1. Noise 생성

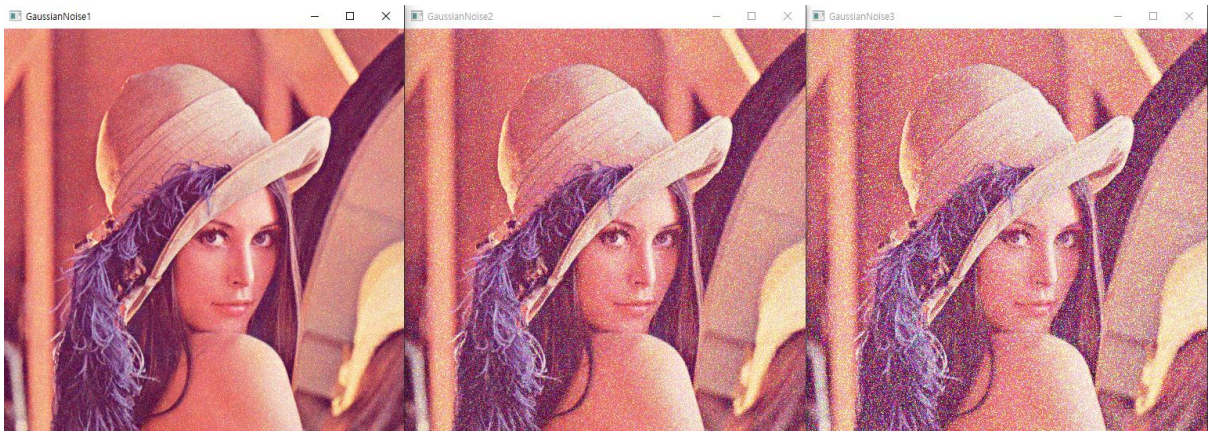
1.1 Gaussian Noise

```
void Gaussian_noise(Mat &input, Mat &output, double std) {
        Mat noise_image(input.rows, input.cols, CV_8UC3, Scalar(0));
        double average = 0.0;
        randn(noise_image, Scalar::all(average), Scalar::all(std));
        input.convertTo(output, CV_8UC3);
        addWeighted(output, 1.0, noise_image, 1.0, 0.0, output);
        output.convertTo(output, input.type());
}
```

- Gaussian Noise 추가해주는 함수 구현. Randn과 addWeighted사용.

- 입력되는 std에 따라 원하는 정도로 gaussian noise 추가.

- 왼쪽부터 std를 30, 60, 100으로 설정하여 만든 예시 이미지(Gaussian_image1~3)



1.2 Salt/Pepper noise

```
void Salt_and_Pepper_noise(Mat &input, double noise_ratio) {
        int rows = input.rows;
        int cols = input.cols;
        int ch = input.channels();
```

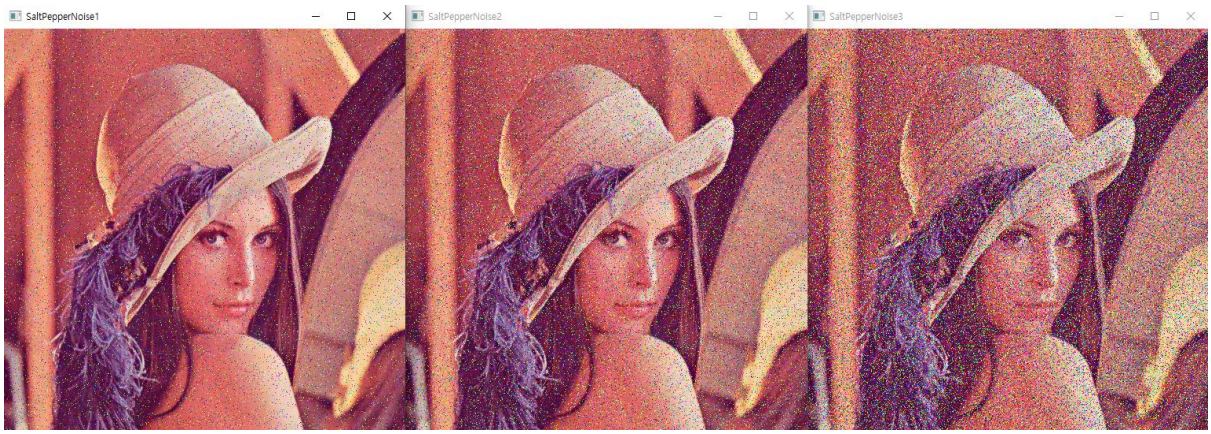
```
        int num_of_noise_pixels = (int)((double)(rows*cols*ch)*noise_ratio);

        if (input.channels() == 3) {      //컬러이미지
                for (int i = 0; i < num_of_noise_pixels; i++) {
                        int r = rand() % rows;
                        int c = rand() % cols;
                        int _ch = rand() % ch;
                        uchar* pixel = input.ptr<uchar>(r) + (c*ch) + _ch;
                        *pixel = (rand() % 2 == 1) ? 255 : 0;
                }
        }
        else if (input.channels() == 1) { //흑백이미지
                for (int i = 0; i < num_of_noise_pixels; i++) {
                        int r = rand() % rows;
                        int c = rand() % cols;
                        uchar* pixel = input.ptr<uchar>(r) + (c*ch);
                        *pixel = (rand() % 2 == 1) ? 255 : 0;
                }
        }
}
```

- noise ratio를 입력받아 일정 비율만큼만 salt / pepper noise를 추가해주는 함수 구현.

- 컬러 이미지를 받은 경우 rgb 채널도 랜덤으로 선택되어 노이즈가 추가된다.

- 왼쪽부터 노이즈비율 0.05, 0.1, 0.25로 설정하여 만든 예시 이미지(S_P_image1~3)



2. Smoothing filter(average filter)를 이용한 gaussian noise 개선

2.1 3X3 average filter

```
float aver_data[] = {    //3X3 moving average
        1 / 9.f,1 / 9.f,1 / 9.f,
        1 / 9.f,1 / 9.f,1 / 9.f,
        1 / 9.f,1 / 9.f,1 / 9.f
};

Mat aver_mask(3, 3, CV_32F, aver_data);

Mat aver_outimage1_1(h, w, CV_8UC3, Scalar(0));
filter(Gaussian_image1, aver_outimage1_1, aver_mask, 3, 3);
Mat aver_outimage2_1(h, w, CV_8UC3, Scalar(0));
filter(Gaussian_image1, aver_outimage2_1, aver_mask, 3, 3);
Mat aver_outimage3_1(h, w, CV_8UC3, Scalar(0));
filter(Gaussian_image1, aver_outimage3_1, aver_mask, 3, 3);
```

- 지난 과제에서 구현한 filter함수에 3X3 average 커널 마스크를 사용하여 smoothing

2.2 5X5 average filter

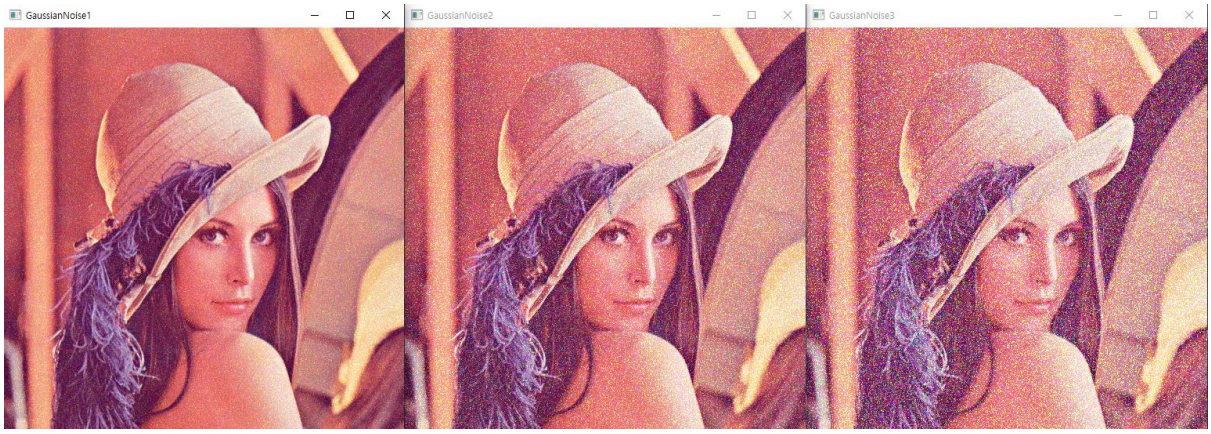
```
float aver_data2[] = {           //5X5 moving average
        1 / 25.f,1 / 25.f,1 / 25.f,1 / 25.f,1 / 25.f,
        1 / 25.f,1 / 25.f,1 / 25.f,1 / 25.f,1 / 25.f,
        1 / 25.f,1 / 25.f,1 / 25.f,1 / 25.f,1 / 25.f,
        1 / 25.f,1 / 25.f,1 / 25.f,1 / 25.f,1 / 25.f,
        1 / 25.f,1 / 25.f,1 / 25.f,1 / 25.f,1 / 25.f
};
Mat aver_mask2(5, 5, CV_32F, aver_data2);

Mat aver_outimage1_2(h, w, CV_8UC3, Scalar(0));
filter(Gaussian_image1, aver_outimage1_2, aver_mask2, 5, 5);
Mat aver_outimage2_2(h, w, CV_8UC3, Scalar(0));
filter(Gaussian_image2, aver_outimage2_2, aver_mask2, 5, 5);
Mat aver_outimage3_2(h, w, CV_8UC3, Scalar(0));
filter(Gaussian_image3, aver_outimage3_2, aver_mask2, 5, 5);
```

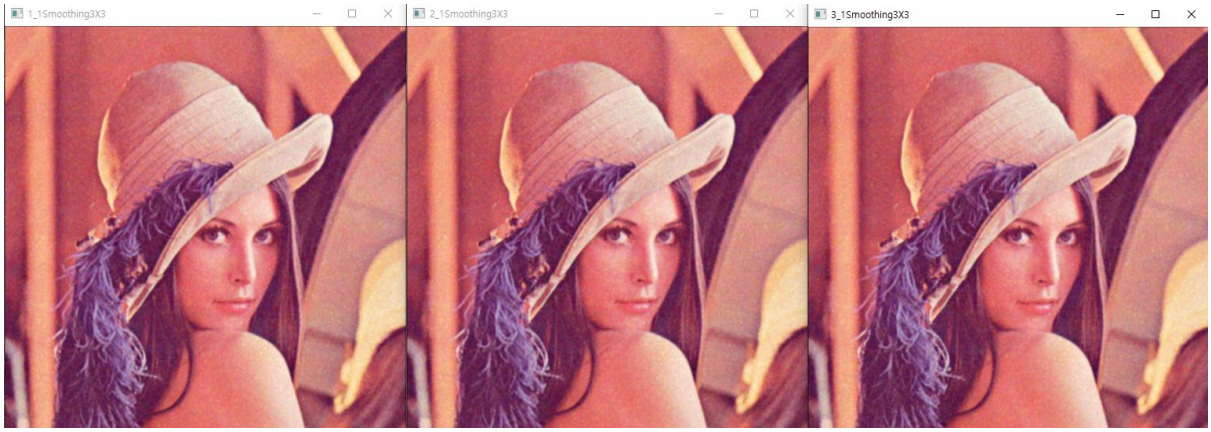
- 지난 과제에서 구현한 filter함수에 5X5 average 커널 마스크를 사용하여 smoothing

2.3 결과비교

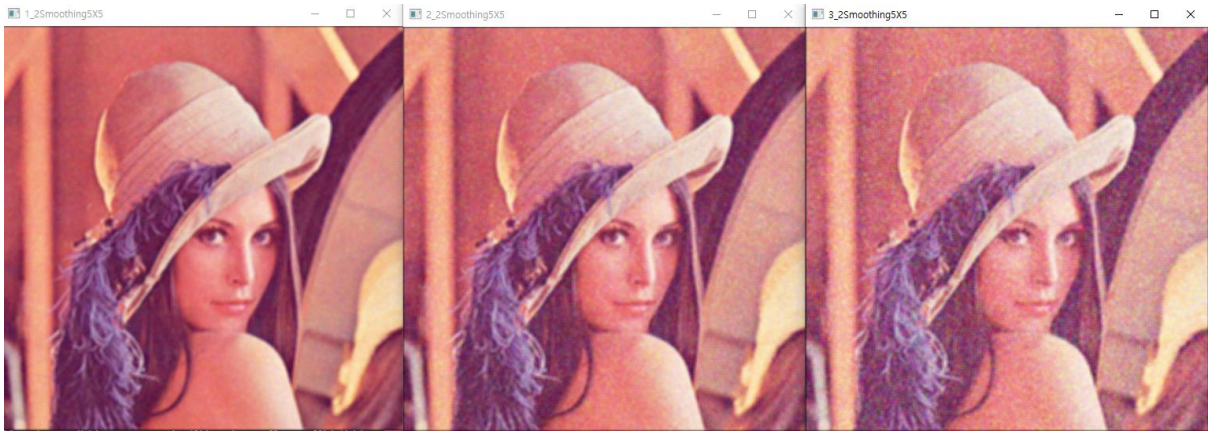
- Gaussian Noise image. 왼쪽부터 std의 값이 30, 60, 100



- 3X3 moving average filter 적용 결과



- 5X5 moving average filter 적용 결과



- filter 커널의 크기가 클수록 성능이 떨어지는 경향이 있음.

3. 3X3 median filter

3.1 median filter 구현

```
void median_filter(Mat &img, Mat &output_img) {
	Mat copy_plate(img.rows + 2, img.cols + 2, CV_32F, Scalar(0));//복사, 계산용

	uchar* copy_data = copy_plate.data;
	uchar* img_data = img.data;
	uchar* output_data = output_img.data;


	//zero padding제외 가운데 원본사진복사
	for (int row = 0; row < img.rows; row++)
	{
		for (int col = 0; col < img.cols; col++)
		{
			if (img.channels() == 3) {
				copy_data[(row + 3 / 2)*(copy_plate.cols) * 3 + (col + 3 / 2)
* 3] = img_data[row * img.cols * 3 + col * 3];
				copy_data[(row + 3 / 2)*(copy_plate.cols) * 3 + (col + 3 / 2)
* 3 + 1] = img_data[row * img.cols * 3 + col * 3 + 1];
				copy_data[(row + 3 / 2)*(copy_plate.cols) * 3 + (col + 3 / 2)
* 3 + 2] = img_data[row * img.cols * 3 + col * 3 + 2];
			}
			else if (img.channels() == 1) {
				copy_data[(row + 3 / 2)*(copy_plate.cols) + (col + 3 / 2)] =
img_data[row * img.cols + col];
			}
		}
	}

	for (int row = 0; row < img.rows; row++)
	{
		for (int col = 0; col < img.cols; col++)
		{
			if (img.channels() == 3) {		//컬러이미지 일때

				float arr_b[9];
				float arr_g[9];
				float arr_r[9];
				for (int r = 0; r < 3; r++) {
					for (int c = 0; c < 3; c++) {
						arr_b[3*r + c] = copy_data[(row +
r)*(copy_plate.cols) * 3 + (col + c) * 3];
						arr_g[3*r + c] = copy_data[(row +
r)*(copy_plate.cols) * 3 + (col + c) * 3 + 1];
						arr_r[3*r + c] = copy_data[(row +
r)*(copy_plate.cols) * 3 + (col + c) * 3 + 2];
					}
				}
```

```
                        quickSort(arr_b, 9);
                        quickSort(arr_g, 9);
                        quickSort(arr_r, 9);
                        output_data[row * img.cols * 3 + col * 3] = arr_b[4];
                        output_data[row * img.cols * 3 + col * 3 + 1] = arr_g[4];
                        output_data[row * img.cols * 3 + col * 3 + 2] = arr_r[4];
                }
                else if (img.channels() == 1) {   //흑백이미지 일때

                        float arr[9];
                        for (int r = 0; r < 3; r++) {
                                for (int c = 0; c < 3; c++) {
                                        arr[3*r + c] = copy_data[(row +
r)*(copy_plate.cols) + (col + c)];
                                }
                        }

                        quickSort(arr, 9);
                        output_data[row * img.cols + col] = arr[4];

                }
        }
    }
}
```

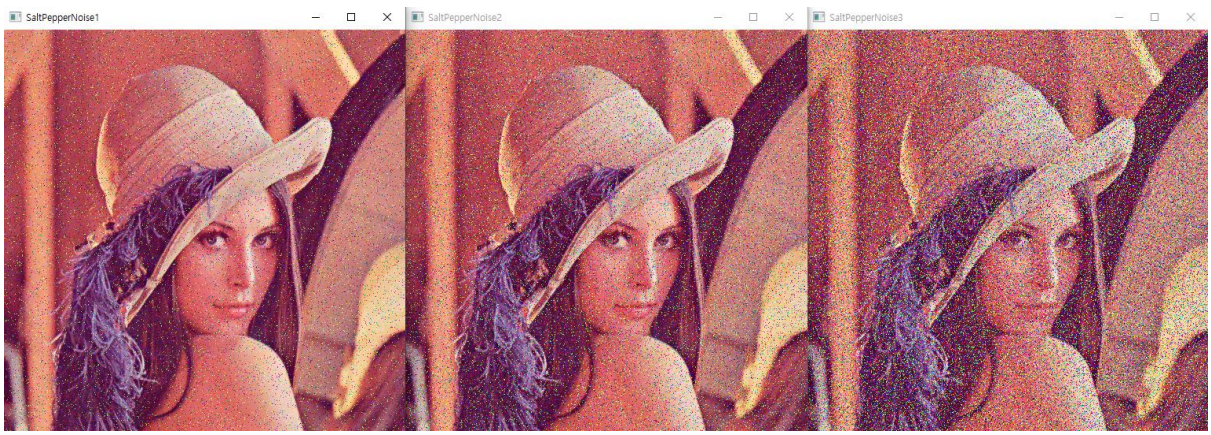
- 외부영역은 zero padding 후 연산.

- 중간값 arr[4]를 구하기 전 정렬에는 quick sort 알고리즘을 사용.

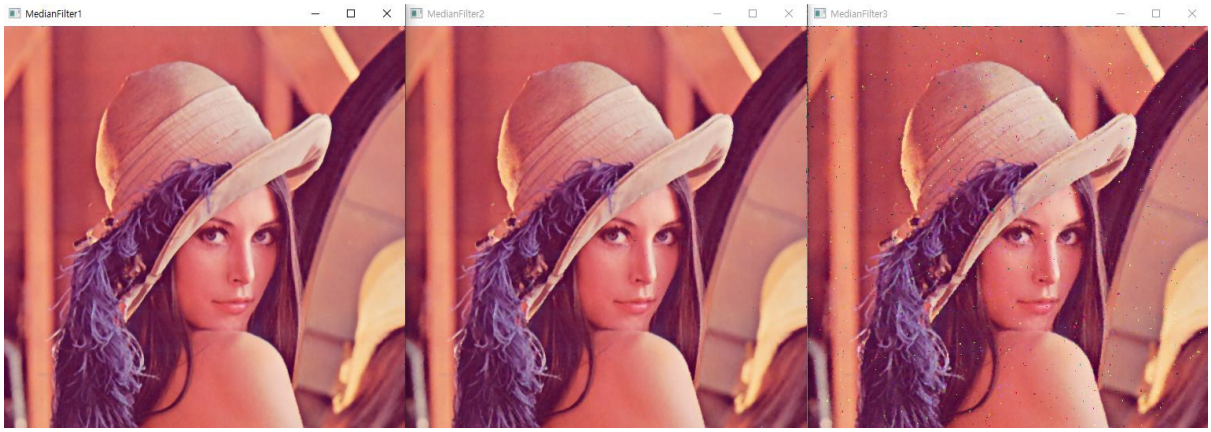
- 컬러이미지의 경우 rgb 채널별로 median filter 연산을 수행하였음.

3.2 결과비교

- 왼쪽부터 salt / pepper 노이즈비율 0.05, 0.1, 0.25로 설정하여 만든 예시 이미지(S_P_image1~3)



- 각각의 이미지에 median filter 적용 결과



- noise 비율이 높은 이미지의 경우 어느정도 noise가 남아있는 경우가 발생함.

- 이는 median filter의 크기를 3X3보다 크게 키우면 해결 가능하지만, filter의 크기가 커질수록 원본의 정보와 더 멀어짐.