

# MATLAB을 이용한 디지털 영상처리의 기초

# 제 11장 영상의 위상기하적 처리

## 11.1 서론

우리는 종종 한 영상의 아주 기본적인 모양에만 관심을 가지는 경우가 있는데, 영상에 구멍(hole) 등 특정 물체의 수 또는 존재 여부에 관심을 가지는 경우이다. 한 영상의 이들 기본적인 성질의 조사를 디지털위상기하학(digital topology) 혹은 영상위상기하학(image topology)이라 하고 이 장에서 이 주제를 위한 더욱 기본적인 모양을 몇 가지 조사해 보기로 한다.

예를 들면 blob들의 모임을 보이기 위해, 영상의 문턱치 처리와 형태학적 열림 처리로서 깔끔한 처리를 아래와 같이 고려해 보자.

```
>> n=imread('nodules1.tif');  
>> nt=~im2bw(n,0.5);  
>> n2=imopen(nt,strel('disk',5));
```

 반지름이 5인 원

영상 n2를 그림 11.1에 보였다. blob들의 수는 형태학적 방법으로 결정될 수 있다. 그러나 영상의 지형적 처리는 다른 이와는 다른 분야이고, 물체의 수를 세는 것에 매우 강력한 방법이다. 보는 바와 같이 골격화는 위상기하학적 방법을 이용하여 매우 효과적으로 실행될 수 있다.

Strel ('disk',5) % structural element (matlab 함수)

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

## 제 11장 영상의 위상기하적 처리

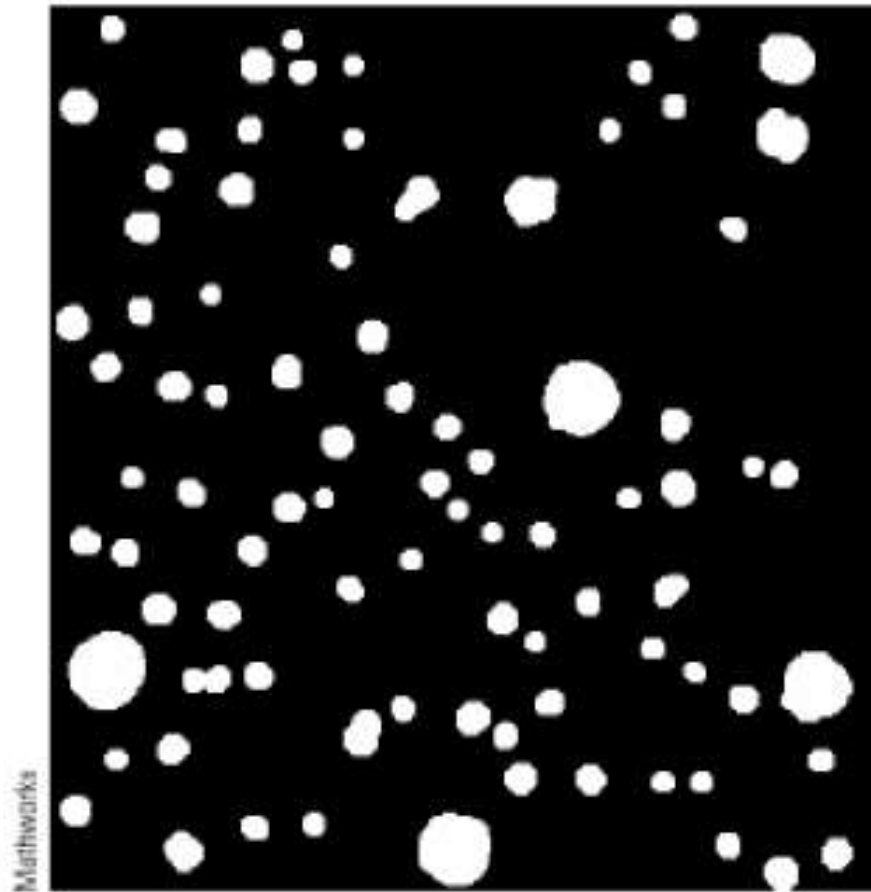


그림 11.1 blob의 수는 몇 개 ?

# 제 11장 영상의 위상기하적 처리

## 11.2 이웃화소(Neighbors)와 인접화소(Adjacency)

먼저 인접의 개념을 정의하면, 이것은 하나의 화소에 대하여 한 화소를 건너서 존재하는 것은 무시하는 조건이다. 이 장에서는 이진 영상만을 대상으로 하고, 화소의 위치만 다루게 된다.

하나의 화소 P는 아래와 같이 각각 4개의 이웃과 8개의 이웃을 가지고 있다.

	•	
•	P	•
	•	

•	•	•
•	P	•
•	•	•

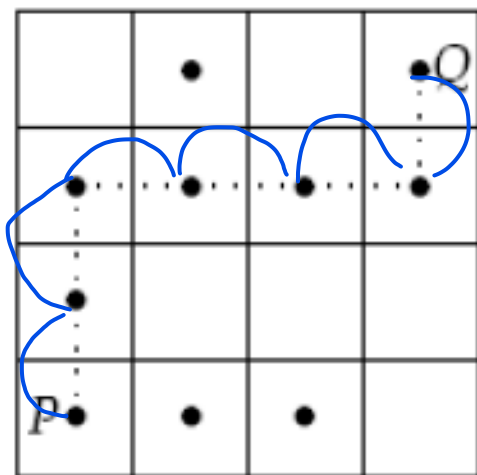
점에 데이터가 있으면 연결 판별

화소 P가 4개의 이웃이 있는 경우는 4-인접이고, 8개의 이웃이 있는 경우는 8-인접이다.

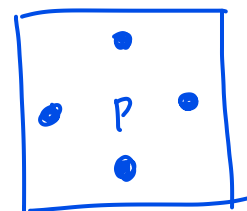
# 제 11장 영상의 위상기하적 처리

## 11.3 경로(path)와 성분(components)

P와 Q는 어떤 2개의 화소(서로 인접할 필요는 없음)이고, P와 Q는 아래와 같이 화소들의 수열로 연결될 수 있다고 가정한다.



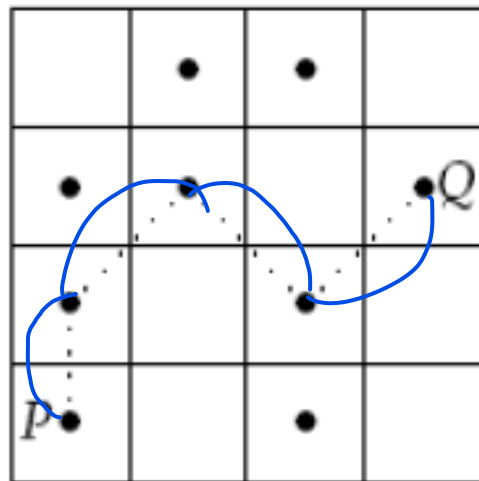
P에서 Q 까지



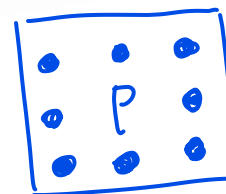
Distance= 6 ( 4연결 )

위 다이어그램에서 경로로서 단지 4-인접 화소로만 구성한다면 P와 Q는 4-연결 형태이다. 만일 8-인접 화소로 경로를 구성하면 8-연결 형태로도 가능 하다. 다음 그림은 8-연결 화소의 예를 나타낸다.

## 제 11장 영상의 위상기하적 처리



P에서 Q까지

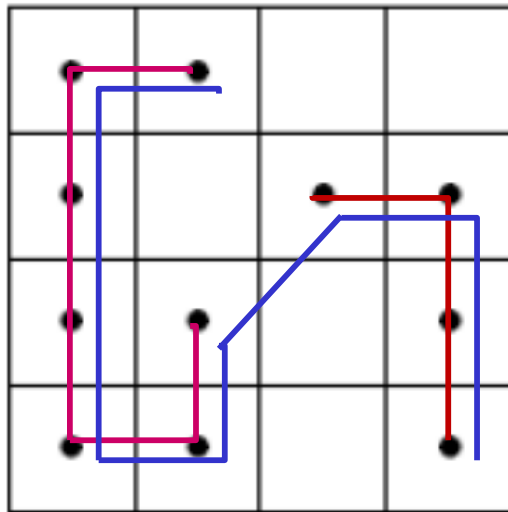


Distance= 4 ( 8연결 )

서로 4-연결된 모든 화소들의 집합을 4-성분이라 한다. 만일 모든 화소들이 8-연결이면 그 집합은 8-성분이라 한다.

예를 들면 다음의 영상은 2개의 4-성분(하나의 성분은 왼쪽 2개의 열에 있는 모든 화소들이고, 또 하나는 오른쪽 2개의 열에 모든 화소들이다.) 그러나 전체가 하나의 8-성분이 된다.

# 제 11장 영상의 위상기하적 처리



Red: 2개의 4 연결 대각선 불가 → object 2개  
 Blue: 1 개의 8 연결 " 가능 → object 1개

아래와 같이 보다 형식적으로 경로(path)를 정의할 수 있다.  
 P에서 Q까지 4-경로는 화소들의 수열이다.

$$P = p_0, p_1, p_2, \dots, p_n = Q$$

각  $i = 0, 1, 2, 3, \dots, n-1$ 에 대하여 화소  $p_i$  는 화소  $p_{i+1}$  까지 4-인접이다.

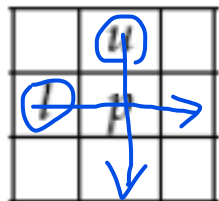
8-경로는 P와 Q를 연결하는 수열에서 화소들은 8-인접인 경우이다.





# 제 11장 영상의 위상기하적 처리

## 11.5 성분의 라벨링 *object* 를 세는 방법

이 절에서는 이진 영상의 모든 4-성분을 라벨링하는 알고리즘을 다룬다. 이때 처리는 좌측 위에서 우측으로, 위에서 아래로 진행한다. 만일  $p$ 가 현재 화소이면 아래와 같이  $u$ 가 위쪽 4-이웃이고,  $l$ 이 좌측 4-이웃이라고 하자.



좌측에서 우측으로 움직이면서 1행씩 영상을 스캔한다. 영상에서 화소들에 라벨을 부여하고, 이들 라벨들에 영상의 성분들의 수치를 붙인다. 

주석을 목적으로, 영상에 존재하는 화소는 foreground(목적)화소로 처리하고, 영상에 없는 화소는 background(배경)화소로 처리한다. 알고리즘은 아래와 같이 설명할 수 있다. 

## 제 11장 영상의 위상기하적 처리

1.  $p$ 의 상태를 확인하라. 만일  $p$ 가 배경화소이면 다음 스캔의 위치로 이동하라. 목적화소이면  $u$ 와  $l$ 을 확인하라. 만일 그들( $u$ 와  $l$ )이 모두 배경화소들이면  $p$ 에 새로운 라벨을 부여하라.(이것은 새로운 연결성분이 생성되는 경우이다.)
    - $u$ 와  $l$ 중의 하나가 목적화소이면  $p$ 에 라벨을 부여하라.
    - $u$ 와  $l$ 이 모두 목적화소이고, 같은 라벨을 가지면  $p$ 에 그 라벨을 부여하라.
    - $u$ 와  $l$ 이 모두 목적화소이지만, 다른 라벨이면 그들 라벨 중의 하나를  $p$ 에 부여하고 그들 2개의 라벨이 서로 등가로 만들어라. ( $u$ 와  $l$ 이  $p$ 에서 동일한 4-연결이 되기 때문이다.)
  2. 스캔의 마지막에서, 모든 목적화소들은 라벨링이 되었지만, 몇 개의 라벨이 등가일 수도 있다. 그 라벨들을 등가분류로 분류하고 각 분류에 다른 라벨을 부여한다.
  3. 각 목적화소에 있는 라벨을 이전 단계에서 등가분류로 부여된 라벨을 치환하면서 영상을 통해 2번째로 처리를 하라.
- 실제로 이 알고리즘의 예를 보면, 아래 그림과 같이 2개의 4-성분을 가지는 이진 영상에서 왼쪽 위에 3개의 화소들과 오른쪽 아래에 5개의 화소들이 있는 경우이다.

## 제 11장 영상의 위상기하적 처리

	•		
•	•		•
		•	•
	•	•	

단계 1 : 위를 따라 움직이기 시작한다. 첫 번째 목적화소는 2번째 칸에 있고, 이것의 왼쪽과 위쪽의 이웃에는 배경화소와 존재하지 않는 화소이므로 라벨-1을 아래와 같이 부여한다.

	•1		
•	•		•
		•	•
	•	•	

## 제 11장 영상의 위상기하적 처리

2번째 행에서, 첫째 목적화소는 역시 왼쪽 및 위쪽의 이웃이 배경 및 무화소이므로 새로운 라벨-2를 아래와 같이 부여한다.

	•1		
•2	•		•
		•	•
	•	•	

현재 2번째 행에서 2번째(목적)화소는 위쪽과 왼쪽에 목적화소가 될 이웃이 존재한다. 그러나 이들은 서로 다른 라벨을 가진다. 따라서 2번째 화소에 이들 중의 하나를 부여하고, 라벨 1과 2는 등가로 아래와 같이 처리한다.

	•1		
•2	•1		•
		•	•
	•	•	

{1, 2}는 등가

## 제 11장 영상의 위상기하적 처리

2번째 행에서 3번째 목적화소는 위와 왼쪽에 모두 배경화소인 이웃을 가지므로 새로운 라벨-3을 아래와 같이 부여한다.

3번째 행에서, 첫째 목적화소는 위와 왼쪽에 배경화소가 있으므로 새로운 라벨-4를 부여한다. 3번째 행에서 2번째 목적화소는 위와 왼쪽에 모두 목적화소가 있고 라벨이 다르므로 라벨-3 또는 라벨-4 중 하나로 부여하고 위와 같이 아래와 같이 등가로 처리한다.

4번째 행에서 첫째 목적화소는 위와 왼쪽에 배경화소만 있으므로 새로운 라벨-5를 부여한다.

	•1		
•2	•1		•3
		•	•
	•	•	

# 제 11장 영상의 위상기하적 처리

	•1		
•2	•1		•3
		•4	•3
	•	•	

{3,4} 는 등가

여한다. 4번째 행의 2번째 목적화소는 위와 왼쪽에 목적화소가 있으므로 라벨-4 또는 라벨-5 중의 하나를 부여하고, 라벨-4와 라벨-5를 아래와 같이 등가로 처리한다.

	•1		
•2	•1		•3
		•4	•3
	•5	•4	

{4,5} 는 등가 -> {3,4,5} 는 모두 등가

이렇게 해서 단계 1이 완성된다.

단계 2. 다음의 라벨들의 등가분류를 처리한다.

{1,2} 및 {3,4,5}.

첫째 분류에 라벨-1을 부여하고 2번째 분류에 라벨-2를 부여하라.

# 제 11장 영상의 위상기하적 처리

단계 3. 단계 1로부터 라벨 1과 2를 가진 각 화소에 라벨 1을 부여하고, 라벨 3,4,5를 가진 각 화소에 라벨 2를 아래와 같이 부여한다.

	•1		
•1	•1		•2
		•2	•2
	•2	•2	

이로서 알고리즘의 실행이 끝난다. *objects는 2개*

이 알고리즘은 영상의 8-성분으로 수정될 수 있지만, 단계 1에서 아래와 같이 p의 대각요소를 고려할 필요가 있다.

d	u	e
l	p	

이 알고리즘은 앞의 것과 유사하다. 단계 1이 아래와 같이 다르다.

## 제 11장 영상의 위상기하적 처리

단계 1. 만일  $p$ 가 배경화소이면 다음 스캔 위치로 이동하라. 만일  $p$ 가 목적화소이면  $d, u, e$  및  $l$ 을 확인하라. 만일 그들이 모두 배경화소들이면  $p$ 에 새로운 라벨을 부여하라. 만일 하나의 화소라도 목적화소가 있으면  $p$ 에 그 라벨을 부여하라. 만일 2개 이상의 목적화소가 있으면 그들 중 어느 하나를 부여하고 그들의 모든 화소들을 등가로 처리한다.

단계 2와 3은 이전과 동일하다.

이 알고리즘은 `bwlabel` 함수로 구현된다. 작은 영상으로 아래와 같이 예를 들어 보기로 한다.

```
>> i=zeros(8,8);  
>> i(2:4,3:6)=1;  
>> i(5:7,2)=1;  
>> i(6:7,5:8)=1;  
>> i(8,4:5)=1;  
>> i
```



# 제 11장 영상의 위상기하적 처리

이 결과는 아래와 같다.

0	0	0	0	0	0	0	0	0						
0	0	1	1	1	1	1	1	0	0	Red: 4연결				
0	0	1	1	1	1	1	1	0	0	Blue: 8연결				
0	0	1	1	1	1	1	1	0	0					
0	1	2	1	0	0	0	0	0	0					
0	1	2	1	0	0	1	3	2	1	3	2	1	3	2
0	1	2	1	0	0	1	3	2	1	3	2	1	3	2
0	0	0	1	4	2	1	3	2	0	0	0			

{3,4} 는 등가

이 영상은 2가지의 8-성분과 4-성분 볼 수 있다. 이를 보기 위해 아래와 같이 처리하면 4-성분에 대한 결과를 아래와 같이 얻을 수 있다

```
>> bwlabel(i,4)
```

*bwlabel*

## 제 11장 영상의 위상기하적 처리

4연결: 대각 성분을 건너지 못한다.

0	0	0	0	0	0	0	0
0	0	2	2	2	2	0	0
0	0	2	2	2	2	0	0
0	0	2	2	2	2	0	0
0	1	0	0	0	0	0	0
0	1	0	0	3	3	3	3
0	1	0	0	3	3	3	3
0	0	0	3	3	0	0	0

같은 방법으로 아래와 같이 처리하면 8-성분의 결과를 아래와 같이 얻는다.

```
>> bwlabel(i, 8)
```

8연결: 대각 성분을 건널 수 있다.

0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	1	0	0	0	0	0	0
0	1	0	0	2	2	2	2
0	1	0	0	2	2	2	2
0	0	0	2	2	0	0	0

## 제 11장 영상의 위상기하적 처리

실제 영상을 실험하기 위해, bacteria.tif 의 영상에서 박테리아의 수를 계산해보자. 먼저 박테리아만 볼 수 있는 이진 영상을 얻기 위해 문턱치 처리를 해야 하고, 그 후에 그 결과에 bwlabel을 적용한다. 여기서 만들어진 가장 큰 라벨을 간단히 아래와 같이 구하여 물체들의 수를 구할 수 있다.

```
>> b=imread('bacteria.tif');  
>> bt=b<100; 100 보다 작은 픽셀은 백색  
>> bl=bwlabel(bt);  
>> max(bl(:))
```

이렇게 하여 필요한 수의 결과를 아래와 같이 얻을 수 있다.

```
>> 21
```

## 11.6 Look-Up 테이블

Look-Up 테이블은 이진 영상처리에서 매우 적절하고 효과적인 방법이다. 화소의 3 X 3 이웃을 생각하자. 9개의 이웃 화소들이 존재하므로 서로 다른 상태의 조합으로 표시할 수 있는 총 경우의 수는  $2^9=512$ 개 이다. 이진 연산의 출력은 0이 아니면 1이므로 look-up 테이블은, 대응하는 이웃들로부터 출력을 표현하는 각 요소는 길이가 512인 벡터이다. 이것은 3.4절에서 설명한 look-p 테이블과 약간 다르다. 3.4 절에서의 look-up 테이블은 단일 화소 값에 적용되었지만, 여기서는 이웃 화소들에 적용된다.

이웃 화소들과 출력 화소들 사이에 1대1 대응이 되도록 모든 이웃 화소들을 정렬하는 일종의 속임수이다. 이것은 이웃 화소의 각 화소에 아래와 같이 가중치를 준다.

1	8	64
2	16	128
4	32	256



Pattern 번호	결과 (1 이면 경계)
0	0 또는 1
1	0 또는 1
2	0 또는 1
.	0 또는 1
.	0 또는 1
.	0 또는 1
.	0 또는 1
.	0 또는 1
.	0 또는 1
.	0 또는 1
511	0 또는 1

## 제 11장 영상의 위상기하적 처리

이웃 화소의 값은 하나의 값을 가진 화소들의 가중치를 더해서 얻어진다. 그래서 그 값은 look-up 테이블에 index(지표)이다. 예를 들면 다음은 이웃 화소와 그들의 값들을 나타낸다.

0	1	0
1	1	0
0	0	1

1	0	0
0	1	1
1	1	1

1	8	64
2	16	128
4	32	256

과 곱을 한것

$$\text{Value} = 2 + 8 + 16 + 256 = 282 \text{ 번째 패턴}$$

$$\text{Value} = 1 + 4 + 16 + 32 + 128 + 256 = 437 //$$

## 제 11장 영상의 위상기하적 처리

look-up 테이블을 적용하기 위해 먼저 테이블을 만들어야 한다. 각 요소를 하나씩 look-up 테이블을 만들기가 귀찮은 일이다. 그래서 규칙에 따라 look-up 테이블을 정의하는 makelut 함수를 이용한다. 이의 구문은 아래와 같다.

`makelut(function, n, P1, P2, ...)`

마스크 크기

여기서 함수는 MATLAB의 매트릭스 함수를 정의하는 문자열이고, n은 2 혹은 3이며, p1, p2.. 등은 그 함수를 pass하게 될 선택적인 파라미터이다. makelut은 2X2 이웃 화소들에 대한 look-up 테이블을 허용한다. 이 경우에 look-up 테이블은 단지  $2^4=16$  요소들을 가진다.

우리는 한 영상의 4-경계를 구한다고 가정하자. 하나의 배경화소에 4-인접하는 목적화소이면 경계화소로 정의한다. 그래서 makelut로 사용될 함수는 3X3 이웃의 중앙의 화소가 경계화소이면 아래와 같이 1을 return 한다.

```
>> f=inline('x(5)&~(x(2)*x(4)*x(6)*x(8))'); 4 연결 함수
```

참이면  
1 반환

25p

## 제 11장 영상의 위상기하적 처리


이 함수에 대하여  $x(5)$ 는  $3 \times 3$  매트릭스  $x$ 의 중심화소이고,  $x(2)$ ,  $x(4)$ ,  $x(6)$ ,  $x(8)$ 은 중심에 4-인접 화소들이 되도록 하는 단일 값의 매트릭스 인덱싱 구조이다. 이제 look-up 테이블을 아래와 같이 만들 수 있다.

```
>> lut=makelut(f,3); 함수, 마스크 크기
```

여기서 영상을 아래와 같이 적용하면, 그림 11.2를 얻을 수 있다.

```
>> c=imread('circles.tif');  
>> cw=applylut(c,lut); c에 lut 적용  
>> imshow(c),figure,imshow(cw)
```





1	2	3
4	5	6
7	8	9

#### 4 연결을 고려할때

X(5) 가 1 일때 (목적화소) x(2), x(4), x(6), x(8) 중 하나라도 0 이면

x(5)는 경계가 되고 1을 return 한다.

Pattern 의 수  $2^4 = 16$  개의 pattern 을 갖는다.

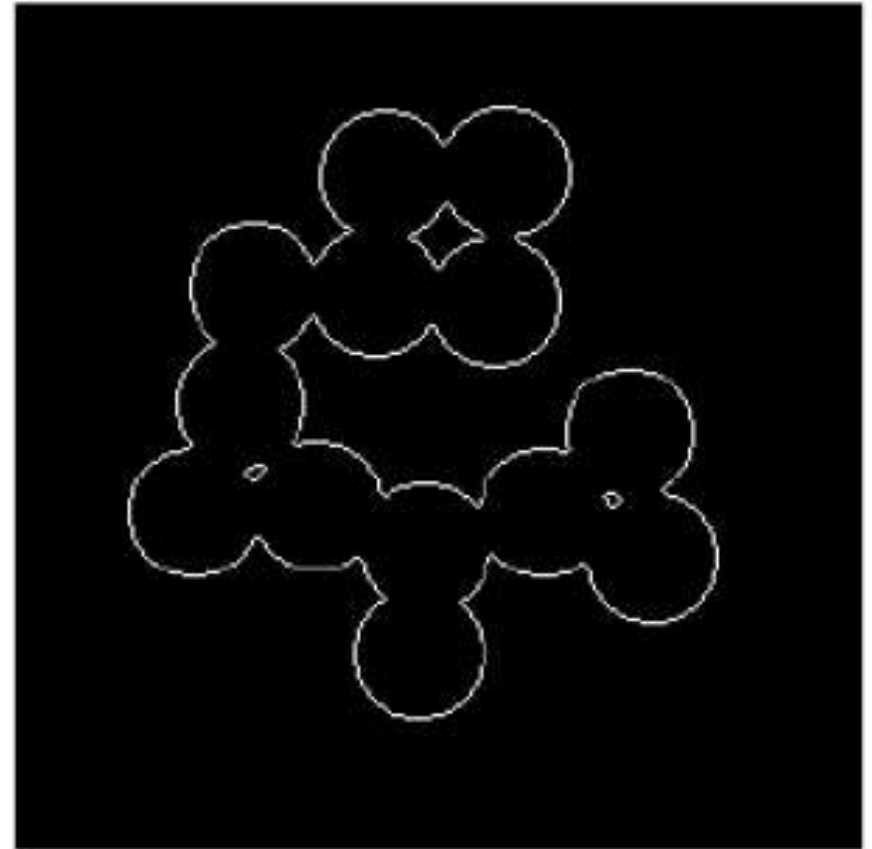
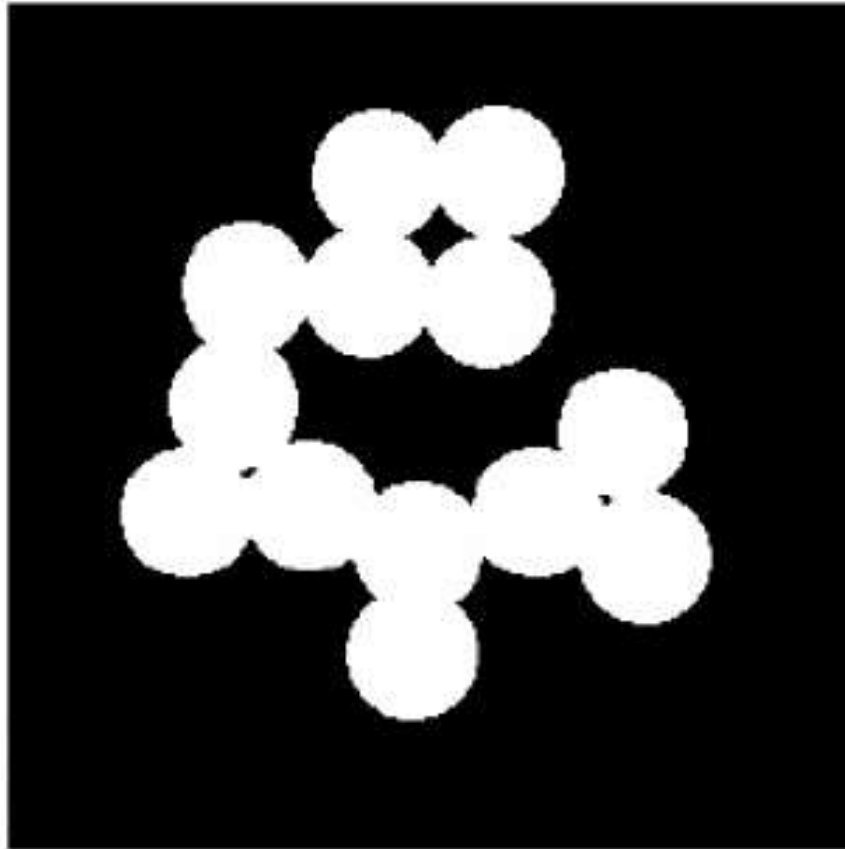


그림 11.2 circle 영상과 그 경계검출 결과


## 제 11장 영상의 위상기하적 처리

또, 8-경계의 화소들을 구하기 위해 아래와 같이 이 함수를 쉽게 조정할 수 있는데 목적화소들은 배경화소들에 8-인접이 된다.

```
>> f8=inline('x(5)&~(x(1)*x(2)*x(3)*x(4)*x(6)*x(7)*x(8)*x(9))');  
>> lut=makelut(f8,3);  
>> cw=applylut(c,lut);  
>> imshow(cw)
```

28p

이 결과는 그림 11.3과 같다. 이 방법은 보다 많은 화소들이 경계화소로 분류되므로 더욱 강력한 경계를 검출할 수 있다. 11.8 절에서 언급하겠지만, look-up 테이블은 골격화를 실행하는데 우수한 특성을 보인다.



1	2	3
4	5	6
7	8	9

**8 연결을 고려할때**

**X(5) 가 1일때 (목적화소) x(1), x(2), x(3),x(4), x(6), x(7),x(8),x(9) 중**

**하나라도 0 이면**

**x(5)는 경계가 되고 1을 return 한다.**

**Pattern 의 수  $2^8 = 256$  개의 pattern 을 갖는다.**

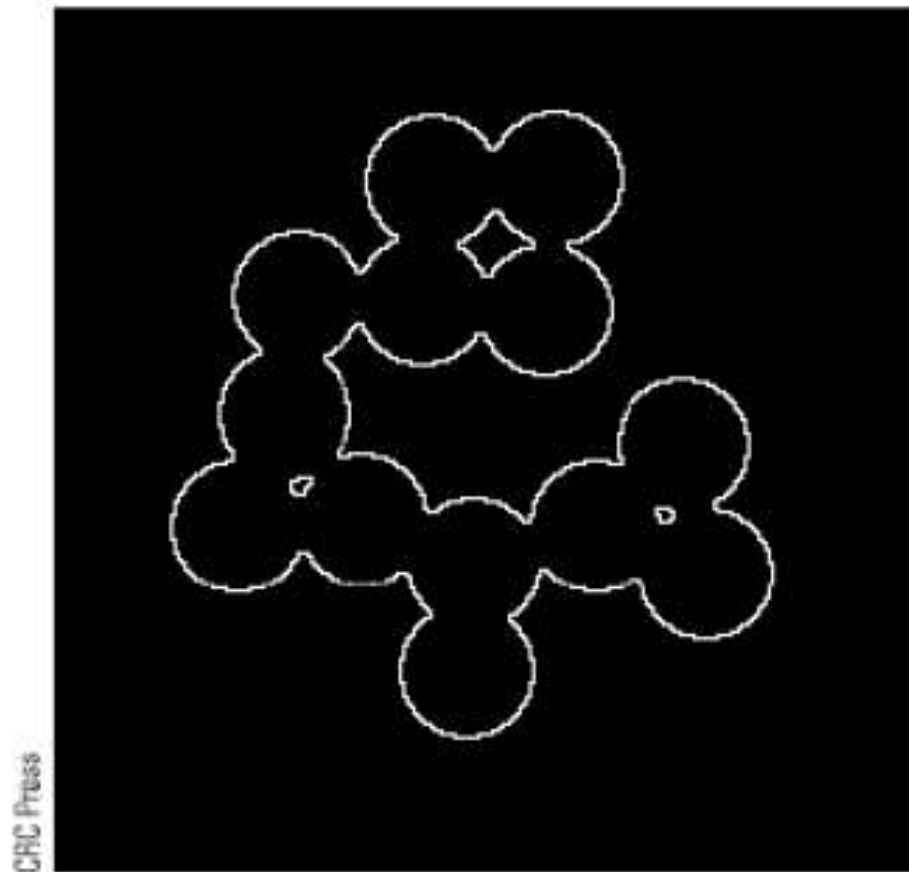


그림 11.3 circle 영상의 8-경계검출 결과

# 제 11장 영상의 위상기하적 처리

## 11.7 거리(Distance)

격자모양에서 2점  $x$ 와  $y$ 사이의 거리를 측정하는 함수의 정의가 필요하다. 거리함수  $d(x,y)$ 는 다음을 만족하는 경우에 metric(미터법)이라고 한다.

1.  $d(x, y) = d(y, x)$  (symmetry).
2.  $d(x, y) \geq 0$  and  $d(x, y) = 0$  if and only if  $x = y$  (positivity).
3.  $d(x, y) + d(y, z) \leq d(x, z)$  (the triangle inequality).

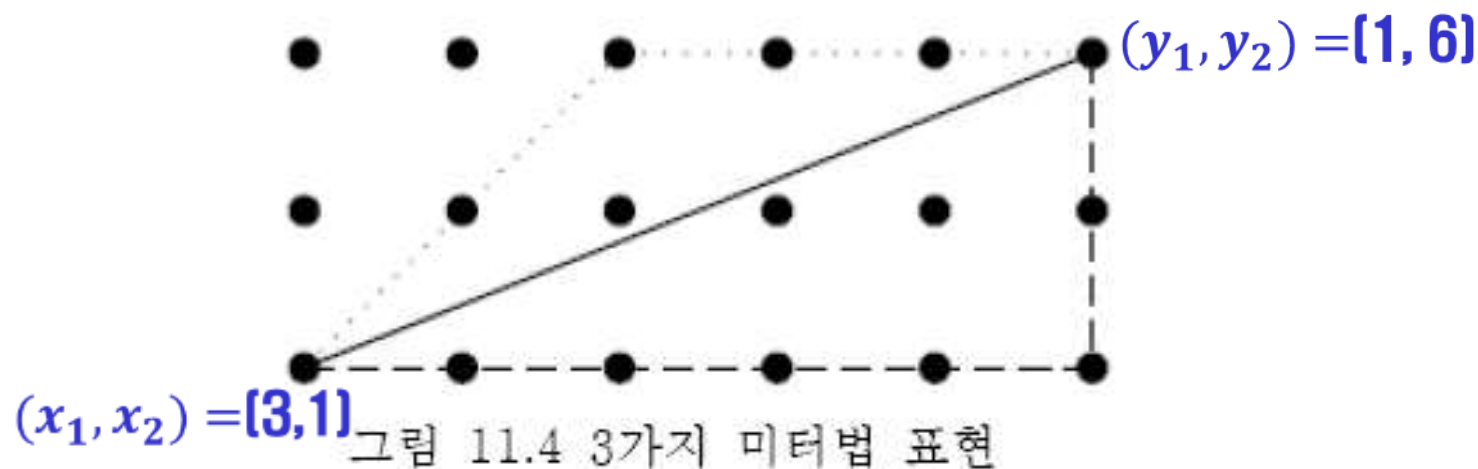
metric 표준 거리는  $x=(x_1,x_2)$  와  $y=(y_1,y_2)$ 인 경우 Euclidian distance 로 나타내고, 아래와 같이 표현된다.

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

$d$   $[y_1, y_2]$   
 $[x_1, x_2]$  픽셀 좌표

이것은  $x$ 와  $y$ 사이의 직선의 길이이다. 위 1과 2는 미터법에 의해 쉽게 만족됨을 알 수 있고, 이것은 항상 양(positive)이며,  $x_1=y_1$ 과  $x_2=y_2$  일 때만, 즉  $x=y$ 일 때 0이다. 위 3번의 성질은 쉽게 증명할 수 있는데 3개의 점  $x,y,z$ 가 주어지면  $x$ 에서  $z$ 까지 가는데  $y$ 를 통해서 가는 것보다 직접 가는 편이 가깝다.

## 제 11장 영상의 위상기하적 처리



그러나 격자에 제한 조건이 있으면 유클리드 거리는 응용이 불가능하다. 그림 11.4는 유클리드 미터법의 최단거리와 4-연결(쇄선) 및 8-연결(점선) 경로를 나타내었다. 이 그림에서 유클리드 거리는  $\sqrt{5^2 + 2^2} \approx 5.39$ 이고, 4-경로와 8-경로의 길이는 각각 7과 5이다.

4-경로 및 8-경로에 의한 거리의 미터법은 다음의 함수로 주어진다.

$$d_4(x, y) = |x_1 - y_1| + |x_2 - y_2| = |2| + |5| = 7$$

$$d_8(x, y) = \max\{|x_1 - y_1|, |x_2 - y_2|\} = \max\{|2|, |5|\} = 5$$

유클리드 거리와 마찬가지로 첫 2개의 성질은 삼각형 부등식으로 증명될 수 있고, 미터법  $d_4$ 는 가끔 taxicab metric으로 알려져 있다.



# 제 11장 영상의 위상기하적 처리

## 11.7.1 거리 변환

여러 응용에서 영역  $R$ 에서 모든 화소의 거리를 구할 필요가 있다. 거리를 구하는 것은 위에서 정의한 표준 유클리드 거리를 이용할 수 있다. 그러나 이것은  $R$ 에서  $(x,y)$ 의 거리를 계산하고,  $R$ 에서  $(x,y)$ 에서 화소들까지 모든 가능한 거리를 결정할 필요가 있으며 가장 작은 값을 택한다. 이것은 계산이 비효율적이다.  $R$ 이 크다면 많은 제곱의 평방근을 계산해야 한다. 이를 줄이기 위해 아래와 같이 최소거리를 구할 수 있다. 왜냐하면 제곱의 평방근 함수가 증가하기 때문이다.

$$md(x,y) = \sqrt{\min_{(p,q) \in R} ((x-p)^2 + (y-q)^2)},$$

영상 내 object 가 있고 그 영역을  
 $R$  이라고 할때  
 $R$  로부터 외부의 한점  $(x,y)$   
까지의 거리를 구한다.

여기서는 하나의 제곱과 평방근을 포함한다. 그러나 이 정의는 계산이 느리다. 계산할 연산의 수와 최소값을 구하는데 많은 시간이 소요된다.

거리 변환은 이러한 거리를 구하는데 계산적으로 효과적인 방법이다. 아래의 일련의 단계로 설명한다.



# 제 11장 영상의 위상기하적 처리

## 단계 1

영상에서 R로부터 각 화소  $(x,y)$ 까지의 거리를 나타내는 라벨  $d(x,y)$ 를 붙여라.  
R에서 각 화소가 0인 라벨을 붙여서 시작하고, R에 존재하지 않으면  $\infty$ 를 붙여라.

## 단계 2

영상을 통해 1화소씩 이동한다. 각 화소  $(x,y)$ 에 대하여 아래와 같이  $\infty+1=\infty$ 를 이용하여 해당하는 라벨로 치환 한다.

$$\min\{d(x,y), d(x+1,y)+1, d(x-1,y)+1, d(x,y-1)+1, d(x,y+1)+1\}$$

## 단계 3

모든 화소들이  $\infty$  값으로 변환될 때까지 단계 2를 반복하라.

가 아닌

	1	
1	0	1
	1	

단계 2의 몇 가지 예를 보기위해 아래와 같이 이들의 이웃 화소들을 가진다고 가정하자.

$\infty$	$\infty$	$\infty$
2	$\infty$	$\infty$
$\infty$	3	$\infty$

## 제 11장 영상의 위상기하적 처리

우리는 중심 화소(그 라벨이 변화하는 것)와 4개의 화소들, 위, 아래, 왼쪽 및 오른쪽 화소에만 관심을 가진다. 이들 4개의 화소들에 대하여 아래와 같이 1을 더한다.

$$\begin{array}{ccc} & \infty & \\ 3 & \infty & \infty \\ & 4 & \end{array}$$

이들 5개의 최소값은 3이고, 따라서 이것은 중심 화소에 대하여 새로운 라벨이다. 여기서 또 아래와 같은 이웃을 가진다고 가정하자.

$$\begin{array}{ccc} 2 & \infty & \infty \\ \infty & \infty & \infty \\ 3 & \infty & 5 \end{array}$$

다시, 4개의 각 화소, 위, 아래, 좌, 우측에 1을 더하고 중심 화소 값은 아래와 같이 그대로 유지한다.

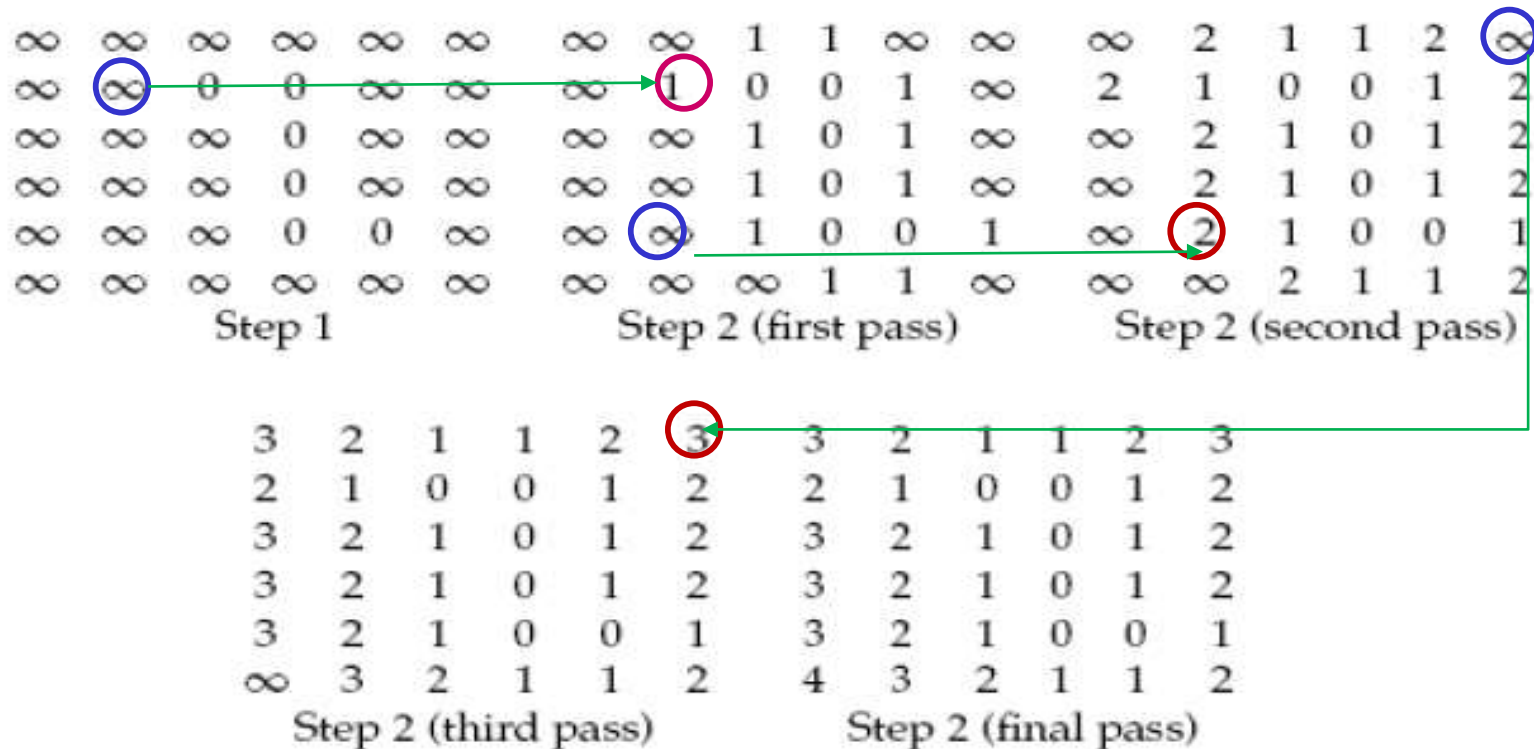
$$\begin{array}{ccc} & \infty & \\ \infty & \infty & \infty \\ & \infty & \end{array}$$

이들 값들의 최소값은  $\infty$ 이고, 이 단계에서 화소의 라벨은 변하지 않는다.

# 제 11장 영상의 위상기하적 처리

단계 1의 처리 후에 영상의 라벨이 아래와 같다고 가정하자.

Infinity padding



이 단계에서 정지한다. 왜냐하면 모든 라벨 값들이 유한 값이기 때문이다.

얼핏 보기에 주어진 거리 값들은 사실 실제 거리의 좋은 근사가 되지 못한다는 것을 알 수 있다. 보다 정확성을 기하기 위하여 위 변환을 일반화할 필요가 있다. 이의 한 가지 방법은 마스크의 개념을 이용하는 것이다. 위 내용에 사용한 마스크는 아래와 같다.

# 제 11장 영상의 위상기하적 처리

1  
1 0 1  
1

변환에서 단계 2는 이웃하는 화소들에 대응하는 마스크 요소의 값을 더하고, 최소값을 선택한다. 간단한 연산으로 정확성을 얻기 위해 마스크는 일반적으로 정수의 값들로 구성하지만, 최종 결과는 스케일링을 요구한다.

아래의 마스크를 생각하자.

4 3 4  
3 0 3  
4 3 4



inf	11	10
inf	inf	9
inf	inf	11

위의 영상을 적용하면 단계 1은 위와 같고, 단계 2에서 아래와 같이 처리하여 모든 화소의 값이 유한이면 정지한다.

$\infty$	4	3	3	4	$\infty$	7	4	3	3	4	7	7	4	3	3	4	7
$\infty$	3	0	0	3	$\infty$	6	3	0	0	3	6	6	3	0	0	3	6
$\infty$	4	3	0	3	$\infty$	7	4	3	0	3	6	7	4	3	0	3	6
$\infty$	$\infty$	3	0	3	$\infty$	8	6	3	0	3	6	8	6	3	0	3	6
$\infty$	$\infty$	3	0	0	3	$\infty$	6	3	0	0	3	9	6	3	0	0	3
$\infty$	$\infty$	4	3	3	4	$\infty$	7	4	3	3	4	10	7	4	3	3	4
Step 2 (first pass)						Step 2 (second)						Step 2 (third)					


## 제 11장 영상의 위상기하적 처리

여기서 모든 화소 값들을 3으로 나누면 아래와 같다.

2.3	1.3	1	1	1.3	2.3
2	1	0	0	1	2
2.3	1.3	1	0	1	2
2.7	2	1	0	1	2
3	2	1	0	0	1
3.3	2.3	1.3	1	1	1.3

이들 값들은 처음에 설명한 변환으로 얻어진 것보다 훨씬 유클리드 거리에 가깝다.  
아래의 마스크를 이용하면 더욱 정확한 특성을 얻을 수 있다.

	11		11	
11	7	5	7	11
	5	0	5	
11	7	5	7	11
	11		11	



이 결과를 5로 나누어서 최종 결과를 아래와 같이 얻을 수 있다.

## 제 11장 영상의 위상기하적 처리

11	7	5	5	7	11
10	5	0	0	5	10
11	7	5	0	5	10
14	10	5	0	5	7
16	10	5	0	0	5
16	11	7	5	5	7

Result of transform

2.2	1.4	1	1	1.4	2.2
2	1	0	0	1	2
2.2	1.4	1	0	1	2
2.8	2	1	0	1	1.4
3.2	2	1	0	0	1
3.2	2.2	1.4	1	1	1.4

After division by 5

무한대가 되기 전에 많은 통과(생략)를 필요로 한다. 보다 빠른 방법은 2개의 통과를 요구하는데, 첫 통과는 영상의 좌측 위에서 출발하여 우측 아래로 이동한다. 2번째 통과는 영상의 우측 아래에서 출발하여 위에서 좌로 이동하면서 아래쪽에서 위쪽으로 이동한다. 이 방법에서 영상을 반으로 나누어서 그 중 하나는 좌측 위에서 출발하는 부분(첫 번째 통과)의 값들만을 조사하고, 나머지 반은 우측 아래에서 출발하는 부분(2번째 통과)의 값들만을 조사한다.

마스크의 싸은 그림 11.5와 같고, 굵은 선은 원래의 마스크가 2개의 반으로 나누어지는 모양을 나타낸다.



## 제 11장 영상의 위상기하적 처리

Mask 1

	1	
1	0	

Forward mask

	0	1
	1	

Backward mask

Mask 2

4	3	4
3	0	

Forward mask

	0	3
4	3	4

Backward mask

그림 11.5 2개의 통과(pass) 거리 변환p 대한 마스크의 쌍

## 제 10장 영상의 형태적 처리

우리는 이들 마스크를 다음과 같이 적용한다. 먼저 0이 아닌 값들을 가지는 영상(공간필터링과 같은)의 둘레에 영상의 에지에서 마스크들이 처리할 값들을 가지도록 마스크를 적용한다. 순방향 pass에 대하여 위치  $(i,j)$ 에서 각 화소마다 그 이웃들을 순방향 마스크에서 주어진 값들을 더하라. 최소값을 취하여 현재의 라벨을 치환하라. 역방향 마스크를 이용하고, 오른쪽 아래에서 출발하는 것을 제외하고는 역방향 pass에서도 같은 과정으로 처한다.

Mask 3

	11		11	
11	7	5	7	11
	5	0		

Forward mask

		0	5	
11	7	5	7	11
	11		11	

Backward mask



## 제 11장 영상의 위상기하적 처리

마스크를 이용하고, 오른쪽 아래에서 출발하는 것을 제외하고는 역방향 pass에서도 같은 과정으로 처한다.

영상에 순방향마스크 1과 2를 적용하면 순방향 pass의 결과는 아래와 같다.

$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
$\infty$	$\infty$	0	0	1	2
$\infty$	$\infty$	1	0	1	2
$\infty$	$\infty$	2	0	1	2
$\infty$	$\infty$	3	0	0	1
$\infty$	$\infty$	4	1	1	2

Use of mask 1

$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
$\infty$	$\infty$	0	0	3	6
$\infty$	4	3	0	3	6
8	7	5	0	3	6
11	8	4	0	0	3
12	8	4	3	3	4

Use of mask 2

역방향마스크의 적용 후에 위와 같이 거리변환을 얻는다.

## 제 11장 영상의 위상기하적 처리

### MATLAB에서의 구현

우리는 위에서와 같이 2번째 방법을 적용하여 거리변환을 쉽게 수행할 수 있다. 함수는 아래와 같이 변환을 구현한다.

1. 마스크의 사이즈를 이용하여 적당히 0으로 영상을 채운다.
2. 각 0을  $\infty$ 로 바꾸고, 각 1을 0으로 바꾼다.
3. 순방향 및 역방향마스크를 만든다.
4. 순방향 pass를 실행한다. 각 라벨을 그 이웃들의 최소값과 순방향마스크를 더한 값으로 치환한다.
5. 역방향 pass를 실행한다. 각 라벨을 그 이웃들의 최소값과 역방향마스크를 더한 값으로 치환한다.

이들 각 단계를 다음과 같이 구분하여 생각하자.

1. 영상은  $r \times c$ 이고, 마스크는  $r_m \times c_m$ 이라고 가정한다. 여기서, 2개의 마스크는 차수가 홀수이다. 영상의 각 측면에다 열의 수를  $(c_m - 1)/2$ 과 같게 더하고, 영상의 위와 아래의 양측에다 행의 수를  $(r_m - 1)/2$ 과 같게 더할 필요가 있다. 바꾸어 말하면 더 큰 배열의 사이즈  $(r + r_m - 1) \times (c + c_m - 1)$ 에 영상을 아래와 같이 덮을 수 있다.

## 제 11장 영상의 위상기하적 처리

```
>> [r,c]=size(image);  
>> [mr,mc]=size(mask);  
>> nr=(mr-1)/2;  
>> nc=(mc-1)/2;  
>> image2=zeros(r+mr-1,c+mc-1);  
>> image2(nr+1:r+nr,nc+1:c+nc)=image; Filtering을 위한 padding
```

2. 이것은 find 함수를 이용하여 쉽게 처리할 수 있다. 먼저, 모든 0들을  $\infty$ 로 바꾸어라. 다음에 모든 1들을 0들로 아래와 같이 바꾸어라.

```
>> image2(find(image2)==0)=Inf; Background 화소는 Inf로  
>> image2(find(image2)==1)=0; Object 화소는 0으로 채운다.
```

## 제 11장 영상의 위상기하적 처리

3. 순방향 마스크가 주어진다고 가정하자. 우리는 그림 11.5에 보인 마스크에서 모든 빈자리(blank entry)에  $\infty$ 를 넣어서 가장 간단하게 이를 처리할 수 있다. 즉, 이것은 이들 위치에 있는 화소들이 최종의 최소화에 전혀 영향을 미치지 않는다는 것을 의미한다. 역방향 마스크는 2회의  $90^\circ$  회전으로 얻을 수 있다. 예를 들면, 아래와 같다.

```
>> mask1=[Inf 1 Inf;1 0 Inf;Inf Inf Inf];  
>> backmask=rot90(rot90(mask1));
```

Mask 1

Inf	1	Inf
1	0	Inf
Inf	Inf	Inf

4. 우리는 아래와 같은 삼중 루프로서 순방향 마스크를 구현할 수 있다.

```
>> for i=nr+1:r+nr,  
    for j=nc+1:c+nc,  
        image2(i,j)=min(min(image2(i-nr:i+nr,j-nc:j+nc)+mask));  
    end;  
end;
```

## 제 11장 영상의 위상기하적 처리

5. 역방향 패스는 아래와 유사하게 처리할 수 있다.

```
>> for i=r+nr:-1:nr+1,  
    for j=c+nc:-1:nc+1,  
        image2(i,j)=min(min(image2(i-nr:i+nr,j-nc:j+nc)+backmask));  
    end;  
end;
```

완전한 함수는 그림 11.6에 보였다.

## 제 11장 영상의 위상기하적 처리

```
function res=disttrans(image,mask)
%
% This function implements the distance transform by
% applying MASK to IMAGE, using the two step algorithm
% with "forward" and "backwards" masks.
backmask=rot90(rot90(mask));
[mr,mc]=size(mask);
if ((floor(mr/2)==ceil(mr/2)) | (floor(mc/2)==ceil(mc/2))) then
    error('The mask must have odd dimensions.')
end;
[r,c]=size(image);
nr=(mr-1)/2;
nc=(mc-1)/2;
image2=zeros(r+mr-1,c+mc-1);
image2(nr+1:r+nr,nc+1:c+nc)=image;
%
% This is the first step; replacing R values with 0 and other
% values with infinity
%
image2(find(image2==0))=Inf;
image2(find(image2==1))=0;
%
```

## 제 11장 영상의 위상기하적 처리

```
% Forward pass
%
for i=nr+1:r+nr,
    for j=nc+1:c+nc,
        image2(i,j)=min(min(image2(i-nr:i+nr,j-nc:j+nc)+mask));
    end;
end;
%
% Backward pass
%
for i=r+nr:-1:nr+1,
    for j=c+nc:-1:nc+1,
        image2(i,j)=min(min(image2(i-nr:i+nr,j-nc:j+nc)+backmask));
    end;
end;

res=image2(nr+1:r+nr,nc+1:c+nc);
```

그림 11.6 거리변환을 계산하기 위한 함수



## 제 11장 영상의 위상기하적 처리

이를 시도해보기로 한다. 먼저 영상과 아래와 같이 3개의 마스크를 만든다.

```
>> im=[0 0 0 0 0 0;...  
0 0 1 1 0 0;...  
0 0 0 1 0 0;...  
0 0 0 1 0 0;...  
0 0 0 1 1 0;...  
0 0 0 0 0 0]
```

im =

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

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

```
>> mask1=[Inf 1 Inf;1 0 Inf;Inf Inf Inf]
```

mask1 =

Inf	1	Inf
1	0	Inf
Inf	Inf	Inf



## 제 11장 영상의 위상기하적 처리

```
>> mask2=[4 3 4;3 0 Inf;Inf Inf Inf]
```

```
mask2 =
```

4	3	4
3	0	Inf
Inf	Inf	Inf

```
>> mask3=[Inf 11 Inf 11 Inf;...
```

```
11 7 5 7 11;...
```

```
Inf 5 0 Inf Inf;...
```

```
Inf Inf Inf Inf Inf;...
```

```
Inf Inf Inf Inf Inf]
```

```
mask3 =
```

Inf	11	Inf	11	Inf
11	7	5	7	11
Inf	5	0	Inf	Inf
Inf	Inf	Inf	Inf	Inf
Inf	Inf	Inf	Inf	Inf

## 제 11장 영상의 위상기하적 처리

여기서 우리는 아래와 같이 변환을 적용할 수 있다.

```
>> disttrans(im,mask1)
```

```
ans =
```

3	2	1	1	2	3
2	1	0	0	1	2
3	2	1	0	1	2
4	3	2	1	1	2

```
>> disttrans(im,mask2)
```

```
ans =
```

7	4	3	3	4	7
6	3	0	0	3	6
7	4	3	0	3	6
8	6	3	0	3	4
9	6	3	0	0	3
10	7	4	3	3	4

## 제 11장 영상의 위상기하적 처리

```
>> disttrans(im,mask3)
```

```
ans =
```

11	7	5	5	7	11
10	5	0	0	5	10
11	7	5	0	5	10
14	10	5	0	5	7
15	10	5	0	0	5
16	11	7	5	5	7

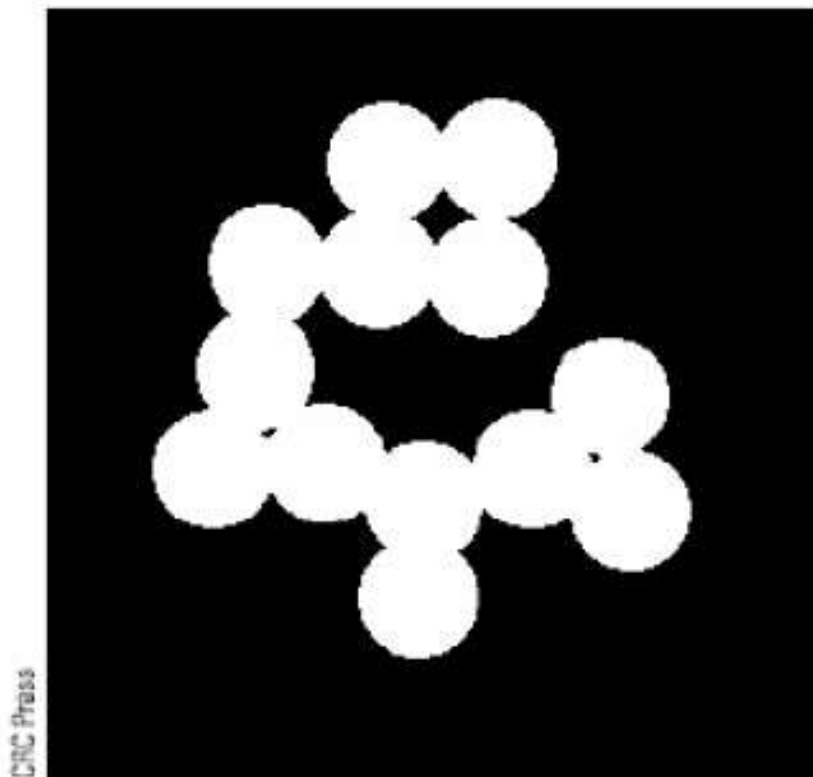
여기서 마스크2와 마스크3을 이용한 변환의 결과들은 실제의 거리에 대한 근사치를 얻기 위해 적당한 값들로 나누어야 한다.

## 제 11장 영상의 위상기하적 처리

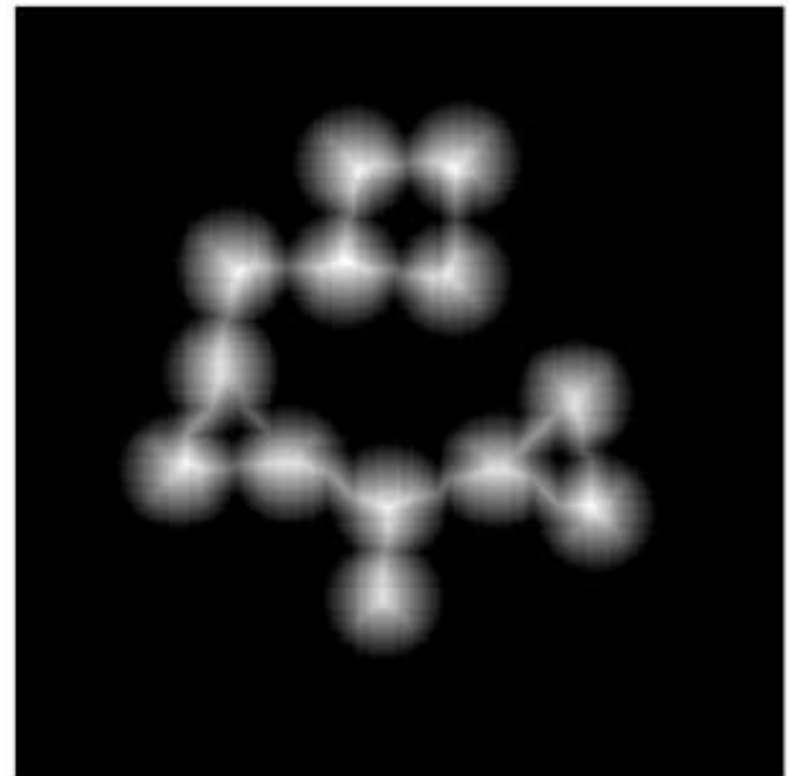
물론, 아래와 같이 circles.tif와 같은 큰 영상에 대한 거리변환을 적용할 수 있다.

```
>> c=~imread('circles.tif');  
>> imshow(c)  
>> cd=disttrans(c,mask1);  
>> figure,imshow(mat2gray(cd))
```

우리는 흰색 배경에 흑색 원을 만들기 위해 원(circle) 영상을 반전시킨다. 이것은 변환이 원래(original) 영상의 내부로 향하는 거리를 구하는 것을 의미한다. 이렇게 하는 이유는 결과 영상을 만들기가 더욱 쉽기 때문이다. 원(circle) 영상을 그림 11.7 (a)에 보였고, 거리변환은 그림 11.7 (b)에 보였다.



(a)



(b)

그림 11.7 거리변환의 예 (a) 원(circle) 영상 (b) 거리변환 결과



## 성분 라벨링 연습

