영상처리 실제 - 6주차 실습

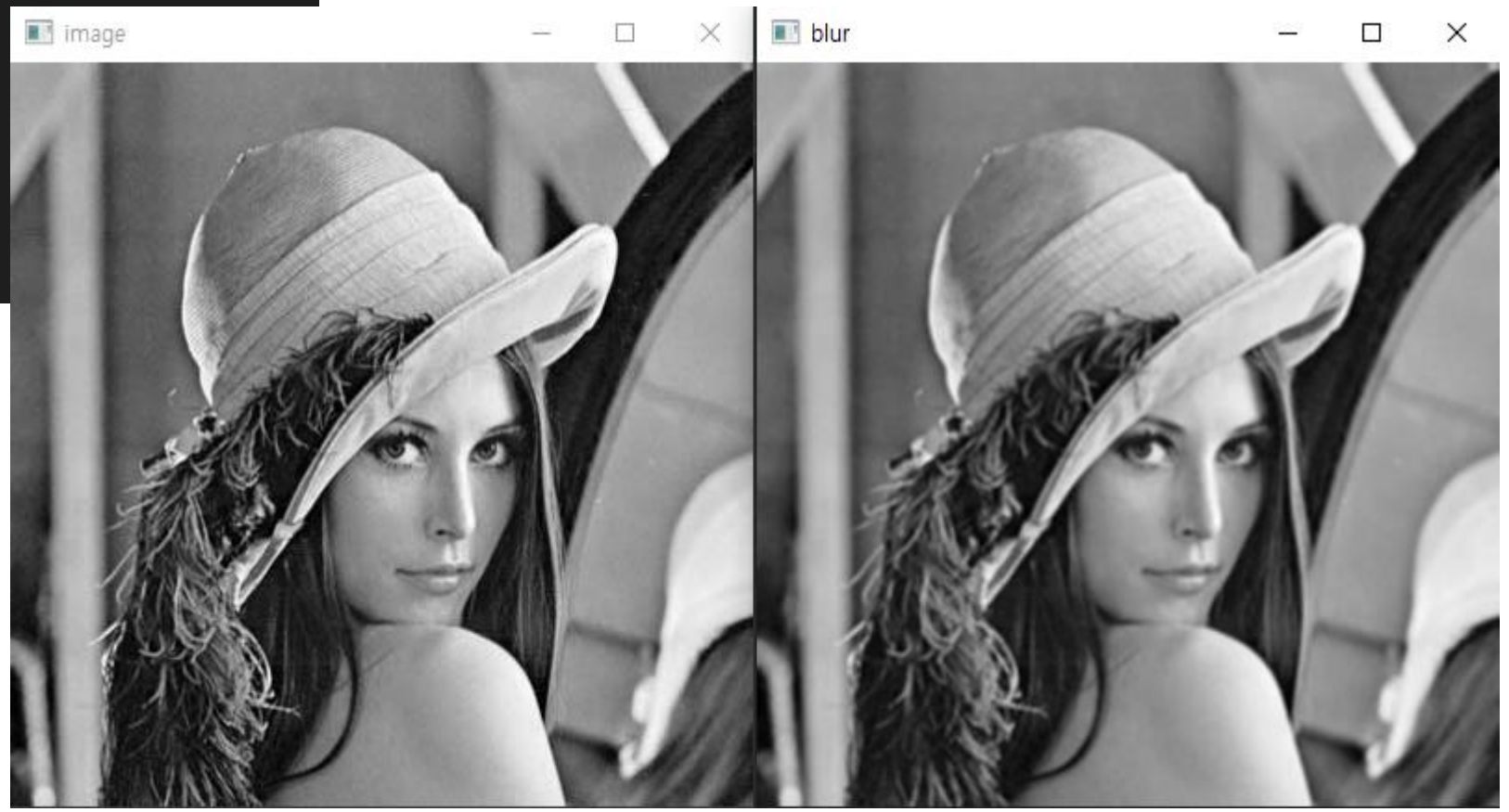
2023254009 최현동

: 10.공간필터링 – p.8

```
void filter(Mat img, Mat& dst, Mat mask)
{
    dst = Mat(img.size(), CV_32F, Scalar(0));
    Point h_m = mask.size() / 2;

    for (int i = h_m.y; i < img.rows - h_m.y; i++)
    {
        for (int j = h_m.x; j < img.cols - h_m.x; j++)
        {
            float sum = 0;
            for (int u = 0; u < mask.rows; u++) //마스크 원소 회순
            {
                for (int v = 0; v < mask.cols; v++)
                {
                    int y = i + u - h_m.y;
                    int x = j + v - h_m.x;
                    sum += mask.at<float>(u, v) * img.at<uchar>(y, x);//회선수식
                }
            }
            dst.at<float>(i, j) = sum;
        }
    }
}
    //공간필터링 p.8
#if 1
    Mat image = imread("D:\\999.Image\\filter_blur.jpg", IMREAD_GRAYSCALE);
    CV_Assert(image.data);

    float data[] = {
        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 mask(3, 3, CV_32F, data);
    Mat blur;
    filter(image, blur, mask);
    blur.convertTo(blur, CV_8U);

    imshow("image", image);
    imshow("blur", blur);
    waitKey();
#endif
```

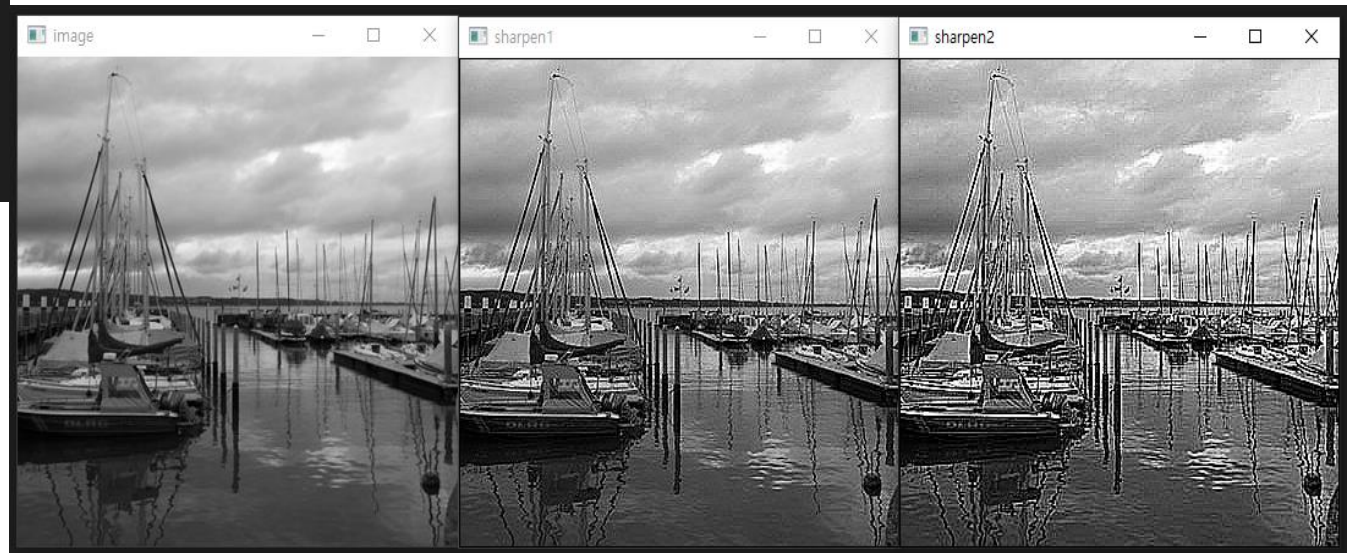
: 10.공간필터링 – p.10

```
//공간필터링 p.10
#if 1
    Mat image = imread("D:\\999.Image\\lenna.jpg", IMREAD_GRAYSCALE);

    float weight[] = {
        1 / 9.0f, 1 / 9.0f , 1 / 9.0f,
        1 / 9.0f, 1 / 9.0f , 1 / 9.0f,
        1 / 9.0f, 1 / 9.0f , 1 / 9.0f
    };
    Mat mask(3, 3, CV_32F, weight);
    Mat blur;
    filter2D(image, blur, -1, mask);
    blur.convertTo(blur, CV_8U);
    imshow("image", image);
    imshow("blur", blur);
    waitKey();
#endif
```



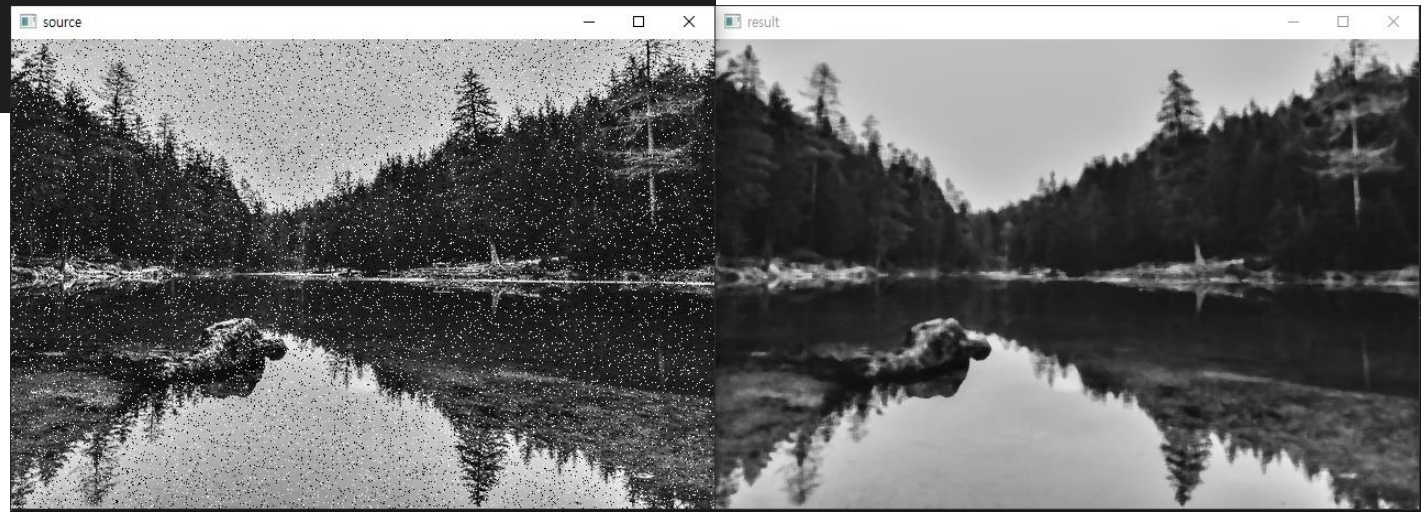
: 10.공간필터링 – p.16

```
void filter(Mat img, Mat& dst, Mat mask)
{
    dst = Mat(img.size(), CV_32F, Scalar(0));
    Point h_m = mask.size() / 2;

    for (int i = h_m.y; i < img.rows - h_m.y; i++)
    {
        for (int j = h_m.x; j < img.cols - h_m.x; j++)
        {
            float sum = 0;
            for (int u = 0; u < mask.rows; u++) //마스크 원소 회순
            {
                for (int v = 0; v < mask.cols; v++)
                {
                    int y = i + u - h_m.y;
                    int x = j + v - h_m.x;
                    sum += mask.at<float>(u, v) * img.at<uchar>(y, x);//회선수식
                }
            }
            dst.at<float>(i, j) = sum;
        }
    }
}
    //공간필터링 p.16
#if 1
    Mat image = imread("D:\\999.Image\\filter_sharpen.jpg", IMREAD_GRAYSCALE);

    CV_Assert(image.data);

    float data1[] = {
        0, -1, 0,
        -1, 5, -1,
        0, -1, 0,
    };
    float data2[] = {
        -1, -1, -1,
        -1, 9, -1,
        -1, -1, -1,
    };

    Mat mask1(3, 3, CV_32F, data1);
    Mat mask2(3, 3, CV_32F, data2);

    Mat sharpen1, sharpen2;
    filter(image, sharpen1, mask1);
    filter(image, sharpen2, mask2);
    sharpen1.convertTo(sharpen1, CV_8U);
    sharpen2.convertTo(sharpen2, CV_8U);
    imshow("image", image);
    imshow("sharpen1", sharpen1);
    imshow("sharpen2", sharpen2);
    waitKey();
#endif
```



: 10.공간필터링 – p.21

```
    //공간필터링 p.21
#if 1
    Mat src = imread("D:\\999.Image\\city1.jpg", IMREAD_GRAYSCALE);
    if (src.empty())
    {
        return -1;
    }
    Mat dst;
    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
    medianBlur(src1, dst, 5);
    imshow("srorce", src1);
    imshow("result", dst);
    waitKey();
#endif
```



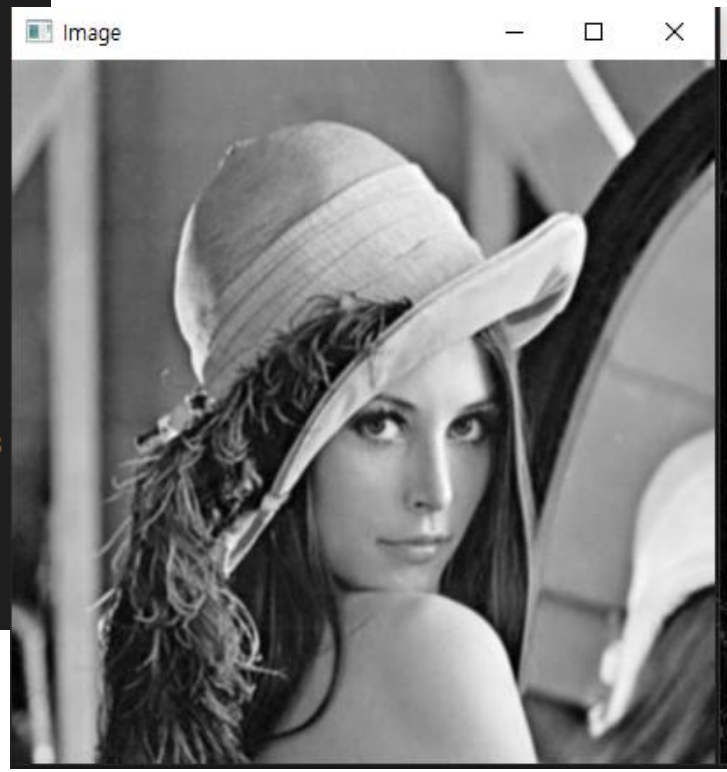
: 10.공간필터링 – p.27~28

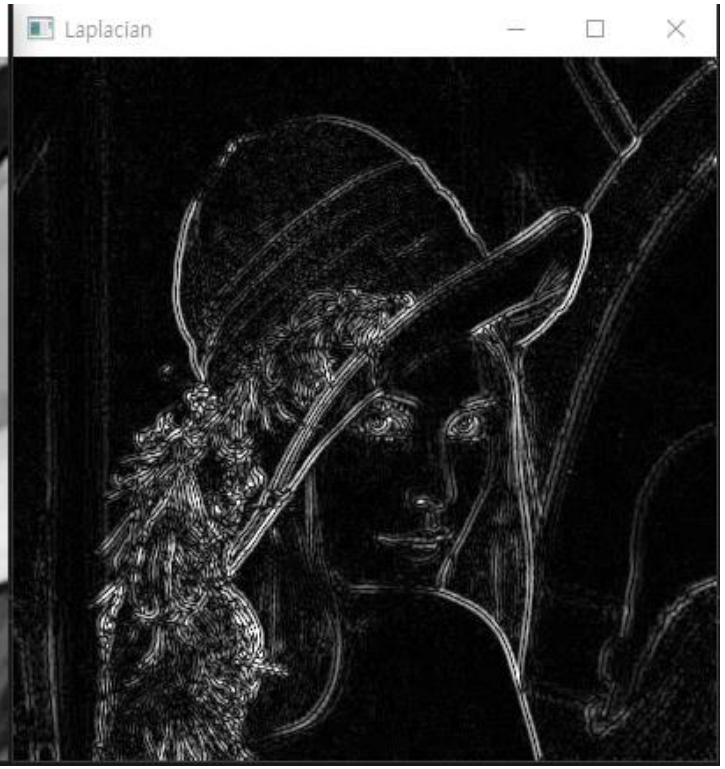
```cpp
void differential(Mat image, Mat& dst, float data1[], float data2[])
{
    Mat dst1, mask1(3, 3, CV_32F, data1);
    Mat dst2, mask2(3, 3, CV_32F, data2);

    filter2D(image, dst1, CV_32F, mask1);
    filter2D(image, dst2, CV_32F, mask2);
    magnitude(dst1, dst2, dst);
    dst.convertTo(dst, CV_8U);
    convertScaleAbs(dst1, dst1); //절대값 및 형 변환 동시 수행
    convertScaleAbs(dst2, dst2);
    imshow("dst1 - 수직 마스크", dst1);
    imshow("dst2 - 수평 마스크", dst2);
}
```

```cpp
    //공간필터링 p.27~28
#if 1
    Mat image = imread("D:\\999.Image\\edge_test1.jpg", IMREAD_GRAYSCALE);
    CV_Assert(image.data);

    //프르윗 마스크 원소
    float data1[] = { //수직마스크
        -1, 0, 1,
        -1, 0, 1,
        -1, 0, 1,
    };
    float data2[] = { //수평마스크
        -1, -1, -1,
        0, 0, 0,
        1, 1, 1,
    };

    Mat dst;
    differential(image, dst, data1, data2);
    imshow("image", image);
    imshow("프리윗 에지", dst);
    waitKey();
#endif
```

: 10.공간필터링 – p.37

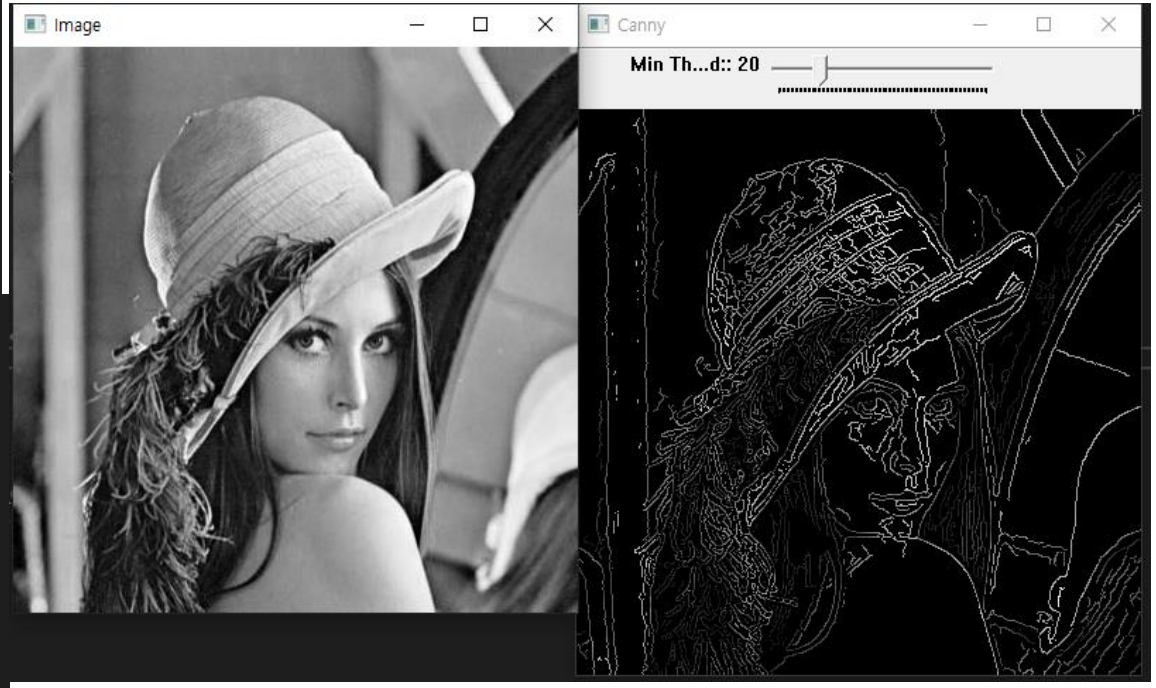
```
    //공간필터링 p.37
#if 1
    Mat src, src_gray, dst;
    int kernel_size = 3;
    int scale = 1;
    int delta = 0;
    int ddepth = CV_16S;
    src = imread("D:\\999.Image\\lenna.jpg", IMREAD_GRAYSCALE);
    if (src.empty())
    {
        return -1;
    }
    GaussianBlur(src, src, Size(3, 3), 0, 0, BORDER_DEFAULT);
    Mat abs_dst;
    Laplacian(src, dst, ddepth, kernel_size, scale, delta, BORDER_DEFAULT);
    convertScaleAbs(dst, abs_dst);
    imshow("Image", src);
    imshow("Laplacian", abs_dst);
    waitKey();
#endif
```





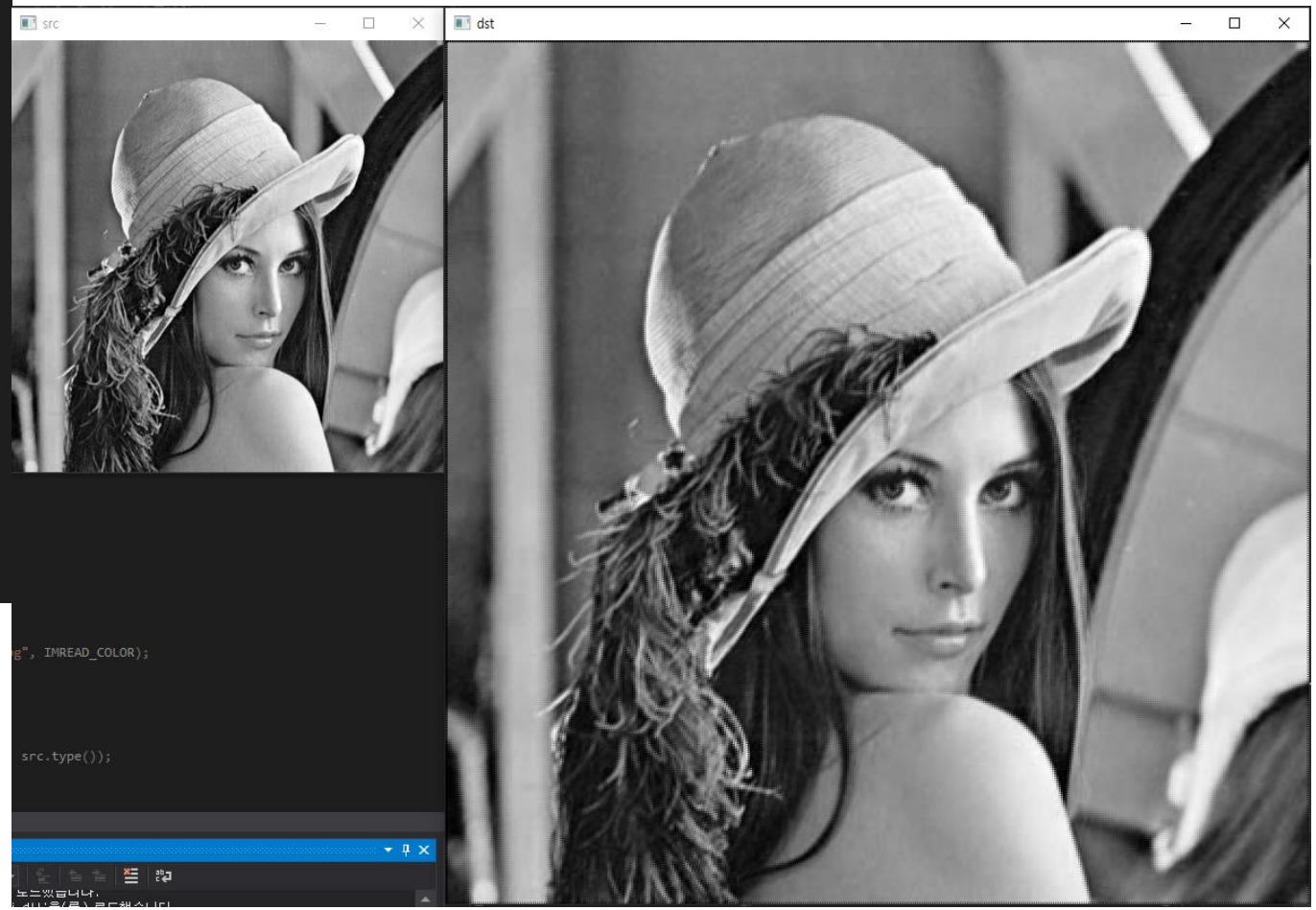
: 10.공간필터링 – p.44

```
//공간필터링 p.44
Mat detected_edges;
int lowThreshold = 0;
const int max_lowThreshold = 100;
const int ratio = 3;
const int kernel_size = 3;

static void CannyThreshold(int, void*)
{
    blur(src, detected_edges, Size(3, 3));
    Canny(detected_edges, detected_edges, lowThreshold, lowThreshold * ::ratio, kernel_size);
    dst = Scalar::all(0);
    src.copyTo(dst, detected_edges);
    imshow("Image", src);
    imshow("Canny", dst);
}
    //공간필터링 p.44
#if 1
    src = imread("D:\\999.Image\\lenna.jpg", IMREAD_GRAYSCALE);
    if (src.empty())
    {
        return -1;
    }
    dst.create(src.size(), src.type());
    namedWindow("Canny", WINDOW_AUTOSIZE);
    createTrackbar("Min Threshold:", "Canny", &lowThreshold, max_lowThreshold, CannyThreshold);
    CannyThreshold(0, 0);
    waitKey(0);

#endif
```



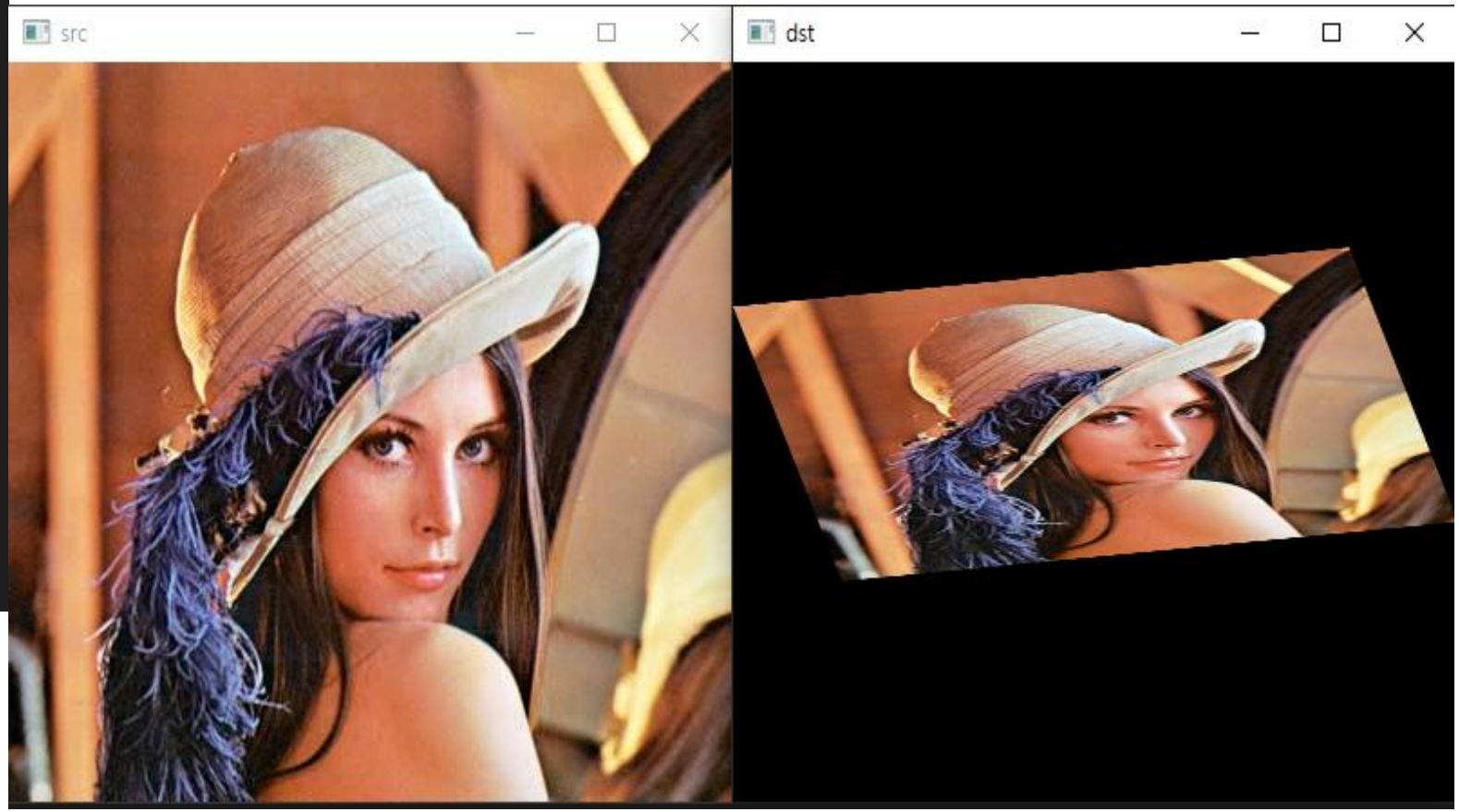
: 11.기하학적 변환 – p.10

```cpp
    //기하학적 변환 - p.10
#if 1
    Mat src = imread("D:\\999.Image\\lenna.jpg", IMREAD_GRAYSCALE);
    Mat dst = Mat::zeros(Size(src.cols * 2, src.rows * 2), src.type());

    for (int y = 0; y < dst.rows; y++)
    {
        for (int x = 0; x < dst.cols; x++)
        {
            float gx = ((float)x) / 2.0;
            float gy = ((float)y) / 2.0;
            int gxi = (int)gx;
            int gyi = (int)gy;
            float c00 = GetPixel(src, gxi, gyi);
            float c01 = GetPixel(src, gxi+1, gyi);
            float c10 = GetPixel(src, gxi, gyi+1);
            float c11 = GetPixel(src, gxi + 1, gyi+1);
            int value = (int)Blerp(c00, c10, c01, c11, gx - gxi, gy - gyi);
            dst.at<uchar>(y, x) = value;
        }
    }
    imshow("src", src);
    imshow("dst", dst);
    waitKey();
#endif
```



: 11.기하학적 변환 – p.18

```
    //기하학적 변환 - p.18
#if 1
    Mat src = imread("D:\\999.Image\\lenna.jpg", IMREAD_COLOR);
    Point2f srcTri[3];
    Point2f dstTri[3];
    Mat warp_mat(2, 3, CV_32FC1);

    Mat warp_dst;
    warp_dst = Mat::zeros(src.rows, src.cols, src.type());
    srcTri[0] = Point2f(0, 0);
    srcTri[1] = Point2f(src.cols - 1.0f, 0);
    srcTri[2] = Point2f(0, src.rows - 1.0f);
    dstTri[0] = Point2f(src.cols * 0.0f, src.rows * 0.33f);
    dstTri[1] = Point2f(src.cols * 0.85f, src.rows * 0.25f);
    dstTri[2] = Point2f(src.cols * 0.15f, src.rows * 0.7f);
    warp_mat = getAffineTransform(srcTri, dstTri);
    warpAffine(src, warp_dst, warp_mat, warp_dst.size());

    imshow("src", src);
    imshow("dst", warp_dst);
    waitKey();
#endif
```



: 11.기하학적 변환 – p.23

```
    //기하학적 변환 - p.23
#if 1
    Mat src = imread("D:\\999.Image\\book.jpg");

    Point2f inputp[4];
    inputp[0] = Point2f(30, 81);
    inputp[1] = Point2f(274, 247);
    inputp[2] = Point2f(298, 40);
    inputp[3] = Point2f(598, 138);

    Point2f outputp[4];
    outputp[0] = Point2f(0, 0);
    outputp[1] = Point2f(0, src.rows);
    outputp[2] = Point2f(src.cols, 0);
    outputp[3] = Point2f(src.cols, src.rows);

    Mat h = getPerspectiveTransform(inputp, outputp);
    Mat out;
    warpPerspective(src, out, h, src.size());
    imshow("Source Image", src);
    imshow("Warped Source Image", out);
    waitKey();
#endif
```