영상 처리

- 실습 4 : Gaussian filter -

제출일자	2021.04.11
분 반	01
이 름	강인한
학 번	201701969

과제1-1	[구현 코드]

2차원 가우시안 마스크는 다음과 같다.

```python
def my_get_Gaussian2D_mask(msize, sigma=1):
    ###########################################
    # ToDo
    # 2D gaussian filter 만들기
    ###########################################
    msizeDivTwo = msize//2
    y, x = np.mgrid[-(msizeDivTwo):msizeDivTwo+1,-(msizeDivTwo):msizeDivTwo+1]


    # 2차 gaussian mask 생성
    gaus2D = (1/(2*math.pi))*((math.exp(1))**(-1*((x**2)+(y**2))/2))
    # mask의 총 합 = 1
    gaus2D /= np.sum(gaus2D)
    return gaus2D
```

mgrid 함수를 이용하여 y와 x에 값을 넣는다. msize가 3일 경우 y는

```
y = [[-1,-1,-1],
     [ 0, 0, 0],
     [ 1, 1, 1]]
```

```
x = [[-1, 0, 1],
     [-1, 0, 1],
     [-1, 0, 1]]
```

x는 다음과 같다.

2차원 gaus는 다음과 같다.

```python
# 2차 gaussian mask 생성
gaus2D = (1/(2*math.pi))*((math.exp(1))**(-1*((x**2)+(y**2))/2))
# mask의 총 합 = 1
gaus2D /= np.sum(gaus2D)
return gaus2D
```

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

1차원 가우시안 마스크는 다음과 같다.

```python
def my_get_Gaussian1D_mask(msize, sigma=1):
    ########################################
    # ToDo
    # 1D gaussian filter 만들기
    ########################################

    x = np.full((1, msize), np.mgrid[-(msize//2):msize//2+1])


    #gaus1D = ???
    gaus1D = ((1/((2*math.pi)**(1/2))) * math.exp(1)**((-0.5)*(x**2)))
    # mask의 총 합 = 1
    gaus1D /= np.sum(gaus1D)
    return gaus1D
```

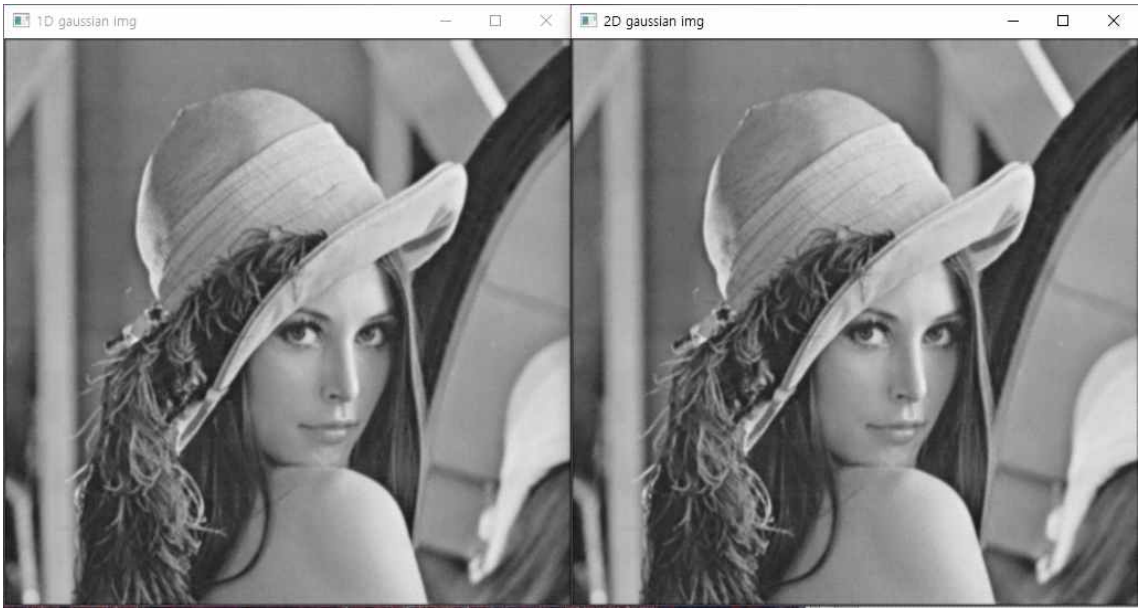
x, y는 편의성을 위해 2차원으로 구성한다.

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

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

1-2	[실행 결과 및 이미지]

```
mask size :  5
1D gaussian filter
zero padding
<mask>
[[0.05448868]
 [0.24420134]
 [0.40261995]
 [0.24420134]
 [0.05448868]]
zero padding
<mask>
[[0.05448868 0.24420134 0.40261995 0.24420134 0.05448868]]
1D time :  7.8150579
2D gaussian filter
zero padding
<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]]
```



1-3	[시간 비교 및 느낀 점]

처음으로 과제가 어렵지 않았던 것 같습니다. 1차원 두 개를 사용하는 게 2차원 한 개를 사용하는 것보다 훨씬 시간이 적게 걸린다.

느낀 점으로는 과제가 어렵지 않고 할만했던 것 같습니다.

과제2-1	[구현 코드]

```python
def my_bilinear(src, scale):
    ########################
    # TODO                 #
    # my_bilinear 완성     #
    ########################
    (h, w) = src.shape
    h_dst = int(h * scale + 0.5)
    w_dst = int(w * scale + 0.5)
    dst = np.zeros((h_dst, w_dst))
    # interbilinear polation 적용
    for row in range(h_dst):
        for col in range(w_dst):
            # 참고로 꼭 한줄로 구현해야 하는건 아닙니다 여러줄로 하셔도 상관없습니다.(저도 엄청길게 구현했습니다.)
            t,s=float(col/scale),float(row/scale)
            m,n= int(t), int(s)
            if m>= h-1 or n>= w-1:
                dst[col,row]=src[m,n]
                continue

            dst[col,row] =src[m,n]*(m+1-t)*(n+1-s)+src[m+1,n]*(n+1-s)*(t-m)+src[m,n+1]*(s-n)*(m+1-t)+src[m+1,n+1]*(t-m)*(s-n)
    return dst
```

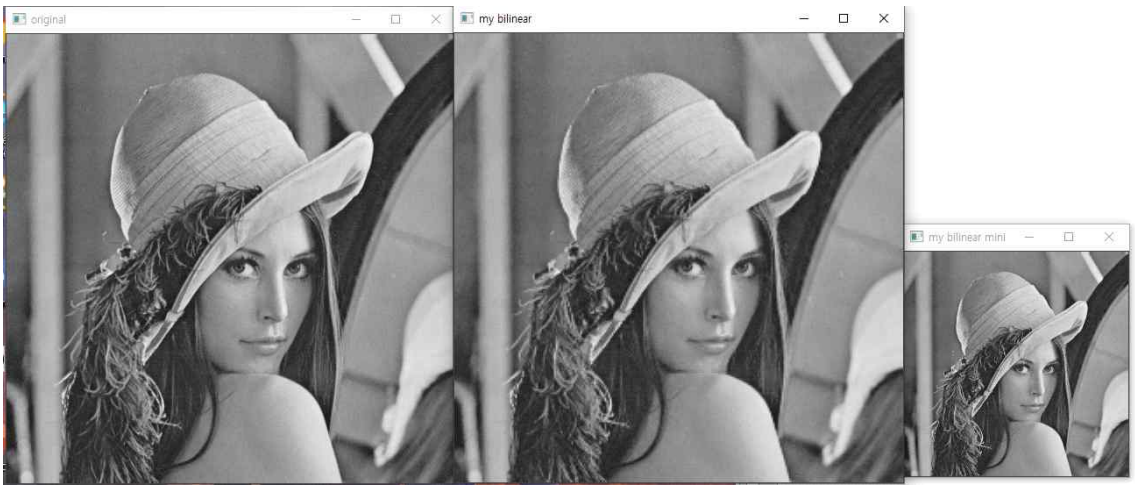
반복문을 통해 dst[col, row] 에 기존의 픽셀 값인 src 식을 다음 식에 적용한다.

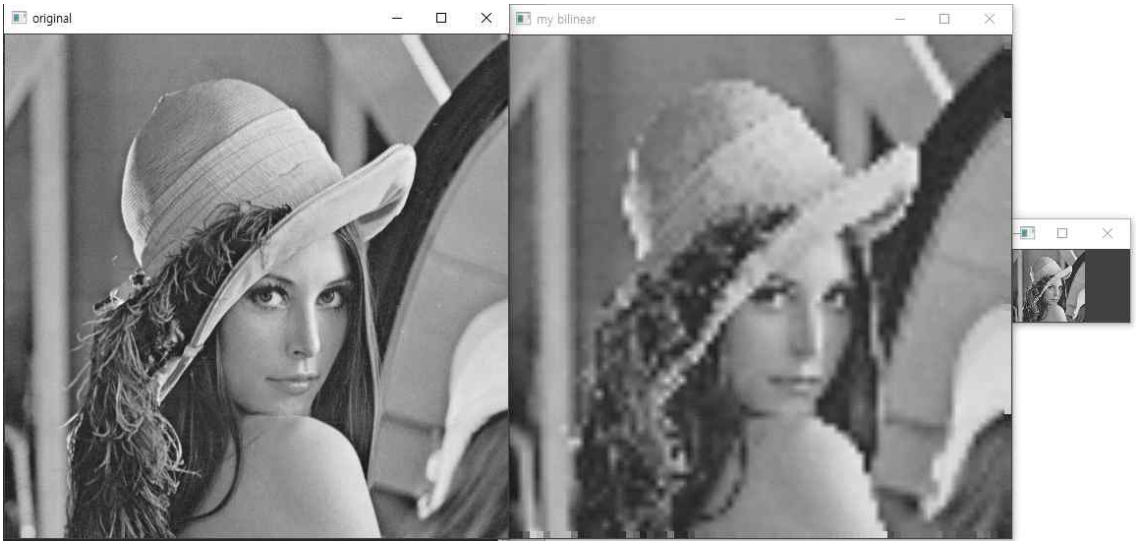
$$
\begin{aligned}
f(u,v) = {} & (1-s)(1-t)\cdot f(m,n) \\
& + s(1-t)\cdot f(m,n+1) \\
& + (1-s)t\cdot f(m+1,n) \\
& + st\cdot f(m+1,n+1)
\end{aligned}
$$

if문을 통해 제일 바깥에 있는 쪽의 픽셀을 정해준다.

과제2-2	[이미지]

1/2





2-3	[느낌 및 난이도]

처음으로 크기 조절을 해봤는데 재밌었던 것 같습니다. 아주 어렵지도 않아서 더 흥미롭습니다.