

[Computer Vision Programming] Lab 4. Image filtering

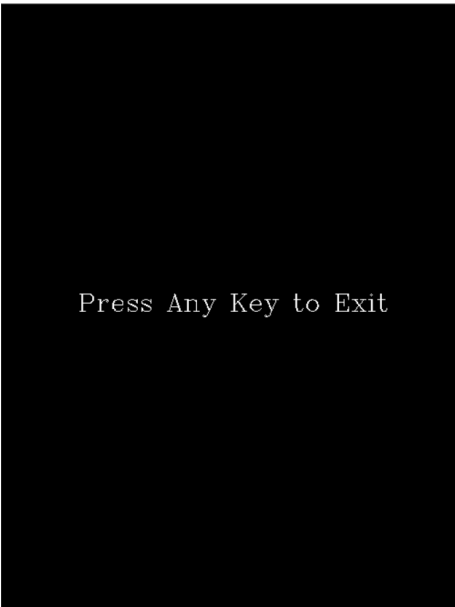
전자공학과 21611591 김난희

Lab 1 : OpenCV-Blur

-결과 사진

for문을 통해 Filter kernel 사이즈를 늘려가면서 테스트. 사이즈는 1부터 29까지 변화함. 점점 Blur 효과가 커지는 것을 볼 수 있음.



		
<p>Smoothed Image: 25x25 사이즈</p>	<p>Smoothed Image: 29x29 사이즈</p>	<p>for문 종료 후 아무 키나 눌러도 종료된다는 문구를 올림</p>

kernel 사이즈를 변경하는 for문 도중에 아무 키나 누를 경우 for문이 빨리 변화하며, Esc키를 누를 경우는 종료됨.

—소스 코드

for문은 Image의 blur효과를 보기 위함, for문 후에는 종료 안내를 위한 소스 코드.

```

1 //21611591 김난희
2 #include <opencv2/core.hpp>
3 #include "opencv2/imgproc/imgproc.hpp"
4 #include "opencv2/highgui/highgui.hpp"
5
6 //include <iostream>
7 #include <stdlib.h> // srand 함수 사용을 위해서
8 #include <time.h> // time 함수 사용을 위해서
9
10 using namespace cv;
11 //using namespace std;
57 int main(int argc, char** argv)
58 {
59     /***** 1. OpenCV-Blur *****/
60
61     //Load an image from file
62     Mat src = imread("myimg.jpg");
63
64     //show the loaded image
65     imshow("(Lab1) Original Image - 21611591 김난희", src);
66
67     //결과를 저장할 Mat 객체 선언
68     Mat dst;
69     char zBuffer[35];
70
71     for (int i = 1; i < 31; i = i + 2)
72     {
73         //copy the text to the "zBuffer"
74         _snprintf_s(zBuffer, 35, "Kernel Size : %d x %d", i, i);
75
76         //smooth the image in the "src" and save it to "dst"
77         blur(src, dst, Size(i, i), cv::Point(-1, -1), BORDER_REPLICATE); //Point(-1,-1): kernel center

```

```

79 //put the text in the "zBuffer" to the "dst" image
80 //사진과 함께 글자를 넣음, 내 사진에 맞게 위치와 사이즈 변경해줌
81 putText(dst, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX, 0.9, Scalar(255, 255, 255)
82 );
83
84 //show the blurred image with the text
85 imshow("(Lab1) Smoothed Image - 21611591 김난희", dst);
86
87 //wait for 2 seconds
88 int c = waitKey(2000);
89
90 //if the "esc" key is pressed during the wait, return
91 if (c == 27)
92 {
93     return 0;
94 }
95 }
96 //make the "dst" image, black
97 dst = Mat::zeros(src.size(), src.type());
98
99 //copy the text to the "zBuffer"
100 _snprintf_s(zBuffer, 35, "Press Any Key to Exit");
101
102 //사진을 마치고 종료키를 눌러 종료해도 된다는 글자를 넣음
103 //put the text in the "zBuffer" to the "dst" image
104 putText(dst, zBuffer, Point(src.cols / 6, src.rows / 2), CV_FONT_HERSHEY_COMPLEX, 0.9, Scalar(255, 255, 255));
105
106 //show the black image with the text
107 imshow("(Lab1) Smoothed Image - 21611591 김난희", dst);
108
109 //wait for a key press infinitely
110 //waitKey(0);

```

Lab 2: Median Filter

—결과 사진

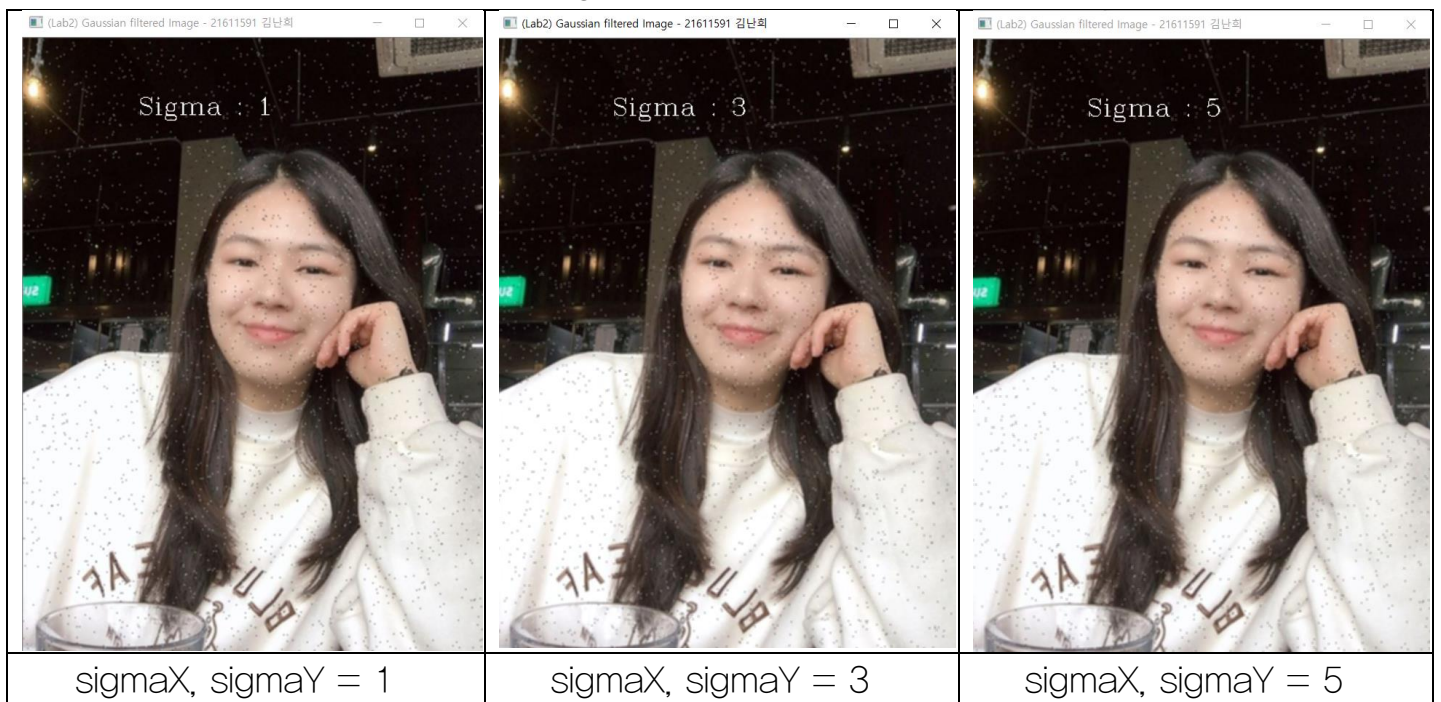
〈Filter Kernel Size 변화〉

for문을 통해 Filter kernel 사이즈를 늘려가면서 테스트. 사이즈는 1부터 9까지 변화함.





〈가우시안 필터링 경우 Sigma 변화〉 – Kernel Size는 (3,3)으로 동일



-Discussion

Median filter와 Gaussian filter의 kernel size를 증가할수록,

median filter는 3x3 size에서 salt and pepper noise가 완벽히 제거되어 보였다. 그 후로는 점점 edge가 뭉그러지는 모습을 보였다. image에서 픽셀에 접근하여 kernel size 안에서 중간 값을 다시 맵핑 하기 때문에, salt and pepper와 같은 0과 255로 이루어진 노이즈를 제거하기 쉽다.

gaussian filter는 noise가 끝까지 완벽히 제거된 모습을 볼 수 없었고, 점점 blur 효과가 커졌다.

Gaussian filter의 sigma를 변화시켰을 때는, (위 실험은 kernel size를 작게 잡아 큰 차이가 없어 보이지만,) 미세하게 점점 blur 효과가 됨을 볼 수 있었다.

사용한 GaussianBlur와 MedianBlur 함수 정의

void GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT)

각 파라미터들은 아래와 같은 의미를 갖는다.

- src: 소스 이미지
- dst: 목표 이미지, 즉 처리된 이미지를 담을 변수
- ksize: 가우시안 커널 사이즈로써 홀수여야 한다.
- sigmaX: 가우시안 커널의 x방향의 표준 편차
- sigmaY: 가우시안 커널의 y방향의 표준 편차

void medianBlur(InputArray src, OutputArray dst, int ksize)

각 파라미터들은 아래와 같은 의미를 갖는다.

- src: 소스 이미지
- dst: 목표 이미지, 즉 처리된 이미지를 담을 변수
- ksize: 커널 사이즈로써 홀수여야하고 1보다 커야 한다. 즉 3부터 가능하다.

매개변수 중에서 sigmaX, sigmaY 는 각각 X축, Y축 방향의 표준편차를 의미하며, sigmaX 의 값을 0 으로 설정하면 Kernel의 사이즈(ksize)에 따라 다음 식에 의해서 자동으로 계산된다.

$$\text{sigma} = 0.3 * ((\text{ksize} - 1) * 0.5 - 1) + 0.8$$

-소스 코드

```
13 void salt_pepper(cv::Mat image, int n) // 소금-후추 잡음 첨가 함수
14 {
15     int i, j;
16     srand((int)time(NULL));
17
18     for (int k = 0; k < n; k++)
19     {
20         i = rand() % image.cols; // 이미지의 열크기 내에서 랜덤 수 생성, x 좌표값
21         j = rand() % image.rows; // 이미지의 행크기 내에서 랜덤 수 생성, y 좌표값
22
23         //cout << "at (" << j << ", " << i << ")", add salt or pepper!" << endl; // 랜덤하게 결정된 픽셀 위치 출력, (x, y)
24
25         int salt_or_pepper = (rand() % 2) * 255; // 랜덤하게 0 또는 255, 0이면 후추, 255면 소금
26
27         if (image.type() == CV_8UC1) // 그레이레벨 영상이라면
28         {
29             image.at<uchar>(j, i) = salt_or_pepper; // 랜덤하게 선택된 픽셀에 0 또는 255를 대입
30         }
31         else if (image.type() == CV_8UC3) // 3채널 컬러 영상이라면
32         {
33             // 랜덤하게 선택된 픽셀에 0 또는 255를 대입, 흰색 또는 검정색을 만들기 위해 세 컬러 채널에 동일한 것이 들어감. (0, 0, 0) 또는 (255, 255, 255)
34             image.at<cv::Vec3b>(j, i)[0] = salt_or_pepper; // B 채널
35             image.at<cv::Vec3b>(j, i)[1] = salt_or_pepper; // G 채널
36             image.at<cv::Vec3b>(j, i)[2] = salt_or_pepper; // R 채널
37         }
38     }
39 }
40 }
```

아래 소스 코드는 main함수 내부이다.


```

112 /***** 2. Median Filter *****/
113
114 //Mat src = imread("myimg.jpg"); //Load an image from file
115 imshow("(Lab2) Original Image - 21611591 김난희", src); //Show the Original image
116
117 //---Noise 첨가---//
118 Mat spn_img; //salt and pepper noise를 담은 Mat 이미지 객체 선언함
119 src.copyTo(spn_img); //원본 이미지를 복사해 원본과 별개로 그 위에 noise를 씌우기 위함
120
121 salt_pepper(spn_img, 5000); //원본 image에 5000알의 소금 또는 후추를 뿌림
122 imshow("(Lab2) (salt and pepper) Noise Added Image - 21611591 김난희", spn_img); //Show salt and pepper noise image
123
124 //---Filtering---//
125 Mat MedianF_img; //noise 영상에서 median filter가 씌워진 결과를 보여줄 객체 선언
126 Mat GaussianF_img; //noise 영상에서 gaussian filter가 씌워진 결과를 보여줄 객체 선언
127
128 //char zBuffer[35]; //image에 글자를 띄울 buffer 선언
129
130 for (int i = 1; i <= 9; i += 2) //kernel size 변화 //kernel size는 홀수로 들어갈 수 있음
131 {
132     medianBlur(spn_img, MedianF_img, i); //median filter를 씌움, kernel size
133     GaussianBlur(spn_img, GaussianF_img, Size(i, i), 0); //gaussian filter를 씌움, kernel size, sigma 0이면 kernel size에 따라 자동 계산
134
135     _snprintf_s(zBuffer, 35, "Kernel Size : %d x %d", i, i); // kernel size에 따라 다르게 글자를 표시함
136     //사진과 함께 글자를 넣음, 내 사진에 맞게 위치와 사이즈 변경해줌
137     putText(MedianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX, 0.9, Scalar(255, 255, 255));
138     putText(GaussianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX, 0.9, Scalar(255, 255, 255));
139
140     imshow("(Lab2) Median filtered Image - 21611591 김난희", MedianF_img); //median filter를 씌운 이미지를 보여줌
141     imshow("(Lab2) Gaussian filtered Image - 21611591 김난희", GaussianF_img); //gaussian filter를 씌운 이미지를 보여줌
142
143     //wait for 2 seconds
144     int c = waitKey(2000);
145 }
146 for (int i = 1; i <= 5; i += 2) //sigma 변화 //sigma는 홀수로 가능
147 {
148     GaussianBlur(spn_img, GaussianF_img, Size(5, 5), i, i); //gaussian filter를 씌움, kernel size, sigma
149
150     _snprintf_s(zBuffer, 35, "Sigma : %d", i); // Sigma에 따라 다르게 글자를 표시함
151     //사진과 함께 글자를 넣음, 내 사진에 맞게 위치와 사이즈 변경해줌
152     putText(GaussianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX, 0.9, Scalar(255, 255, 255));
153
154     imshow("(Lab2) Gaussian filtered Image - 21611591 김난희", GaussianF_img); //gaussian filter를 씌운 이미지를 보여줌
155
156     //wait for 2 seconds
157     int c = waitKey(2000);
158 }

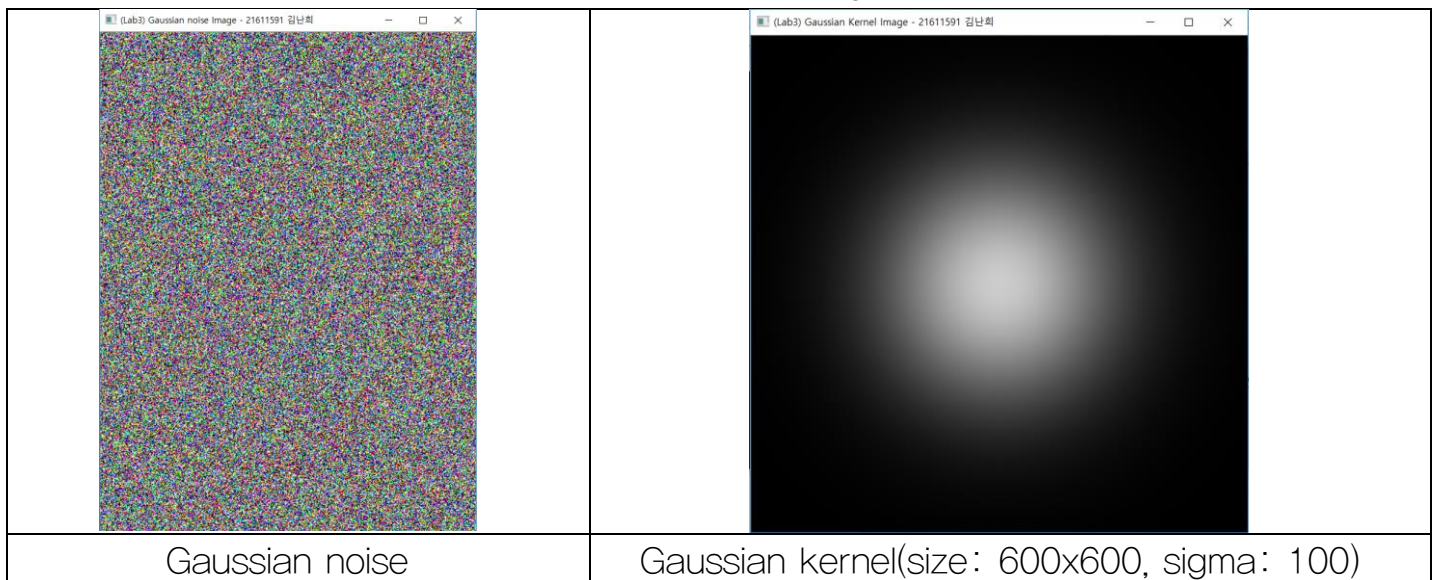
```

Lab 3: Gaussian Filter

이번에는 Gaussian filter의 큰 효과를 보기 위해, Gaussian noise를 선택하여 noise image를 생성함.

—결과 사진

〈Gaussian noise and kernel〉 kernel은 실제로는 5x5정도로 작으나, 눈에 보이지 않아 큰 사이즈(100배 확대)로 출력함. kernel size는 보통 sigma의 6배 정도로 사용함.



〈Filter Kernel Size 변화〉

			
Original Image	Added Gaussian Noise Image	Median Filtered Image: 1x1 사이즈	Gaussian Filtered Image: 1x1 사이즈
			
Median Filtered Image: 3x3 사이즈	Gaussian Filtered Image: 3x3 사이즈	Median Filtered Image: 5x5 사이즈	Gaussian Filtered Image: 5x5 사이즈
			
Median Filtered Image: 7x7 사이즈	Gaussian Filtered Image: 7x7 사이즈	Median Filtered Image: 9x9 사이즈	Gaussian Filtered Image: 9x9 사이즈

〈가우시안 필터일 경우 Sigma 변화〉 – Kernel Size는 (3,3)으로 동일



–Discussion

Median filter와 Gaussian filter의 kernel size를 증가할수록,

median filter는 중간 값으로 다시 맵핑 하는 과정이기 때문에 normal 분포를 따르는 sigma가 없어 kernel size 변화를 보여주었다. median은 노이즈가 점점 제거된다고 보기보단, edge가 뭉그러져서 노이즈가 보이지 않게 된다고 할 수 있다. 노이즈 분포가 가우시안을 따르기 때문에 median 필터로는 제거하기 충분하지 않았다. gaussian filter는 median filter만큼 edge를 죽이지 않고 차츰 노이즈를 제거하여, 7x7 size가 되었을 때부터 노이즈가 거의 보이지 않는다. 가우시안 노이즈를 제거하는 데에는 가우시안 필터를 사용하는 것이 큰 효과가 있음을 보여준다. kernel size를 sigma의 6배정도로 해서 둘을 잘 맞춰주면, 더욱 효과가 좋을 것이다.

Gaussian filter의 sigma를 변화시켰을 때는, (위 실험은 kernel size를 작게 잡아 큰 차이가 없어 보이지만,) 약간의 blur 효과와 함께 노이즈가 제거됨을 볼 수 있었다.

–소스 코드

```
41 Mat gaussian(cv::Mat Origin, cv::Mat dst, float sigma = 30.0) //블로그에서 30.0을 넣어놓아 default로 설정함
42 {
43     //---노이즈 image 생성---//
44     Mat noise(Origin.size(), CV_32FC3); //노이즈만 첨가된 이미지 생성 //3채널 double
45     float average = 0.0; //평균이 0이고
46     float std = sigma; //표준 편차가 sigma이도록 설정
47
48     //평균 average, 표준편차 std인 정규분포를 따르는 난수 생성 matrix 반환
49     randn(noise, Scalar::all(average), Scalar::all(std));
50     imshow("(Lab3) Gaussian Kernel Image - 21611591 김난희", noise);
51     Origin.convertTo(dst, CV_32FC3); //원본사진에서 dst image로 형변환 복사 //copyTo를 해줄 경우는 오류.
52     addWeighted(dst, 1.0, noise, 1.0, .0, dst); //dst(마지막 인자) = dst * 1.0 + noise * 1.0 + .0
53     dst.convertTo(dst, Origin.type()); //합성 후에 dst 자료형을 다시 원본 사진과 같게 해줌
54
55     return dst; //결과 image return
56 }
```

아래 소스 코드는 main함수 내부이다.


```

160 /***** 3. Gaussian Filter *****/
161 //Mat src = imread("myimg.jpg"); //Load an image from file
162 imshow("(Lab3) Original Image - 21611591 김난희", src); //Show the Original image
163
164 //---Noise 첨가---//
165 Mat gn_img; //gaussian noise를 담은 Mat 이미지 객체 선언함
166
167 gn_img = gaussian(src, gn_img, 20); //가우시안 노이즈 첨가 //sigma = 20
168 imshow("(Lab3) (gaussian) Noise Added Image - 21611591 김난희", gn_img); //Show salt and pepper noise image
169
170 //---Filtering---//
171 Mat MedianF_img; //noise 영상에서 median filter가 씌워진 결과를 보여줄 객체 선언
172 Mat GaussianF_img; //noise 영상에서 gaussian filter가 씌워진 결과를 보여줄 객체 선언
173
174 char zBuffer[35]; //image에 글자를 띄울 buffer 선언
175
176 for (int i = 1; i <= 9; i += 2) //kernel size 변화 //kernel size는 홀수로 들어갈 수 있음
177 {
178     medianBlur(gn_img, MedianF_img, i); //median filter를 씌움, kernel size
179     GaussianBlur(gn_img, GaussianF_img, Size(i, i), 0); //gaussian filter를 씌움, kernel size, sigma 0이면 kernel size에 따라 자동 계산
180
181     _snprintf_s(zBuffer, 35, "Kernel Size : %d x %d", i, i); // kernel size에 따라 다르게 글자를 표시함
182     //사진과 함께 글자를 넣음, 내 사진에 맞게 위치와 사이즈 변경해줌
183     putText(MedianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX, 0.9, Scalar(255, 255, 255));
184     putText(GaussianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX, 0.9, Scalar(255, 255, 255));
185
186     imshow("(Lab3) Median filtered Image - 21611591 김난희", MedianF_img); //median filter를 씌운 이미지를 보여줌
187     imshow("(Lab3) Gaussian filtered Image - 21611591 김난희", GaussianF_img); //gaussian filter를 씌운 이미지를 보여줌
188
189     //wait for 2 seconds
190     int c = waitKey(2000);
191 }
192 for (int i = 1; i <= 5; i += 2) //sigma 변화 //sigma는 홀수로 가능
193 {
194     GaussianBlur(gn_img, GaussianF_img, Size(5, 5), i, i); //gaussian filter를 씌움, kernel size, sigma
195
196     _snprintf_s(zBuffer, 35, "Sigma : %d", i); // Sigma에 따라 다르게 글자를 표시함
197     //사진과 함께 글자를 넣음, 내 사진에 맞게 위치와 사이즈 변경해줌
198     putText(GaussianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX, 0.9, Scalar(255, 255, 255));
199
200     imshow("(Lab3) Gaussian filtered Image - 21611591 김난희", GaussianF_img); //gaussian filter를 씌운 이미지를 보여줌
201
202     //wait for 2 seconds
203     int c = waitKey(2000);
204 }
205 //실제로는 kernel size가 5x5이고 sigma가 kernel size의 1/6배 정도로 하는데, 보이지 않으므로 100배 크게하여 보여줌
206 Mat kernelX = getGaussianKernel(600, 100); //kernel size, sigma, kernel type을 넣어줌 --> 가우시안 coefficient를 얻음
207 Mat kernelY = getGaussianKernel(600, 100);
208 Mat GaussianK = kernelX * kernelY.t(); //x축과 y축을 합쳐서 하나의 완전한 kernel이 만들어짐 //주파수는 합이 곱하기
209
210 normalize(GaussianK, GaussianK, 0.8, 0, NORM_MINMAX); //src, dst, alpha, beta, normalize // normalize 해주며 가우시안 분포를 따르도록 함
211 imshow("(Lab3) Gaussian Kernel Image - 21611591 김난희", GaussianK); //가우시안 커널을 imshow함
212
213 waitKey(0);

```

——전체 소스 코드——

```

//21611591 김난희
#include <opencv2/core.hpp>
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"

//#include <iostream>
#include <stdlib.h> // srand 함수 사용을 위해서
#include <time.h> // time 함수 사용을 위해서

using namespace cv;
//using namespace std;

void salt_pepper(cv::Mat image, int n) // 소금-후추 잡음 첨가 함수
{
    int i, j;
    srand((int)time(NULL));

```

```

    for (int k = 0; k < n; k++)
    {
        i = rand() % image.cols; // 이미지의 열크기 내에서 랜덤 수 생성, x 좌표값
        j = rand() % image.rows; // 이미지의 행크기 내에서 랜덤 수 생성, y 좌표값

        //cout << "at (" << j << ", " << i << " ), add salt or pepper!" << endl; // 랜덤하게 결정된
픽셀 위치 출력, (x, y)

        int salt_or_pepper = (rand() % 2) * 255; // 랜덤하게 0 또는 255, 0이면 후추, 255면 소금

        if (image.type() == CV_8UC1) // 그레이레벨 영상이라면
        {
            image.at<uchar>(j, i) = salt_or_pepper; // 랜덤하게 선택된 픽셀에 0 또는 255을 대입
        }
        else if (image.type() == CV_8UC3) // 3채널 컬러 영상이라면
        {
            // 랜덤하게 선택된 픽셀에 0 또는 255을 대입, 흰색 또는 검정색을 만들기 위해 세 컬러
채널에 동일한 것이 들어감. (0, 0, 0) 또는 (255, 255, 255)
            image.at<cv::Vec3b>(j, i)[0] = salt_or_pepper; // B 채널
            image.at<cv::Vec3b>(j, i)[1] = salt_or_pepper; // G 채널
            image.at<cv::Vec3b>(j, i)[2] = salt_or_pepper; // R 채널
        }
    }
}
Mat gaussian(cv::Mat Origin, cv::Mat dst, float sigma = 30.0) //블로그에서 30.0을 넣어놓아 defalut로 설정함
{
    //---노이즈 image 생성---//
    Mat noise(Origin.size(), CV_32FC3); //노이즈만 첨가된 이미지 생성 //3채널 double
    float average = 0.0; //평균이 0이고
    float std = sigma; //표준 편차가 sigma이도록 설정

    //평균 average, 표준편차 std인 정규분포를 따르는 난수 생성 matrix 반환
    randn(noise, Scalar::all(average), Scalar::all(std));
    imshow("(Lab3) Gaussian noise Image - 21611591 김난희", noise);

    Origin.convertTo(dst, CV_32FC3); //원본사진에서 dst image로 형변환 복사 //copyTo를 해줄 경우는 오류.
    addWeighted(dst, 1.0, noise, 1.0, .0, dst); //dst(마지막 인자) = dst * 1.0 + noise * 1.0 + .0
    dst.convertTo(dst, Origin.type()); //합성 후에 dst 자료형을 다시 원본 사진과 같게 해줌

    return dst; //결과 image return
}
int main(int argc, char** argv)
{
    /***** 1. OpenCV-Blur *****/

    //Load an image from file
    Mat src = imread("myimg.jpg");

    //show the loaded image
    imshow("(Lab1) Original Image - 21611591 김난희", src);

    //결과를 저장할 Mat 객체 선언
    Mat dst;
    char zBuffer[35];

    for (int i = 1; i < 31; i = i + 2)
    {
        //copy the text to the "zBuffer"
        _snprintf_s(zBuffer, 35, "Kernel Size : %d x %d", i, i);
    }
}

```



```

        //smooth the image in the "src" and save it to "dst"
        blur(src, dst, Size(i, i), cv::Point(-1, -1), BORDER_REPLICATE); //Point(-1,-1): kernel
center

        //put the text in the "zBuffer" to the "dst" image
        //사진과 함께 글자를 넣음, 내 사진에 맞게 위치와 사이즈 변경해줌
        putText(dst, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX, 0.9,
Scalar(255, 255, 255)
        );

        //show the blurred image with the text
        imshow("(Lab1) Smoothed Image - 21611591 김난희", dst);

        //wait for 2 seconds
        int c = waitKey(2000);

        //if the "esc" key is pressed during the wait, return
        if (c == 27)
        {
            return 0;
        }
    }
    //make the "dst" image, black
    dst = Mat::zeros(src.size(), src.type());

    //copy the text to the "zBuffer"
    _snprintf_s(zBuffer, 35, "Press Any Key to Exit");

    //사진을 마치고 종료키를 눌러 종료해도 된다는 글자를 넣음
    //put the text in the "zBuffer" to the "dst" image
    putText(dst, zBuffer, Point(src.cols / 6, src.rows / 2), CV_FONT_HERSHEY_COMPLEX, 0.9, Scalar(255,
255, 255));

    //show the black image with the text
    imshow("(Lab1) Smoothed Image - 21611591 김난희", dst);

    //wait for a key press infinitely
    //waitKey(0);

    /***** 2. Median Filter *****/

    //Mat src = imread("myimg.jpg"); //Load an image from file
    imshow("(Lab2) Original Image - 21611591 김난희", src); //Show the Original image

    //---Noise 첨가---//
    Mat spn_img; //salt and pepper noise를 담을 Mat 이미지 객체 선언함
    src.copyTo(spn_img); //원본 이미지를 복사해 원본과 별개로 그 위에 noise를 씌우기 위함

    salt_pepper(spn_img, 5000); //원본 image에 5000알의 소금 또는 후추를 뿌림
    imshow("(Lab2) (salt and pepper) Noise Added Image - 21611591 김난희", spn_img); //Show salt and
pepper noise image

    //---Filtering---//
    Mat MedianF_img; //noise 영상에서 median filter가 씌워진 결과를 보여줄 객체 선언
    Mat GaussianF_img; //noise 영상에서 gaussian filter가 씌워진 결과를 보여줄 객체 선언

    //char zBuffer[35]; //image에 글자를 띄울 buffer 선언

    for (int i = 1; i <= 9; i += 2) //kernel size 변화 //kernel size는 홀수로 들어갈 수 있음
    {
        medianBlur(spn_img, MedianF_img, i); //median filter를 씌움, kernel size

```

```

        GaussianBlur(spn_img, GaussianF_img, Size(i, i), 0); //gaussian filter를 씌움, kernel size,
sigma 0이면 kernel size에 따라 자동 계산

        _snprintf_s(zBuffer, 35, "Kernel Size : %d x %d", i, i); // kernel size에 따라 다르게 글자를
표시함

        //사진과 함께 글자를 넣음, 내 사진에 맞게 위치와 사이즈 변경해줌
        putText(MedianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX,
0.9, Scalar(255, 255, 255));
        putText(GaussianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX,
0.9, Scalar(255, 255, 255));

        imshow("(Lab2) Median filtered Image - 21611591 김난희", MedianF_img); //median filter를 씌운
이미지를 보여줌
        imshow("(Lab2) Gaussian filtered Image - 21611591 김난희", GaussianF_img); //gaussian
filter를 씌운 이미지를 보여줌

        //wait for 2 seconds
        int c = waitKey(2000);
    }
    for (int i = 1; i <= 5; i += 2) //sigma 변화 //sigma는 홀수로 가능
    {
        GaussianBlur(spn_img, GaussianF_img, Size(5, 5), i, i); //gaussian filter를 씌움, kernel
size, sigma

        _snprintf_s(zBuffer, 35, "Sigma : %d", i); // Sigma에 따라 다르게 글자를 표시함
        //사진과 함께 글자를 넣음, 내 사진에 맞게 위치와 사이즈 변경해줌
        putText(GaussianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX,
0.9, Scalar(255, 255, 255));

        imshow("(Lab2) Gaussian filtered Image - 21611591 김난희", GaussianF_img); //gaussian
filter를 씌운 이미지를 보여줌

        //wait for 2 seconds
        int c = waitKey(2000);
    }

    /***** 3. Gaussian Filter *****/
    //Mat src = imread("myimg.jpg"); //Load an image from file
    imshow("(Lab3) Original Image - 21611591 김난희", src); //Show the Original image

    //---Noise 첨가---//
    Mat gn_img; //gaussian noise를 담은 Mat 이미지 객체 선언함

    gn_img = gaussian(src, gn_img, 20); //가우시안 노이즈 첨가 //sigma = 20
    imshow("(Lab3) (gaussian) Noise Added Image - 21611591 김난희", gn_img); //Show salt and pepper
noise image

    //---Filtering---//
    //Mat MedianF_img; //noise 영상에서 median filter가 씌워진 결과를 보여줄 객체 선언
    //Mat GaussianF_img; //noise 영상에서 gaussian filter가 씌워진 결과를 보여줄 객체 선언

    //char zBuffer[35]; //image에 글자를 띄울 buffer 선언

    for (int i = 1; i <= 9; i += 2) //kernel size 변화 //kernel size는 홀수로 들어갈 수 있음
    {
        medianBlur(gn_img, MedianF_img, i); //median filter를 씌움, kernel size
        GaussianBlur(gn_img, GaussianF_img, Size(i, i), 0); //gaussian filter를 씌움, kernel size,
sigma 0이면 kernel size에 따라 자동 계산

        _snprintf_s(zBuffer, 35, "Kernel Size : %d x %d", i, i); // kernel size에 따라 다르게 글자를
표시함

        //사진과 함께 글자를 넣음, 내 사진에 맞게 위치와 사이즈 변경해줌

```



```

        putText(MedianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX,
0.9, Scalar(255, 255, 255));
        putText(GaussianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX,
0.9, Scalar(255, 255, 255));

        imshow("(Lab3) Median filtered Image - 21611591 김난희", MedianF_img); //median filter를 씌운
이미지를 보여줌
        imshow("(Lab3) Gaussian filtered Image - 21611591 김난희", GaussianF_img); //gaussian
filter를 씌운 이미지를 보여줌

        //wait for 2 seconds
        int c = waitKey(2000);
    }
    for (int i = 1; i <= 5; i += 2) //sigma 변화 //sigma는 홀수로 가능
    {
        GaussianBlur(gn_img, GaussianF_img, Size(5, 5), i, i); //gaussian filter를 씌움, kernel size,
sigma

        _snprintf_s(zBuffer, 35, "Sigma : %d", i); // Sigma에 따라 다르게 글자를 표시함
        //사진과 함께 글자를 넣음, 내 사진에 맞게 위치와 사이즈 변경해줌
        putText(GaussianF_img, zBuffer, Point(src.cols / 4, src.rows / 8), CV_FONT_HERSHEY_COMPLEX,
0.9, Scalar(255, 255, 255));

        imshow("(Lab3) Gaussian filtered Image - 21611591 김난희", GaussianF_img); //gaussian
filter를 씌운 이미지를 보여줌

        //wait for 2 seconds
        int c = waitKey(2000);
    }

    //실제로는 kernel size가 5x5이고 sigma가 kernel size의 1/6배 정도로 하는데, 보이지 않으므로 100배
크게하여 보여줌
    Mat kernelX = getGaussianKernel(600, 100); //kernel size, sigma, kernel type을 넣어줌 --> 가우시안
coefficient를 얻음
    Mat kernelY = getGaussianKernel(600, 100);
    Mat GaussianK = kernelX * kernelY.t(); //x축과 y축을 합쳐서 하나의 완전한 kernel이 만들어짐
//주파수는 합이 곱하기

    normalize(GaussianK, GaussianK, 0.8, 0, NORM_MINMAX); //src, dst, alpha, beta, nomalize // nomalize
해주며 가우시안 분포를 따르도록 함
    imshow("(Lab3) Gaussian Kernel Image - 21611591 김난희", GaussianK); //가우시안 커널을 imshow함

    waitKey(0);

    return 0;
}

```

——참고 사이트——

(1) Add noise

<https://kgh1994.tistory.com/6> (randn)

<https://bskyvision.com/303> (salt and pepper noise)

<https://swprog.tistory.com/entry/OpenCV-%EC%9E%A1%EC%9D%8C%EC%9D%B4-%EB%93%A4%EC%96%B4%EA%B0%84-%EC%82%AC%EC%A7%84-%EB%A7%8C%EB%93%A4%EA%B8%B0?category=638899> (gaussian noise)

<https://ghj1001020.tistory.com/567> (gaussian noise 참고)

<https://stackoverflow.com/questions/43931749/adding-gaussian-noise-in-image-opencv-and-c-and-then-denoised>
(gaussian noise 참고)

(2) Image copy

<https://blog.naver.com/pckbj123/100202476334> (copyTo)
<https://darkpgmr.tistory.com/46>(convertTo)

(3) filter

<http://egloos.zum.com/eyes33/v/6091120> (gaussian filter의 sigma)

<https://bskyvision.com/24> (median, gaussian 함수 설명)

<https://swprog.tistory.com/entry/OpenCV-Noise%EC%A0%9C%EA%B1%B0%ED%95%98%EA%B8%B0-median-filtering> (median, gaussian 함수 사용 예시)

[https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#void%20sepFilter2D\(InputArray%20src,%20OutputArray%20dst,%20int%20ddepth,%20InputArray%20kernelX,%20InputArray%20kernelY,%20Point%20anchor,%20double%20delta,%20int%20borderType\)](https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#void%20sepFilter2D(InputArray%20src,%20OutputArray%20dst,%20int%20ddepth,%20InputArray%20kernelX,%20InputArray%20kernelY,%20Point%20anchor,%20double%20delta,%20int%20borderType)) (gaussian kernel)

<https://answers.opencv.org/question/86801/normalize-image-0-255-for-display/> (gaussian kernel)