11장 형태학 처리

- 형태학의 개념
- 형태학을 위한 기초 이론
- 이진 영상에서의 형태학 처리
- 그레이 영상에서의 형태학처리

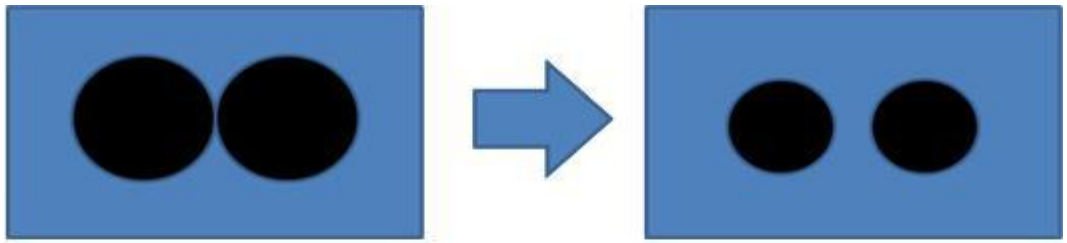
한빛미디어(주)

학습목표

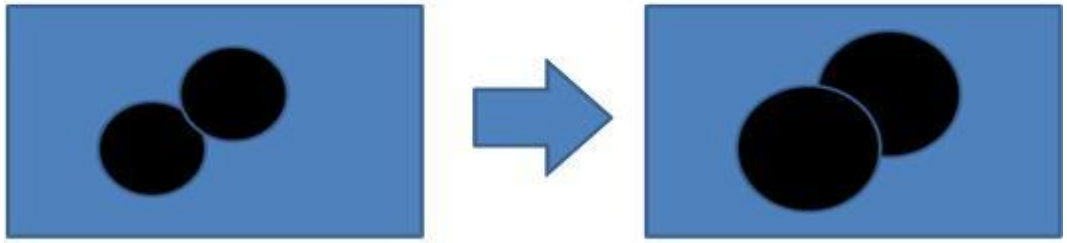
- ✓ 형태학의 개념을 이해한다.
- ✓ 이진 영상에서 침식과 팽창연산을 공부한다.
- ✓ 이진 영상에서 열림과 닫힘을 공부한다.
- ✓ 골격화를 소개한다.
- ✓ 그레이 영상에서의 형태학 처리로 침식, 팽창, 열림, 닫힘을 학습한다.

Section 01 형태학의 개념

- 형태학(Morphology, 모폴로지)
  - 영상의 형태(Shape)를 분석하고 처리하는 기법
  - 영상의 경계, 블록, 골격 등 형태를 표현하거나 서술하는 데 필요한 영상 요소를 추출하는데 형태학 처리 활용
  - 영상의 경계 너비가 일정치 않거나 중간에 단절되어 이를 일정하게 하거나 연결할 때 형태학 처리가 필요함.



(a) 경계선의 단순화



(b) 경계선의 확장

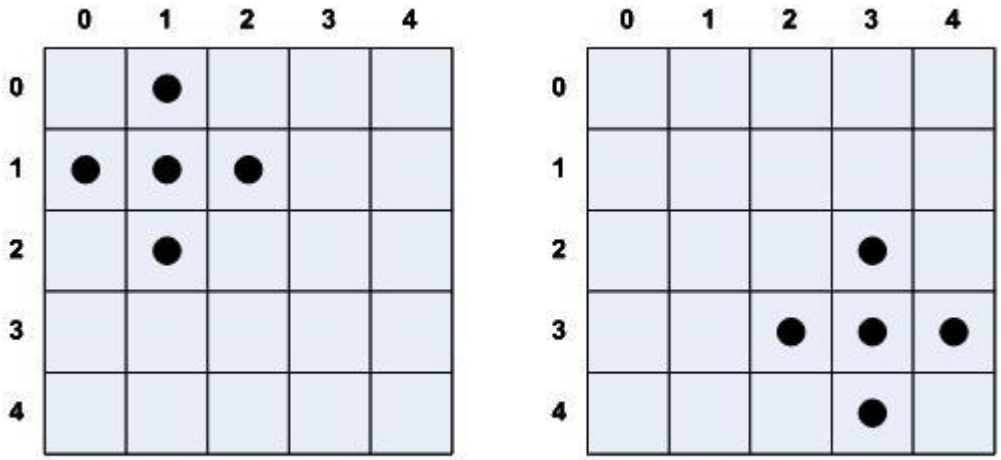
[그림 11-1] 형태학 처리한 예

이동

- **A: 영상 내의 화소 집합, 특정 좌표의 점: ω=(u, v)**
- **Aω: 화소의 집합 A를 (u, v) 방향으로 이동한 결과**

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

- **(a, b)는 A 집합의 화소로, 각 화소는 (u, v)로 이동하게 됨.**



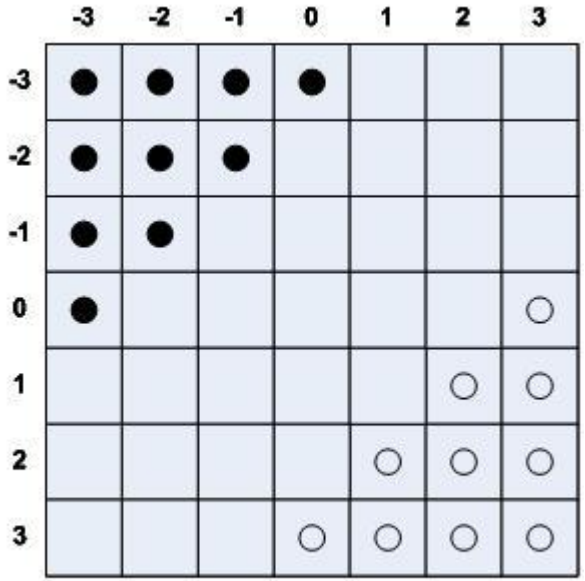
[그림 11-2] 화소 집합의 이동

형태학을 위한 기초 이론(계속)-반사

- 반사
  - 대칭에는 좌우 대칭과 상하 대칭이 있음.
  - 형태학에서 반사는 대칭의 개념과 같음.

$$\hat{A} = \{(-a, -b) : (a, b) \in A\}$$

- A의 화소 좌표 (a, b)는 원점을 대칭하여 반사가 일어나므로 좌표는 (-a, -b)가 됨.



[그림 11-3] 화소 집합의 반사

Section 03 이진 영상에서의 형태학 처리

- 이진 영상에서의 형태학 처리
  - 이진 영상에서 특정 패턴을 찾으려고 이진 영상의 밝기 값에 형태소라는 행렬과 논리적인(AND, OR, NOT, …) 연산을 수행하여 출력 화소를 결정하는 것

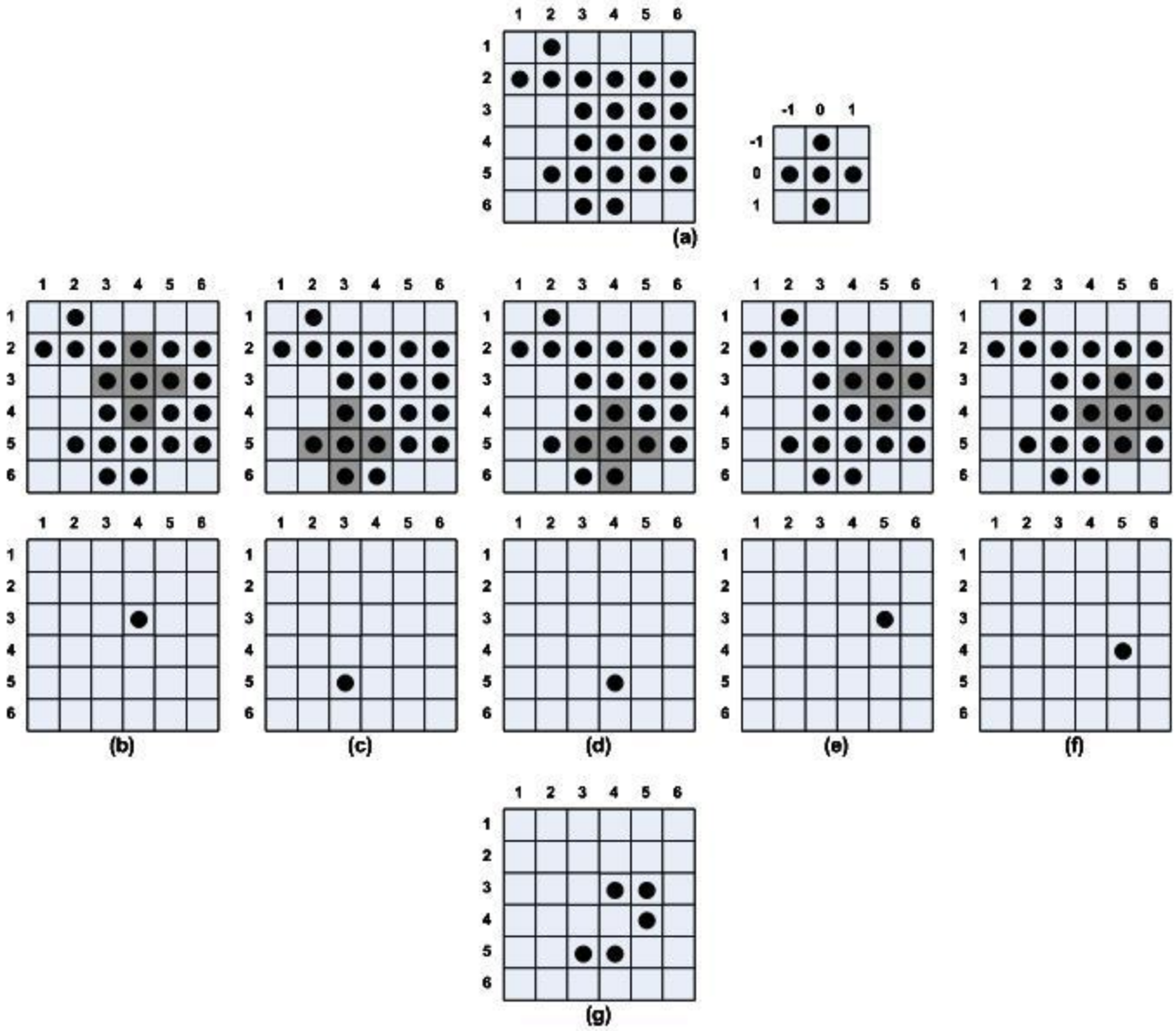
- 침식(Dilation)과 팽창(Erosion)
  - 형태학의 기본이 되는 연산으로, 다른 모든 형태학 연산은 이 두 연산을 조합하여 만듦.

침식

- 물체의 크기를 그 배경과 관련하여 일정하게 줄여주는 것
- 물체의 크기는 줄어들고, 배경은 확대됨.
- 영상의 물체와 배경 사이에 스파이크 잡음이 있을 때 이 잡음을 제거하거나 전체 영상에서 아주 작은 물체를 제거하는 데 응용됨.
- 영상에서의 돌출부는 감소시키고, 내부 돌출부는 증가시켜서 서로 닿은 물체를 분리할 때도 유용함.

$$A-B=\{\omega : B\omega \subseteq A\}$$

침식(계속)



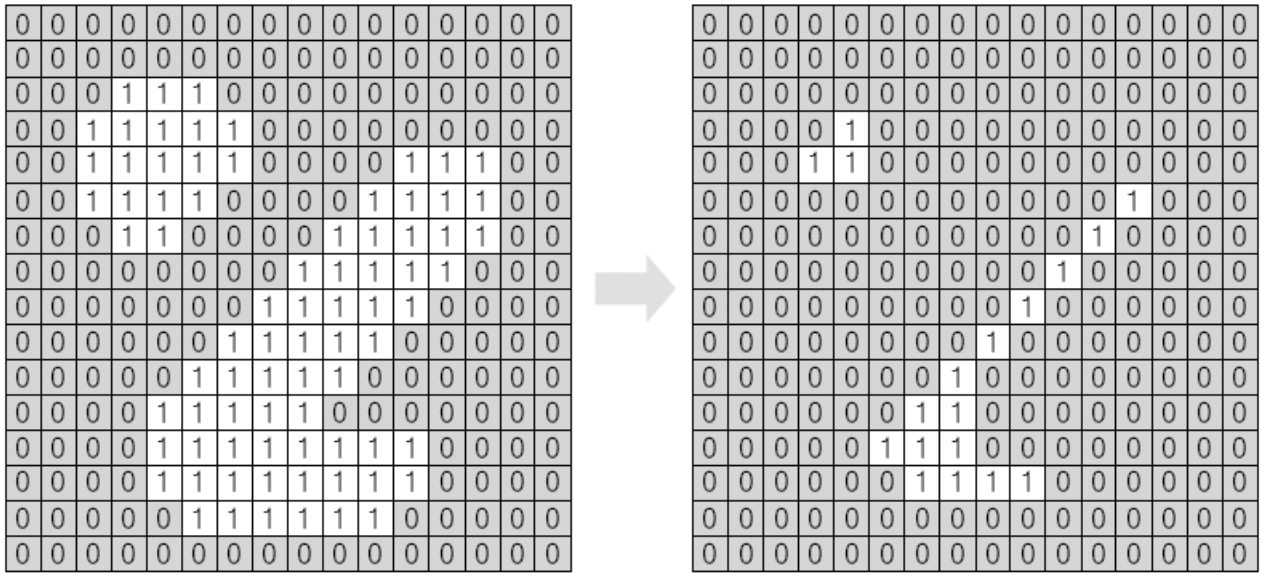
[그림 11-4] 침식의 수행 과정

침식(계속)

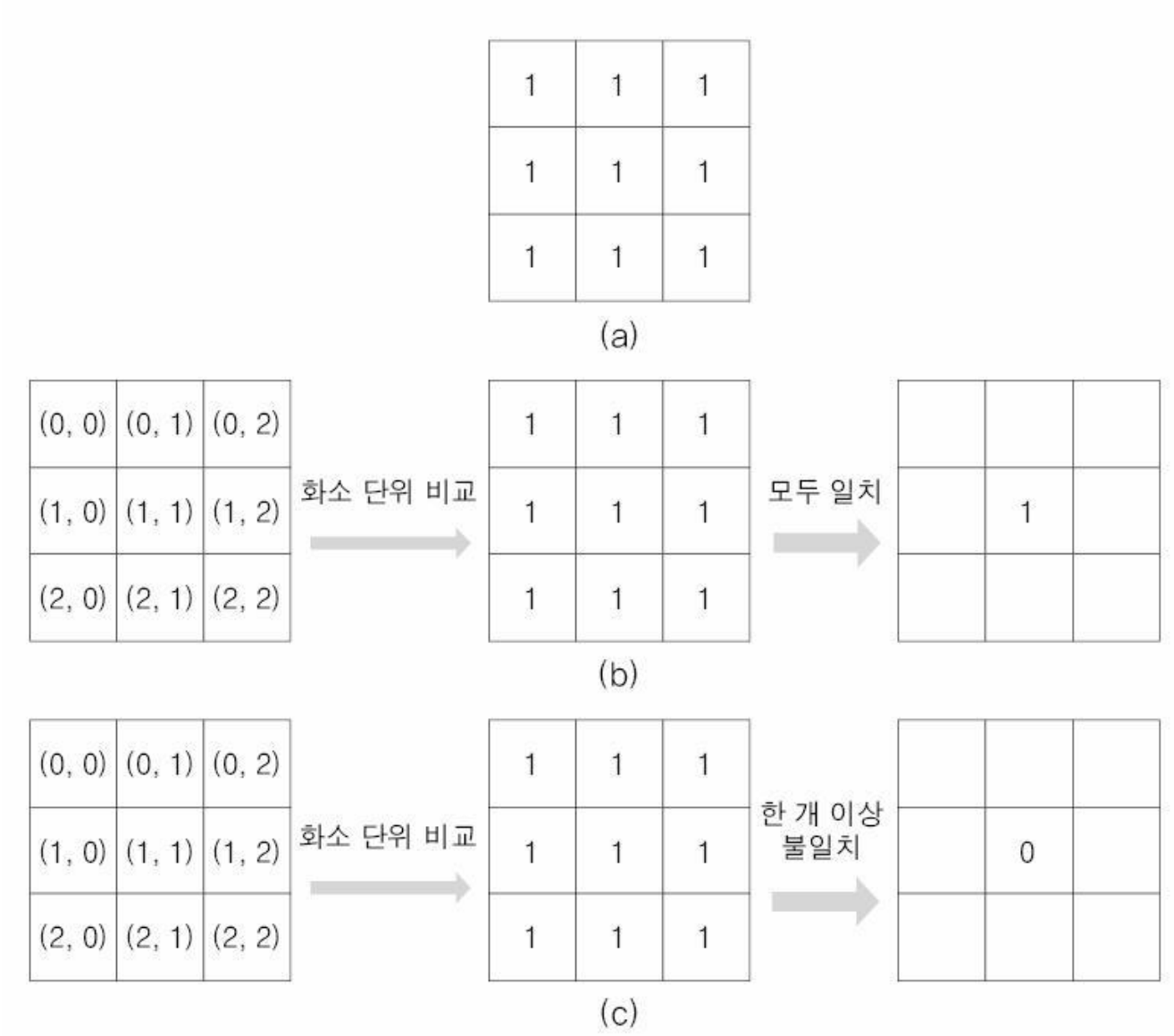
- 교환 법칙 성립

  A-B = B-A

- 형태소의 크기에 따라 침식되는 정도가 결정됨
  - 형태소의 크기가 작으면 침식의 정도도 작고, 크기가 크면 침식의 정도도 큼.
- 같은 형태소를 반복해서 적용하면 침식이 계속 일어나 객체를 완전하게 제거할 수 있음.



[그림 11-6] 이진 영상에서의 침식 처리 이해



[그림 11-5] 이진 영상에 침식 처리 수행

[실습하기 11-1] 침식 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_BINARY_EROSION
Caption	이진 영상 침식 처리

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 인접한 침식연산을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnBinaryErosion
Doc Class	void	OnBinaryErosion

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 11-1] 침식 프로그램

```
void CImageProcessingDoc::OnBinaryErosion()
{
   int i, j, n, m;
   double Mask[3][3] = {{255.,255.,255.},
                        {255.,255.,255.},
                        {255.,255.,255.}};
   // 침식연산을 위한 마스크
   double **tempInput, S = 0.0;

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;
   m_OutputImage = new unsigned char[m_Re_size];

   tempInput = Image2DMem(m_height + 2, m_width + 2);

   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         tempInput[i+1][j+1]
            = (double)m_InputImage[i * m_width + j];
      }
   }
```

[실습하기 11-1] 침식 프로그램

```
for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         for(n=0 ; n<3 ; n++){
            for(m=0 ; m<3 ; m++){
               if(Mask[n][m] == tempInput[i+n][j+m]){
               // 마스크와 같은 값이 있는지 조사
                  S += 1.0;
               }
            }
         }
         if(S == 9.0)
           m_OutputImage[i * m_Re_width + j]=(unsigned char)255.0;
           // 값이 모두 일치하면 출력 값은 255
         else
           m_OutputImage[i * m_Re_width + j]=(unsigned char)0.0;
           // 모두 일치하지 않으면 출력 값은 0
         S = 0.0; // reset
      }
   }
   delete [] tempInput;
}
```

[실습하기 11-1] 침식 프로그램

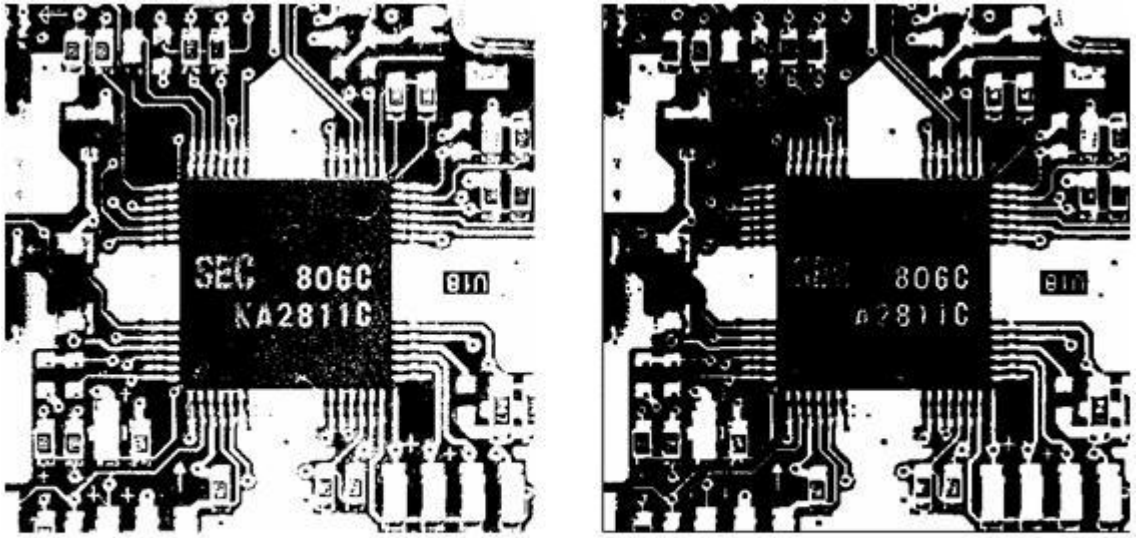
④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnBinaryErosion()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnBinaryErosion();

    Invalidate(TRUE);
}
```

[실습하기 11-1] 침식 프로그램

⑤ 프로그램 실행 결과 영상

- 실제 이진 영상에서 침식 처리를 수행한 결과로, 침식 처리한(b)영상을 보면 객체가 얇아짐.



(a) 원본 영상 (b) 침식 영상

실제 이진 영상에 침식 처리한 결과

팽창

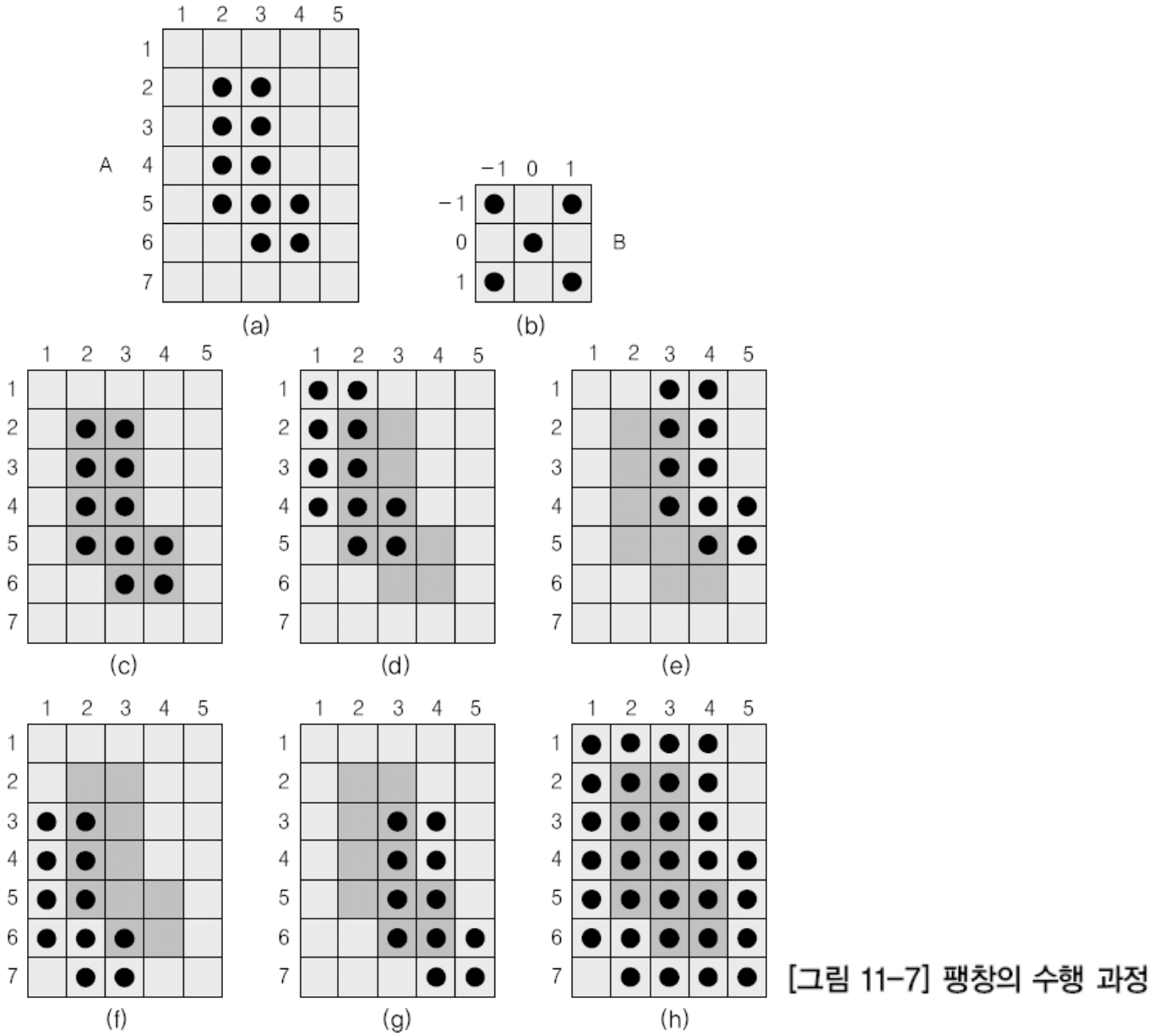
- 물체 내부의 돌출부는 감소하고 외부의 돌출부는 증가시켜서 물체의 크기를 확장하고 배경은 축소하는 기법
- 물체 내부에 발생한 구멍과 같은 공간을 채우거나 짧게 끊어진 영역을 연결하는 데 사용됨.
- 이진 영상에서 팽창연산은 입력 화소가 균일한 곳에서는 변화가 없으나 흑백 화소가 같이 있는 영역에서 동작함.

$$A \oplus B = \bigcup_{w \in B} A_w$$

$$A \oplus B = \{(a, b) + (u, v) : (a, b) \in A, (u, v) \in B\}$$

  - U는 합집합이고, B의 각 점에서 A를 이동한 뒤 더하라는 의미

팽창(계속)



[그림 11-7] 팽창의 수행 과정

팽창(계속)

- 교환 법칙뿐만 아니라 결합 법칙도 성립
  - 침식처럼 교환 법칙 때문에 형태소와 화소 집합의 역할을 바꿔도 같은 결과를 얻음.

$$A \oplus B = B \oplus A$$

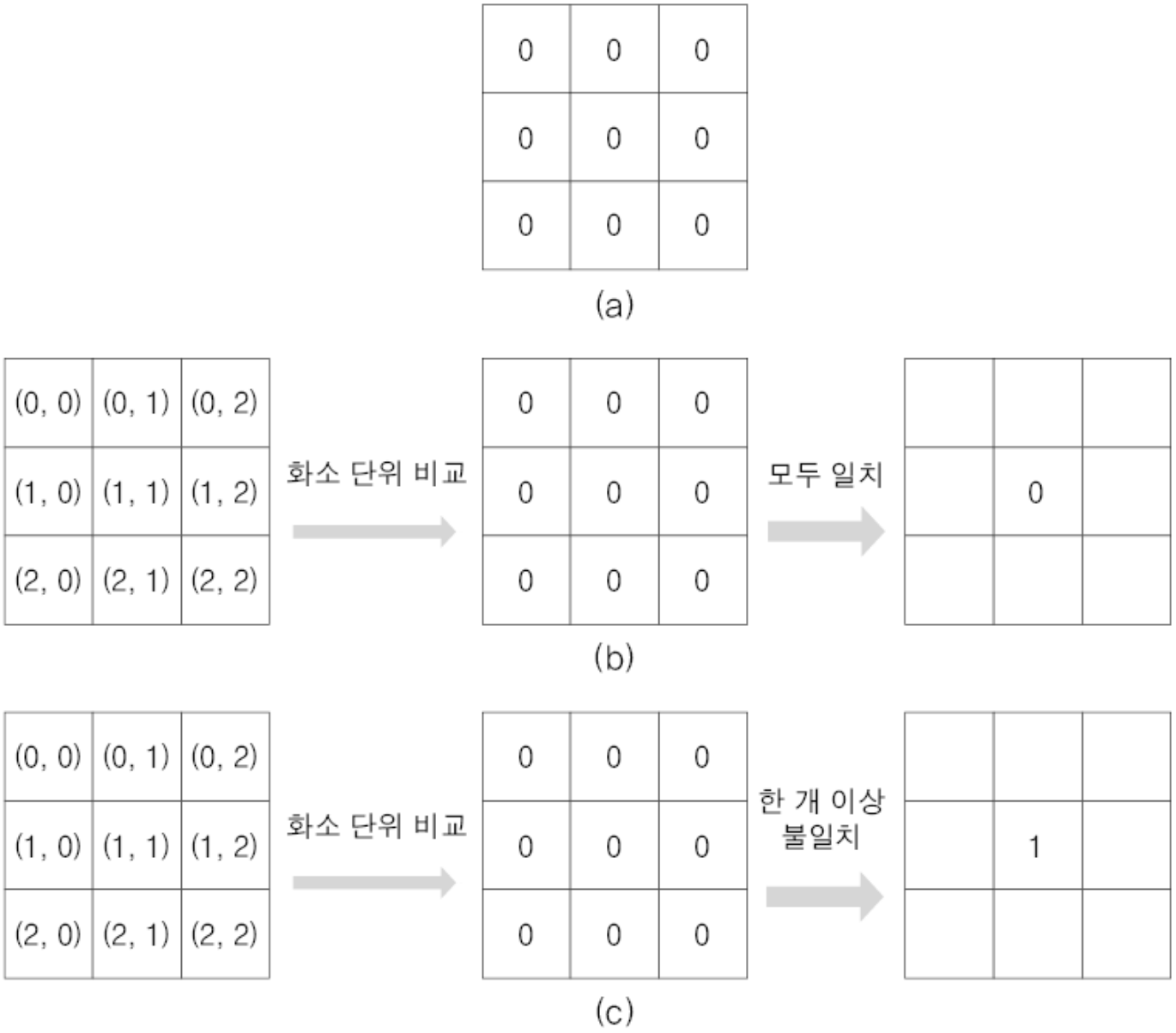
  - 결합 법칙은 두 화소 집합 간의 팽창 결과를 또 다른 화소 집합과 팽창해서 얻은 결과는 팽창의 수행 순서만 바꿔서 수행하는 것과 같다는 개념

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

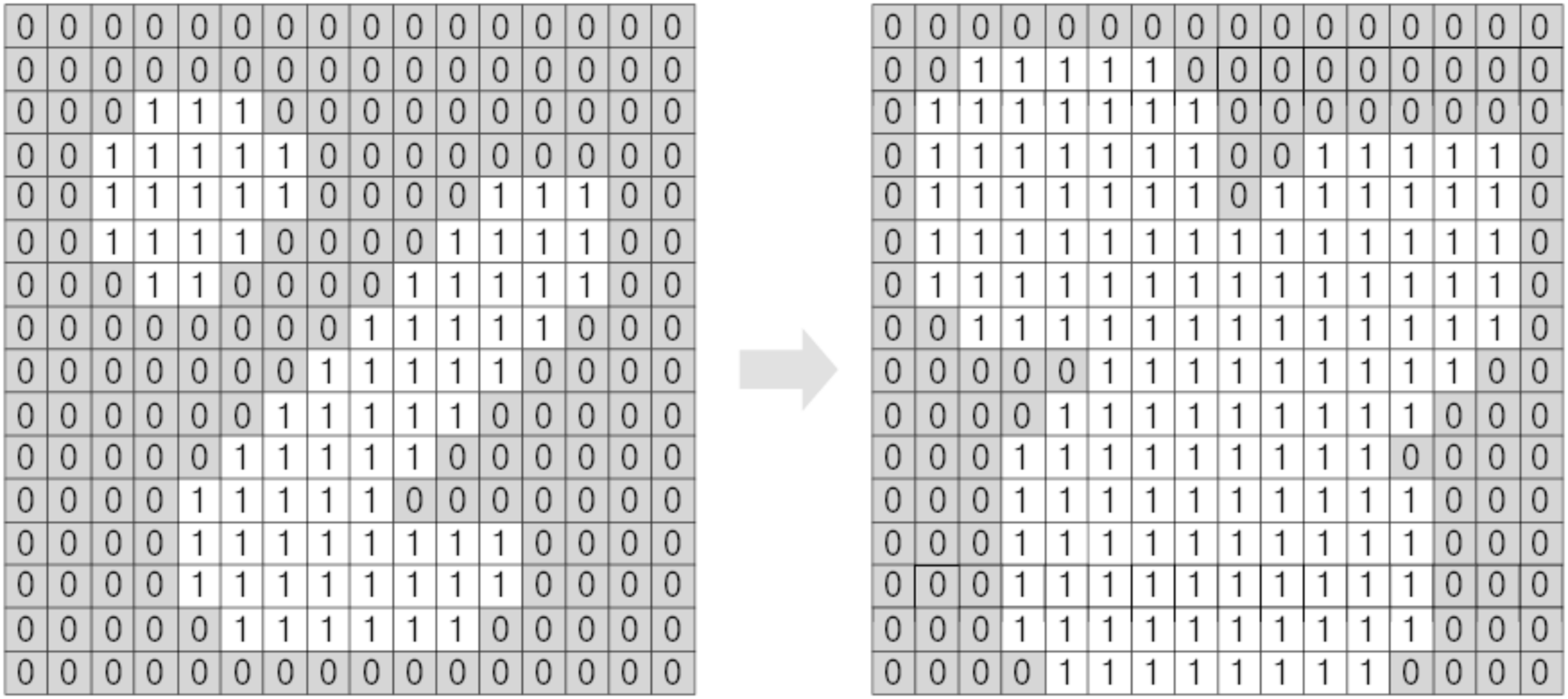
  - 결합 법칙의 다른 표현 형태

$$B = B_1 \oplus B_2 \ : \ A \oplus B = A \oplus (B_1 \oplus B_2) = (A \oplus B_1) \oplus B_2$$

  - 이진 영상에서 간단하게 수행할 수 있으며, 침식의 동작과는 반대되는 개념



[그림 11-8] 이진 영상에서 팽창 처리 수행



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

[그림 11-9] 이진 영상에서 팽창 처리 이해

[실습하기 11-2] 팽창 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_BINARY_DILATION
Caption	이진 영상 팽창 처리

② **[MFC ClassWizard] 대화상자를 이용해 추가된 팽창연산을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnBinaryDilation
Doc Class	void	OnBinaryDilation

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 11-2] 팽창 프로그램

```
void CImageProcessingDoc::OnBinaryDilation()
{
   int i, j, n, m;
   double Mask[3][3] = {{0., 0., 0.},{0., 0., 0.},{0., 0., 0.}};
   // 팽창 처리를 위한 마스크
   double **tempInput, S = 0.0;

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char[m_Re_size];

   tempInput = Image2DMem(m_height + 2, m_width + 2);

   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         tempInput[i+1][j+1]
            = (double)m_InputImage[i * m_width + j];
      }
   }
```

[실습하기 11-2] 팽창 프로그램

```
 for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         for(n=0 ; n<3 ; n++){
            for(m=0 ; m<3 ; m++){
               if(Mask[n][m] ==  tempInput[i+n][j+m]){
               // 마스크와 같은 값이 있는지 조사
                  S += 1.0;
               }
            }
         }
         if(S == 9.0)
            m_OutputImage[i * m_Re_width + j]
               = (unsigned char)0.0;
            // 모두 일치하면 출력 값은 0
         else
            m_OutputImage[i * m_Re_width + j]
               = (unsigned char)255.0;
            // 모두 일치하지 않으면 출력 값은 255
         S = 0.0;
      }
   }
   delete [] tempInput;
}
```

[실습하기 11-2] 팽창 프로그램

④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnBinaryDilation()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnBinaryDilation();

    Invalidate(TRUE);
}
```

[실습하기 11-2] 팽창 프로그램

⑤ 프로그램 실행 결과 영상

- 실제 이진 영상에서 팽창 처리를 수행한 결과 영상. 전체적으로 테두리의 흰색 화소가 증가하여 두꺼워짐.

(a) 원본 영상　　(b) 팽창 영상

실제 이진 영상에서 팽창한 결과 영상

팽창과 침식과의 관계

- 팽창과 침식은 영상의 처리 관점에서는 반대의 효과를 가져옴.
- 수학적으로 생각하면, 팽창과 침식은 각각 여집합과 반사에서 서로 이원적이라고 할 수 있음.
  - 화소의 집합 **A**와 **B**의 팽창에서 여집합은 **A**의 여집합과 **B**의 반사를 침식한 것과 같음.

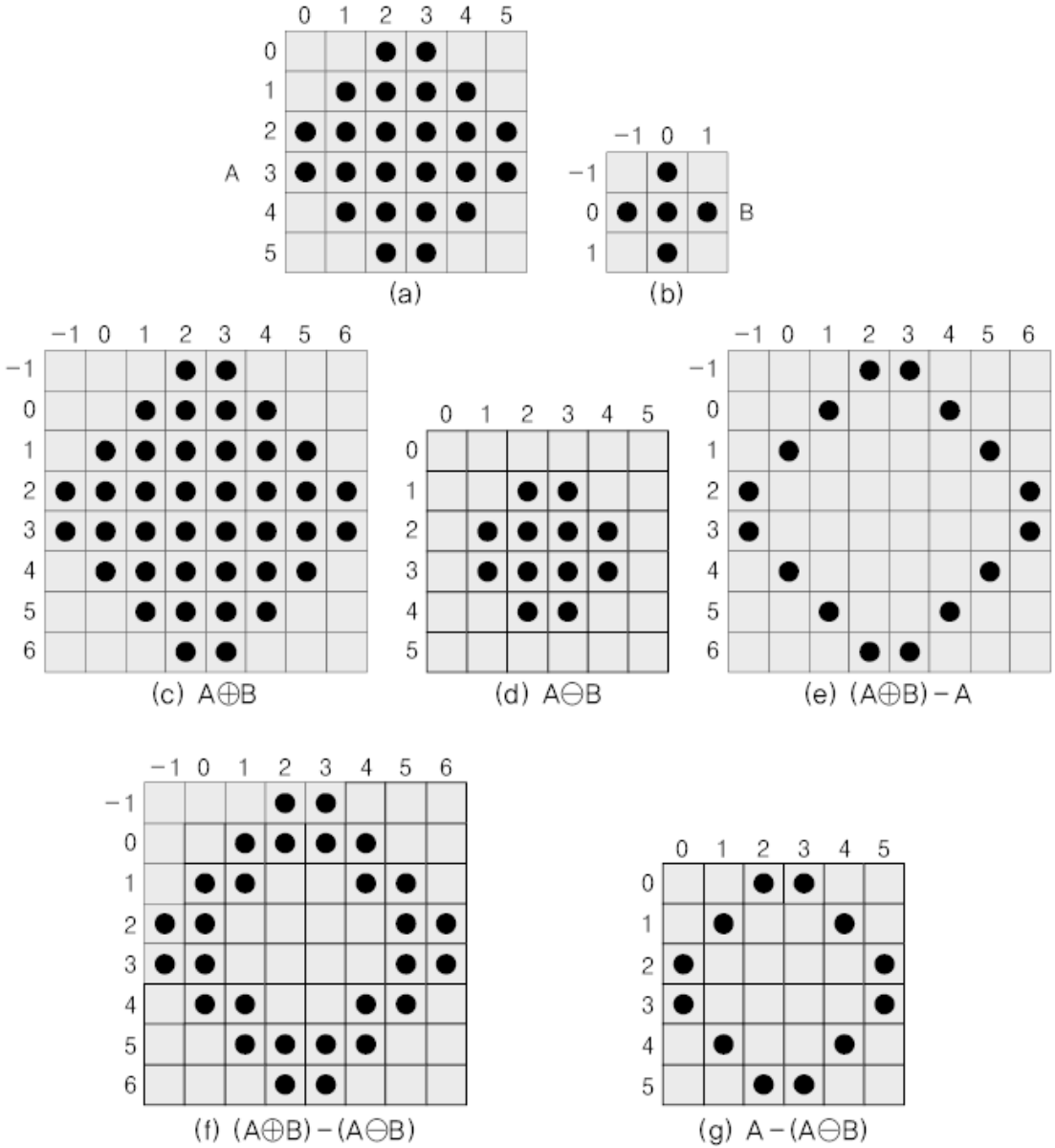
$$\overline{A \oplus B} = \overline{A} \ominus \hat{B}$$

  - **A**와 **B**의 침식에서 여집합은 **A**의 여집합과 **B**의 반사를 팽창한 것과 같음

$$\overline{A \ominus B} = \overline{A} \oplus \hat{B}$$

- 팽창과 침식을 이용한 경계의 검출 정리
  - **A**의 내부 경계 : **A- (A-B)**
  - **A**의 외부 경계 : **(A⊕B)- A**
  - **A**의 형태학적 기울기 = 내부 경계+외부 경계 : **(A⊕B)- (A-B)**

팽창과 침식과의 관계(계속)



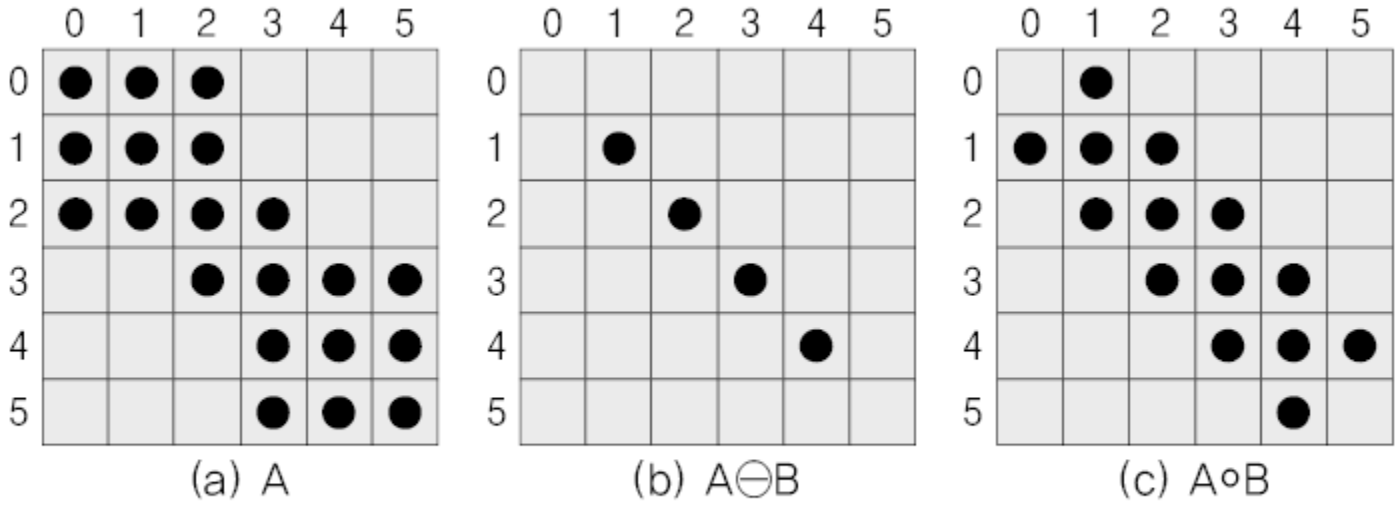
[그림 11-10] 팽창과 침식을 이용한 물체의 경계선 검출

열림

- 침식연산 다음에 팽창연산을 바로 사용하는 알고리즘
- 볼록하게 나온 부분을 제거하고 좁은 연결을 끊여서 영상의 외곽선 부분을 부드럽게 만듦.
- 물체의 형상과 크기는 보존됨.
- 돌출 부분과 좁은 연결 부위를 제거하므로 제거연산이라고도 함.
- 외곽선 부분을 더욱 부드럽게 하려면 열림연산을 반복해서 적용하여 침식연산을 특정 횟수만큼 반복한 뒤 팽창연산도 같은 횟수만큼 반복
- 화소의 집합 A와 형태소나 구조적 요소 B가 있을 때 B가 일으킨 A의 열림은 $A \circ B$로 표기하며, 다음과 같이 정의함.

$$A \circ B = (A - B) \oplus B$$

열림(계속)



[그림 11-11] 열림연산의 수행 과정

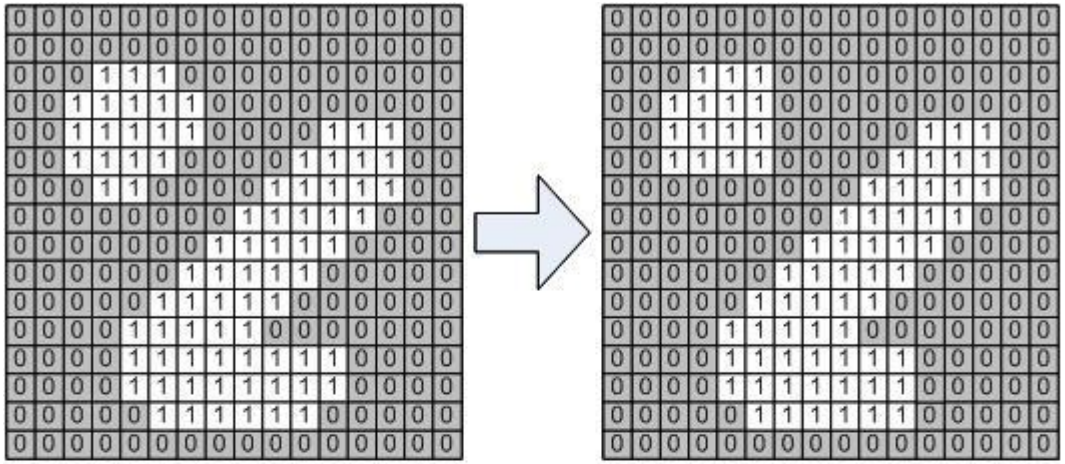
- 열림연산은 성질
  - 열림연산 **A∘B**의 결과는 **A**의 부분 집합이다.

    $$A \supset (A \circ B)$$
  - **A**가 **C**의 부분 집합이면, **(A∘B)**는 **(A∘C)**의 부분 집합이다.

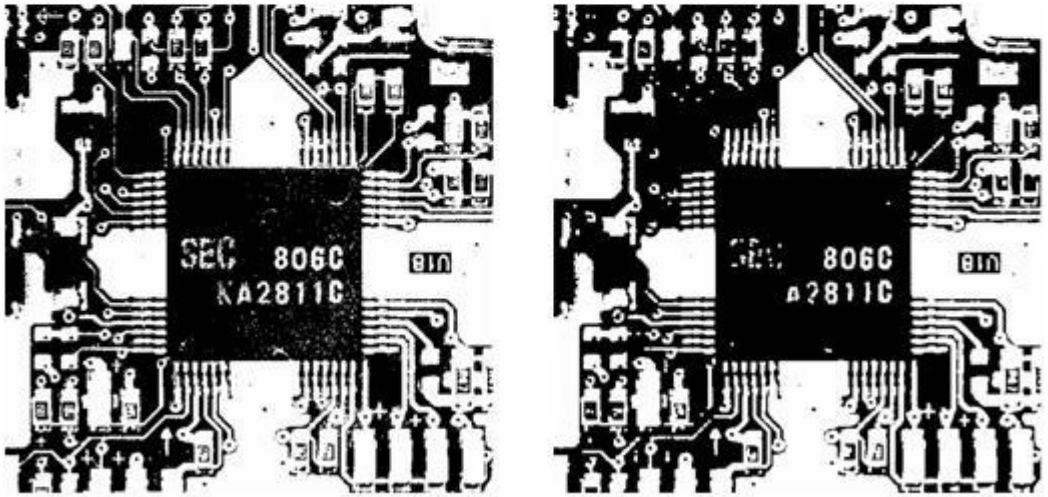
    $$(A \circ B) \subset (A \circ C)$$
  - **Idempotence** 성질은 다음과 같다.

    $$(A \circ B) \circ B = (A \circ B)$$

열림(계속)



[그림 11-12] 열림연산을 이진 영상에서 수행한 개념



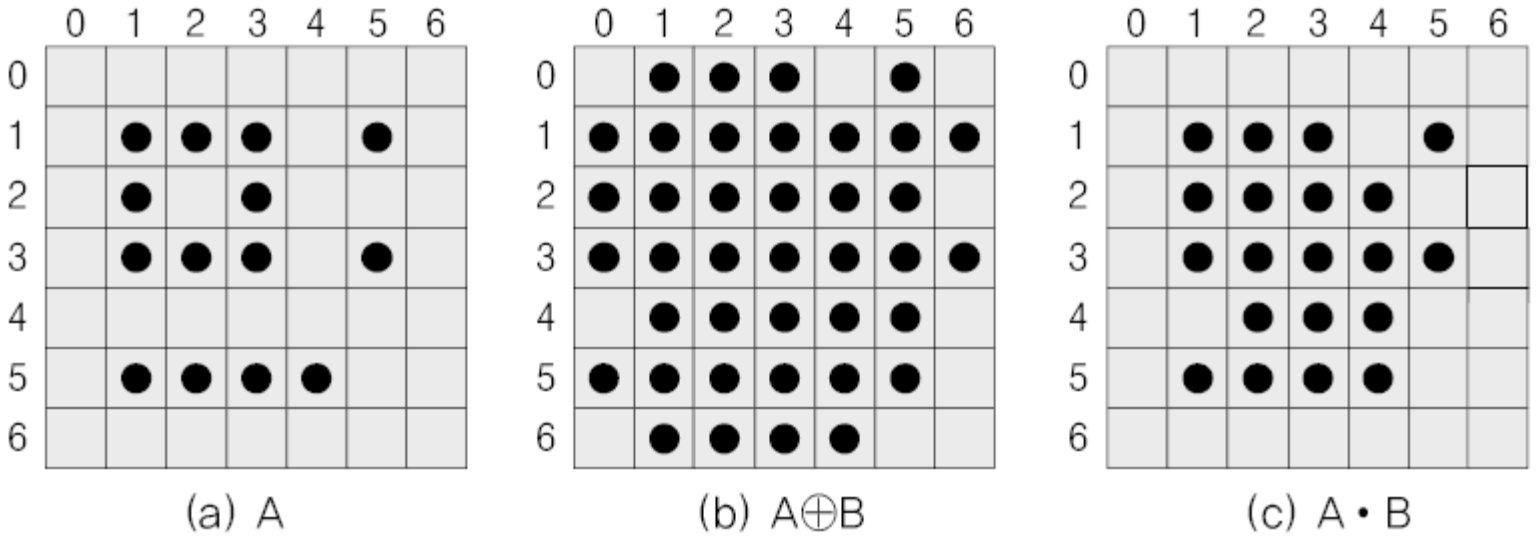
(a) 원 영상 (b) 열림연산 결과 영상

[그림 11-13] 실제 이진 영상에 열림연산을 적용한 결과 영상

닫힘

- 열림과 반대로 팽창을 처리한 뒤에 침식 처리 수행
- 오목하게 들어간 부분이나 작은 구멍을 채우기에 열림 연산과 마찬가지로 영상의 외곽선 부분을 부드럽게 만들며, 객체의 형태와 크기는 보존됨.
- 작은 구멍이나 틈 등을 채우는 역할을 하므로 채움연산이라고도 함.
  - 화소의 집합 A와 형태소 B가 주어졌을 때 B가 일으킨 A의 닫힘은 A•B로 표기

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



[그림 11-14] 닫힘연산의 수행 과정

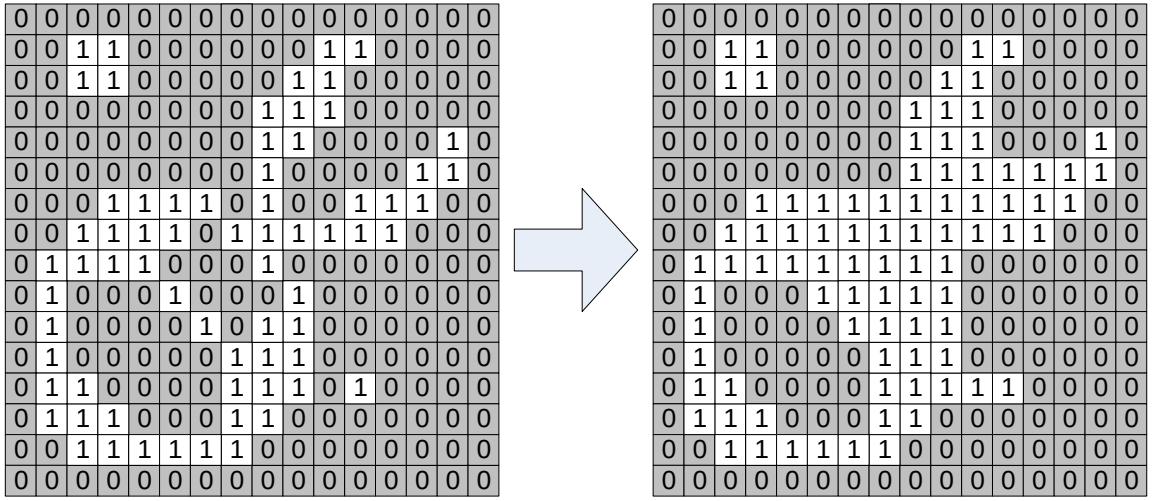
닫힘(계속)

- 닫힘연산 성질
  - A는 A•B의 부분 집합이다.

    $A \subset A \bullet B$
  - A가 C의 부분 집합이면, (A•B)는 (C•B)의 부분 집합이다.

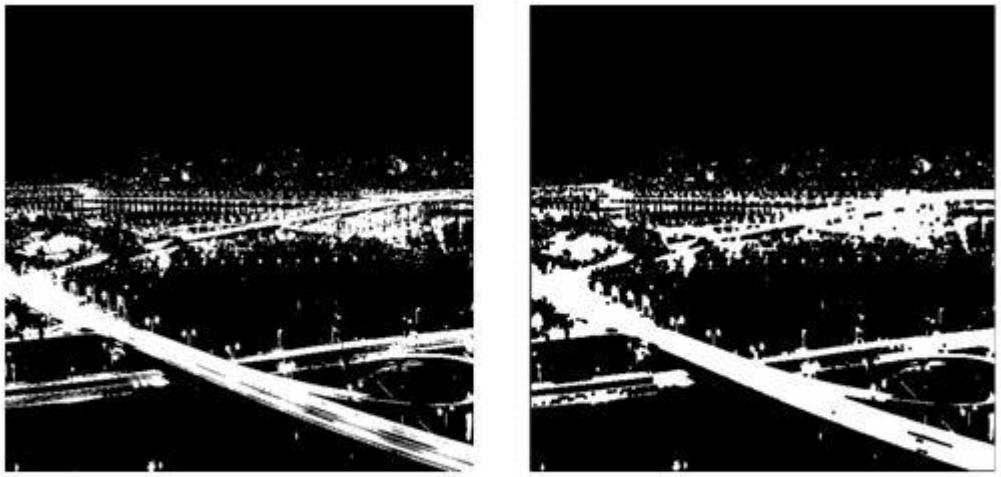
    $(A \bullet B) \subset (C \bullet B)$
  - Idempotence 성질은 다음과 같다.

    $(A \bullet B) \bullet B = (A \bullet B)$



[그림 11-15] 닫힘 · 열림연산을 이진 영상에서 수행한 개념

열림과 닫힘의 관계



(a) 원 영상 (b) 닫힘연산 결과영상

[그림 11-16] 실제 이진 영상에 닫힘연산을 적용한 결과 영상

- 팽창과 침식처럼 여집합과 반사에 이원적이어서 다음과 같이 표현 가능

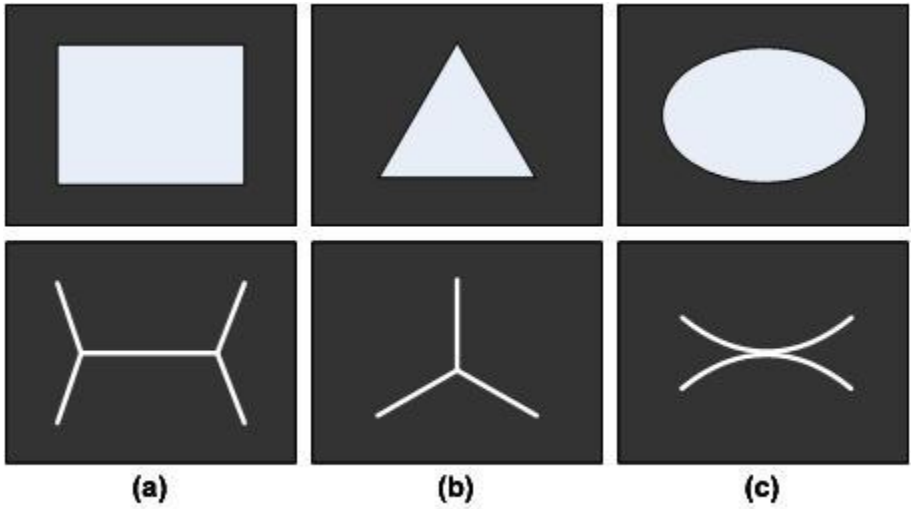
$$\overline{A \circ B} = \bar{A} \bullet \hat{B}$$

- 다음과 같이 반대도 성립

$$\overline{A \bullet B} = \bar{A} \circ \hat{B}$$

골격화(Skeletonization)

- 이진 영상에서 물체의 크기와 모양을 요약하는 선과 곡선의 집합으로 만드는 것
- 골격을 정의하는 방법이 다양하므로 주어진 물체에서 서로 다른 모양의 골격도 있을 수 있음
- 골격화는 침식연산을 이용해 수행
- 서로 다른 방향성이 여러 개 있는 침식 마스크를 이용하여 영상 내의 물체를 서서히 깎아 다듬어 침식을 반복하여 객체의 하부 구조를 표현할 수 있게 됨

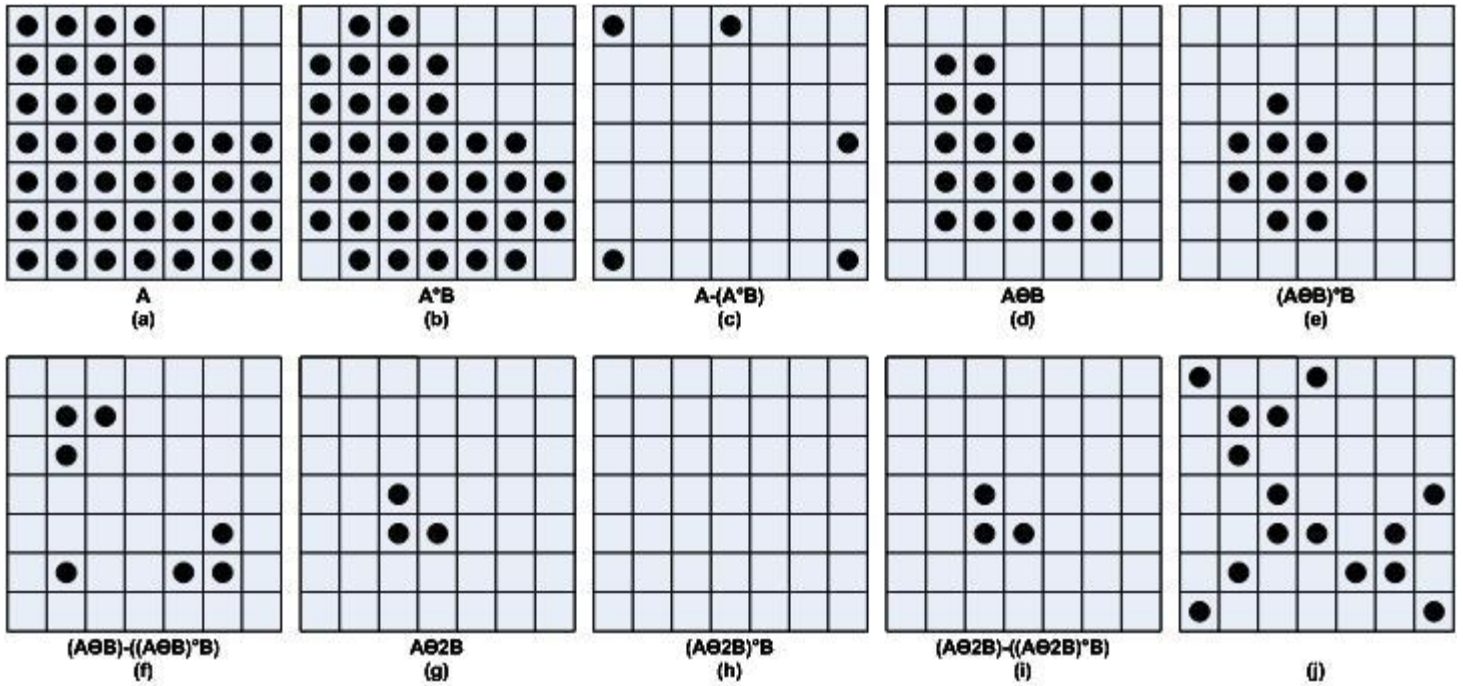


[그림 11-17] 기본 도형의 골격화

골격화(계속)

침식	열림	차집합
A	A ∘ B	A−(A∘B)
A⊖B	(A⊖B)∘B	(A⊖B)−((A⊖B)∘B)
A⊖2B	(A⊖2B)∘B	(A⊖2B)−((A⊖2B)∘B)
A⊖3B	(A⊖3B)∘B	(A⊖3B)−((A⊖3B)∘B)
…	…	…
A⊖kB	(A⊖kB)∘B	(A⊖kB)−((A⊖kB)∘B)

[표 11-1] 골격화 수행 과정



[그림 11-18] 골격화 수행 과정

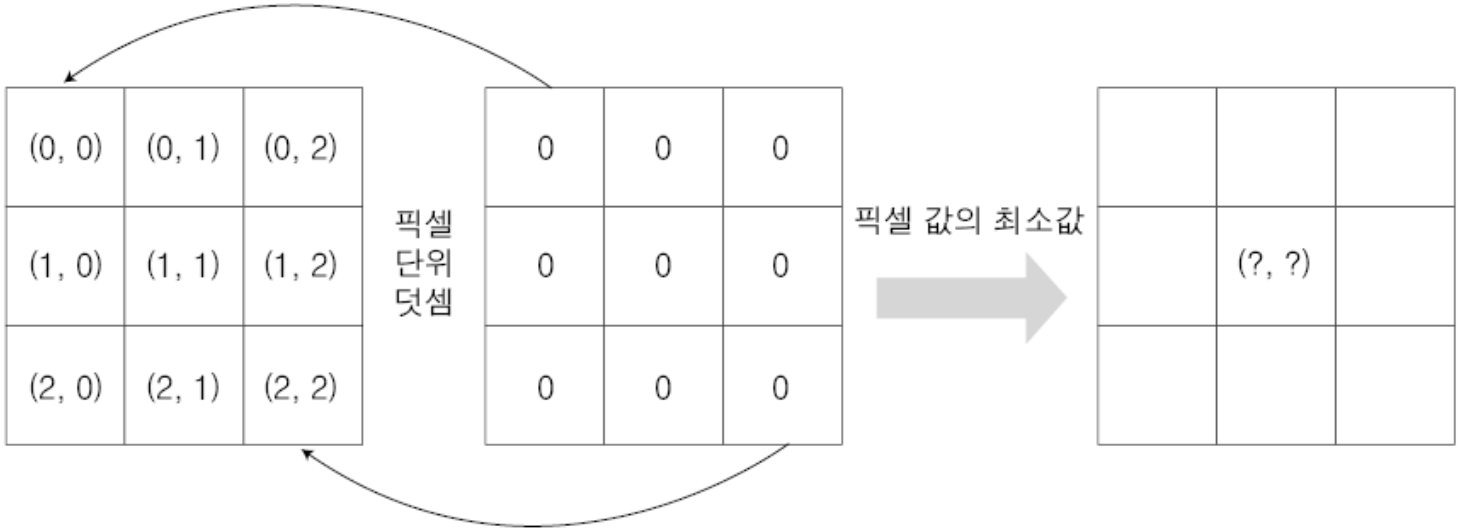
- 그레이 영상의 침식과 팽창연산
  - 물체의 밝기와 배경의 밝기 간에 격차가 클 때는 그레이 영상의 침식과 팽창 연산이 효과적
  - 이때, 사용되는 침식 마스크와 팽창 마스크는 같은 형태소 마스크로, 형태소의 모든 요소 값이 0인 것을 사용
  - 형태소 마스크와의 연산 결과로 얻은 최대값과 최소값을 선택하느냐에 따라 침식과 팽창이 결정됨.

0	0	0
0	0	0
0	0	0

[그림 11-19] 그레이 영상에서 침식과 팽창의 형태소

그레이 영상의 침식연산

- 그레이 영상에서 밝은 객체를 더 어둡게 보이도록 하여 축소하는 효과를 얻는 게 침식
- 밝기가 균일하지 않은 영역에서는 효과적으로 처리되나, 균일한 영역에서는 입력 화소의 밝기와 같은 결과를 보임. 균일하지 않은 영역에서 반복해서 적용하면 물체가 사라짐.
- 화소 집합의 화소 값을 -255~0까지 변화시킬 수 있도록 사용하는 형태소를 구성하여 그레이 영상에 침식연산을 수행할 수 있음. -255에 가까운 값으로 형태소 마스크를 구성하면 더욱 분명하게 침식 효과가 나타남.

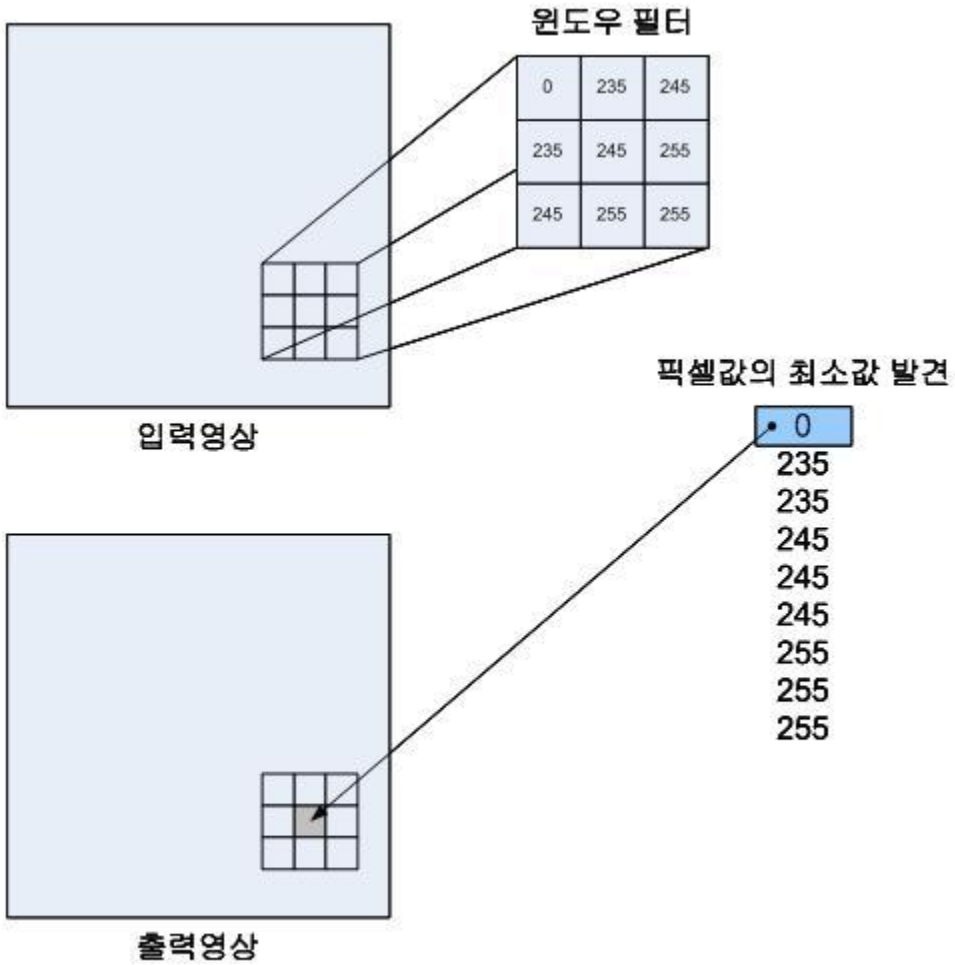


[그림 11-20] 그레이 영상에서 침식 수행 과정

그레이 영상의 침식연산(계속)

- 결과 화소가 결정되는 것을 다음과 같이 표현 가능

  O(x, y) = min{입력 화소 밝기와 형태소 마스크 화소의 각 합}



[그림 11-21] 그레이 영상에서 발견한 최소값으로 침식 수행

[실습하기 11-3] 그레이 영상의 침식 연산 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_GRAY_EROSION
Caption	그레이 영상 침식 처리

② **[MFC ClassWizard] 대화상자를 이용해 그레이 영상의 침식연산을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnGrayErosion
Doc Class	void	OnGrayErosion

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 11-3] 그레이 영상의 침식 연산 프로그램

```
void CImageProcessingDoc::OnGrayErosion()
{
   int i, j, n, m, h;
   double Mask[9], MIN = 10000.0; // MIN = 최소값
   double **tempInput, S = 0.0;

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char[m_Re_size];

   tempInput = Image2DMem(m_height + 2, m_width + 2);

   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         tempInput[i+1][j+1]
            = (double)m_InputImage[i * m_width + j];
      }
   }
```

[실습하기 11-3] 그레이 영상의 침식 연산 프로그램

```
for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         MIN = 10000.0; // reset
         for(n=0 ; n<3 ; n++){
            for(m=0 ; m<3 ; m++){
               Mask[n * 3 + m] = tempInput[i+n][j+m];
               // 3*3 크기의 입력 값을 마스크 배열에 저장
            }
         }
         for(h = 0 ; h<9 ; h++){
            if(Mask[h] < MIN) // 마스크에서 최소값을 구한다.
               MIN = Mask[h];
         }
         m_OutputImage[i * m_Re_width + j]
            = (unsigned char)MIN; // 최소값 출력
      }
   }
}
```

[실습하기 11-3] 그레이 영상의 침식 연산 프로그램

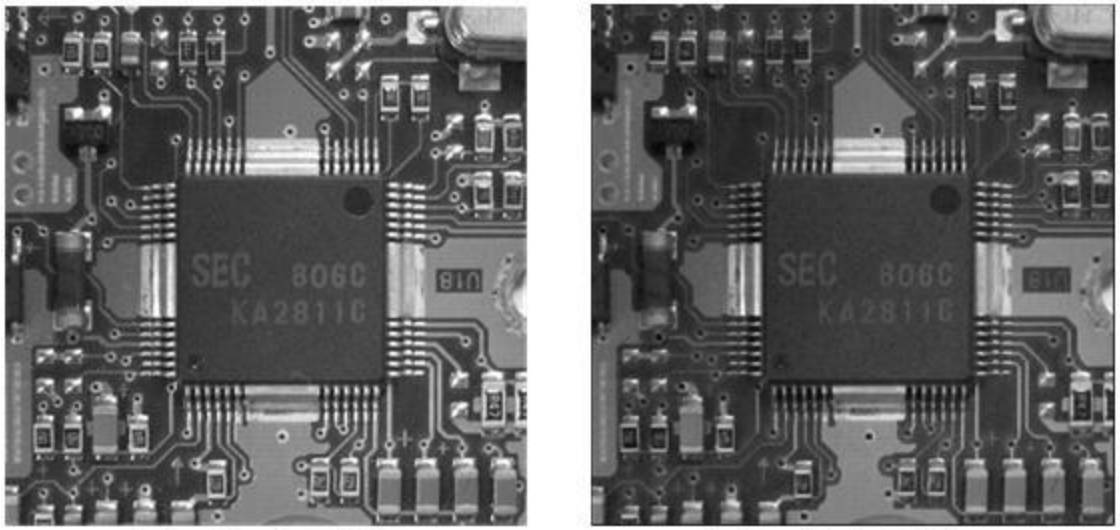
④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnGrayErosion()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnGrayErosion();

    Invalidate(TRUE);
}
```

[실습하기 11-3] 그레이 영상의 침식 연산 프로그램

⑤ 프로그램 실행 결과 영상

- 각 객체의 밝은 색이 줄어들어 축소된 효과

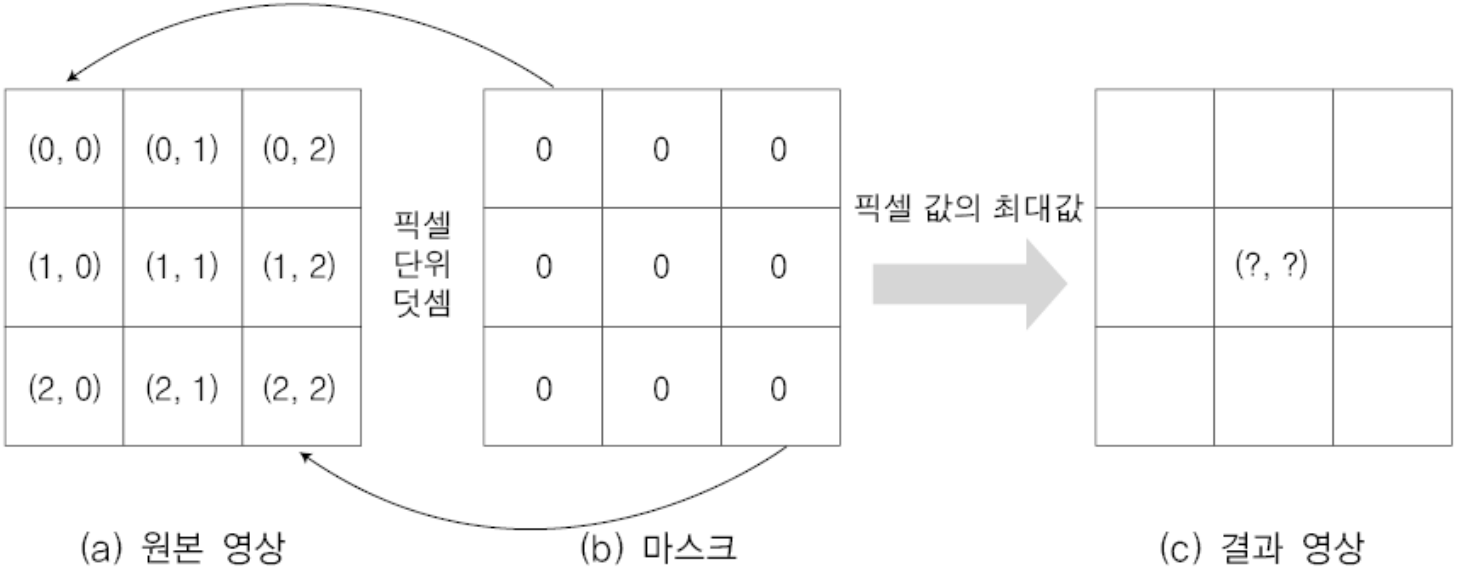


(a) 입력 영상 (b) 결과 영상

실제 그레이 영상에서 침식을 수행한 결과 영상

그레이 영상의 팽창연산

- 그레이 영상에서 객체를 더 밝게 하여 객체를 크게 보이게 하는 효과
- 침식처럼 밝기가 균일하지 않은 영역에서 효과적으로 동작
- 사용되는 형태소 마스크의 특징은 모든 화소를 0~255까지의 범위로 변화시킬 수 있도록 구성할 수 있음.
- 팽창연산의 결과로 화소 값이 255에 가까운 값이 되도록 형태소 마스크를 구성하면 더욱 분명한 팽창 효과를 얻을 수 있음.

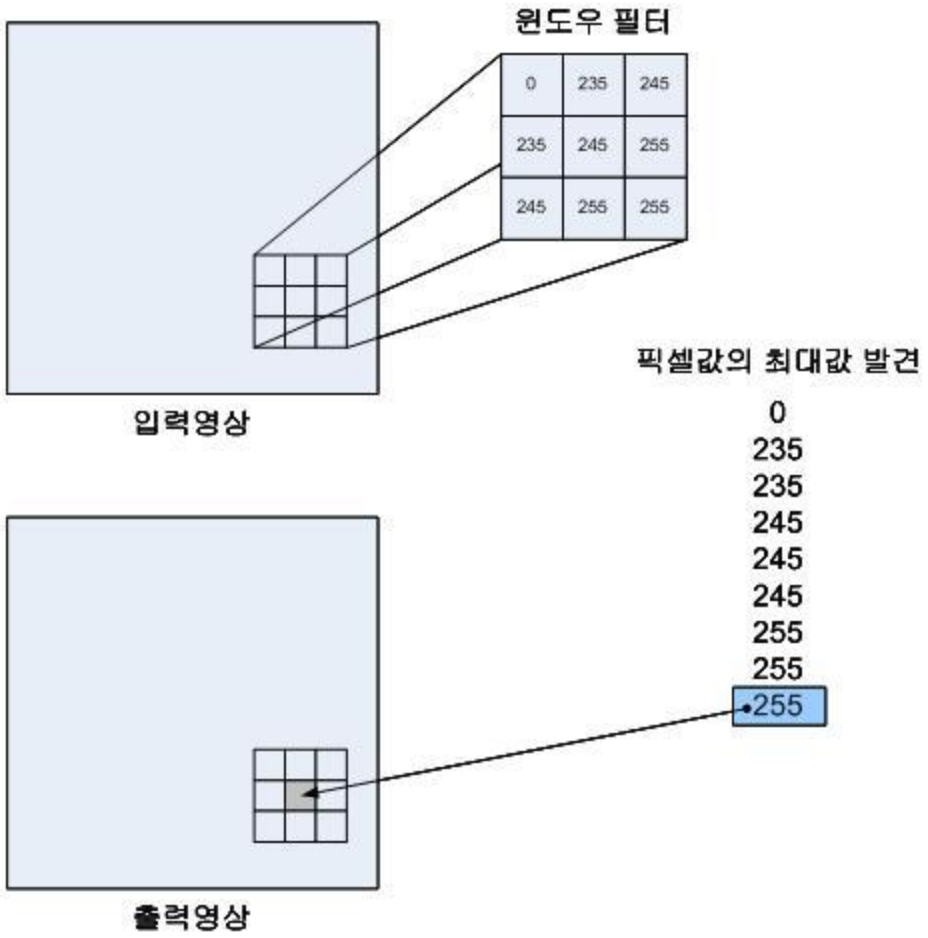


[그림 11-22] 그레이 영상에서 팽창 수행 과정

그레이 영상의 팽창연산(계속)

- 결과 화소가 결정되는 것은 다음과 같이 표현 가능

  O(x, y) = max{입력 화소 밝기와 마스크 화소의 각 합}



[그림 11-23] 그레이 영상에서 발견한 최대값으로 팽창 수행

[실습하기 11-4] 그레이 영상의 팽창 연산 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_GRAY_DILATION
Caption	그레이 영상 팽창 처리

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 그레이 영상의 팽창연산을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnGrayDilation
Doc Class	void	OnGrayDilation

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 11-4] 그레이 영상의 팽창 연산 프로그램

```
void CImageProcessingDoc::OnGrayDilation()
{
   int i, j, n, m, h;
   double Mask[9], MAX = 0.0;
   double **tempInput, S = 0.0;

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char[m_Re_size];

   tempInput = Image2DMem(m_height + 2, m_width + 2);

   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         tempInput[i+1][j+1]
            = (double)m_InputImage[i * m_width + j];
      }
   }
```

[실습하기 11-4] 그레이 영상의 팽창 연산 프로그램

```
for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         MAX = 0.0;
         for(n=0 ; n<3 ; n++){
            for(m=0 ; m<3 ; m++){
               Mask[n * 3 + m] = tempInput[i+n][j+m];
            }
         }
         for(h = 0 ; h<9 ; h++){ // 마스크에서 최대값 구함
            if(Mask[h] > MAX)
               MAX = Mask[h];
         }
         m_OutputImage[i * m_Re_width + j]
            = (unsigned char)MAX; // 최대값 출력
      }
   }
}
```

[실습하기 11-4] 그레이 영상의 팽창 연산 프로그램

④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnGrayDilation()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnGrayDilation();

    Invalidate(TRUE);
}
```

[실습하기 11-4] 그레이 영상의 팽창 연산 프로그램

⑤ 프로그램 실행 결과 영상

- 입력 영상보다 객체의 밝기가 밝아져 팽창된 결과

(a) 입력 영상 (b) 결과 영상

실제 그레이 영상에서 팽창을 수행한 결과 영상

그레이 영상의 열림과 닫힘연산

- 이진 영상에서 열림과 닫힘연산은 침식과 팽창으로 얻음.
- 그레이 영상도 그레이 영상의 침식과 닫힘연산을 이용하해 그레이 영상에서 열림과 닫힘연산을 수행
- 이진 영상의 결과처럼 경계선이 부드러워진 효과

(a) 입력 영상 (b) 열림연산 결과 영상 (c) 닫힘연산 결과 영상

[그림 11-24] 실제 그레이 영상에서 열림과 닫힘연산을 수행한 결과 영상

요약

- 영상의 형태학
  - 영상 내의 형태를 분석하고 처리하는 유용한 영상 처리 기법
  - 영상 내의 경계와 블록, 골격 등의 형태를 표현하거나 서술하는 데 필요한 영상요소를 추출하는 데 형태학 처리를 활용
- 형태학 처리
  - 이진 영상의 밝기 값에 형태소라는 행렬과 논리적인(AND, OR, NOT, …) 연산을 수행하여 출력 화소를 결정하는 것
- 형태소의 크기에 따라 침식되는 정도가 결정됨
  - 형태소의 크기가 작으면 침식의 정도도 작고, 크기가 크면 침식의 정도도 큼.
  - 같은 형태소를 반복해서 적용하면 침식이 계속이 일어나 객체를 완전하게 제거할 수 있음.
- 침식
  - 물체의 크기를 그 배경과 관련하여 일정하게 줄여주는 것
  - 물체의 크기는 줄어들고, 배경은 확대됨
- 팽창
  - 물체 내부의 돌출부는 감소하고 외부의 돌출부는 증가시켜서 물체의 크기를 확장하고 배경은 축소하는 기법

요약

- 골격화
  - 이진 영상에서 물체의 크기와 모양을 요약하는 선과 곡선의 집합으로 만드는 것
- 골격화 수행 방법
  - 서로 다른 방향성이 여러 개 있는 침식 마스크를 이용하여 영상 내의 물체를 서서히 깎아 다듬으면 침식을 반복하여 객체의 하부 구조를 표현할 수 있게 됨.
- 그레이 영상에서 형태학 처리
  - 화소끼리의 비교가 아닌 화소의 명도 값 크기로 결과 영상을 결정하게 됨.
- 그레이 영상에서 침식과 팽창연산
  - 형태소 마스크가 같고, 보통 형태소의 모든 요소 값이 0인 것을 사용
- 그레이 영상의 침식과 닫힘연산
  - 그레이 영상에서 열림과 닫힘연산을 수행하게 되는 것
  - 이진 영상의 결과처럼 경계선이 부드러워짐.

Thank you