Open Software Project

Assignment 06 Technical Report

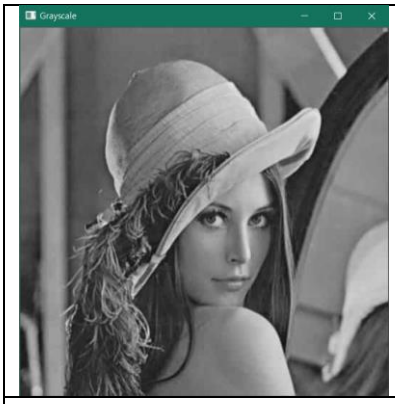
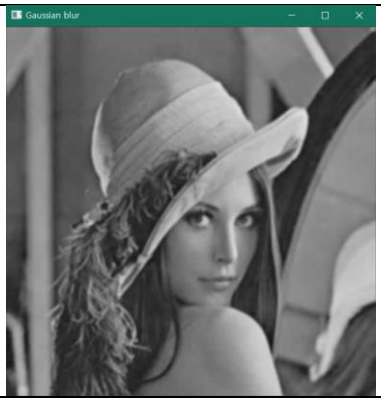
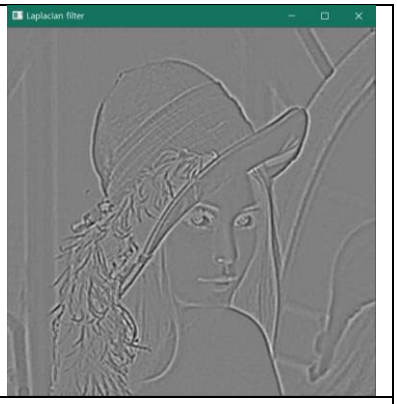
1976235 오진솔

1. LoG_Gray

```
Mat h_f = Gaussianfilter(input_gray, window_radius, sigma_t, sigma_s);
Mat Laplacian = Laplacianfilter(h_f);

(중략)

Mat Gaussianfilter(const Mat input, int n, double sigma_t, double sigma_s) {
    int row = input.rows;
    int col = input.cols;

    // generate gaussian kernel
    Mat kernel = get_Gaussian_Kernel(n, sigma_t, sigma_s, true);
    Mat output = Mat::zeros(row, col, input.type());

    //Intermediate data generation for mirroring
    Mat input_mirror = Mirroring(input, n);

    for (int i = n; i < row + n; i++) {
        for (int j = n; j < col + n; j++) {
            float sum1 = 0.0;
            for (int a = -n; a <= n; a++) {
                for (int b = -n; b <= n; b++) {
                    sum1 += kernel.at<double>(a + n, b + n) * input_mirror.at<double>(i + a, j + b);
                }
            }
            output.at<double>(i-n, j-n) = sum1;
        }
    }
    return output;
}
```

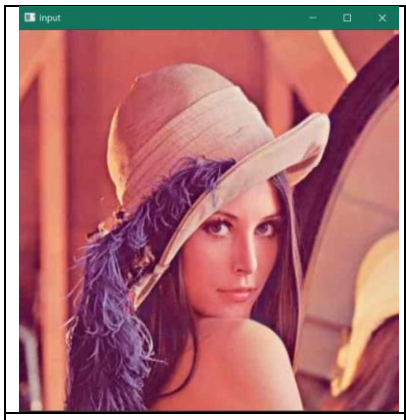
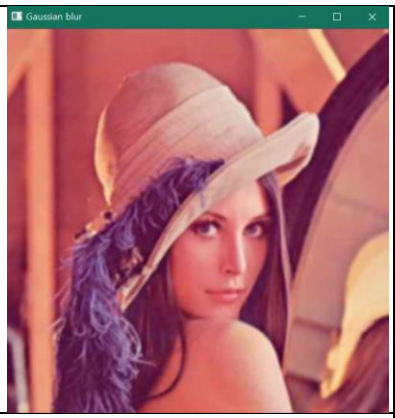
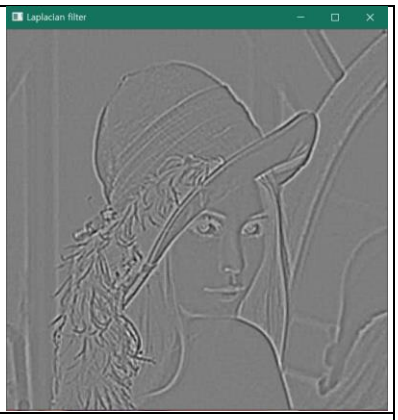
```
Mat Laplacianfilter(const Mat input) {
    int row = input.rows;
    int col = input.cols;

    Mat kernel = get_Laplacian_Kernel();
    Mat output = Mat::zeros(row, col, input.type());

    int n = 1;
    Mat input_mirror = Mirroring(input, n);

    for (int i = n; i < row + n; i++) {
        for (int j = n; j < col + n; j++) {
            float sum1 = 0.0;
            for (int a = -n; a <= n; a++) {
                for (int b = -n; b <= n; b++) {
                    sum1 += kernel.at<double>(a + n, b + n) * input_mirror.at<double>(i + a, j + b);
                }
            }
            output.at<double>(i-n, j-n) = sum1;
        }
    }
    return output;
}
```

이미지에 Gaussian 필터를 적용한 후 Laplacian 필터를 적용한다.

		
Grayscale	Gaussian	LOG (Laplacian of Gaussian)

2. LoG_RGB

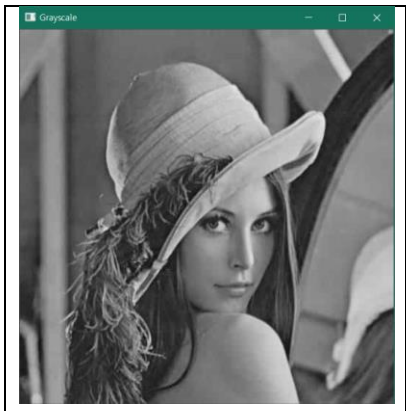
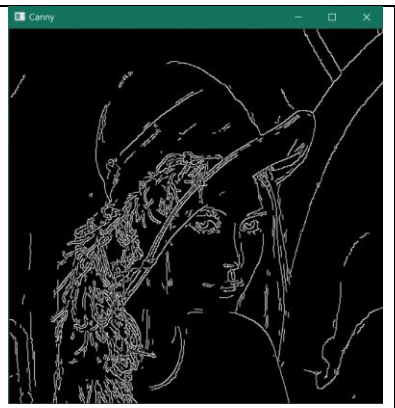
같은 작업을 R,G,B 각각의 채널에 적용한다.

		
Original	Gaussian	LOG (Laplacian of Gaussian)

3. Canny

```
Canny(input_gray, output, 50, 200, 3, false);
```

<result>

		
Grayscale	Canny Edge Detector (t1 = 100, t2 = 150)	Canny Edge detector (t1 = 50, t2 = 200)

<analysis>

threshold 의 중간 범위를 넓게 잡을수록(=양 끝 범위를 좁게 잡을수록) 단순화된다.

4. Harris

<Harris corner detection>

```
cornerHarris(input_gray, output, 2, 3, 0.04, BORDER_DEFAULT);
```

<Non-maximum Suppression>

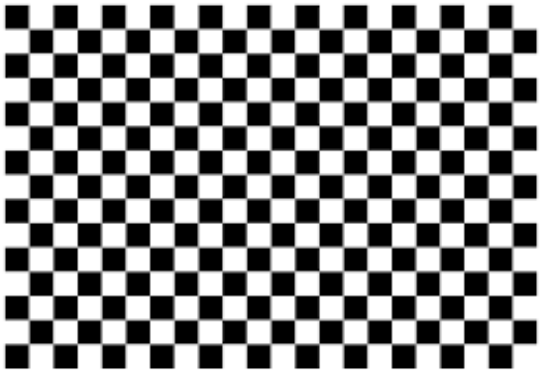
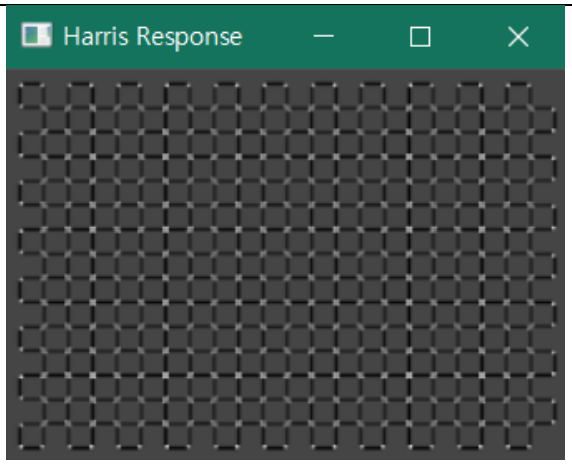
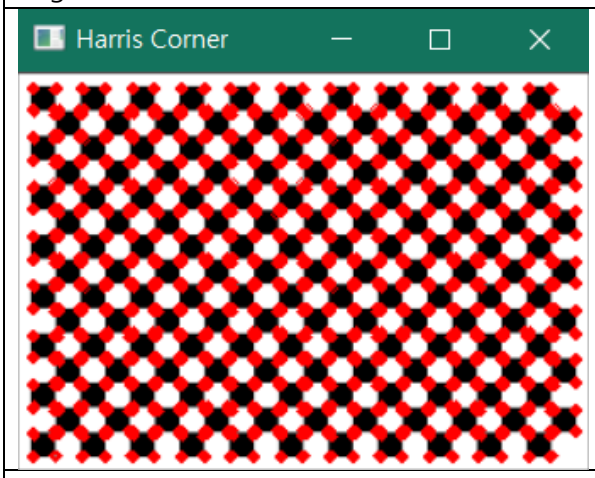
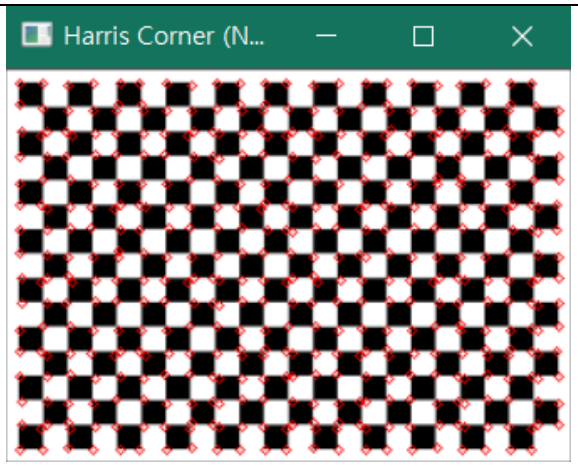
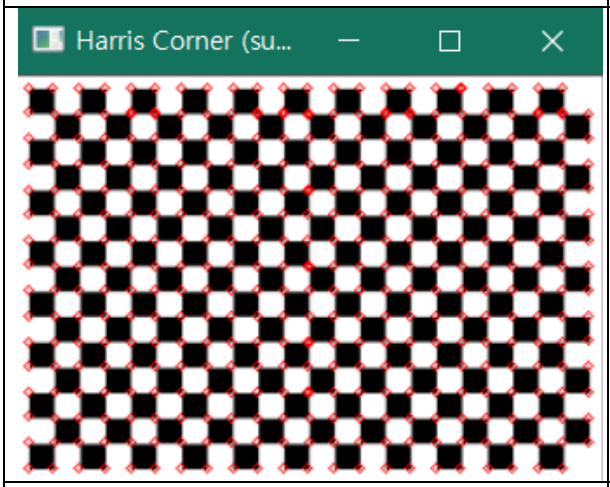
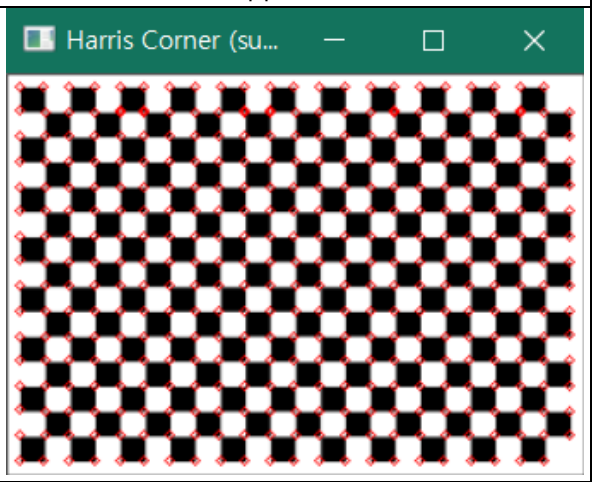
```
for (int i = radius; i < row + radius; i++) {
    for (int j = radius; j < col + radius; j++) {
        if (corner_mat.at<uchar>(i-radius, j-radius) == 1) {
            M = magnitude(input_mirror, i, j);
            for (int a = -radius; a <= radius; a++) {
                for (int b = -radius; b <= radius; b++) {
                    if (i + a >= radius && i + a < row + radius &&
                                                    j + b >= radius && j + b < col + radius) {
                        if (corner_mat.at<uchar>(i + a - radius, j + b - radius)){
                            if (M < magnitude(input_mirror, i + a, j + b)) {
                                corner_mat.at<uchar>(i - radius, j - radius) = 0;
                                 break;
```

corner 여러 개가 탐지되어 모여 있는 곳의 corner 수를 줄여서 정확도를 높인다.

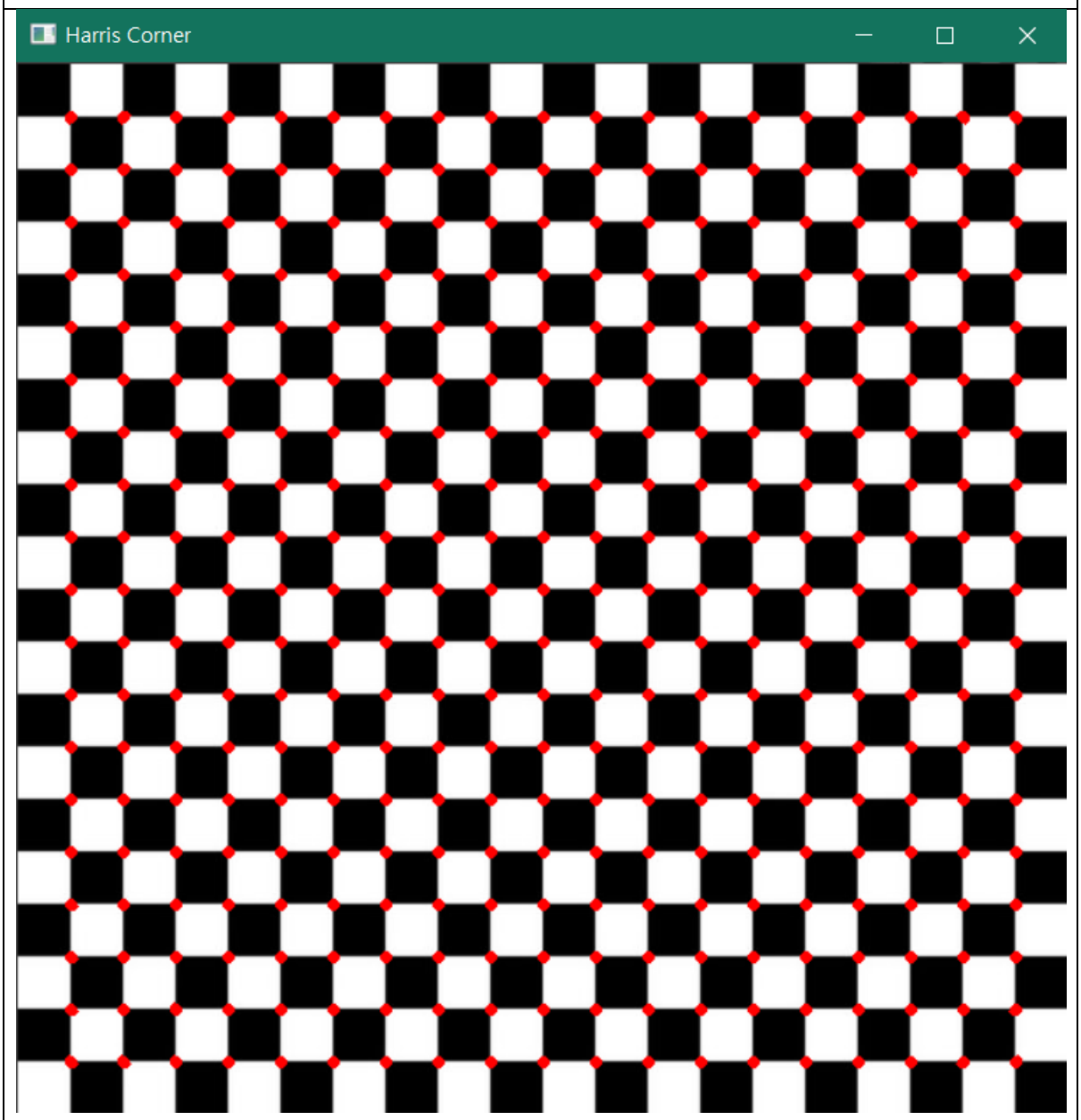
<subpixel>

```
cornerSubPix(input_gray, points, subPixWinSize, zeroZone, termcrit);
```

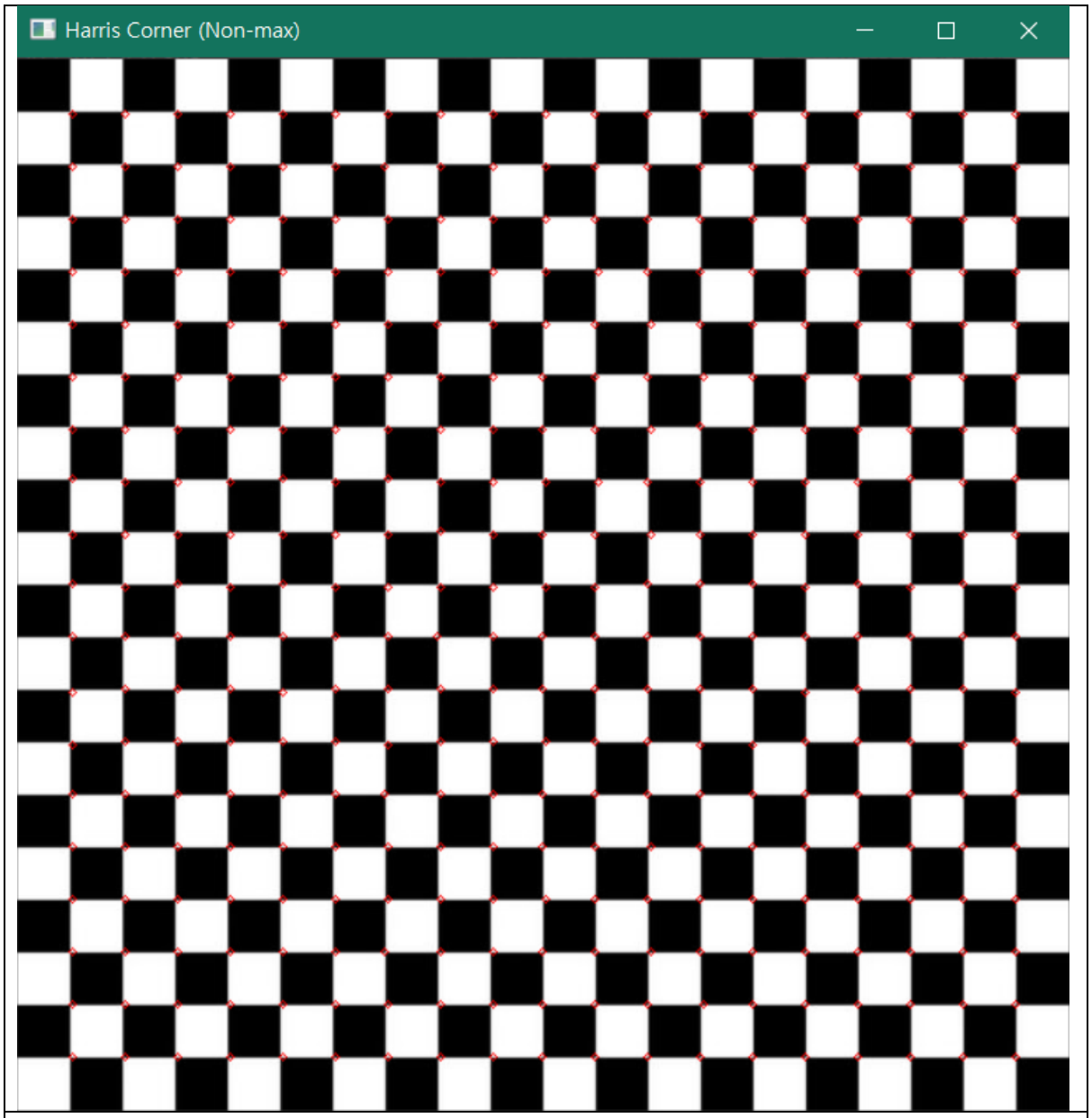
픽셀 단위의 코너 탐지에서 픽셀 사이의 더 정확한 코너를 탐지할 수 있도록 한다.
(int, int) -> (float, float)

checkerboard	
	
original	Harris corner detection (response)
	
Harris corner detection (visualized)	+ non-maximum suppression
	
+subpixel	+non_maximum suppression +subpixel

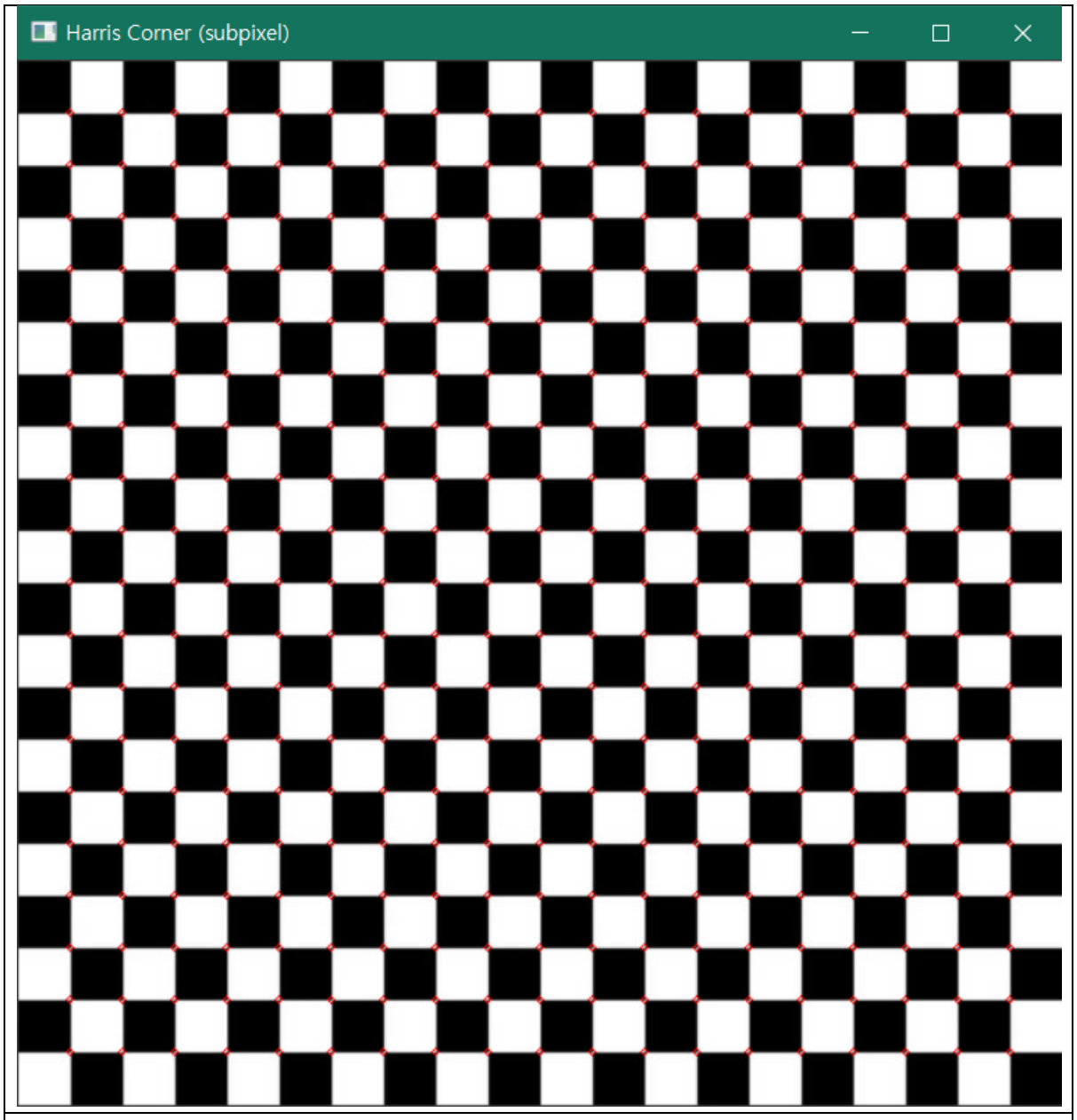
checkerboard2



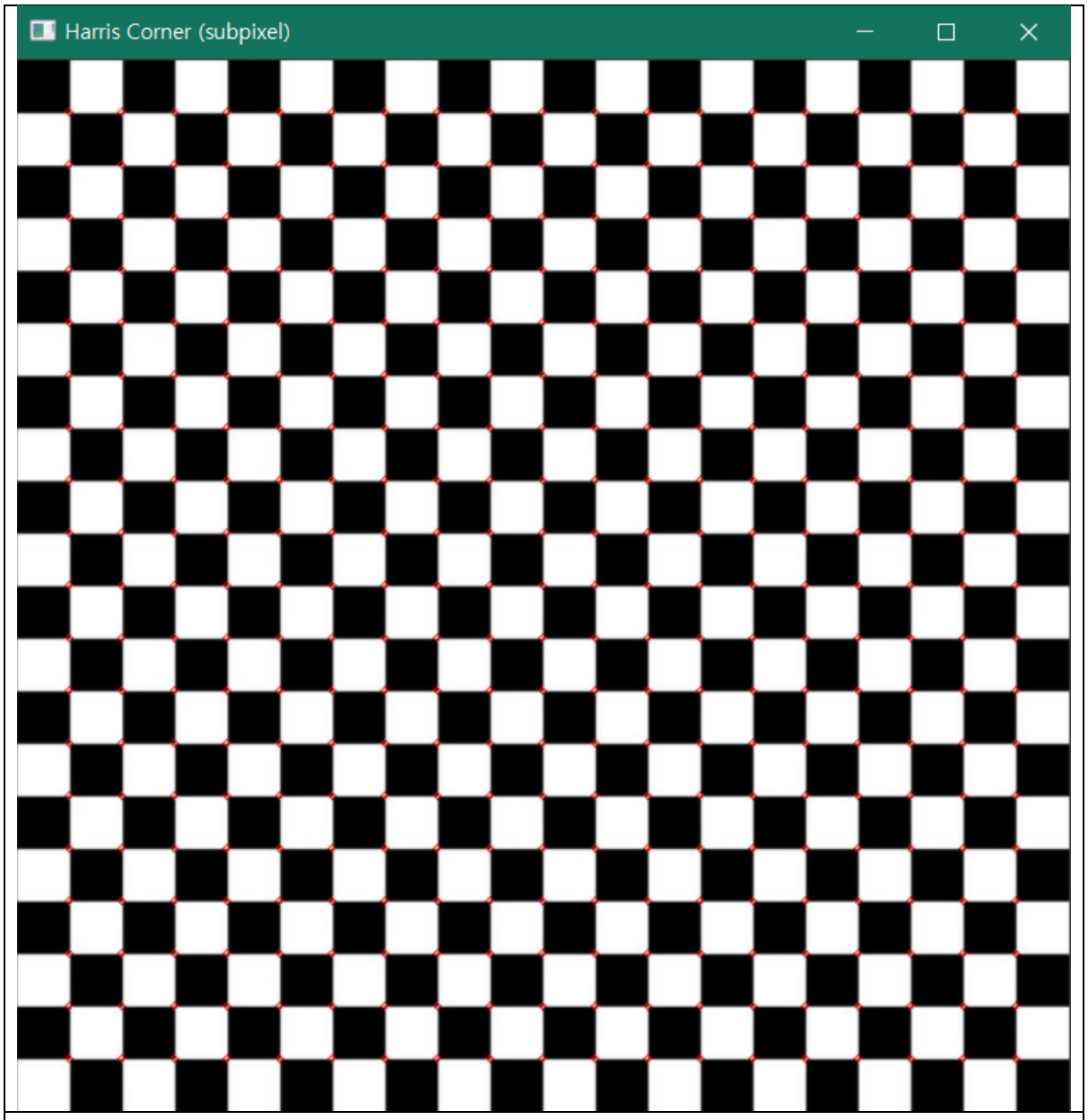
Harris corner detection (visualized)



+ non-maximum suppression

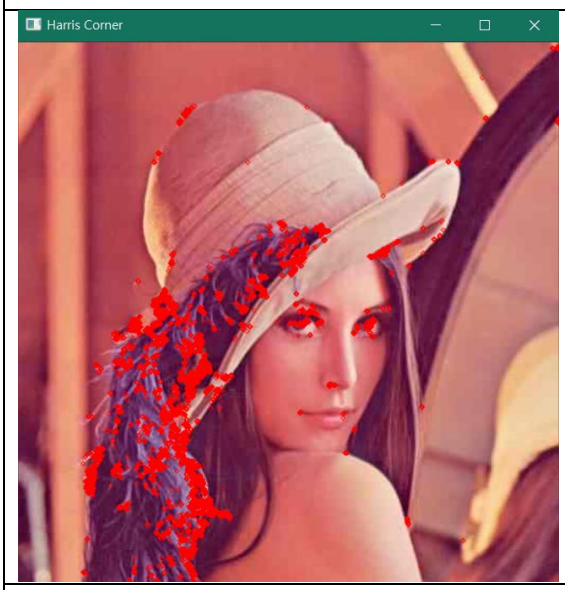
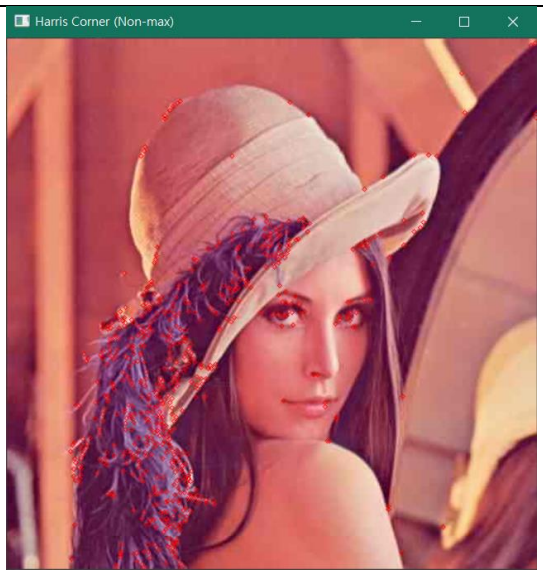
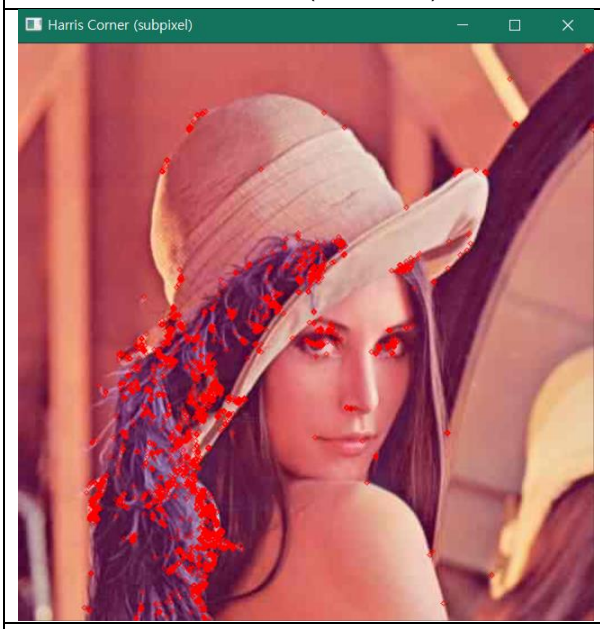
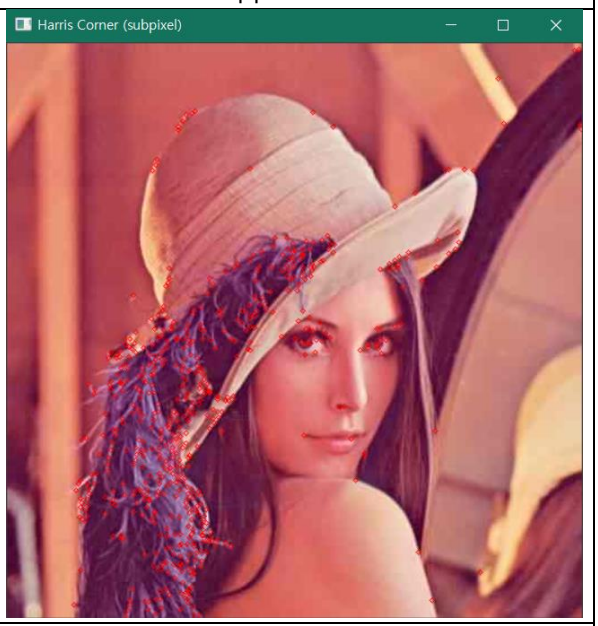


+ subpixel



+ non-maximum suppression

+ subpixel

lena	
	
Harris corner detection (visualized)	+non-maximum suppression
	
+ subpixel	+ non-maximum suppression + subpixel