영상처리 10주차 과제

학번: 201404376

이름: 정성욱

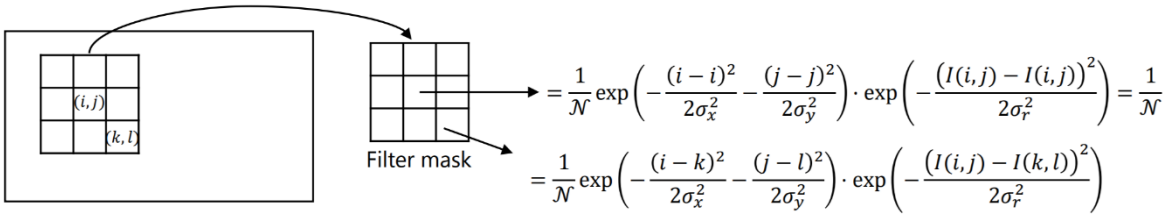
1. 과제 내용

목표 Bilateral Filter에 대한 구현

- Bilateral filtering

2d Gaussians

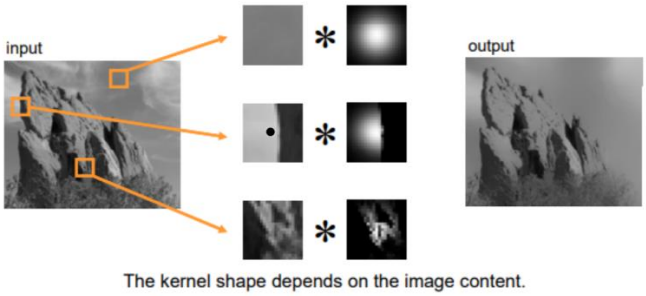
$$f(i,j,k,l) = \frac{1}{\mathcal{N}} \exp\left(-\frac{(i-k)^2}{2\sigma_x^2} - \frac{(j-l)^2}{2\sigma_y^2}\right) \cdot \exp\left(-\frac{\left(I(i,j)-I(k,l)\right)^2}{2\sigma_r^2}\right)$$



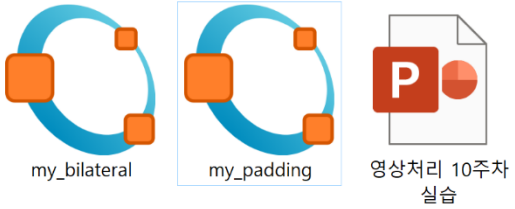
바이 리터럴은 하나의 픽셀에 대해 주변 픽셀에 대해 필터 크기 차이만큼/2*r시그마^2에 대해 exp하는 것과 주변픽셀의 인덱스와 현재의 인덱스의 차이에 대한 제곱에 2*x시그마^2, 2*y시그마^2를 나누어 준 것에 대한 exp에 대한 곱임. N은 갑보정을 위해 해당 필터가 갖는 원소들의 합이 1이 되도록 보정해주는 보정값임. 가우시안과 비슷한 블러 효과를 줌과 동시에 가우시안이 가지던 단점인 edge가 사라지던 현상에 대해 edge가 남겨지도록 극복하는 효과가 있음.

- Bilateral filtering

Same Gaussian kernel everywhere.



The kernel shape depends on the image content.



중간에 패딩 과정이 필요한데 이때는 mirror패딩을 사용해야 하므로, mirror을 사용하기 위해 이전 과제에 해 두었던, my_padding을 가져왔습니다.

```
function filter_img = my_bilateral(img, filter_size, sigma, sigma2)
```

매개변수로 흑백 이미지, 필터의 크기 시그마1 시그마2 를 받음

```
[ox,oy] = size(img);

pad_size = floor(filter_size/2);
pad_img = double(my_padding(img,pad_size,'mirror'));
[px,py] = size(pad_img);
filter_img = zeros(size(img));
```

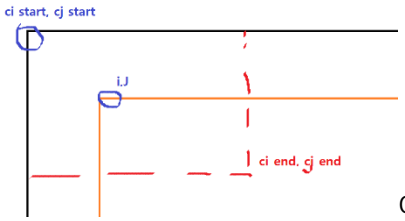
필터를 돌리기 위한 변수들을 저장하고 이미지에 미러패딩을 씌움

```
for i = 1+pad_size:px-pad_size
    for j = 1+pad_size:py-pad_size
        filter=  zeros(filter_size,filter_size);
        ci_start =i-pad_size;    ci_end =i+pad_size;
        cj_start =j-pad_size;    cj_end =j+pad_size;

        fi=1;
        for tk = ci_start:ci_end
        fj=1;
        for tl = cj_start:cj_end
            filter(fi,fj) =exp((-(i-tk)^2-(j-tl)^2)/(2*sigma^2))*exp(-(pad_img(i,j)-pad_img(tk,tl))^2/(2*sigma2^2));
            fj=fj+1;
        end
            fi=fi+1;
        end
        sum_f =  sum(sum(filter));
        filter = filter/sum_f;

        conv_f_i=filter.*double(pad_img(ci_start:ci_start+filter_size-1,cj_start:cj_start+filter_size-1));
        filter_img(ci_start,cj_start) = sum(sum(conv_f_i));
    end
end
```

패딩 된 이미지 루프를 돌 되 패딩 되기전 오리지널 이미지의 위치에 맞춰서 시작함

```
ci_start =i-pad_size;    ci_end =i+pad_size;
cj_start =j-pad_size;    cj_end =j+pad_size;
```

는 패딩된 이미지의 시작과 끝을 의미함 즉,



어떤 필터를 잡아도 첫 이미지 기준 저렇게 잡힘.

저 필터의 범위를 돌면서 필터링이 씌워질 이미지의 i,j위치에 대해

$$\frac{1}{\mathcal{N}}\exp\left(-\frac{(i-k)^2}{2\sigma_x^2}-\frac{(j-l)^2}{2\sigma_y^2}\right)\cdot\exp\left(-\frac{\left(I(i,j)-I(k,l)\right)^2}{2\sigma_r^2}\right)$$

연산을 수행함

(이 이미지 크기는 원본과 같으므로 이에 대한 처리로써 코드에서는 따로 fi 변수를 둠) 이렇게 구해진 필터에 대해 각각이 다른 값 분포를 가지므로 그 필터에 대한 합을 줘야하므로 맨마지막에

```
sum_f =  sum(sum(filter));
filter = filter/sum_f;
```

로 필터 값에 대한 보정을 합니다.

그리고 필터를 씌우는데 같은 인덱스 i,j에 대해서도 고려하지 않은 게 어차피 마지막에

```
filter_img = filter_img-1;
%s= sum(sum(filter));
%filter_img = filter;
filter_img = uint8(filter_img);
```

1을 빼줌으로써 exp(0)*exp(0)에 대해 1더해진값을 빼주므로 값 보정을 취해줍니다

구현 결과

2. 느낀 점

:: 구현하면서 느낀 점, 어려웠던 점, 혹은 설명이 필요하다고 느낀 부분

이번꺼는 할만했습니다.