디지털영상처리II (2021학년도 2학기)

10주: 영상분할(2)

김 남 규 (ngkim@deu.ac.kr)

디지털영상처리II (2021학년도 2학기)

10주: 영상분할(2)

1 연결 성분 분석

김 남 규 (ngkim@deu.ac.kr)

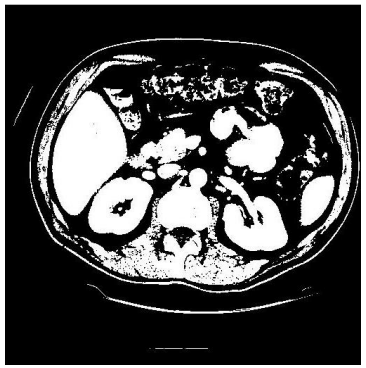
연결 성분 분석(Connected Component Analysis)

- CCA, connected component labeling, region labeling/extraction, blob extraction/discovery …
- 이진 영상에서 연결된 화소군을 분석하고 식별하는 과정: 연결된 화소 → 이웃 화소(neighbors) 분석
- 연결화소군 레이블링(connected component labeling): 이진 영상(binary image)으로 부터 배경은 0으로 연결된 화소군/성분은 각기 다른 정수의 값을 갖는 레이블이 지정된 이미지(a labeled image)를 얻는 과정

	N	
W		E
	S	

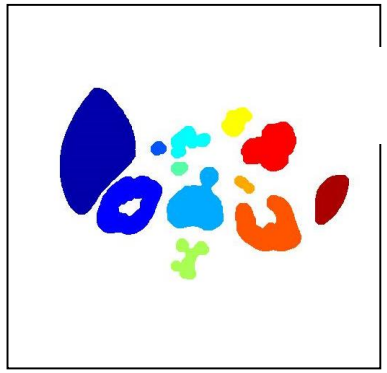
NW	N	NE
W		E
SW	S	SE

Original image



Thresholded image

After morphology



Connected components

4-이웃

8-이웃

CCA 1: 재귀 추적(recursive tracking) 알고리즘

```
Compute the connected components of a binary image.
B is the original binary image.
LB will be the labeled connected component image.


    procedure recursive_connected_components(B, LB);
    {
    LB := negate(B);
    label := 0;
    find_components(LB, label);
    print(LB);
    }

    procedure find_components(LB, label);
    {
    for L := 0 to MaxRow
       for P := 0 to MaxCol
          if LB[L,P] == -1 then
             {
             label := label + 1;
             search(LB, label, L, P);
             }
    }

    procedure search(LB, label, L, P);
    {
    LB[L,P] := label;
    Nset := neighbors(L, P);
    for each (L',P') in Nset
       {
       if LB[L',P'] == -1
       then search(LB, label, L', P');
       }
    }
```

- 이진영상의 1값을 모두 -1를 바꾼다: negate
- -1 화소값을 찾으면 새로운 레이블로 지정하고, -1 값을 갖는 주변을 찾기 위한 search 절차를 수행한다.
- 이웃의 모든 -1 값에 대해 재귀적으로 search를 수행한다

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

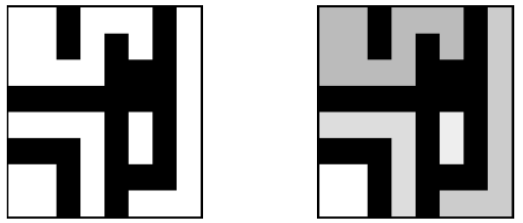
a) binary image

1	1	0	1	1	1	0	2
1	1	0	1	0	1	0	2
1	1	1	1	0	0	0	2
0	0	0	0	0	0	0	2
3	3	3	3	0	4	0	2
0	0	0	3	0	4	0	2
5	5	0	3	0	0	0	2
5	5	0	3	0	2	2	2

b) connected components labeling

1	2	3
4		5
6	7	8

8-이웃



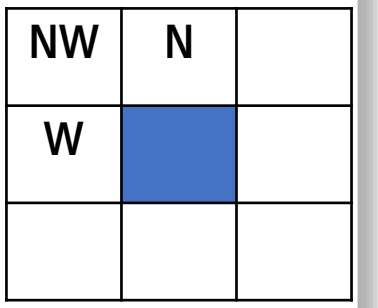
c) binary image and labeling, expanded for viewing

CCA 2: 행단위(row-by-row) 알고리즘

```
Compute the connected components of a binary image.
B is the original binary image.
LB will be the labeled connected component image.


    procedure classical_with_union-find(B,LB);
    {
    "Initialize structures."
    initialize();
    "Pass 1 assigns initial labels to each row L of the image."
    for L := 0 to MaxRow
      {
      "Initialize all labels on line L to zero"
      for P := 0 to MaxCol
        LB[L,P] := 0;
      "Process line L."
      for P := 0 to MaxCol
        if B[L,P] == 1 then
          {
          A := prior_neighbors(L,P);
          if isempty(A)
          then { M := label; label := label + 1; };
          else M := min(labels(A));
          LB[L,P] := M;
          for X in labels(A) and X <> M
            union(M, X, PARENT);
          }
      }
    "Pass 2 replaces Pass 1 labels with equivalence class labels."
    for L := 0 to MaxRow
      for P := 0 to MaxCol
        if B[L,P] == 1
        then LB[L,P] := find(LB[L,P],PARENT);
    } ;
```

- 현재 화소를 중심으로 이전 이웃(prior neighbors) 의 연관성을 보고 레이블링을 전파한다.



- 같은 화소에 복수 레이블링 처리
  - 가장 작은 레이블값 정의 (min)
  - 복수 레이블값은 동일 클래스 구조체에 저장 (union)
- 동일 클래스로 정의된 화소를 하나로 통일한다. (find)
- 동일 클래스 구조체
  = Union-find 데이터 구조체

Union-find 구조체 연산

Construct the union of two sets.
X is the label of the first set.
Y is the label of the second set.
PARENT is the array containing the union-find data structure.

```
procedure union(X, Y, PARENT);
{
j := X;
k := Y;
while PARENT[j] <> 0
   j := PARENT[j];
while PARENT[k] <> 0
   k := PARENT[k];
if j <> k then PARENT[k] := j;
}
```

Find the parent label of a set.
X is a label of the set.
PARENT is the array containing the union-find data structure.

```
procedure find(X, PARENT);
{
j := X;
while PARENT[j] <> 0
   j := PARENT[j];
return(j);
}
```

Union-find 구조체 연산 예

1) 초기화: P[i] = i

i	1	2	3	4	5	6	7	8
P[i]	1	2	3	4	5	6	7	8

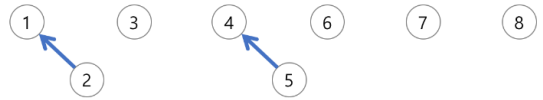
부모 노드 번호

2) union(1, 2) : find(1) = 1, find(2) = 2 -> P[2] = 1

i	1	2	3	4	5	6	7	8
P[i]	1	1	3	4	5	6	7	8

부모 노드 번호

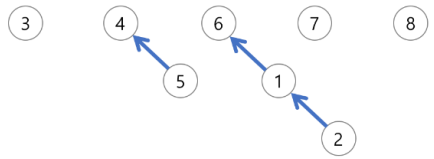
3) union(4, 5) : find(4) = 4, find(5) = 5 -> P[5] = 4



i	1	2	3	4	5	6	7	8
P[i]	1	1	3	4	4	6	7	8

부모 노드 번호

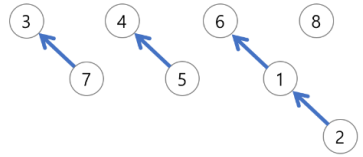
4) union(6, 1) : find(6) = 6, find(1) = 1 -> P[1] = 6



i	1	2	3	4	5	6	7	8
P[i]	6	1	3	4	4	6	7	8

부모 노드 번호

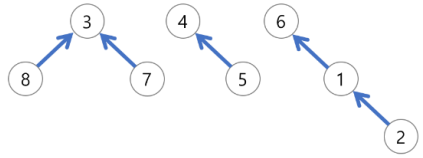
5) union(3, 7) : find(3) = 3, find(7) = 7 -> P[7] = 3



i	1	2	3	4	5	6	7	8
P[i]	6	1	3	4	4	6	3	8

부모 노드 번호

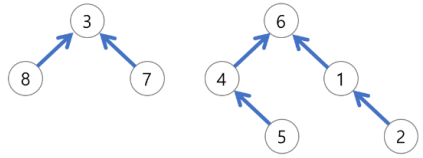
6) union(7, 8) : find(7) = 3, find(8) = 8 -> P[8] = 3



i	1	2	3	4	5	6	7	8
P[i]	6	1	3	4	4	6	3	3

부모 노드 번호

7) union(2, 5) : find(2) = 6, find(5) = 4 -> P[4] = 6



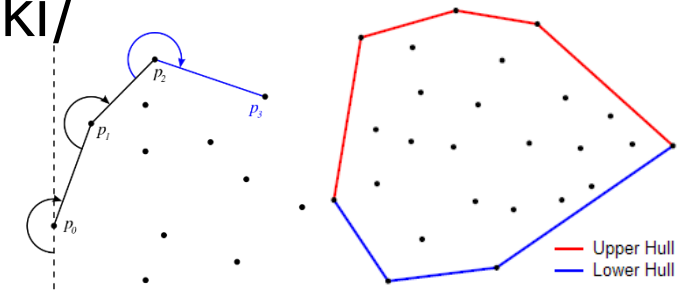
i	1	2	3	4	5	6	7	8
P[i]	6	1	3	6	4	6	3	3

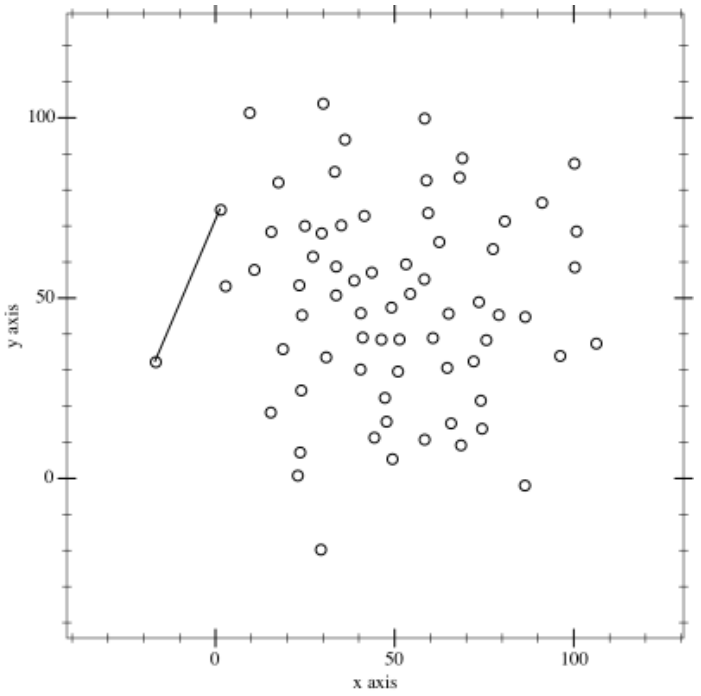
부모 노드 번호

- Node - Parent 2D 배열
- 링크는 트리구조 생성
  : union 연산
- 최상위 부모 탐색
  : find 연산

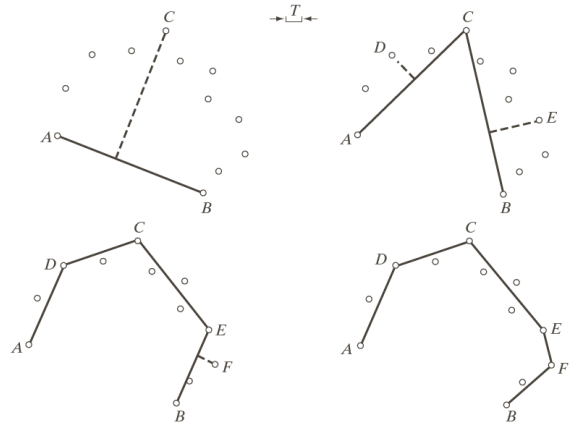
경계 검출(boundary detection)

- 볼록 껍질(convex hull)
  - 주어진 점을 포함하는 가장 작은 볼록 다각형
  - 2D Andrew's monotone chain convex hull algorithm: (https://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/Convex_hull/Monotone_chain)





- 다각형 근사(polygon approximation)
  - 관심 영역의 모습을 표현
  - 초기: min_x, max_y
  - 두 점과 다른 점 간의 거리 측정



디지털영상처리II (2021학년도 2학기)

10주차 : 끝

본 강의 자료의 내용 및 그림은 아래 책으로부터 발췌 되었음

- 파이썬으로 배우는 영상처리, Sandipan Dey 지음, 정성환 , 조보호 , 배종욱 옮김, 도서출판 홍릉, 2020년
- Digital Image Processing, 4th Ed., Rafael C . Gonzalez, Richard E. Woods 지음, Pearson, 2018년
- 컴퓨터 비전(Computer Vision) 기본 개념부터 최신 모바일 응용 예까지 IT CookBook, 오일석 지음, 한빛아카데미, 2014년

김 남 규 (ngkim@deu.ac.kr)