東義大學校
DONG-EUI UNIVERSITY

[공간영역 필터링 구현]

■ 과 목 명 : 디지털영상처리 I
■ 담 당 교 수 : 김남규
■ 제 출 일 : 2023.05.09
■ 학 과 : 응용SW공학
■ 학 번 : 20213067
■ 성 명 : 신민주

Ⅰ. 평균값 필터링과 가우시안 필터링 차이 비교

1. 평균값 필터링

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

int main() {
	Mat image = imread("d:/images/city.jpg", IMREAD_COLOR);

	// 컨볼루션 마스크
	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
	};

	// 1차원으로 행렬을 저장했었는데 2차원 행렬로 변환함
	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);

	return 0;
}
```

2. 가우시안 필터링

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

int main()
{
	Mat src = imread("d:/images/city.jpg", 1);
	Mat dst;
	imshow("src", src);

	for (int i = 1; i<61; i = i + 2)
	{
		GaussianBlur(src, dst, Size(i, i), 0, 0);
		imshow("Gaussian filter", dst);
		waitKey(1000);
	}
```

}

}

3. 비교

평균값 필터링은 중심 화소의 값을 인접 화소값들의 평균값으로 바꾸는 것을 말한다. 이 때문에 가장자리에 존재하지 않는 값을 더할 수도 있어 필터링에 오류가 발생해 블러링이 자연스럽게 되지 않는다. 위 잡음이 제대로 사라지지 않는 것을 볼 수 있다.

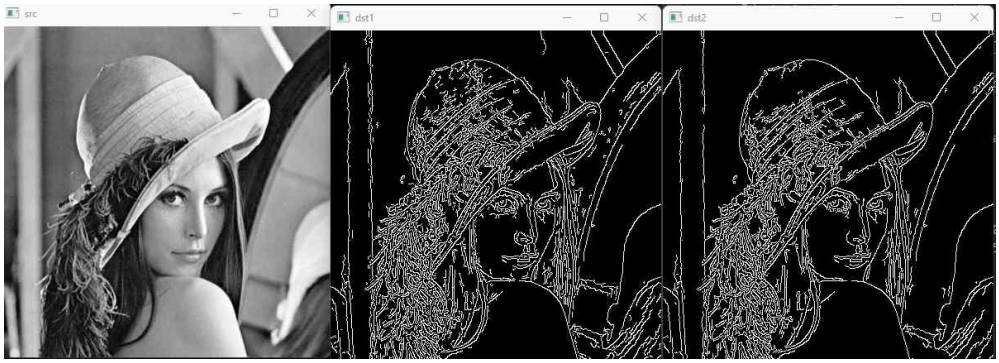
가우시안 필터링은 가우시안 분포를 따르는 가우시안 커널을 이용해서 영상의 잡음을 제거하는 기법이다. 평균 필터는 주변 픽셀에 모두 동일한 가중치를 부여하지만 가우시안은 경계선과 같은 에지 정보를 잘 유지하면서 자연스럽게 스무딩을 적용할 수 있다. 또, 대부분의 컨볼루션 기번의 필터링의 경우 영상의 세밀함이 감소되는 반면에 가우시안 필터링은 세밀함이 잘 보존된다. 여기선 시간이 지남에 따라 블러링이 더 잘되는 것을 알 수 있다. 그렇기 때문에 잡음이 사라지는 것을 알 수 있다.

Ⅱ. 에지 검출 결과 분석

1. canny 코드

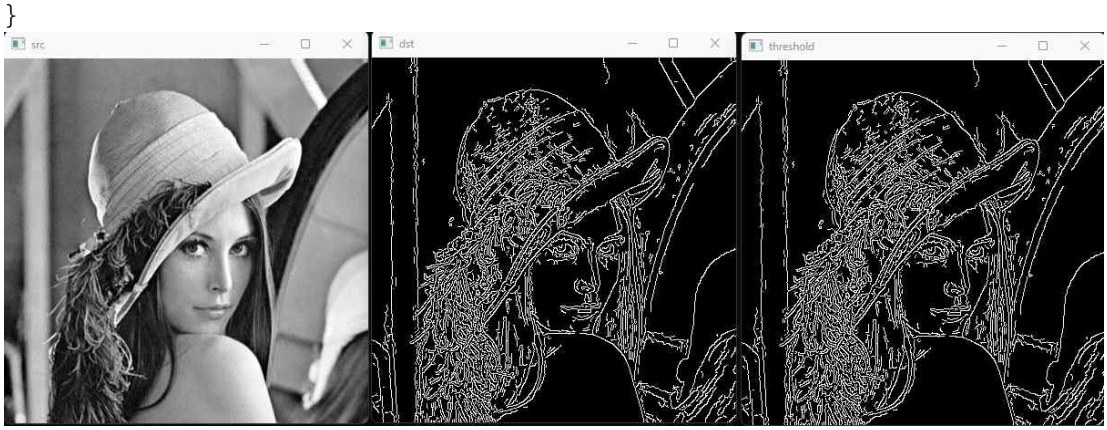
```
#include <iostream>
#include "opencv2/opencv.hpp"

int main(int argc, char* argv[])
{
    cv::Mat src = cv::imread("d:/images/lenna.jpg", cv::IMREAD_GRAYSCALE);

    if (src.empty()) {
        std::cout << "Image load failed!₩n";
        return -1;
    }

    cv::Mat dst1, dst2;
    Canny(src, dst1, 50, 100);
    Canny(src, dst2, 50, 150);

    cv::imshow("src", src);
    cv::imshow("dst1", dst1);
    cv::imshow("dst2", dst2);

    cv::waitKey();
    cv::destroyAllWindows();
```

```
}
```



2. threshold 이진화

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

int main(int argc, char* argv[])
{
    int threshold_value = 127;
    Mat src = imread("d:/images/lenna.jpg", IMREAD_GRAYSCALE);

    if (src.empty()) {
        cout << "Image load failed!₩n";
        return -1;
    }

    Mat dst, thr;
    Canny(src, dst, 50, 100);
    threshold(dst, thr, threshold_value, 255, THRESH_BINARY);

    imshow("src", src);
    imshow("dst", dst);
    imshow("threshold", thr);

    waitKey();
    destroyAllWindows();
}
```



3. 비교

케니 에지와 그 결과를 이진화 한 것과 비교해보면 edge 즉, 테두리를 더 세밀하게 나타내주는 것을 알 수 있다. Canny(src, dst, threshold1, threshold2)로 입력영상, 출력영상, 낮은 경계값, 높은 경계값을 집어넣어서 계산한 값을 threshold(dst, thr, threshold_value, 255, THRESH_BINARY);로 이진화시켜 에지를 자세히 볼 수 있다. 위 내용을 토대로 threshold를 trackbar로 움직이면서 케니

에지의 출력 영상을 다양하게 보여줄 수 있다.

Ⅲ. 모션 블러링

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

int main(int argc, char* argv[])
{
    Mat src = imread("d:/images/1056.jpg");

    if (src.empty()) {
        cout << "Image load failed!₩n";
        return -1;
    }

    // 모션 블러링 적용할 커널 사이즈
    int kernelSize = 20;
    Mat dst;

    blur(src, dst, Size(kernelSize, 1));
    // blur(입력영상, 출력영상, 마스크의 크기 정의)
    imshow("src", src);
    imshow("dst", dst);

    waitKey();
    return 0;
}
```