8장 모폴로지 처리

1

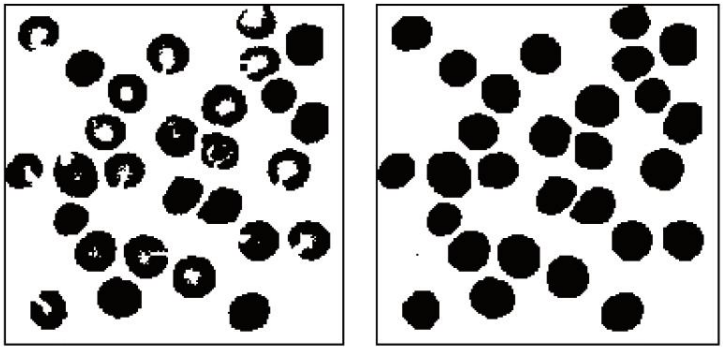
형태학적 처리란?

- 형태학적 처리(mathematical morphology)란 특정한 모양의 형태소(structuring element)를 영상에 적용하여 출력 영상을 생성하는 연산

I. 형태학적 필터링
II. 잡음 제거
III. 세선화
IV. 골격화

2

형태학적 처리란?

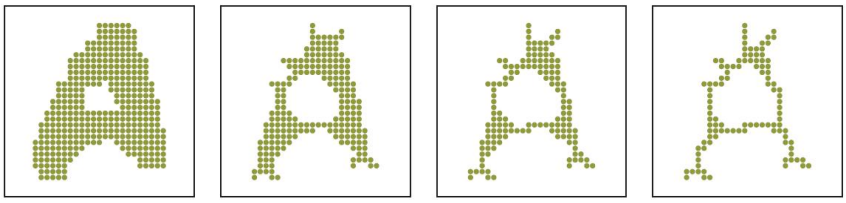


A B

출처: National Instrument

3

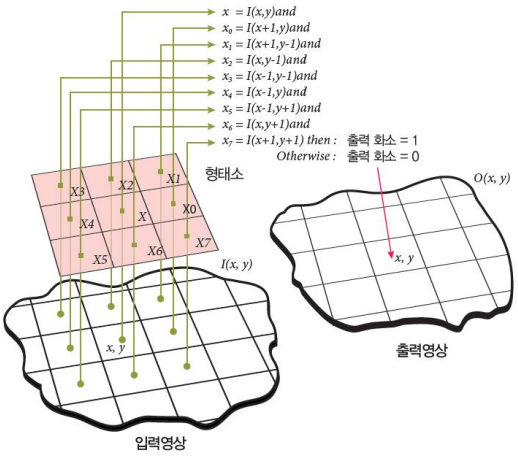
형태학적 처리란?



출처: PubMed, LicenseCC BY 3.0

4

침식과 팽창 연산



5

침식 연산

0	0	1
1	1	1
1	1	1

입력 화소

⊖

1	1	1
1	1	1
1	1	1

형태소

→ 하나라도 일치하지 않으면 → [0] 출력 화소

1	1	1
1	1	1
1	1	1

입력 화소

⊖

1	1	1
1	1	1
1	1	1

형태소

→ 전부 일치하면 → [0] 출력 화소

6

침식 연산

0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1
0	1	1	1	0	1	1	1
1	1	1	0	0	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

7

팽창 연산

0	0	0
0	0	0
0	0	0

입력 화소

⊖

1	1	1
1	1	1
1	1	1

형태소

→ 모두 불일치하면 $\boxed{0}$

출력 화소

0	0	0
0	0	1
0	0	1

입력 화소

⊖

1	1	1
1	1	1
1	1	1

형태소

→ 하나라도 일치하면 $\boxed{1}$

출력 화소

8

팽창 연산

0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1
0	1	1	1	0	1	1	1
1	1	1	0	0	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1

9

```
Mat getStructuringElement(int shape, Size ksize, Point anchor=Point(-1,-1))
```

매개 변수	설명
shape	연산에 사용되는 형태소의 종류. 3가지 형태 중 하나를 선택할 수 있다. • 사각형 상자: MORPH_RECT • 십자가: MORPH_CROSS • 타원형: MORPH_ELLIPSE
ksize	형태소의 크기
anchor	형태소의 기준 위치. 만약 (−1, −1)로 지정하면 기준점은 형태소의 중심이 된다.

10

```
void erode(InputArray src, OutputArray dst, InputArray kernel)
```

매개 변수	설명
src	소스 이미지
dst	출력 이미지
kernel	우리가 침식을 수행하는 데 사용할 형태소이다. 우리가 모양을 지정하지 않으면 디폴트는 단순한 3x3행렬이다. 형태소를 생성하기 위하여 getStructuringElement()함수를 사용한다.

11

```
int main()
{
        Mat src, dst, erosion_dst, dilation_dst;
        src = imread("d:/test.png", IMREAD_GRAYSCALE);

        threshold(src, dst, 127, 255, THRESH_BINARY);
        imshow("dst", dst);

        Mat element = getStructuringElement(MORPH_RECT,
                Size(3, 3),
                Point(-1, -1));

        erode(dst, erosion_dst, element);
        imshow("Erosion Demo", erosion_dst);
        waitKey(0);
        return 0;
}
```


12

실행 결과

13

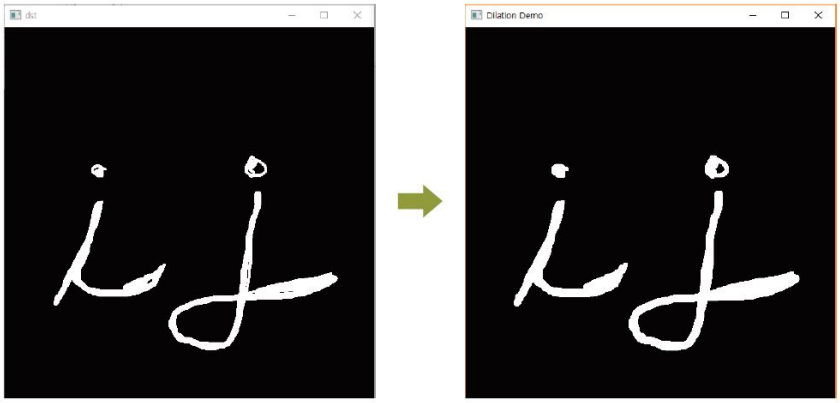
팽창 연산

```
void dialte(InputArray src, OutputArray dst, InputArray kernel)
```

매개 변수	설명
src	소스 이미지
dst	출력 이미지
kernel	우리가 침식을 수행하는 데 사용할 형태소이다. 우리가 모양을 지정하지 않으면 디폴트는 단순한 3 × 3행렬이다. 형태소를 생성하기 위하여 getStructuringElement()함수를 사용한다.

14

팽창 연산



15

열림 연산

- 열림 연산 opening 은 침식 연산 다음에 팽창 연산이 이어지는 것이다.
- dst = open(src, element) = dilate(erode(sec, element))

0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1
0	1	1	1	0	1	1	1
1	1	1	0	0	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1
0	1	1	1	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

16

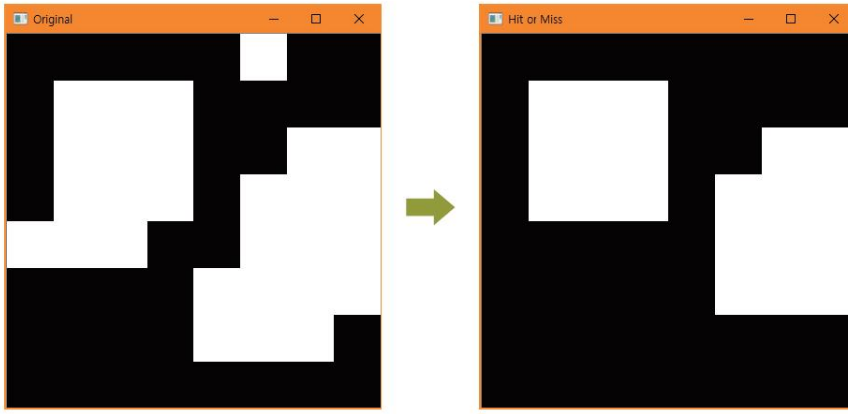
```
int main()
{
        Mat input_image = (Mat_<uchar>(8, 8) <<
                0, 0, 0, 0, 0, 255, 0, 0,
                0, 255, 255, 255, 0, 0, 0, 0,
                0, 255, 255, 255, 0, 0, 255, 255,
                0, 255, 255, 255, 0, 255, 255, 255,
                255, 255, 255, 0, 0, 255, 255, 255,
                0,0, 0, 0, 255, 255, 255, 255,
                0, 0, 0, 0, 255, 255, 255, 0,
                0, 0, 0, 0, 0, 0, 0, 0 );
        Mat kernel = (Mat_<int>(3, 3) <<
                1, 1, 1,
                1, 1, 1,
                1, 1, 1);
        Mat output_image;
```


17

```
        morphologyEx(input_image, output_image, MORPH_OPEN, kernel);

        const int rate = 50;
        resize(input_image, input_image, Size(), rate, rate, INTER_NEAREST);
        imshow("Original", input_image);
        resize(output_image, output_image, Size(), rate, rate, INTER_NEAREST);
        imshow("Open", output_image);
        waitKey(0);
        return 0;
}
```


18

실행 결과



19

닫힘 연산

- 닫힘 연산은 제거 연산의 반대가 된다. 즉 먼저 팽창 연산 다음에 침식 연산이 행해진다.
- dst = elose(src, element) = erode(dilate(sec, element))

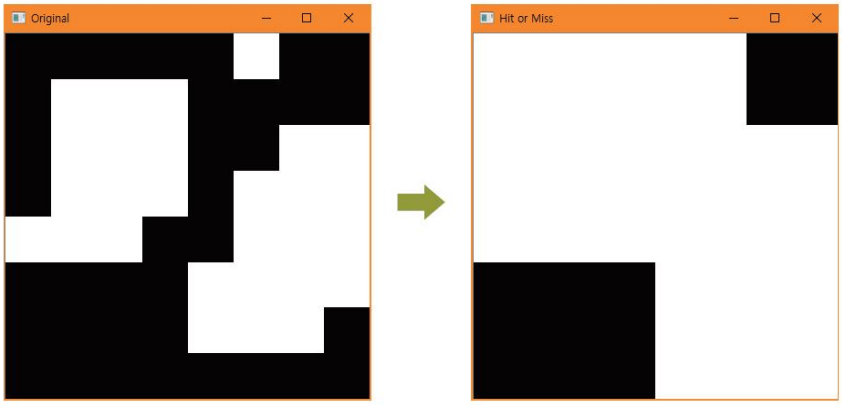
0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1
0	1	1	1	0	1	1	1
1	1	1	0	0	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1

1	1	1	1	1	1	0	0
1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

20

달침 연산



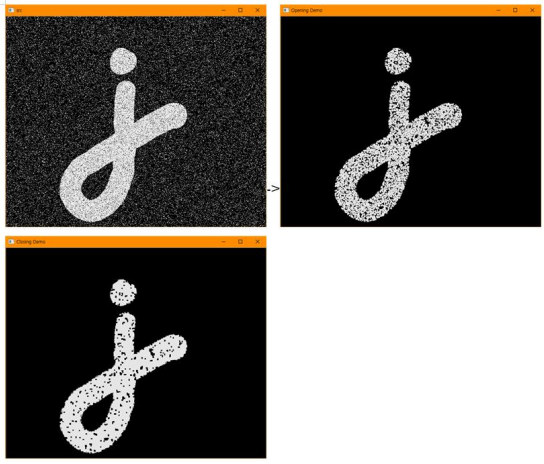
21

```
int main()
{
        Mat src, dst, open, close;
        src = imread("d:/test1.png", IMREAD_GRAYSCALE);

        Mat saltpepper_noise = Mat::zeros(src.rows, src.cols, CV_8U);
        randu(saltpepper_noise, 0, 255);

        Mat black = saltpepper_noise < 30;
        Mat white = saltpepper_noise > 225;

        Mat saltpepper_img = src.clone();
        saltpepper_img.setTo(255, white);
        saltpepper_img.setTo(0, black);

        imshow("src", saltpepper_img);
```

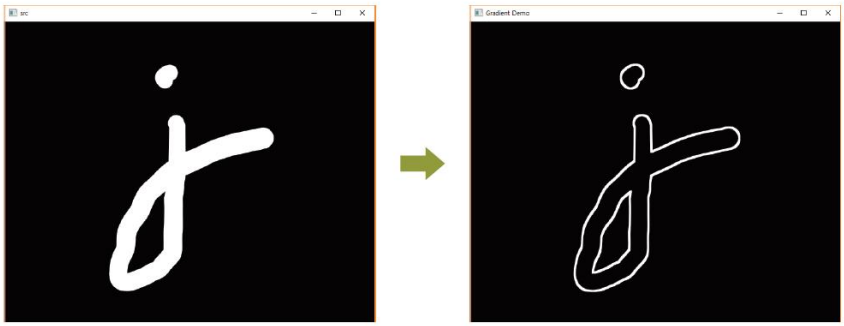

22

```
        Mat element = getStructuringElement(MORPH_RECT, Size(3, 3));

        morphologyEx(saltpepper_img, open, MORPH_OPEN, element);
        imshow("Opening Demo", open);

        morphologyEx(open, close, MORPH_CLOSE, element);
        imshow("Closing Demo", close);
        waitKey(0);
        return 0;
}
```


실행 결과



형태학적인 그라디언트



25

형태학적인 그라디언트

- 외곽선을 추출하려면 먼저 영상에 침식 연산을 적용한다. 앞에서 설명한 바와 같이 영상 내의 물체는 한 화소 축소 된다. 그 다음에 원 영상에서 침식 영상을 뺀다. 그 결과는 물체의 외곽선만을 보여주는 영상이다

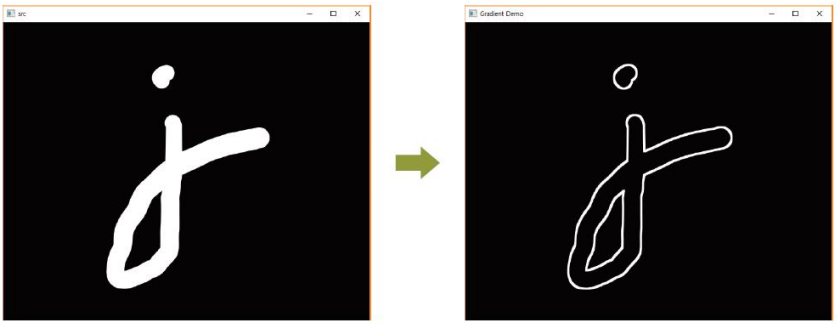
$$A - (A \ominus B)$$

26

```
int main()
{
        Mat src, dst, open, close;
        src = imread("d:/letterj.png", IMREAD_GRAYSCALE);

        imshow("src", src);

        Mat element = getStructuringElement(MORPH_RECT, Size(5, 5));
        morphologyEx(src, open, MORPH_GRADIENT, element);
        imshow("Gradient Demo", open);
        waitKey(0);
        return 0;
}
```


27

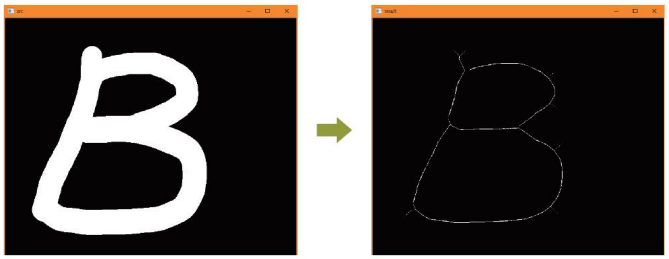
실행 결과



28

골격화

- 골격화는 골격선을 구하는 연산으로 중심축medial axis , 또는 세선화 thinning 라고도 불리며 어떤 물체의 중심을 지나는 직선이나 곡선을 말한다



29

알고리즘

```
img = 입력 영상;
do
{
    skeleton  = skeleton | (img - open(img));
    img = erosion(img);
} while(img가 빈 영상이 될 때까지)
```


30

코드

```
int main()
{
        Mat img = imread("d:/letterb.png", CV_LOAD_IMAGE_GRAYSCALE);
        threshold(img, img, 127, 255, cv::THRESH_BINARY);

        imshow("src", img);
        Mat skel(img.size(), CV_8UC1, Scalar(0));
        Mat element = getStructuringElement(MORPH_CROSS, Size(3, 3));
        Mat temp, eroded;
```

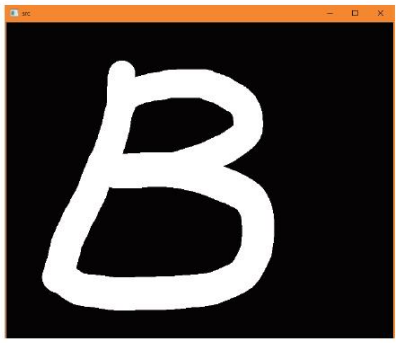

31

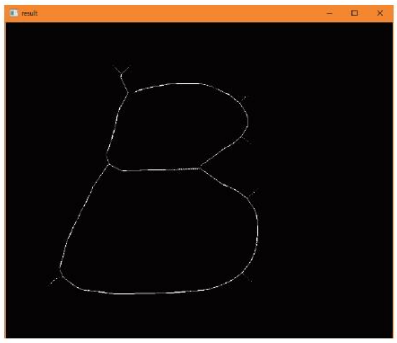
코드

```
        do
        {
                erode(img, eroded, element);
                dilate(eroded, temp, element);
                subtract(img, temp, temp);
                bitwise_or(skel, temp, skel);
                eroded.copyTo(img);
        } while ((countNonZero(img) != 0));

        imshow("result", skel);
        waitKey(0);
        return 1;
}
```


32

실행 결과





33

Hit-or-Miss 변환

- Hit-or-Miss 변환Hit-or-Miss transform 은 영상에서 특정한 패턴을 찾는 데 사용할 수 있는 이진 형태학적 연산이다

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

34

0	0	0	0	0	0	0	0
0	255	255	255	0	0	0	255
0	255	255	255	0	0	0	0
0	255	255	255	0	255	0	0
0	0	255	0	0	0	0	0
0	0	255	0	0	255	255	0
0	255	0	255	0	0	255	0
0	255	255	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0
0	0	255	0	0	0	0	0
0	0	0	0	0	0	0	0

35

코드

```
int main()
{
        Mat input_image = (Mat_<uchar>(8, 8) <<
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 255, 255, 255, 0, 0, 0, 255,
                0, 255, 255, 255, 0, 0, 0, 0,
                0, 255, 255, 255, 0, 255, 0, 0,
                0, 0, 255, 0, 0, 0, 0, 0,
                0, 0, 255, 0, 0, 255, 255, 0,
                0, 255, 0, 255, 0, 0, 255, 0,
                0, 255, 255, 255, 0, 0, 0, 0);
        Mat kernel = (Mat_<int>(3, 3) <<
                0, 1, 0,
                1, -1, 1,
                0, 1, 0);
        Mat output_image;
```

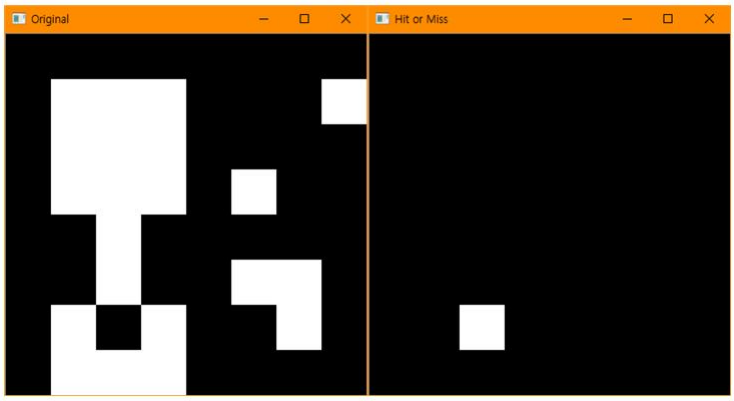

36

코드

```
        morphologyEx(input_image, output_image, MORPH_HITMISS, kernel);

        // 영상을 확대하여 표시한다.
        const int rate = 50;
        resize(input_image, input_image, Size(), rate, rate, INTER_NEAREST);
        imshow("Original", input_image);
        resize(output_image, output_image, Size(), rate, rate, INTER_NEAREST);
        imshow("Hit or Miss", output_image);
        waitKey(0);
        return 0;
}
```


37

실행 결과



38

Q & A