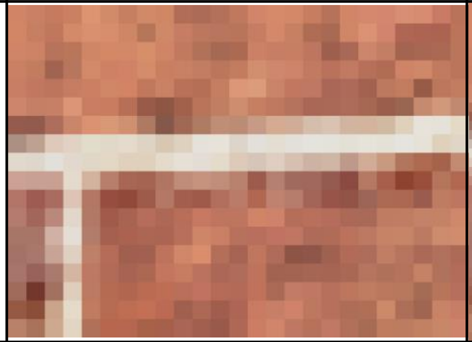
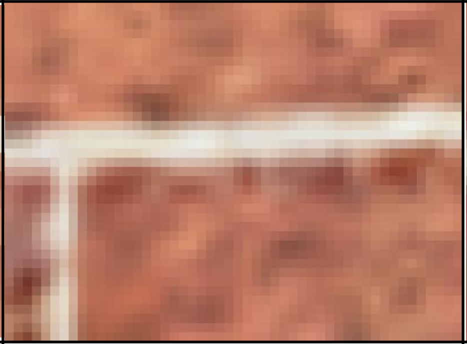
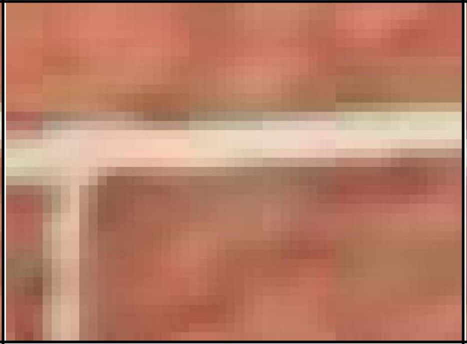
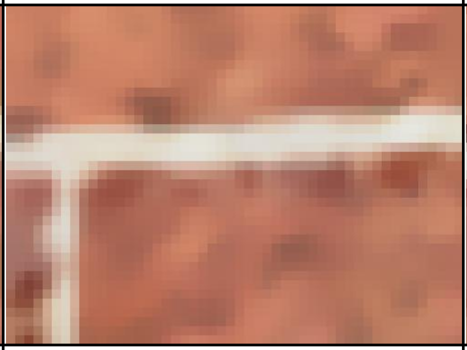
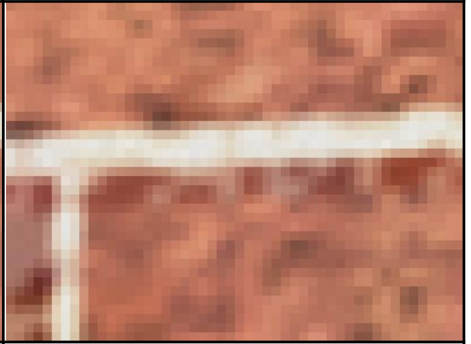
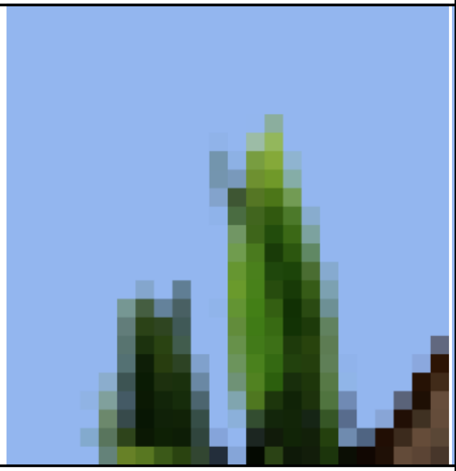
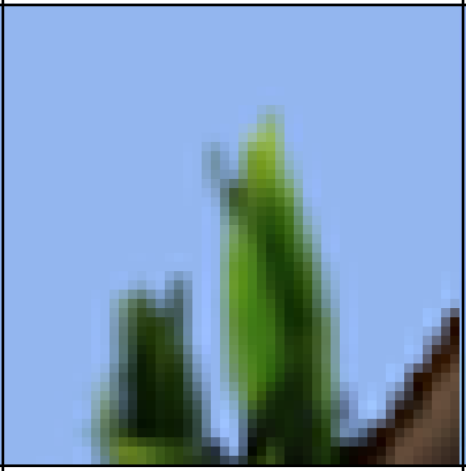
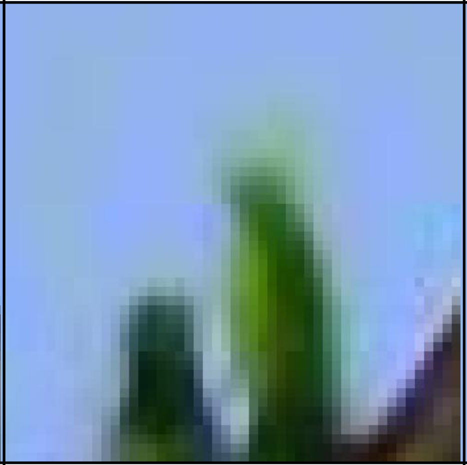
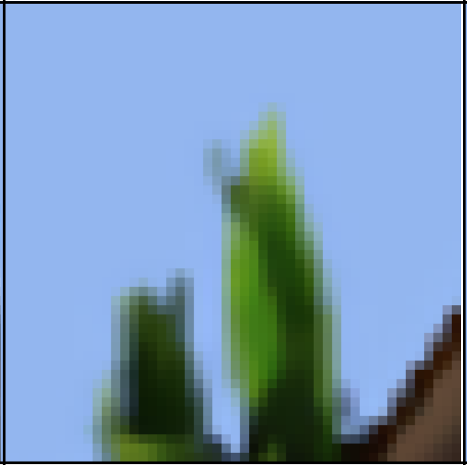
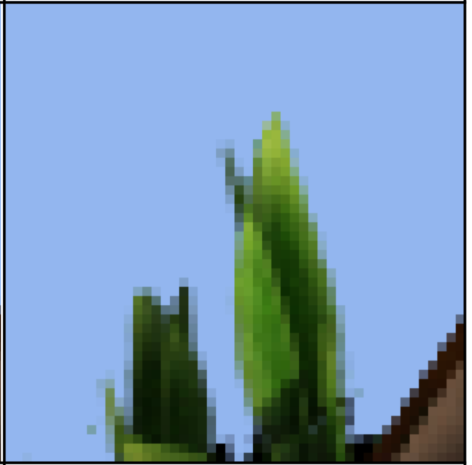
Bilateral method 분석
C언어.ver

염지현

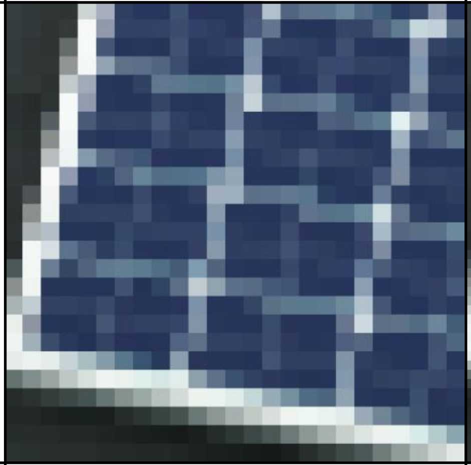
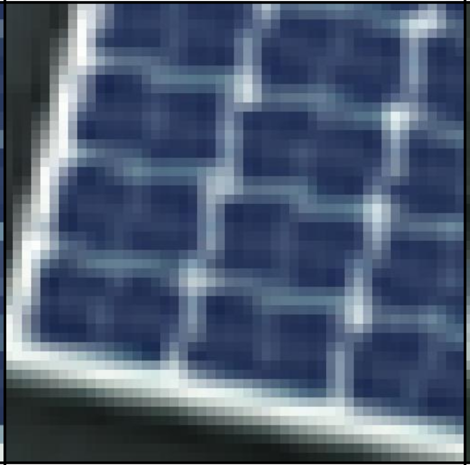
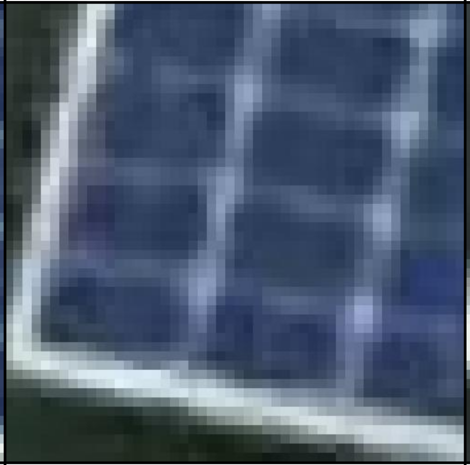
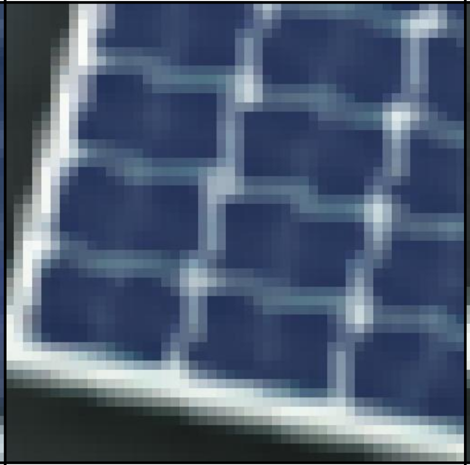
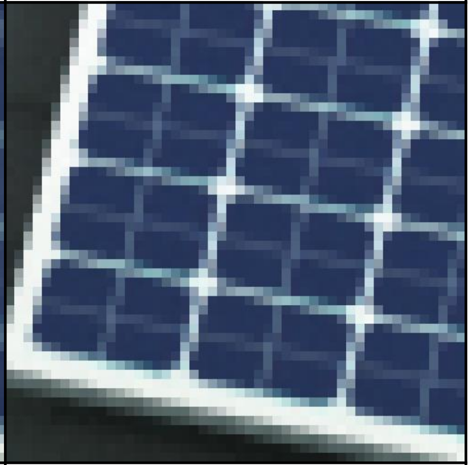
CafeInterior

Meth od	FHD	Bicubic	FSRCNN	Bilateral	UHD
Image					
PSNR (RGB)	-	37.77	35.46	37.61	-
Time (초)	-			300초 (Bicubic 제외)	-

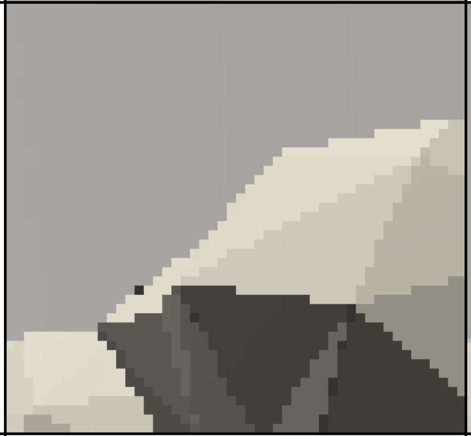
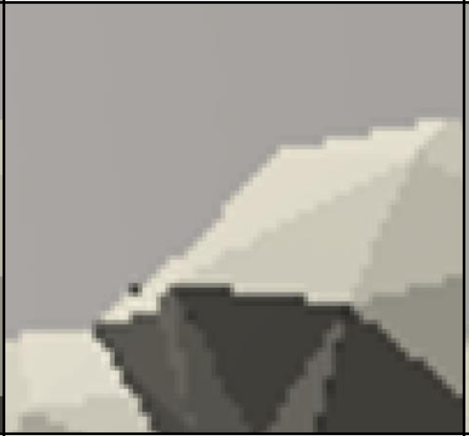
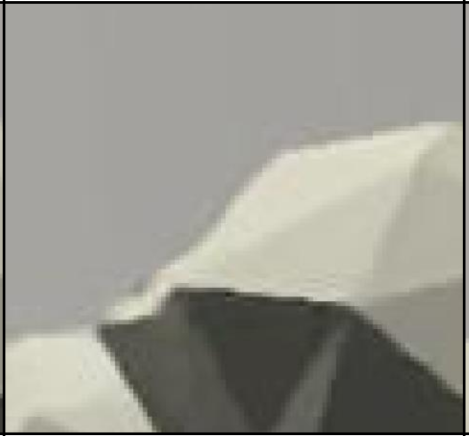
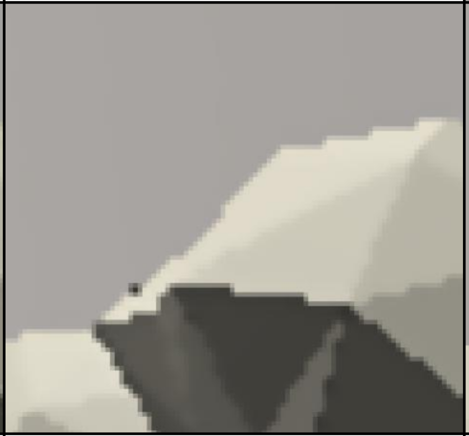
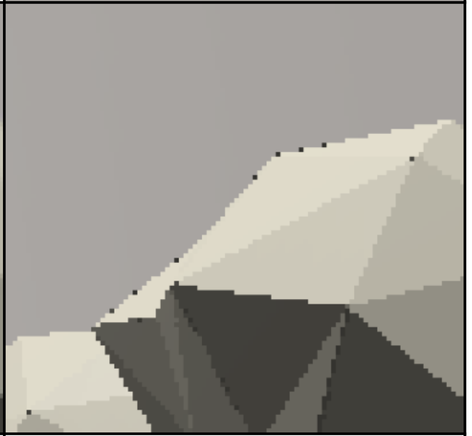
Village

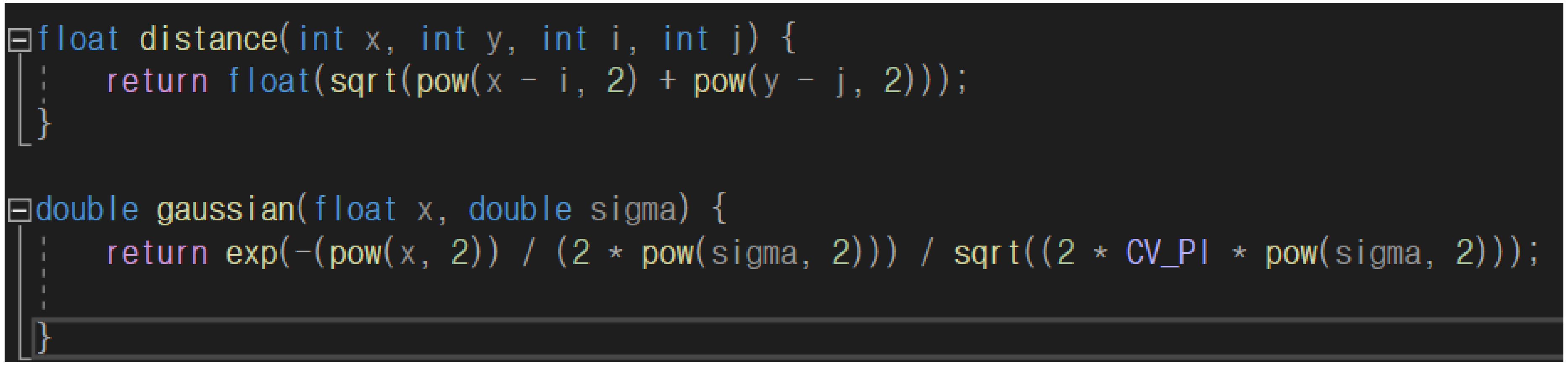
Meth od	FHD	Bicubic	FSRCNN	Bilateral	UHD
Image					
PSNR (RGB)	-	37.23	34.06	36.77	-
Time (초)	-			300초 (Bicubic 제외)	-

PolyTown

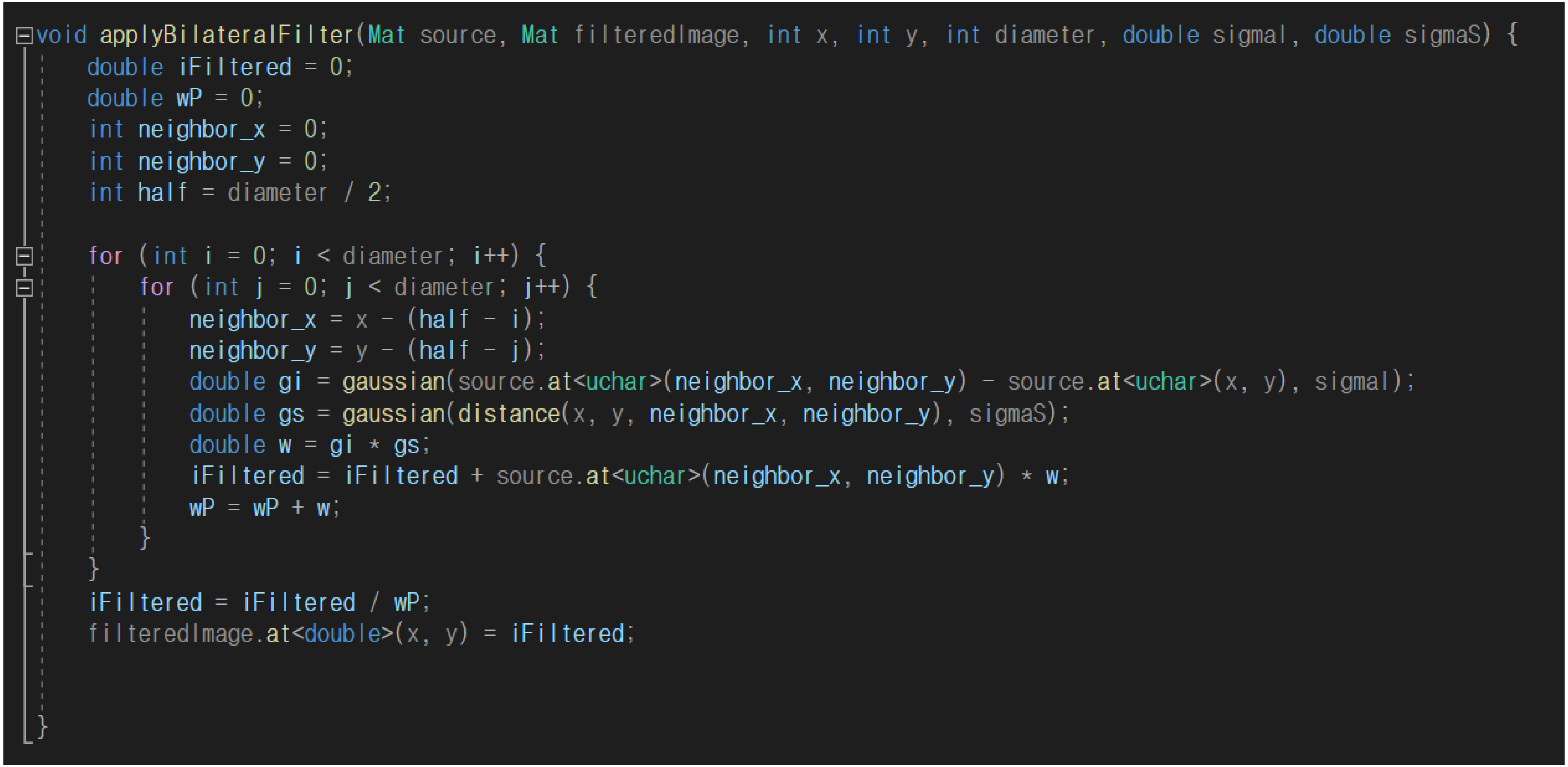
Meth od	FHD	Bicubic	FSRCNN	Bilateral	UHD
Image					
PSNR (RGB)	-	38.36	35.29	38.37	-
Time (초)	-			300초 (Bicubic 제외)	-

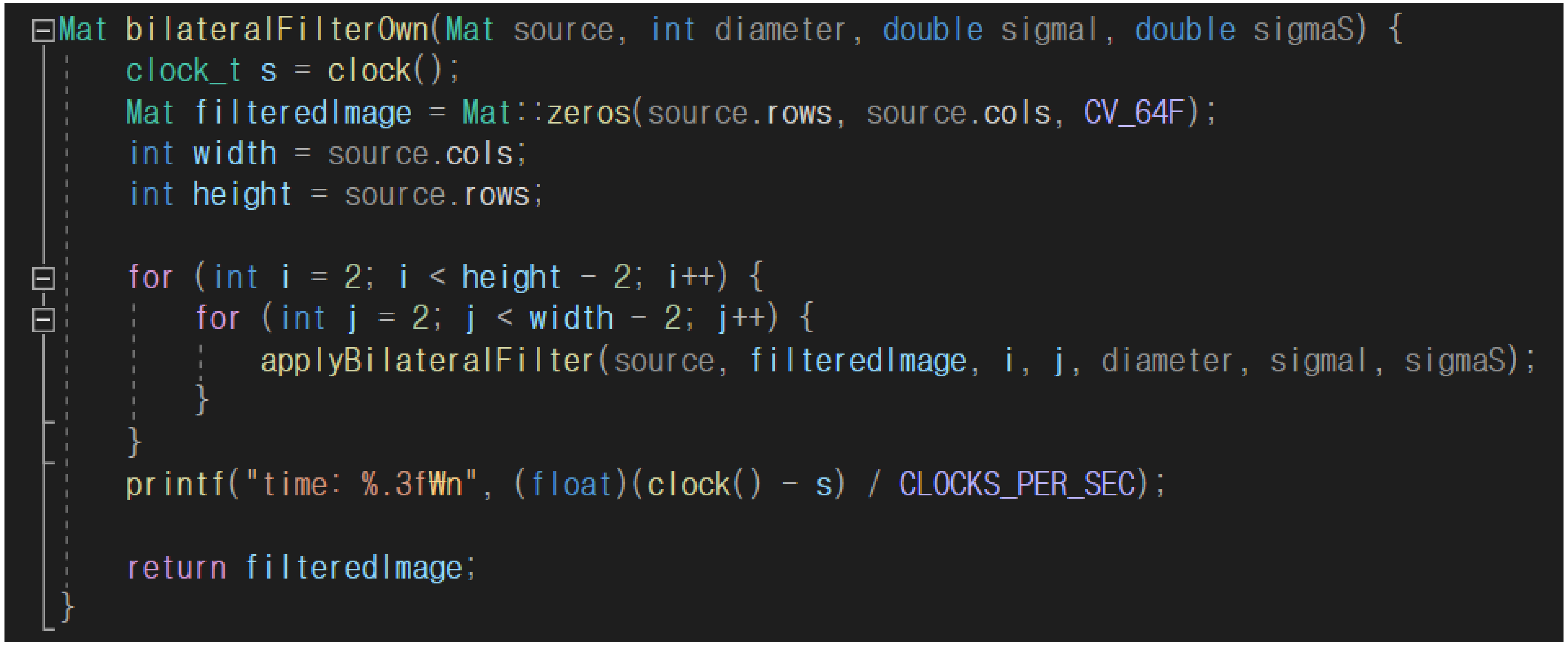
museum

Meth od	FHD	Bicubic	FSRCNN	Bilateral	UHD
Image					
PSNR (RGB)	-	40.8	36.4	40.91	-
Time (초)	-			300초 (Bicubic 제외)	-

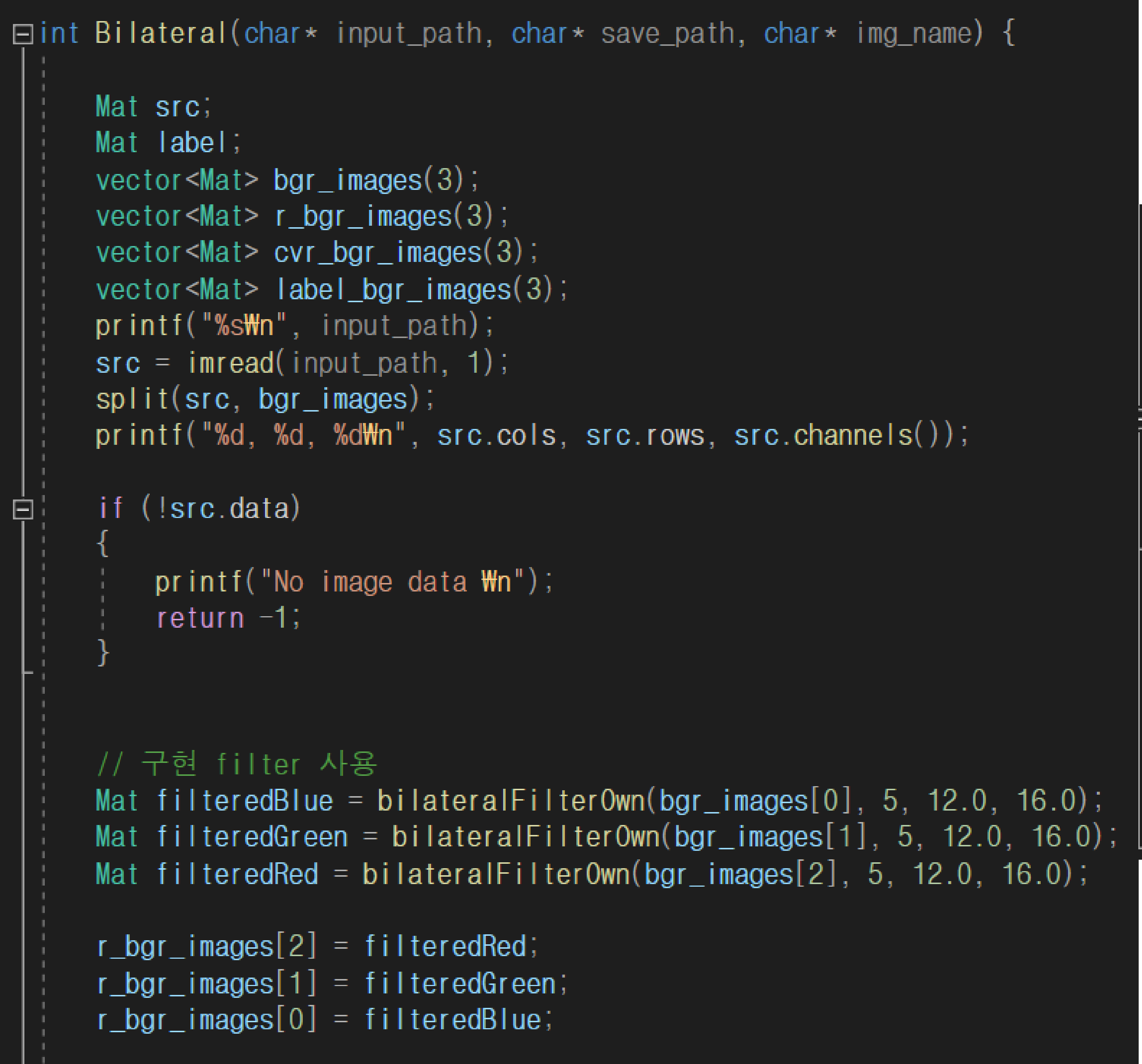


```
float distance(int x, int y, int i, int j) {
    return float(sqrt(pow(x - i, 2) + pow(y - j, 2)));
}

double gaussian(float x, double sigma) {
    return exp(-(pow(x, 2)) / (2 * pow(sigma, 2))) / sqrt((2 * CV_PI * pow(sigma, 2)));

}
```

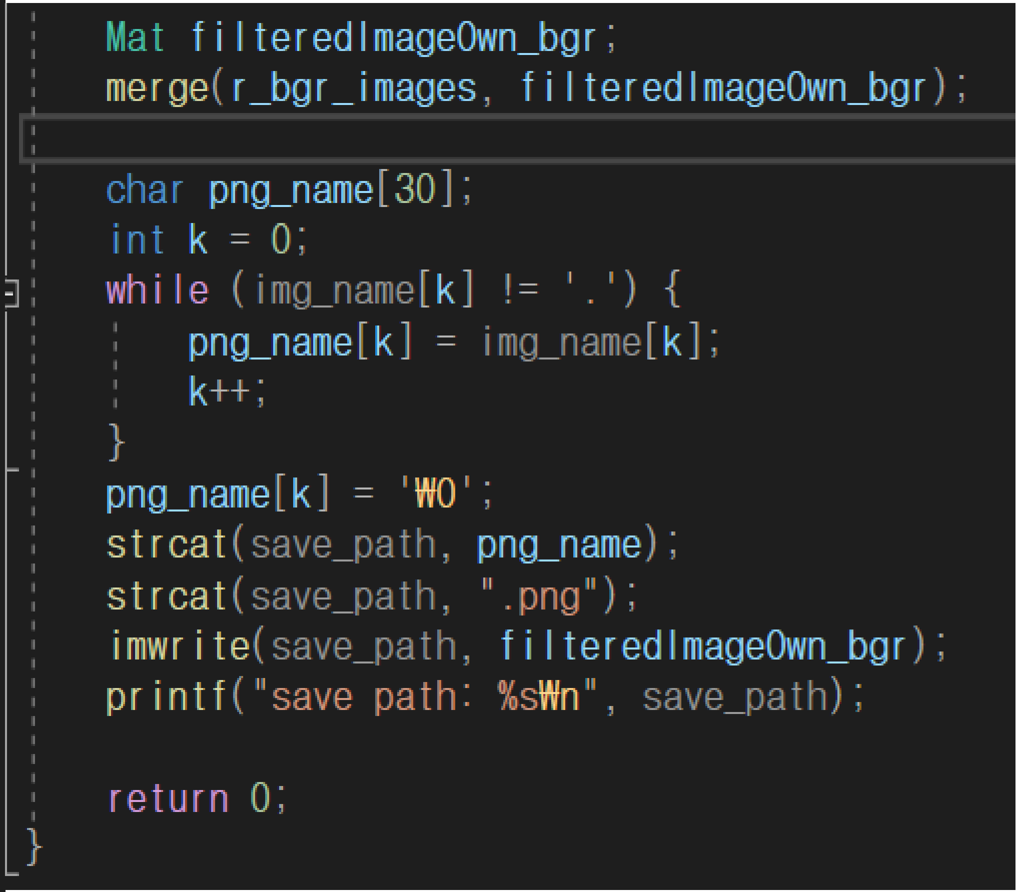

```
void applyBilateralFilter(Mat source, Mat filteredImage, int x, int y, int diameter, double sigmaI, double sigmaS) {
    double iFiltered = 0;
    double wP = 0;
    int neighbor_x = 0;
    int neighbor_y = 0;
    int half = diameter / 2;

    for (int i = 0; i < diameter; i++) {
        for (int j = 0; j < diameter; j++) {
            neighbor_x = x - (half - i);
            neighbor_y = y - (half - j);
            double gi = gaussian(source.at<uchar>(neighbor_x, neighbor_y) - source.at<uchar>(x, y), sigmaI);
            double gs = gaussian(distance(x, y, neighbor_x, neighbor_y), sigmaS);
            double w = gi * gs;
            iFiltered = iFiltered + source.at<uchar>(neighbor_x, neighbor_y) * w;
            wP = wP + w;
        }
    }
    iFiltered = iFiltered / wP;
    filteredImage.at<double>(x, y) = iFiltered;


}
```



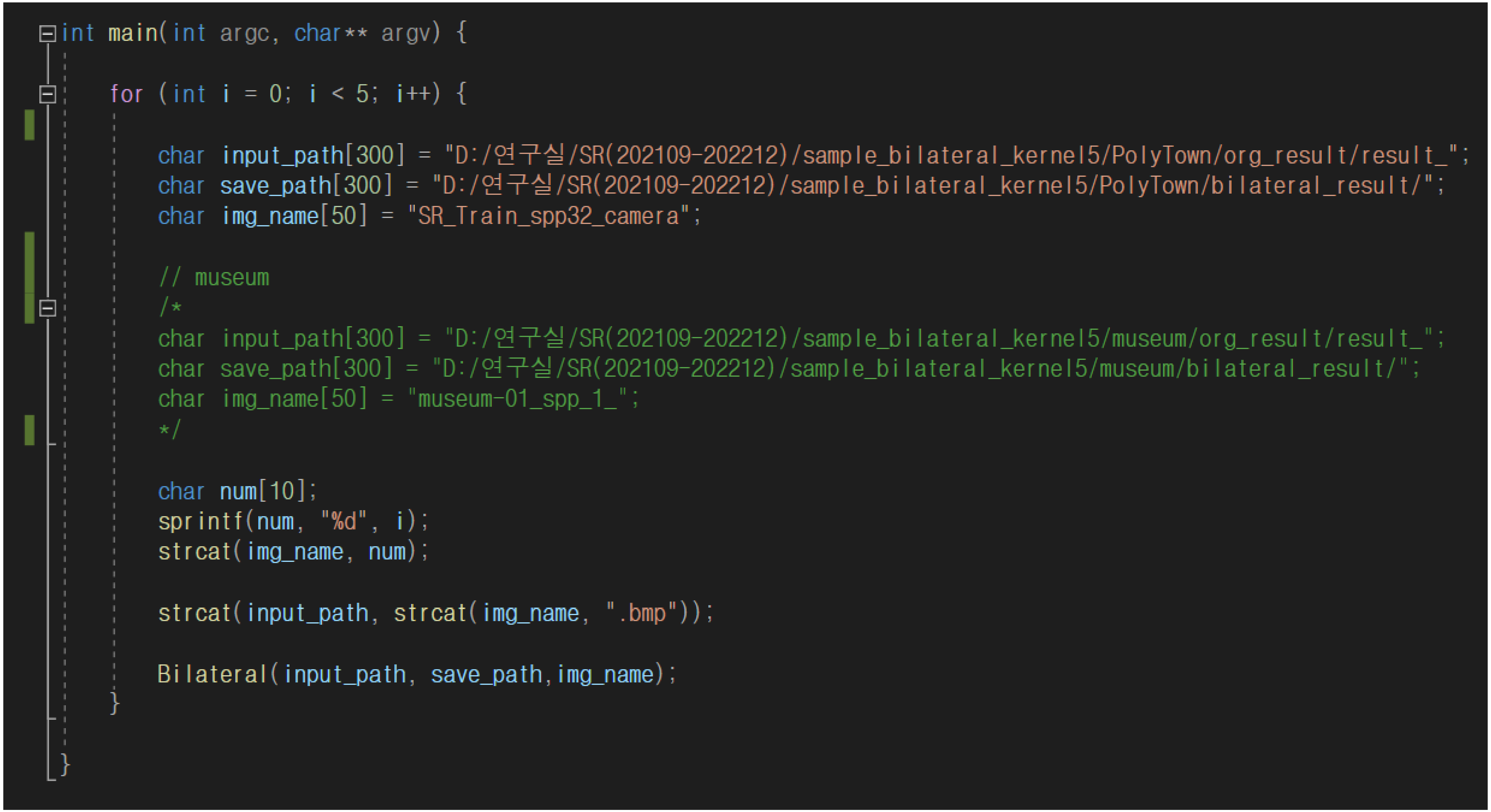
```
Mat bilateralFilterOwn(Mat source, int diameter, double sigmaI, double sigmaS) {
    clock_t s = clock();
    Mat filteredImage = Mat::zeros(source.rows, source.cols, CV_64F);
    int width = source.cols;
    int height = source.rows;

    for (int i = 2; i < height - 2; i++) {
        for (int j = 2; j < width - 2; j++) {
            applyBilateralFilter(source, filteredImage, i, j, diameter, sigmaI, sigmaS);
        }
    }
    printf("time: %.3f₩n", (float)(clock() - s) / CLOCKS_PER_SEC);

    return filteredImage;
}
```



```
int Bilateral(char* input_path, char* save_path, char* img_name) {

    Mat src;
    Mat label;
    vector<Mat> bgr_images(3);
    vector<Mat> r_bgr_images(3);
    vector<Mat> cvr_bgr_images(3);
    vector<Mat> label_bgr_images(3);
    printf("%s₩n", input_path);
    src = imread(input_path, 1);
    split(src, bgr_images);
    printf("%d, %d, %d₩n", src.cols, src.rows, src.channels());

    if (!src.data)
    {
        printf("No image data ₩n");
        return -1;
    }


    // 구현 filter 사용
    Mat filteredBlue = bilateralFilterOwn(bgr_images[0], 5, 12.0, 16.0);
    Mat filteredGreen = bilateralFilterOwn(bgr_images[1], 5, 12.0, 16.0);
    Mat filteredRed = bilateralFilterOwn(bgr_images[2], 5, 12.0, 16.0);

    r_bgr_images[2] = filteredRed;
    r_bgr_images[1] = filteredGreen;
    r_bgr_images[0] = filteredBlue;
```



```
    Mat filteredImageOwn_bgr;
    merge(r_bgr_images, filteredImageOwn_bgr);

    char png_name[30];
    int k = 0;
    while (img_name[k] != '.') {
        png_name[k] = img_name[k];
        k++;
    }
    png_name[k] = '₩0';
    strcat(save_path, png_name);
    strcat(save_path, ".png");
    imwrite(save_path, filteredImageOwn_bgr);
    printf("save path: %s₩n", save_path);

    return 0;
}
```



```c
int main(int argc, char** argv) {

    for (int i = 0; i < 5; i++) {

        char input_path[300] = "D:/연구실/SR(202109-202212)/sample_bilateral_kernel5/PolyTown/org_result/result_";
        char save_path[300] = "D:/연구실/SR(202109-202212)/sample_bilateral_kernel5/PolyTown/bilateral_result/";
        char img_name[50] = "SR_Train_spp32_camera";

        // museum
        /*
        char input_path[300] = "D:/연구실/SR(202109-202212)/sample_bilateral_kernel5/museum/org_result/result_";
        char save_path[300] = "D:/연구실/SR(202109-202212)/sample_bilateral_kernel5/museum/bilateral_result/";
        char img_name[50] = "museum-01_spp_1_";
        */

        char num[10];
        sprintf(num, "%d", i);
        strcat(img_name, num);

        strcat(input_path, strcat(img_name, ".bmp"));

        Bilateral(input_path, save_path,img_name);
    }

}
```