2020년 컴퓨터그래픽스

- HW 02 -

제출일자	2020.09.28.
이 름	장수훈
학 번	201402414
분 반	00

구현코드

```python
def my_filtering(src, filter):
    (h, w) = src.shape
    (f_h, f_w) = filter.shape
    pad_img = my_padding(src, filter)
    dst = np.zeros((h, w))

    ##################################################
    # TODO                                           #
    # filtering 구현                                 #
    # 4중 for문을 이용해 구현할것!                   #
    ##################################################
    for i in range(h):
        for j in range(w):
            temp = 0
            for k in range(f_h):
                for l in range(f_w):
                    temp = temp + (pad_img[i + k, j + l] * filter[k,l])
            dst[i,j] = temp

    return dst
```

필터링을 하기 위해 필터링이 이미지 크기 배열에 filter 인자로 받은거 크기만큼 배열을 곱하여 가운데 값을 구하게 코딩을 하였다.

```python
if fshape[0] and fshape[1] == 3:
    filter = np.array([[1/9, 1/9, 1/9],
                       [1/9, 1/9, 1/9],
                       [1/9, 1/9, 1/9]])

elif fshape[0] == 1 and fshape[1] == 3:
    filter = np.array([[1/3, 1/3, 1/3]])

elif fshape[0] == 3 and fshape[1] == 1:
    filter = np.array([[1 / 3],
                       [1 / 3],
                       [1 / 3]])

if fshape[0] and fshape[1] == 7:
    filter = np.array([[1/49, 1/49, 1/49, 1/49, 1/49, 1/49, 1/49],
                       [1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49],
                       [1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49],
                       [1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49],
                       [1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49],
                       [1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49],
                       [1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49, 1 / 49]
                       ])
```

인자로 받은 필터크기만큼 1D와 2D처리를 해주었는데, 받은 크기만큼 mask를 자동 생성하려했으나 3과 7로만 확인하면 되므로 이렇게 하였음

```
x = np.ceil(fshape[0] / 2)
y = np.ceil(fshape[1] / 2)

if x < 2:
    y = np.arange(-(y - 1), y, dtype=np.float32)
    exp_root = (y ** 2) / (-2 * sigma ** 2)
    filter_gaus = np.exp(exp_root)
    filter_gaus = (filter_gaus / np.sum(filter_gaus)).reshape(1, filter_gaus.shape[0])
elif y < 2:
    x = np.arange(-(x - 1), x, dtype=np.float32)
    exp_root = (x ** 2) / (-2 * sigma ** 2)
    filter_gaus = np.exp(exp_root)
    filter_gaus = (filter_gaus / np.sum(filter_gaus)).reshape(1, filter_gaus.shape[0])
else:

    grid = np.arange(-(x - 1), y, dtype=np.float32)
    x, y = np.meshgrid(grid, grid)
    exp_root = (x ** 2 + y ** 2) / (-2 * sigma ** 2)
    filter_gaus = np.exp(exp_root)
    filter_gaus = filter_gaus / np.sum(filter_gaus)

return filter_gaus
```

이것 또한 1d와 2d를 나누어 ppt의 식대로 진행하였다. 하지만 실행을 해보면 1D가 2D보다

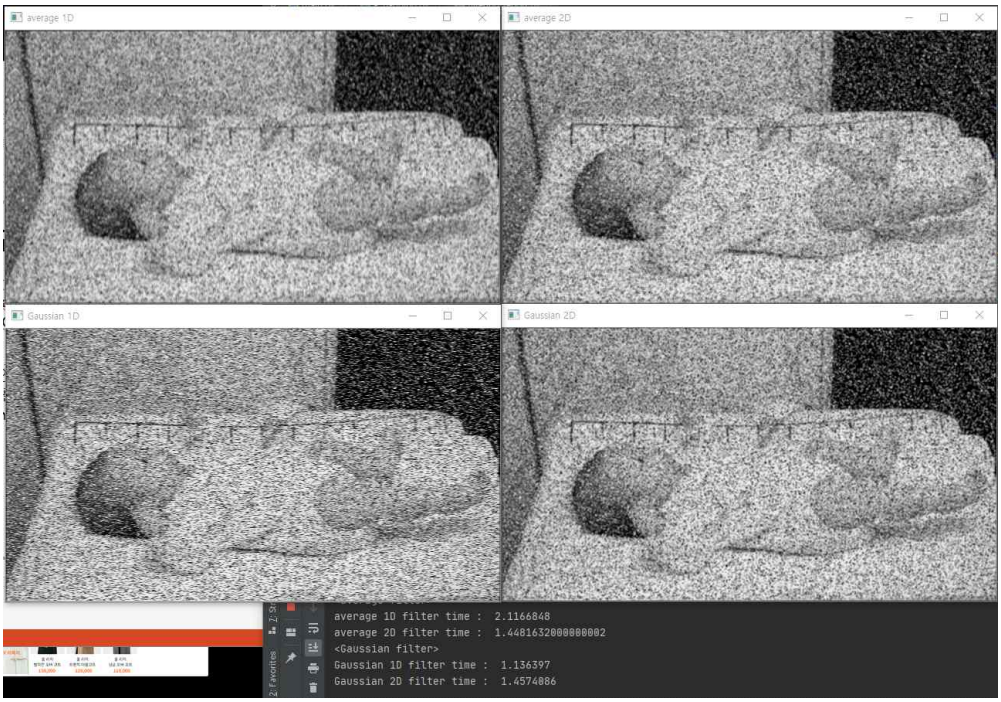
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

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

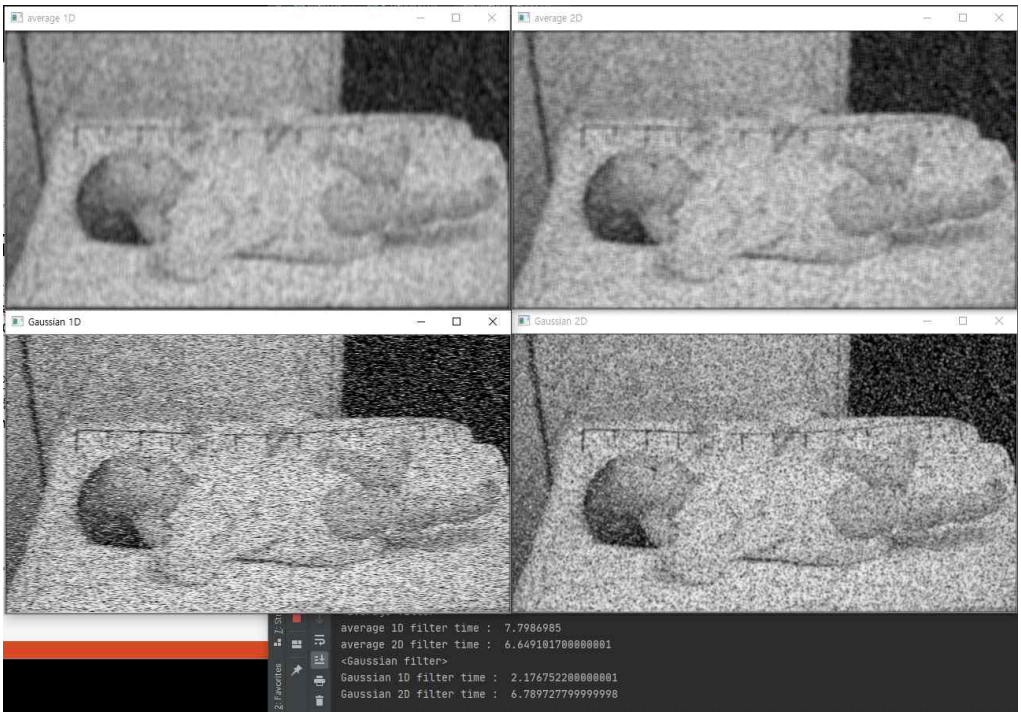
$$G(y) = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-y^2}{2\sigma^2}}\right)$$

결과값

filter_size = 3



filter_size = 7



느낀점

average 필터에서 1D 와 2D 의 시간차이에서 약간의 오류가 있는걸로 보여 구글링을 해본 결과 이미지의 크기와 필터의 크기, rgb이미지인가 아닌가에 따라 속도의 오류가 있을 수 있다는 것을 찾을 수 있었다.

Gaussian filter에서 sigma의 값이 의미하는 것은 Gaussian filter window의 사이즈 결정인데 위에 첨부한 식에서 x,y의 값들이 sigma의 3배를 넘어가면 exponential값이 현저하게 떨어져 이 이후 영역은 의미가 없다고 한다.

난이도

영상처리를 안들어 봐서 그런지 매우 어려운 난이도의 문제였다. 실습은 그나마 천천히 따라 하면서 변화 값들을 확인하며 따라 갈 수 있었는데 과제의 난이도는 실습의 난이도와 차이가 많이 나 힘들었다. python을 사용한 수업이 컴퓨터 그래픽스가 처음이여서 그런것도 한 몫 한 것 같다.