나. 주파수 영역에서의 필터링

공간영역에서의 필터링은 적절히 선택된 필터를 이용하여 콘볼루션 과정을 통해 이루어 졌다. 이에 반해 주파수 영역에서의 필터링 수행은 시간영역에서의 콘볼루션은 주파수영역에서의 곱으로 표현되는 푸리에변환 성질을 이용하여 수행하게 된다.

원본 영상을 $f(x,y)$, 필터를 $h(x,y)$, 그리고 출력 영상을 $y(x,y)$라 하면 콘볼루션은 아래 수식 (5-7)에 의해 정의된다. 그리고 이러한 공간영역에서 콘볼루션은 주파수 영역에서는 식 (5-8)과 같이 곱으로 표현되게 된다. 여기에 사용되는 필터 $H(u,v)$는 원하는 주파수 성분만을 포함하도록 직접 설계하게 된다. 그림 5-4는 주파수 영역에서의 필터링 과정을 보여주고 있다.

$$y(x,y) = f(x,y) * h(x,y) \tag{5-7}$$

$$Y(u,v) = F(u,v) * H(u,v) \tag{5-8}$$

- 저역 통과 필터(lowpass filter) : 영상의 주파수 성분 중에서 저주파(low frequency)에 해당하는 성분만을 통과시키는 필터
- 고역 통과 필터(highpass filter): 영상의 주파수 성분 중에서 고주파(high frequency)에 해당하는 성분만을 통과시키는 필터

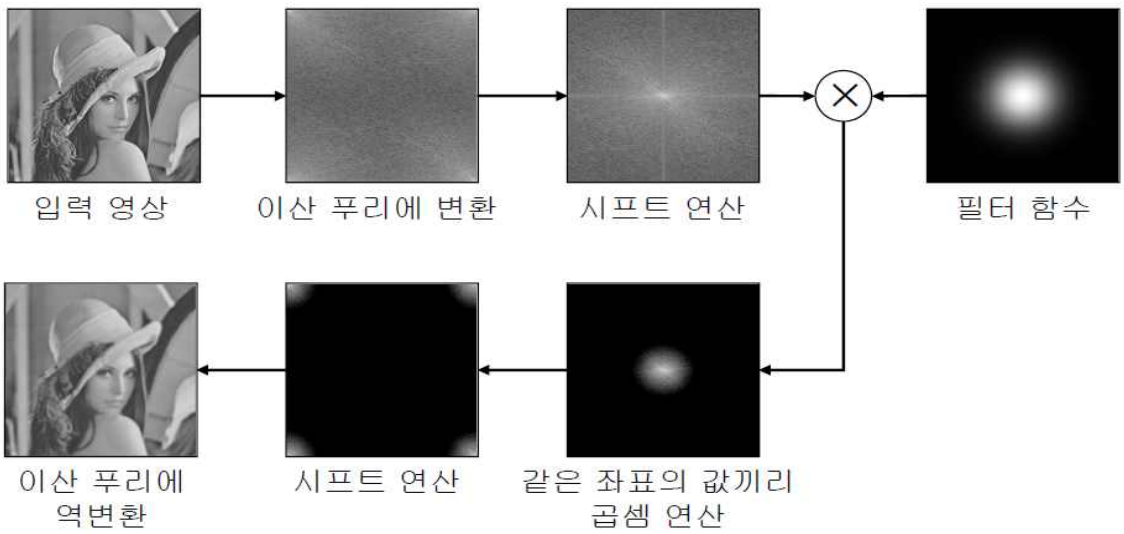


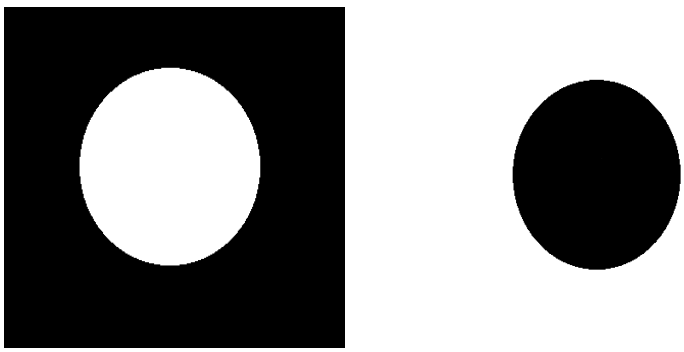
그림 5-4 주파수 영역에서의 필터링 과정

1) 이상적(Ideal) 저역 및 고역 필터링

이상적 저역 및 고역 필터는 아래 수식과 같이 정의된다. 2차원 주파수 영역에서 원하는 영역의 성분만을 이상적 필터를 이용하여 걸러내게 된다. 그림 5-5는 이상적 저역 및 고역필터의 예를 보여주고 있다.

$$H_{Low} = \begin{cases} 1, & \text{if } D(u,v) \le D_0 \\ 0, & \text{if } D(u,v) > D_0 \end{cases} \tag{5-9}$$

$$H_{High} = \begin{cases} 0, & \text{if } D(u,v) \le D_0 \\ 1, & \text{if } D(u,v) > D_0 \end{cases} \tag{5-10}$$



(a) Ideal low-pass filter (b) Ideal high-pass filter

그림 5-5 이상적 저역 및 고역 필터의 예

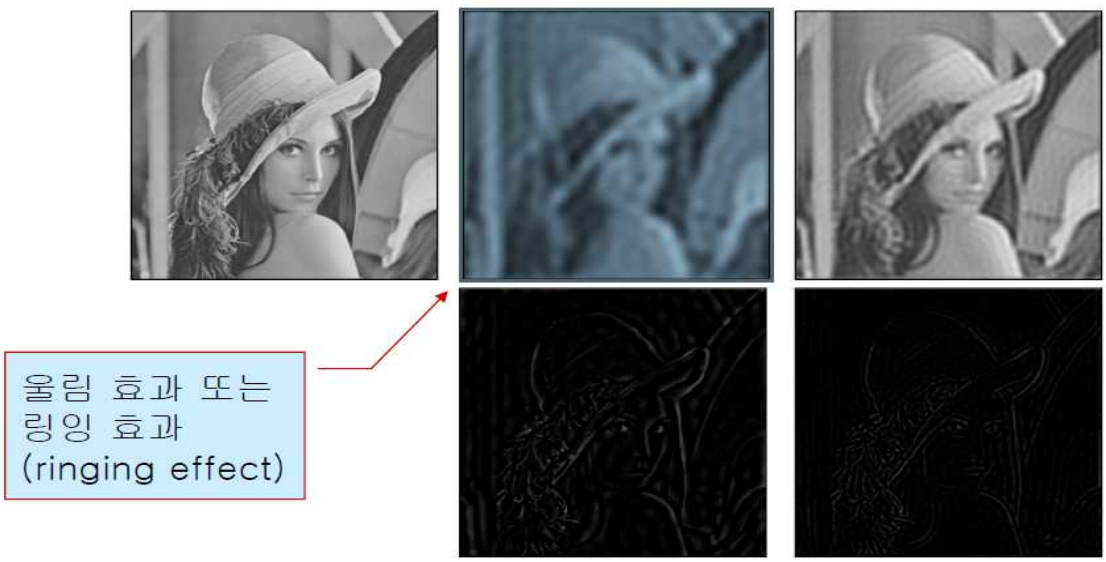


그림 5-6 이상적 저역 및 고역 필터링의 결과

< Example 5-2 > 주파수 영역에서의 Ideal 필터 구현 예

```
import numpy as np
from scipy import signal, misc
import matplotlib.pyplot as plt
from scipy import ndimage

#Make Filter
col = 256
row = 256
Low_Mask = np.zeros(shape=[row, col], dtype=np.float32)
High_Mask = np.zeros(shape=[row, col], dtype=np.float32)
BP_Mask = np.zeros(shape=[row, col], dtype=np.float32)
BS_Mask = np.zeros(shape=[row, col], dtype=np.float32)

Radius1 = 30
Radius2 = 60
cx = 128
cy = 128

for y in range(col):
    for x in range(row):
        dist = np.sqrt((cx-x)**2 + (cy-y)**2)
        if dist > Radius1:
            Low_Mask[x,y] = 0
            High_Mask[x,y] = 1
        else:
            Low_Mask[x,y] = 1
            High_Mask[x,y] = 0

for y in range(col):
    for x in range(row):
        dist = np.sqrt((cx-x)**2 + (cy-y)**2)
        if dist > Radius1 and dist < Radius2:
            BP_Mask[x,y] = 1
            BS_Mask[x,y] = 0
        else:
            BP_Mask[x,y] = 0
            BS_Mask[x,y] = 1


plt.subplot(221), plt.imshow(Low_Mask), plt.gray(), plt.axis('off'), plt.title('Ideal Lowpass')
plt.subplot(222), plt.imshow(High_Mask), plt.gray(), plt.axis('off'), plt.title('Ideal Highpass')
plt.subplot(223), plt.imshow(BP_Mask), plt.gray(), plt.axis('off'), plt.title('Ideal Bandpass')
plt.subplot(224), plt.imshow(BS_Mask), plt.gray(), plt.axis('off'), plt.title('Ideal Bandstop')
plt.show()
```

2) 주파수 영역에서의 2차원 가우시안 필터링

이상적 필터링은 ideal한 필터의 특성인 경계에서의 불연속으로 인해 복구된 영상에서 링잉 효과(ringing effect)로 인한 화질의 저하가 초래 되었다. 이러한 단점을 극복하기 위해 필터의 경계부근에서 부드럽게 떨어지는 특성을 가지는 2차원 가우시안 필터를 사용하여 필터링을 수행하게 된다. 가우시안 저역 및 고역 필터의 식은 아래와 같다.

$$H_{Low}(u,v) = e^{-D^2(u,v)/2D_0^2} \tag{5-11}$$

$$H_{High}(u,v) = 1 - e^{-D^2(u,v)/2D_0^2} \tag{5-12}$$

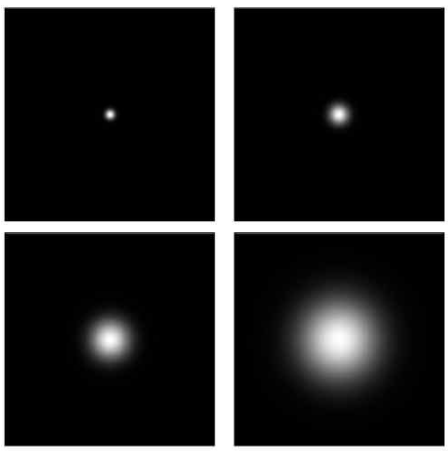


그림 5-7 2차원 가우시안 저역 필터의 예

그림 5-8 2차원 가우시안 저역 및 고역 필터링의 결과

< Example 5-3 > 주파수 영역에서의 Gaussian 필터 구현 예

```
import numpy as np
from scipy import signal, misc
import matplotlib.pyplot as plt
from scipy import ndimage

col = 256
row = 256
cx = 128
cy = 128
D0 = 20

Gaussian_L = np.zeros(shape=[col, row], dtype=np.float32)
Gaussian_H = np.zeros(shape=[col, row], dtype=np.float32)

for x in range(col):
    for y in range(row):
        D = np.sqrt((cx - x) ** 2 + (cy - y) ** 2)
        Gaussian_L[x, y] = np.exp(-D**2 / (2*D0**2) )

Gaussian_H = 1 - Gaussian_L

plt.subplot(121),plt.imshow(Gaussian_L), plt.gray(), plt.axis('off'), plt.title('Gaussian Lowpass')
plt.subplot(122),plt.imshow(Gaussian_H), plt.gray(), plt.axis('off'),plt.title('Gaussian Highpass')
plt.show()
```

3) 버터워스(Butterworth) 필터

버터워스 필터는 주파수 영역에서의 영상 필터링에 가장 많이 사용되는 필터이다. 버터워스 필터를 이용하여, low-pass, high-pass, band-pass, 그리고 band-stop 필터를 만들어 낼 수 있다. 아래 수식은 각각의 통과 대역에 해당하는 버터워스 필터 수식을 보여주고 있다.

- n*th* order butterworth lowpass filter

$$H_L(u,v) = \frac{1}{1+[r(u,v)/r_0]^{2n}}, \tag{5-13}$$

where $r(u,v) = \sqrt{u^2+v^2}$ and r_0 :radius of the filter

- n*th* order butterworth highpass filter

$$H_H(u,v) = \frac{1}{1+[r_0/r(u,v)]^{2n}}, \tag{5-14}$$

- n*th* order butterworth bandstop filter

$$H_{BS}(u,v) = \frac{1}{1+[Wr(u,v)/(r^2(u,v)-r_0^2)]^{2n}}, \tag{5-15}$$

where W는 대역폭

- n*th* order butterworth bandpass filter

$$H_{BP}(u,v) = 1 - H_{BS}(u,v), \tag{5-16}$$

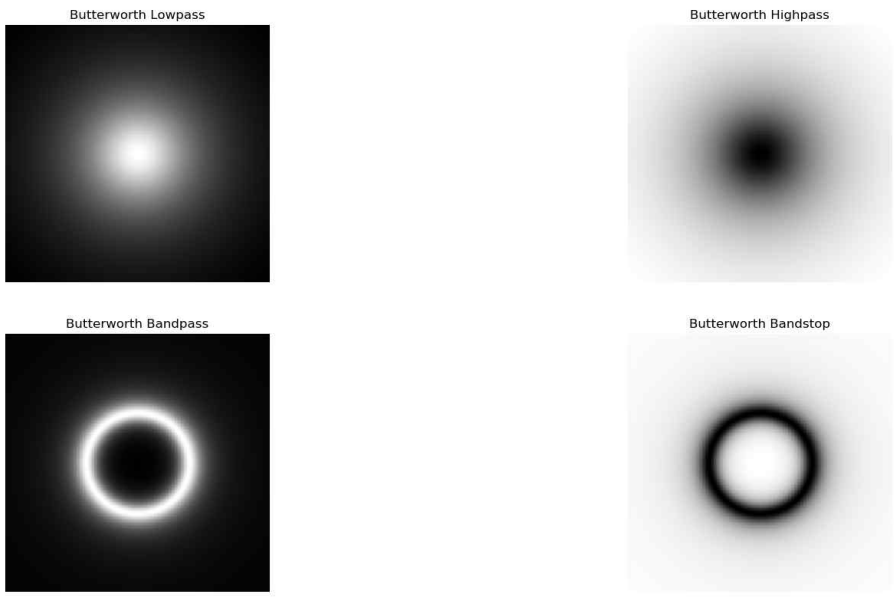


그림 5-9 2차원 버터워스 필터의 예

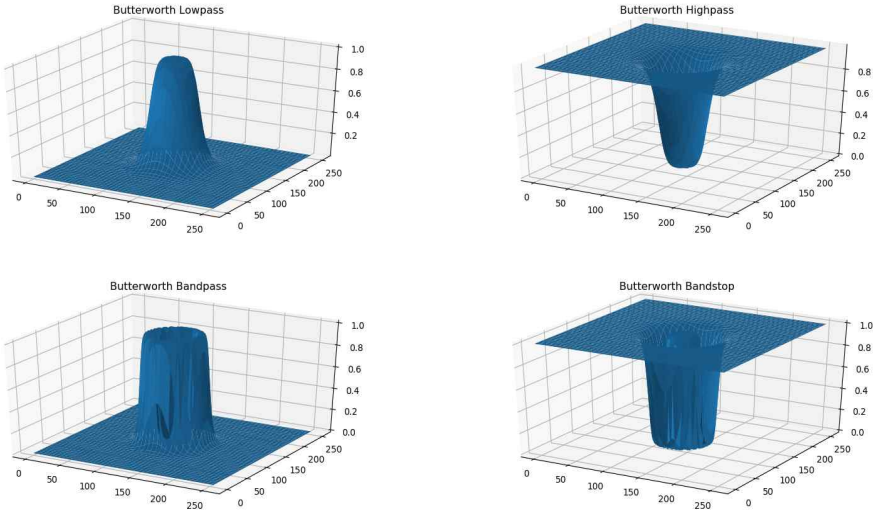


그림 5-10 2차원 버터워스 필터의 3D plot

< Example 5-3 > 주파수 영역에서의 Butterworth 필터 구현 예

```
import numpy as np
from scipy import signal, misc
import matplotlib.pyplot as plt
from scipy import ndimage

col = 256
row = 256
cx = 128
cy = 128
D0 = 50
N = 1
W = 20
BW_L = np.zeros(shape=[col,row], dtype=np.float32)
BW_H = np.zeros(shape=[col,row], dtype=np.float32)
BW_BP = np.zeros(shape=[col,row], dtype=np.float32)
BW_BS = np.zeros(shape=[col,row], dtype=np.float32)

for x in range(col):
    for y in range(row):
        D = np.sqrt((cx-x)**2 + (cy-y)**2)
        s = (W * D) / (D ** 2 - D0 ** 2)
        BW_L[x, y] = 1 / (1 + pow(D/D0, 2*N ))
        BW_H[x, y] = 1 / (1 + pow(D0/D, 2*N ))
        BW_BS[x, y] = 1 / (1 + pow(s, 2*N ))

BW_BP = 1 - BW_BS

plt.subplot(221), plt.imshow(BW_L), plt.gray(), plt.axis('off'), plt.title('Butterworth Lowpass')
plt.subplot(222), plt.imshow(BW_H), plt.gray(), plt.axis('off'), plt.title('Butterworth Highpass')
plt.subplot(223), plt.imshow(BW_BP), plt.gray(), plt.axis('off'), plt.title('Butterworth Bandpass')
plt.subplot(224), plt.imshow(BW_BS), plt.gray(), plt.axis('off'), plt.title('Butterworth Bandstop')
plt.show()
```