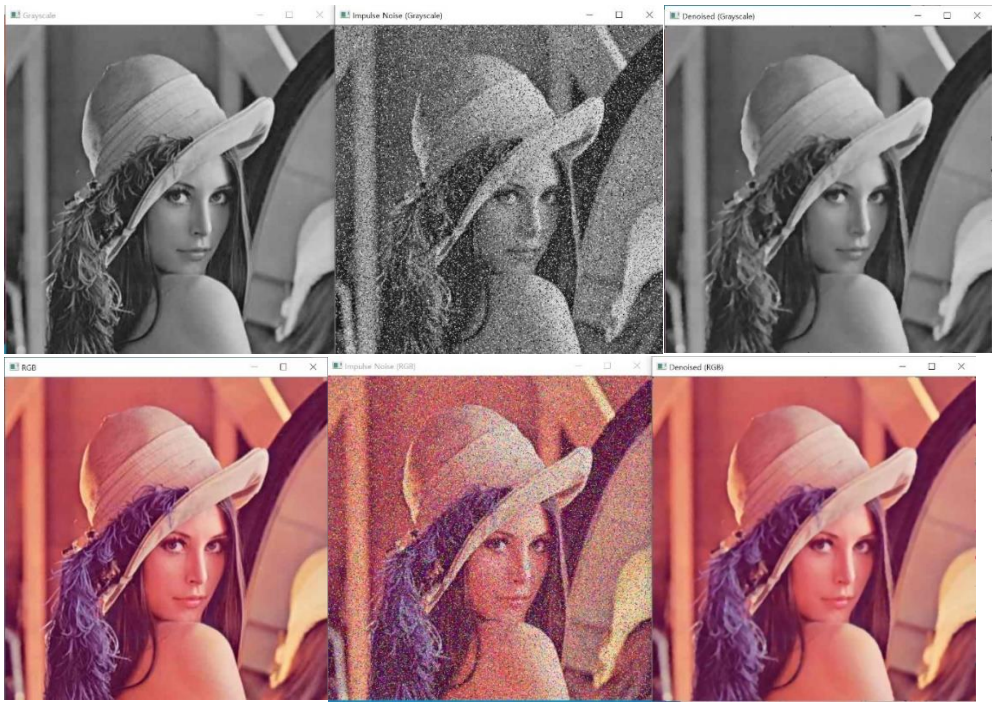
Technical Report

1829008 김민영

<1> Practice1

Salt an Pepper Noise Removal

1. result image



2. Explanation of code (salt_and_pepper.cpp 참고)

Main 함수 : Salt-and-pepper noise 를 만들고, 이후 median filtering 을 이용하여 이를 제거한 결과를 출력한다.

Add_salt_pepper_Noise 함수 : salt-pepper noise 를 만든다.

Salt_pepper_noise_removal_Gray, Salt_pepper_noise_removal_RGB 함수 : median filtering 을 이용하여 noise 제거(border 방식 : zero-padding, mirroring, adjust-kernel)

3. Analysis

Salt-and pepper noise : Salt(흰색-255) Pepper(검정색-0)를 랜덤하게 뿌린다.

Median filter : window 안에 있는 픽셀들을 정렬시키고, 중간값을 output 으로 선택한다.

Median filter 는 low-pass filtering 보다 Salt-and pepper noise 제거가 잘 된다. Result image 를 보면 예상한대로 잘 제거가 된 것을 확인할 수 있다.

<2> Practice2 : Gaussian Noise

1. result image

<original> <gaussian noise> <guassian filter denoise> <bilateral filter denoise>

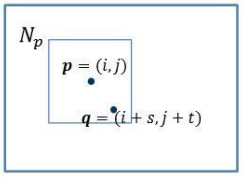
2. Explanation of code (Gaussian_Biliteral.cpp 참고)

Main 함수: Gaussian Noise를 만들고, 이후 1) Gaussian filtering, 2)bilateral filtering을 이용하여 이를 제거한 결과를 출력한다.

Add_Gaussian_noise 함수 : Gaussian noise 를 만들어 이미지에 추가한다.

Gaussianfilter_Gray, Gaussianfilter_ RGB 함수 : gaussian filtering 을 이용하여 noise 제거(저번과제와 비슷)

Bilateralfilter_Gray, Bilateralfilter_RGB 함수 : Bilateralfilter 를 이용하여 noise 제거



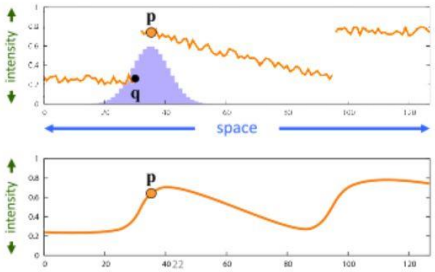
$$O_p = \frac{1}{W_p} \sum_{q \in N_p} G_{\sigma_s}(|p - q|) G_{\sigma_r}(|I_p - I_q|) I_q \qquad W_p = \sum_{q \in N_p} G_{\sigma_s}(|p - q|) G_{\sigma_r}(|I_p - I_q|)$$

```
kernelvalue = kernel.at<float>(x + n, y + n)*exp(-(pow(input.at<double>(i, j) -
input.at<double>(i + x, j + y), 2.0)) / 2 * sigma_r);
```

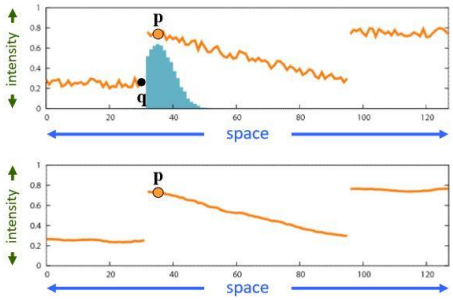
3. Analysis

Gaussian Noise: Adaptive white noise

Gaussian Filter :



Bilateral Filter :



->Result image 를 보면, gaussian filter 를 적용한 output 에서는 over smoothing 이 발생했고 bilateral filter 에서는 over smoothing 이 없는 것을 확인할 수 있다.