OpenCV로 배우는 컴퓨터 비전
7.2.1~7.4.3

2017010715허지혜

7.2 Blurring

A technique to soften a video like a picture out of focus
Also called "smoothing"

When we use ..

1. Smoothing out rough input images
2. Pre-processing to eliminate the effect of noise

7.2.1. Averaging Blurring

One type of blur filter, **Averaging Filter**

Set the arithmetic mean of a specific pixel and surrounding pixels as the resulting image pixel value

The effect of breaking sharp edges and eliminating the effect of noise

It is difficult to distinguish objects when used excessively

blur(InputArray src, OutputArray dst, Size ksize,
Point anchor = Point(-1,-1), int borderType = BORDER_DEFAULT)

지정값	설명
src1	입력 영상, 다채널 영상은 각 채널별로 블러링 수행, 입력 영상의 깊이는 CV_8U, CV_16U, CV_16S, CV_32F, CV_64F 중 하나여야 한다.
dst	출력 영상, src와 같은 크기, 같은 채널 수를 갖는다.
ksize	블러링 커널 크기
anchor	고정점 좌표, Point(-1,-1)을 지정하면 커널 중심을 고정점으로 사용한다.
borderType	가장자리 픽셀 확장 방식

7.2.1. Averaging Filter

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

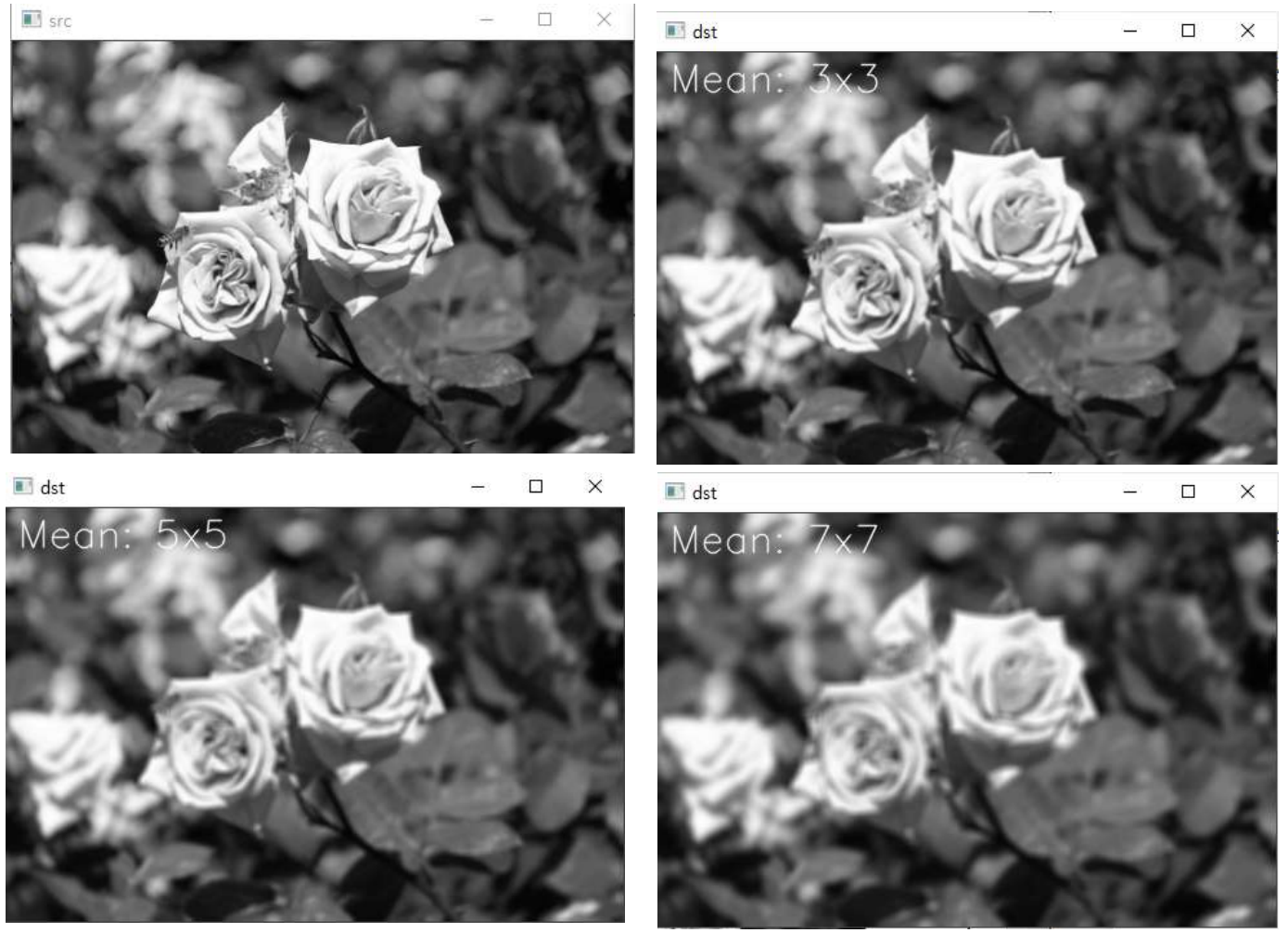
$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{25} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Mask size Up → Softening

7.2.1. Averaging Filter

```python
import numpy as np
import cv2


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

    if src is None:
        print('Image load failed!')
        return

    cv2.imshow('src', src)

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

        desc = "Mean: %dx%d" % (ksize, ksize)
        cv2.putText(dst, desc, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
                    1.0, 255, 1, cv2.LINE_AA)

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

    cv2.destroyAllWindows()
```

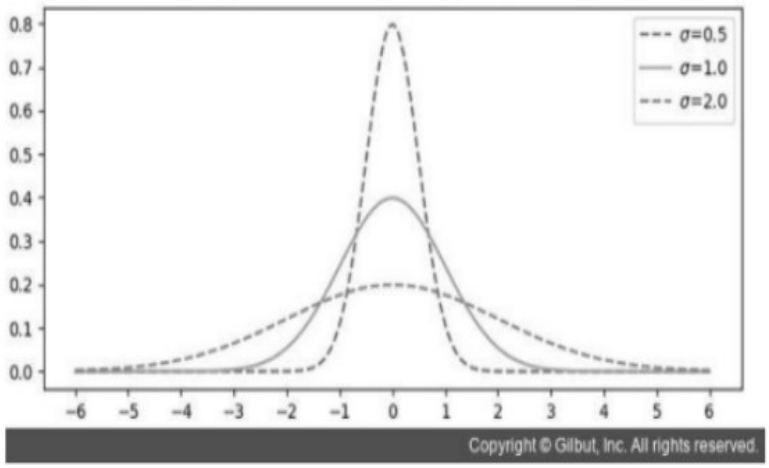
7.2.1. Averaging Filter

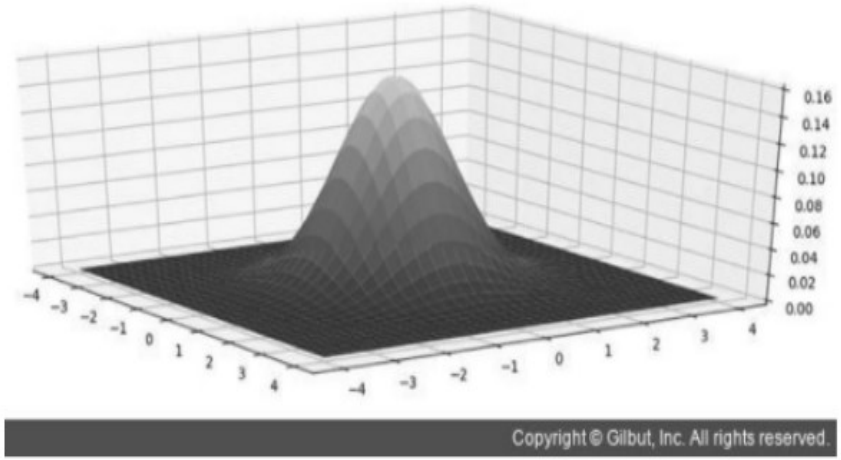


7.2.2. Gausian Filter

One type of blur filter, **GaussianFilter**

It is generated by approximating the Gaussian distribution function





a maximum value : 0
volume is 1

a maximum value : (0,0)
volume is 1

7.2.2. Gausian Filter

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

$$= \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{x^2}{2\sigma_x^2}} \times \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{y^2}{2\sigma_y^2}} = G_{\sigma_x}(x)\cdot G_{\sigma_y}(y)$$

The size of the filter mask
= (8*(standard deviation) + 1)

However, CV_8U depth image
= (6*(standard deviation)+1)

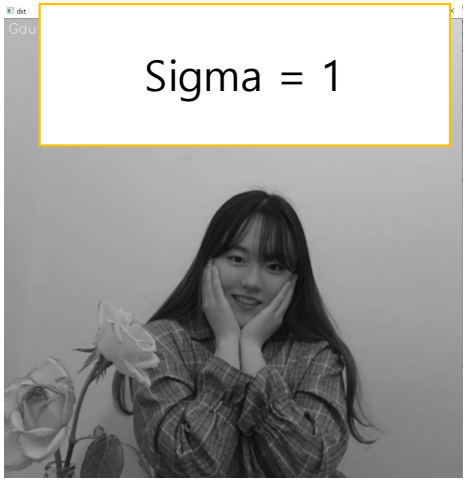
GaussianBlur(InputArray src, OutputArray dst, Size ksize,
double sigmaX, double sigmaY = 0,int borderType = BORDER_DEFAULT)

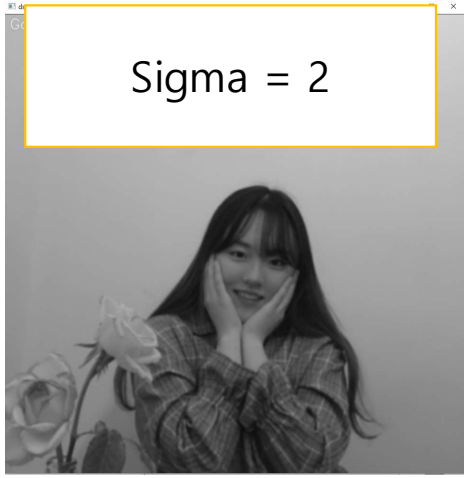
지정값	설명
src	입력 영상, 다채널 영상은 각 채널별로 블러링 수행
dst	출력 영상, src와 같은 크기, 같은 타입을 갖는다.
ksize	블러링 커널 크기,ksize에 Size()를 지정하면 표준 편차로부터 커널 크기를 자동으로 결정한다.
sigmaX	x 방향으로의 가우시안 커널 표준 편차
sigmaY	y 방향으로의 가우시안 커널 표준 편차
borderType	가장자리 픽셀 확장 방식

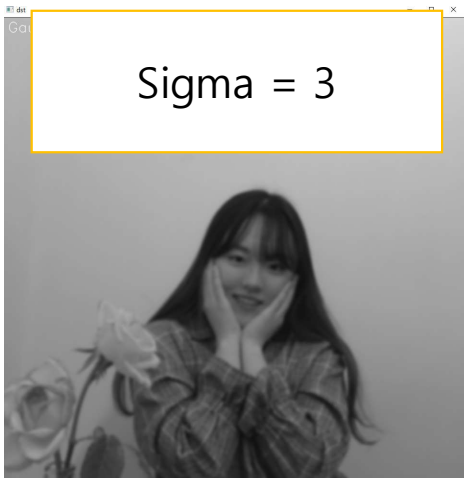
7.2.2. Gausian Filter

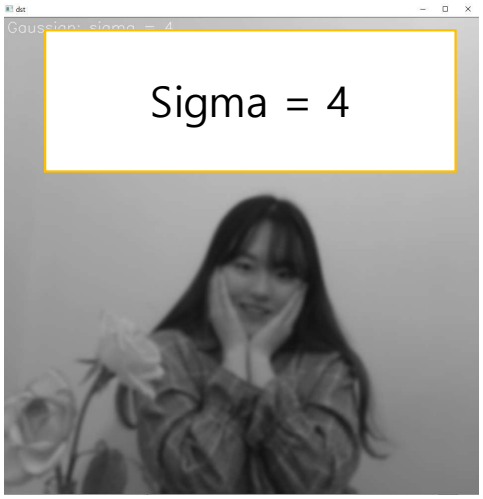
```
def blurring_gaussian():
    src = cv2.imread('rose.bmp', cv2.IMREAD_GRAYSCALE)

    if src is None:
        print('Image load failed!')
        return

    cv2.imshow('src', src)

    for sigma in range(1, 6):
        dst = cv2.GaussianBlur(src, (0, 0), sigma)

        desc = "Gaussian: sigma = %d" % (sigma)
        cv2.putText(dst, desc, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
                    1.0, 255, 1, cv2.LINE_AA)

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

    cv2.destroyAllWindows()


if __name__ == '__main__':
    blurring_mean()
    blurring_gaussian()
```

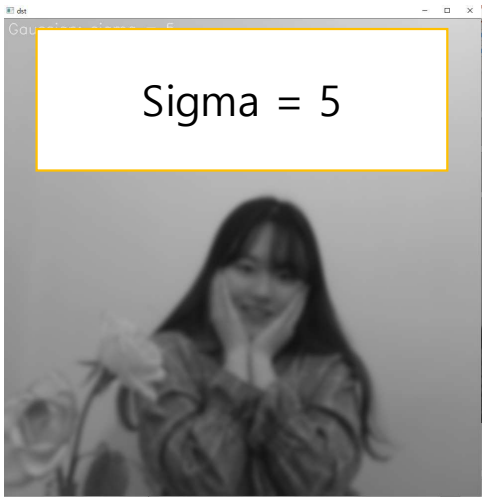
7.2.2. Gausian Filter











7.3. Sharpening

It is a filtering technique that changes the image
to give a clear feeling

7.3.1 Unsharp Mask Filter

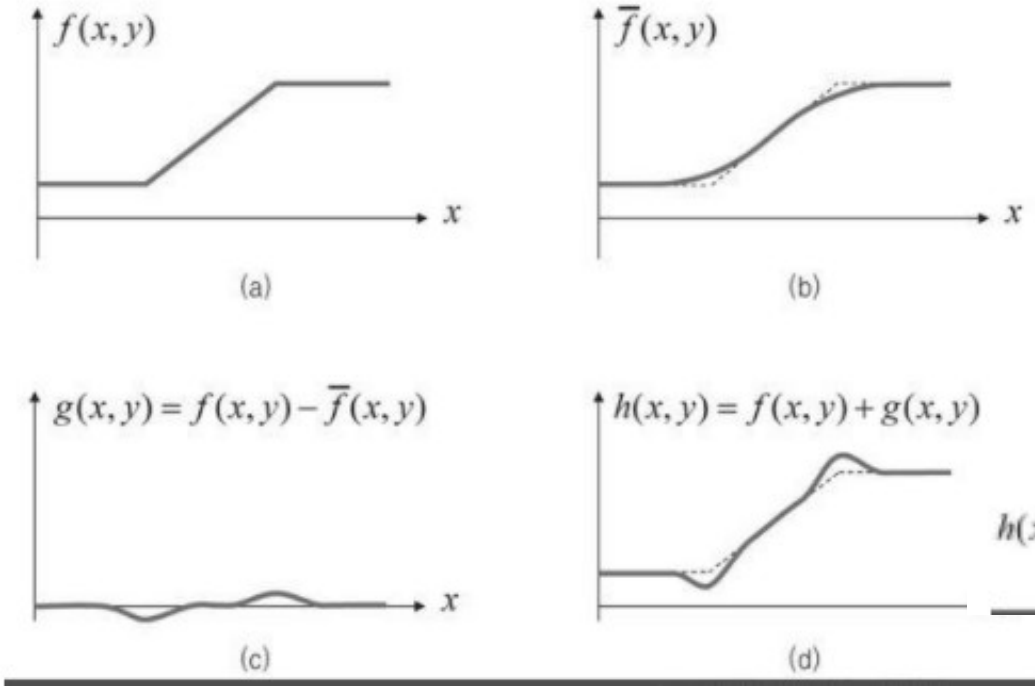
Blurred images are used to implement sharpening

A filter that creates a sharp image inversely using an unsharp image is called an **unsharp mask filter**

7.3.1 Unsharp Mask Filter

Input video

Blurring video



Adjust the sharpness with weights

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

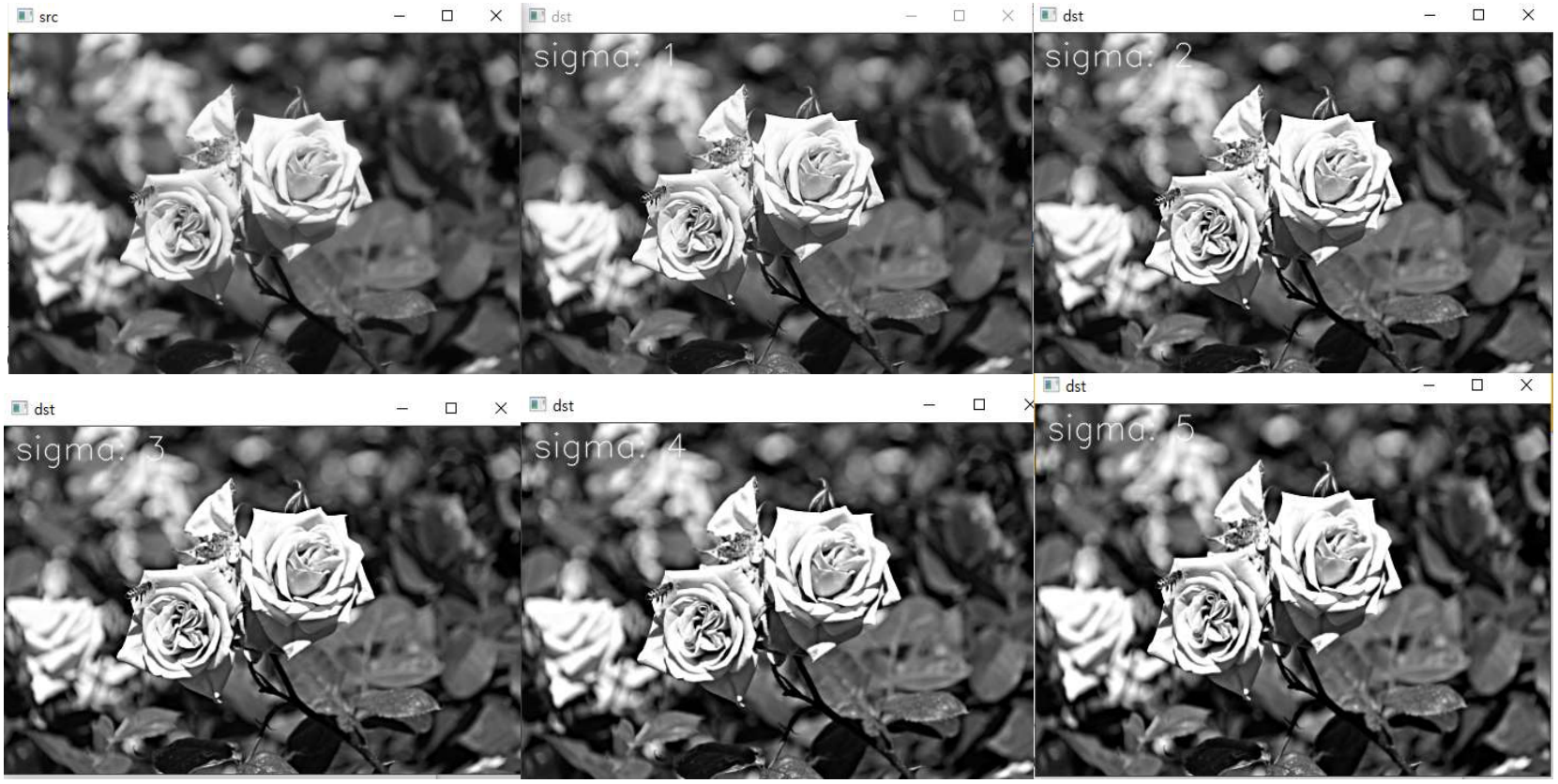
Input video – Blurring video = Sharp part = g(x,y)

Input video + Sharp part = h(x,y)

7.3.1 Unsharp Mask Filter

```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('src', src)

for sigma in range(1, 6):
    blurred = cv2.GaussianBlur(src, (0, 0), sigma)

    alpha = 1.0
    dst = cv2.addWeighted(src, 1 + alpha, blurred, -alpha, 0.0)

    desc = "sigma: %d" % sigma
    cv2.putText(dst, desc, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
                1.0, 255, 1, cv2.LINE_AA)

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

cv2.destroyAllWindows()
```

7.3.1 Unsharp Mask Filter



7.4 Noise Reduction Filter

One of the preprocessing steps is noise reduction filtering.

7.4.1 Image And Noise Model

The way noise is generated is called "**noise model**"

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

S(x,y) : Scene looking at the camera lens (original)
n(x,y) : noise

randn(InputOutputArray dst, InputArray mean, InputArray stddev)

지정값	설명
dst	가우시안 난수로 채워질 행렬, dst 행렬은 미리 할당 되어야 한다.
mean	가우시안 분포 평균
stddev	가우시안 분포 표준 편차

7.4.1 Image And Noise Model

```python
import numpy as np
import cv2
import random


def noise_gaussian():
    src = cv2.imread('lenna.bmp', cv2.IMREAD_GRAYSCALE)

    if src is None:
        print('Image load failed!')
        return

    cv2.imshow('src', src)

    for stddev in [10, 20, 30]:
        noise = np.zeros(src.shape, np.int32) # 부호 있는 정수형을 사용
        cv2.randn(noise, 0, stddev)

        dst = cv2.add(src, noise, dtype=cv2.CV_8UC1) # 영상 깊이는 cv_8uc1

        desc = 'stddev = %d' % stddev
        cv2.putText(dst, desc, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
                    1.0, 255, 1, cv2.LINE_AA)
        cv2.imshow('dst', dst)
        cv2.waitKey()

    cv2.destroyAllWindows()

if __name__ == '__main__':
    noise_gaussian()
```

7.4.1 Image And Noise Model

7.4.2. Bidirectional filter

Disadvantages of Gaussian Filter

The effect of reducing not only the noise but also the edge component immediately at the edge where the pixel value changes rapidly.

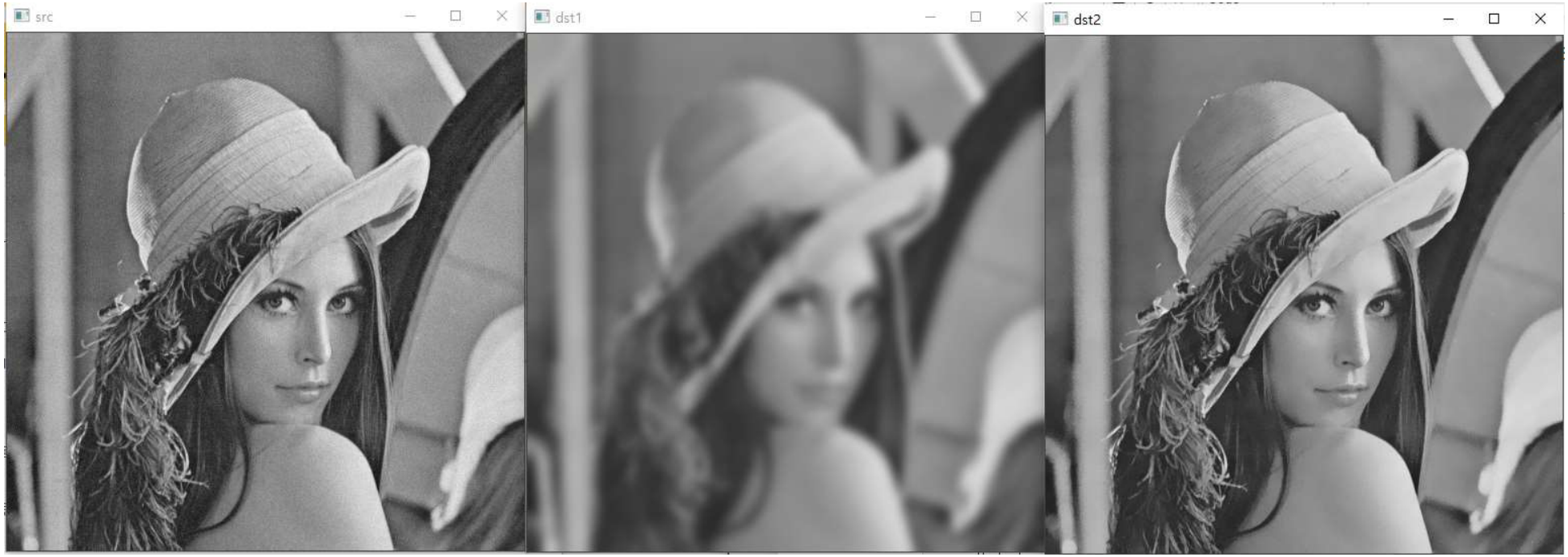
Bidirectional filter

Algorithm to remove only noise while maintaining edge information

7.4.2. Bidirectional filter

```python
def filter_bilateral():
    src = cv2.imread('lenna.bmp', cv2.IMREAD_GRAYSCALE)

    if src is None:
        print('Image load failed!')
        return

    noise = np.zeros(src.shape, np.int32)
    cv2.randn(noise, 0, 5)
    cv2.add(src, noise, src, dtype=cv2.CV_8UC1)

    dst1 = cv2.GaussianBlur(src, (0, 0), 5)
    dst2 = cv2.bilateralFilter(src, -1, 10, 5)

    cv2.imshow('src', src)
    cv2.imshow('dst1', dst1)
    cv2.imshow('dst2', dst2)
    cv2.waitKey()
    cv2.destroyAllWindows()

if __name__ == '__main__':
    filter_bilateral()
```

7.4.2. Bidirectional filter



Noise model pictures

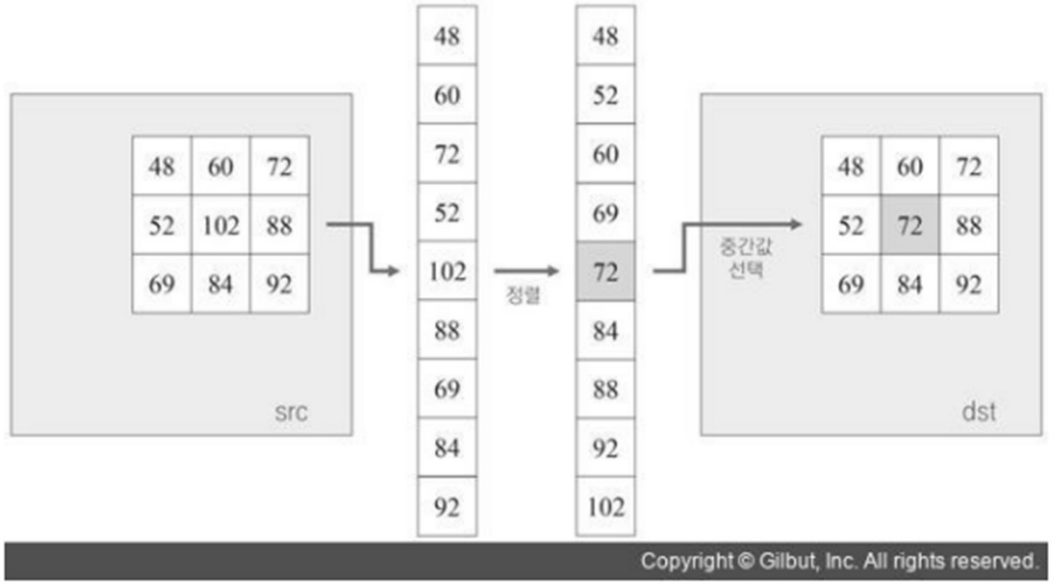
Gaussian filter pictures

Bidirectional filter pictures

7.4.3. Median filter

In the input image, select the median value among the own pixel values

Works effectively when the noise pixel value has a large difference from the surrounding pixel value.



7.4.3. Median filter

```python
def filter_median():
    src = cv2.imread('lenna.bmp', cv2.IMREAD_GRAYSCALE)

    if src is None:
        print('Image load failed!')
        return

    # 소금,후추 잡음을 추가
    for i in range(0, int(src.size / 10)):
        x = random.randint(0, src.shape[1] - 1)
        y = random.randint(0, src.shape[0] - 1)
        src[x, y] = (i % 2) * 255

    # 가우시안과 미디안으로 비교해보기
    dst1 = cv2.GaussianBlur(src, (0, 0), 1)
    dst2 = cv2.medianBlur(src, 3)

    cv2.imshow('src', src)
    cv2.imshow('dst1', dst1)
    cv2.imshow('dst2', dst2)
    cv2.waitKey()
    cv2.destroyAllWindows()


if __name__ == '__main__':
    filter_median()
```

7.4.3. Median filter

Pepper salt pictures

Gaussian filter pictures

Median filter pictures

END