**디지털 영상처리 과제**

**2011903054 행정학과**

**문석현**

**1. 가우시안 노이즈(gaussian noise) –**

**소스 코드**

#include <opencv2/opencv.hpp>

#include <iostream>

#include<opencv2/highgui.hpp>

using namespace cv;

int main()

{

Mat inputimg;

inputimg = imread("lena.png", cv::IMREAD\_COLOR);

resize(inputimg, inputimg, Size(), 0.3, 0.3, cv::INTER\_AREA);

Mat noise\_image(inputimg.size(), CV\_16SC3);

double average = 0.0;

double std = 30.0;

randn(noise\_image, Scalar::all(average), Scalar::all(std));

Mat temp\_image;

inputimg.convertTo(temp\_image, CV\_16SC3);

addWeighted(temp\_image, 1.0, noise\_image, 1.0, 0.0, temp\_image);

temp\_image.convertTo(temp\_image, inputimg.type());

namedWindow("original", cv::WINDOW\_AUTOSIZE);

namedWindow("GaussianNoise", cv::WINDOW\_AUTOSIZE);

imshow("original", inputimg);

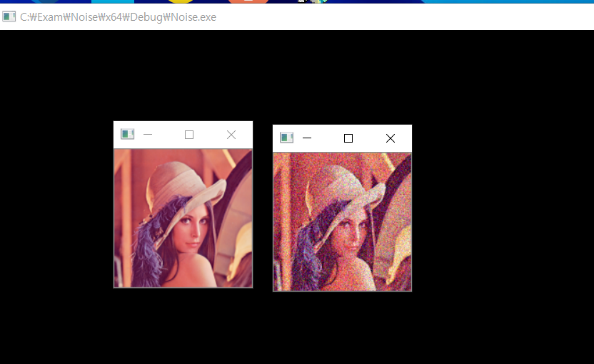
imshow("GaussianNoise", temp\_image);

waitKey(0);

return 0;

}

결과 화면



2. salt/pepper noise

**소스코드**

#include <iostream>

#include<opencv2/core.hpp>

#include <opencv2/opencv.hpp>

#include<opencv2/highgui.hpp>

#include <stdlib.h>

#include <time.h>

using namespace cv;

using namespace std;

void addSaltAndPepperNoise(Mat&img, double noise\_ratio)

{

int rows = img.rows;

int cols = img.cols;

int ch = img.channels();

int num\_of\_noise\_pixels = (int)((double)(rows \* cols \* ch)\*noise\_ratio);

for (int i = 0; i < num\_of\_noise\_pixels; i++)

{

int r = rand() % rows;

int c = rand() % cols;

int \_ch = rand() % ch;

uchar \* pixel = img.ptr<uchar>(r) + (c\*ch) + \_ch;

\*pixel = (rand() % 2 == 1) ? 255 : 0;

}

}

int main(int argc, char\*\*argv)

{

Mat inputimg;

Mat spimg;

inputimg = imread("lena.png", cv::IMREAD\_COLOR);

resize(inputimg, inputimg, Size(), 1, 1, cv::INTER\_AREA);

Mat noise\_image(inputimg.size(), CV\_16SC3);

double average = 0.0;

double std = 30.0;

randn(noise\_image, Scalar::all(average), Scalar::all(std));

spimg = inputimg.clone();

addSaltAndPepperNoise(spimg, 0.05);

Mat temp\_image;

inputimg.convertTo(temp\_image, CV\_16SC3);

addWeighted(temp\_image, 1.0, noise\_image, 1.0, 0.0, temp\_image);

temp\_image.convertTo(temp\_image, inputimg.type());

namedWindow("original", cv::WINDOW\_AUTOSIZE);

namedWindow("salt&pepperNoise", cv::WINDOW\_AUTOSIZE);

imshow("original", inputimg);

imshow("salt&pepperNoise", temp\_image);

waitKey(0);

return 0;

}

**결과화면**

****

**3. 가우시안 노이즈 필터 제거**

**<소스코드>**

#include <iostream>

#include<opencv2/core.hpp>

#include <opencv2/opencv.hpp>

#include<opencv2/highgui.hpp>

#include <stdlib.h>

#include <time.h>

using namespace cv;

using namespace std;

void addSaltAndPepperNoise(Mat&img, double noise\_ratio)

{

int rows = img.rows;

int cols = img.cols;

int ch = img.channels();

int num\_of\_noise\_pixels = (int)((double)(rows \* cols \* ch)\*noise\_ratio);

for (int i = 0; i < num\_of\_noise\_pixels; i++)

{

int r = rand() % rows;

int c = rand() % cols;

int \_ch = rand() % ch;

uchar \* pixel = img.ptr<uchar>(r) + (c\*ch) + \_ch;

\*pixel = (rand() % 2 == 1) ? 255 : 0;

}

}

int main(int argc, char\*\*argv)

{

Mat inputimg;

Mat spimg;

inputimg = imread("result.png", cv::IMREAD\_COLOR);

resize(inputimg, inputimg, Size(), 1, 1, cv::INTER\_AREA);

Mat noise\_image(inputimg.size(), CV\_16SC3);

double average = 0.0;

double std = 30.0;

randn(noise\_image, Scalar::all(average), Scalar::all(std));

spimg = inputimg.clone();

addSaltAndPepperNoise(spimg, 0.03);

Mat gaussianSmoothedimg;

GaussianBlur(spimg, gaussianSmoothedimg, Size(5, 5), 0.05);

Mat temp\_image;

inputimg.convertTo(temp\_image, CV\_16SC3);

addWeighted(temp\_image, 1.0, noise\_image, 1.0, 0.0, temp\_image);

temp\_image.convertTo(temp\_image, inputimg.type());

namedWindow("original", cv::WINDOW\_AUTOSIZE);

namedWindow("Gaussiansmoothinh", cv::WINDOW\_AUTOSIZE);

imshow("original", inputimg);

imshow("Gaussiansmoothing", temp\_image);

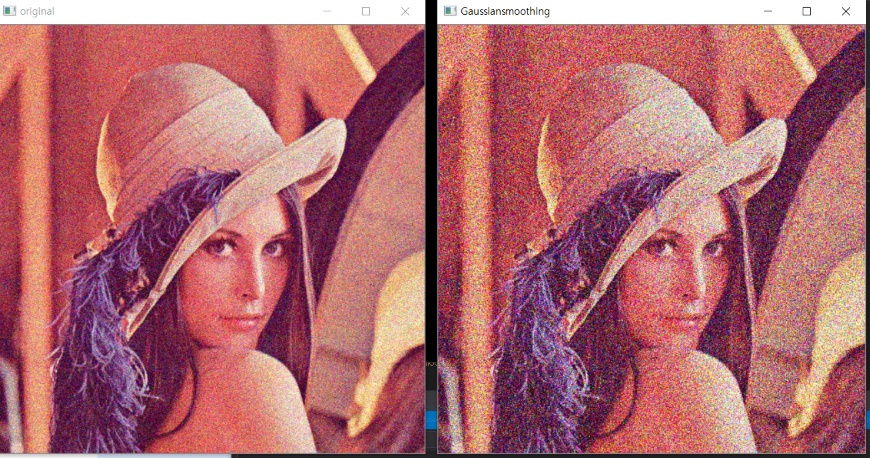
waitKey(0);

return 0;

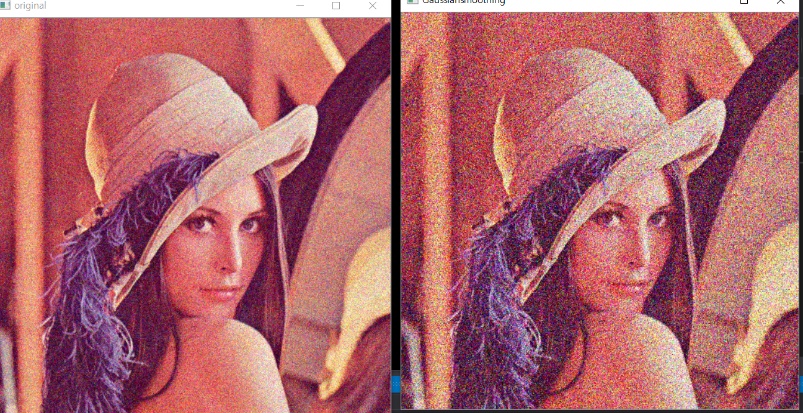
}

<결과화면>

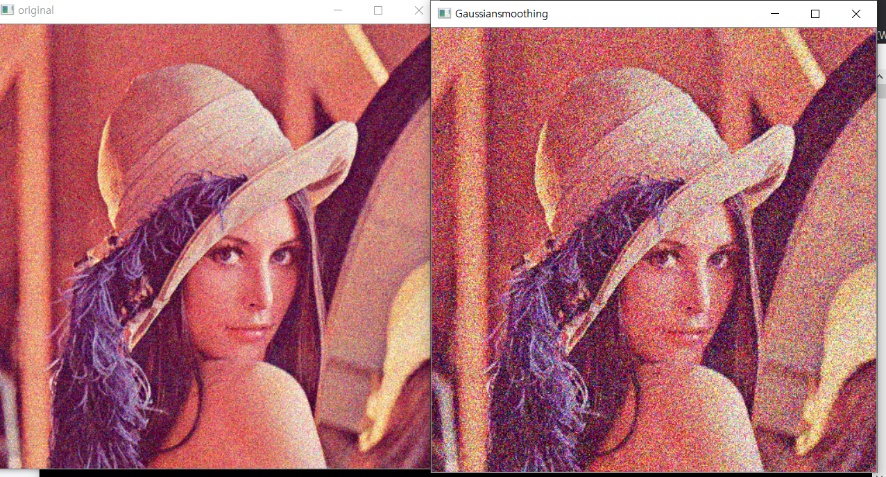
1. 크기가 5x5일 때

X, y의 가중치가 3인경우  
****

X,y의 가중치가 6인경우

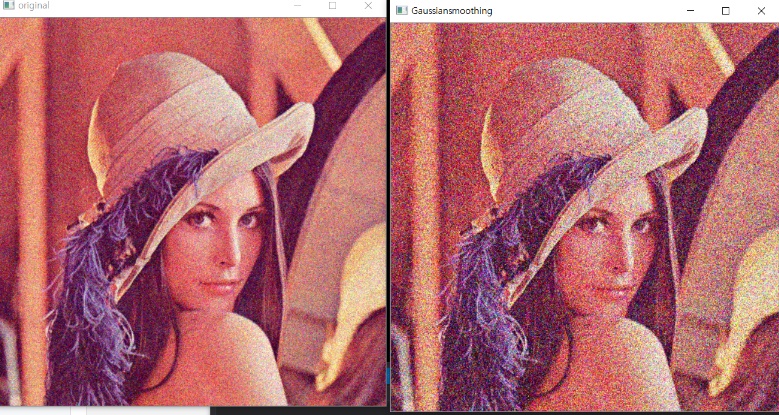
****

**X,y의 가중치가 10인경우**

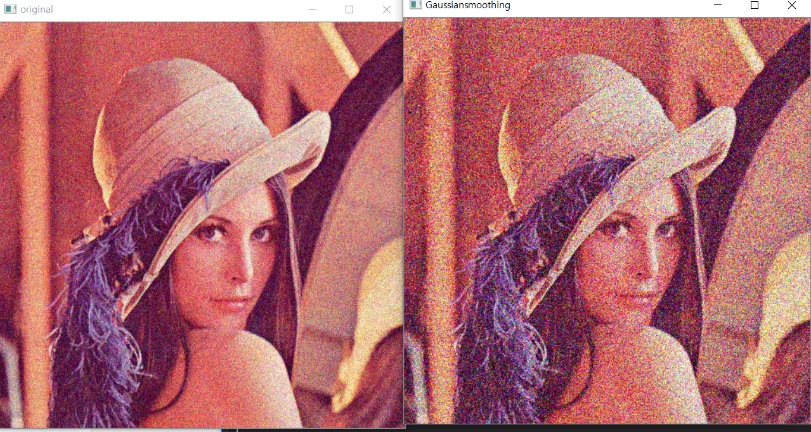
****

**2. 크기가 3x3 인경우**

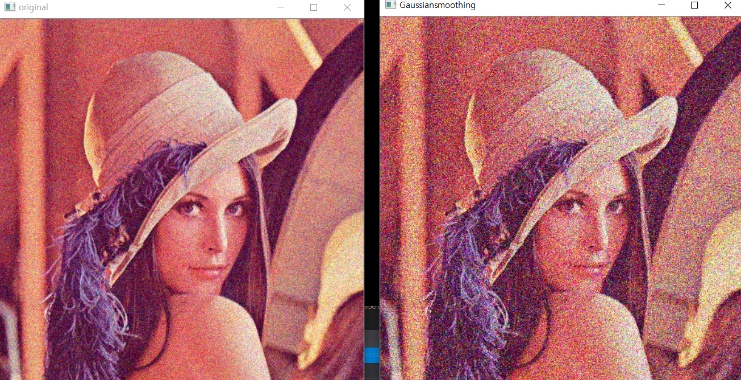
**X, y 가중치가 3인경우**

****

**X,y 가중치가 5인경우**

****

**X,y 가중치가 10인경우**

****

**3. mean filter**

**<소스코드>**

#include<iostream>

#include<opencv2/core.hpp>

#include<opencv2/imgproc.hpp>

#include<opencv2/highgui.hpp>

#include<random>

using namespace std;

using namespace cv;

void saltAndPepper(Mat image, int n) {

default\_random\_engine generator;

uniform\_int\_distribution<int> randomRow(0, image.rows - 1);

uniform\_int\_distribution<int> randomCol(0, image.cols - 1);

int i, j;

for (int k = 0; k < n; k++) {

// random image coordinate

i = randomCol(generator);

j = randomRow(generator);

if (image.type() == CV\_8UC1) { // gray-level image

// single-channel 8-bit image

image.at<uchar>(j, i) = 255;

}

else if (image.type() == CV\_8UC3) { // color image

// 3-channel image

image.at<Vec3b>(j, i)[0] = 255;

image.at<Vec3b>(j, i)[1] = 255;

image.at<Vec3b>(j, i)[2] = 255;

}

}

}

int main() {

Mat image = imread("lena.png", IMREAD\_COLOR);

Mat medianFilteredImage;

Mat meanFilteredImage;

saltAndPepper(image, 3000);

medianBlur(image, medianFilteredImage, 25);

blur(image, meanFilteredImage, Size(5, 5));

imwrite("saltAndPepper.png", image);

imwrite("medianFilteredImage.png", medianFilteredImage);

imwrite("meanFilteredImage.png", meanFilteredImage);

namedWindow("original", cv::WINDOW\_AUTOSIZE);

namedWindow("meanfilter", cv::WINDOW\_AUTOSIZE);

imshow("original", image);

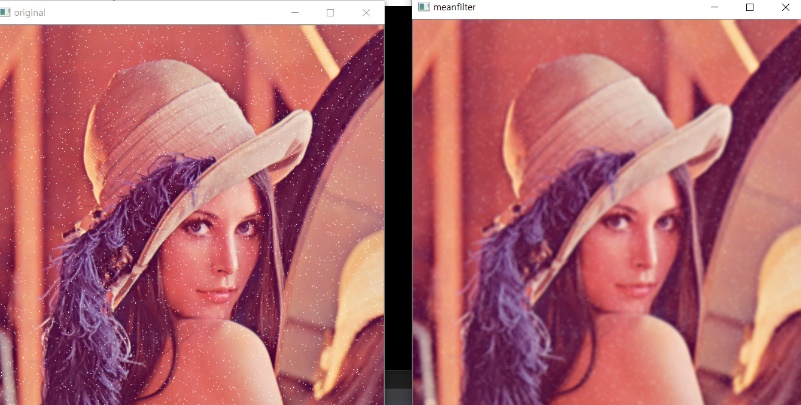
imshow("meanfilter", meanFilteredImage);

waitKey(0);

return 0;

}

**25%**

****