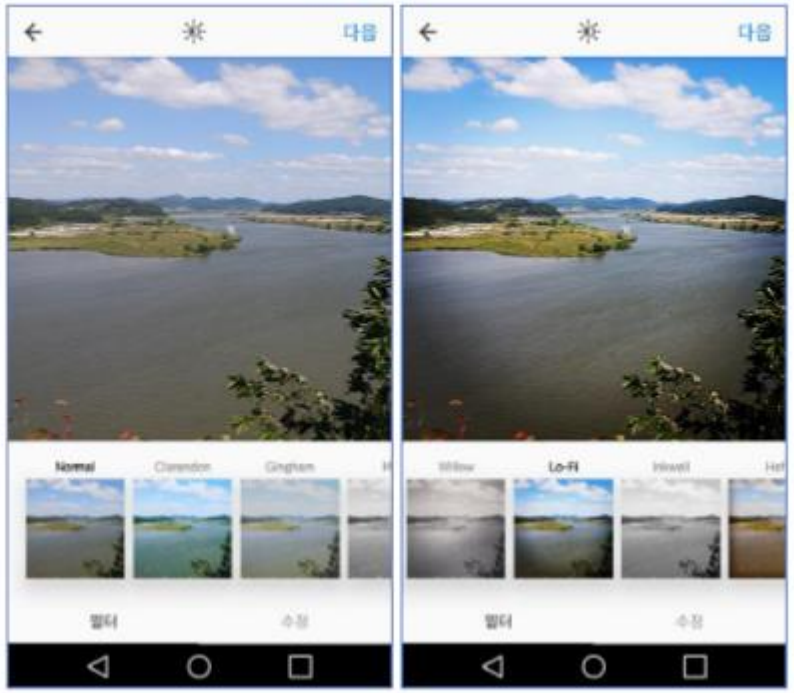
필터링

필터링 이해하기

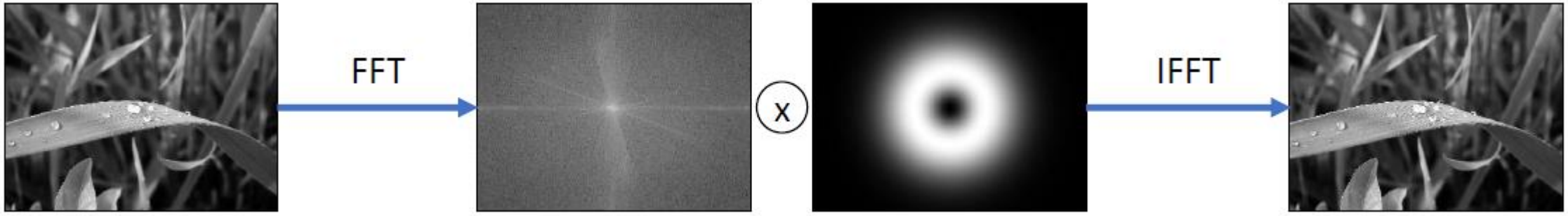
영상의 필터링 (image filtering)

- 영상에서 필요한 정보만 통과시키고 원치 않는 정보는 걸러내는 작업



필터링 이해하기

주파수 공간에서의 필터링 (Frequency domain filtering)



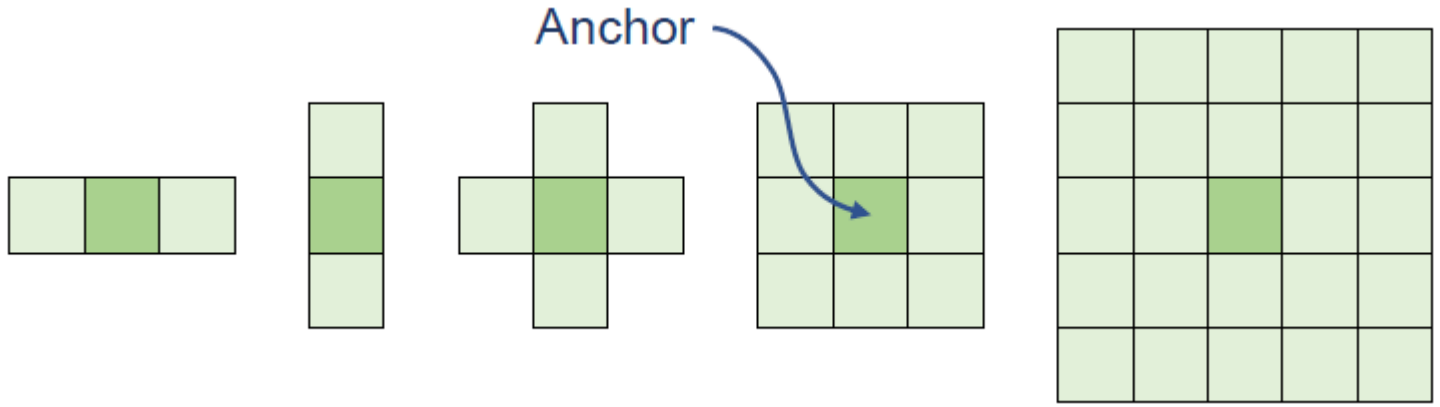
공간적 필터링 (Spatial domain filtering)

- 영상의 픽셀 값을 직접 이용하는 필터링 방법
  - 대상 좌표의 픽셀 값과 주변 픽셀 값을 동시에 사용
- 주로 마스크(mask) 연산을 이용함

  (마스크 = 커널(kernel) = 윈도우(window) = 템플릿(template))

필터링 이해하기

다양한 모양과 크기의 마스크

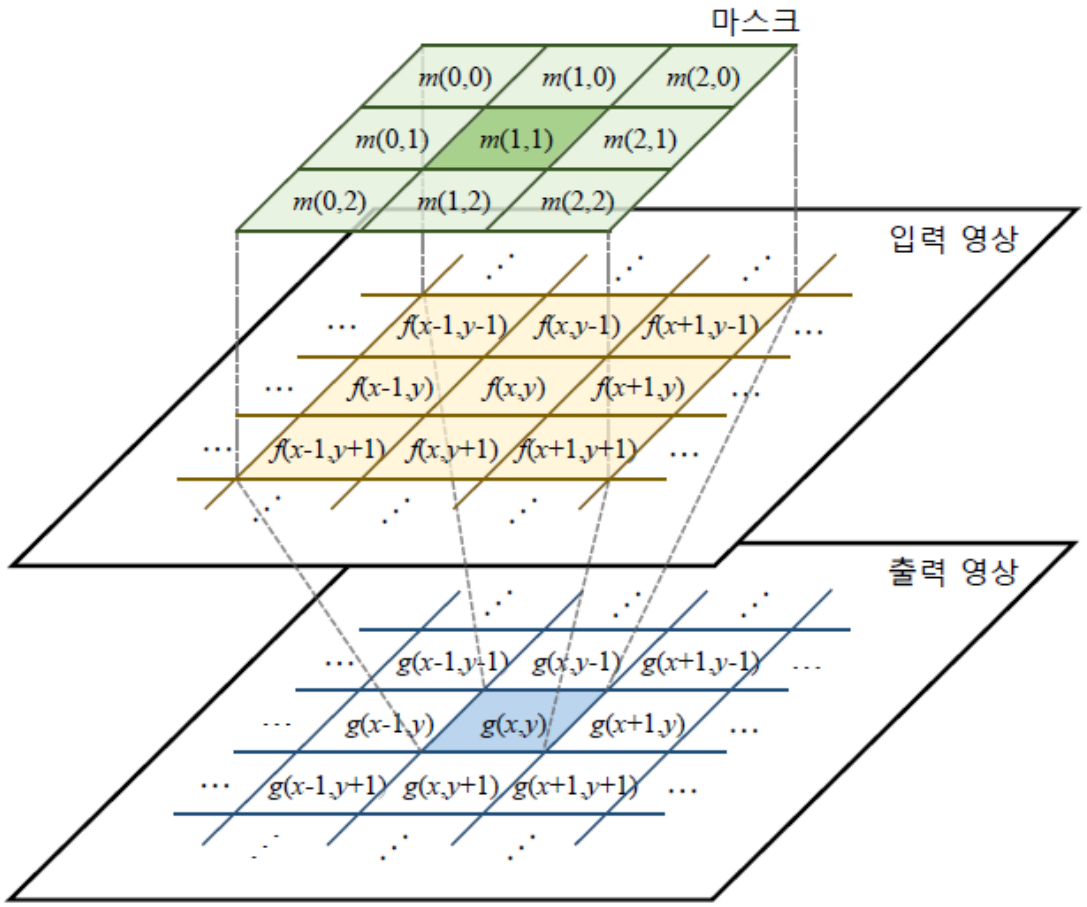


마스크의 형태와 값에 따라 필터의 역할이 결정됨

- 영상 부드럽게 만들기
- 영상 날카롭게 만들기
- 에지(edge) 검출
- 잡음 제거

필터링 이해하기

3X3 크기의 마스크를 이용한 공간적 필터링



$$
\begin{aligned}
g(x,y) = {} & m(0,0)f(x-1,y-1) \\
& +m(1,0)f(x,y-1) \\
& +m(2,0)f(x+1,y-1) \\
& +m(0,1)f(x-1,y) \\
& +m(1,1)f(x,y) \\
& +m(2,1)f(x+1,y) \\
& +m(0,2)f(x-1,y+1) \\
& +m(1,2)f(x,y+1) \\
& +m(2,2)f(x+1,y+1)
\end{aligned}
$$

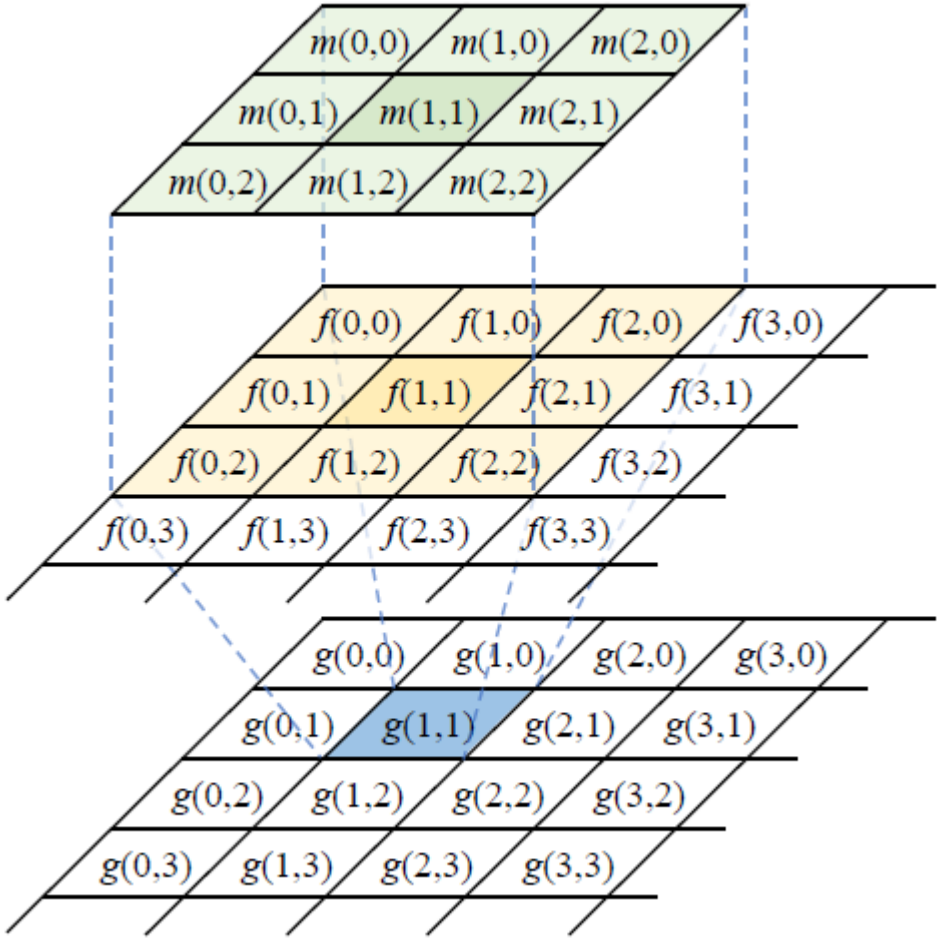
Correlation (Convolution)

$$
g(x,y) = \sum_{j=0}^{2}\sum_{i=0}^{2} m(i,j)f(x+i-1,y+j-1)
$$

필터링 이해하기

3X3 크기의 마스크를 이용한 공간적 필터링

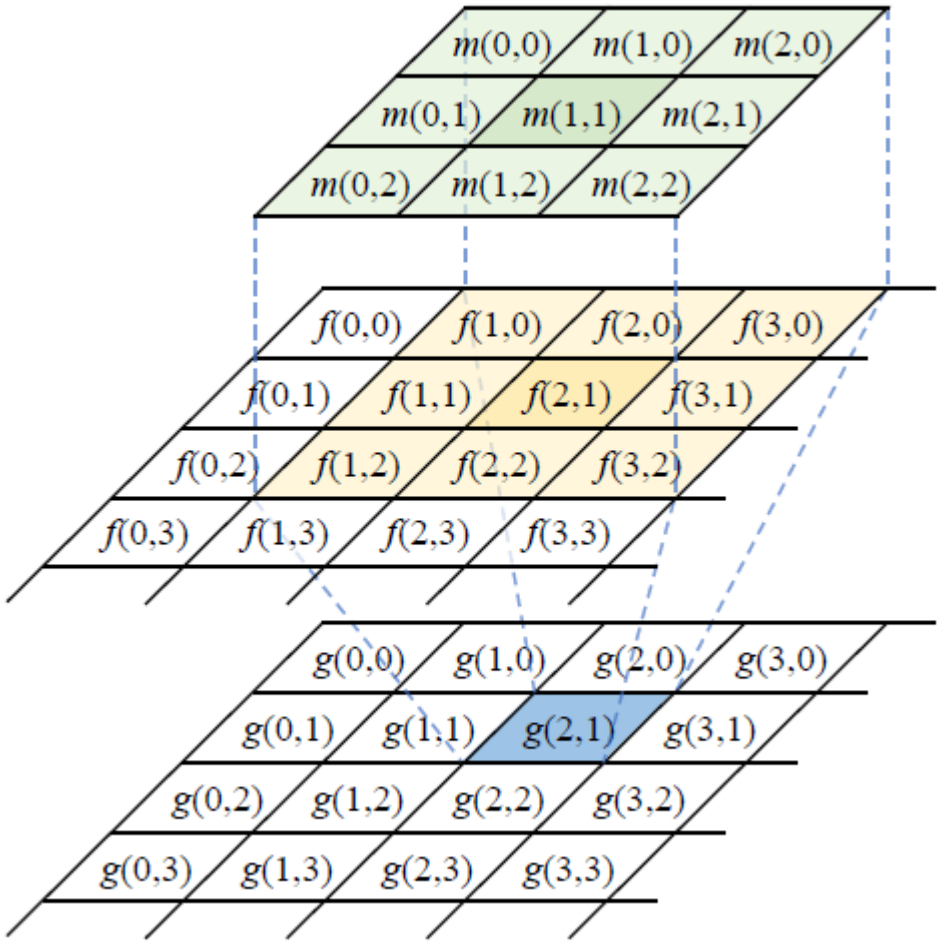
- (1, 1) 좌표에서 필터링



필터링 이해하기

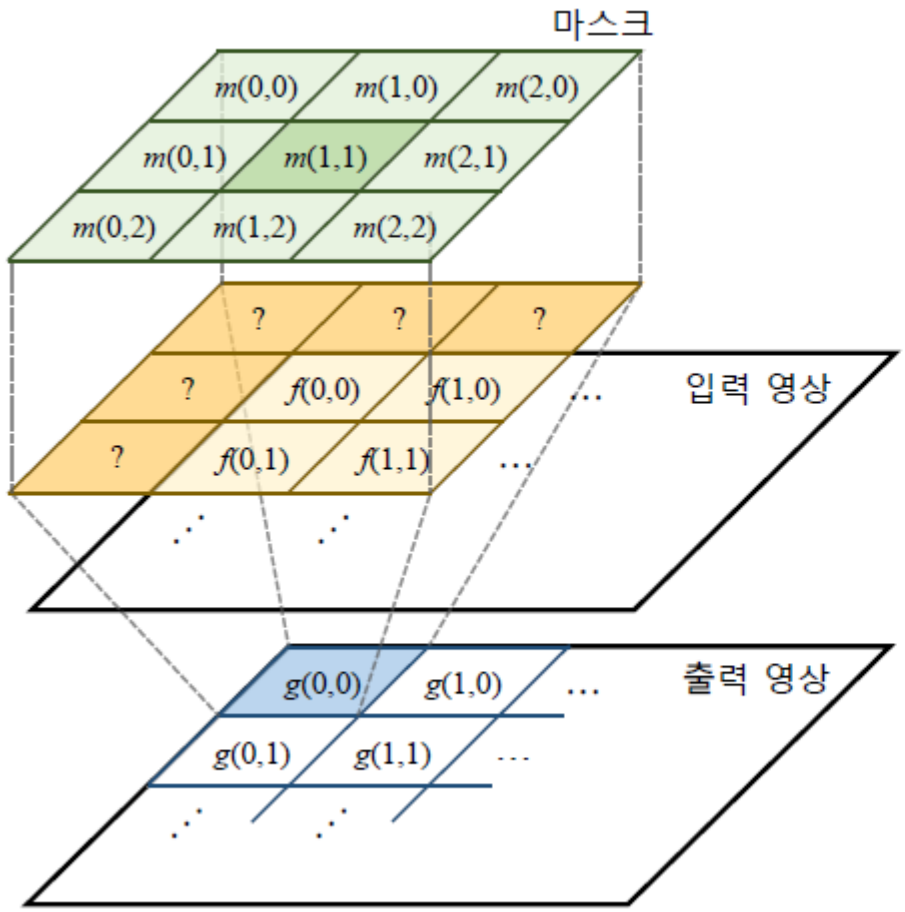
3X3 크기의 마스크를 이용한 공간적 필터링

- (2, 1) 좌표에서 필터링

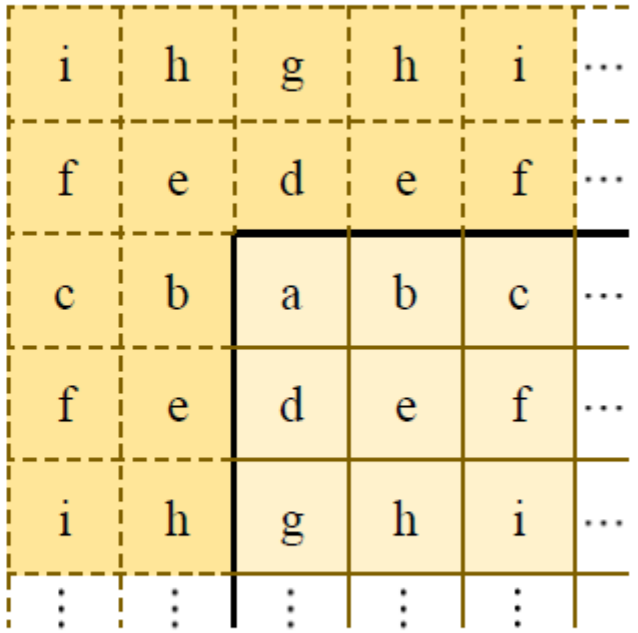


필터링 이해하기

최외곽 픽셀 처리



최외곽 바깥에 가상의 픽셀이 있다고 가정



필터링 이해하기

OpenCV 필터링에서 지원하는 가장자리 픽셀 확장 방법

BorderTypes 열거형 상수	설명
BORDER_CONSTANT	0 0 0 a b c d e f g h 0 0 0
BORDER_REPLICATE	a a a a b c d e f g h h h h
BORDER_REFLECT	c b a a b c d e f g h h g f
BORDER_REFLECT_101	d c b a b c d e f g h g f e
BORDER_REFLECT101	BORDER_REFLECT_101과 같음
BORDER_DEFAULT	BORDER_REFLECT_101과 같음

필터링 이해하기

기본적인 2D 필터링

cv2.filter2D(src, ddepth, kernel, dst=None, anchor=None, delta=None, borderType=None) --> dst

- src : 입력 영상
- ddepth: 출력 영상 데이터 타입 . e.g) cv2.CV_8U, cv2.CV_32F, cv2.CV_64F
  - 1 을 지정하면 src 와 같은 타입의 dst 영상을 생성
- kernel: 필터 마스크 행렬. 실수형
- anchor: 고정점 위치. (-1, 1) 이면 필터 중앙을 고정점으로 사용
- delta: 추가적으로 더할 값
- borderType: 가장자리 픽셀 확장 방식
- dst: 출력 영상

블러링(1): 평균 값 필터

평균 값 필터(Mean filter)

- 영상의 특정 좌표 값을 주변 픽셀 값들의 산술 평균으로 설정
- 픽셀들 간의 그레이스케일 값 변화가 줄어들어 날카로운 에지가 무뎌지고, 영상에 있는 잡음의 영향이 사라지는 효과

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$\frac{1}{25}\times\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

블러링(1): 평균 값 필터

실제 영상에 평균 값 필터를 적용한 결과

- 마스크 크기가 커질수록 평균 값 필터 결과가 더욱 부드러워짐

  → 더 많은 연산량이 필요

원본 영상

3x3 크기의 마스크

5x5 크기의 마스크

블러링(1): 평균 값 필터

filter2D() 함수를 이용한 평균값 필터링 예제

```python
import sys
import numpy as np
import cv2

src = cv2.imread('rose.bmp', cv2.IMREAD_GRAYSCALE)

if src is None:
    print('Image load failed!')
    sys.exit()

#kernel = np.ones((3, 3), dtype=np.float64) / 9.
#dst = cv2.filter2D(src, -1, kernel)
dst = cv2.blur(src, ksize: (3, 3))

cv2.imshow( winname: 'src', src)
cv2.imshow( winname: 'dst', dst)
cv2.waitKey()

cv2.destroyAllWindows()
```

블러링(1): 평균 값 필터

평균 값 필터링 함수

cv2.blur(src, ksize, dst=None, anchor=None, borderType=None) -> dst

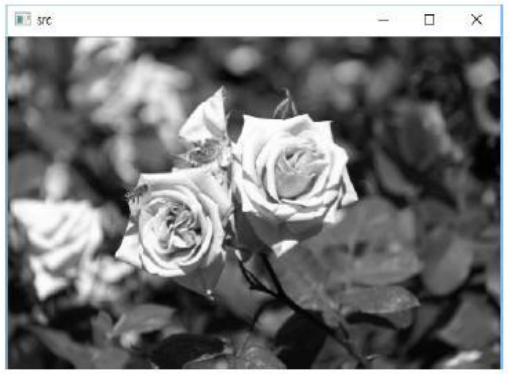
- src: 입력 영상
- ksize: 평균값 필터 크기. (width, height) 형태의 튜플.
- dst: 결과 영상. 입력 영상과 같은 크기 & 같은 타입.

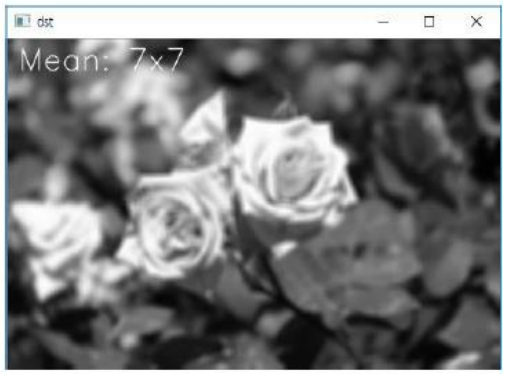
$$\text{kernel} = \frac{1}{\text{ksize.width} \times \text{ksize.height}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

블러링(1): 평균 값 필터

다양한 크기의 커널을 사용한 평균값 필터링 예제

```python
import sys
import numpy as np
import cv2

src = cv2.imread('rose.bmp', cv2.IMREAD_GRAYSCALE)

if src is None:
    print('Image load failed!')
    sys.exit()

cv2.imshow( winname: 'src', src)

for ksize in (3, 5, 7):
    dst = cv2.blur(src,  ksize: (ksize, ksize))

    desc = 'Mean: {}x{}'.format( *args: ksize, ksize)
    cv2.putText(dst, desc,  org: (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
                fontScale: 1.0,  color: 255,  thickness: 1, cv2.LINE_AA)

    cv2.imshow( winname: 'dst', dst)
    cv2.waitKey()

cv2.destroyAllWindows()
```

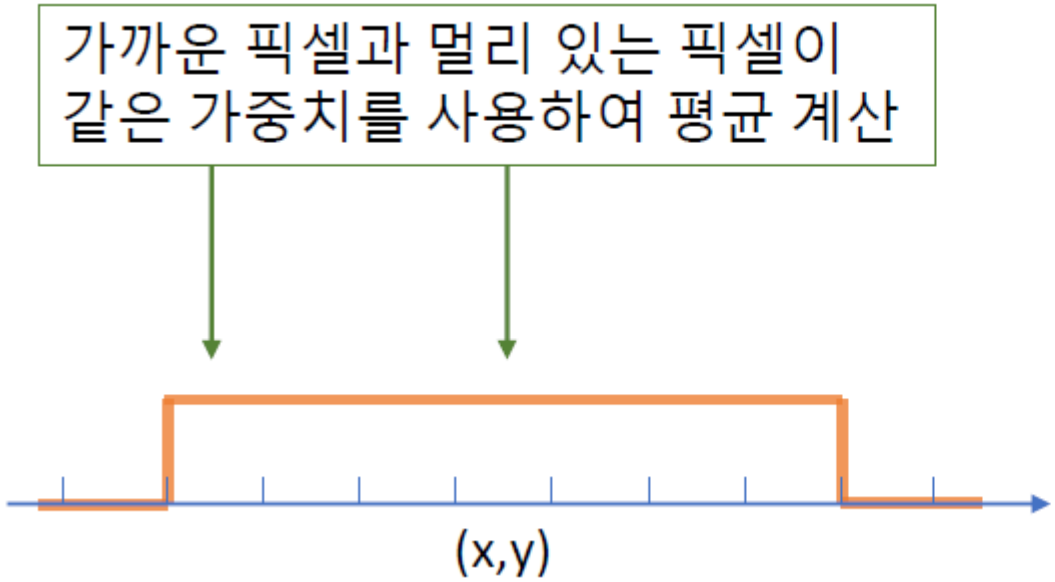


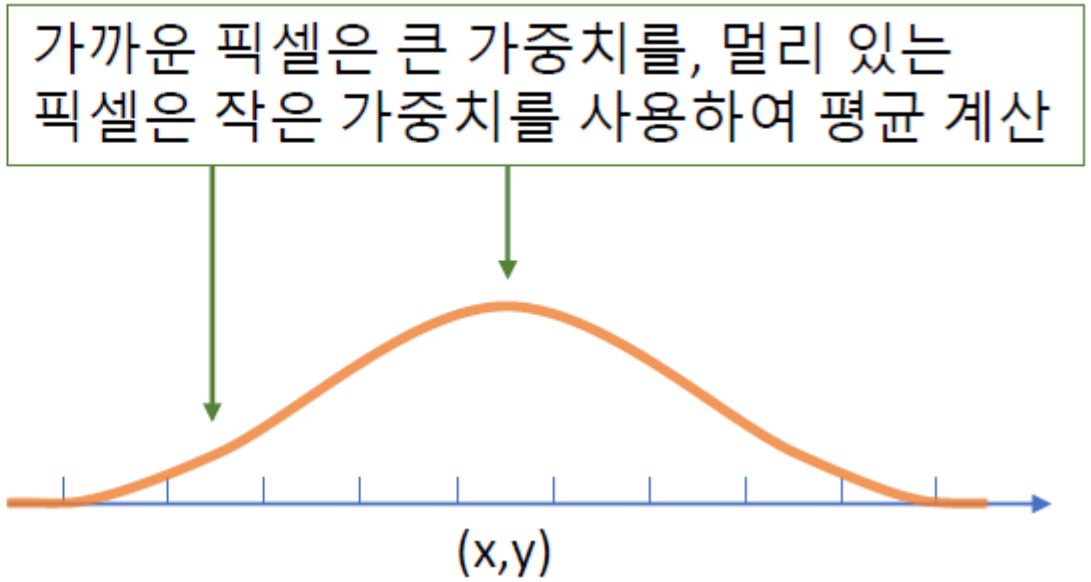


블러링 (2): 가우시안 필터

평균값 필터에 의한 블러링의 단점

- 필터링 대상 위치에서 가까이 있는 픽셀과 멀리 있는 픽셀이 모두 같은 가중치를 사용하여 평균을 계산
- 멀리 있는 픽셀의 영향을 많이 받을 수 있음



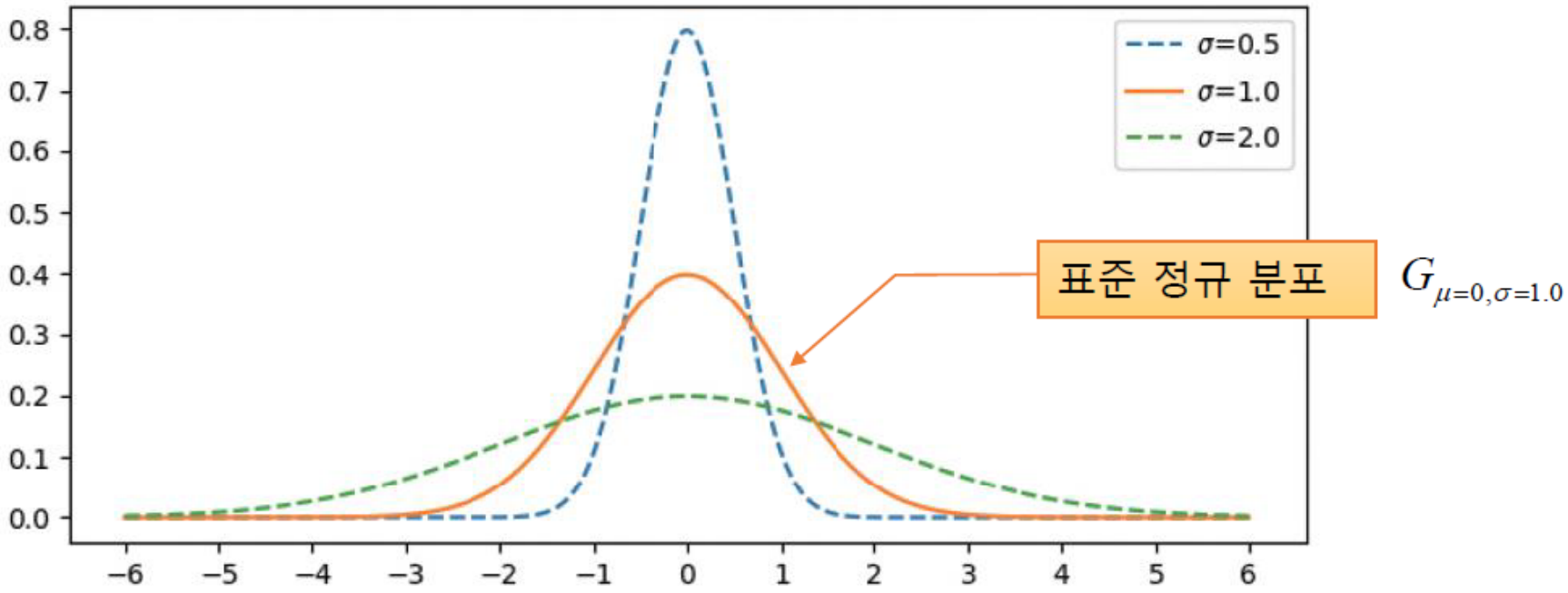


블러링 (2): 가우시안 필터

(1차원) 가우시안 함수 (Gaussian function)

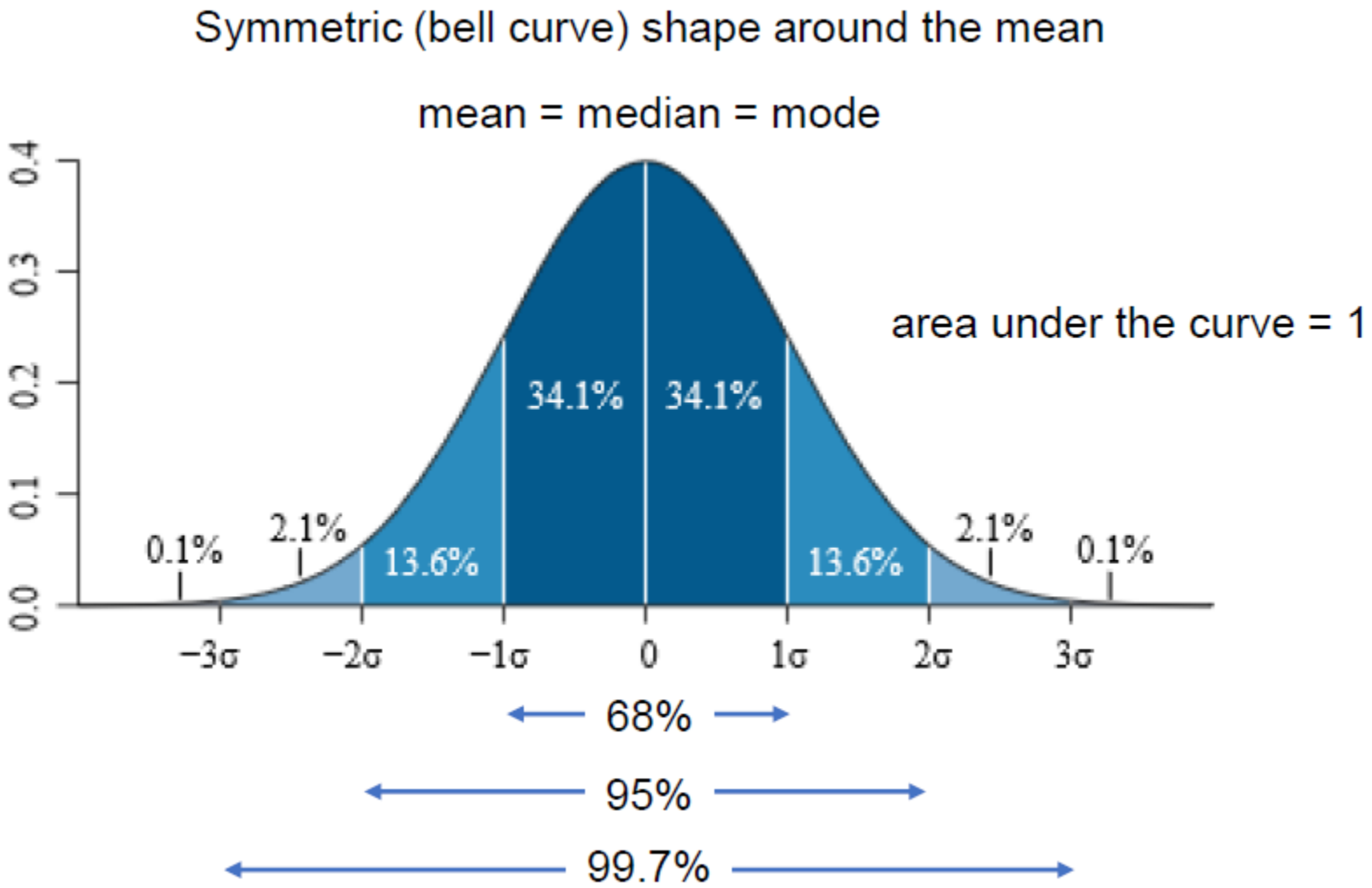
$$G_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- μ : 평균
- σ : 표준편차



블러링 (2): 가우시안 필터

가우시안 함수의 특징

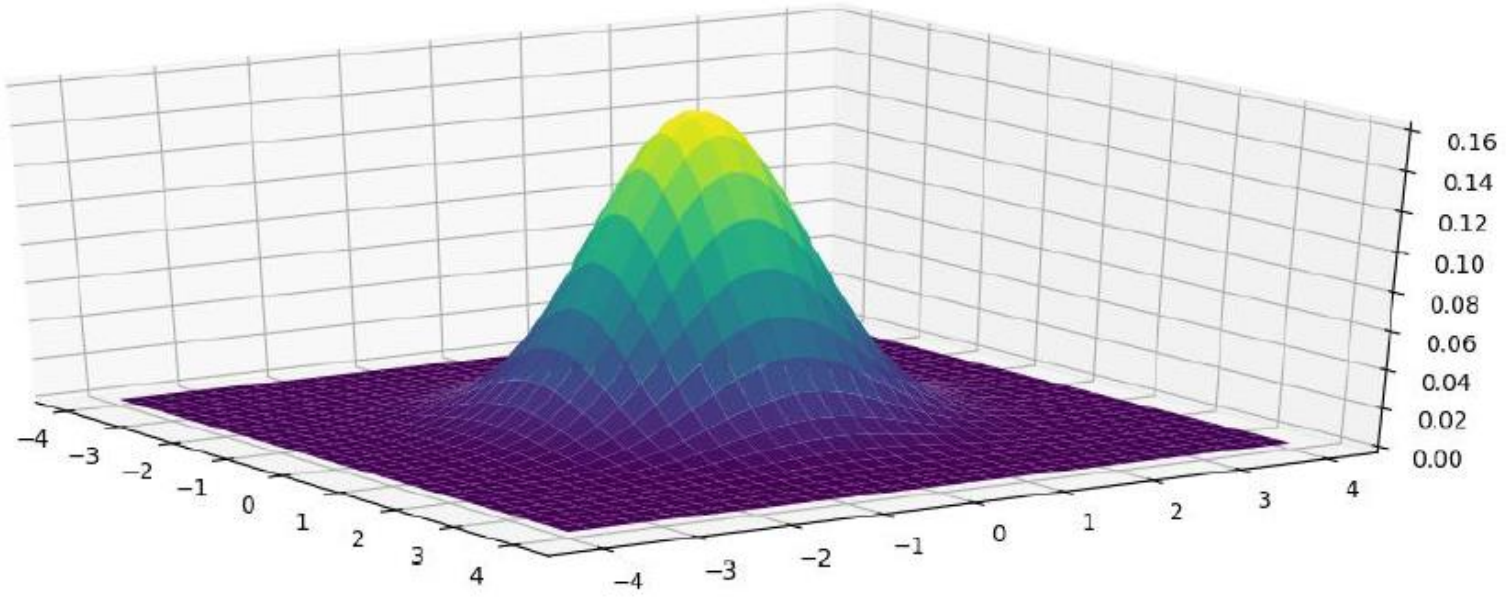


블러링 (2): 가우시안 필터

2차원 가우시안 함수 ($\mu_x = \mu_y = 0$, $\sigma_x = \sigma_y = \sigma$)

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

$$\begin{cases} \mu_x = \mu_y = 0 \\ \sigma_x = \sigma_y = \sigma \end{cases}$$



블러링 (2): 가우시안 필터

2차원 가우시안 필터 마스크 (σ = 1.0)

- 필터 마스크 크기 : $(8\sigma+1)$ 또는 $(6\sigma+1)$

4	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0002	0.0011	0.0018	0.0011	0.0002	0.0000	0.0000
2	0.0000	0.0002	0.0029	0. 0131	0.0215	0.0131	0.0029	0.0002	0.0000
1	0.0000	0.0011	0.0131	0.0585	0.0965	0.0585	0.0131	0.0011	0.0000
0	0.0001	0.0018	0.0215	0.0965	0.1592	0.0965	0.0215	0.0018	0.0001
-1	0.0000	0.0011	0.0131	0.0585	0.0965	0.0585	0.0131	0.0011	0.0000
-2	0.0000	0.0002	0.0029	0. 0131	0.0215	0.0131	0.0029	0.0002	0.0000
-3	0.0000	0.0000	0.0002	0.0011	0.0018	0.0011	0.0002	0.0000	0.0000
-4	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000
	-4	-3	-2	-1	0	1	2	3	4

블러링 (2): 가우시안 필터

가우시안 필터링 함수

cv2.GaussianBlur(src, ksize, sigmaX, dst=None, sigmaY=None, borderType=None) --> dst

- src: 입력 영상. 각 채널 별로 처리됨
- dst: 출력 영상. src와 같은 크기, 같은 타입
- ksize: 가우시안 커널 크기. (0, 0)을 지정하면 sigma 값에 의해 자동 결정됨
- sigmaX: x 방향 sigma.
- sigmaY: y 방향 sigma. 0이면 sigmaX와 같게 설정
- borderType: 가장자리 픽셀 확장 방식

블러링 (2): 가우시안 필터

다양한 크기의 sigma를 사용한 가우시안 필터링

```python
import sys
import numpy as np
import cv2

src = cv2.imread('rose.bmp', cv2.IMREAD_GRAYSCALE)

if src is None:
    print('Image load failed!')
    sys.exit()

cv2.imshow( winname: 'src', src)

for sigma in range(1, 6):
    # sigma 값을 이용하여 가우시안 필터링
    dst = cv2.GaussianBlur(src, ksize: (0, 0), sigma)

    desc = 'sigma = {}'.format(sigma)
    cv2.putText(dst, desc, org: (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
                fontScale: 1.0, color: 255, thickness: 1, cv2.LINE_AA)

    cv2.imshow( winname: 'dst', dst)
    cv2.waitKey()

cv2.destroyAllWindows()
```

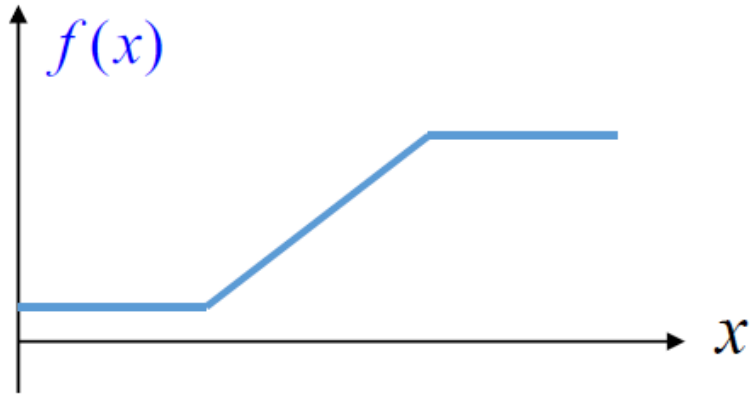
블러링 (2): 가우시안 필터

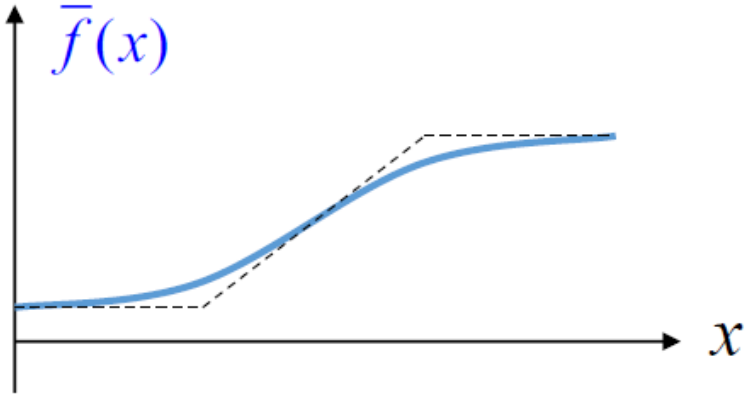
다양한 크기의 sigma를 사용한 가우시안 필터링 실행 결과

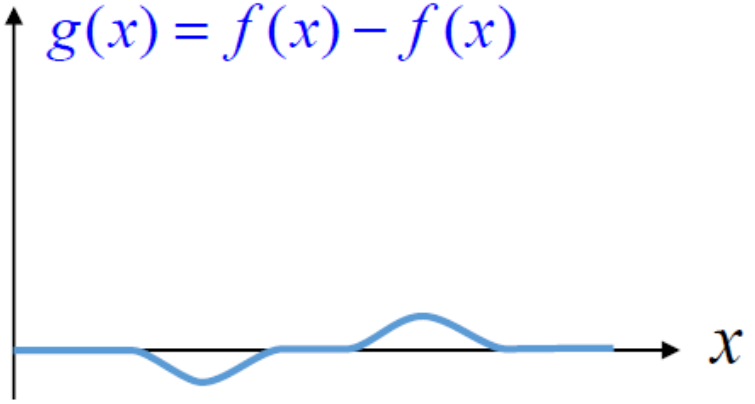
샤프닝: 언샤프 마스크 필터

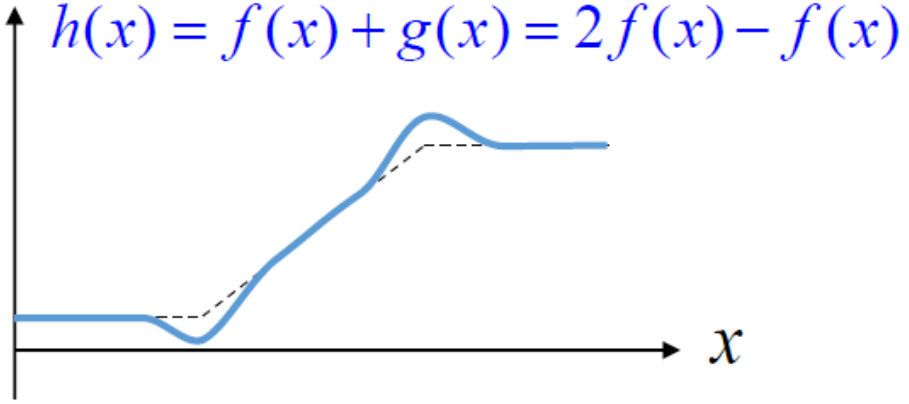
언샤프 마스크(Unsharp mask) 필터링

- 날카롭지 않은(unsharp) 영상, 즉, 부드러워진 영상을 이용하여 날카로운 영상을 생성





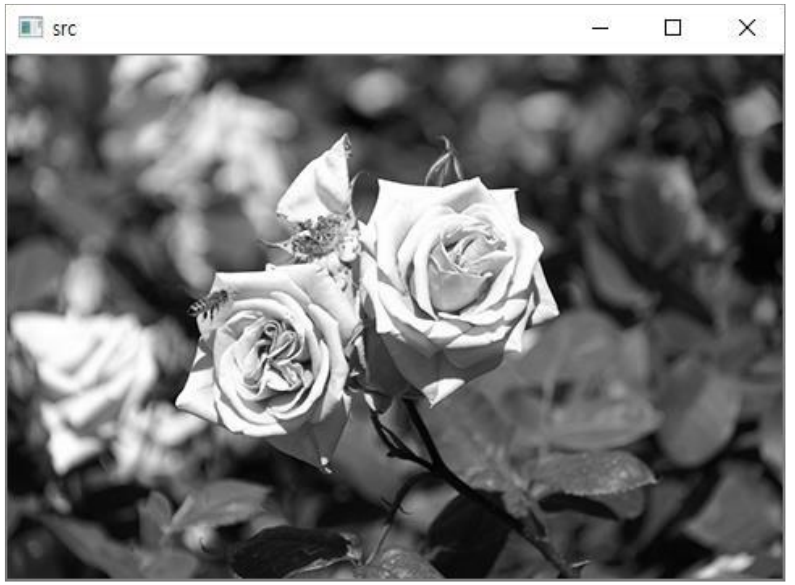


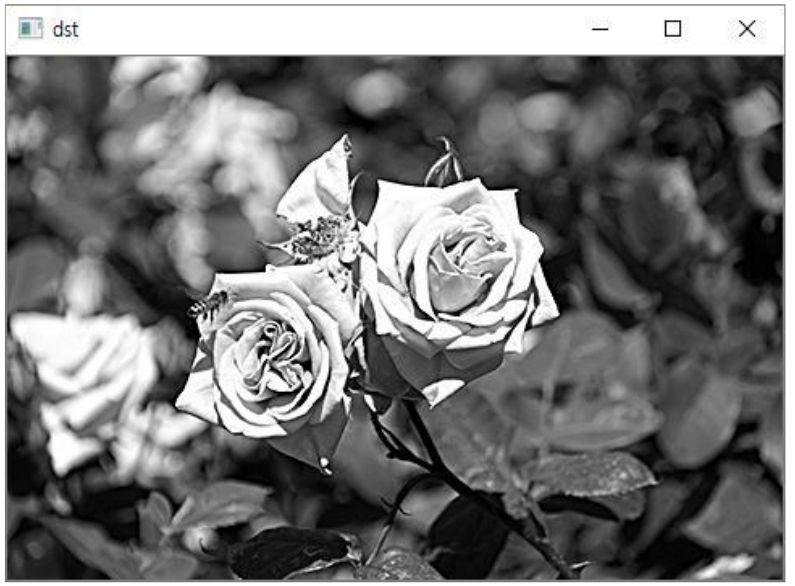


샤프닝: 언샤프 마스크 필터

언샤프 마스크 필터 구현하기

```python
import sys
import numpy as np
import cv2

src = cv2.imread('rose.bmp', cv2.IMREAD_GRAYSCALE)

if src is None:
    print('Image load failed!')
    sys.exit()

blr = cv2.GaussianBlur(src, ksize: (0, 0), sigmaX: 2)
dst = np.clip(2.0*src - blr, a_min: 0, a_max: 255).astype(np.uint8)

cv2.imshow( winname: 'src', src)
cv2.imshow( winname: 'dst', dst)
cv2.waitKey()

cv2.destroyAllWindows()
```





샤프닝: 언샤프 마스크 필터

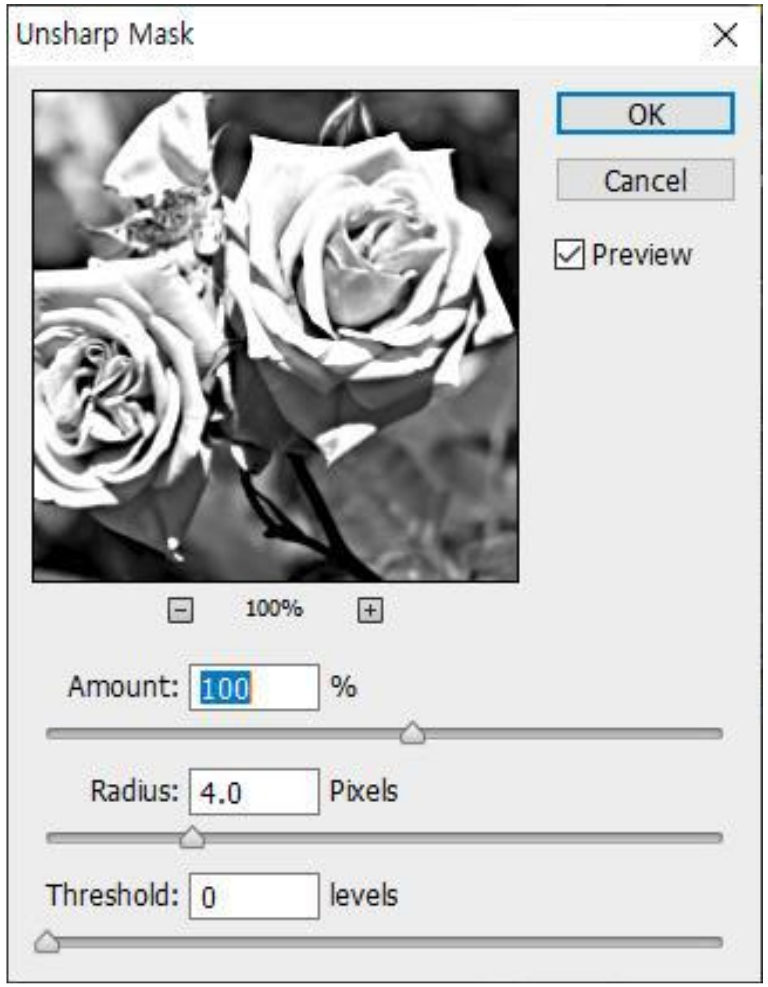
언샤프 마스크 필터 구현하기

- 샤프닝 정도를 조절할 수 있도록 수식 변경

$$h(x,y) = f(x,y) + \alpha \cdot g(x,y)$$

$$h(x,y) = f(x,y) + \alpha(f(x,y) - \bar{f}(x,y))$$
$$= (1+\alpha) f(x,y) - \alpha \cdot \bar{f}(x,y)$$

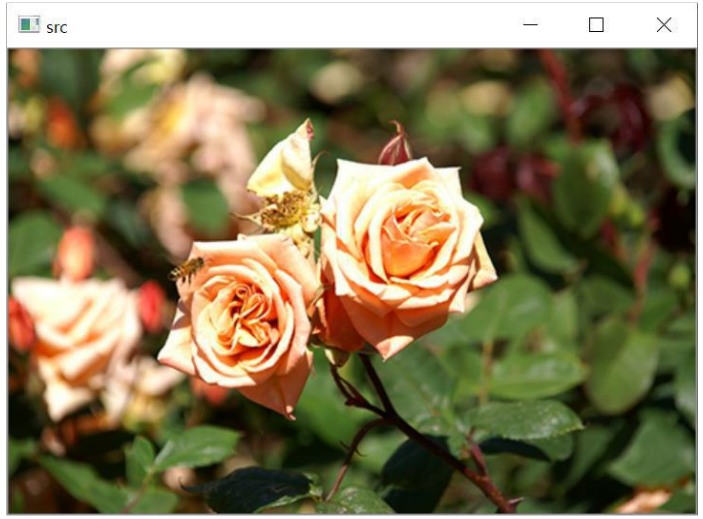
$$h(x,y) = (1+\alpha) f(x,y) - \alpha \cdot G_\sigma(f(x,y))$$

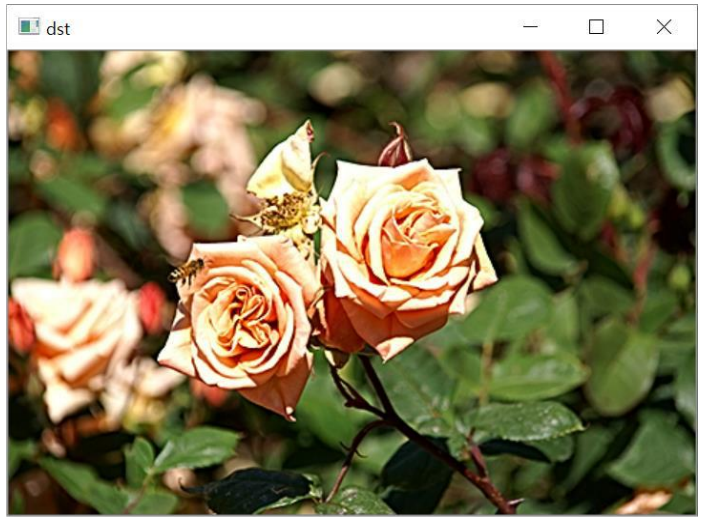


샤프닝: 언샤프 마스크 필터

컬러 영상에 대한 언샤프 마스크 필터 구현하기

```python
import sys
import numpy as np
import cv2

src = cv2.imread('rose.bmp')

if src is None:
    print('Image load failed!')
    sys.exit()

src_ycrcb = cv2.cvtColor(src, cv2.COLOR_BGR2YCrCb)

src_f = src_ycrcb[:, :, 0].astype(np.float32)
blr = cv2.GaussianBlur(src_f, ksize: (0, 0), sigmaX: 2.0)
src_ycrcb[:, :, 0] = np.clip(2. * src_f - blr, a_min: 0, a_max: 255).astype(np.uint8)

dst = cv2.cvtColor(src_ycrcb, cv2.COLOR_YCrCb2BGR)

cv2.imshow( winname: 'src', src)
cv2.imshow( winname: 'dst', dst)
cv2.waitKey()

cv2.destroyAllWindows()
```





영상의 잡음

영상의 잡음(Noise)

- 영상의 픽셀 값에 추가되는 원치 않는 형태의 신호

$$f(x,y) = s(x,y) + n(x,y)$$

획득된 영상 — 원본 신호 — 잡음

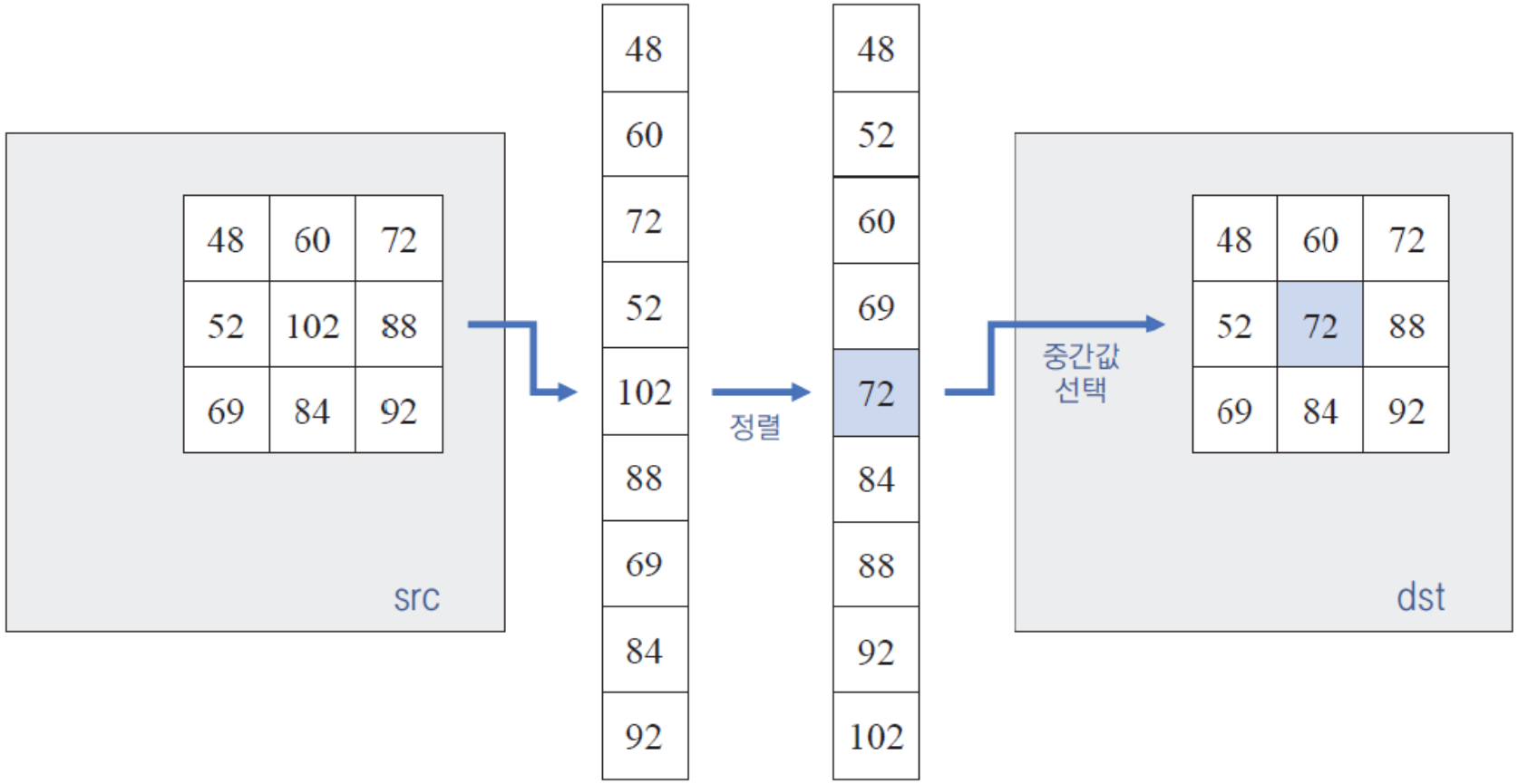
잡음의 종류

- 가우시안 잡음 (Gaussian noise)
- 소금&후추 잡음 (Salt&Pepper)

잡음 제거 (1): 미디언 필터

미디언 필터 (Median filter)

- 주변 픽셀들의 값들을 정렬하여 그 중앙값(median)으로 픽셀 값을 대체
- 소금 후추 잡음 제거에 효과적



잡음 제거 (1): 미디언 필터

미디언 필터링 함수

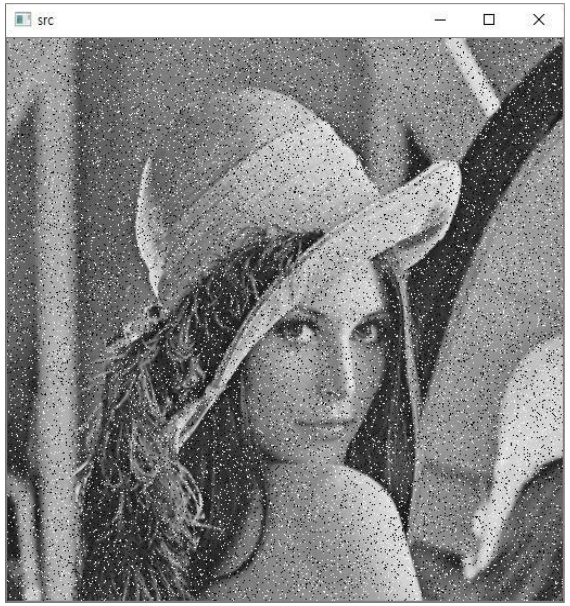
cv2.medianBlur(src, ksize, dst=None)) --> dst

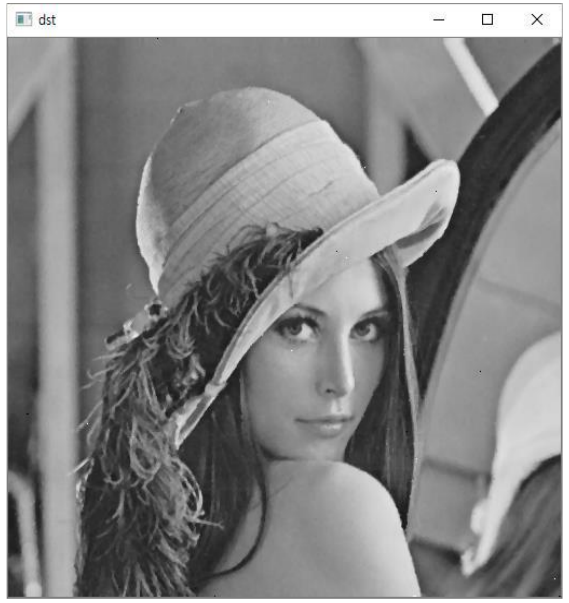
- src: 입력 영상. 각 채널 별로 처리됨
- ksize: 커널 크기. 1 보다 큰 홀수를 지정.
- dst: 출력 영상. src 와 같은 크기, 같은 타입

잡음 제거 (1): 미디언 필터

미디언 필터링 예제

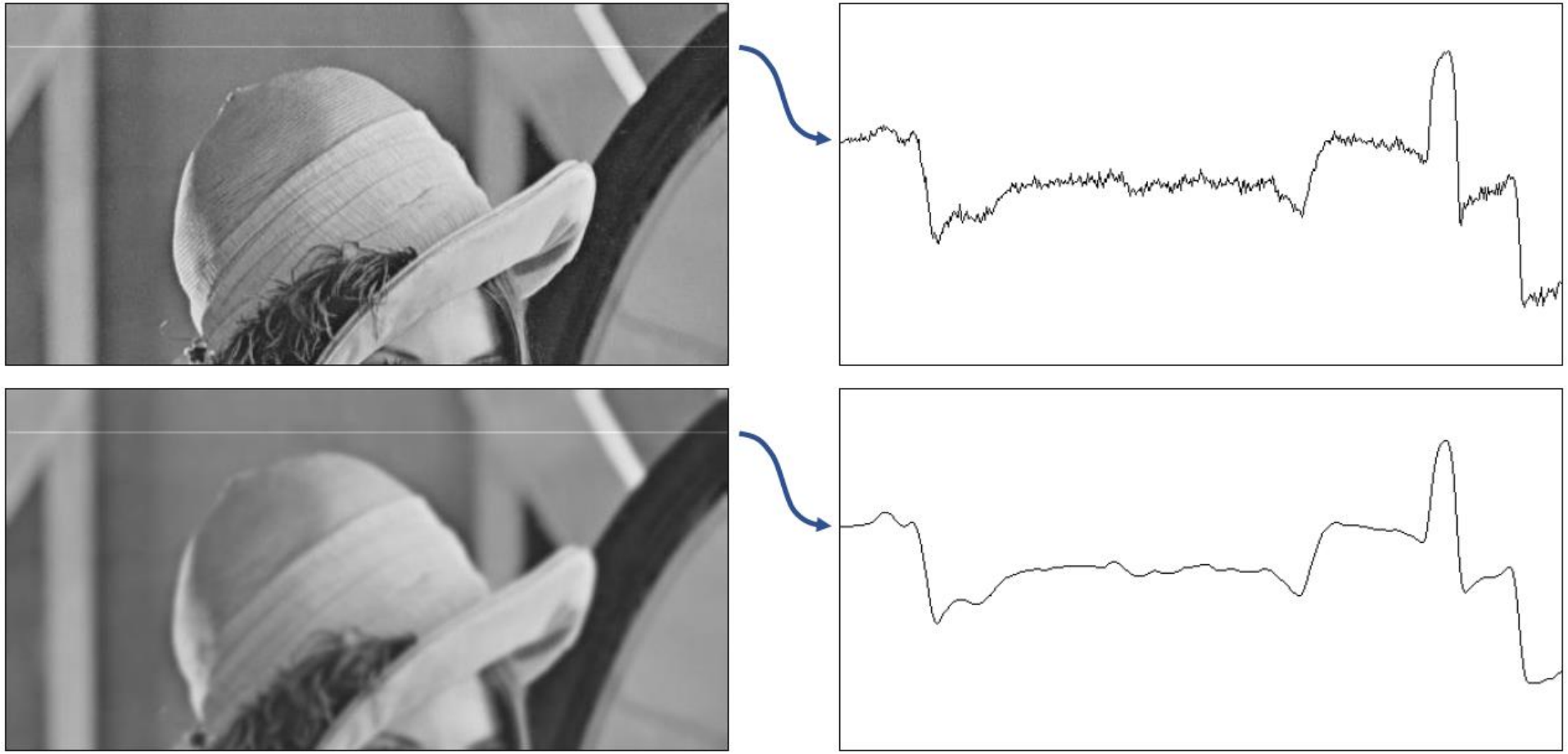
```python
import sys
import numpy as np
import cv2

src = cv2.imread('noise.bmp', cv2.IMREAD_GRAYSCALE)

if src is None:
    print('Image load failed!')
    sys.exit()

dst = cv2.medianBlur(src, ksize: 3)

cv2.imshow( winname: 'src', src)
cv2.imshow( winname: 'dst', dst)
cv2.waitKey()

cv2.destroyAllWindows()
```





가우시안 필터

- **가우시안 잡음 제거에는 가우시안 필터가 효과적**



잡음 제거 (2): 양방향 필터

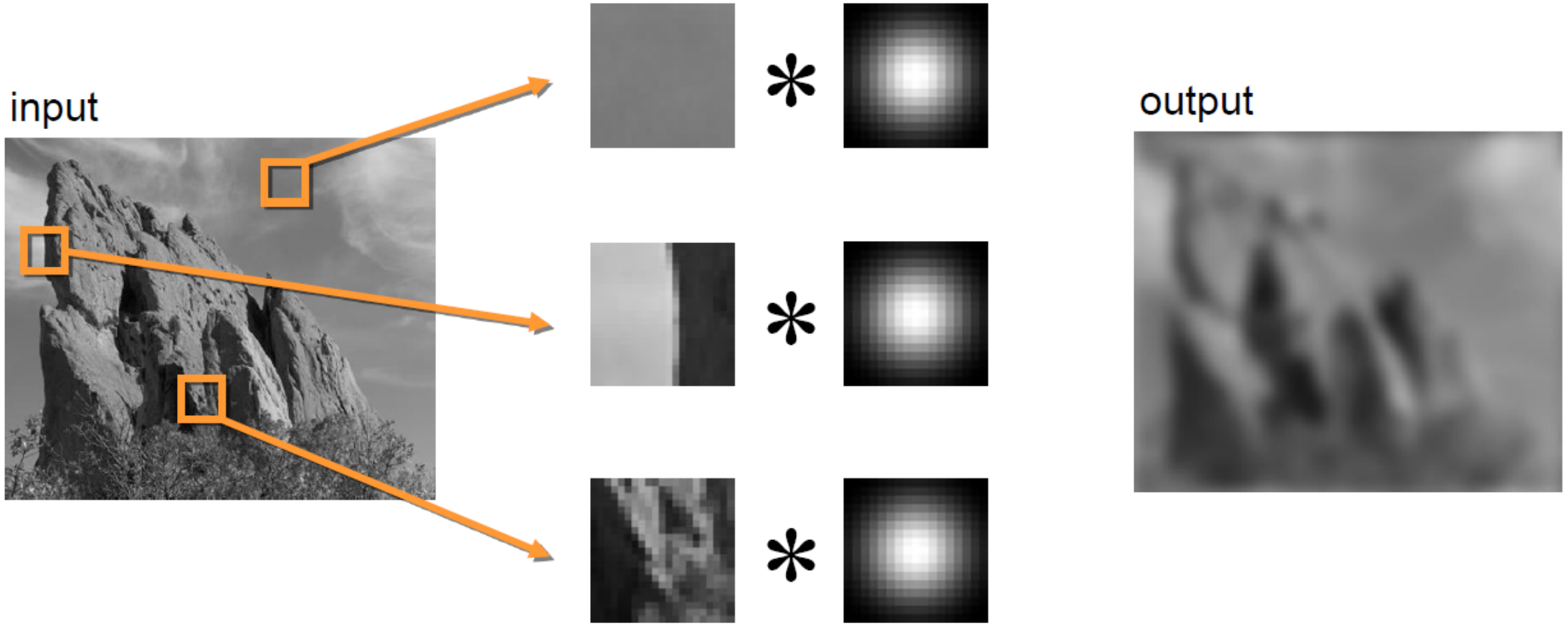
양방향 필터 (Bilateral filter)

- 에지 보전 잡음 제거 필터 (edge preserving noise removal filter) 의 하나
- 평균 값 필터 또는 가우시안 필터는 에지 부근에서도 픽셀 값을 평탄하게 만드는 단점이 있음
- 기준 픽셀과 이웃 픽셀과의 거리, 그리고 픽셀 값의 차이를 함께 고려하여 블러링 정도를 조절

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|) G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |) I_{\mathbf{q}}$$

잡음 제거 (2): 양방향 필터

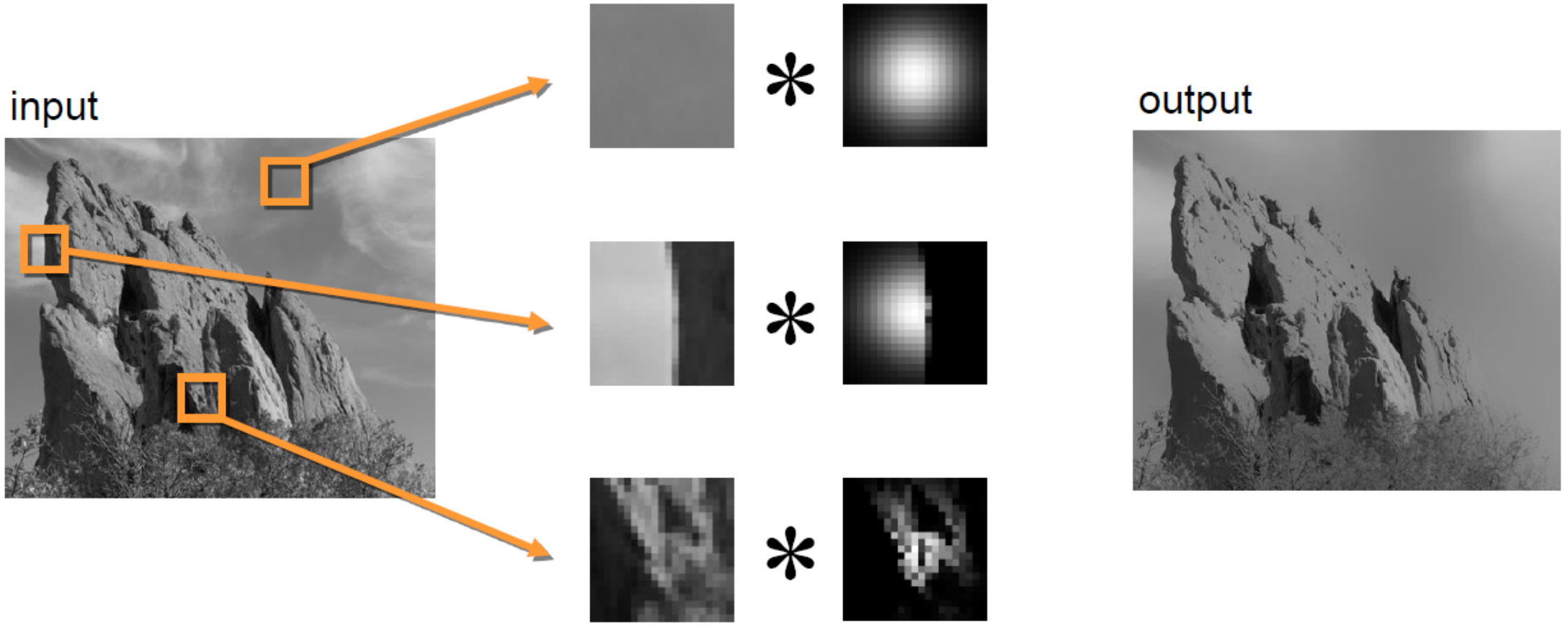
- **(일반적인) 가우시안 필터링 : 영상 전체에서 blurring**



Same Gaussian kernel everywhere.

잡음 제거 (2): 양방향 필터

- **양방향 필터 : 에지가 아닌 부분에서만 blurring**



The kernel shape depends on the image content.

잡음 제거 (2): 양방향 필터

양방향 필터링 함수

cv2.bilateralFilter(src, d, sigmaColor, sigmaSpace, dst=None, borderType=None)) --> dst

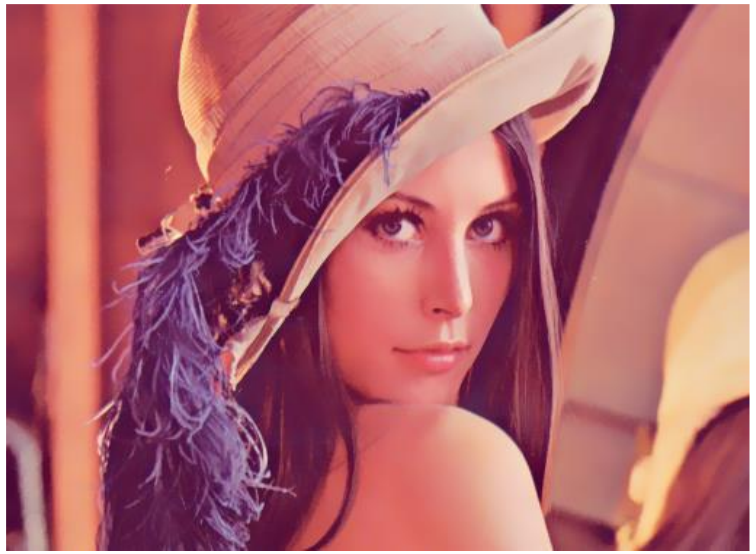
- src : 입력 영상. 8 비트 또는 실수형, 1 채널 또는 3 채널
- d: 필터링에 사용될 이웃 픽셀의 거리(지름).

  음수(-1)를 입력하면 sigmaSpace 값에 의해 자동 결정됨
- sigmaColor: 색 공간에서 필터의 표준 편차
- sigmaSpace: 좌표 공간에서 필터의 표준 편차
- dst: 출력 영상. src 와 같은 크기, 같은 타입
- borderType: 가장자리 픽셀 처리 방식

잡음 제거 (2): 양방향 필터

양방향 필터링 예제

```python
import sys
import numpy as np
import cv2

src = cv2.imread('lenna.bmp')

if src is None:
    print('Image load failed!')
    sys.exit()

dst = cv2.bilateralFilter(src, -1, sigmaColor: 10, sigmaSpace: 5)

cv2.imshow( winname: 'src', src)
cv2.imshow( winname: 'dst', dst)
cv2.waitKey()

cv2.destroyAllWindows()
```



[프로젝트] 카툰 필터 카메라

카툰 필터 카메라

- 카메라 입력 영상에 실시간으로 재미있는 필터링을 적용하는 기능

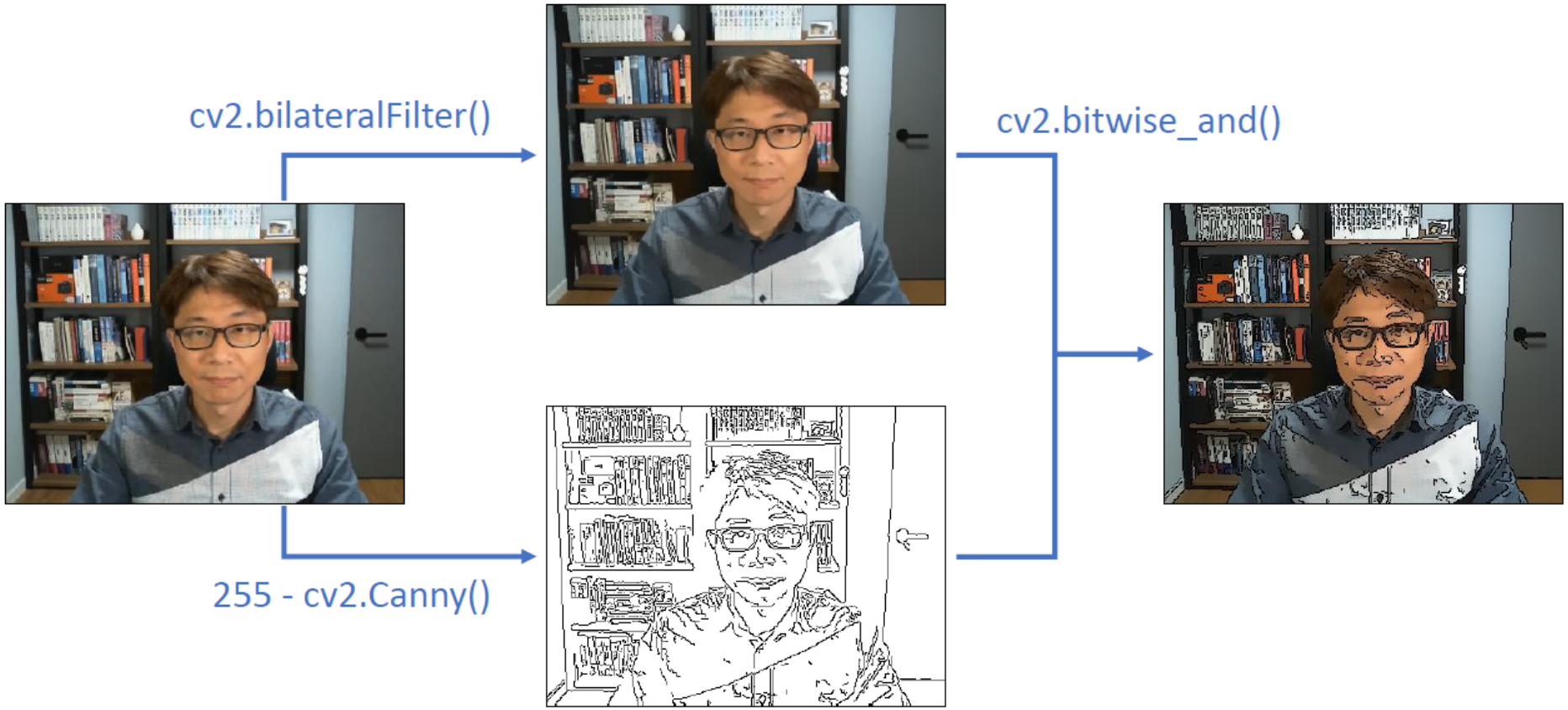
구현할 기능

- 카툰 필터
- 스케치 필터
- 스페이스바를 누를 때마다 모드 변경

[프로젝트] 카툰 필터 카메라

카툰 필터

- 입력 영상의 색상을 단순화시키고, 에지 부분을 검정색으로 강조



[프로젝트] 카툰 필터 카메라

카툰 필터

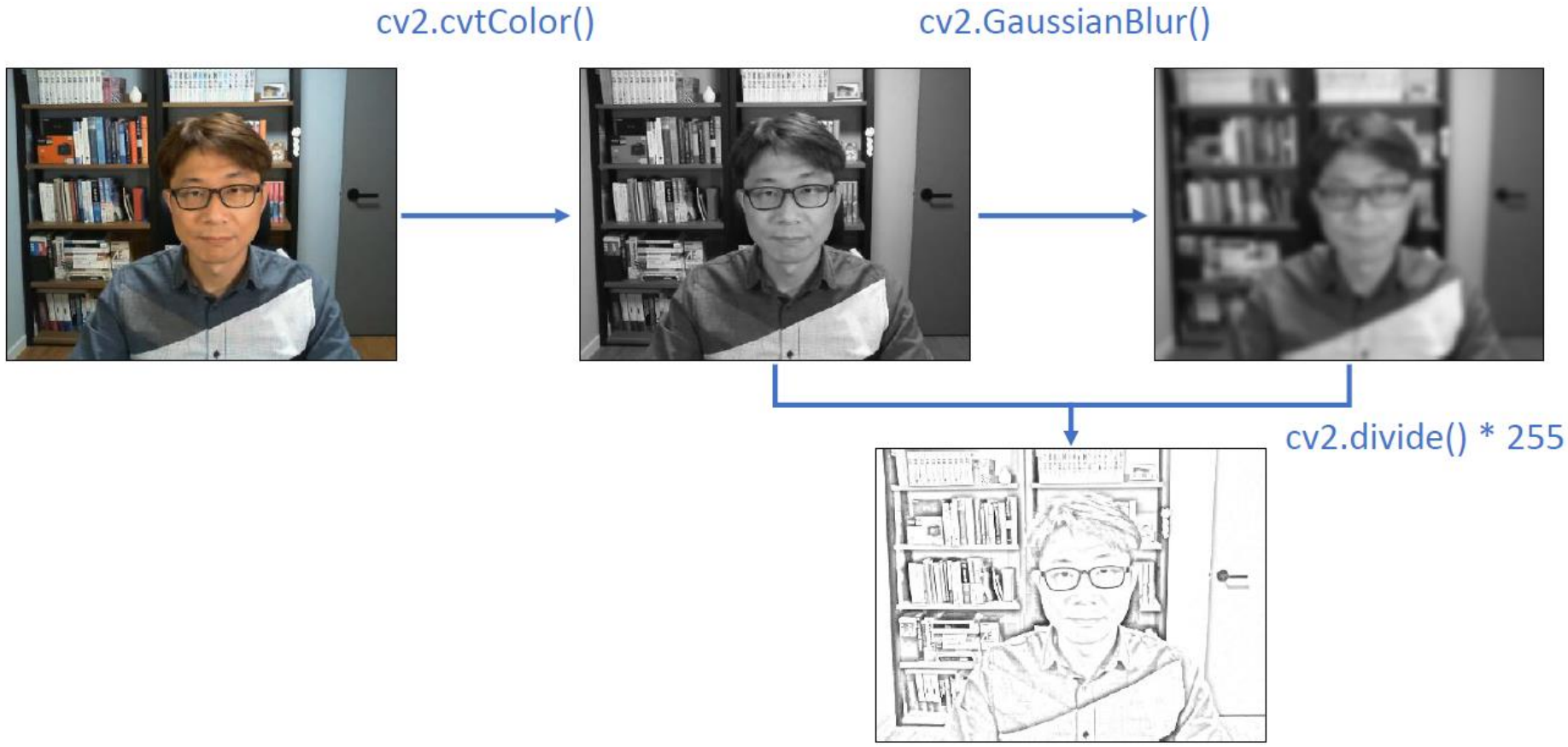
- 입력 영상의 색상을 단순화시키고, 에지 부분을 검정색으로 강조

```python
# 카툰 필터 카메라

import sys
import numpy as np
import cv2


def cartoon_filter(img):
    h, w = img.shape[:2]
    img2 = cv2.resize(img, dsize: (w//2, h//2))

    blr = cv2.bilateralFilter(img2, -1, sigmaColor: 20, sigmaSpace: 7)
    edge = 255 - cv2.Canny(img2, 80, 120)
    edge = cv2.cvtColor(edge, cv2.COLOR_GRAY2BGR)

    dst = cv2.bitwise_and(blr, edge)
    dst = cv2.resize(dst, dsize: (w, h), interpolation=cv2.INTER_NEAREST)

    return dst

```

[프로젝트] 카툰 필터 카메라

스케치 필터

- 평탄한 영역은 흰색
- 에지 근방에서 어두운 영역을 검정색으로 설정 . 밝은 영역은 흰색



[프로젝트] 카툰 필터 카메라

스케치 필터

- 평탄한 영역은 흰색
- 에지 근방에서 어두운 영역을 검정색으로 설정 . 밝은 영역은 흰색

```
22      def pencil_sketch(img):
23          gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
24          blr = cv2.GaussianBlur(gray, ksize: (0, 0), sigmaX: 3)
25          dst = cv2.divide(gray, blr, scale=255)
26          return dst
27
28
29      cap = cv2.VideoCapture(0)
30
31      if not cap.isOpened():
32          print('video open failed!')
33          sys.exit()
34
35      cam_mode = 0
36
```

[프로젝트] 카툰 필터 카메라

- 스케치 필터

```
while True:
    ret, frame = cap.read()

    if not ret:
        break

    if cam_mode == 1:
        frame = cartoon_filter(frame)
    elif cam_mode == 2:
        frame = pencil_sketch(frame)
        frame = cv2.cvtColor(frame, cv2.COLOR_GRAY2BGR)

    cv2.imshow( winname: 'frame', frame)
    key = cv2.waitKey(1)

    if key == 27:
        break
    elif key == ord(' '):
        cam_mode += 1
        if cam_mode == 3:
            cam_mode = 0


cap.release()
cv2.destroyAllWindows()
```