



강의 참고자료

Morphological Operations

2024학년도 1학기
서경대학교 김진헌

차례

- 1. 모폴로지 변환의 종류
 - ▣ 1.1 Erosion: 침식
 - ▣ 1.2 Dilation: 팽창
- 2. Structuring Element(kernel)에 대하여..
 - ▣ 2.1 getStructuringElement() 함수
 - ▣ 2.2 함수에서 반환된 커널의 출력 사례
- 3. 모폴로지 연산의 효과
 - ▣ 3.1 Effect of Erosion
 - ▣ 3.2 Effect of Dilation
- 4. 기본 모폴로지 함수- dilate & erode
- 5. 이진영상과 그레이 영상의 처리 효과 비교
- 6. 모폴로지 연산의 변형
- 7. morphologyEx() 함수

1. 모폴로지 변환의 종류

OpenCV 참조 링크: [Morphological Transformations](#)

- * Morphological filtering is a technique for reducing noise and details of an image while preserving lines and shapes.
- 기본적인 모폴로지 동작
 - ▣ 침식(Erosion) : 배경(background, 어두운 화소)의 수가 증가
 - 이진영상의 경우 1의 화소 수가 감소. 1로 구성된 전경의 크기가 줄어든다.
 - ▣ 팽창(Dilation) : 전경(foreground, 밝은 화소)의 수가 증가
 - 이진영상의 경우 1로 구성된 전경의 경계가 늘어나 전경의 크기가 커진다.
- 혼합 처리
 - ▣ Opening & Closing: 팽창과 침식을 결합
 - ▣ Top Hat & Black Hat: 팽창과 침식의 차분

1.1 Erosion: 침식

<http://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm>

□ Erosion: 침식

- ▣ 주어진 **structuring element**(혹은 **kernel**)에 해당하는 위치에 있는 화소 값 중에 가장 작은 값을 반환한다.
- ▣ 반환되는 값은 커널의 중심부의 위치에 대응하는 화소 값을 대체한다.
- ▣ 커널이 커지거나 반복 실행하면 침식 효과도 크다.

$$\text{dst}(x, y) = \min_{(x', y') : \text{element}(x', y') \neq 0} \text{src}(x + x', y + y')$$

커널의 원소 값이 0이 아닌 위치에 있는 소스 영상의 화소 값 중 가장 작은 값을 반환한다.

□ 실행결과

- ▣ 계조 영상: 밝은 화소는 줄어들고 어두운 화소가 증가된다.
- ▣ 이진 영상: 커널 안의 이진 화소 중 하나라도 0이면 0을 출력하므로 1로 구성된 전경 객체(foreground object)가 줄어드는 효과가 나타난다.

1.2 Dilation: 팽창

<https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm>

□ Dilation: 팽창

- ▣ 주어진 **structuring element**(혹은 **kernel**)에 해당하는 위치에 있는 화소 값 중에 가장 큰 값을 반환한다.
- ▣ 반환되는 값은 커널의 중심부의 위치에 대응하는 화소 값을 대치한다.
- ▣ 커널이 커지거나 반복 실행하면 팽창 효과도 크다.

$$\text{dst}(x, y) = \max_{(x', y') : \text{element}(x', y') \neq 0} \text{src}(x + x', y + y')$$

커널의 원소 값이 0이 아닌 위치에 있는 소스 영상의 화소 값 중 가장 큰 값을 반환한다.

□ 실행결과

- ▣ 계조 영상: 밝은 화소는 증가되고 어두운 화소는 감소된다.
- ▣ 이진 영상: 커널 안의 이진 화소 중 하나라도 1이면 1을 출력하므로 1로 구성된 객체가 늘어나는 효과가 나타난다.

2. Structuring Element(kernel)에 대하여..

<http://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm>

- The **structuring element** is a small binary image, i.e. a small matrix of pixels, each with a value of zero or one:
 - The matrix dimensions specify the *size* of the structuring element.
 - The pattern of ones and zeros specifies the *shape* of the structuring element.
 - An *origin* of the structuring element is usually one of its pixels, although generally the origin can be outside the structuring element.

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

Square 5x5 element

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

Diamond-shaped 5x5 element

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



Cross-shaped 5x5 element

■ ↔ Origin

2.1 getStructuringElement() 함수

- Returns a structuring element of the specified size and shape for morphological operations.
- `cv.getStructuringElement(shape, ksize[, anchor]) ->retval`

- Shape: 커널의 모양을 지정

- **MORPH_RECT** – a rectangular structuring element $E_{ij} = 1$ 
- **MORPH_ELLIPSE** – an elliptic structuring element, that is, a filled ellipse inscribed into the rectangle `Rect(0, 0, esize.width, esize.height)` 
- **MORPH_CROSS** – a cross-shaped structuring element


$$E_{ij} = \begin{cases} 1 & \text{if } i=\text{anchor.y or } j=\text{anchor.x} \\ 0 & \text{otherwise} \end{cases} \quad \text{+}$$

- **ksize** – Size of the structuring element. (**width, height**)
- **anchor** – Anchor position within the element.
 - The default value `(-1, -1)` means that the anchor is at the center.


2.2 함수에서 반환된 커널의 출력 사례

- `print(cv2.getStructuringElement(cv2.MORPH_RECT, (3,3)))`

```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]], dtype=uint8)
```


(3,3) 

```
array([[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]], dtype=uint8)
```


(5,5) 

- `print(cv2.getStructuringElement(cv2.MORPH_CROSS, (5,5)))`


```
[[0 0 1 0 0]
 [0 0 1 0 0]
 [1 1 1 1 1]
 [0 0 1 0 0]
 [0 0 1 0 0]]
```

(5,5) 

```
[[0 1 0]
 [0 1 0]
 [1 1 1]
 [0 1 0]
 [0 1 0]]
```

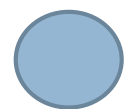
(3,5) 

```
[[0 1 0]
 [1 1 1]
 [0 1 0]]
```


(3,3) 

- `print(cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5,5)))`

```
[[0 0 1 0 0]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [0 0 1 0 0]]
```

(5,5) 

```
[[0 0 0 0 1 0 0 0 0]
 [0 1 1 1 1 1 1 1 0]
 [1 1 1 1 1 1 1 1 1]
 [0 1 1 1 1 1 1 1 0]
 [0 0 0 0 1 0 0 0 0]]
```

(9,5) 

3. 모폴로지 연산의 효과

https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html



erosion: 1^이 침식

Erosion: foreground object의 둘레에 대해 침식 연산을 행한다.
→ 고립된 잡음은 제거한다. 전경 객체가 가늘어 진다.

```
kernel = np.ones((5,5),np.uint8)  
erosion = cv.erode(img,kernel,iterations = 1)
```



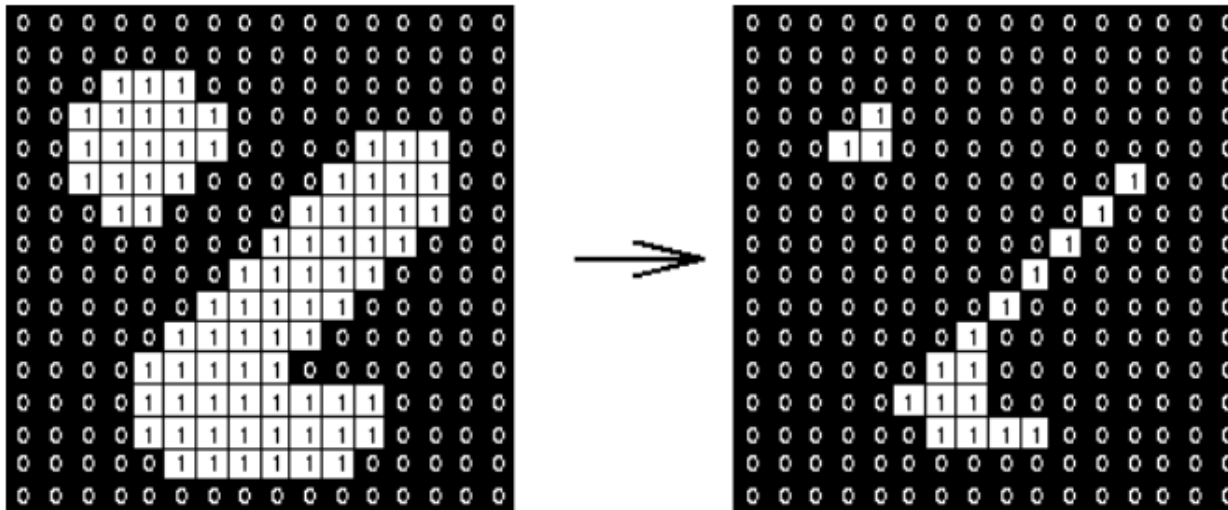
dilation: 1^이 팽창

Dilation: foreground object의 둘레에 대해 팽창 연산을 행한다.
→ 보통 침식하고 나면 전경 객체가 줄어드므로 침식 다음으로는 팽창 연산이 잇따른다.

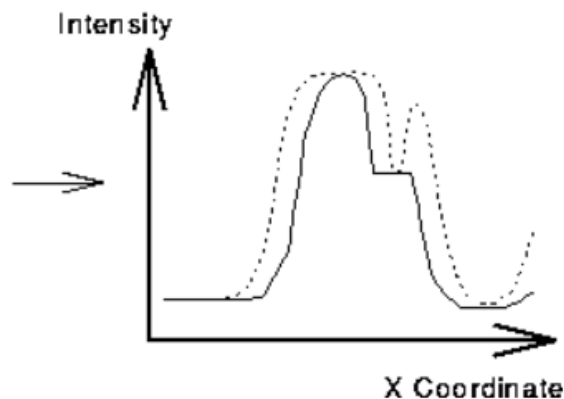
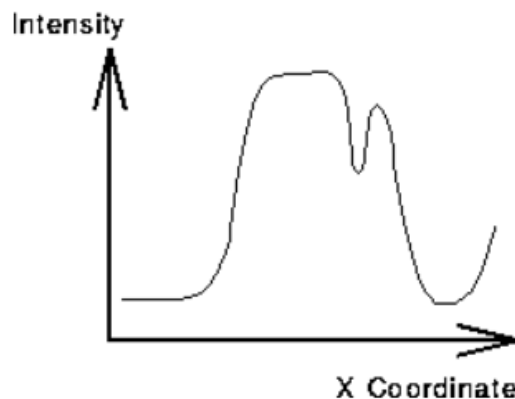
```
kernel = np.ones((5,5),np.uint8)  
dilation = cv.dilate(img,kernel,iterations = 1)
```

3.1 Effect of Erosion

<https://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm>

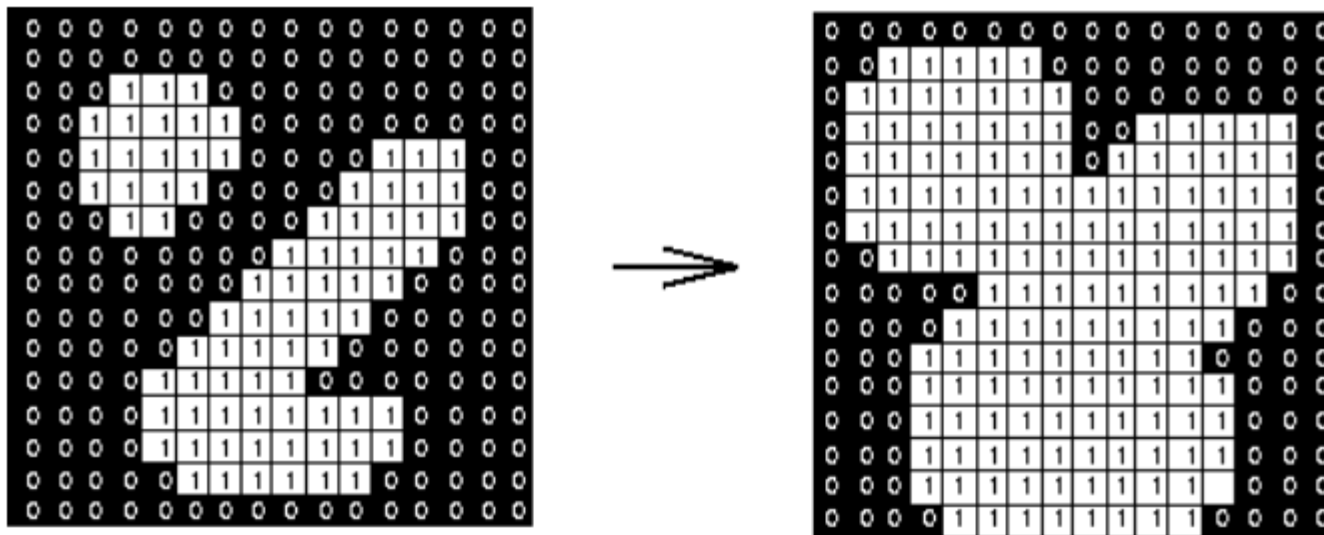


Width of eroding kernel: \longleftrightarrow

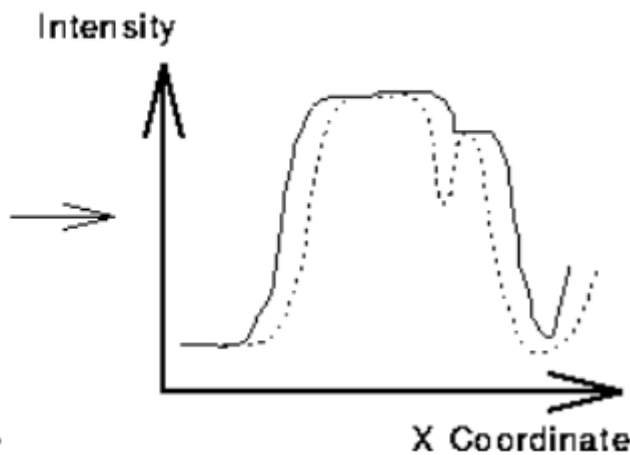
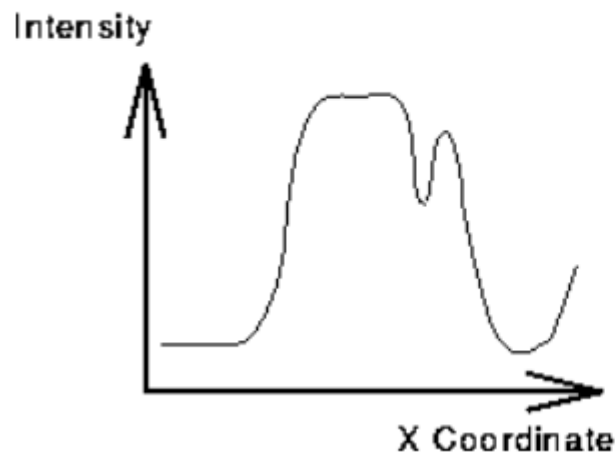


3.2 Effect of Dilation

<https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm>



Width of dilating kernel: \longleftrightarrow



4. 기본 모폴로지 함수- dilate & erode

- **Python : `dst=cv2.erode` ^{혹은} `dilate`(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]])**
 - ▣ src - 입력 영상. 다채널 영상의 경우 채널 별로 수행. Depth는 CV_8U, CV_16U, CV_16S, CV_32F or CV_64F 중의 하나여야 한다. * *bool type 지원안함.*
 - ▣ kernel - 침식 동작을 위한 구조(structuring element)
 - if element=Mat(), a 3 x 3 rectangular structuring element is used.
 - ▣ anchor - 커널(혹은 엘리먼트)안의 중심점의 위치
 - default value (-1, -1) means that the anchor is at the element center.
 - ▣ iterations -반복 회수. 반복회수와 커널의 크기는 상관 관계가 있다.
 - *cf.* (3,3)크기의 커널로 2회 반복한 것과 (5,5)크기의 커널로 1회 반복한 것과 (거의) 같다.
 - ▣ dst -반환값. src와 같은 size, 같은 type을 반환

5. 이진영상과 그레이 영상의 처리 효과 비교

팽창의 사례로 살펴본...

□ Morphological Dilation of a Binary Image

Structuring Element



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

Input Image

1	1
1	
0	
0	
0	

Output Image

□ Morphological Dilation of a Grayscale Image

Structuring Element



16	14	14	17	19	15	21
53	57	61	62	64	60	68
126	128	124	122	125	125	127
132	130	133	132	131	132	130
140	138	137	143	138	137	134
143	141	138	142	140	134	144
138	142	137	139	138	132	136

Input Image

16	16
57	
128	
132	
140	
143	
142	

Output Image

6. 모폴로지 연산의 변형

https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html



Opening: 침식 후 팽창 연산

침식을 먼저 시행하여 잡음이 줄어들면서 다시 팽창. 원래 크기 보존.
→ 잡음 제거에 유리

```
opening = cv.morphologyEx(img, cv.MORPH_OPEN, kernel)
```

```
opening = dilate(erode(src, element))
```



Closing: 팽창 후 침식 연산

팽창을 먼저 하기 때문에 구멍 혹은 끊긴 선이 연결되고 이후 침식 연산에서 다시 줄어든다. 원래 크기를 보존한다.

→ 전경 객체의 작은 구멍을 메우거나 끊긴 선을 연결하는데 유리

```
closing = cv.morphologyEx(img, cv.MORPH_CLOSE, kernel)
```

```
closing = erode(dilate(src, element))
```

7. morphologyEx() 함수

OpenClose, TOPHAT, BLACKHAT

- Performs advanced morphological transformations.
 - C++ : void **morphologyEx**(InputArray **src**, OutputArray **dst**, int **op**, InputArray **kernel**, Point **anchor**=Point(-1,-1), int **iterations**=1, int **borderType** = BORDER_CONSTANT, const Scalar& **borderValue** = morphologyDefaultBorderValue())
 - Python : **dst=cv2.morphologyEx**(src, op, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]])

- **MORPH_OPEN** – an opening operation

dst = open(**src**, **element**) = dilate(erode(**src**, **element**))

- **MORPH_CLOSE** – a closing operation

dst = close(**src**, **element**) = erode(dilate(**src**, **element**))

- **MORPH_GRADIENT** – a morphological gradient

dst = morph_grad(**src**, **element**) = dilate(**src**, **element**) – erode(**src**, **element**)

- **MORPH_TOPHAT** – “top hat”

dst = tophat(**src**, **element**) = **src** – open(**src**, **element**)

- **MORPH_BLACKHAT** – “black hat”

dst = blackhat(**src**, **element**) = close(**src**, **element**) – **src**