형태학 기반 영상 처리
(Morphology-based Image Processing)

담당교수: 김민기

Contents

- ❖형태학이란?
- ❖형태학 기본 연산
- ❖침식(erosion) / 팽창(dilation) 연산
- ❖열림(opening) / 닫힘(closing) 연산
- ❖형태학 연산을 이용한 경계 추출
- ❖실습: 형태학 연산의 응용

형태학이란?

❖형태학 연산

- 구조 요소(structuring element)를 이용한 형태 변환 연산

  (참고) 구조 요소 = 형태소(morphological element)

❖영상 처리에서 형태학

- 영상의 객체들의 형태(shape)를 분석하고 처리하는 기법
- 영상의 경계(boundary), 골격(skeleton) 등의 형태를 표현하는 요소 추출
- 영상 내의 잡음을 제거하거나 객체를 뚜렷하게 함

형태학 기본 연산

❖이동(translation)

: 화소의 집합을 A, 구조요소(형태소)의 집합을 $\omega = (u, v)$라고 하자.
A를 ω방향으로 이동한 결과를 A_ω라고 할 때, A_ω는 다음과 같다.

$$A_\omega = \{(a, b) + (u, v) : (a, b) \epsilon A\}$$

Ex) A $\omega = (2, 1)$ A_ω

	1			
1	1	1		
	1			

			1	
		1	1	1
			1	

형태학 기본 연산

❖이동(translation)

$$A_\omega = \{(a,b) + (u,v) : (a,b) \epsilon A\}$$

Ex) $\omega = \{(0,0),(1,0)\}$

A

		1		
	1			

A_ω

		1	1	
	1	1		

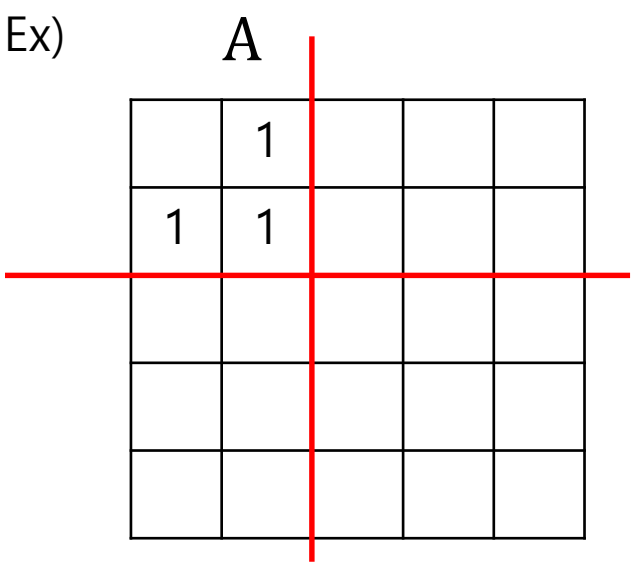
형태학 기본 연산

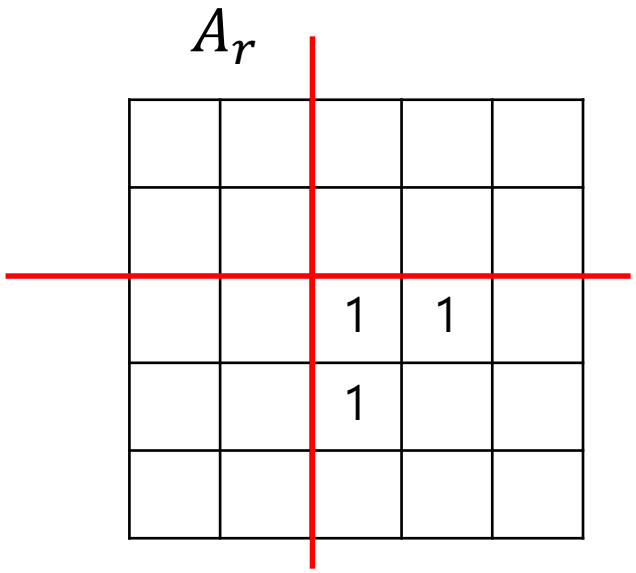
❖반사(reflection) = 대칭(symmetry)

: 화소의 집합을 A, 원점 대칭시킨 결과를 A_r이라고 할 때, A_r은 다음과 같이 표현할 수 있다.

$$A_r = \{(-a, -b): (a, b) \epsilon A\}$$

Ex)





침식 연산(Erosion Operation)

❖Erosion operation

$$A \ominus \omega = \{ B : B_\omega \subseteq A \}$$

Ex) $\omega = \{(0,0), (0,-1)\}$

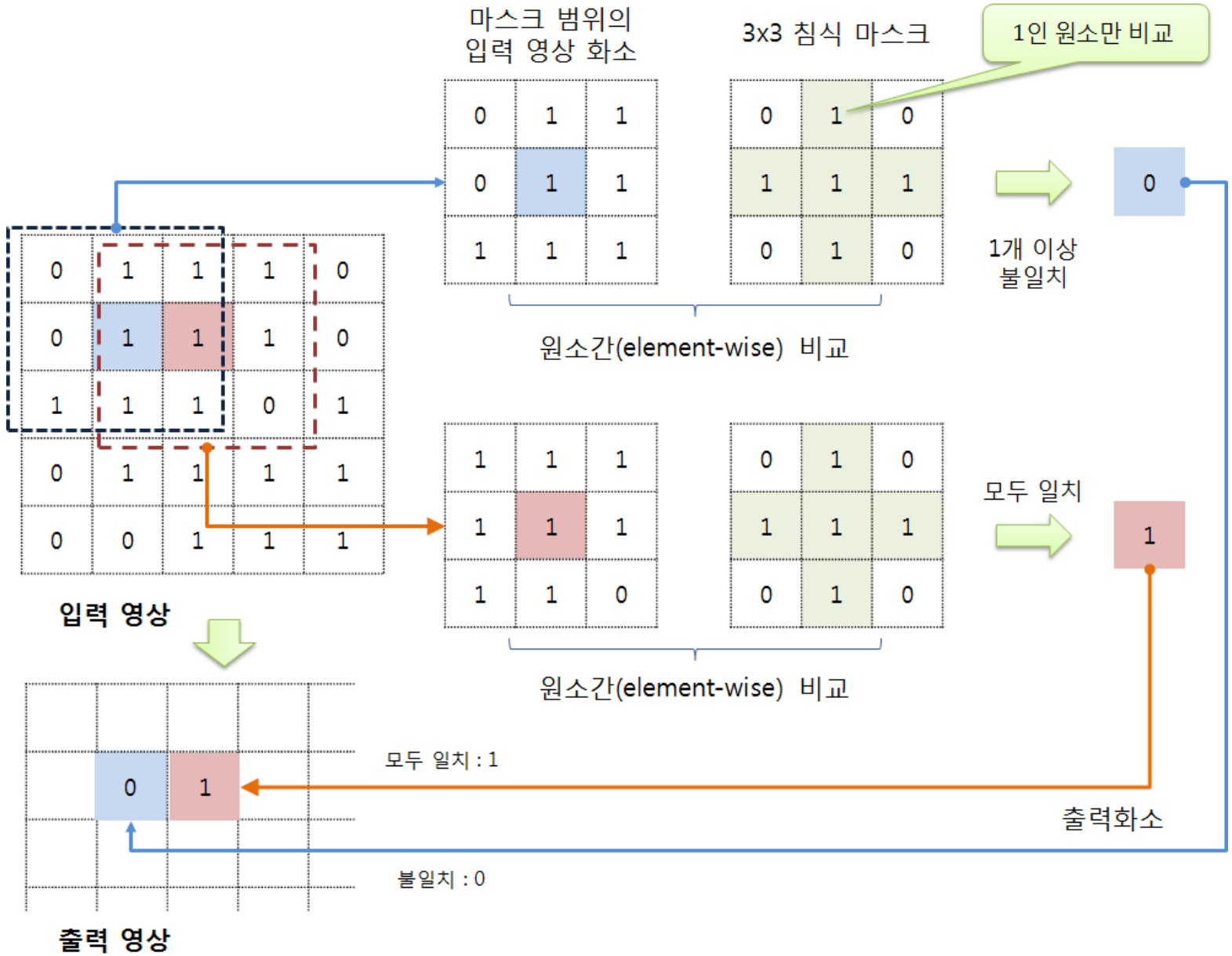
A

	1	1	1	
		1	1	
			1	

$A \ominus \omega$

		1	1	
			1	

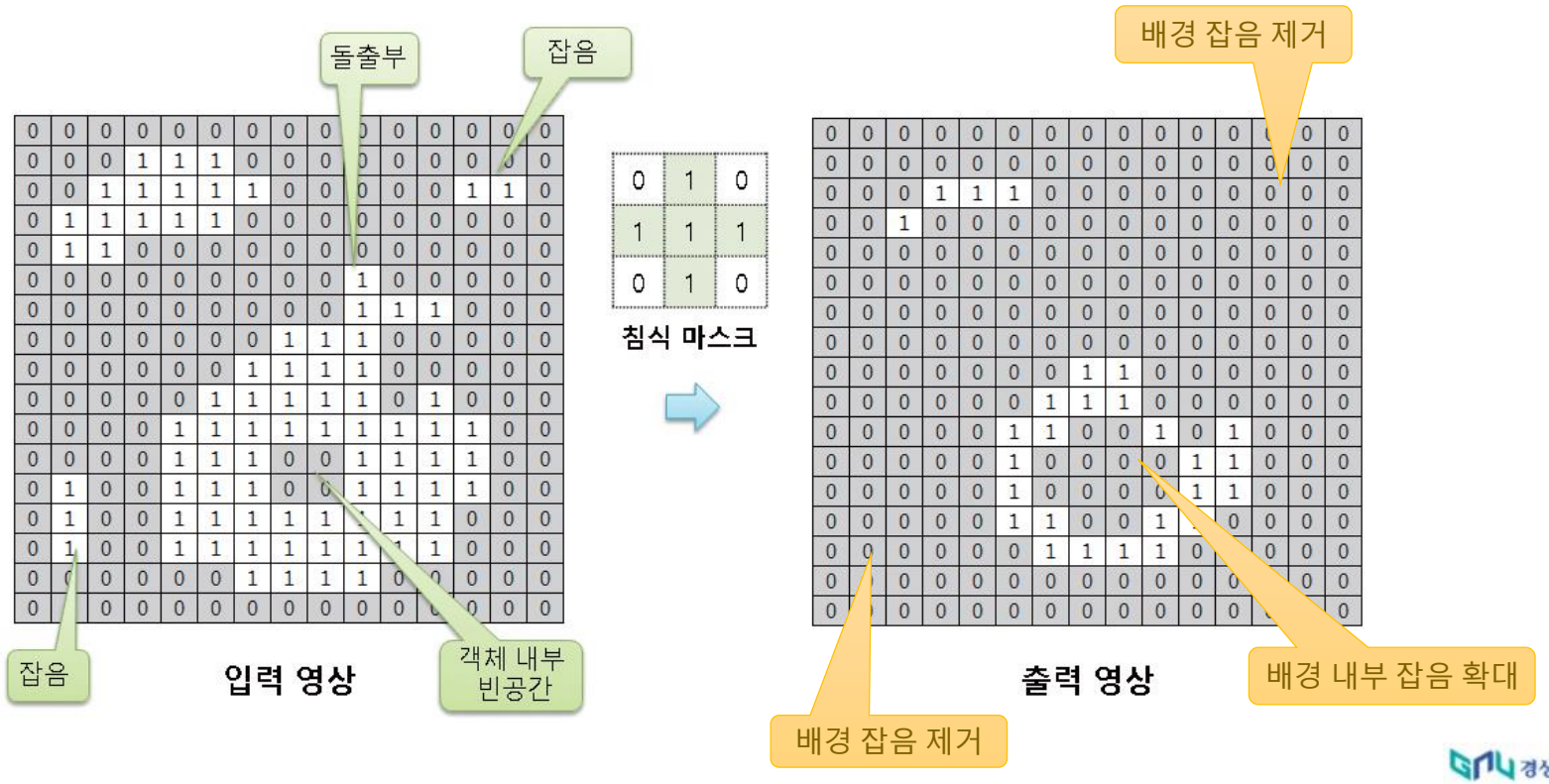
* 침식 연산은 교환법칙이 성립하지 않음 $A \ominus B \neq B \ominus A$


마스크 범위의
입력 영상 화소
3x3 침식 마스크
1인 원소만 비교
0 1 1
0 1 1
1 1 1
0 1 0
1 1 1
0 1 0
1개 이상
불일치
0
원소간(element-wise) 비교
0 1 1 1 0
0 1 1 1 0
1 1 1 0 1
0 1 1 1 1
0 0 1 1 1
1 1 1
1 1 1
1 1 0
0 1 0
1 1 1
0 1 0
모두 일치
1
원소간(element-wise) 비교
입력 영상
모두 일치 : 1
출력화소
0 1
불일치 : 0
출력 영상

침식 연산(Erosion Operation)

❖객체를 침식 시키는 연산

- 객체의 크기 축소, 배경 확장
- 영상 내에 존재하는 잡음 같은 작은 크기의 객체 제거 가능
- 소금-후추 잡음과 같은 임펄스 잡음 제거

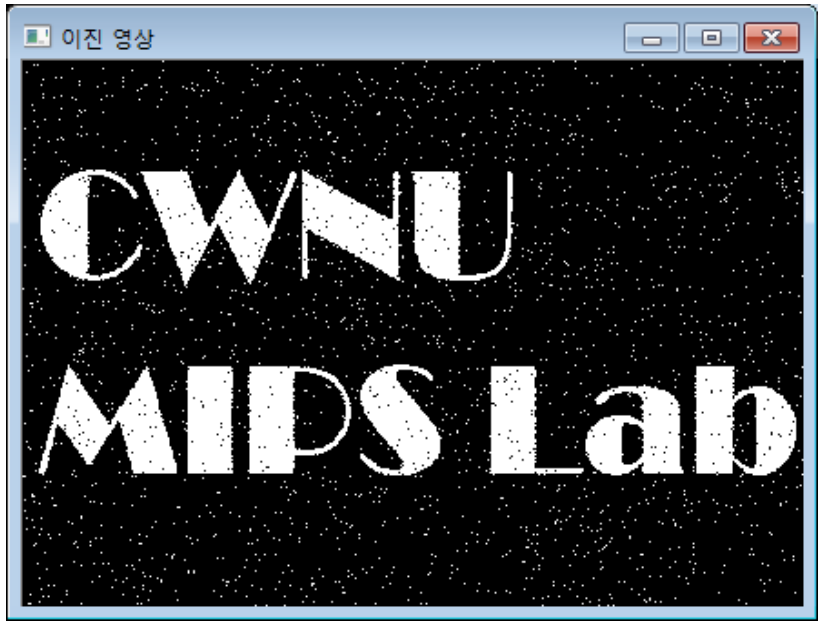


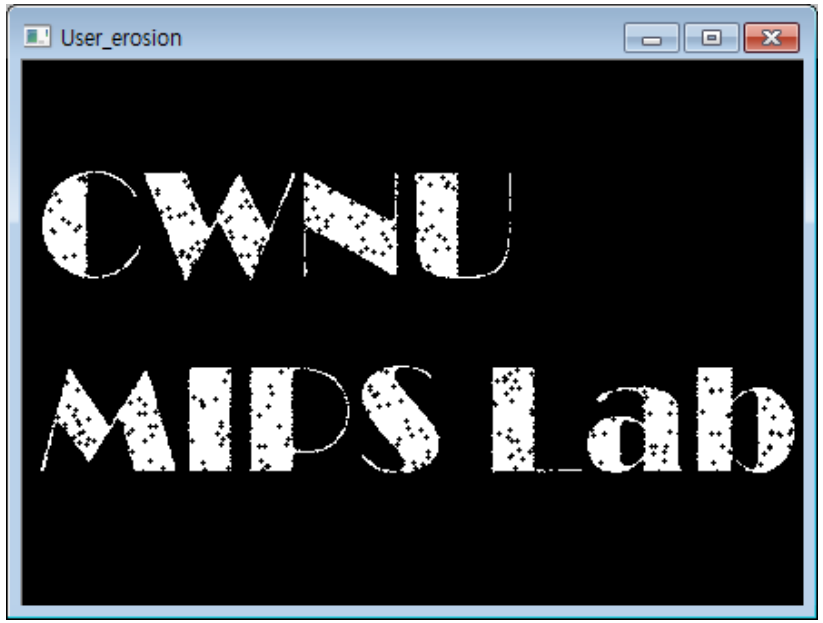
예제 7.4.1 모폴로지 침식 연산 – 14.erode.py

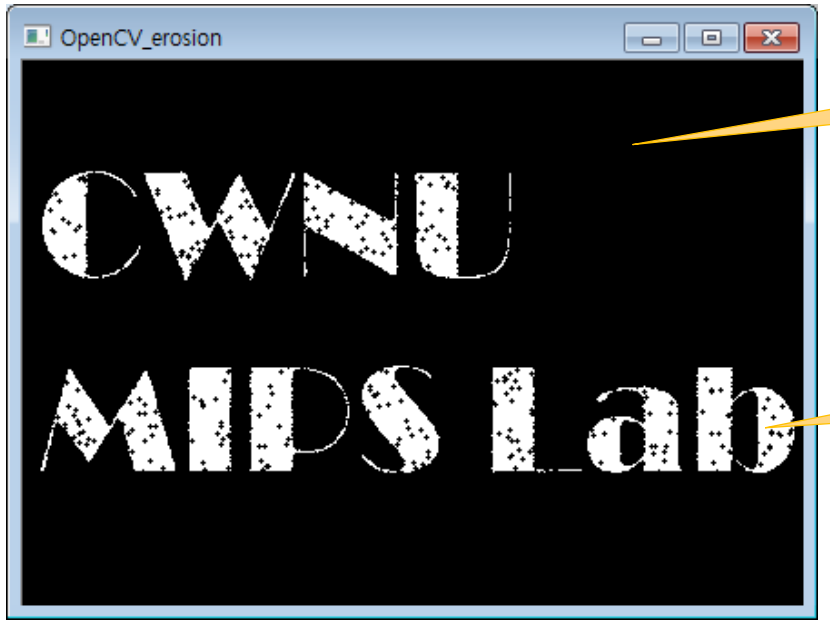
```
01  import numpy as np, cv2
02
03  def erode(img, mask=None):                              # 침식 연산 함수
04      dst = np.zeros(img.shape, np.uint8)
05      if mask is None: mask = np.ones((3, 3), np.uint8)
06      ycenter, xcenter = np.divmod(mask.shape[:2], 2)[0]  # 마스크 중심 좌표
07
08      mcnt = cv2.countNonZero(mask)                       # 마스크 1인 원소 개수
09      for i in range(ycenter, img.shape[0] - ycenter):    # 입력 행렬 반복 순회
10          for j in range(xcenter, img.shape[1] - xcenter):
11              y1, y2 = i - ycenter, i + ycenter + 1       # 마스크 높이 범위
12              x1, x2 = j - xcenter, j + xcenter + 1       # 마스크 너비 범위
13              roi = img[y1:y2, x1:x2]                     # 마스크 영역
14              temp = cv2.bitwise_and(roi, mask)           # 논리곱으로 일치 원소 지정
15              cnt = cv2.countNonZero(temp)                # 일치 원소 개수 계산
16              dst[i, j] = 255 if (cnt == mcnt) else 0     # 출력 화소에 저장
17      return dst
18
19  image = cv2.imread("images/morph.jpg", cv2.IMREAD_GRAYSCALE)
20  if image is None: raise Exception("영상파일 읽기 오류")
```

```python
image = cv2.imread("images/morph.jpg", cv2.IMREAD_GRAYSCALE)
if image is None: raise Exception("영상파일 읽기 오류")

data = [  0, 1, 0,                                          # 마스크 원소 지정
          1, 1, 1,
          0, 1, 0]
mask = np.array(data, np.uint8).reshape(3, 3)               # 마스크 행렬 생성
th_img = cv2.threshold(image, 128, 255, cv2.THRESH_BINARY)[1]   # 영상 이진화

dst1 = erode(th_img, mask)                                  # 사용자 정의 침식 함수
dst2 = cv2.erode(th_img, mask)                              # OpenCV의 침식 함수
# dst2 = cv2.morphologyEx(th_img, cv2.MORPH_ERODE, mask)    # OpenCV의 침식 함수2

cv2.imshow("image", image)
cv2.imshow("binary image", th_img)
cv2.imshow("User erode", dst1)
cv2.imshow("OpenCV erode", dst2)
cv2.waitKey(0)
```

입력영상







배경 잡음 제거

객체내부 잡음 확대

침식 연산(Erosion Operation)

침식 연산(Erosion Operation)

❖OpenCV 침식 연산

```
erode (InputArray src,
       OutputArray dst,
       InputArray  kernel,     // 구조요소(=형태소) 커널
       Point  anchor=Point(-1, -1),
       int   iterations=1,
       int   borderType=BORDER_CONSTANT,
       const  Scalar&  borderValue=morphologyDefaultBorderValue()
)
```

팽창 연산 (Dilation Operation)

❖팽창 연산(dilation)

$$A \oplus \omega = \bigcup A_\omega$$

Ex) $\omega = \{(0,0),(0,-1)\}$

A

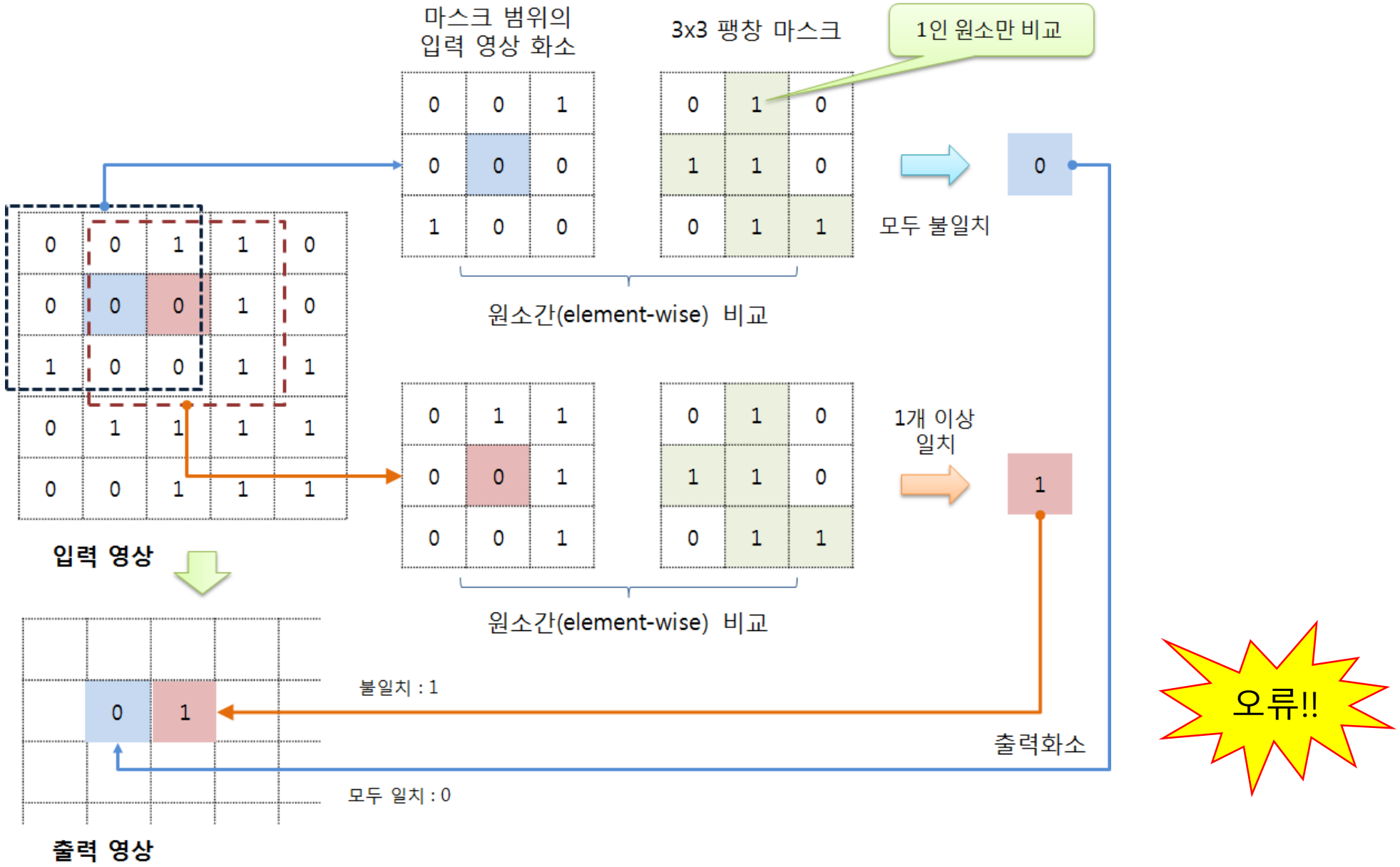
	1	1	1	
		1	1	
			1	

$A \oplus \omega$

	1	1	1	
	1	1	1	
		1	1	
			1	

* 팽창 연산은 교환법칙과 결합법칙 모두 성립

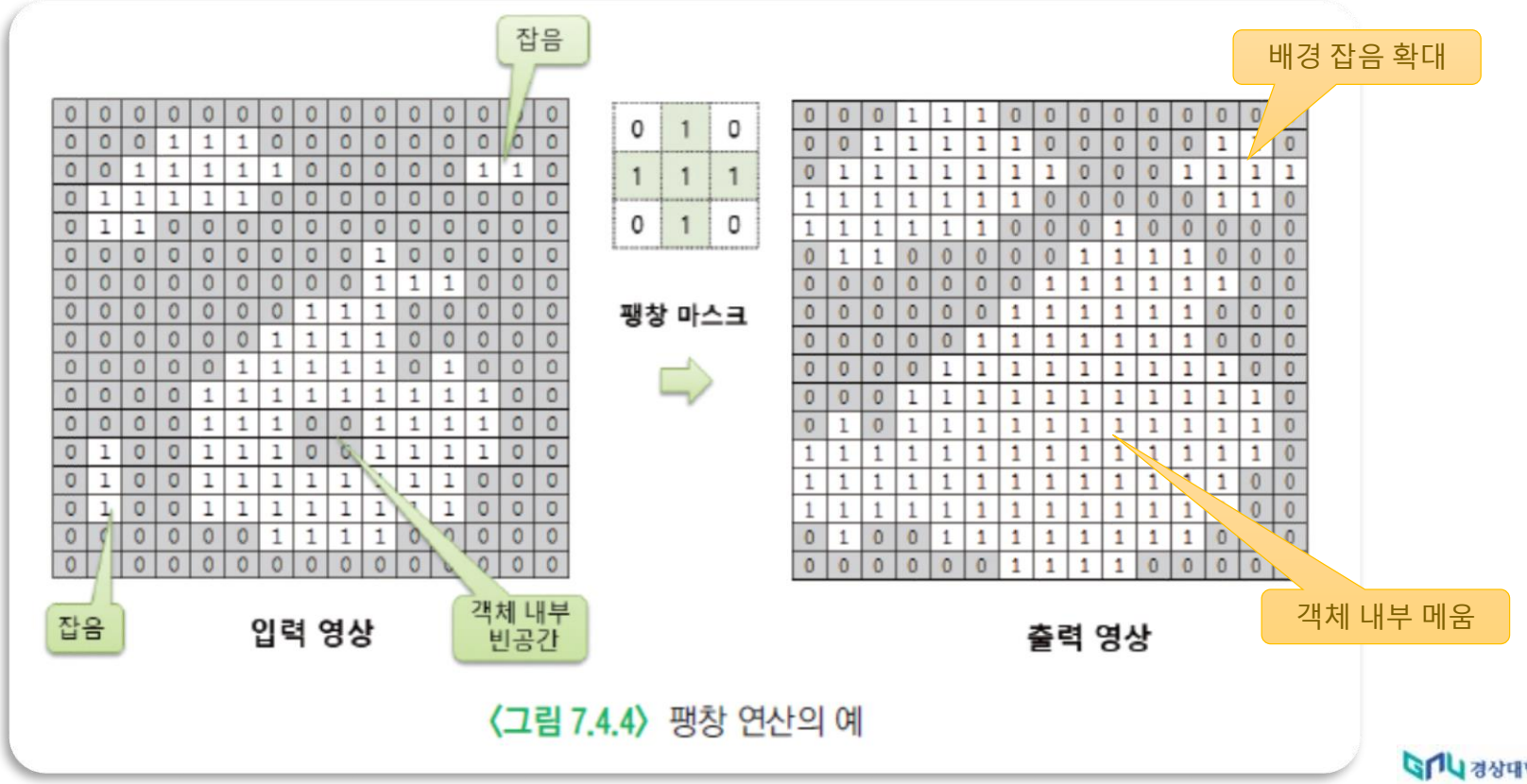
$A \oplus B = B \oplus A$ $A \oplus (B \oplus C) = (A \oplus B) \oplus C$


마스크 범위의
입력 영상 화소
3x3 팽창 마스크
1인 원소만 비교
모두 불일치
원소간(element-wise) 비교
1개 이상
일치
원소간(element-wise) 비교
입력 영상
불일치 : 1
출력화소
모두 일치 : 0
출력 영상
오류!!

팽창 연산 (Dilation Operation)

❖객체를 팽창시키는 연산

- 객체의 최외곽 화소를 확장 → 객체크기 확대, 배경축소
- 객체 팽창으로 객체 내부의 빈 공간도 메워짐 → 객체 내부 잡음 제거



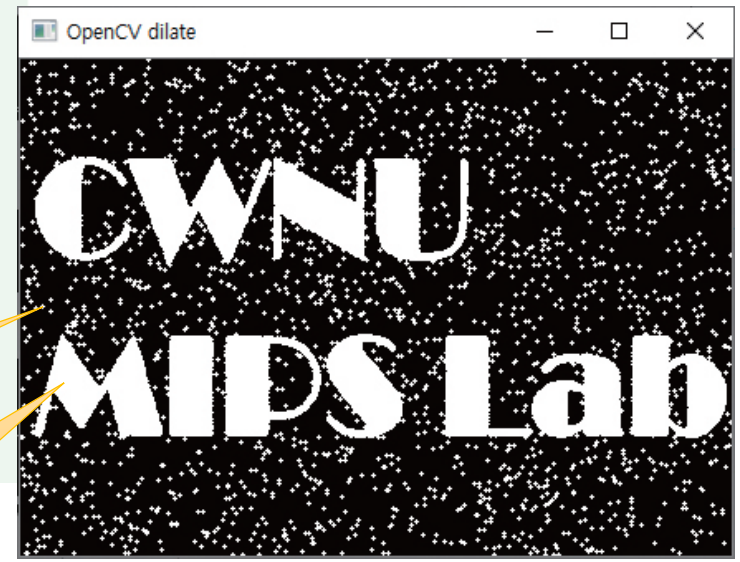
〈그림 7.4.4〉 팽창 연산의 예

팽창 연산 (Dilation Operation)

❖OpenCV 팽창 연산

```
dilate (InputArray src,
        OutputArray dst,
        InputArray  kernel,     // 구조요소(=형태소) 커널
        Point  anchor=Point(-1, -1),
        int   iterations=1,
        int   borderType=BORDER_CONSTANT,
        const  Scalar&  borderValue=morphologyDefaultBorderValue()
)
```

```python
image = cv2.imread("images/morph.jpg", cv2.IMREAD_GRAYSCALE)
if image is None: raise Exception("영상파일 읽기 오류")

mask = np.array([[0, 1, 0],                                    # 마스크 계수 초기화
                 [1, 1, 1],
                 [0, 1, 0]]).astype('uint8')
th_img = cv2.threshold(image, 128, 255, cv2.THRESH_BINARY)[1]  # 영상 이진화
dst1 = dilate(th_img, mask)                                    # 사용자 정의 팽창 함수
dst2 = cv2.dilate(th_img, mask)                                # OpenCV의 팽창 함수
# dst2 = cv2.morphologyEx(th_img, cv2.MORPH_DILATE, mask)

cv2.imshow("User dilate", dst1)
cv2.imshow("OpenCV dilate", dst2)
cv2.waitKey(0)
```



배경 잡음 증가

객체 내부 잡음 제거

침식팽창 연산을 명도 영상에 적용하면?

- 침식 연산을 명도 영상에 적용하는 것은 min 필터링과 같다.

$$dst(x,y) = \min(src(x+x', y+y')),$$

where (x′, y′)는 kernel에서 0이 아닌 위치의 이웃

- 팽창 연산을 명도 영상에 적용하는 것은 max 필터링과 같다.

$$dst(x,y) = max(src(x+x', y+y')),$$

where (x′, y′)는 kernel에서 0이 아닌 위치의 이웃

열림 연산 (Opening Operation)

❖열림연산 = 침식연산 후 팽창연산

- 침식 연산으로 인해서 객체는 축소되고, 배경 부분의 미세한 잡음 제거
- 팽창 연산으로 인해서 축소되었던 객체들이 다시 원래 크기로

열림 연산 (Opening Operation)

❖열림 연산(opening)

$$A \circ w = (A \ominus w) \oplus w$$

- 볼록하게 돌출된 부분을 제거하고, 좁은 연결을 끊음

- $A \supset (A \circ w)$
- $A \subset C \Rightarrow (A \circ w) \subset (C \circ w)$
- $(A \circ w) \circ w = A \circ w$ // 멱등 법칙(idempotence)

닫힘 연산 (Closing Operation)

❖닫힘 연산 = 팽창연산 후 침식연산

- 팽창 연산으로 객체가 확장되어서 객체 내부의 빈 공간이 메워짐
- 침식 연산으로 다음으로 확장되었던 객체의 크기가 원래대로 축소

닫힘 연산 (Closing Operation)

❖닫힘 연산(closing)

$$A \bullet w = (A \oplus w) \ominus w$$

- 오목하게 들어간 부분이나 작은 구멍을 채움

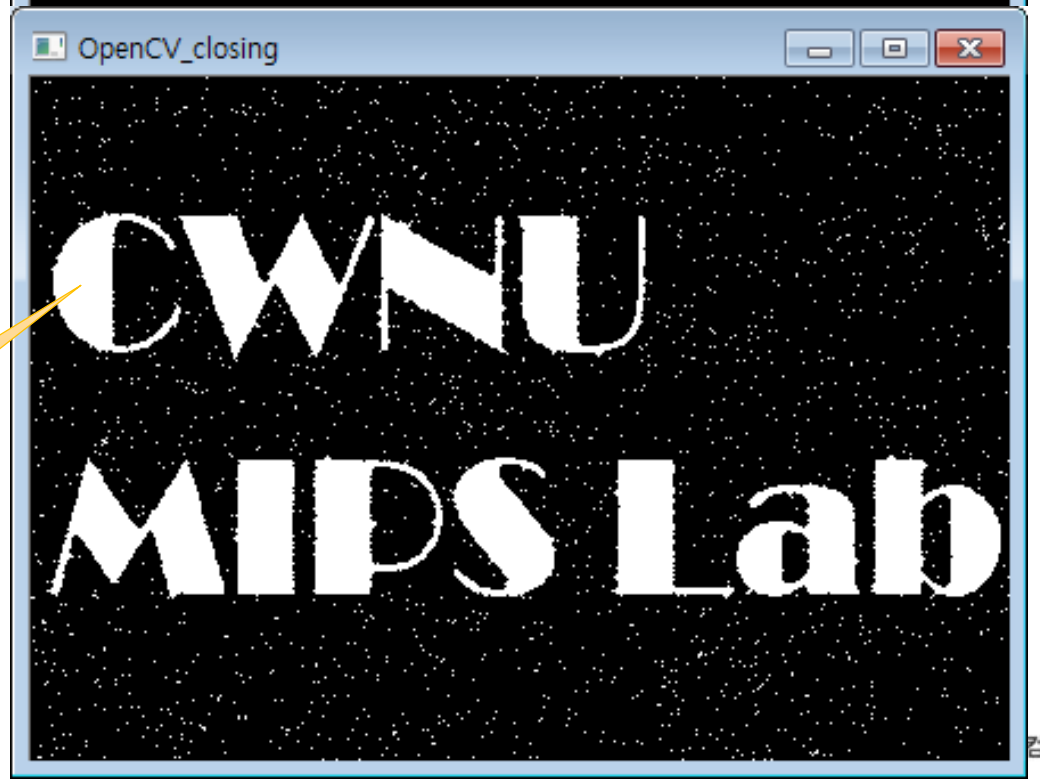
- $A \subset (A \bullet w)$
- $A \subset C \Rightarrow (A \bullet w) \subset (C \bullet w)$
- $(A \bullet w) \bullet w = A \bullet w$ // 멱등 법칙(idempotence)

예제 7.4.3 모폴로지 닫힘 & 열림 연산 – 16.close_open.py

```
01  import numpy as np, cv2
02  from Common.filters import erode, dilate              # filters 모듈의 저자 구현 함수 임포트
03
04  def opening(img, mask):                               # 열림 연산 함수
05      tmp = erode(img, mask)                            # 침식
06      dst = dilate(tmp, mask)                           # 팽창
07      return dst
08
09  def closing(img, mask):                               # 닫힘 연산 함수
10      tmp = dilate(img, mask)                           # 팽창
11      dst = erode(tmp, mask)                            # 침식
12      return dst
13
14  image = cv2.imread("images/morph.jpg", cv2.IMREAD_GRAYSCALE)
15  if image is None: raise Exception("영상파일 읽기 오류")
16
```

```
14  image = cv2.imread("images/morph.jpg", cv2.IMREAD_GRAYSCALE)
15  if image is None: raise Exception("영상파일 읽기 오류")
16
17  mask = np.array([[0, 1, 0],                          # 마스크 생성 및 초기화
18                   [1, 1, 1],
19                   [0, 1, 0]]).astype('uint8')
20  th_img = cv2.threshold(image, 128, 255, cv2.THRESH_BINARY)[1]     # 영상 이진화
21
22  dst1 = opening(th_img, mask)                                      # 저자 구현 열림 함수
23  dst2 = closing(th_img, mask)                                      # 저자 구현 닫힘 함수
24  dst3 = cv2.morphologyEx(th_img, cv2.MORPH_OPEN, mask)             # OpenCV의 열림 함수
25  dst4 = cv2.morphologyEx(th_img, cv2.MORPH_CLOSE, mask, iterations=1)      # 닫힘 함수
26
27  cv2.imshow("User_opening", dst1);     cv2.imshow("User_closing", dst2)
28  cv2.imshow("OpenCV_opening", dst3);   cv2.imshow("OpenCV_closing", dst4)
29  cv2.waitKey(0)
```

배경 잡음 제거



객체 내부 잡음 제거

형태학 연산을 이용한 경계 추출

❖A의 내부 경계 $A - (A \ominus w)$

❖A의 외부 경계 $(A \oplus w) - A$

❖A의 형태학적 기울기(Gradient)

= A의 내부 경계 + A의 외부 경계

$$(A \oplus w) - (A \ominus w)$$

OpenCV 형태학 연산

❖형태학 연산

```
morphologyEx (InputArray src,
        OutputArray dst,
         int   op1),      // 형태학 연산자
         InputArray  kernel,     // 구조요소(=형태소) 커널
         Anchor=Point(-1, -1),
         int   iteration=1,
         int   borderType=BORDER_CONSTANT,
         const  Scalar&  borderValue=morphologyDefaultBorderValue()
)
```

1) op = MORPH_OPEN | MORPH_CLOSE | MORPH_GRADIENT
| TOPHAT | BLACKHAT

OpenCV 형태학 연산

- OPEN: $\mathrm{dst} = (src \ominus kernel) \oplus kernel$
- CLOSE: $\mathrm{dst} = (src \oplus kernel) \ominus kernel$
- GRADIENT: $\mathrm{dst} = (src \oplus kernel) - (src \ominus kernel)$
- TOPHAT: $\mathrm{dst} = \mathrm{src} - (src \circ kernel)$
- BLACKHAT: $\mathrm{dst} = (src \bullet kernel) - src$

https://vovkos.github.io/doxyrest-showcase/opencv/sphinx_rtd_theme/page_tutorial_py_morphological_ops.html?highlight=morphologyex

OpenCV 형태학 연산

❖구조요소(=형태소) 정의

```
getStructuringElement (
        int  shape1),          // 구조요소(=형태소)의 모양
        Size   ksize,          // 구조요소(=형태소) 커널의 크기
        Point  anchor(-1, -1)
)
```

1) Shape = MORPH_RECT | MORPH_ELLIPSE | MORPH_CROSS

형태학 응용: 자동차 번호판 검출

번호판 위치 검출
(ROI detection)

↓

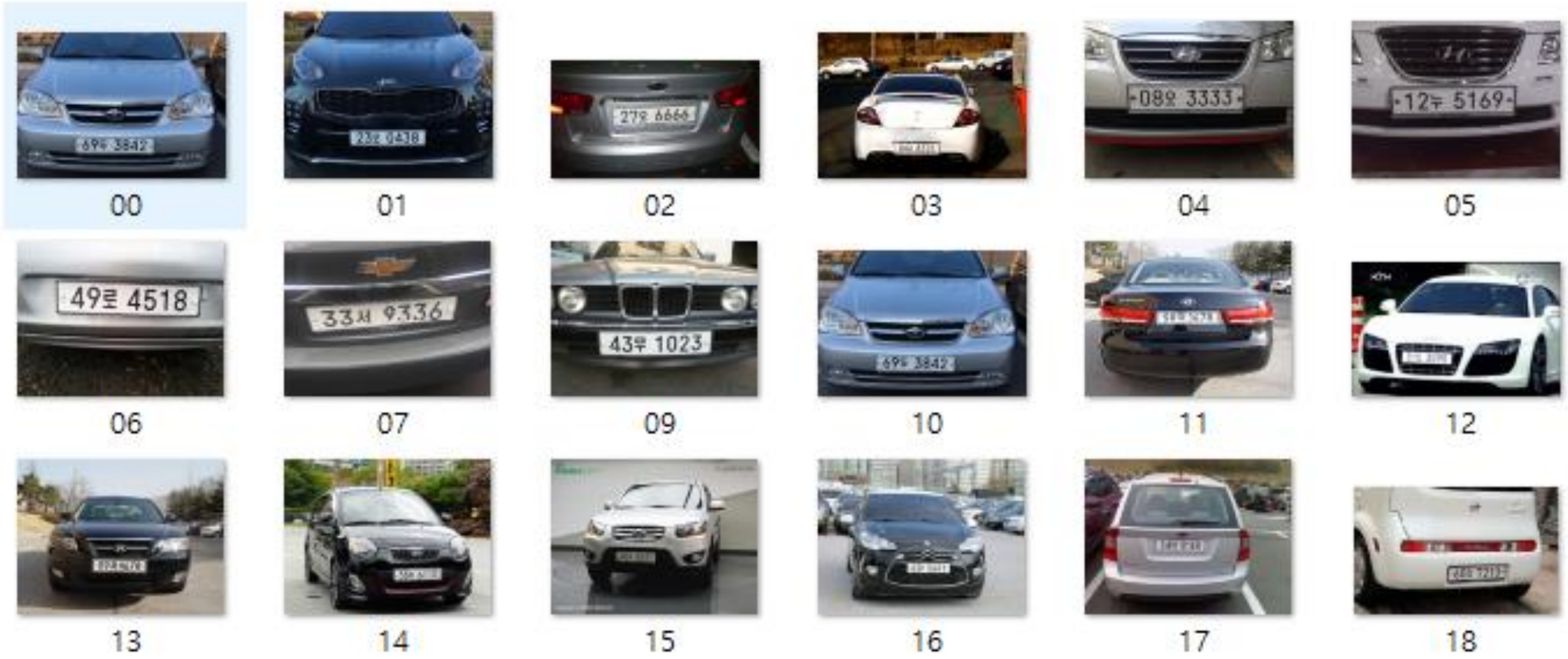
문자 분할
(Segmentation)

↓

문자 인식
(Recognition)

형태학을 이용한 자동차 번호판 검출

❖[심화예제 7.4.4] 번호판 후보 객체 검출



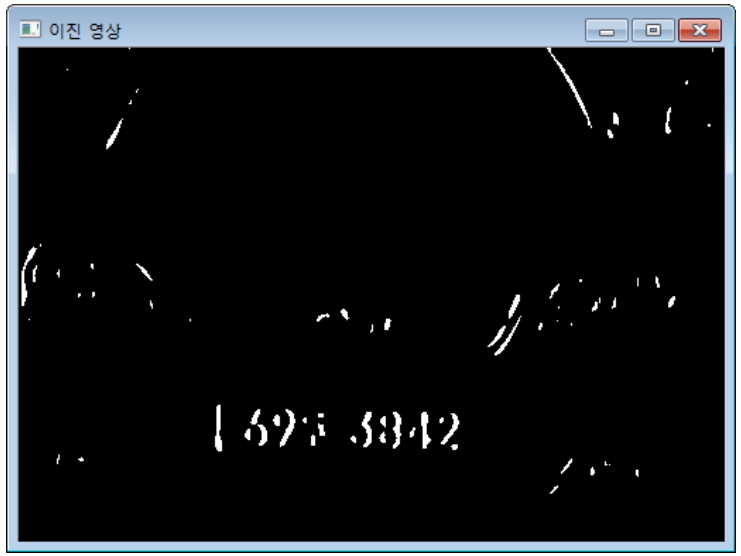
심화예제 7.4.4 번호판 후보 객체 검출 – 17.detect_plate.py

```
01  import numpy as np, cv2
02
03  while True:
04      no = int(input("차량 영상 번호( 0:종료) : "))                # 차량 번호 입력
05      if no == 0: break
06
07      fname = "images/test_car/{0:02d}.jpg".format(no)            # 영상파일 이름 구성
08      image = cv2.imread(fname, cv2.IMREAD_COLOR)
09      if image is None:
10          print(str(no) + "번 영상파일이 없습니다.")
11          continue
12
13      mask = np.ones((5, 17), np.uint8)                           # 닫힘 연산 마스크
14      gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)              # 명암도 영상 변환
15      gray = cv2.blur(gray, (5, 5))                               # 블러링
16      gray = cv2.Sobel(gray, cv2.CV_8U, 1, 0, 5)                  # 소벨 에지 검출
17                                          수직에지 검출
18      ## 이진화 및 닫힘 연산 수행
19      th_img = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)[1]
20      morph = cv2.morphologyEx(th_img, cv2.MORPH_CLOSE, mask, iterations=3)
21
22      cv2.imshow("image", image)
23      cv2.imshow("binary image", th_img)
24      cv2.imshow("opening", morph)
25      cv2.waitKey()
```

형태학을 이용한 자동차 번호판 검출

❖[심화예제 7.4.4]

```
Run:  19.detect_plate
C:\Python\python.exe
  D:/source/chap07/19.detect_plate.py
차량 영상 번호( 0:종료 ) : 5
차량 영상 번호( 0:종료 ) : 2
차량 영상 번호( 0:종료 ) : 1
```



번호판 문자 분할: Projection

❖ 수평/수직 투영 (horizontal & vertical projection)

번호판 문자 인식: Template Matching

❖템플릿 정합

```
matchTemplate (
        InputArray image,
        InputArray templ,
        OutputArray  result,      // 템플릿 정합 결과
        int   method1)
)
```

image

	1	
1		1
1		1
	1	

templ

	1	
	1	
	1	
	1	

1) method = TM_SQRDIFF | TM_CCOEFF | TM_CCORR
| TM_SQRDIFF_NORMED | TM_CCOEFF_NORMED | TM_CCORR_NORMED

Match template : method options

$$R_{SQDIFF}(x,y) = \sum_{s=0}^{w-1}\sum_{t=0}^{h-1}(T(s,t) - I(x+s,y+t))^2$$

$$R_{SQDIFF\_NORMED}(x,y) = R_{SQDIFF}(x,y)/D(x,y)$$

$$D(x,y) = \sqrt{\sum_{s=0}^{w-1}\sum_{t=0}^{h-1}T(s,t)^2 \times \sum_{s=0}^{w-1}\sum_{t=0}^{h-1}I(x+s,y+t)^2}$$

$$R_{CCORR}(x,y) = \sum_{s=0}^{w-1}\sum_{t=0}^{h-1}(T(s,t) \times I(x+s,y+t))$$

$$R_{CCORR\_NORMED}(x,y) = R_{CCORR}(x,y)/D(x,y)$$

Summary

❖형태학 연산

- 침식 연산: 객체 외부에 존재하는 작은 크기의 잡음을 제거
- 팽창 연산: 객체 내부의 빈 공간을 메우는 역할
- 열림 연산: 침식 연산 수행 후 팽창 연산을 수행
- 닫힘 연산: 팽창 연산 수행 후 침식 연산을 수행

❖형태학을 이용한 경계 추출

- 내부 경계: $A - (A \ominus w)$
- 외부 경계: $(A \oplus w) - A$
- Gradient: $(A \oplus w) - (A \ominus w)$