마. Sharpening

블러링과 반대의 효과를 보이는것은ㄹ 샤프닝(sharpening)이라고 한다. 즉, 블러링은 저주파 통과라면 샤프닝은 반대인 고주파 통과 효과이다. 앞에서도 언급하였듯이, 영상에서 변화가 거의 없는 부분이 배경이라며, 변화가 급격히 발생하는 부분은 경계인 에지(edge)에 해당한다. 영상에서 저주파 영역에 해당하는 부분은 제거하고, 고주파에 해당하는 부분만 통과시키는 역할을 수행한다.

1) Unsharp Mask 필터

영상에서 고주파 성분은 저주파 필터링을 이용해서 얻을 수도 있다. 개념적으로 저주파 필터링을 수행하면 고주파 성분이 제거된 영상을 얻게 된다. 이 결과를 원본 영상에서 빼주게 되면 고주파 성분을 얻을 수 있게 된다. 아래 식 (4-5)와 같이 표현 할 수 있다.

$$g(x,y) = f(x,y) - \overline{f}(x,y) \tag{4-5}$$

$f(x,y)$: 원본 영상

$\overline{f}(x,y)$: 저주파 통과된 영상

$g(x,y)$: 고주파 성분이 남은 영상

Unsharp Mask는 이러한 결과를 이용해서, 영상을 선명하게 만드는 효과이다. 즉, 원본영상에서 저주파 필터을 통해 고주파 성분만 남기고, 이를 다시 원본 영상에 더해주게 되면, 고주파 성분이 강조된 선명한 영상을 얻을 수 있게 된다. 이를 그림으로 표현한 것이 그림 4-10이고 수식으로는 4-6으로 표현 할 수 있다.

$$\begin{aligned} h(x,y) &= f(x,y) + g(x,y) \\ &= f(x,y) + [f(x,y) - \overline{f}(x,y)] \end{aligned} \tag{4-6}$$

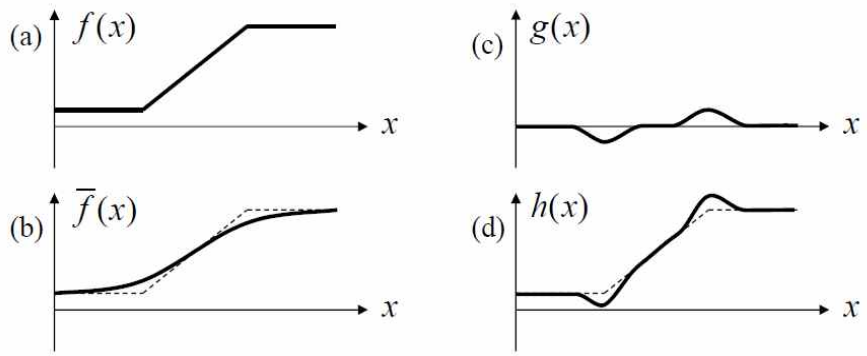


그림 4-10 Unsharp Mask 필터의 개념

그림 4-11 Unsharp Mask 필터링 결과의 예

2) Laplacian 필터

일반적으로 영상에서 고주파 성분은 이웃 픽셀값과 큰 차이를 보이며 급격히 값이 변하는 부분이다. 이웃 픽셀과 값 차이가 크다는 것은 변화가 심하다는 말로 표현 할 수 있고 이를 수학적 개념으로 표현하면 미분에 해당하게 된다.

예를 들어 x방향으로 일차 미분(이웃 픽셀과의 값의 차이 변화를 의미)은 다음과 같이 식 (4-7)로 표현되고, x방향으로 이차 미분은 (4-8)로 표현할 수 있다.

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{4-7}$$

$$\frac{\partial^2 f}{\partial x^2} = [f(x+1) - f(x)] - [f(x) - f(x-1)] \tag{4-8}$$

$$= f(x+1) + f(x-1) - 2f(x)$$

1	-2	1
$f(x-1)$	$f(x)$	$f(x+1)$

그림 4-12 x방향으로 이차 미분필터의 값

이와 같은 성질을 이용하여 x와 y방향으로 이차 미분은 수행하게 되면 x와 y방향 이차 미분 필터인 Laplacian 필터를 구할 수 있게 된다. 수식으로는 (4-9)와 같이 표현된다. 그림 4-13은 x와 y방향 이차 미분 필터인 Laplacian 필터 값과 대각선 방향까지 포함한 Laplacian 필터 값을 보여주고 있다.

$$\Delta^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{4-9}$$

$$= [f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)] - 4f(x,y)$$

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

(a) x와 y방향 이차 미분 필터 (b) 대각선 방향까지 포함한 필터 값

그림 4-13 Laplacian 이차 미분필터의 값

(a)원본영상 (b)4방향 (c)8방향

그림 4-14 Laplacian 필터링 결과의 예

3) Unsharp mask 필터의 수식 변경

Unsharp mask 필터링 과정을 Laplacian필터를 이용해 수정하여 표현할 수 있다. Unsharp mask 필터링은 원본 영상에 고주파 성분 신호를 더해주어 구하게 되는데, 고주파 성분을 Laplacian 필터로 대체하여 표현 할 수 있다.

$$h(x,y) = f(x,y) + \Delta^2 f(x,y) \tag{4-10}$$

$$= f(x,y) - [f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)] - 4f(x,y)$$

$$= 5f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1)$$

0	-1	0
-1	5	-1
0	-1	0

-1	-1	-1
-1	9	-1
-1	-1	-1

그림 4-14 Laplacian 이차 미분필터를 이용한 Unsharp mask 필터의 값

4) High-boost 필터

Unsharp mask 필터링에서 원본 영상의 화질을 좀 더 보존하고자 한다면, 아래와 같이 수식을 변경하여 표현할 수 있고, 이를 High-boost 필터링이라 한다. 수식에서 α의 값에 따라 원본 영상의 보존 정도를 결정할 수 있다. $\alpha=1$이면, Unsharp mask와 동일하게 된다.

$$h(x,y)=\alpha f(x,y)+\Delta^2 f(x,y) \tag{4-11}$$

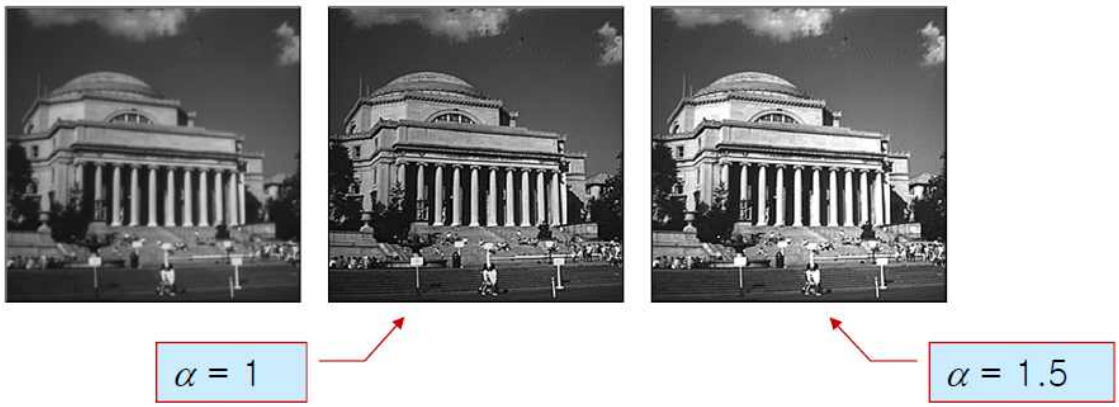


그림 4-15 High-boost 필터링의 예

5) 잡음생성 및 잡음 제거

영상에서 고려할 수 있는 대표적인 잡음은 균일분포(uniform distribution), 정규분포(normal distribution), 그리고 소금-후추(salt-and-pepper) 잡음을 들 수 있다.

- 균일분포(uniform distribution)는 정해진 범위 내에서(예를 들어 $0\sim1$) 유사 임의수를 생성하게 된다.

- 정규분포(normal distribution)는 평균이 0이고 표준편차가 1인 가우시안 분포 잡음을 생성하게 된다.
- 소금-후추(salt-and-pepper) 잡음은 입력 영상의 임의의 좌표 픽셀 값을 0또는 255로 설정하는 잡음을 의미한다.

< Example 4-1 > Noise Generation

```
def noisy(noise_typ,image):
    if noise_typ == "gauss":
        row, col = image.shape
        mean = 0
        var = 0.5
        sigma = var**0.5
        gauss = np.random.normal(mean,sigma,(row,col))
        gauss = gauss.reshape(row,col)
        noisy = image + 10.0*gauss
        return noisy

    elif noise_typ == "s&p":
        row,col = image.shape
        s_vs_p = 0.5
        amount = 0.05
        out = np.copy(image)
        # Salt mode
        num_salt = np.ceil(amount * image.size * s_vs_p)
        coords = [np.random.randint(0, i - 1, int(num_salt)) for i in image.shape]
        out[coords] = 255

        # Pepper mode
        num_pepper = np.ceil(amount* image.size * (1. - s_vs_p))
        coords = [np.random.randint(0, i - 1, int(num_pepper)) for i in image.shape]
        out[coords] = 0
        return out

    elif noise_typ == "poisson":
        vals = len(np.unique(image))
        vals = 2 ** np.ceil(np.log2(vals))
        noisy = np.random.poisson(image * 10.0*vals) / float(vals)
        return noisy

    elif noise_typ =="speckle":
        row,col = image.shape
        gauss = np.random.randn(row,col)
        gauss = gauss.reshape(row,col)
        noisy = image + image * 0.1*gauss
        return noisy

# Image Read
lena = misc.imread('lena_256.bmp')
col, row = lena.shape
IM = noisy("s&p", lena)
```

가우시안 잡음의 경우, 블러링 필터를 이용해서 잡음을 어느 정도 제거 할 수 있다. 하지만, 영상이 전체적으로 블러링되어 화질이 저하되는 것은 막을 수 없다. 소금-후추 잡음은 미디언(median)필터를 이용하여 제거하면 잡음을 어느 정도 효과적으로 제거할 수 있다.

미디언 필터링은 입력 영상의 (x,y) 좌표 주변 픽셀들의 값들을 오름 또는 내림 차순으로 정렬하여 그 중앙에 있는 픽셀 값을 사용한다. 하지만, 이 방법 역시 전체적으로 화질이 저하되는 현상이 발생한다.

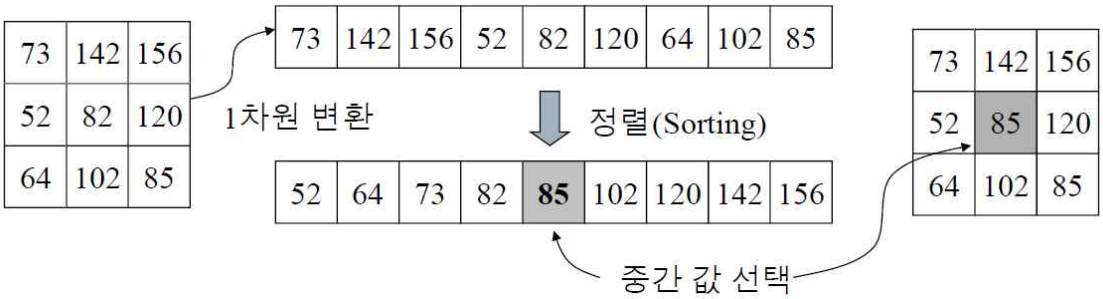


그림 4-16 미디언(median) 필터링의 예

< 실습 4-4 >

1. < Example 4-1 > Noise Generation 프로그램을 이용하여 잡음영상을 만드시오.
   (a) 가우시안 잡음영상을 만드시오. var = 0.5, 1.0, 2.0, 3.0, 4.0 일 때의 영상을 각각 만들어 화면에 원본영상과 함께 출력하시오.
   (b) (a)의 잡음영상을 평균필터링을 수행하여 잡음을 제거하고 그 결과를 화면에 출력하시오.
   (c) 소금-후추 잡음영상을 만드시오. amount = 0.05, 0.1, 0.2, 0.3, 0.4일 때의 영상을 각각 만들어 화면에 원본영상과 함께 출력하시오.
   (d) (c)의 잡음영상을 미디언 필터링을 수행하여 잡음을 제거하고 그 결과를 화면에 출력하시오.

5. 주파수 영역 필터링

디지털 영상신호를 주파수 영역으로 이동하여 처리하기 위해서는 2차원 푸리에 변환이 사용된다. 기본적으로 차원이 하나에서 두 개로 확대되는 것 말고는 기본적인 푸리에 변환과 동일하다.

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)} \tag{5-1}$$

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)} \tag{5-2}$$

가. 영상의 스펙트럼

2차원 영상의 푸리에 변환결과는 1차원 푸리에 변환과 마찬가지로 복소수로 표현된다. 따라서 아래 수식(5-3(과 같이 크기성분과 위상성분으로 분리하여 표현한다.

$$F(u,v) = |F(u,v)|e^{j\phi(u,v)} \tag{5-3}$$

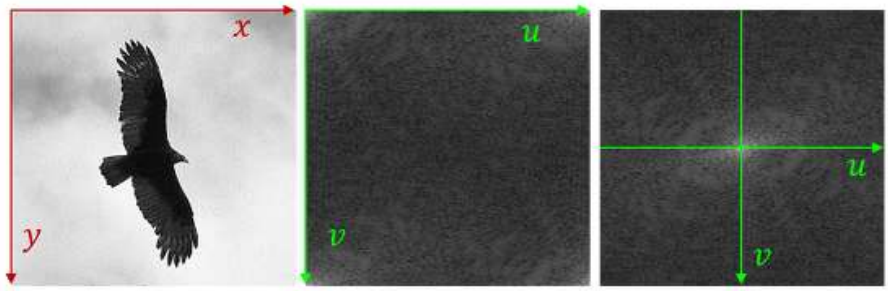
$$\text{where, } |F(u,v)| = \left[R^2(u,v)+I^2(u,v)\right]^{1/2}\text{: Magnitude Spectrum} \tag{5-4}$$

$$\phi(u,v) = \tan^{-1}\left(\frac{I(u,v)}{R(u,v}\right) \quad \text{: Phase Spectrum} \tag{5-5}$$

푸리에 스펙트럼(Fourier spectrum)을 이미지로 시각화하는 데에는 2가지 문제점이 있다. 먼저, 푸리에 스펙트럼은 저주파 영역은 매우 큰 값을 갖는 반면에 대부분의 다른 영역은 0에 가까운 값을 갖는다. 따라서 푸리에 스펙트럼을 그대로 이미지로 시각화하면 검은 바탕 위에 흰점 하나만 존재하는 형태가 된다. 이러한 문제를 해결하기 위해서 스펙트럼을 이미지로 표현할 때에는 수식 (5-6)과 같이 스펙트럼에 log를 취하는 것이 일반적이다. 수식(5-6)에서 1을 더해주는 이유는 로그 함수의 입력 값이 0이 되는 경우를 방지하기 위함이다.

$$D(u,v) = c\log[|F(u,v)|+1] \tag{5-6}$$

다음으로, 원래의 스펙트럼 이미지는 그림 5-1(b)의 첫 번째처럼 모서리로 갈수록 값이 높아지기 때문에 스펙트럼의 형태를 파악하기 힘들다. 따라서 이러한 문제를 해결하기 위해 그림 그림 5-1(b)의 두 번째처럼처럼 원점이 중심(center)에 오도록 스펙트럼의 위치를 이동시킨(shift) 형태의 이미지를 사용하는 것이 일반적이다. 그림 5-2는 영상의 푸리에 스펙트럼을 화면에 출력하는 방법을 보여주고 있다. 표5-1은 2차원 푸리에 변환의 몇 가지 성질을 요약한 것이다.



(a) 원본영상:$f(x,y)$ (b) 크기 스펙트럼:$F(u,v)$

그림 5-1 푸리에 스펙트럼과 좌표계

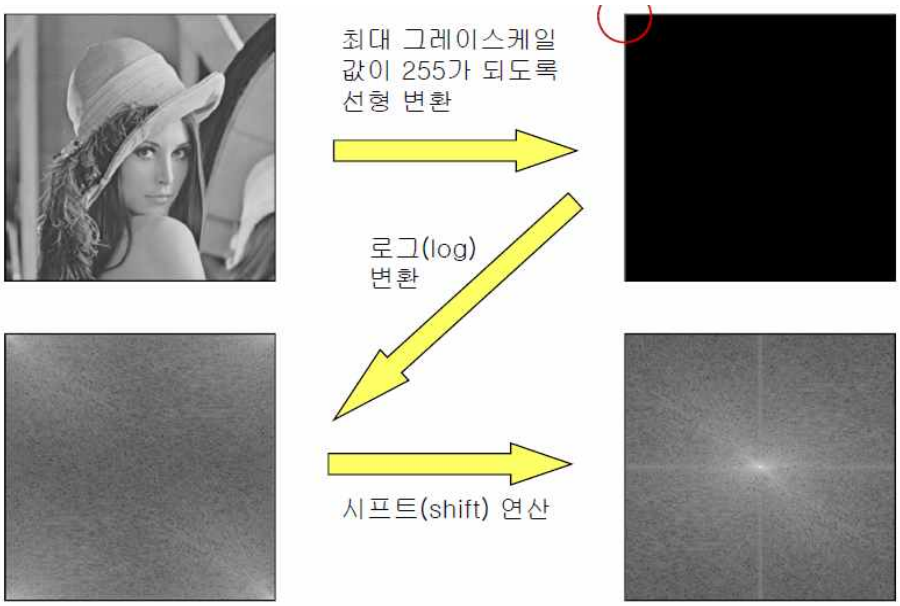


그림 5-2 영상의 푸리에 스펙트럼을 화면에 출력하는 방법

표 5-1, 2차원 푸리에 변환의 성질

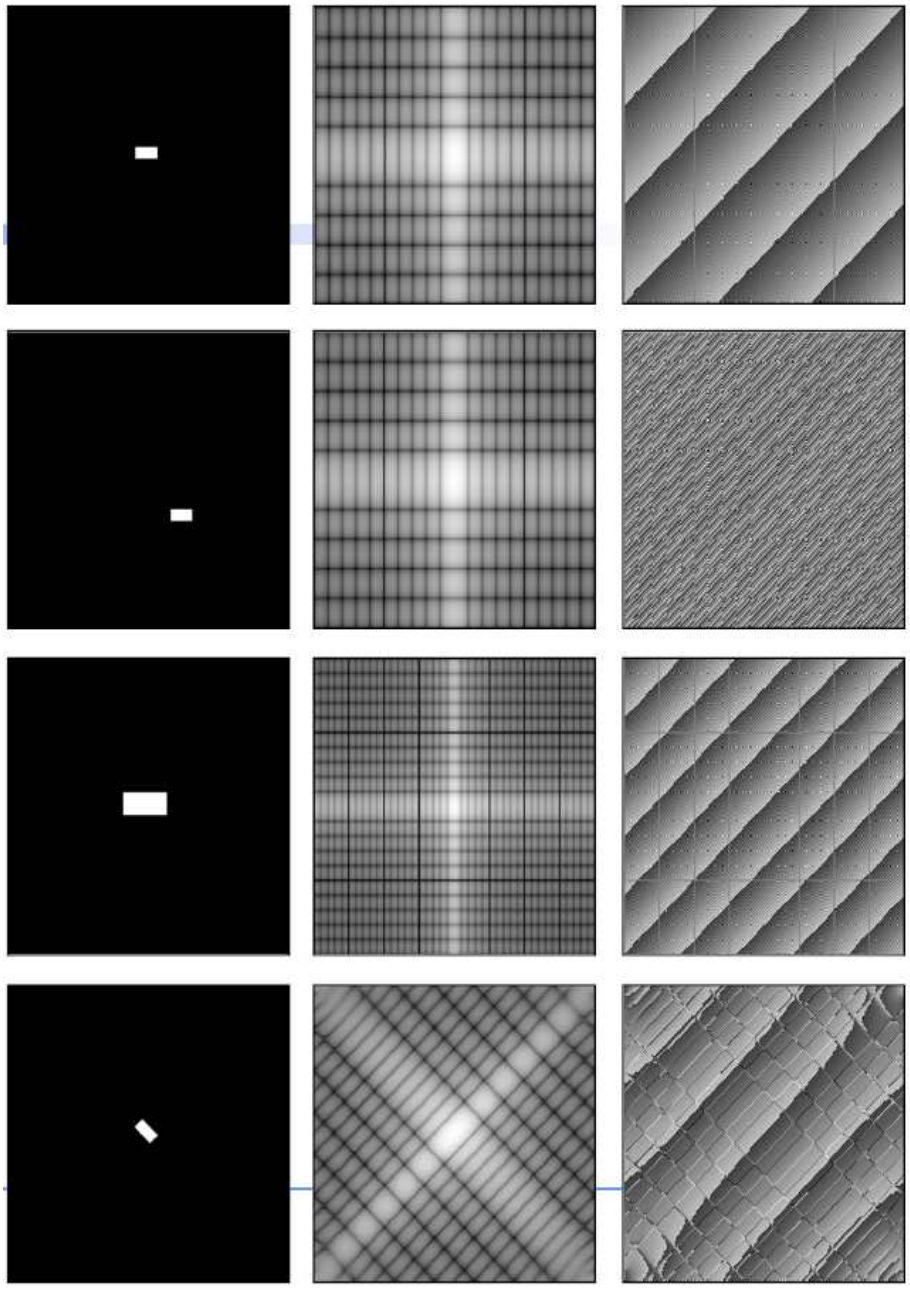
선형성	$f(x,y)=af_1(x,y)+bf_2(x,y)$ $\Leftrightarrow F(u,v)=aF_1(u,v)+bF_2(u,v)$
이동변환	$f(x-x_0,y-y_0)\Leftrightarrow F(u,v)e^{-j2\pi(ux_0/M+vy_0/N)}$
주파수 이동변환	$f(x,y)e^{-j2\pi(ux_0/M+vy_0/N)}\Leftrightarrow F(u-u_0,v-v_0)$
이동변환 이용 shift	$f(x,y)(-1)^{x+y}\Leftrightarrow F(u-M/2,v-N/2)$
크기변환	$f(ax,by)\Leftrightarrow\frac{1}{\lvert ab\rvert}F(u/a,v/b)$
회전변환	$f(r,\theta+\theta_0)\Leftrightarrow F(\omega,\phi+\theta_0)$

(a)원본영상

(b) 이동변환

(c)크기변환

(d) 회전변환



< 영산신호 > <크기 스펙트럼> <위상 스펙트럼>

그림 5-3 기하학적 변환에 따른 푸리에 변환 결과

< Example 5-1 > 주파수 영역에서의 스펙트럼 분석 및 복구

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

lena = misc.imread('lena_256.bmp')

# Apply FFT
F = np.fft.fft2(lena)
Mag = np.abs(F)
Pha = np.angle(F)

fftshiftMag = np.fft.fftshift(Mag)
RefftshiftMag = np.fft.fftshift(fftshiftMag)

MakeComplexImage = RefftshiftMag*np.exp(1j*Pha)
Image_Restore = np.real(np.fft.ifft2(MakeComplexImage))

plt.subplot(231),
plt.imshow(lena), plt.gray(), plt.axis('off'), plt.title('Original Image')
plt.subplot(232),
plt.imshow(np.log10(Mag+1)),plt.gray(), plt.axis('off'), plt.title('Magnitude')
plt.subplot(233),
plt.imshow(Pha), plt.gray(), plt.axis('off'), plt.title('Phase')
plt.subplot(234),
plt.imshow(np.log10(fftshiftMag+1)), plt.gray(),plt.axis('off'), plt.title('Shifted Magnitude')
plt.subplot(235),
plt.imshow(np.log10(RefftshiftMag+1)),plt.gray(),plt.axis('off'),plt.title('Re-Shifted Magnitude')
plt.subplot(236),
plt.imshow(Image_Restore),plt.gray(),plt.axis('off'), plt.title('Restored Image')
plt.show()
```