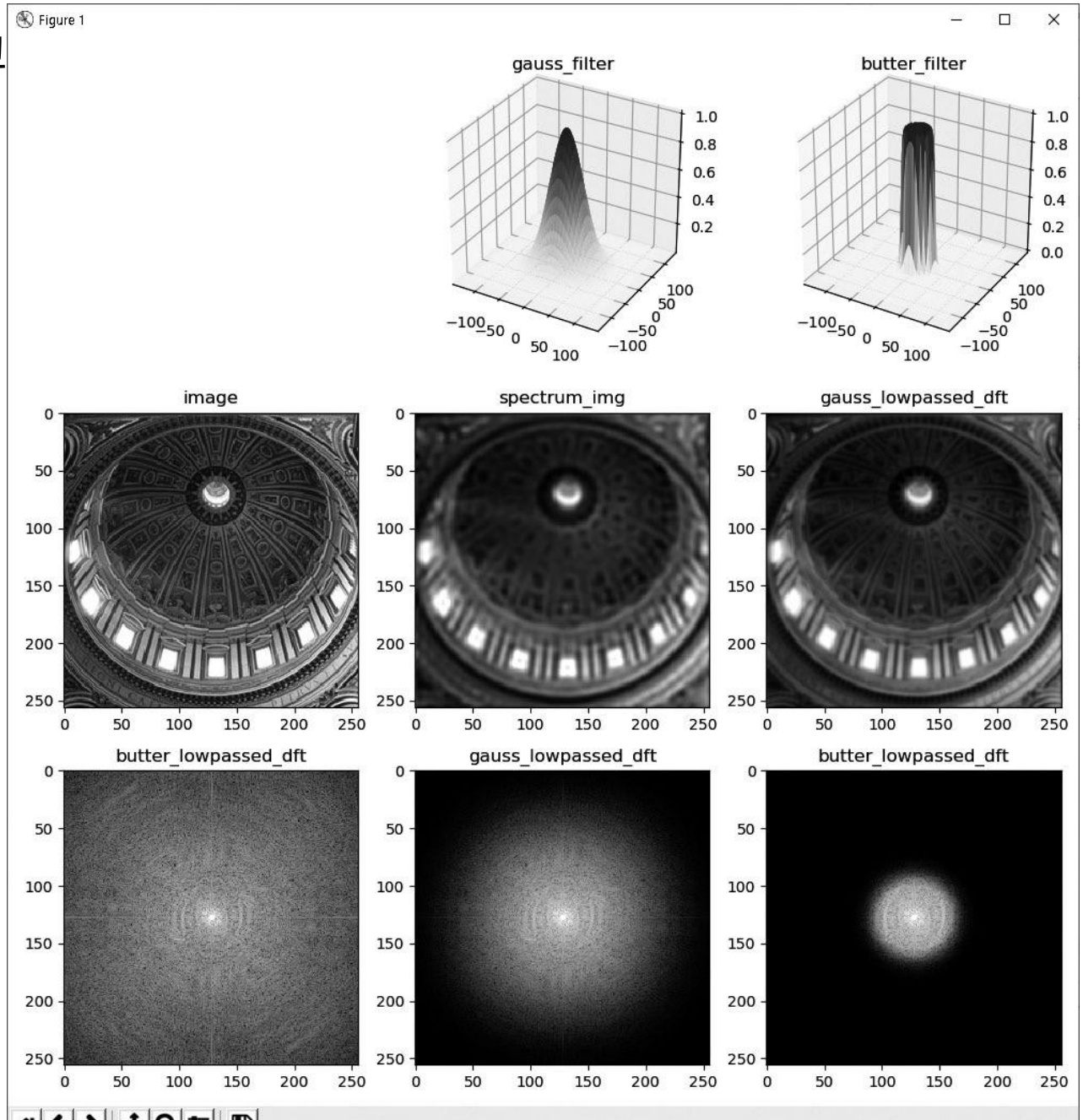
9.4.3 버터워스, 가우시안 필터링

```
38  if mode==3:                                          # OpenCV 함수는 2채널 사용하기에
39      gauss_filter, butter_filter = gauss_filter[:, :, 0], butter_filter[:, :, 0]
40
41  plt.figure(figsize=(10,10))                          # 그래프 생성
42  ax1 = plt.subplot(332, projection='3d')              # 3차원 그래프
43  ax1.plot_surface(x1, y1, gauss_filter, cmap='RdPu'), plt.title("gauss_filter")
44  ax2 = plt.subplot(333, projection='3d')
45  ax2.plot_surface(x2, y2, butter_filter, cmap='RdPu'), plt.title("butter_filter")
46
47  titles = ["input image", "butter_lowpassed_image", "gauss_lowpassed_image",
48            "input spectrum", "gauss_lowpassed_spectrum", "butter_lowpassed_spectrum"]
49  images = [image, butter_img, gauss_img, spectrum, spectrum1, spectrum2]  # 결과 영상 행렬
50  plt.gray()                                           # 명암도 영상으로 표시
51  for i, t in enumerate(titles):
52      plt.subplot(3,3,i+4), plt.imshow(images[i]), plt.title(t)
53  plt.tight_layout(), plt.show()
```

한 채널만 그래프로 그림

9.4.3 버터워스, 가우시안 필터링

- 실행결과



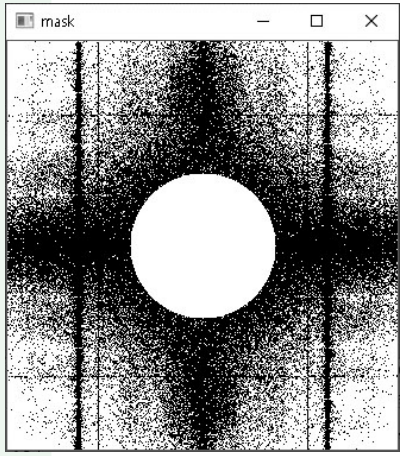
심화예제 – 모아레 제거

심화예제 9.4.3 모아레 제거 – 08.remove_moire.py

```
01  import numpy as np, cv2
02  from Common.fft2d import FFT, IFFT, calc_spectrum      # 고속 푸리에 변환 관련 함수 임포트
03
04  def onRemoveMoire(val):
05      radius = cv2.getTrackbarPos("radius", title)       # 트랙바 위치 값
06      th = cv2.getTrackbarPos("threshold", title)
07
08      mask= cv2.threshold(spectrum_img, th, 255, cv2.THRESH_BINARY_INV)[1]
09      y, x = np.divmod(mask.shape, 2)[0]                 # 마스크 중심 좌표
10      cv2.circle(mask, (x, y), radius, 255, -1)          # 마스크 중심에 흰색 원 그림
11
12      if dft.ndim<3:
13          remv_dft = np.zeros(dft.shape, np.complex)
14          remv_dft.imag = cv2.copyTo(dft.imag, mask=mask)      # 허수부 복사
15          remv_dft.real = cv2.copyTo(dft.real, mask=mask)      # 실수부 복사
16      else:
17          remv_dft = cv2.copyTo(dft, mask=mask)
18
19      result[:, image.shape[1]:] = IFFT(remv_dft, image.shape, mode)    # 관심 영역에 저장
20      cv2.imshow(title, calc_spectrum(remv_dft))         # 마스킹된 스펙트럼
21      cv2.imshow("result", result)
22
```

마스크 영상에 원을 그림

OpenCV 함수



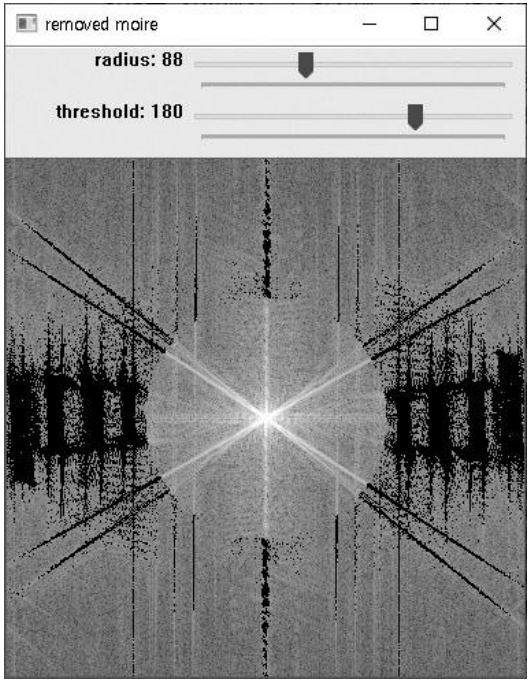
심화예제– 모아레 제거

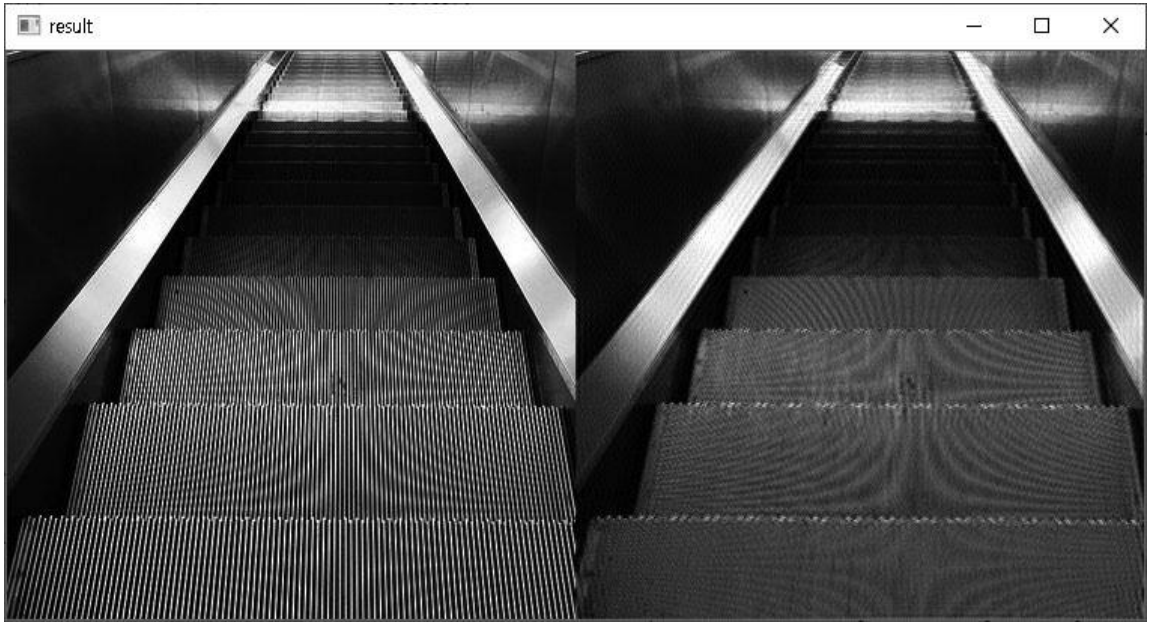
```
23  image = cv2.imread("images/mo2.jpg", cv2.IMREAD_GRAYSCALE)
24  if image is None: raise Exception("영상파일 읽기 에러")
25
26  mode = 3                                          # FFT 방법 선택
27  result = cv2.repeat(image, 1, 2)                  # 원본 영상+결과 영상
28  dft, spectrum_img = FFT(image, mode)              # OpenCV dft() 함수 수행
29
30  title = "removed moire"
31  cv2.imshow("result", result)
32  cv2.imshow(title, spectrum_img)
33  cv2.createTrackbar("radius", title, 10, 255, onRemoveMoire)
34  cv2.createTrackbar("threshold", title, 120, 255, onRemoveMoire)
35  cv2.waitKey(0)
```

디폴트로 OpenCV dft() 함수 수행

심화예제– 모아레 제거

- 실행결과





9.5 이산 코사인 변환

- 이산 코사인 변환(DCT: Discrete Cosine Transform)
  - 1974년 미국의 텍사스 대학(University of Texas)에서 라오 교수 팀이 발표한 직교변환에 관한 논문
    - 영상 신호의 에너지 집중 특성이 뛰어나서 영상 압축에 효과적인 주파수 변환 방법을 찾던 중
  - 이산 푸리에 변환(DFT)에서 실수부만 취하고, 허수부분 제외

주파수 영역 신호

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

공간 영역 신호

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

단, $k=0,\cdots,N-1$, $\quad C(k)=\begin{cases}\sqrt{\frac{1}{N}}, & \text{if } k=0\\ \sqrt{\frac{2}{N}}, & \text{if } k\neq 0\end{cases}$

9.5 이산 코사인 변환

- 2차원 DCT

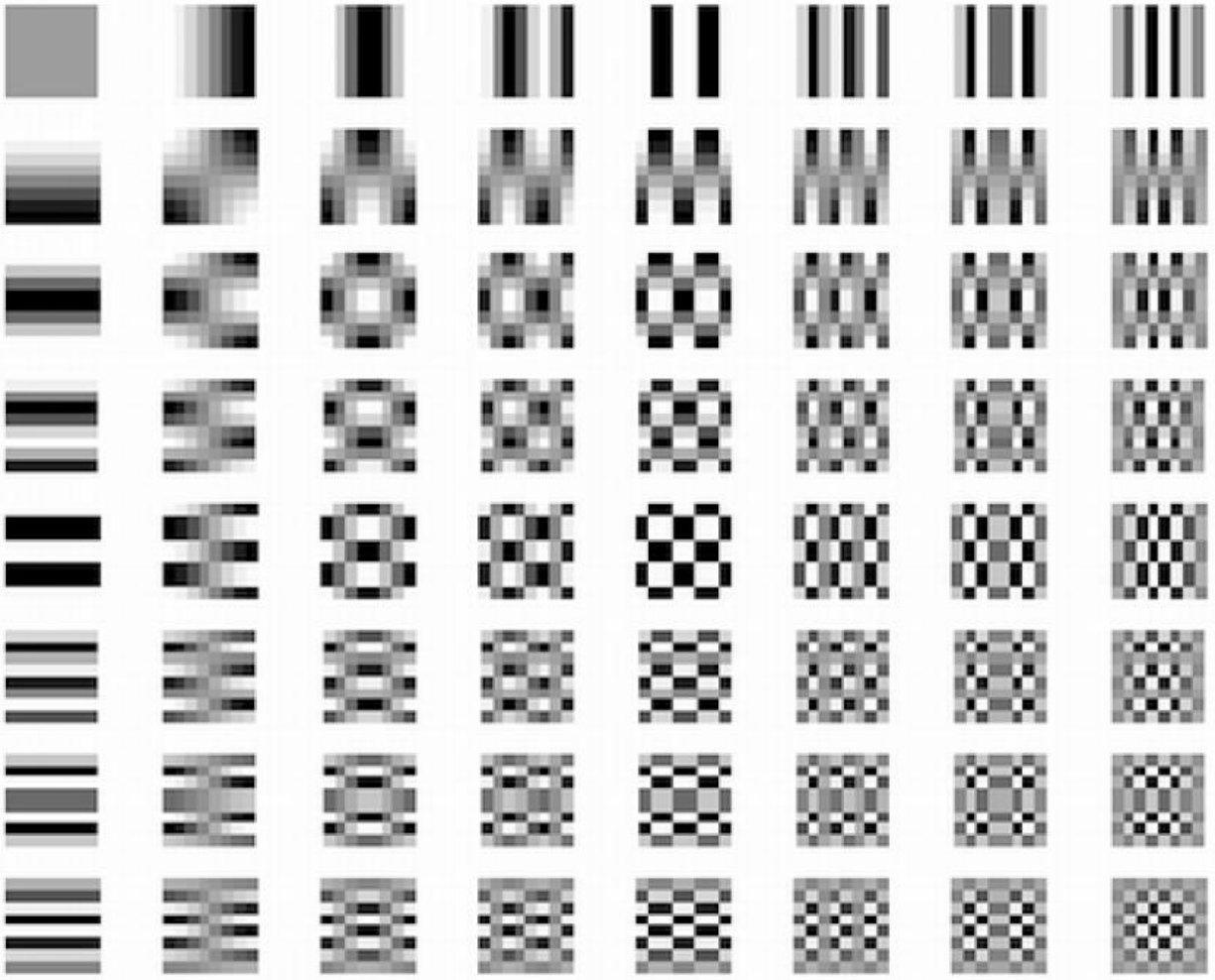
$$F(k,l)=C(k)\cdot C(l)\cdot \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} g[n,m]\cdot \cos\left(\frac{(2n+1)k\pi}{2N}\right)\cdot \cos\left(\frac{(2m+1)l\pi}{2M}\right)$$

$$g[n,m]=\sum_{k=0}^{N-1}\sum_{l=0}^{M-1} C(k)\cdot C(l)\cdot F(k,l)\cdot \cos\left(\frac{(2n+1)k\pi}{2N}\right)\cdot \cos\left(\frac{(2n+1)l\pi}{2M}\right)$$

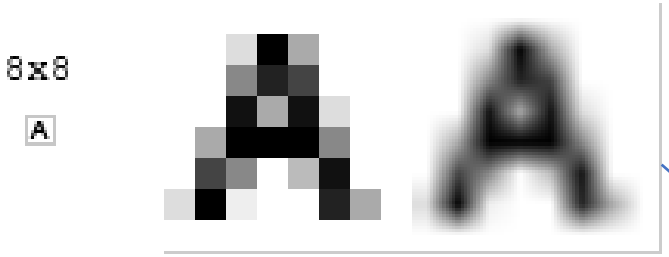
$$\text{단, } k=0,\cdots,N-1, \quad C(k)=\begin{cases}\sqrt{\frac{1}{N}}, & \text{if } k=0\\ \sqrt{\frac{2}{N}}, & \text{if } k\neq 0\end{cases}$$

$$l=0,\cdots,M-1, \quad C(l)=\begin{cases}\sqrt{\frac{1}{N}}, & \text{if } l=0\\ \sqrt{\frac{2}{N}}, & \text{if } l\neq 0\end{cases}$$

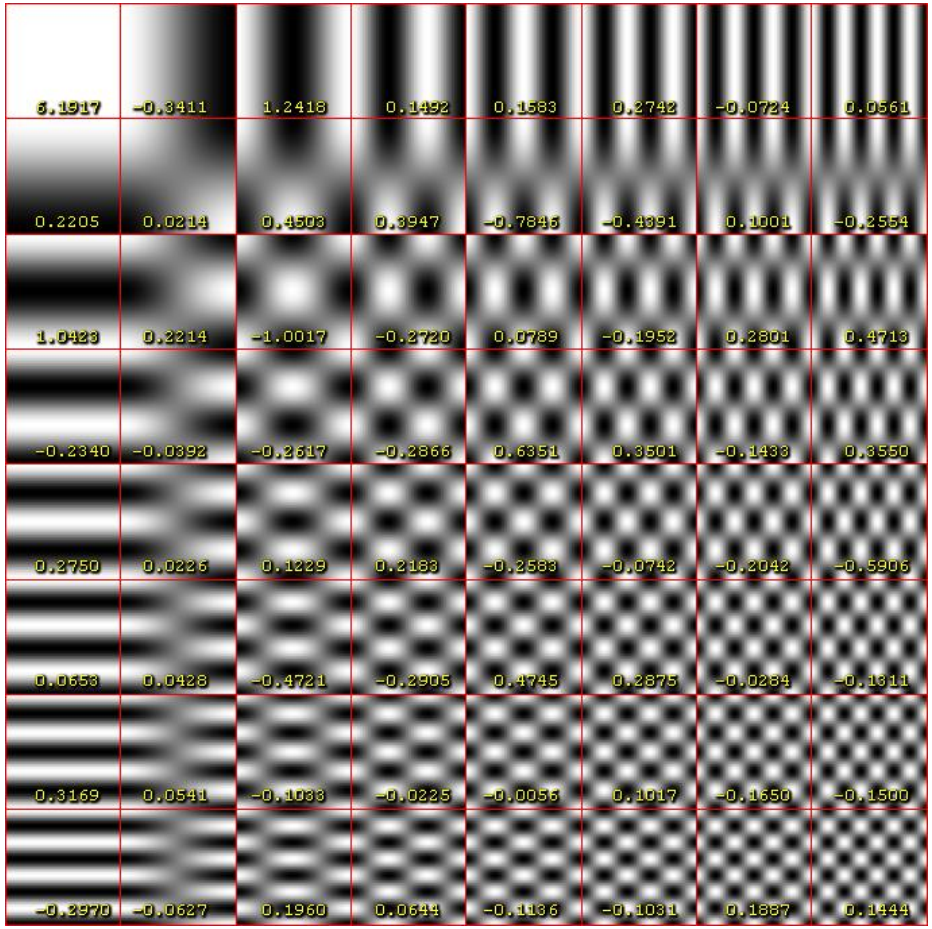
- 전체 영상을 한 번에 변환시키는 것이 아니라 영상을 작은 블록으로 나누어서 수행하기도 함
  - 블록의 크기를 키울수록 압축의 효율이 높아지지만, 변환의 구현이 어려워지고 속도
  - 일반적으로 8×8 크기 사용



기저 신호



A 모양의
이미지의 DCT



\+ 6.192 ×

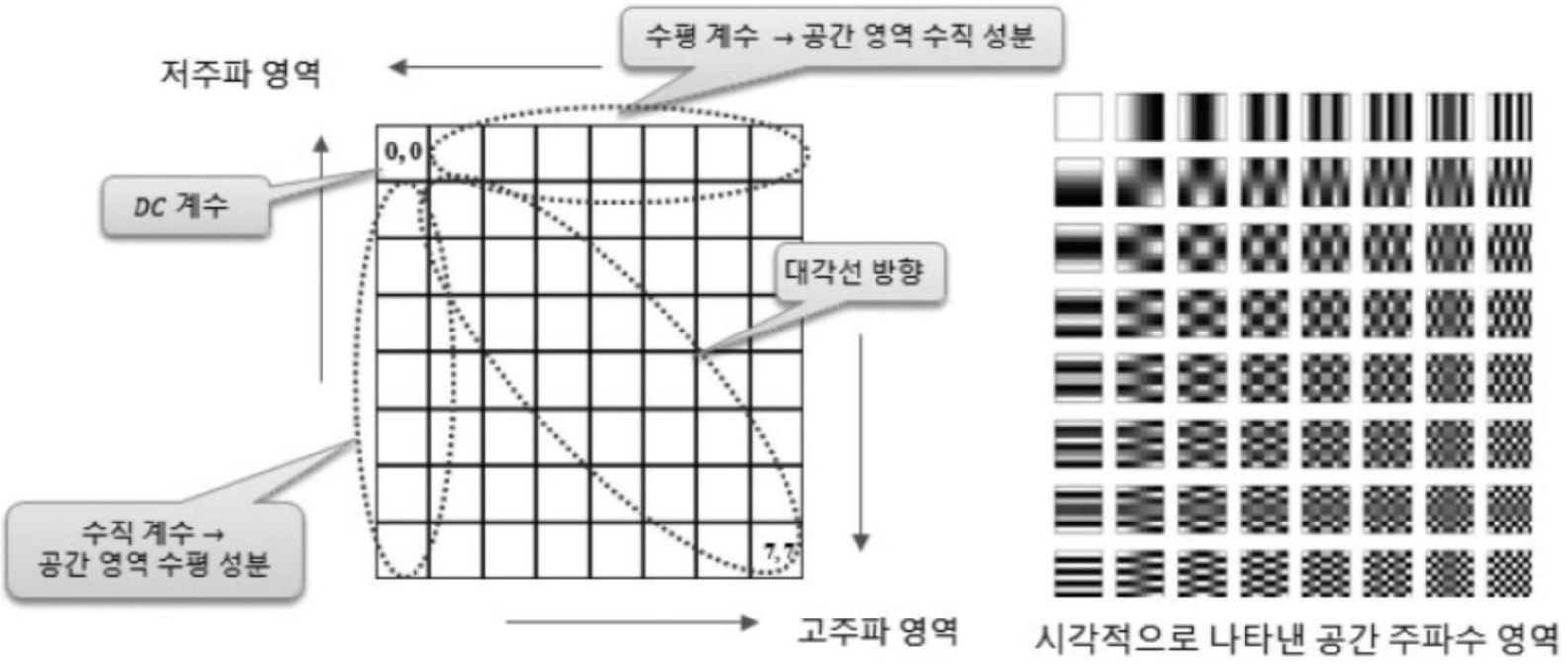
DCT of the image =

$$\begin{bmatrix} 6.1917 & -0.3411 & 1.2418 & 0.1492 & 0.1583 & 0.2742 & -0.0724 & 0.0561 \\ 0.2205 & 0.0214 & 0.4503 & 0.3947 & -0.7846 & -0.4391 & 0.1001 & -0.2554 \\ 1.0423 & 0.2214 & -1.0017 & -0.2720 & 0.0789 & -0.1952 & 0.2801 & 0.4713 \\ -0.2340 & -0.0392 & -0.2617 & -0.2866 & 0.6351 & 0.3501 & -0.1433 & 0.3550 \\ 0.2750 & 0.0226 & 0.1229 & 0.2183 & -0.2583 & -0.0742 & -0.2042 & -0.5906 \\ 0.0653 & 0.0428 & -0.4721 & -0.2905 & 0.4745 & 0.2875 & -0.0284 & -0.1311 \\ 0.3169 & 0.0541 & -0.1033 & -0.0225 & -0.0056 & 0.1017 & -0.1650 & -0.1500 \\ -0.2970 & -0.0627 & 0.1960 & 0.0644 & -0.1136 & -0.1031 & 0.1887 & 0.1444 \end{bmatrix}$$

출처: Wikipedia

9.5 이산 코사인 변환

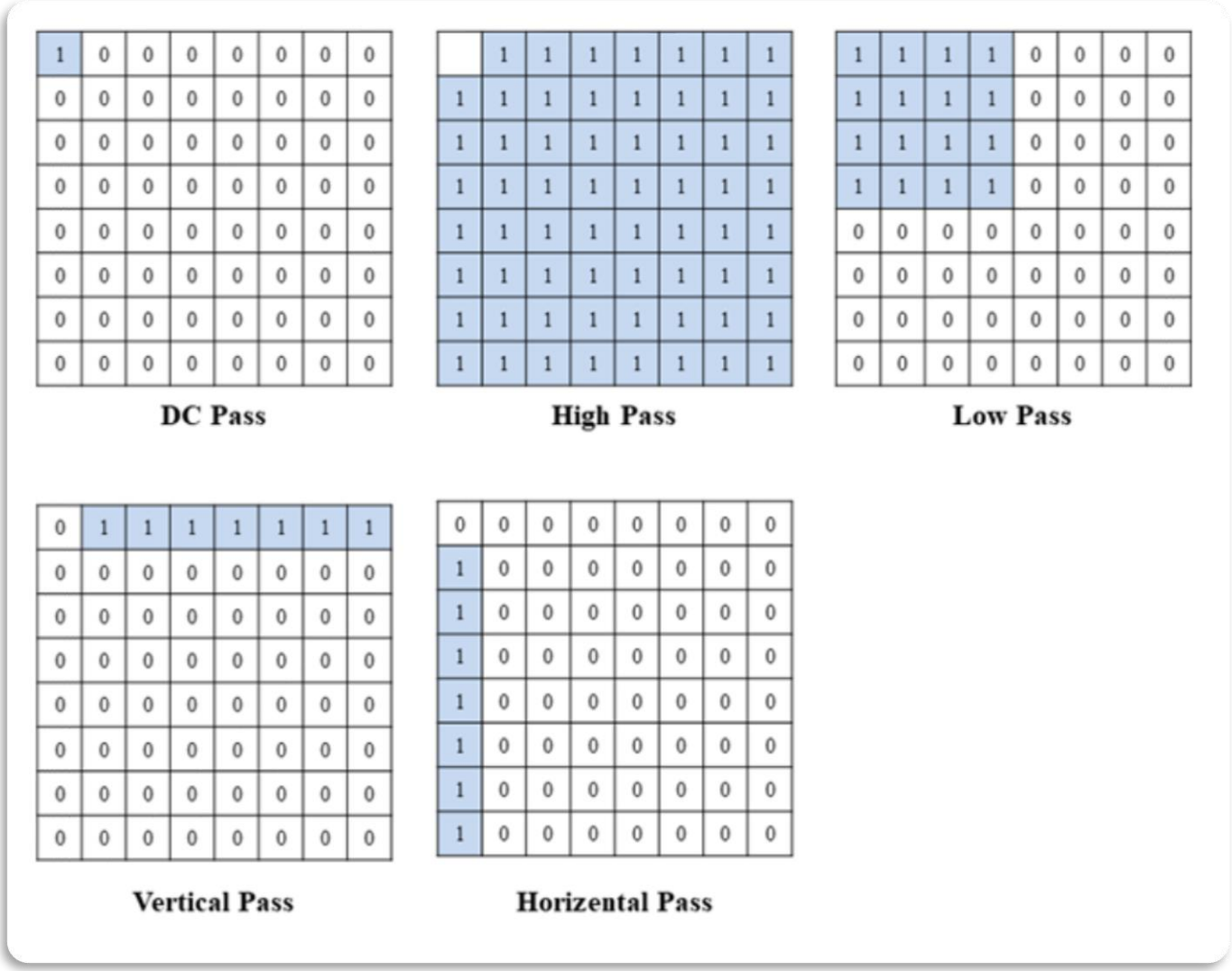
- DCT 계수의 주파수 특성
  - 왼쪽 상단으로 갈수록 저주파 영역이며, 오른쪽 하단으로 갈수록 고주파 영역



〈그림 9.5.1〉 Forward DCT 변환 계수의 주파수 성분

9.5 이산 코사인 변환

- DCT 주파수 영역 필터 구성



9.5 이산 코사인 변환

예제 9.5.2 DCT를 이용한 주파수 영역 필터링 – 10.DCT_filtering.py

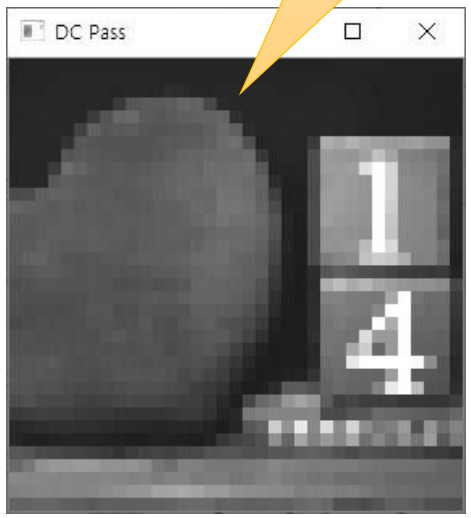
```
01  import numpy as np, cv2
02  from Common.dct2d import dct2, idct2, scipy_dct2, scipy_idct2 # dct 관련 함수 임포트
03
04  def dct2_mode(block, mode):                          # 2차원 dct 함수 – 방식 선택
05      if mode==1: return dct2(block)
06      elif mode==2: return scipy_dct2(block)
07      elif mode==3: return cv2.dct(block.astype('float32'))
08
09  def idct2_mode(block, mode):                         # 2차원 idct 함수 – 방식 선택
10      if mode==1: return idct2(block)
11      elif mode==2: return scipy_idct2(block)
12      elif mode==3: return cv2.dct(block, flags=cv2.DCT_INVERSE)
13
14  def dct_filtering(img, filter, M, N):                # 주파수 영역 필터링 함수
15      dst = np.empty(img.shape, np.float32)
16      for i in range(0, img.shape[0], M):              # 입력 영상 순회
17          for j in range(0, img.shape[1], N):
18              block = img[i:i+M, j:j+N]                # 블록 참조
19              new_block = dct2_mode(block, mode)       # 블록 DCT 수행
20              new_block = new_block * filter           # 곱셈을 통한 필터링
21              dst[i:i+M, j:j+N] = idct2_mode(new_block, mode)   # 역DCT
22      return cv2.convertScaleAbs(dst)
```

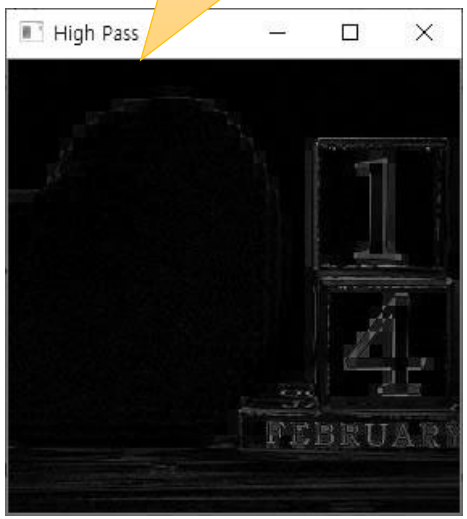
9.5 이산 코사인 변환

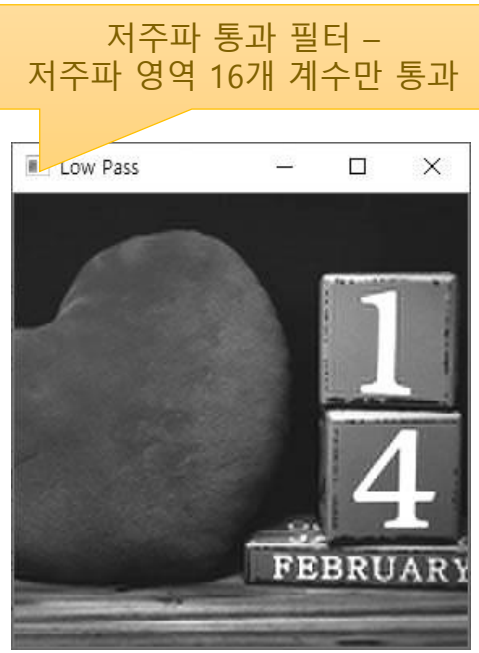
```
24  image = cv2.imread("images/dct.jpg", cv2.IMREAD_GRAYSCALE)
25  if image is None: raise Exception("영상파일 읽기 에러")
26
27  mode = 3                                                    # dct 방식 선택
28  M, N = 8, 8                                                 # 블록 크기
29  filters = [np.zeros((M, N), np.float32) for i in range(5)]  # 5개 필터 생성 및 0으로 초기화
30  titles = ['DC Pass', 'High Pass', 'Low Pass', 'Vertical Pass', 'Horizental Pass' ]
31
32  filters[0][0, 0] = 1                                        # DC 계수만 1 지정 - DC Pass
33  filters[1][:], filters[2][0, 0] = 1, 0                      # 모든 계수 1, DC만 0 지정 - High Pass
34  filters[2][:M//2, :N//2] = 1                                # 1/4 영역 1 지정 - Low Pass
35  filters[3][0, 1:] = 1                                       # 수직 성분 – Vertical Pass
36  filters[4][1:, 0] = 1                                       # 수평 성분 – Horizental Pass
37
38  for filter, title in zip(filters, titles):
39      dst = dct_filtering(image, filter, M, N)
40      cv2.imshow(title, dst)
41  cv2.imshow("image", image)
42  cv2.waitKey(0)
```

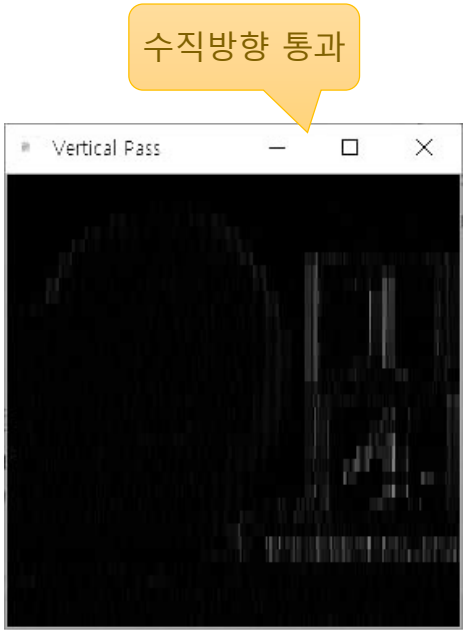
9.5 이산 코사인 변환

- 실행결과









원본영상

9. 실습 과제

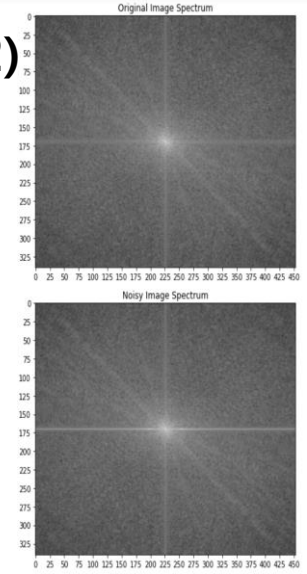
- (과제)
- **예제 9.1의 코드를 참고**
  - 1) 자신이 가진 흑백이미지에 다음과 같은 신호를 더하시오.
    (C는 아래 예시와 같은 이미지가 되도록 적당한 값을 선택하시오.)

$$I'(y,x) = I(y,x) + C \cdot \cos((1/8)\pi x)$$

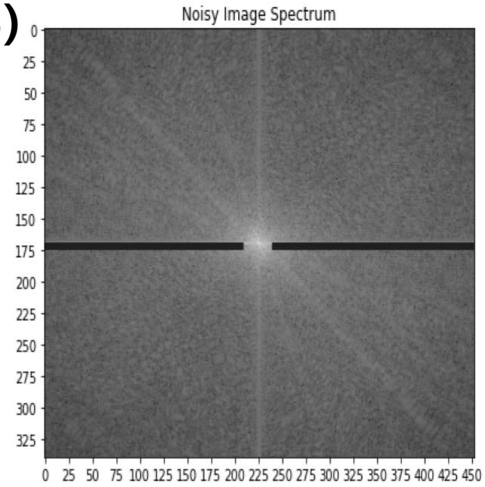
  - 2) 위 I′(y,x)에 DFT를 적용하여 주파수 도메인을 얻으시오
  - 3) 주파수에 자신만의 필터링을 적용 (아래 그림 참고)
  - 4) IDFT를 적용하여 이미지를 복원하시오.

1)

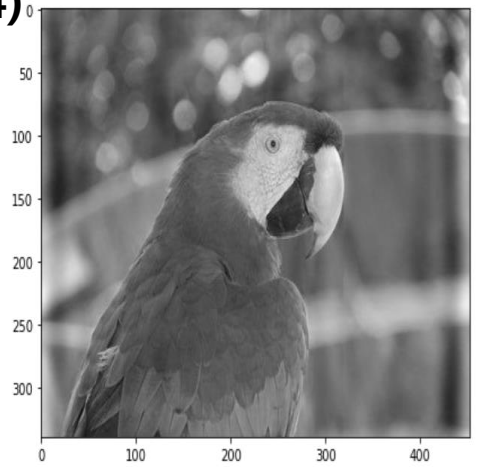
2)



3)



4)



9.1 공간 주파수의 이해 (참고사항

- **예제 9.1**
- 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)

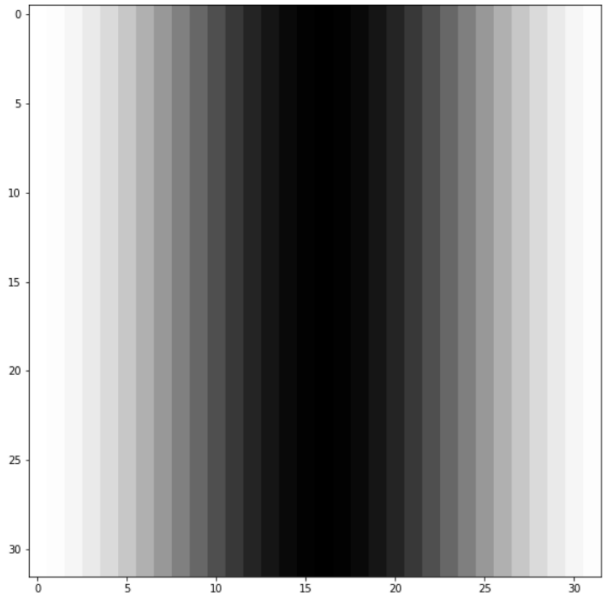
```
import matplotlib.pylab as pylab
import numpy as np
import numpy.fft as fp

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. 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