

6장 공간 필터링

1

공간 필터링이란?

□ 인접 화소들의 값을 참조하여 화소의 값을 변경하는 처리

입력 연산
 $f(x, y)$

연산

출력 연산
 $g(x, y)$

$g(x, y) = T[f(x, y)]$

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

2

1

고급 필터링의 예: 평균 필터링

100	100	100
100	50	100
100	100	100

→

100	100	100
100	94	100
100	100	100

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019

3

```
int main()
{
    Mat src = imread("D:/lenna.jpg", IMREAD_GRAYSCALE);

    Mat dst(src.size(), CV_8U, Scalar(0));
    if (src.empty()) { return -1; }

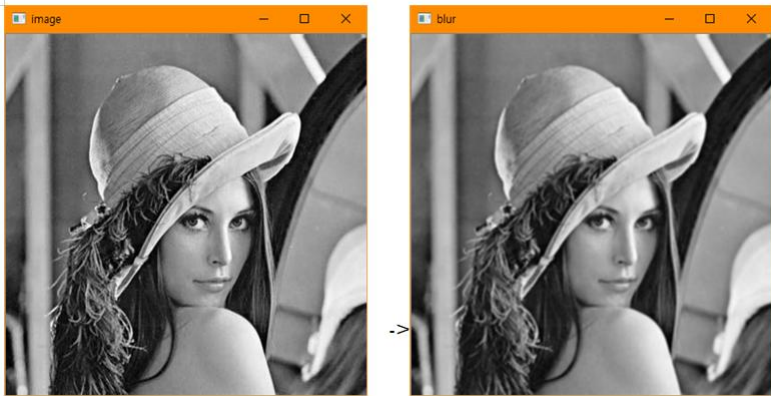
    // (1, 1)부터 (rows-1, cols-1) 까지의 화소만 처리
    for (int y = 1; y < src.rows - 1; y++) {
        for (int x = 1; x < src.cols - 1; x++) {
            int sum = 0;
            sum += src.at<uchar>(y - 1, x - 1);
            sum += src.at<uchar>(y, x - 1);
            sum += src.at<uchar>(y + 1, x - 1);
            sum += src.at<uchar>(y - 1, x);
            sum += src.at<uchar>(y, x);
            sum += src.at<uchar>(y + 1, x);
            sum += src.at<uchar>(y - 1, x + 1);
            sum += src.at<uchar>(y, x + 1);
            sum += src.at<uchar>(y + 1, x + 1);
            dst.at<uchar>(y, x) = sum / 9;
        }
    }
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019

4

실행 결과



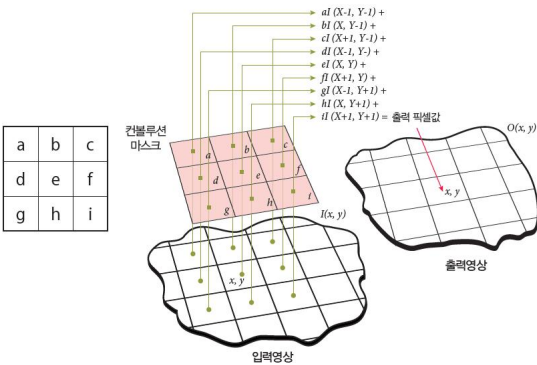
OpenCV로 배우는 디지털 영상 처리 © 인피니티박스 2019



5

컨볼루션

□ 컨볼루션은 중심 화소의 값을 인접 화소값들의 가중 합으로 대체하는 연산이다



OpenCV로 배우는 디지털 영상 처리 © 인피니티박스 2019




6

컨버루션

$$\begin{aligned} O(x, y) = & aI(x - 1, y - 1) + bI(x, y - 1) + cI(x + 1, y - 1) \\ & + dI(x - 1, y) + eI(x, y) + fI(x + 1, y) \\ & + gI(x - 1, y + 1) + hI(x, y + 1) + iI(x + 1, y + 1) \end{aligned} \tag{식 6.1}$$
$$O(x, y) = \sum_{k=-1}^{k=+1} \sum_{l=-1}^{l=+1} h(k, l)I(x + k, y + l)$$

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



7

컨버루션

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

마스크

3	6	6	4	7	8	2	1
3	4	3	8	8	3	3	2
5	7	7	7	7	4	3	2
8	9	9	9	9	9	3	2
8	3	3	4	3	2	1	1
8	9	9	8	8	3	3	2
6	4	3	8	8	3	3	2
7	4	3	8	8	3	3	2


입력 영상

$$\begin{aligned} & 1/9 * 3 + 1/9 * 6 + 1/9 * 6 + \\ & 1/9 * 3 + 1/9 * 4 + 1/9 * 3 + \\ & 1/9 * 5 + 1/9 * 7 + 1/9 * 6 = 4.88 \cong 4 \end{aligned}$$

		4					

출력 영상

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



8

평균 필터링의 구현

```
int main()
{
    Mat image = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);

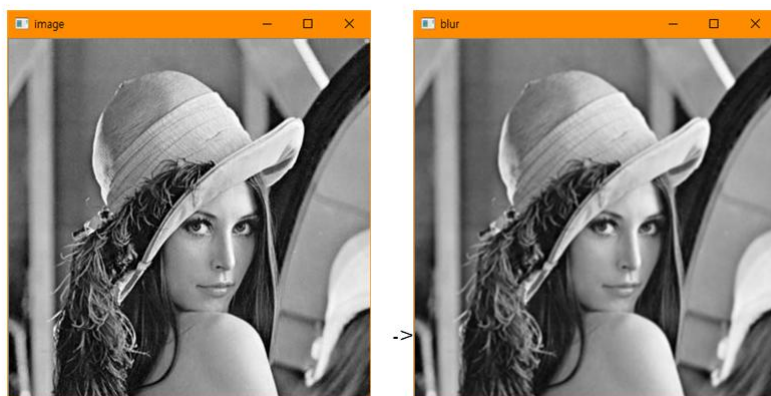
    float weights[] = {
        1 / 9.0F, 1 / 9.0F, 1 / 9.0F,
        1 / 9.0F, 1 / 9.0F, 1 / 9.0F,
        1 / 9.0F, 1 / 9.0F, 1 / 9.0F
    };

    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);
    return 0;
}
```

9

실행 결과



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



10

가장자리 화소의 처리

외부 화소들은 0으로 가정된다.

0⁸0¹0⁶

17241³8⁵15⁷

2357⁴14⁹16²

46132022

101218213

11182529

커널의 중심

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

11

가장자리 화소의 처리

```
void filter2D(Mat src, Mat dst, int ddepth, Mat kernel, Point anchor, double delta, int borderType);
```

매개 변수	설명
BORDER_CONSTANT	iiiiii abcdefgh iiiiiii 여기서 i는 지정된 값
BORDER_REPLICATE	aaaaaa abcdefgh hhhhhhh
BORDER_REFLECT	fedcba abcdefgh hgfedcb
BORDER_WRAP	cdefgh abcdefgh abcdefg

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

12

6

퍼그가 필터링 인식 결과

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

int main()
{
    Mat src = imread("D:/dog.jpg");
    Mat dst;

    blur(src, dst, Size(11, 11));
    imshow("source", src);
    imshow("result", dst);

    waitKey(0);
    return 0;
}
```

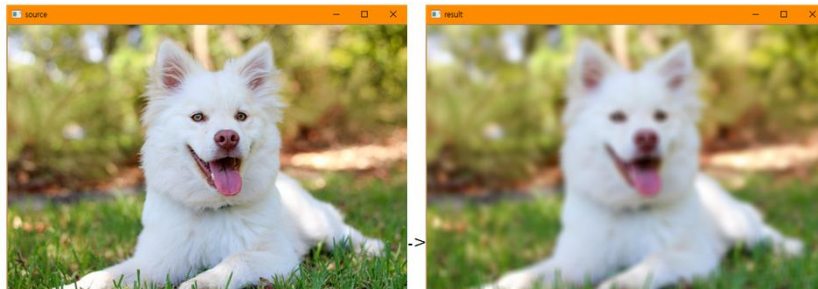
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



13

실행 결과



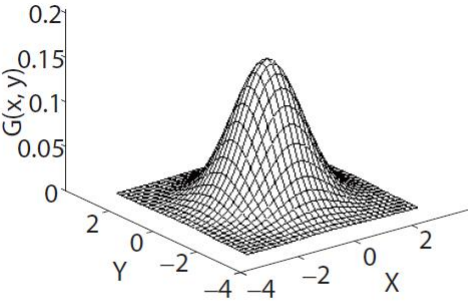
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019




14

가우시안 필터링


$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019




15

가우시안 필터링

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



16

가우시안 필터링

```
void GaussianBlur(InputArray src, OutputArray dst, Size ksize, double
sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT )
```

매개 변수	설명
src	입력 행렬
dst	출력 행렬
ksize	커널의 크기
sigmaX	x 방향의 가우시안 표준 편차
sigmaY	y 방향의 가우시안 표준 편차
borderType	경계선 처리 방법

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



17

가우시안 필터링

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

```
int main()
{
    Mat src = imread("d:/lenna.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);
    }
}
```

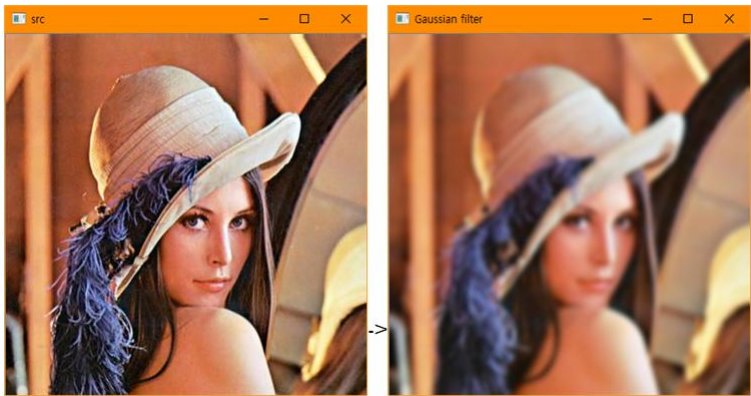
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



18

실행 결과



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



19

샤프닝

□ 샤프닝은 영상을 날카롭게 만드는 처리

-1	-1	-1
-1	9	-1
-1	-1	-1

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



20

샤프닝

```
int main() {
    Mat src = imread("D:/lenna.jpg", IMREAD_COLOR);
    if (src.empty()) return 1;

    Mat dst;
    float weights[9] = { -1, -1, -1, -1, 9, -1, -1, -1, -1 };

    Mat mask = Mat(3, 3, CV_32F, weights);

    filter2D(src, dst, -1, mask, Point(-1, -1), 0, BORDER_DEFAULT);
    imshow("src", src);
    imshow("sharpen", dst);

    waitKey(0);
    return 0;
}
```

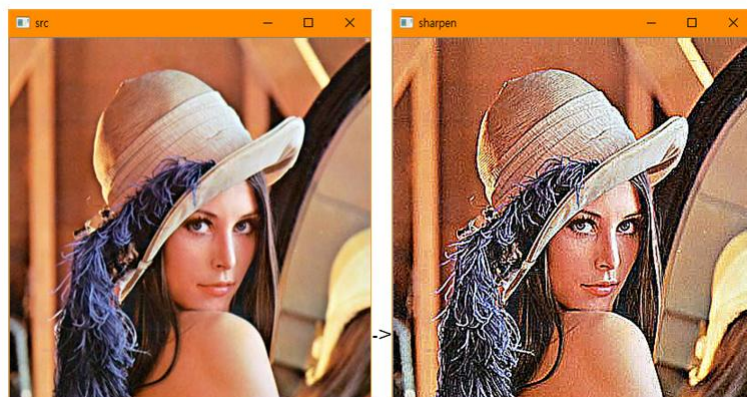
OpenCV로 배우는 디지털 영상 처리

© 인피니티박스 2019



21

실행 결과



OpenCV로 배우는 디지털 영상 처리

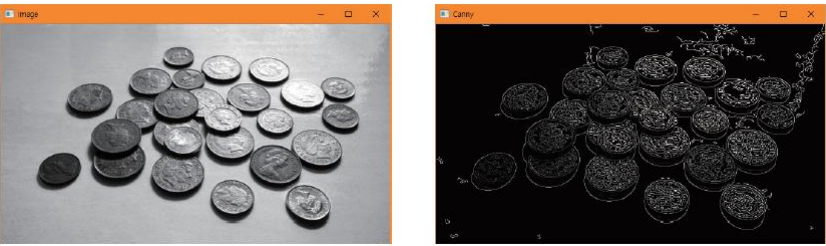
© 인피니티박스 2019



22

에지 검출

- 에지(edge)는 영상에서 화소의 밝기가 급격하게 변하는 부분
- 에지를 검출할 수 있으면 물체의 윤곽선을 알 수 있다.



에지 검출 방법

- 1차 미분값을 이용한 방법,
- 2차 미분값을 이용한 방법
- 그 밖의 방법



영상에서의 기울기 개념

인접 픽셀간의 밝기 기울기

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

25

1 차 미분을 이용한 에지 검출

1차원 영상을 가정하자. 에지는 밝기값이 점프하는 곳이다.

영상을 1차 미분한 그래프이다. 에지는 1차미분의 최대값에 해당한다.

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

26

1 차 미분을 이용한 에지 검출

$$G[f(x, y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$G[f(x, y)] = \sqrt{G_x^2 + G_y^2} \quad \alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$



1 차 미분을 이용한 에지 검출

이름	G_x	G_y
Roberts	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Prewitt	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$



```

int main()
{
    Mat src;
    Mat grad;
    int scale = 1;
    int delta = 0;
    src = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    if (src.empty()) { return -1; }

    Mat grad_x, grad_y;
    Mat abs_grad_x, abs_grad_y;

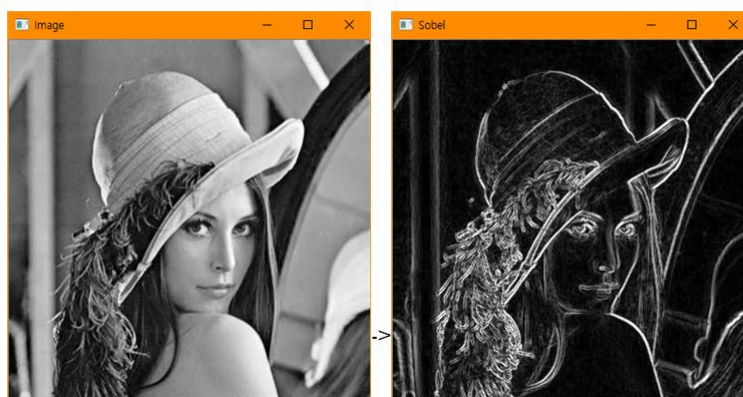
    Sobel(src, grad_x, CV_16S, 1, 0, 3, scale, delta, BORDER_DEFAULT);
    Sobel(src, grad_y, CV_16S, 0, 1, 3, scale, delta, BORDER_DEFAULT);

    convertScaleAbs(grad_x, abs_grad_x);
    convertScaleAbs(grad_y, abs_grad_y);
    addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0, grad);
    imshow("Image", src);
    imshow("Sobel", grad);
    waitKey(0);
    return 0;
}

```

29

실행 결과



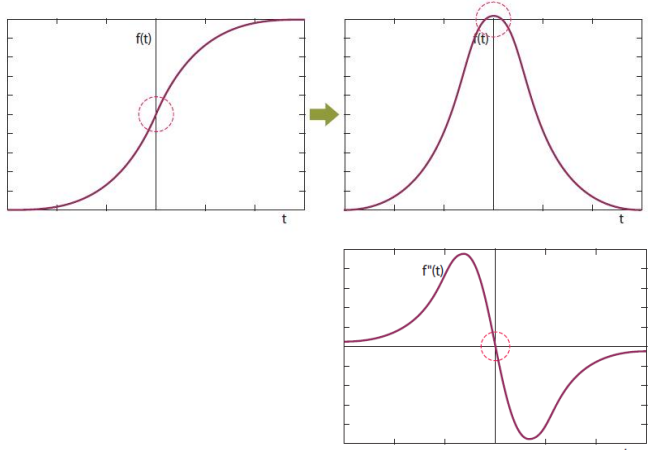
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



30

2차 미분 연산자



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

31

2차 미분 연산자

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

-1	-1	-1
-1	8	-1
-1	-1	-1

또는

0	-1	0
-1	4	-1
0	-1	0

32


```

int main()
{
    Mat src, src_gray, dst;
    int kernel_size = 3;
    int scale = 1;
    int delta = 0;
    int ddepth = CV_16S;
    src = imread("d:/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;
}

```

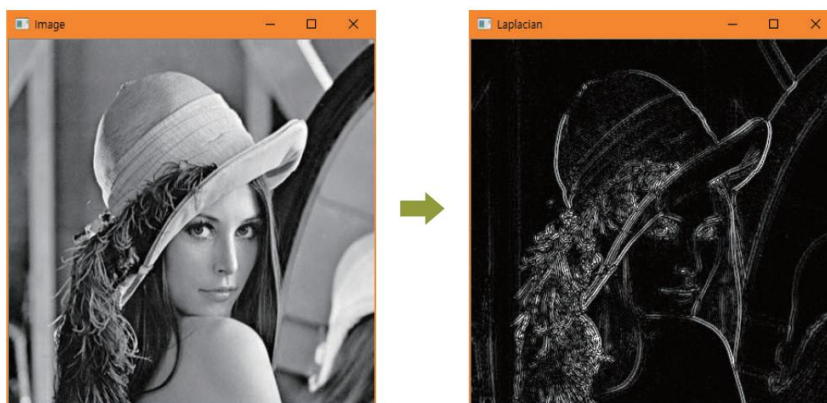
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



33

실행 결과



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



34

캐니 에지 연산자

□ Step #1 잡음 억제

$$K = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



35

캐니 에지 연산자

□ Step #2 그라디언트 계산하기

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

$$G = \sqrt{G_x^2 + G_y^2} \quad \theta = \arctan\left(\frac{G_y}{G_x}\right)$$

OpenCV로 배우는 디지털 영상 처리

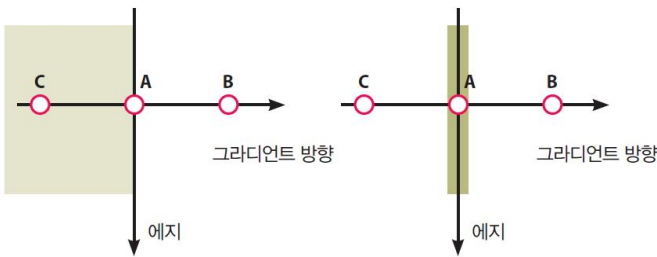
© 인피니티북스 2019



36

캐니 에지 연산자

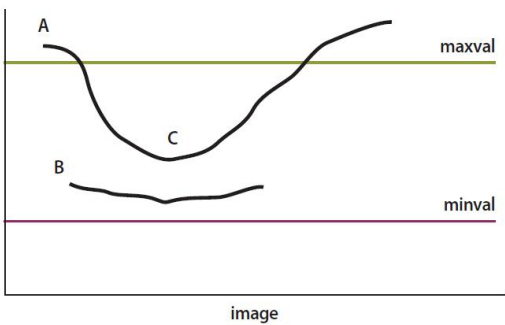
Step #3 비최대 억제



37

캐니 에지 연산자

Step #4 히스테리시스



38

캐니 에지 연산자

```
Canny(src_image, detected_edges, lowThreshold, highThreshold, kernel_size );
```

매개 변수	설명
src_image	입력 영상
detected_edges	출력 영상(입력 영상과 같을 수 있음)
lowThreshold	하위 임계값
highThreshold	상위 임계값
kernel_size	3(내부적으로 사용되는 Sobel 마스크의 크기)으로 정의한다.

OpenCV로 배우는 디지털 영상 처리 © 인피니티박스 2019

```
static void CannyThreshold(int, void*)
{
    blur(src, detected_edges, Size(3, 3));
    Canny(detected_edges, detected_edges, lowThreshold,
lowThreshold*ratio, kernel_size);
    dst = Scalar::all(0);
    src.copyTo(dst, detected_edges);
    imshow("Image", src);
    imshow("Canny", dst);
}
int main()
{
    src = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    if (src.empty()) { return -1; }
    dst.create(src.size(), src.type());
    namedWindow("Canny", CV_WINDOW_AUTOSIZE);
    createTrackbar("Min Threshold:", "Canny", &lowThreshold,
max_lowThreshold, CannyThreshold);
    CannyThreshold(0, 0);
    waitKey(0);
    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리 © 인피니티박스 2019

실행 결과







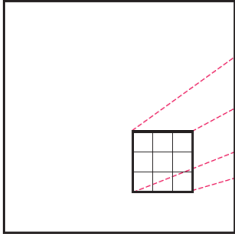
OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



41

조각가 필터링

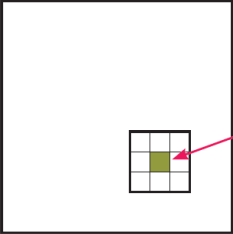
입력 영상



필터 윈도우

13	4	8
2	9	6
7	25	16

출력



정렬된 화소들

2

4

6

7

8


9

13

16

25

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



42

중간값 필터링

	-3	3	3	3	3	7	7	7	7	7	
	-3	3	3	3.3	4.7	5.7	6.7	6.7	6.7	7	
	-3	3	3	3	3	7	7	7	7	7	

스무딩 필터링 후의 영상(에지가 망가진다)

	-3	3	3	3	3	7	7	7	7	7	
	-3	3	3	3	3	7	7	7	7	7	
	-3	3	3	3	3	7	7	7	7	7	

중간값 필터링 후의 영상(에지가 보존된다)

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



43

```
int main()
{
    Mat src = imread("d:/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);

    Mat black_img = noise_img < 10;
    Mat white_img = noise_img > 245;

    Mat src1 = src.clone();
    src1.setTo(255, white_img);
    src1.setTo(0, black_img);
    medianBlur(src1, dst, 5);
    imshow("source", src1);
    imshow("result", dst);

    waitKey(0);
    return 0;
}
```

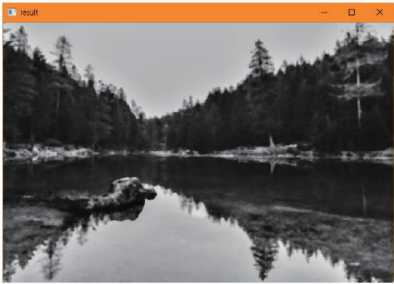
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



44

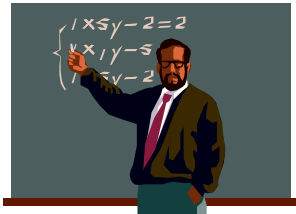
실행 결과



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



Q & A



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

