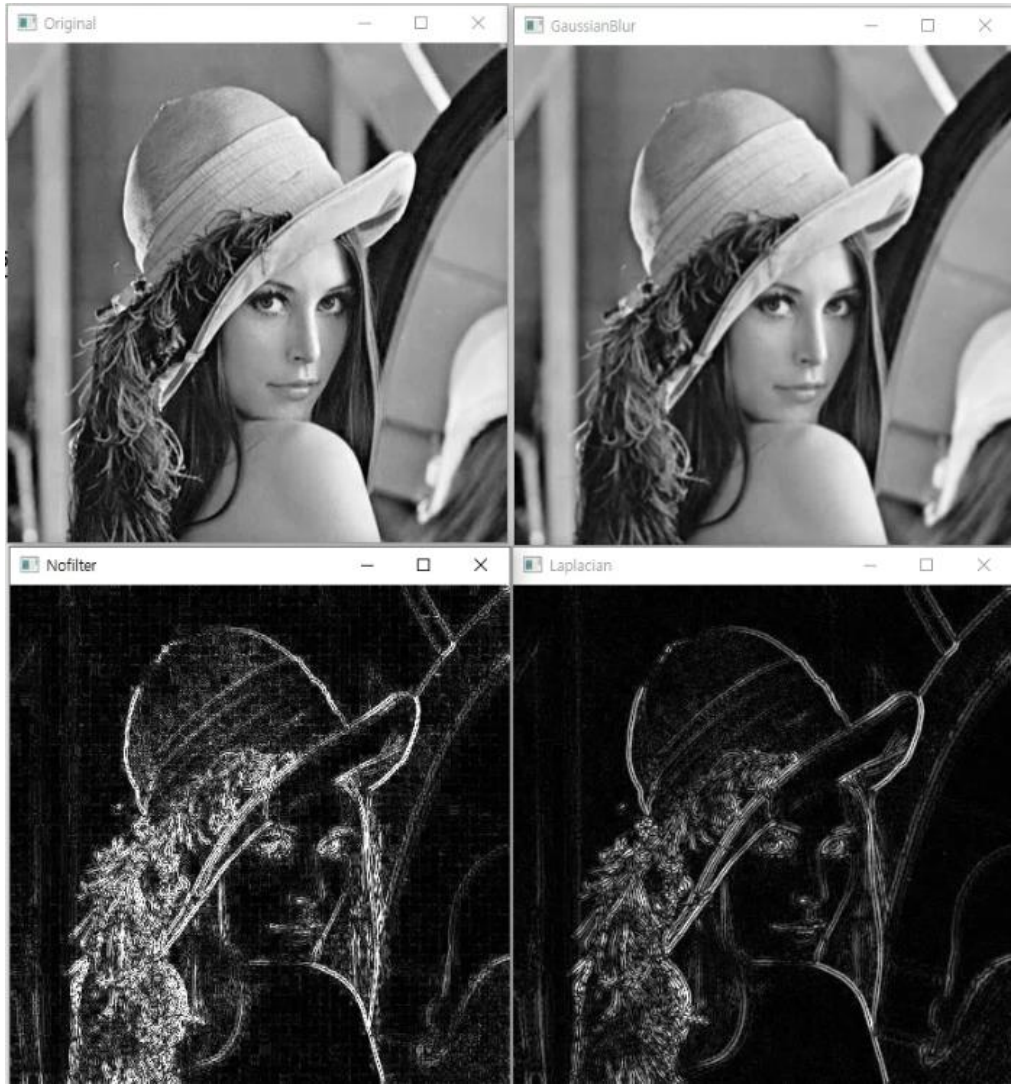
영상처리 실제 - 6주차 과제

2023254009 최현동

: 10 – 공간필터링 – HW1

- 필터링을 하지 않을 경우, 노이즈로 인해 정확한 엣지 검출이 되지 않는다.

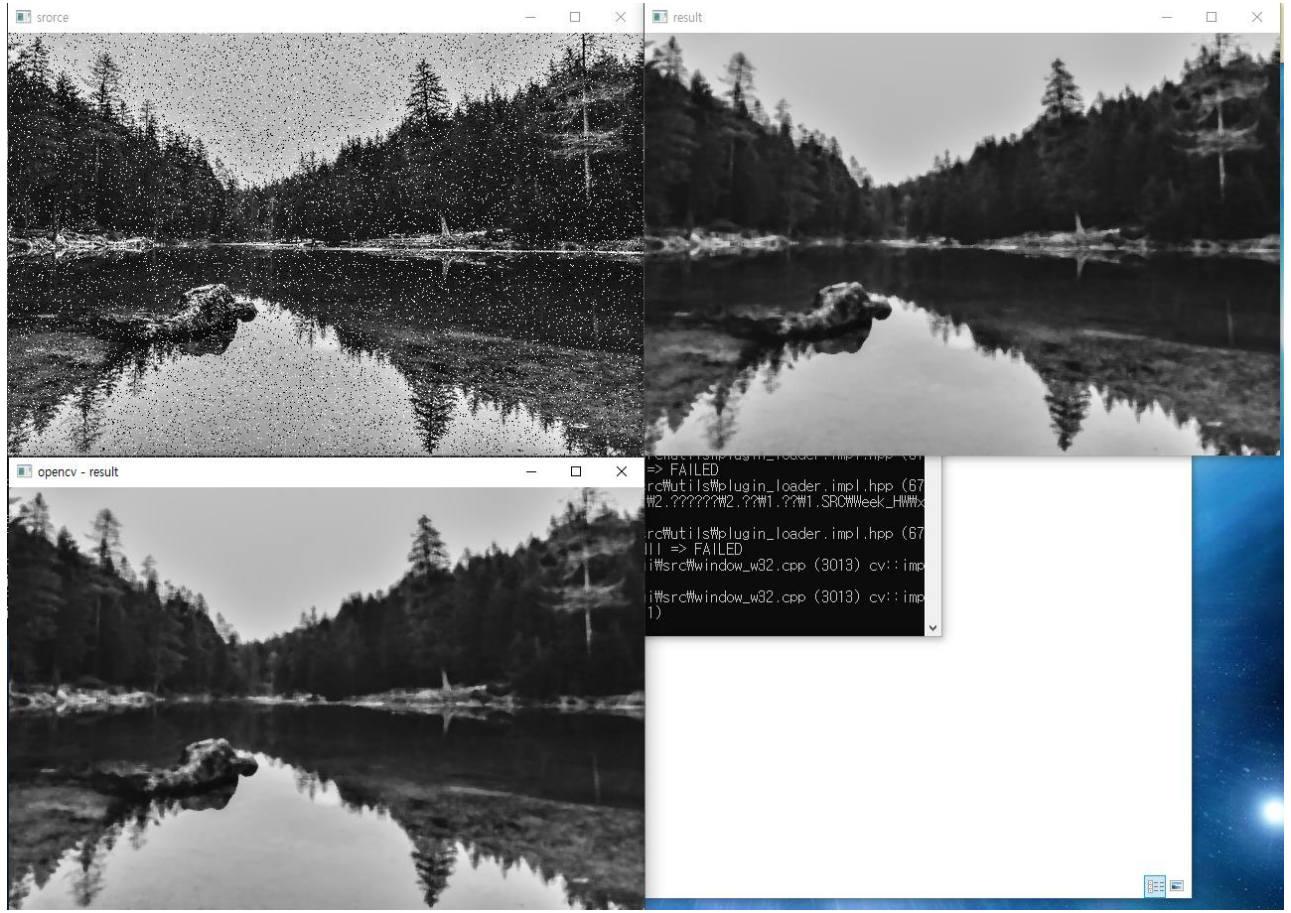
```
    //10-공간필터링 - HW1
#if 1
    Mat src, src_gray, dst, dst_Nofilter;
    int kernel_size = 3;
    int scale = 1;
    int delta = 0;
    int ddepth = CV_16S;
    src = imread("D:\\999.Image\\lenna.jpg", IMREAD_GRAYSCALE);
    if (src.empty())
    {
        return -1;
    }
    Mat src_Nofilter = src.clone();
    GaussianBlur(src, src, Size(3, 3), 0, 0, BORDER_DEFAULT);
    Mat abs_dst;
    Mat abs_dst_Nofilter;
    Laplacian(src, dst, ddepth, kernel_size, scale, delta, BORDER_DEFAULT);
    Laplacian(src_Nofilter, dst_Nofilter, ddepth, kernel_size, scale, delta, BORDER_DEFAULT);
    convertScaleAbs(dst, abs_dst);
    convertScaleAbs(dst_Nofilter, abs_dst_Nofilter);
    imshow("Original", src_Nofilter);
    imshow("GaussianBlur", src);
    imshow("Laplacian", abs_dst);
    imshow("Nofilter", abs_dst_Nofilter);
    waitKey();
#endif
```



: 10 – 공간필터링 – HW2

```cpp
void medianFilter(Mat input, Mat& output, int ksize)
{
    vector<uchar> neighbors;
    uchar sample = 0;
    uchar median = 0;

    int nType = output.type();

    for (int y = 0; y < output.rows; y++)
    {
        for (int x = 0; x < output.cols; x++)
        {
            for (int s = 0; s < ksize; s++)
            {
                for (int t = 0; t < ksize; t++)
                {

                    //padding
                    sample = input.at<uchar>(min(output.rows - 1, max(0, y + t)), min(output.cols - 1, max(0, x + s)));
                    neighbors.push_back(sample);
                }
            }
        //find median value >dst(y,x)대입
        sort(neighbors.begin(), neighbors.end());
        median = neighbors[neighbors.size() / 2];
        output.at<uchar>(y, x) = median;
        neighbors.clear();
        }
    }
}
```

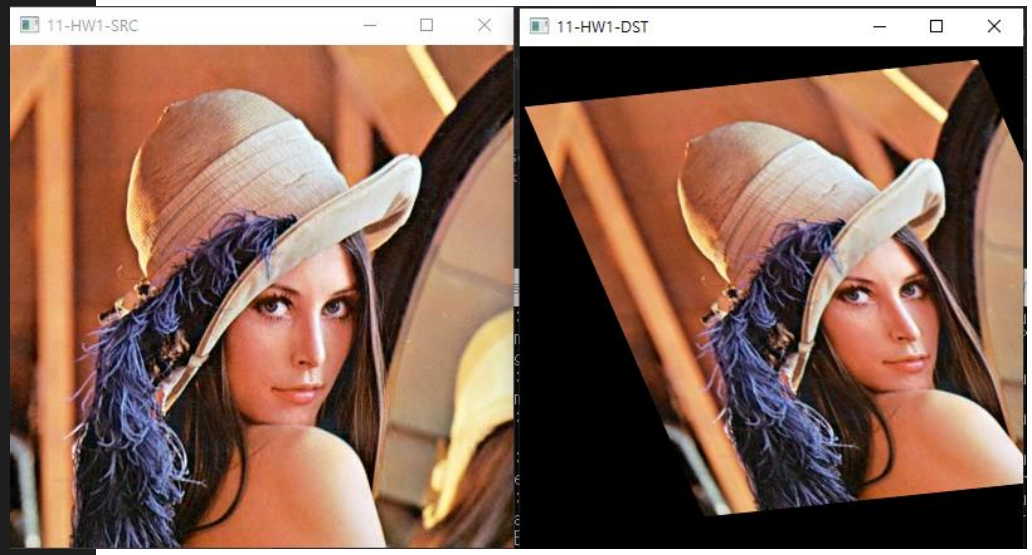
```cpp
    //10-공간필터링 - HW2
#if 1
    Mat src = imread("D:\\999.Image\\city1.jpg", IMREAD_GRAYSCALE);
    if (src.empty())
    {
        return -1;
    }
    Mat dst(src.size(), src.type());
    Mat dst_opencv;
    Mat noise_img = Mat::zeros(src.rows, src.cols, CV_8U);
    randu(noise_img, 0, 255);//noise_img의 모든화소를 0~255까지의 난수로 채움

    Mat black_img = noise_img < 10; // noise_img의 화소값이 10 보다 작으면 1이되는 black_img 생성
    Mat white_img = noise_img > 245; // noise_img의 화소값이 245 보다 크면 1이되는 white_img 생성

    Mat src1 = src.clone();
    src1.setTo(255, white_img); //white_img의 화소값이 1이면 src1의 화소값을 255로 한다. salt noise
    src1.setTo(0, black_img); //black_img의 화소값이 1이면 src1의 화소값을 0로 한다. pepper noise
    medianFilter(src1, dst, 5);
    medianBlur(src1, dst_opencv, 5);
    imshow("srorce", src1);
    imshow("result", dst);
    imshow("opencv - result", dst_opencv);
    waitKey();
#endif
```



: 11 – 기하학적변환 – HW1

```cpp
Mat _11_HW1_src;
Mat warp_mat(2, 3, CV_32FC1);
Mat warp_dst;
int nMouseClickCount_11_HW1_src = 0;
int nMouseClickCount_11_HW1_dst = 0;
Point2f nMousePt_11_HW1_src[3];
Point2f nMousePt_11_HW1_dst[3];

void onMouse_11_HW1_src(int event, int x, int y, int flags, void* param)
{
    if (event == EVENT_LBUTTONDOWN)
    {
        // 마우스의 왼쪽 버튼을 누르면
        switch (nMouseClickCount_11_HW1_src)
        {
        case 0:
            nMousePt_11_HW1_src[0].x = x;
            nMousePt_11_HW1_src[0].y = y;
            nMouseClickCount_11_HW1_src++;
            break;
        case 1:
            nMousePt_11_HW1_src[1].x = x;
            nMousePt_11_HW1_src[1].y = y;
            nMouseClickCount_11_HW1_src++;
            break;
        case 2:
            nMousePt_11_HW1_src[2].x = x;
            nMousePt_11_HW1_src[2].y = y;
            nMouseClickCount_11_HW1_src = 0;
            break;
        }

    }
}
```

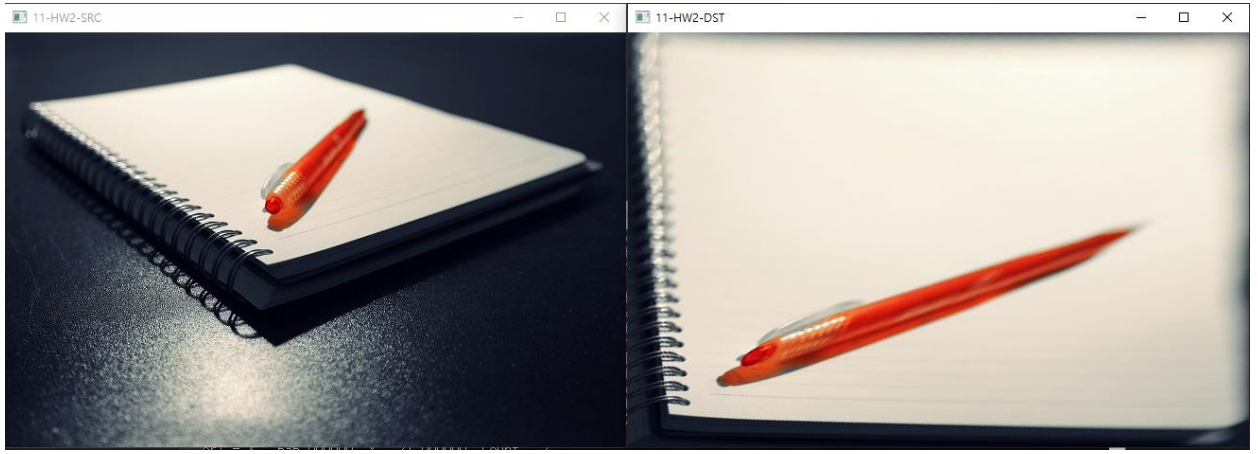
```cpp
void onMouse_11_HW1_dst(int event, int x, int y, int flags, void* param)
{
    if (event == EVENT_LBUTTONDOWN)
    {
        // 마우스의 왼쪽 버튼을 누르면
        switch (nMouseClickCount_11_HW1_dst)
        {
        case 0:
            nMousePt_11_HW1_dst[0].x = x;
            nMousePt_11_HW1_dst[0].y = y;
            nMouseClickCount_11_HW1_dst++;
            break;
        case 1:
            nMousePt_11_HW1_dst[1].x = x;
            nMousePt_11_HW1_dst[1].y = y;
            nMouseClickCount_11_HW1_dst++;
            break;
        case 2:
            nMousePt_11_HW1_dst[2].x = x;
            nMousePt_11_HW1_dst[2].y = y;
            nMouseClickCount_11_HW1_dst = 0;

            warp_mat = getAffineTransform(nMousePt_11_HW1_src, nMousePt_11_HW1_dst);
            warpAffine(_11_HW1_src, warp_dst, warp_mat, warp_dst.size());
            imshow("11-HW1-DST", warp_dst);
            waitKey();

            break;
        }
    }
}
```




```cpp
    //11- 기하학적변환 - HW1
#if 1
    _11_HW1_src = imread("D:\\999.Image\\lenna.jpg");
    warp_dst = Mat::zeros(_11_HW1_src.rows, _11_HW1_src.cols, _11_HW1_src.type());
    imshow("11-HW1-SRC", _11_HW1_src);
    imshow("11-HW1-DST", warp_dst);
    setMouseCallback("11-HW1-SRC", onMouse_11_HW1_src, 0);
    setMouseCallback("11-HW1-DST", onMouse_11_HW1_dst, 0);
    waitKey();
#endif
```

: 11 – 기하학적변환 – HW2

```
    //11- 기하학적변환 - HW2
#if 1
    _11_HW2_src = imread("D:\\999.Image\\book.jpg");
    perspective_dst = Mat::zeros(_11_HW2_src.rows, _11_HW2_src.cols, _11_HW2_src.type());
    imshow("11-HW2-SRC", _11_HW2_src);
    imshow("11-HW2-DST", perspective_dst);
    setMouseCallback("11-HW2-SRC", onMouse_11_HW2_src, 0);
    setMouseCallback("11-HW2-DST", onMouse_11_HW2_dst, 0);
    waitKey();
#endif

Mat _11_HW2_src;
Mat perspective_Transform;
Mat perspective_dst;
int nMouseClickCount_11_HW2_src = 0;
int nMouseClickCount_11_HW2_dst = 0;
Point2f nMousePt_11_HW2_src[4];
Point2f nMousePt_11_HW2_dst[4];
void onMouse_11_HW2_src(int event, int x, int y, int flags, void* param)
{
    if (event == EVENT_LBUTTONDOWN)
    {
        // 마우스의 왼쪽 버튼을 누르면
        switch (nMouseClickCount_11_HW2_src)
        {
        case 0:
            nMousePt_11_HW2_src[0].x = x;
            nMousePt_11_HW2_src[0].y = y;
            nMouseClickCount_11_HW2_src++;

            printf("ori - X : %f, Y : %f, Count : %d\n", nMousePt_11_HW2_src[0].x, nMousePt_11_HW2_src[0].y, nMouseClickCount_11_HW2_src);
            break;
        case 1:
            nMousePt_11_HW2_src[1].x = x;
            nMousePt_11_HW2_src[1].y = y;
            nMouseClickCount_11_HW2_src++;
            printf("ori - X : %f, Y : %f, Count : %d\n", nMousePt_11_HW2_src[1].x, nMousePt_11_HW2_src[1].y, nMouseClickCount_11_HW2_src);
            break;
        case 2:
            nMousePt_11_HW2_src[2].x = x;
            nMousePt_11_HW2_src[2].y = y;
            nMouseClickCount_11_HW2_src++;
            printf("ori - X : %f, Y : %f, Count : %d\n", nMousePt_11_HW2_src[2].x, nMousePt_11_HW2_src[2].y, nMouseClickCount_11_HW2_src);
            break;
        case 3:
            nMousePt_11_HW2_src[3].x = x;
            nMousePt_11_HW2_src[3].y = y;
            nMouseClickCount_11_HW2_src = 0;
            printf("ori - X : %f, Y : %f, Count : %d\n", nMousePt_11_HW2_src[3].x, nMousePt_11_HW2_src[3].y, nMouseClickCount_11_HW2_src);
            break;
        }

    }
}
```




```
void onMouse_11_HW2_dst(int event, int x, int y, int flags, void* param)
{
    if (event == EVENT_LBUTTONDOWN)
    {
        // 마우스의 왼쪽 버튼을 누르면
        switch (nMouseClickCount_11_HW2_dst)
        {
        case 0:
            nMousePt_11_HW2_dst[0].x = x;
            nMousePt_11_HW2_dst[0].y = y;
            nMouseClickCount_11_HW2_dst++;

            printf("dst - X : %f, Y : %f, Count : %d\n", nMousePt_11_HW2_dst[0].x, nMousePt_11_HW2_dst[0].y, nMouseClickCount_11_HW2_dst);
            break;
        case 1:
            nMousePt_11_HW2_dst[1].x = x;
            nMousePt_11_HW2_dst[1].y = y;
            nMouseClickCount_11_HW2_dst++;
            printf("dst - X : %f, Y : %f, Count : %d\n", nMousePt_11_HW2_dst[1].x, nMousePt_11_HW2_dst[1].y, nMouseClickCount_11_HW2_dst);
            break;
        case 2:
            nMousePt_11_HW2_dst[2].x = x;
            nMousePt_11_HW2_dst[2].y = y;
            nMouseClickCount_11_HW2_dst++;
            printf("dst - X : %f, Y : %f, Count : %d\n", nMousePt_11_HW2_dst[2].x, nMousePt_11_HW2_dst[2].y, nMouseClickCount_11_HW2_dst);
            break;
        case 3:
            nMousePt_11_HW2_dst[3].x = x;
            nMousePt_11_HW2_dst[3].y = y;
            nMouseClickCount_11_HW2_dst = 0;
            printf("dst - X : %f, Y : %f, Count : %d\n", nMousePt_11_HW2_dst[3].x, nMousePt_11_HW2_dst[3].y, nMouseClickCount_11_HW2_dst);

            perspective_Transform = getPerspectiveTransform(nMousePt_11_HW2_src, nMousePt_11_HW2_dst);
            warpPerspective(_11_HW2_src, perspective_dst, perspective_Transform, _11_HW2_src.size());
            imshow("11-HW2-DST", perspective_dst);
            waitKey();

            break;
        }

    }
}
```