영상처리 실제 6주차 실습_공간필터링

2023254015 장욱진

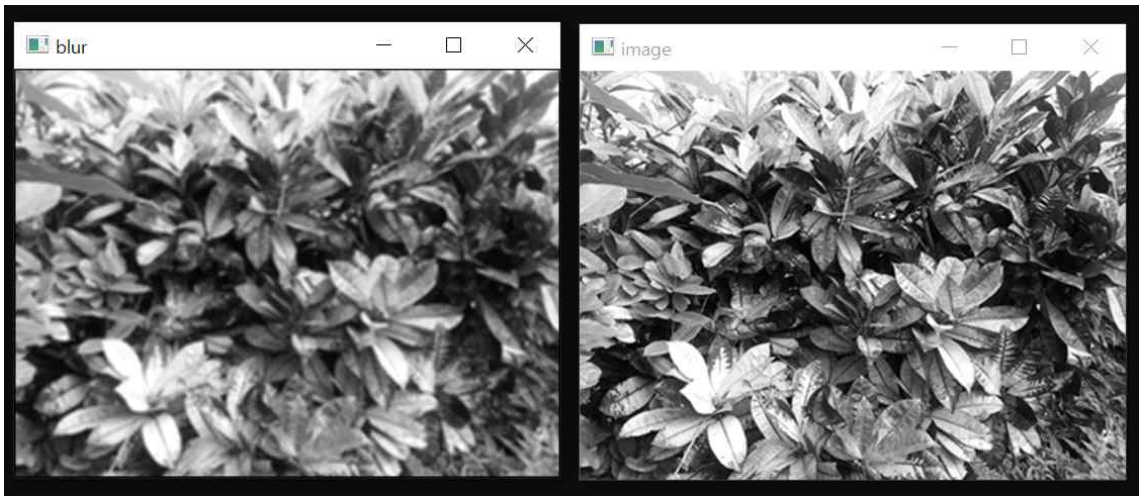
```
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;

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++) // 입력 행렬 반복 순회
	{
		for (int j = h_m.x; j < img.cols; j++)
		{
			float sum = 0;

			for (int u = 0; u < mask.rows; u++) //마스크 원소를 순회한다.
			{
				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); //회선 수식!
				}

			}
			dst.at<float>(i, j) = sum; //회선 누적값 출력화소 저장!
		}

	}
}

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, dst1, CV_32F, mask2);
	magnitude(dst1, dst2, dst);
	dst.convertTo(dst, CV_8U);

	convertScaleAbs(dst1, dst1);
	convertScaleAbs(dst2, dst2);

	imshow("dst1 - 수직 마스크", dst1);
	imshow("dst2 - 수평 마스크", dst2);
}

Mat CannyThreshold_src, CannyThreshold_detected_edges, CannyThreshold_dst;
int lowThreshold;
int const max_lowThreshold = 100;
int ratio_1 = 3;
int kernel_size = 3;

static void CannyThreshold(int, void*)
{
	blur(CannyThreshold_src, CannyThreshold_detected_edges, Size(3, 3));
	Canny(CannyThreshold_detected_edges,		CannyThreshold_detected_edges,
lowThreshold, lowThreshold * ratio_1, kernel_size);
	CannyThreshold_dst = Scalar::all(0);
```

```
        CannyThreshold_src.copyTo(CannyThreshold_dst,
CannyThreshold_detected_edges);
        imshow("Image", CannyThreshold_src);
        imshow("Canny", CannyThreshold_dst);
}

void page8()
{
        Mat image = imread("./filter_blur.jpg", IMREAD_GRAYSCALE);

        CV_Assert(image.data);

        float data[] =
        {                                                       //샤프닝 마스크 지정
                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); //회선 수행
        blur.convertTo(blur, CV_8U);


        imshow("image", image), imshow("blur", blur);
        waitKey(0);
}

void page10()
{
        Mat image = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
        float weights[] = {
        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, weights);
        Mat blur;
        filter2D(image, blur, -1, mask);
        blur.convertTo(blur, CV_8U);
        imshow("image", image);
        imshow("blur", blur);
        waitKey(0);
}

void page16()
{
        Mat image = imread("./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
        };
```

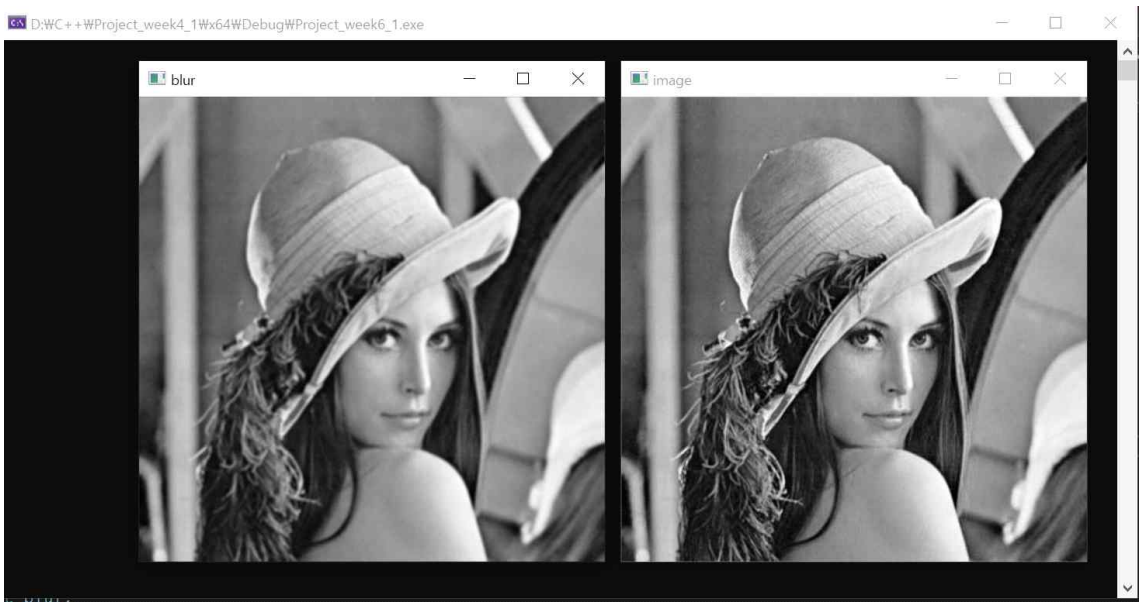
```
	Mat mask1(3, 3, CV_32F, data1);
	Mat mask2(3, 3, CV_32F, data2);

	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();
}

int page21()
{
	Mat src = imread("./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_img 의 모든 화소를 0 부터 255 까지의 난수로 채움

	Mat black_img = noise_img < 10; // noise_img 의 화소값이 10 보다 작으면 1이되는 black_img 생성
	Mat white_img = noise_img > 245; // noise_img 의 화소값이 245 보다 크면 1이되는 white_img 생성
	Mat src1 = src.clone();

	src1.setTo(255, white_img); // white_img 의 화소값이 1 이면 src1 화소값을 255 로 한다=> salt noise
	src1.setTo(0, black_img); // black_img 의 화소값이 1 이면 src1 화소값을 0 으로 한다=> pepper noise

	medianBlur(src1, dst, 5);
	imshow("source", src1);
	imshow("result", dst);

	waitKey(0);
	return 0;
}

void page27()
{
	Mat image = imread("./sample.jpg", IMREAD_GRAYSCALE);

	CV_Assert(image.data);

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

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

	Mat dst;
```

```
	differential(image, dst, data1, data2);
	imshow("image", image), imshow("프리윗에지", dst);
	waitKey();
}

int page37()
{
	Mat src, src_gray, dst;

	int kernel_size = 3;
	int scale = 1;
	int delta = 0;
	int ddepth = CV_16S;

	src = imread("./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(0);

	return 0;
}

int page44()
{
	CannyThreshold_src = imread("./lenna.jpg", IMREAD_GRAYSCALE);
	if (CannyThreshold_src.empty()) { return -1; }
	CannyThreshold_dst.create(CannyThreshold_src.size(),
CannyThreshold_src.type());
	namedWindow("Canny", CV_WINDOW_AUTOSIZE);
	createTrackbar("Min Threshold:", "Canny", &lowThreshold, max_lowThreshold,
CannyThreshold);
	CannyThreshold(0, 0);
	waitKey(0);

	return 0;
}

int main()
{
	page8();
	page10();
	page16();
	page21();
	page27();
	page37();
	page44();

	return 0;
}
```

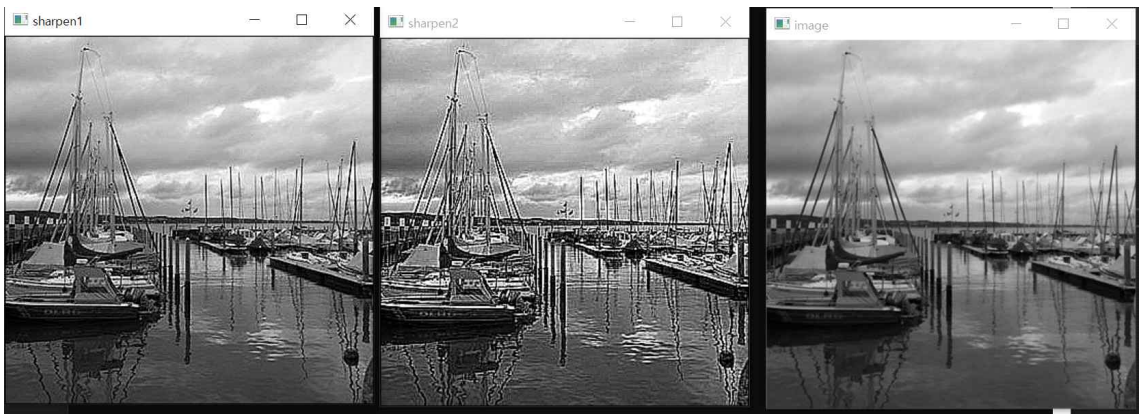
결과화면



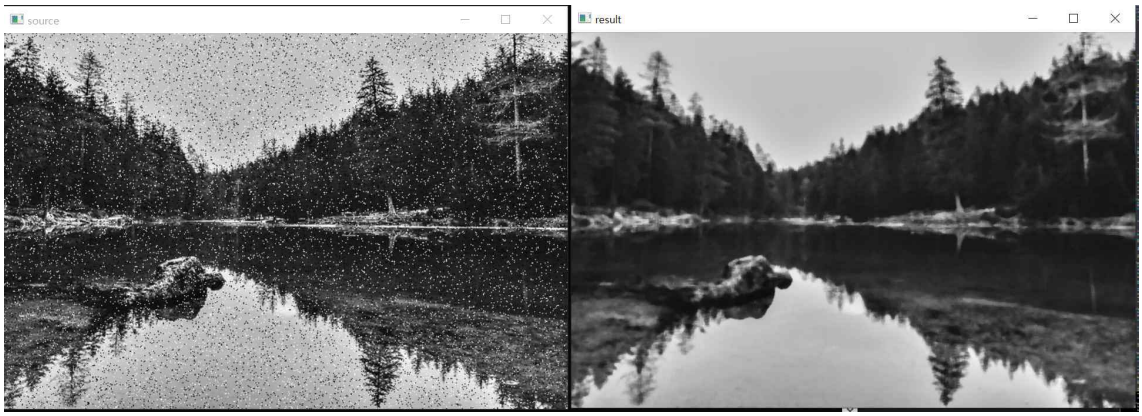
<page8 결과화면>



<page10 결과화면>

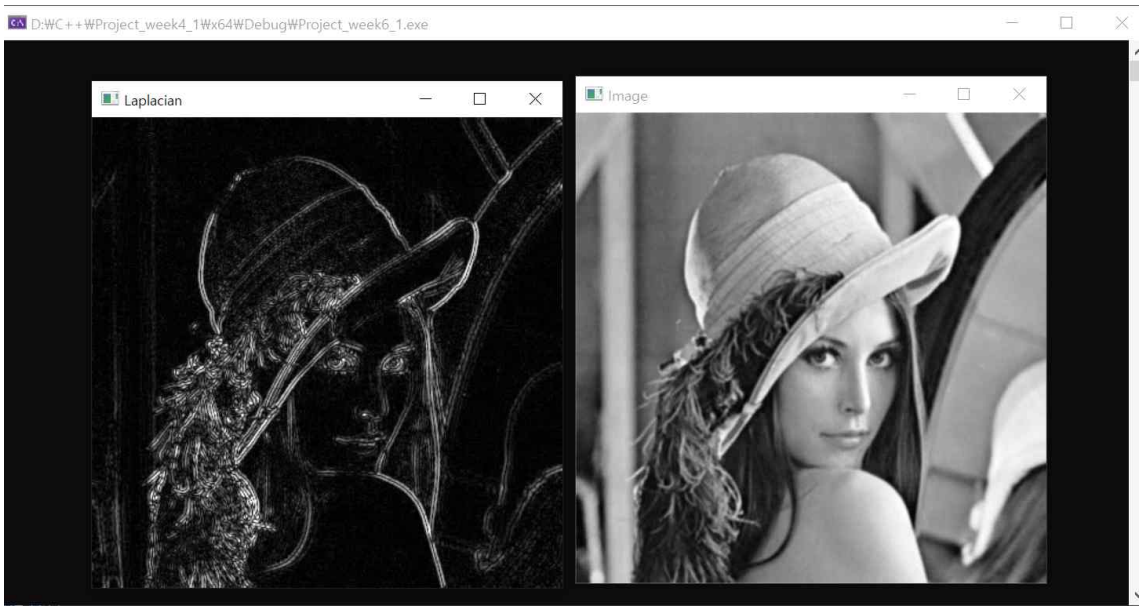


<page16 결과화면>

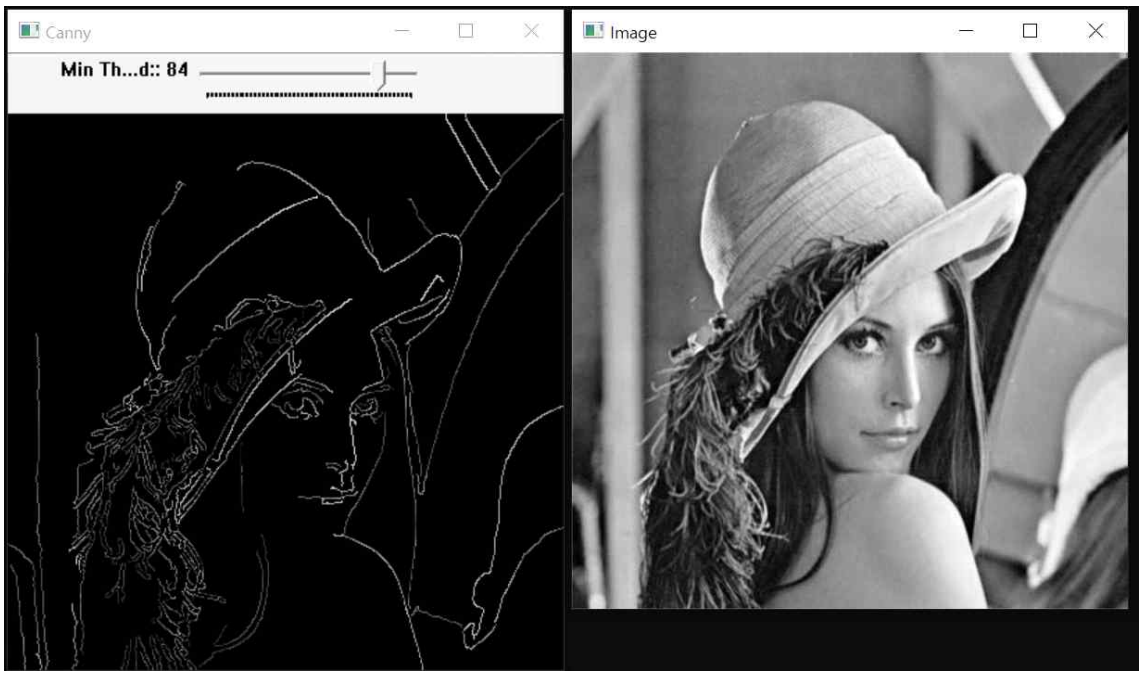


<page21 결과화면>

<page27 결과화면>



<page37 결과화면>



<page44 결과화면>