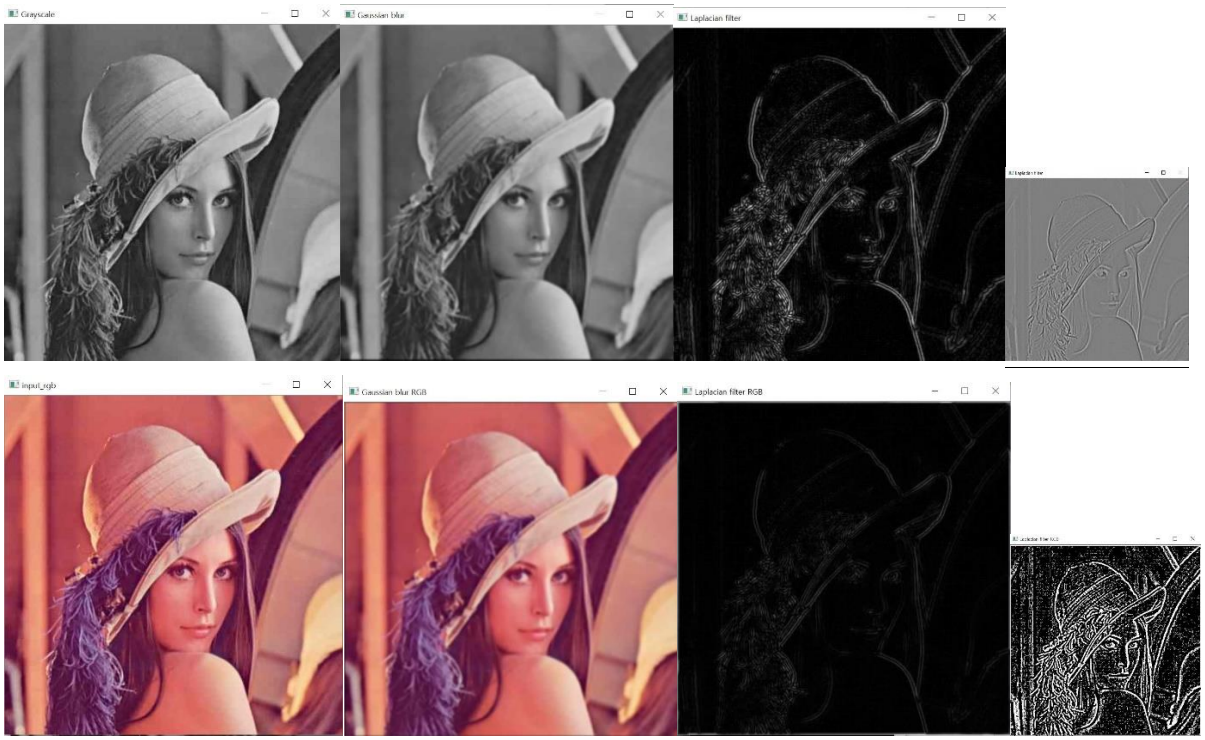
Technical Report

1829008 김민영

<1> practice : Implement the Laplacian of Gaussian

1. result Image



2. Explanation of codes (LaplacianOfGuassian.cpp)

Main 함수 : gray scale, RGB scale 에서, Guassianfilter 를 적용한 뒤, Laplacian filter 를 적용하여 결과를 출력한다.

get_Guassian_Kernel 함수: 정규분포 공식에 따라 kernel 에 값을 저장해놓고 이를 반환해준다.

get_Laplacian_Kernel 함수: Laplacian kernel 3x3 행렬을 kernel 에 값을 저장해놓고 이를 반환해준다.

Gaussianfilter_gray, GaussianfilterRGB 함수: Guassian kernel 을 가져와 convolution 을 통해 guassian filter 를 적용시킨다.

Laplacianfilter_gray 함수 : Mirroring 함수를 호출시켜 input_mirror Matrix 를 만들고, convolution 을 통해 Laplacian filter 를 적용시킨다.

LaplacianfilterRGB 함수 : Mirroring 함수를 호출시키지 않고, 함수 내에서 직접 mirroring 방식으로 Laplacian filter 를 적용하였다.

Mirroring 함수 : input Matrix 를 mirroring 한 결과값을 반환한다.

3. Analysis(전체적인 과정)

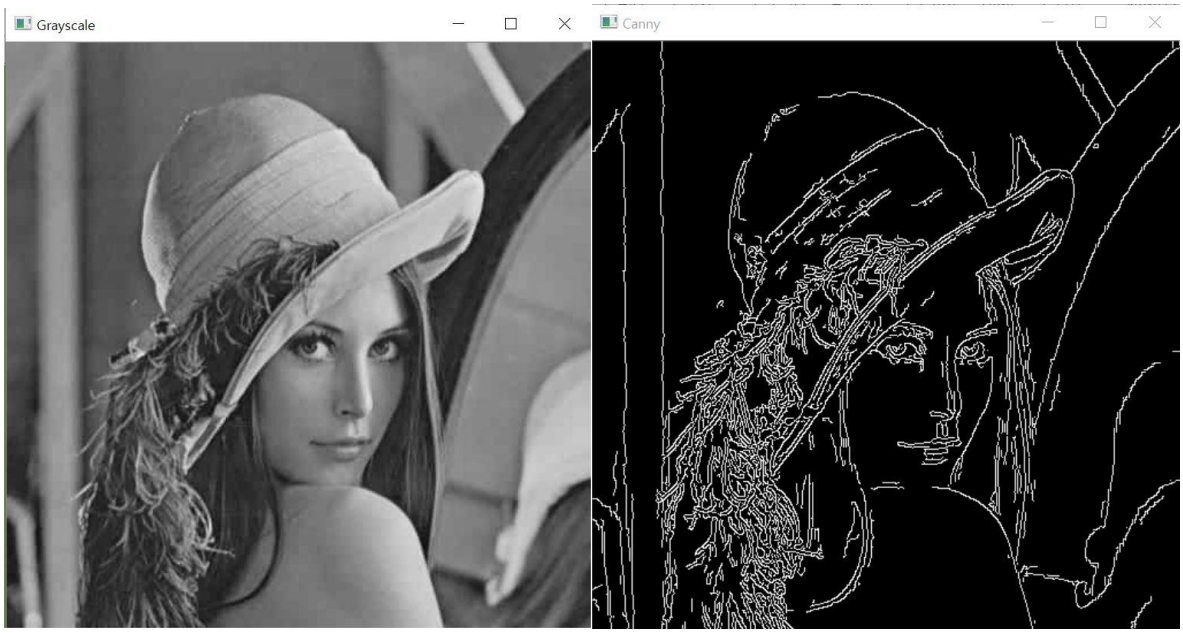
Marr-HildrethMethod (Laplacianof Gaussian: LoG)

1. Apply the Gaussian filter for removing unnecessary noise.

2. Apply the Laplacian filter

<2> Practice : Canny Edge Detector

1. Result image



2. Explanation of codes (Canny.cpp)

openCV 에서 제공하는 Canny Edge detector function 을 이용하여 코드를 실행시킨다.

```
Canny(input_gray, output, 50, 100, 3, true);
```

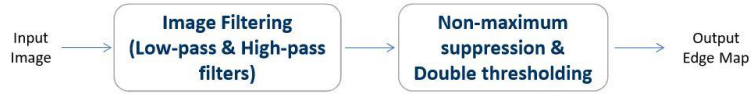
InputArray : 8 biy image

OutputArray : output edge map/ single channel 8-bit image, which has the same size as image

```
void cv::Canny        (InputArray          image,
                       OutputArray         edges,
                       double              threshold1,
                       double              threshold2,
                       int                 apertureSize = 3,
                       bool                L2gradient = false
                      )
```

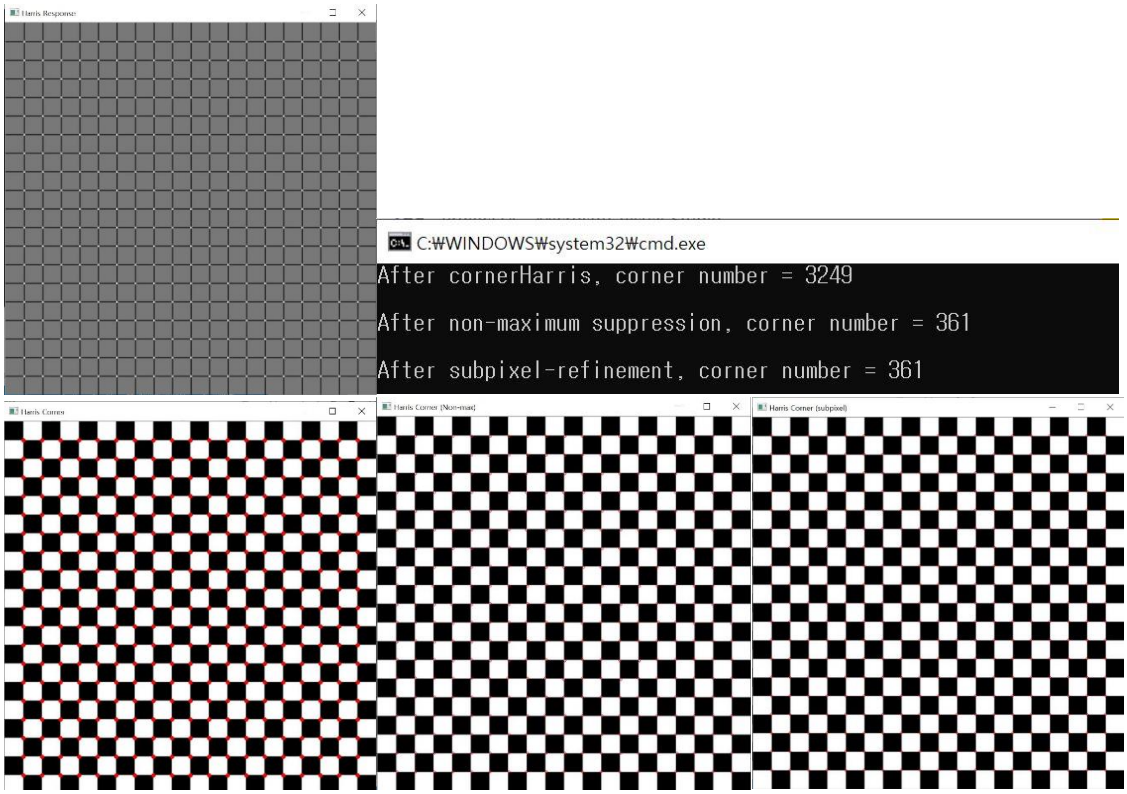
3. Analysis

Canny Edge detector 는 가장 유명한 방법이다. 작동 원리는 아래와 같다.



<3> practice : Harris Corner Detector

1. result Image



2. Explanation of codes (Harris.cpp)

Main 함수 : openCV 의 cornerHarris 함수를 통한 harris response, Harris corner, non-maximum suppressopn 을 통한 Harris corner(Non-max), Harris Corner(subpixel) 을 출력한다.

MatToVec 함수 : Matrix 를 Vertor 로 변환한다.

NonMaximum_Suppression 함수 :

Mirroring 함수 : input Matrix 를 Mirroring 한 input2 Matrix 를 반환한다.

type2str 함수 : input Matrix 의 type 을 알려준다.

```
cornerHarris(input_gray, output, 2, 3, 0.05, BORDER_DEFAULT);
```

➔ cornerHarris 사용

```
cornerSubPix(input_gray, points, subPixWinSize, Size(-1, -1), termcrit);
```

➔ cornerSubPix

```
Mat NonMaximum_Suppression(const Mat input, Mat corner_mat, int radius)
{
	int row = input.rows;
	int col = input.cols;
	int kernel_size = (2 * radius + 1);
	float max = 0.0;
	Mat input_mirror = Mirroring(input, radius);
	for (int i = radius; i < row + radius; i++) {
		for (int j = radius; j < col + radius; j++) {
			float max = 0;
			for (int a = -radius; a <= radius; a++) {// for each kernel window
				for (int b = -radius; b <= radius; b++) {
					if (input_mirror.at<float>(i + a, j + b) > max)
						max = input_mirror.at<float>(i + a, j + b);
				}
			}
			for (int a = -radius; a <= radius; a++) {// for each kernel window
				for (int b = -radius; b <= radius; b++) {
					if (i + a - radius >= 0 & i + a - radius < row&j + b - radius >= 0 & j + b - radius < col)
						if (input_mirror.at<float>(i + a, j + b) < max)
							corner_mat.at<uchar>(i + a - radius, j + b - radius) = 0;
				}
			}
		}
	}
	return input;
}
```

➔ Non-Maximum suppression

3. Analysis(전체적인 과정)

Step 1) ComputeM matrix for each window to get their cornernessscores.

Step 2) Find points whose surrounding window gave large corner response (R> threshold)

Step 3) Take the points of local maxima, i.e., perform non-maximum suppression