

Image Processing

실습 4.

2021. 03. 28.

실습 수업 소개

- **과목 홈페이지**
 - 충남대학교 사이버 캠퍼스 (<http://e-learn.cnu.ac.kr>)
- **TA 연락처**
 - 신준호
 - wnsgh578@naver.com
- **튜터 연락처**
 - 19 한승오
 - sh.h4ns@gmail.com
- **실습 중 질문사항**
 - 실시간 수업중 질문 or 메일을 통한 질문
 - 메일로 질문할 때 [IP] 를 제목에 붙여주세요

실습 수업 소개

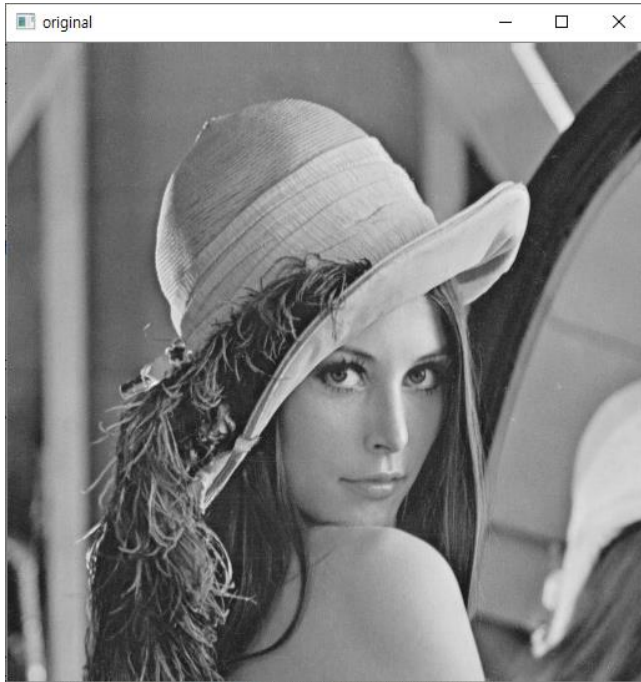
- 실습 출석
 - 사이버캠퍼스를 통해 Zoom 출석
 - Zoom 퇴장 전 채팅 기록[학번 이름] 남기고 퇴장
 - 위 두 기록을 통해 출석 체크 진행 예정

목 차

- 실습
 - Image Filtering
 - Gaussian Filter
 - Separability of Gaussian Filter
- 과제
 - Average Filtering
 - Sharpening Filtering
 - Gaussian Filtering

image filtering

- **average filter(평균값 필터)**
 - image를 부드럽게 해주는 효과
 - 잡음을 제거하는데 사용됨



original



3x3 average filter

image filtering

- average filter(평균값 필터) 실습1

`filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]])`

- src : 이미지
- ddepth : 이미지 깊이(자료형 크기). -1이면 입력과 동일
- kernel : 커널 행렬

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

mask

```
import cv2
import numpy as np

def my_average_filter_3x3(src):
    mask = np.array([[1/9, 1/9, 1/9],
                     [1/9, 1/9, 1/9],
                     [1/9, 1/9, 1/9]])
    dst = cv2.filter2D(src, -1, mask)

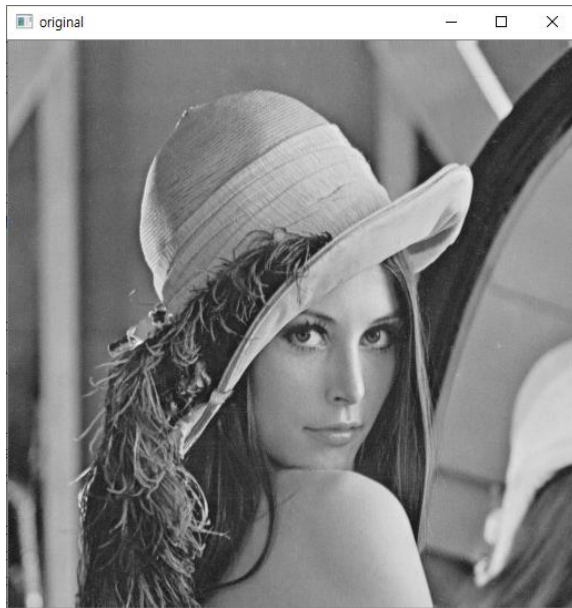
    return dst

if __name__ == '__main__':
    src = cv2.imread("Lena.png", cv2.IMREAD_GRAYSCALE)
    dst = my_average_filter_3x3(src)

    cv2.imshow('original', src)
    cv2.imshow('average_filter', dst)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

image filtering

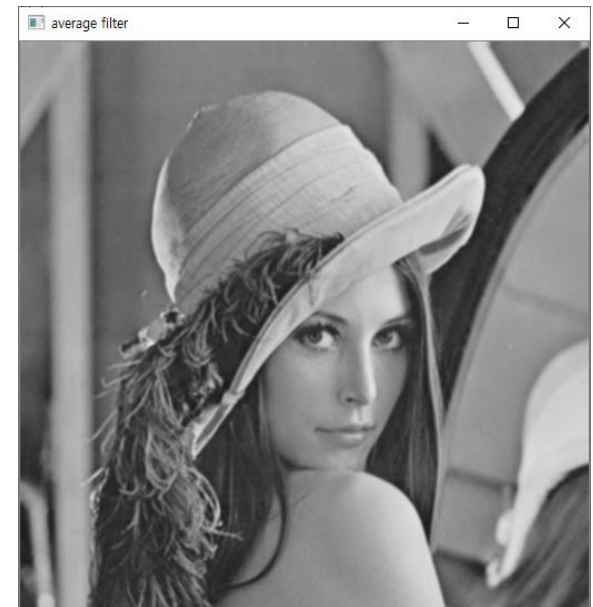
- average filter(평균값 필터) 실습1



original

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

mask



average filter

image filtering

- average filter(평균값 필터) 실습2

1/12	1/12	1/12
1/12	1/12	1/12
1/12	1/12	1/12

mask

```
import cv2
import numpy as np

def my_average_filter_3x3(src):
    mask = np.array([[1/12, 1/12, 1/12],
                     [1/12, 1/12, 1/12],
                     [1/12, 1/12, 1/12]])
    dst = cv2.filter2D(src, -1, mask)

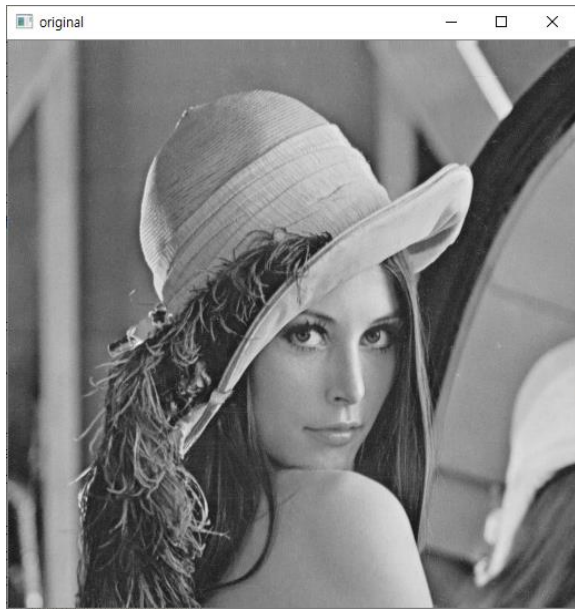
    return dst

if __name__ == '__main__':
    src = cv2.imread("Lena.png", cv2.IMREAD_GRAYSCALE)
    dst = my_average_filter_3x3(src)

    cv2.imshow('original', src)
    cv2.imshow('average_filter', dst)
    cv2.waitKey()
    cv2.destroyAllWindows()
```


image filtering

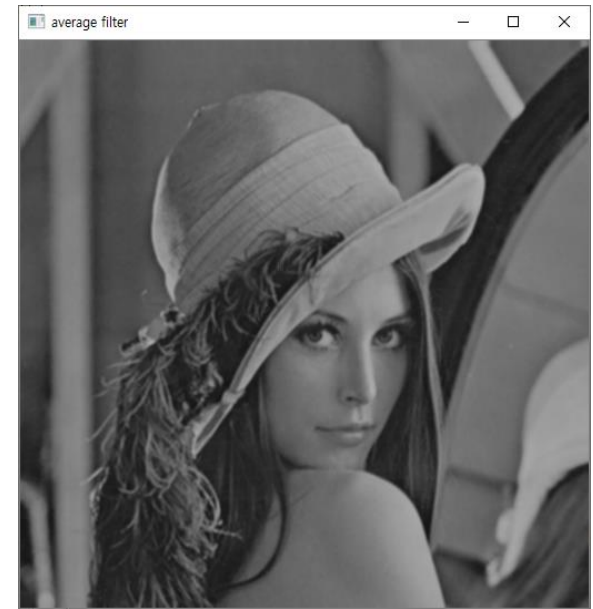
- average filter(평균값 필터) 실습2



original

1/12	1/12	1/12
1/12	1/12	1/12
1/12	1/12	1/12

mask



average filter

image filtering

- average filter(평균값 필터) 실습3

1/4	1/4	1/4
1/4	1/4	1/4
1/4	1/4	1/4

mask

```
import cv2
import numpy as np

def my_average_filter_3x3(src):
    mask = np.array([[1/4, 1/4, 1/4],
                     [1/4, 1/4, 1/4],
                     [1/4, 1/4, 1/4]])
    dst = cv2.filter2D(src, -1, mask)

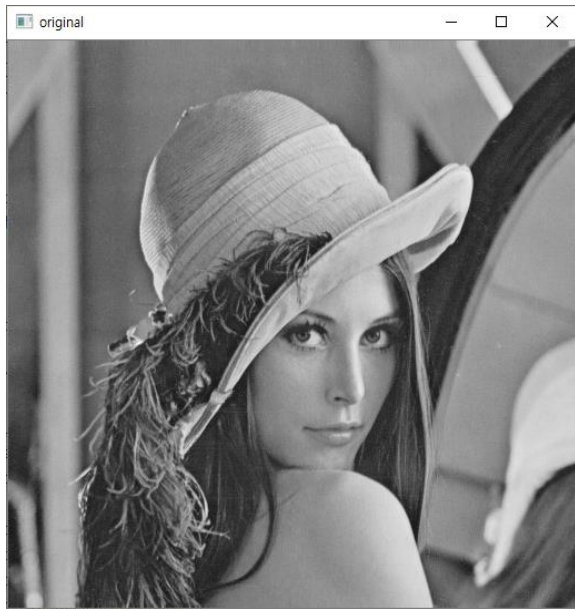
    return dst

if __name__ == '__main__':
    src = cv2.imread("Lena.png", cv2.IMREAD_GRAYSCALE)
    dst = my_average_filter_3x3(src)

    cv2.imshow('original', src)
    cv2.imshow('average_filter', dst)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

image filtering

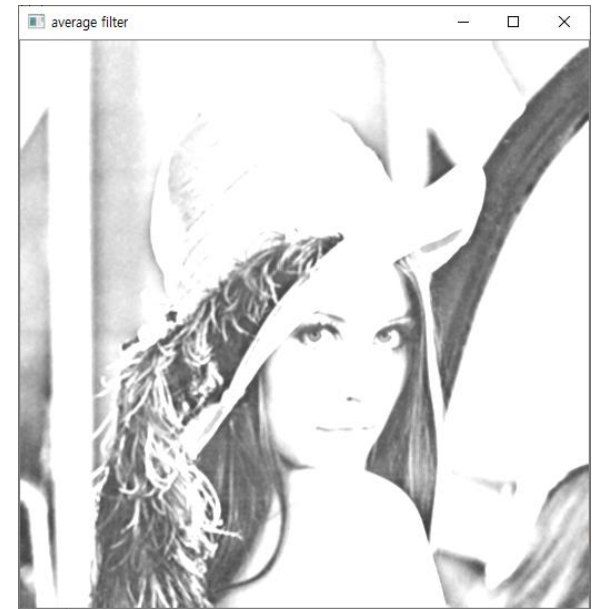
- average filter(평균값 필터) 실습3



original

$1/4$	$1/4$	$1/4$
$1/4$	$1/4$	$1/4$
$1/4$	$1/4$	$1/4$

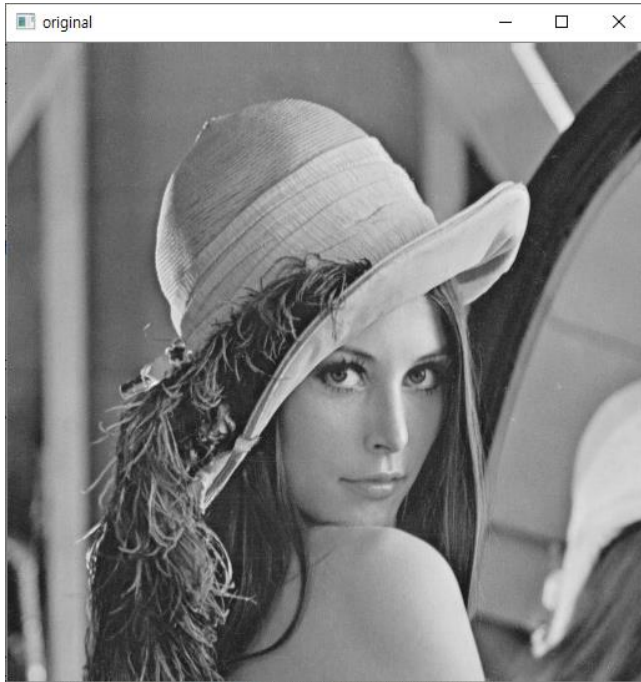
mask



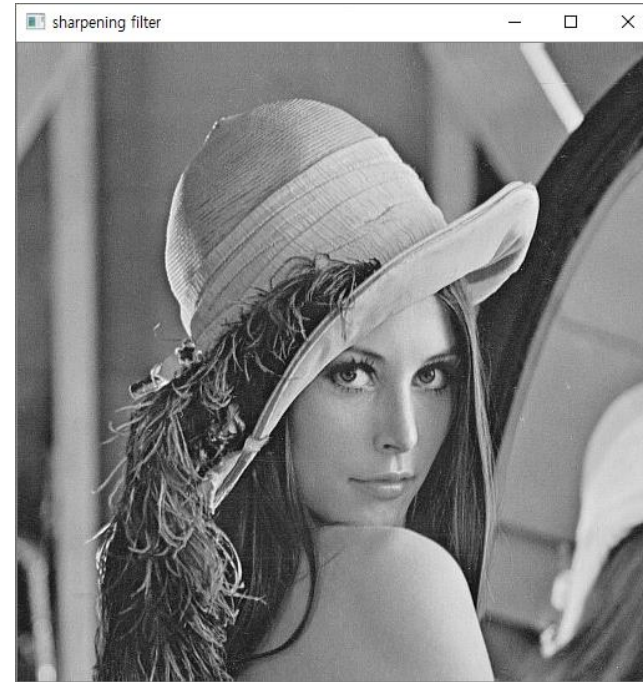
average filter

image filtering

- **sharpening filter**
 - image를 선명하게 해주는 효과



original



3x3 sharpening filter

image filtering

- sharpening filter 실습1

0	0	0		1/9	1/9	1/9
0	2	0		1/9	1/9	1/9
0	0	0		1/9	1/9	1/9

-1/9	-1/9	-1/9
-1/9	17/9	-1/9
-1/9	-1/9	-1/9

mask

```
import cv2
import numpy as np

def my_sharpening_filter_3x3(src):
    mask = np.array([[ -1/9, -1/9, -1/9],
                     [ -1/9, 17/9, -1/9],
                     [ -1/9, -1/9, -1/9]])
    dst = cv2.filter2D(src, -1, mask)

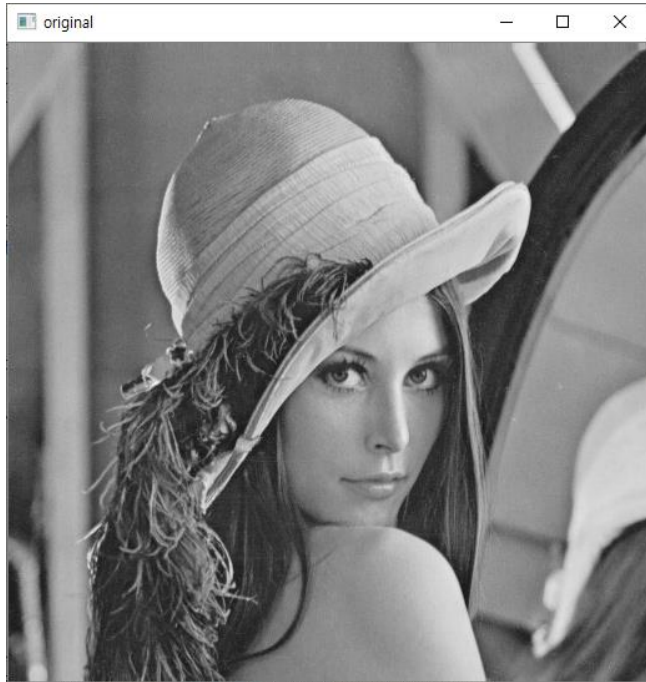
    return dst

if __name__ == '__main__':
    src = cv2.imread("Lena.png", cv2.IMREAD_GRAYSCALE)
    dst = my_sharpening_filter_3x3(src)

    cv2.imshow('original', src)
    cv2.imshow('sharpening filter', dst)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

image filtering

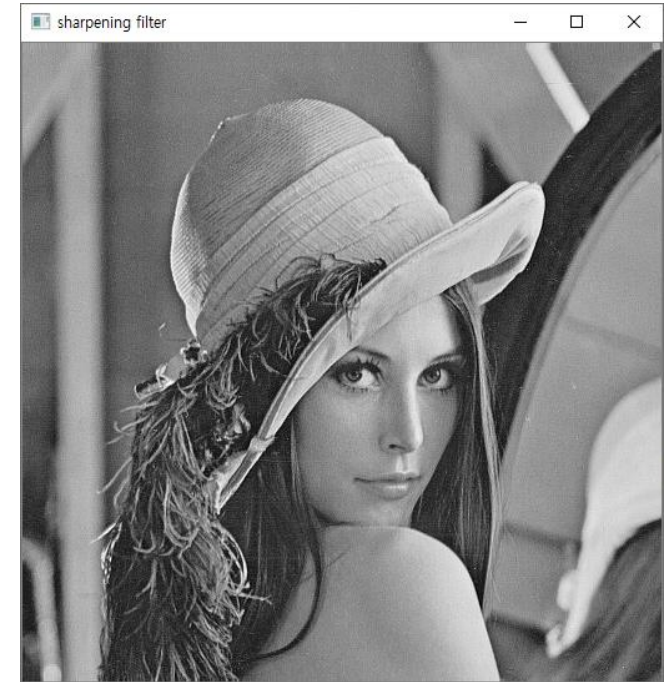
- sharpening filter 실습1



original

$-1/9$	$-1/9$	$-1/9$
$-1/9$	$17/9$	$-1/9$
$-1/9$	$-1/9$	$-1/9$

mask




sharpening filter


image filtering

- sharpening filter 실습2

0	0	0
0	3	0
0	0	0



1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



-1/9	-1/9	-1/9
-1/9	26/9	-1/9
-1/9	-1/9	-1/9

mask

```
import cv2
import numpy as np

def my_sharpening_filter_3x3(src):
    mask = np.array([[ -1/9, -1/9, -1/9],
                     [ -1/9, 26/9, -1/9],
                     [ -1/9, -1/9, -1/9]])
    dst = cv2.filter2D(src, -1, mask)

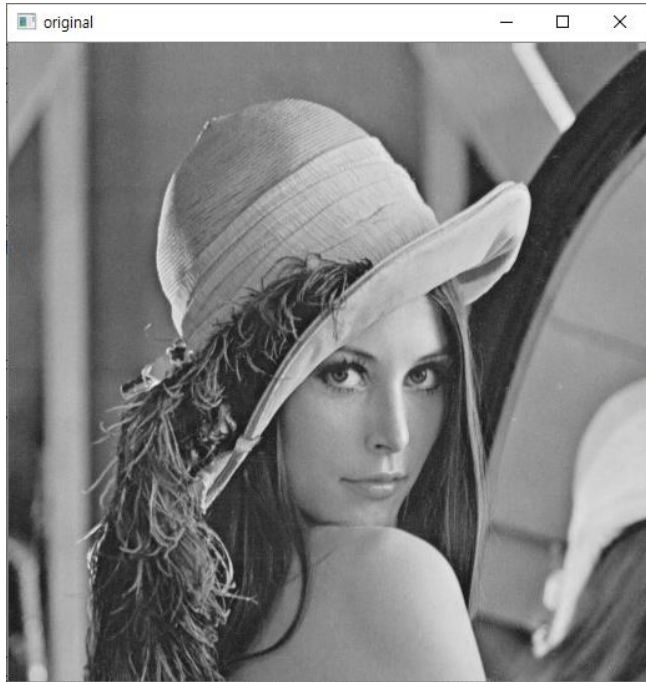
    return dst

if __name__ == '__main__':
    src = cv2.imread("Lena.png", cv2.IMREAD_GRAYSCALE)
    dst = my_sharpening_filter_3x3(src)

    cv2.imshow('original', src)
    cv2.imshow('sharpening filter', dst)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

image filtering

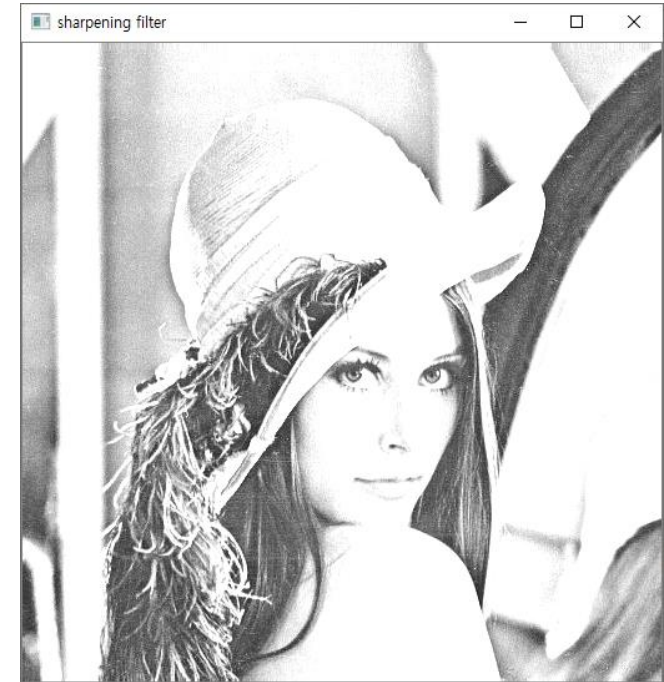
- sharpening filter 실습2



original

$-1/9$	$-1/9$	$-1/9$
$-1/9$	$26/9$	$-1/9$
$-1/9$	$-1/9$	$-1/9$

mask



sharpening filter

padding

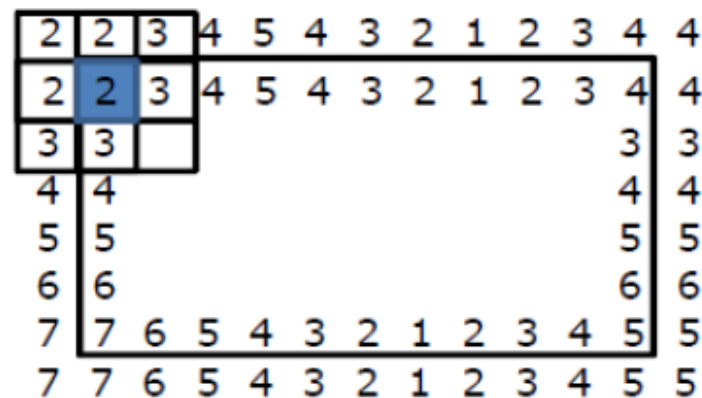
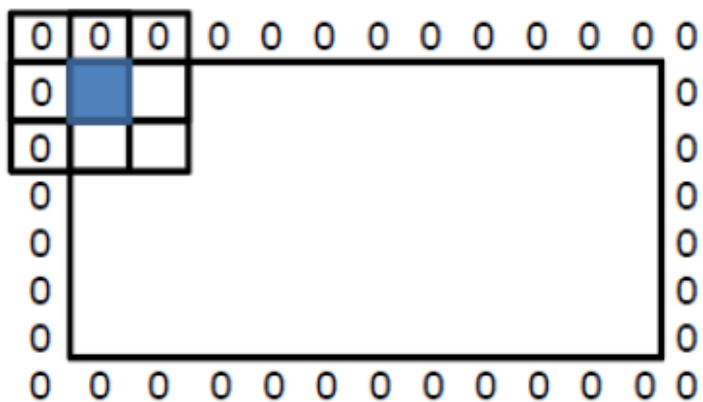
- 실제 이미지에는 없는 가장자리 부분을 채우는 역할을 함

- **Zero padding**

- 단순히 0으로 채움
- 실습에서 주로 사용

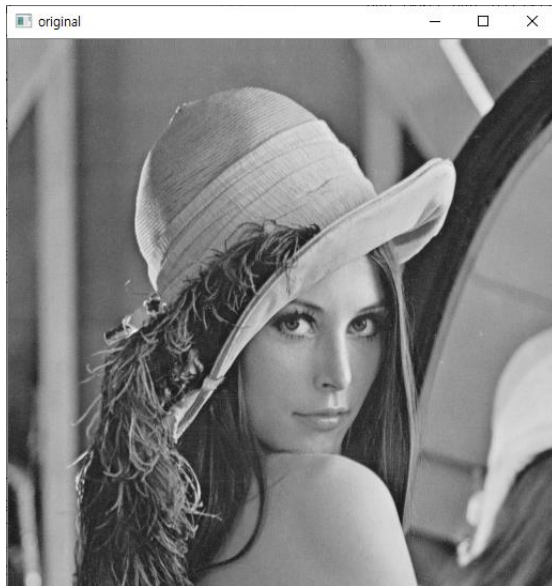
- **Repetition padding**

- 가장자리의 값을 복사해 옴



padding

- **zero padding**
 - 단순히 0으로 채움



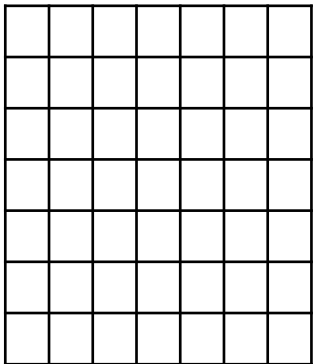
original



zero padding

padding

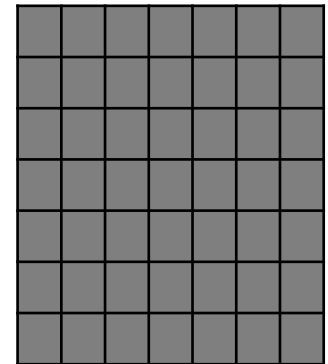
- padding을 안한다면?



7x7 image



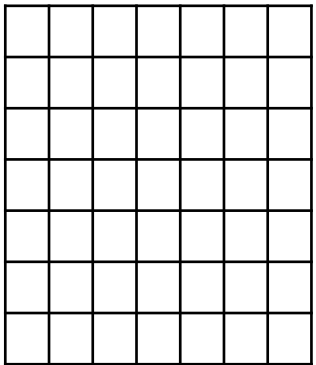
filtering



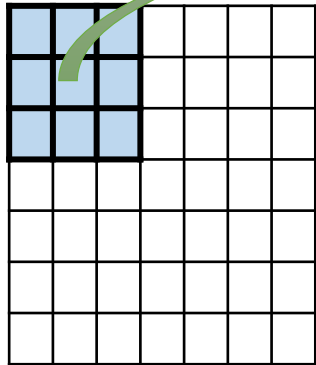
7x7 dst image
all pixel = 0.0

padding

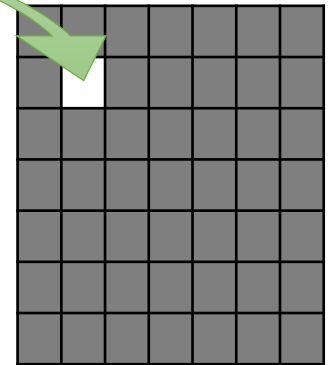
- padding을 안한다면?



7x7 image



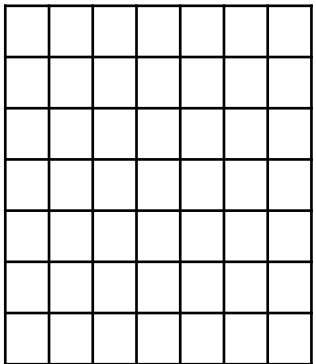
7x7 image
filtering



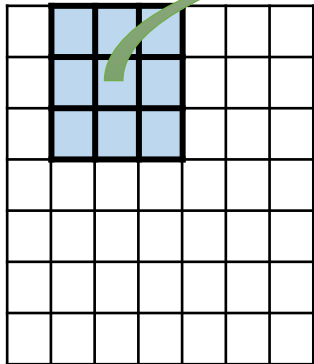
7x7 dst image

padding

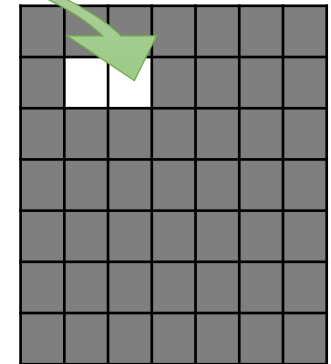
- padding을 안한다면?



7x7 image



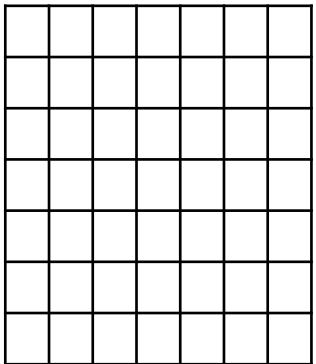
7x7 image
filtering



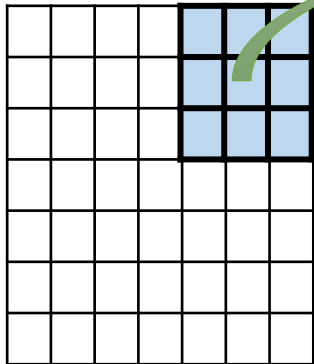
7x7 dst image

padding

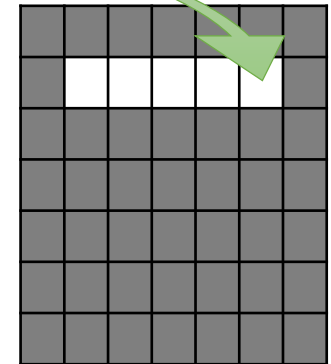
- padding을 안한다면?



7x7 image



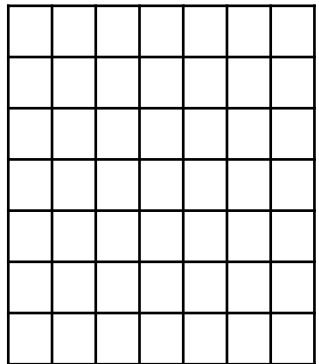
7x7 image
filtering



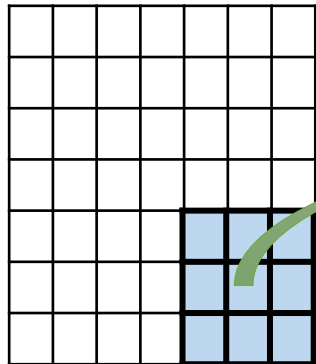
7x7 dst image

padding

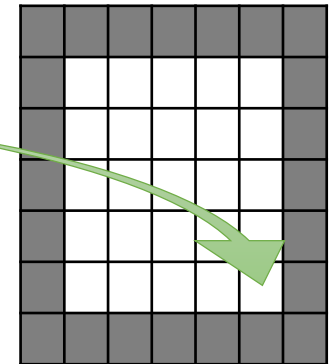
- padding을 안한다면?



7x7 image



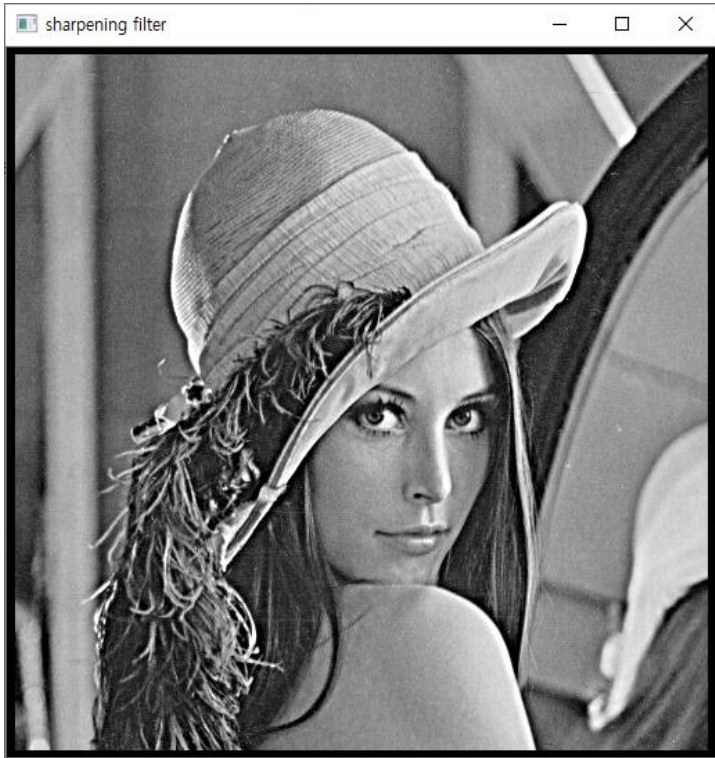
7x7 image
filtering



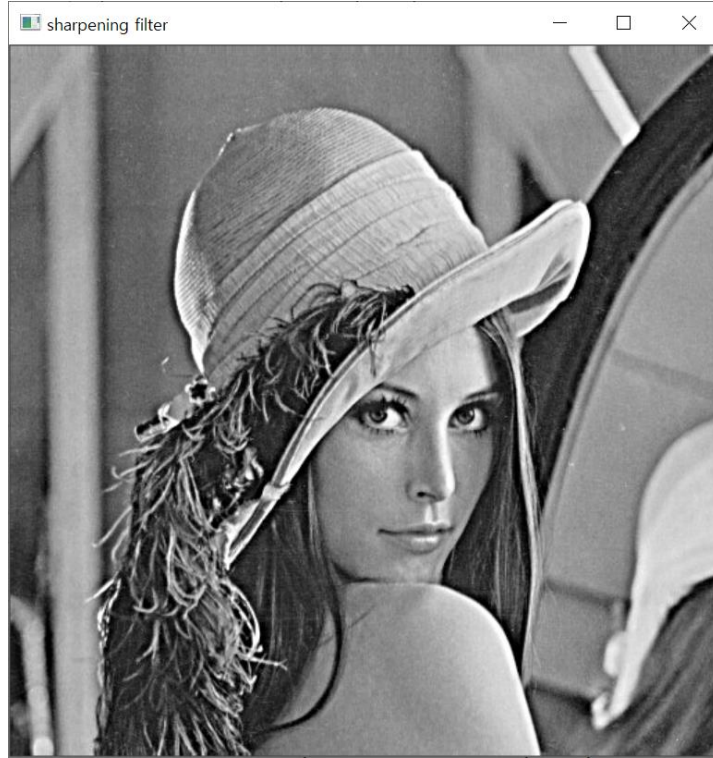
7x7 dst image

padding

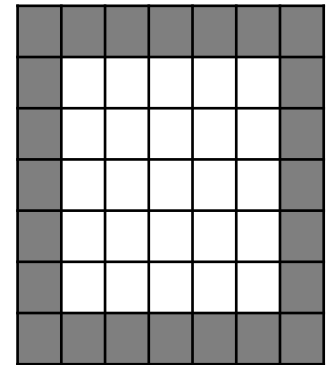
- padding을 안한다면?



padding을 안한 경우



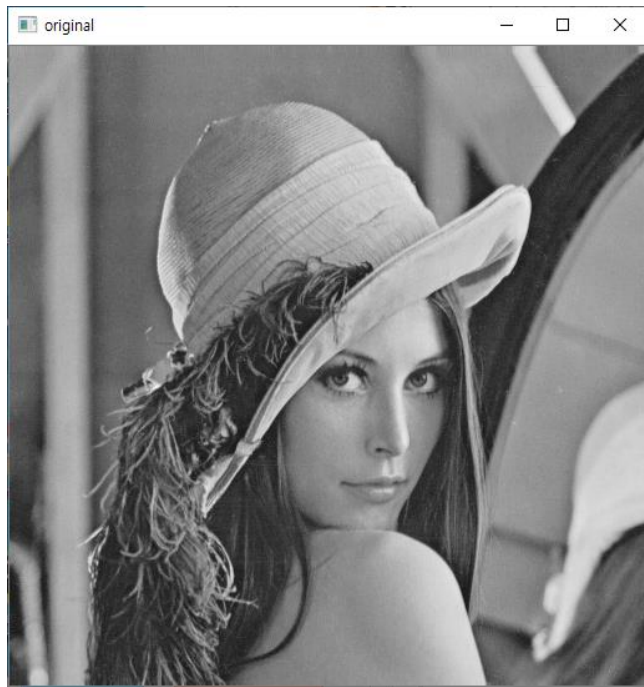
padding을 한 경우



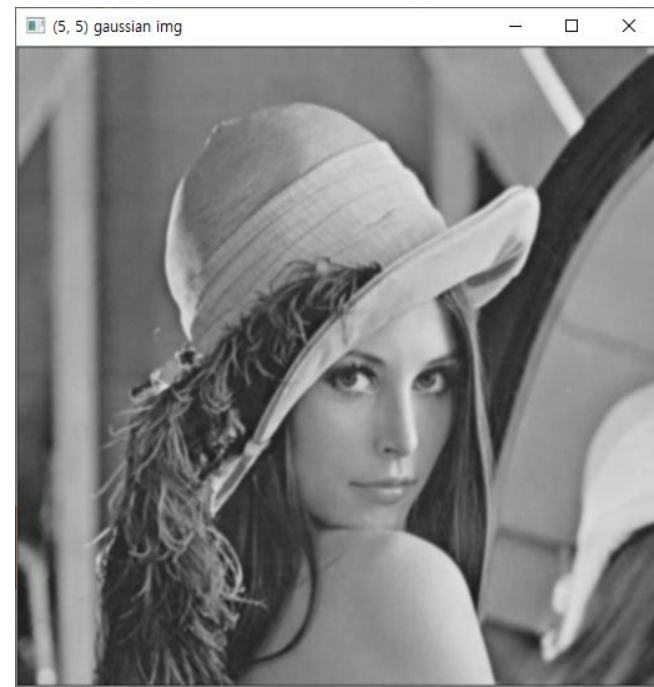
7x7 dst image

Gaussian filter

- **Gaussian filter**
 - 2D Gaussian filter
 - 1D Gaussian filter



original



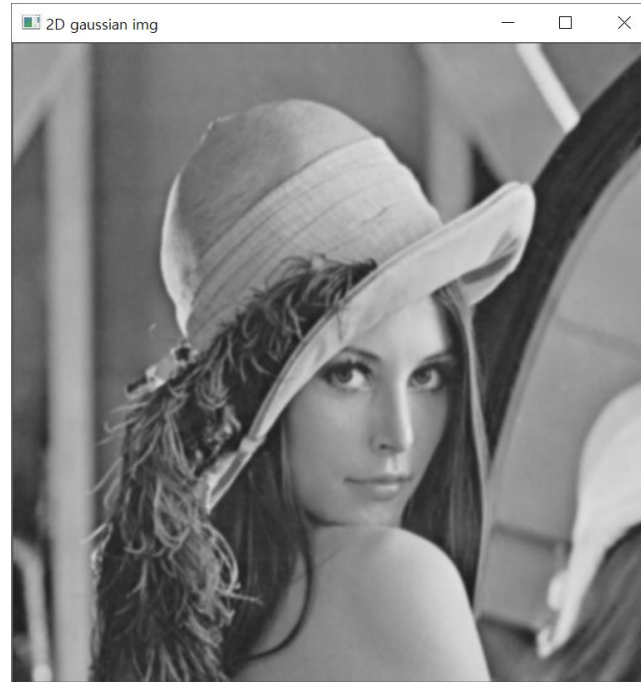
Gaussian filter

Gaussian filter

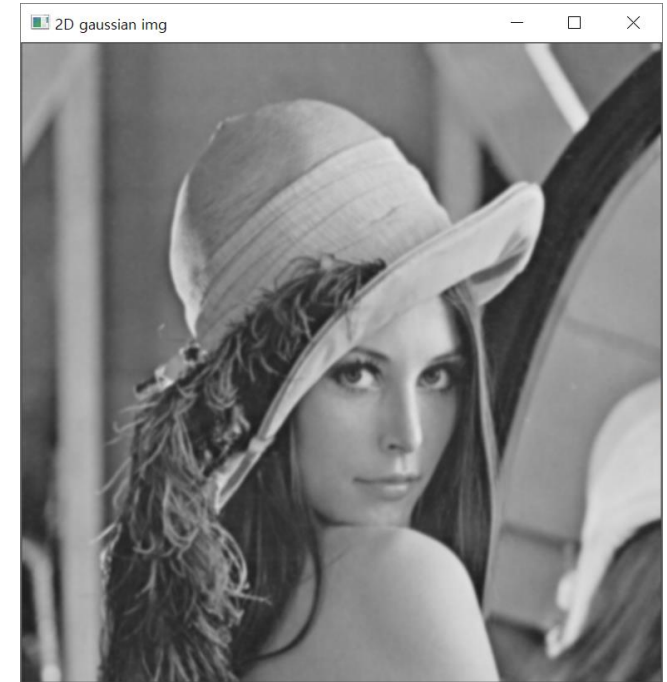
- 2D Gaussian filter



original



5 x 5 Gaussian filter
sigma=1



13 x 13 Gaussian filter
sigma = 1

Gaussian filter

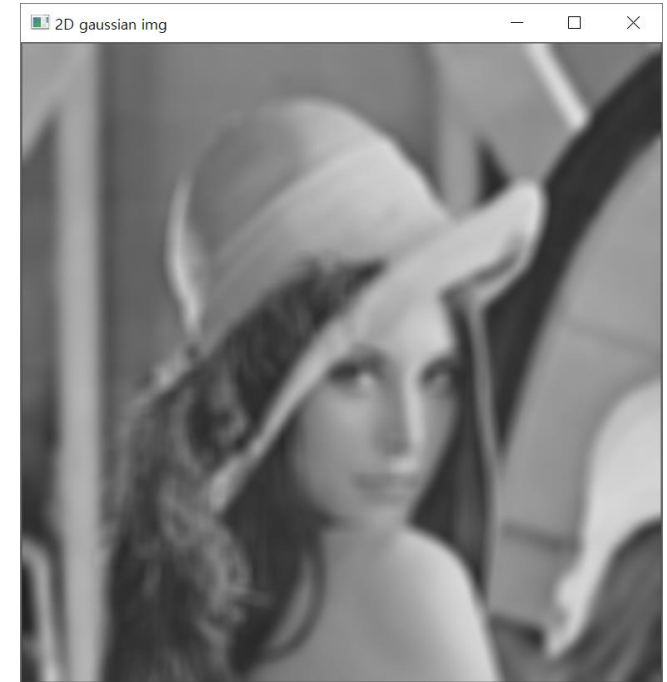
- 2D Gaussian filter



13 x 13 Gaussian filter
sigma = 1



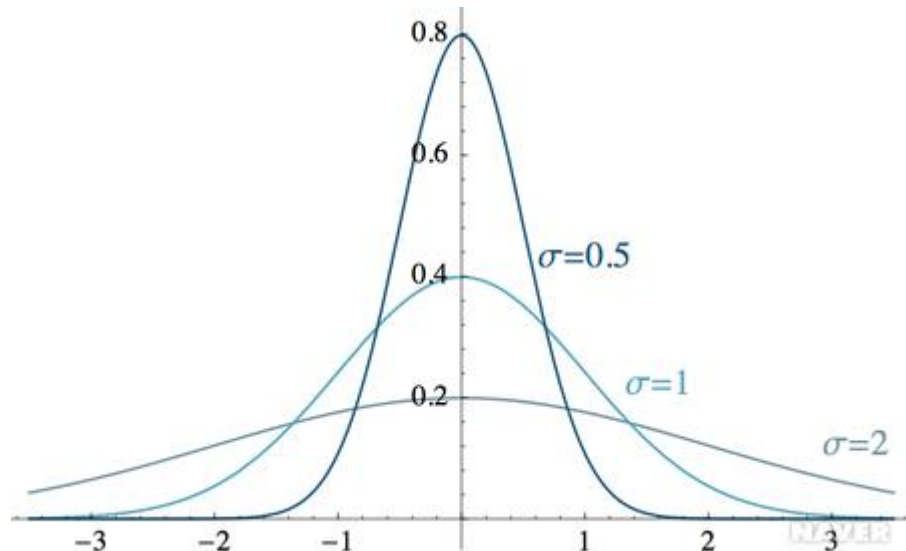
5 x 5 Gaussian filter
sigma=9



13 x 13 Gaussian filter
sigma = 9

Gaussian filter

- 2D Gaussian filter
 - 필터의 크기와 시그마 값의 관계



출처 : <https://terms.naver.com/entry.naver?docId=3405308&cid=47324&categoryId=47324>

<mask>

```
[[0.11110741 0.11111296 0.11110741]
 [0.11111296 0.11111852 0.11111296]
 [0.11110741 0.11111296 0.11110741]]
```

Gaussian filter

필터크기 : 3

sigma : 100

average filtering

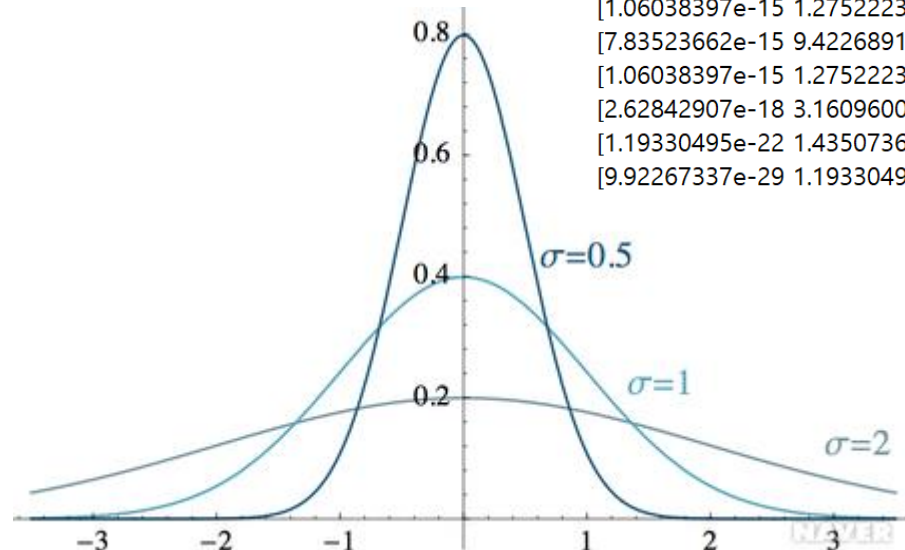
```
[[0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]]
```

Average filter

필터크기 : 3

Gaussian filter

- 2D Gaussian filter
 - 필터의 크기와 시그마 값의 관계



출처 : <https://terms.naver.com/entry.naver?docId=3405308&cid=47324&categoryId=47324>

<mask>

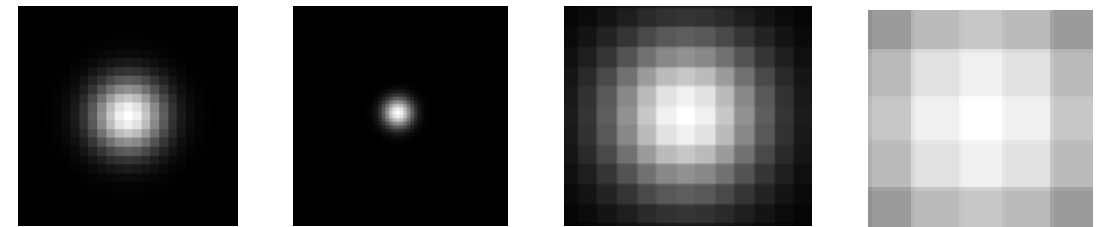
```
[[6.96247819e-08 2.80886418e-05 2.07548550e-04 2.80886418e-05 6.96247819e-08]
 [2.80886418e-05 1.13317669e-02 8.37310610e-02 1.13317669e-02 2.80886418e-05]
 [2.07548550e-04 8.37310610e-02 6.18693507e-01 8.37310610e-02 2.07548550e-04]
 [2.80886418e-05 1.13317669e-02 8.37310610e-02 1.13317669e-02 2.80886418e-05]
 [6.96247819e-08 2.80886418e-05 2.07548550e-04 2.80886418e-05 6.96247819e-08]]
```

5x5 sigma:0.5

<mask>

```
[[9.92267337e-29 1.19330495e-22 2.62842907e-18 1.06038397e-15 7.83523662e-15 1.06038397e-15 2.62842907e-18 1.19330495e-22 9.92267337e-29]
 [1.19330495e-22 1.43507365e-16 3.16096006e-12 1.27522230e-09 9.42268913e-09 1.27522230e-09 3.16096006e-12 1.43507365e-16 1.19330495e-22]
 [2.62842907e-18 3.16096006e-12 6.96247786e-08 2.80886404e-05 2.07548540e-04 2.80886404e-05 6.96247786e-08 3.16096006e-12 2.62842907e-18]
 [1.06038397e-15 1.27522230e-09 2.80886404e-05 1.13317663e-02 8.37310570e-02 1.13317663e-02 2.80886404e-05 1.27522230e-09 1.06038397e-15]
 [7.83523662e-15 9.42268913e-09 2.07548540e-04 8.37310570e-02 6.18693477e-01 8.37310570e-02 2.07548540e-04 9.42268913e-09 7.83523662e-15]
 [1.06038397e-15 1.27522230e-09 2.80886404e-05 1.13317663e-02 8.37310570e-02 1.13317663e-02 2.80886404e-05 1.27522230e-09 1.06038397e-15]
 [2.62842907e-18 3.16096006e-12 6.96247786e-08 2.80886404e-05 2.07548540e-04 2.80886404e-05 6.96247786e-08 3.16096006e-12 2.62842907e-18]
 [1.19330495e-22 1.43507365e-16 3.16096006e-12 1.27522230e-09 9.42268913e-09 1.27522230e-09 3.16096006e-12 1.43507365e-16 1.19330495e-22]
 [9.92267337e-29 1.19330495e-22 2.62842907e-18 1.06038397e-15 7.83523662e-15 1.06038397e-15 2.62842907e-18 1.19330495e-22 9.92267337e-29]]
```

9x9 sigma:0.5



다음 주 실습때 필터를 시각화 하는 것을 해볼 예정

Gaussian filter

- 2D Gaussian filter

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

x : $-n \sim n$ 범위의 mask에서의 x좌표(열)
 y : $-n \sim n$ 범위의 mask에서의 y좌표(행)
 σ : Gaussian 분포의 표준편차

n = mask의 행or열 길이// 2
 ex) mask의 크기가 5이면
 $n = 5//2 = 2$

5 x 5 gaussian filter $\sigma = 1$

0.0029	0.0133	0.0219	0.0133	0.0029
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0029	0.0133	0.0219	0.0133	0.0029

밝기 유지를 위해 총합은 1

5 x 5 gaussian filter $\sigma = 1$

$\frac{1}{sum} \cdot$

$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+0}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+0}{2}}$	$\frac{1}{2\pi} e^0$	$\frac{1}{2\pi} e^{-\frac{1+0}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+0}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$

Gaussian filter

- 2D Gaussian filter 만드는 법1
 - 단순히 n^2 번 반복하기

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

x : $-n \sim n$ 범위의 mask에서의 x좌표(열)
 y : $-n \sim n$ 범위의 mask에서의 y좌표(행)
 σ : Gaussian 분포의 표준편차

n = mask의 행or열 길이 // 2
 ex) mask의 크기가 5이면
 $n = 5 // 2 = 2$

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Gaussian filter

$x = -2$
 $y = -2$
 $\text{sigma} = 1$

$$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$$

0.0029	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Gaussian filter

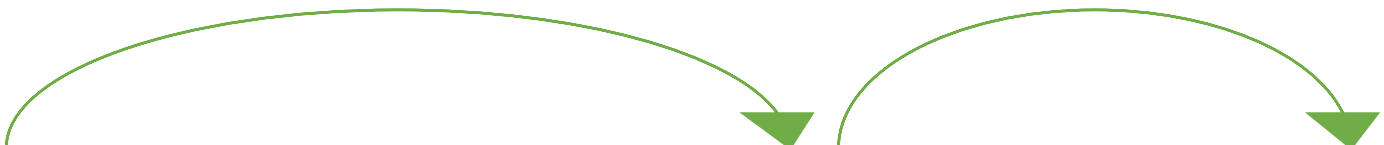
Gaussian filter

- 2D Gaussian filter 만드는 법1
 - 단순히 n^2 번 반복하기

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

x : $-n \sim n$ 범위의 mask에서의 x좌표(열)
 y : $-n \sim n$ 범위의 mask에서의 y좌표(행)
 σ : Gaussian 분포의 표준편차

n = mask의 행or열 길이// 2
ex) mask의 크기가 5이면
 $n = 5//2 = 2$



0.0029	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Gaussian filter

$x = -1$
 $y = -2$
 $\text{sigma} = 1$

$$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$$

0.0029	0.0131	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Gaussian filter

Gaussian filter

- 2D Gaussian filter 만드는 법1
 - 단순히 n^2 번 반복하기

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

x : $-n \sim n$ 범위의 mask에서의 x좌표(열)
 y : $-n \sim n$ 범위의 mask에서의 y좌표(행)
 σ : Gaussian 분포의 표준편차

n = mask의 행or열 길이 // 2
 ex) mask의 크기가 5이면
 $n = 5 // 2 = 2$

0.0029	0.0131
...
...
...
...	0

Gaussian filter

$x = 2$
 $y = 2$
 $\sigma = 1$

$$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$$

0.0029	0.0131
...
...
...
...	0.0029

Gaussian filter

Gaussian filter

- 2D Gaussian filter 만드는 법1
 - 단순히 n^2 번 반복하기

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

x : $-n \sim n$ 범위의 mask에서의 x좌표(열)
 y : $-n \sim n$ 범위의 mask에서의 y좌표(행)
 σ : Gaussian 분포의 표준편차

n = mask의 행or열 길이// 2
ex) mask의 크기가 5이면
 $n = 5//2 = 2$

```
[[0.00291502 0.01306423 0.02153928 0.01306423 0.00291502]
 [0.01306423 0.05854983 0.09653235 0.05854983 0.01306423]
 [0.02153928 0.09653235 0.15915494 0.09653235 0.02153928]
 [0.01306423 0.05854983 0.09653235 0.05854983 0.01306423]
 [0.00291502 0.01306423 0.02153928 0.01306423 0.00291502]]
```

Gaussian filter



총 합을 1로 만들기

```
[[0.00296902 0.01330621 0.02193823 0.01330621 0.00296902]
 [0.01330621 0.0596343 0.09832033 0.0596343 0.01330621]
 [0.02193823 0.09832033 0.16210282 0.09832033 0.02193823]
 [0.01330621 0.0596343 0.09832033 0.0596343 0.01330621]
 [0.00296902 0.01330621 0.02193823 0.01330621 0.00296902]]
```

Gaussian filter

Gaussian filter

- 2D Gaussian filter 만드는 법2
 - numpy를 이용해 한 번에 만들기

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$x : -n \sim n$ 범위의 mask에서의 x좌표(열)
 $y : -n \sim n$ 범위의 mask에서의 y좌표(행)
 σ : Gaussian 분포의 표준편차

$n = \text{mask의 행or열 길이} // 2$
 ex) mask의 크기가 5이면
 $n = 5 // 2 = 2$

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Gaussian filter



0.0029	0.0131
...
...
...
...	0.0029

Gaussian filter
 (총 합을 1로 만들기 전)

Gaussian filter

- 2D Gaussian filter 만드는 법2
 - numpy를 이용해 한 번에 만들기

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$x : -n \sim n$ 범위의 mask에서의 x좌표(열) $n = \text{mask의 행or열 길이} // 2$
 $y : -n \sim n$ 범위의 mask에서의 y좌표(행) ex) mask의 크기가 5이면
 $\sigma : \text{Gaussian 분포의 표준편차}$ $n = 5 // 2 = 2$

$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+0}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+0}{2}}$	$\frac{1}{2\pi} e^0$	$\frac{1}{2\pi} e^{-\frac{1+0}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+0}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$

<<- Gaussian filter
(총 합을 1로 만들기 전)

○ 표시한 부분만 다름

Gaussian filter

- 2D Gaussian filter 만드는 법2
 - numpy를 이용해 한 번에 만들기

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$x : -n \sim n$ 범위의 mask에서의 x좌표(열)
 $y : -n \sim n$ 범위의 mask에서의 y좌표(행)
 σ : Gaussian 분포의 표준편차

$n = \text{mask의 행or열 길이} // 2$
 ex) mask의 크기가 5이면
 $n = 5 // 2 = 2$

4+4	1+4	0+4	1+4	4+4
4+1	1+1	0+1	1+1	4+1
4+0	1+0	0+0	1+0	4+0
4+1	1+1	0+1	1+1	4+1
4+4	1+4	0+4	1+4	4+4

$$\frac{1}{2\pi} e^{-\frac{???}{2}}$$



$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+0}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+0}{2}}$	$\frac{1}{2\pi} e^0$	$\frac{1}{2\pi} e^{-\frac{1+0}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+0}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+1}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+1}{2}}$
$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{0+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{1+4}{2}}$	$\frac{1}{2\pi} e^{-\frac{4+4}{2}}$

Gaussian filter

- 2D Gaussian filter 만드는 법2
 - numpy를 이용해 한 번에 만들기

4+4	1+4	0+4	1+4	4+4
4+1	1+1	0+1	1+1	4+1
4+0	1+0	0+0	1+0	4+0
4+1	1+1	0+1	1+1	4+1
4+4	1+4	0+4	1+4	4+4

<<- 만드는 방법은?

1. np.zeros((5,5)) 를 하고 각각 값을 채운다.
2. np.mgrid[] 를 사용한다.

```
y, x = np.mgrid[-1:2, -1:2]
'''
y = [[-1,-1,-1],
      [ 0, 0, 0],
      [ 1, 1, 1]]
x = [[-1, 0, 1],
      [-1, 0, 1],
      [-1, 0, 1]]
'''
```

Gaussian filter

- 2D Gaussian filter 만드는 법2
 - numpy를 이용해 한 번에 만들기

```
x = np.array([[1, 2], [3, 4]])  
print(x)
```

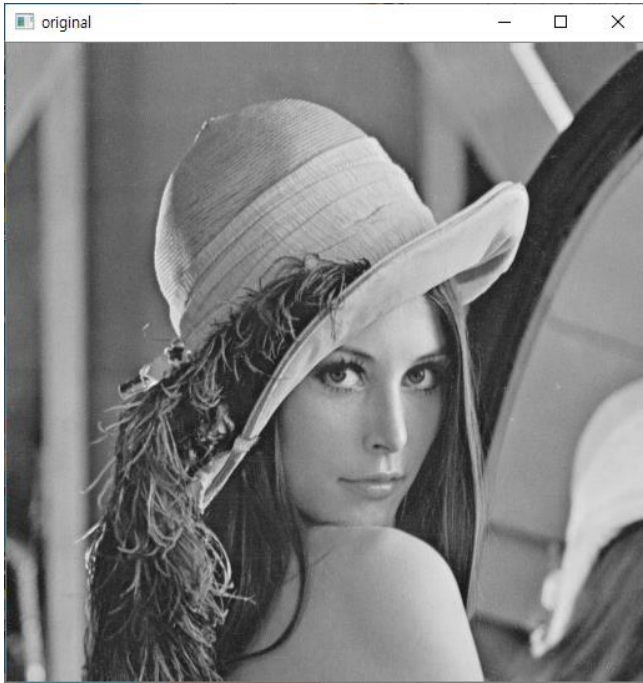
```
x = x / np.sum(x)  
print(x)
```

```
[[1 2]  
 [3 4]]  
[[0.1 0.2]  
 [0.3 0.4]]
```

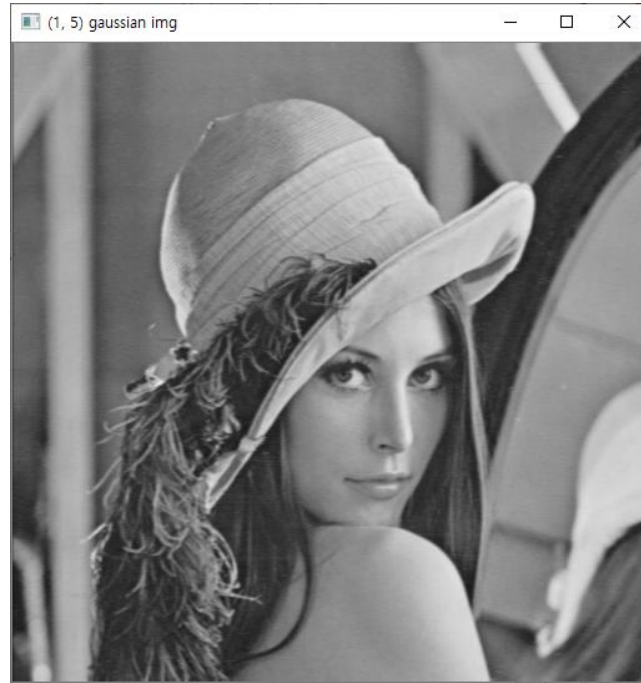
< numpy 를 사용해서 총 합을 1로 만드는 방법 >

Gaussian filter

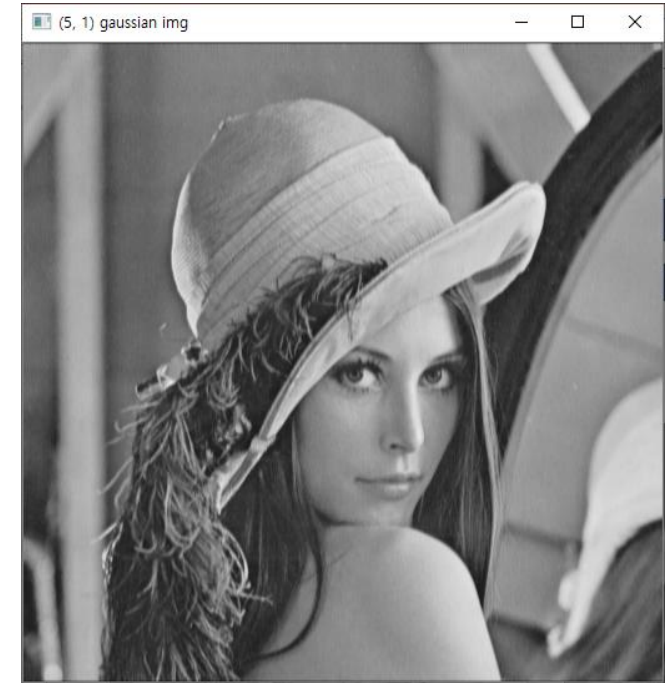
- 1D Gaussian filter



original



1 x 5 Gaussian filter
sigma=1



5 x 1 Gaussian filter
sigma = 1

Gaussian filter

- 1D Gaussian filter

$$G(x) = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}} \right)$$

$$G(y) = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-y^2}{2\sigma^2}} \right)$$

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- 2차 Gaussian 식을, 각 축에 대해서 분리한 식.
- 1x5 Gaussian kernel

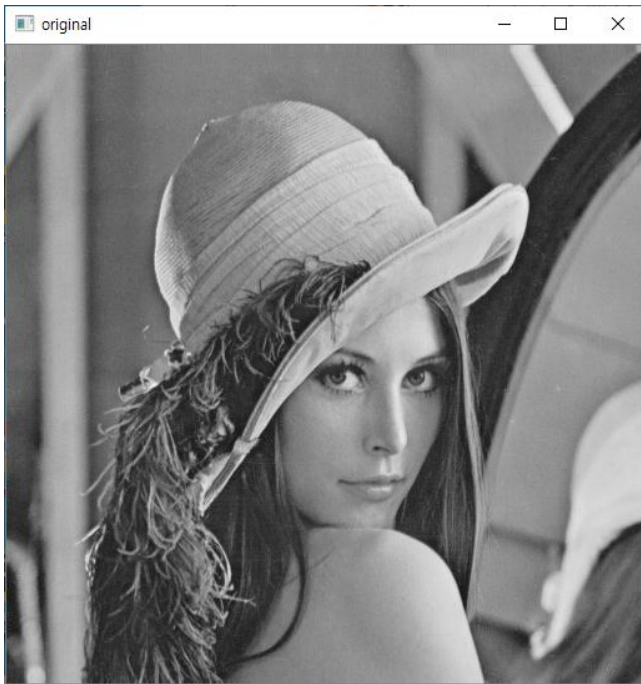
0.0544	0.2442	0.4026	0.2442	0.0544
--------	--------	--------	--------	--------

- 5x1 Gaussian kernel

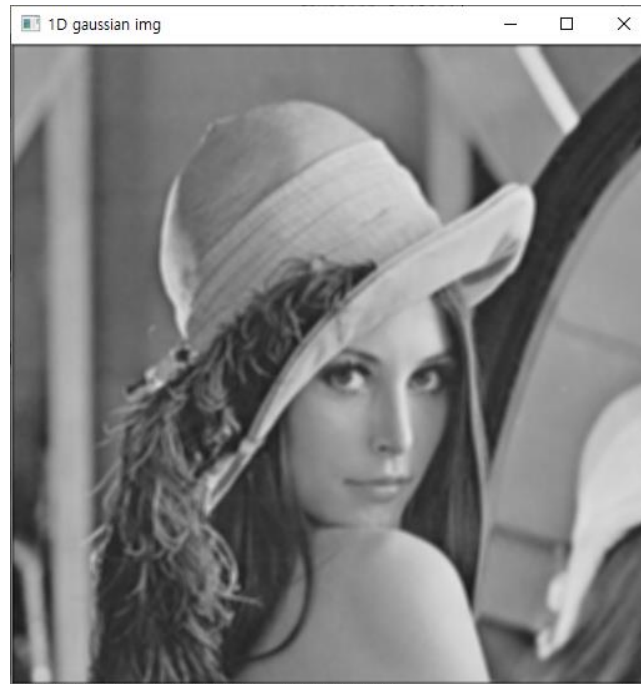
0.0544
0.2442
0.4026
0.2442
0.0544

Gaussian filter

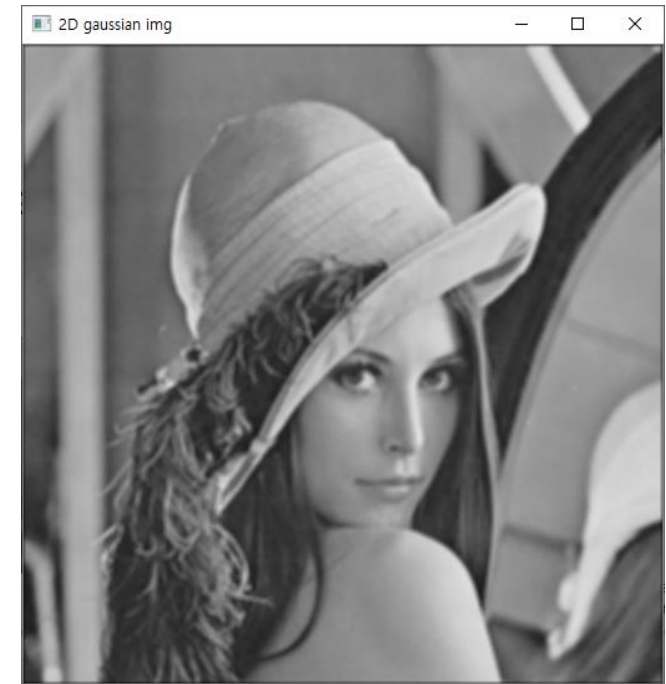
- 2D Gaussian filter and 1D Gaussian filter



original



(5 x 1), (1 x 5) Gaussian filter
sigma=3



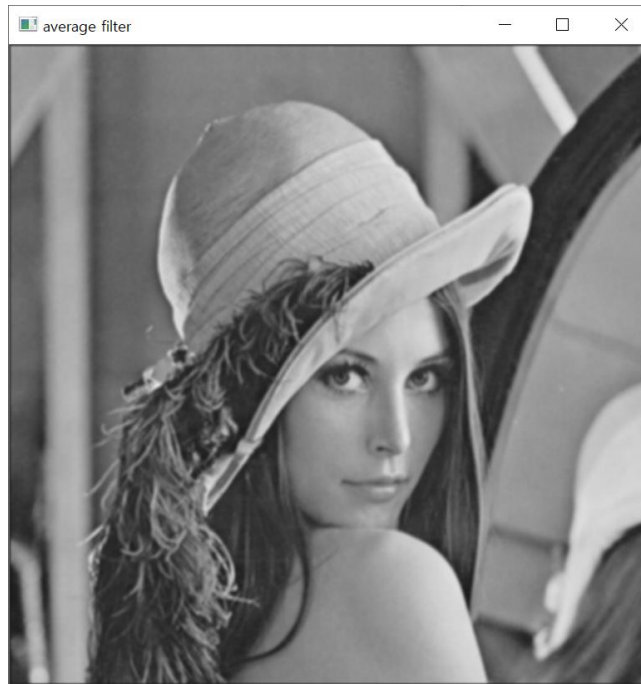
(5 x 5) Gaussian filter
sigma = 3

과제 1

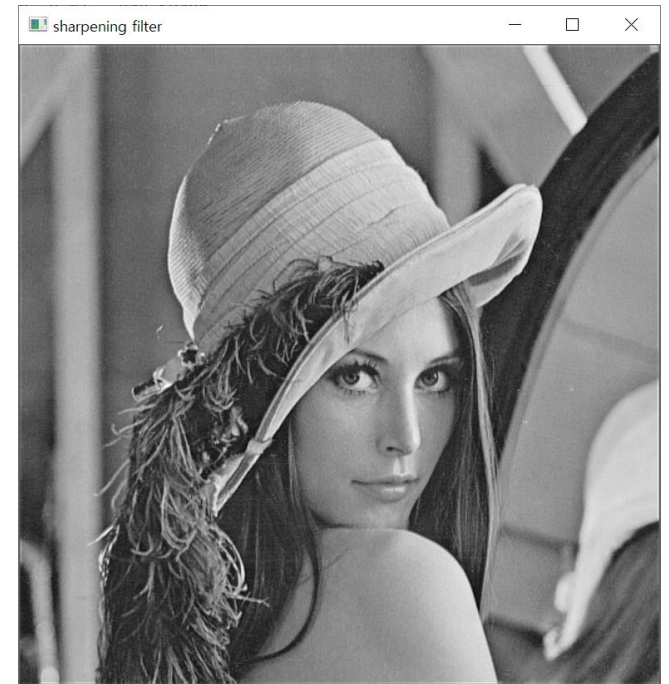
- average filter 및 sharpening filter 구현
 - my_filtering()함수 완성



original



3x3 average filter



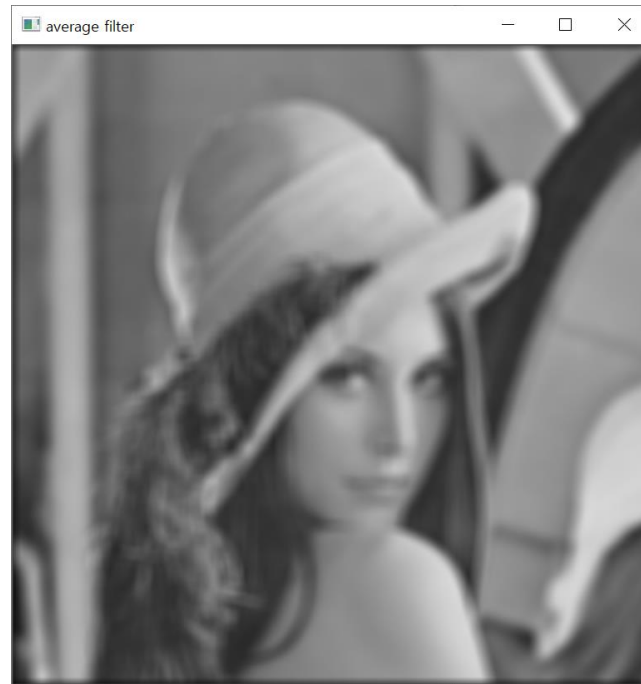
3x3 sharpening filter

과제 1

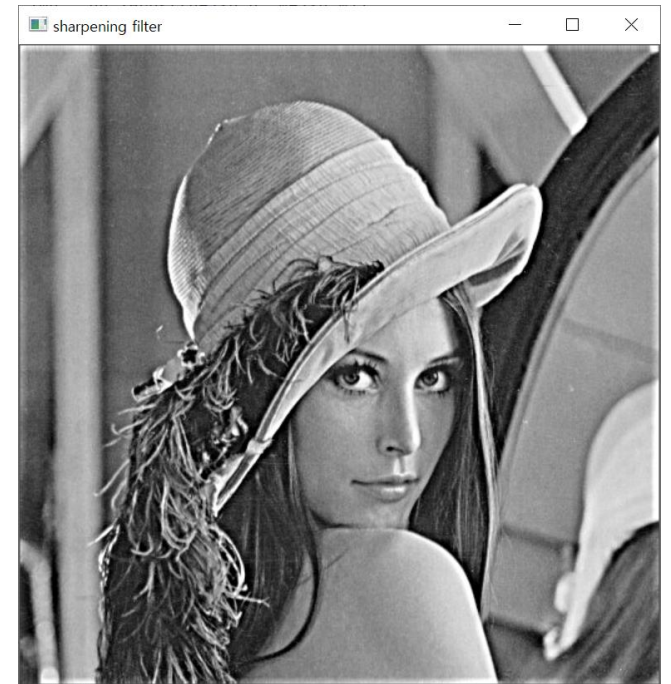
- average filter 및 sharpening filter 구현
 - my_filtering()함수 완성



original



11x13 average filter



11x13 sharpening filter

과제 1

- **get_average_mask(fshape):**
 - fshape : mask size
 - **get_sharpening_mask(fshape):**
 - fshape : mask size
 - **my_filtering(src, mask, pad_type='zero')**
 - src : 흑백 이미지
 - mask : filtering에 사용할 mask
 - pad_type : padding 타입 : 'zero' or 'repetition'
- return
- dst : filtering 결과 이미지

과제1

- sharpening filter

- (3x3)

0	0	0
0	2	0
0	0	0

—

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

=

-1/9	-1/9	-1/9
-1/9	17/9	-1/9
-1/9	-1/9	-1/9

- (5x5)

0	0	0	0	0
0	0	0	0	0
0	0	2	0	0
0	0	0	0	0
0	0	0	0	0

—

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

=

-1/25	-1/25	-1/25	-1/25	-1/25
-1/25	-1/25	-1/25	-1/25	-1/25
-1/25	-1/25	49/25	-1/25	-1/25
-1/25	-1/25	-1/25	-1/25	-1/25
-1/25	-1/25	-1/25	-1/25	-1/25

과제1

- average filter

- (3x3)

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

- (3x5)

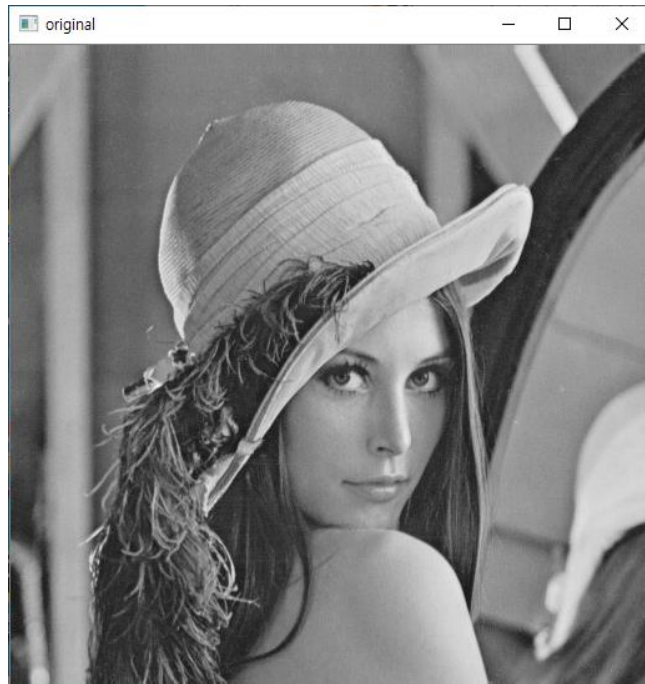
1/15	1/15	1/15	1/15	1/15
1/15	1/15	1/15	1/15	1/15
1/15	1/15	1/15	1/15	1/15

- (5x5)

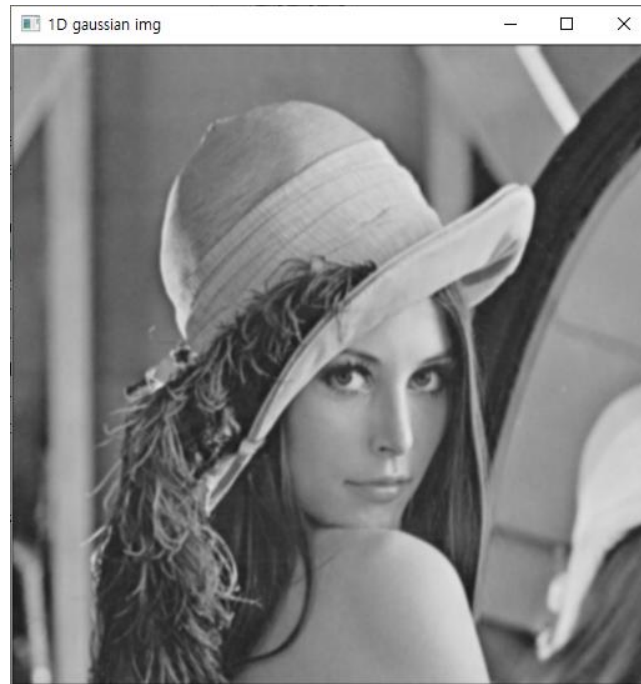
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

과제2

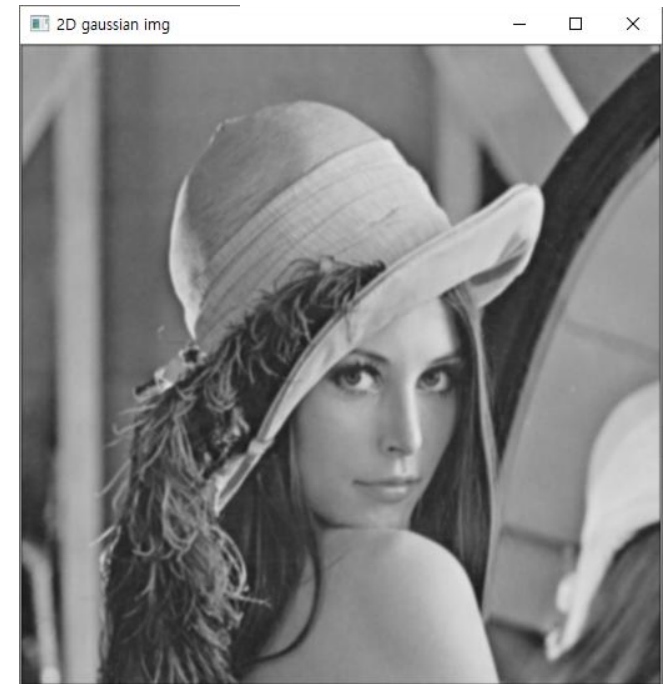
- 1차 Gaussian filter 완성 및 2차 Gaussian filter와 시간 비교



original



1차 Gaussian filter
sigma = 1



2차 Gaussian filter
sigma = 1

```
mask size : 5
1D gaussian filter
<mask>
[[[0.05448868]
  [0.24420134]
  [0.40261995]
  [0.24420134]
  [0.05448868]]]
<mask>
[[[0.05448868 0.24420134 0.40261995 0.24420134 0.05448868]]]
1D time : 1.7350328000000002
2D gaussian filter
<mask>
[[[0.00296902 0.01330621 0.02193823 0.01330621 0.00296902]
  [0.01330621 0.0596343 0.09832033 0.0596343 0.01330621]
  [0.02193823 0.09832033 0.16210282 0.09832033 0.02193823]
  [0.01330621 0.0596343 0.09832033 0.0596343 0.01330621]
  [0.00296902 0.01330621 0.02193823 0.01330621 0.00296902]]]
2D time : 3.2982677999999996
```


과제2

- 1차 Gaussian filter 완성 및 2차 Gaussian filter와 시간 비교

```
def my_get_Gaussian2D_mask(msize, sigma=1):  
    #####  
    # ToDo  
    # 2D gaussian filter 만들기  
    #####  
    y, x = ???  
    ...  
    y, x = np.mgrid[-1:2, -1:2]  
    y = [[-1, -1, -1],  
         [ 0,  0,  0],  
         [ 1,  1,  1]]  
    x = [[-1,  0,  1],  
         [-1,  0,  1],  
         [-1,  0,  1]]  
    ...  
  
    # 2차 gaussian mask 생성  
    gaus2D = ???  
    # mask의 총 합 = 1  
    gaus2D /= ???  
  
    return gaus2D
```

```
def my_get_Gaussian1D_mask(msize, sigma=1):  
    #####  
    # ToDo  
    # 1D gaussian filter 만들기  
    #####  
    x = ???  
    ...  
    x = np.full((1, 3), [-1, 0, 1])  
    x = [[ -1,  0,  1]]  
  
    x = np.array([[ -1,  0,  1]])  
    x = [[ -1,  0,  1]]  
    ...  
  
    gaus1D = ???  
  
    # mask의 총 합 = 1  
    gaus1D /= ???  
    return gaus1D
```

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$$G(x) = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}} \right)$$

$$G(y) = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-y^2}{2\sigma^2}} \right)$$

과제2

- **my_get_Gaussian2D_mask(msize, sigma=1)**
 - msize : mask size
- **my_get_Gaussian2D_mask(msize, sigma=1)**
 - msize : mask size
- **1차 Gaussian filter 완성 및 2차 Gaussian filter와 시간 비교**
 - Gaussian filter 만드는 코드 완성 및 시간 비교하기

과제

• 보고서 작성 방법

• 과제 1

- 2개의 mask 작성 함수 및 filtering 함수 작성하여 작성 방법에 대한 내용 포함
- Original 이미지 및 4개의 필터링을 적용한 이미지(3x3 average_filter, 3x3 sharpening filter, 11x13 average filter, 11x13 sharpening filter)를 보고서에 비교 및 작성

▪ 과제 2

- 2d gaussian mask와 1d gaussian mask를 만드는 함수 작성 및 방법에 대한 내용 포함
- Original image 및 1d gaussian, 2d gaussian filtering을 적용한 이미지를 보고서에 포함
- 1d gaussian과 2d gaussian filtering의 속도 차이를 비교한 내용을 작성할 것

과제

• 제출 방법

- 코드 파일

- 구현 결과가 포함된 python 파일(.py)
- 이번 과제에서는 my_filtering과 my_gaussian을 모두 사용

- 보고서

- [IP]201900000_홍길동_2주차_과제.pdf
- 보고서 양식 사용
- PDF 파일 형식으로 제출(pdf가 아닌 다른 양식으로 제출시 감점)

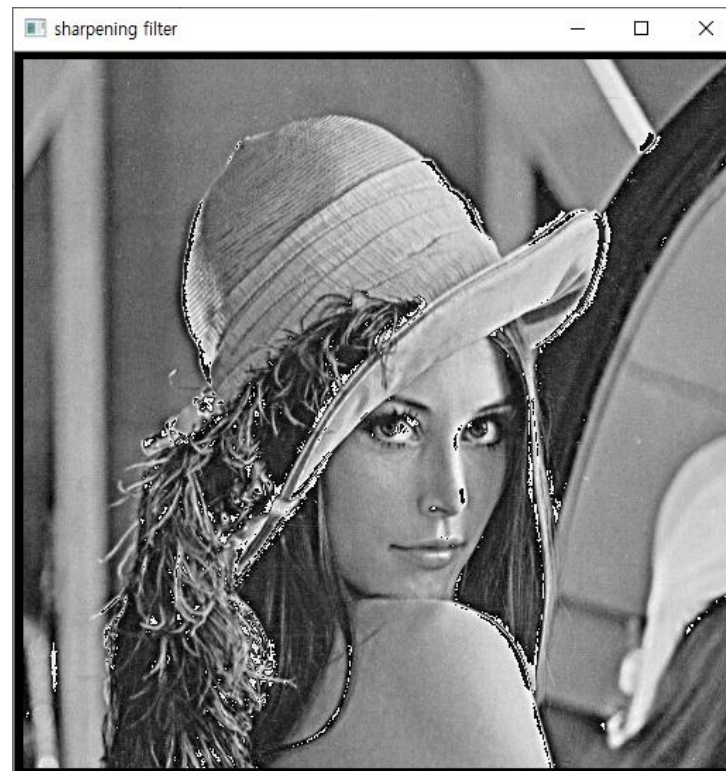
- 제출 파일

- [IP]201900000_홍길동_2주차_과제.zip
- .py 파일과 pdf 보고서를 하나의 파일로 압축한 후, 양식에 맞는 이름으로 제출

과제

• 주의사항

- 결과 이미지가 이상하게 보임
 - 오버플로우 문제를 해결하면 결과가 제대로 나옴
- 4중 for문을 사용하면 시간이 너무 오래 걸림
 - numpy 기초 실습때 배운 내용을 한번 다시 보기
- 과제 1 구현에 `cv2.filter2D()` 함수 사용 금지
 - filtering을 바로 수행해주는 cv2 함수 사용 금지



출석체크

- Zoom 퇴장 전, [학번 이름]을 채팅창에 올린 후 퇴장해 주시기 바랍니다.

QnA