영상 분할

미디어기술콘텐츠학과
강호철

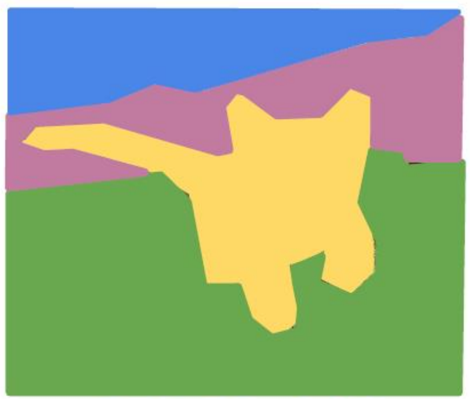
Computer Vision Task

Classification

CAT

No spatial extent

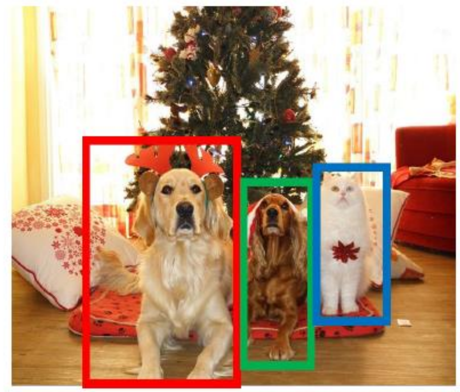
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



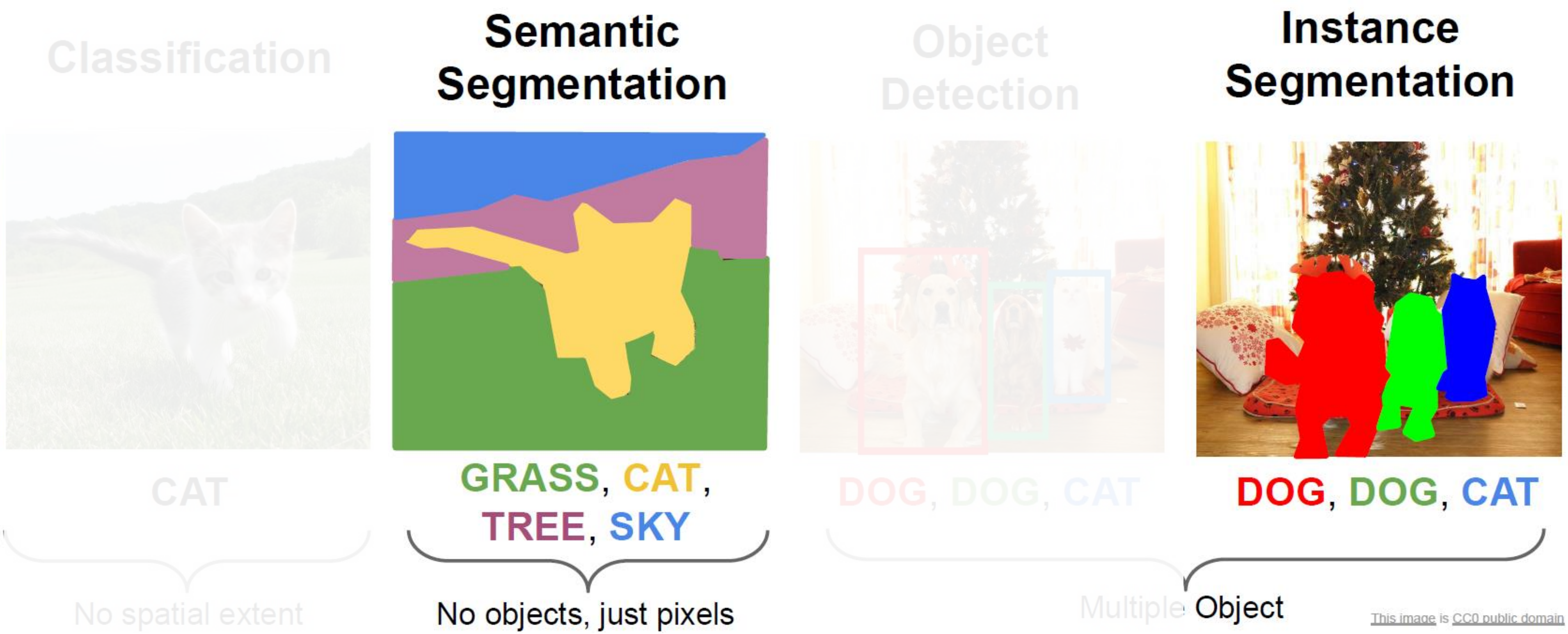
DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

Multiple Object

Image Segmentation



Deep learning is necessary

Image Segmentation

Semantic Segmentation

Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels

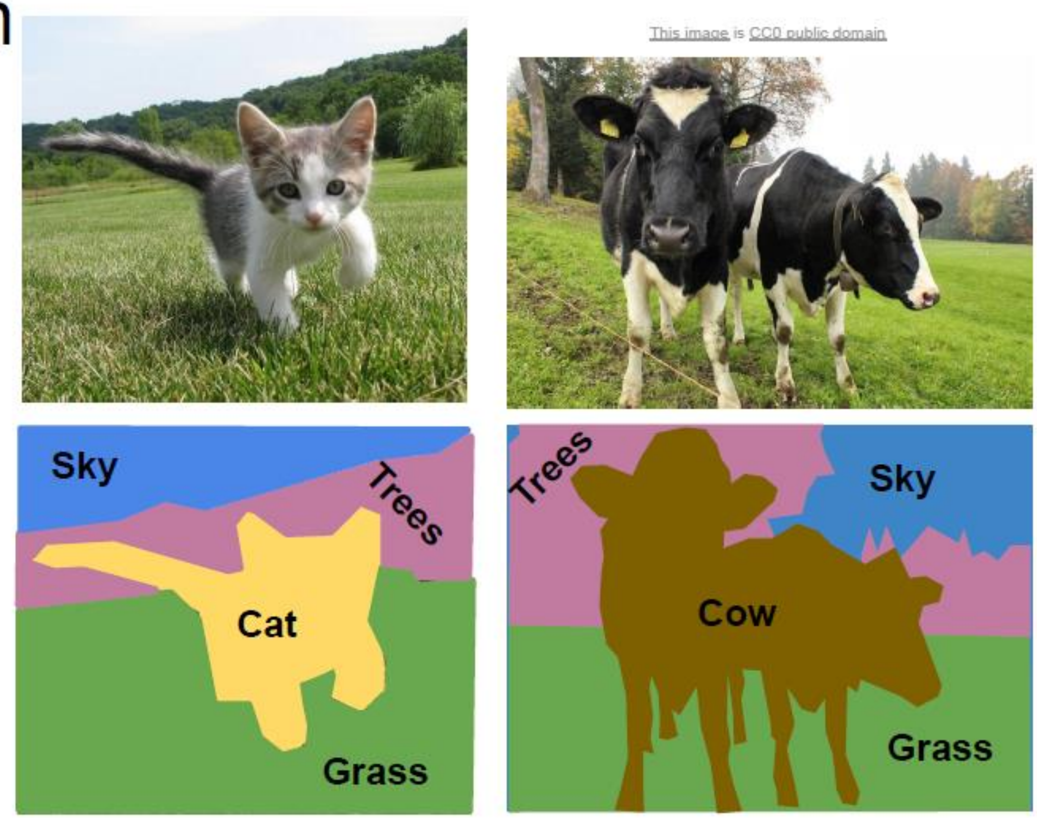
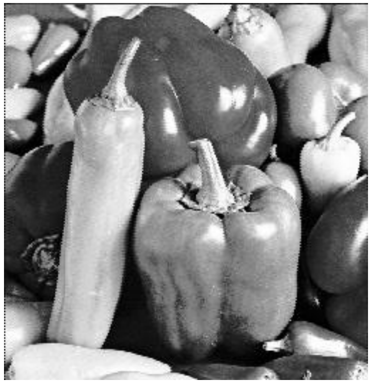


Image Segmentation

- 개념
  - 입력 영상에서 픽셀 단위로 배경 및 객체를 구하는 작업
  - 영상의 밝기 값, 컬러, 텍스쳐, gradient 등 정보를 이용

- 방법
  - Thresholding
  - Region growing
  - Watershed
  - Meanshift
  - Active contour model
  - K-Means clustering
  - …



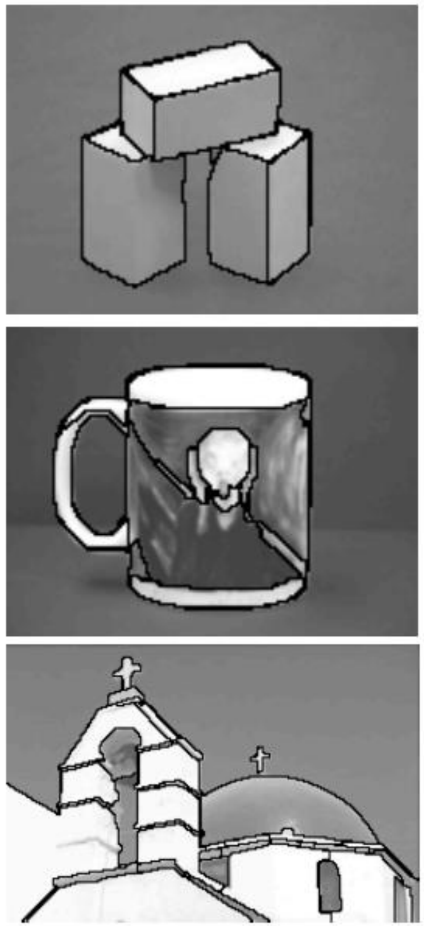
Objected Image

Segmented Image

Image Segmentation

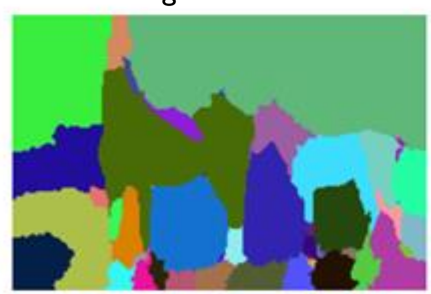
Objected Image

Segmented Image



Image

Segmentation



Region Merging

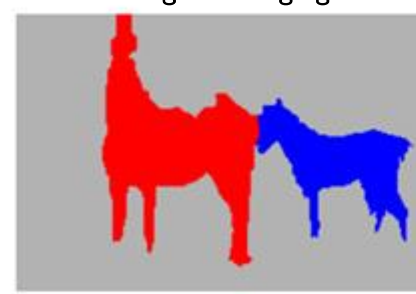


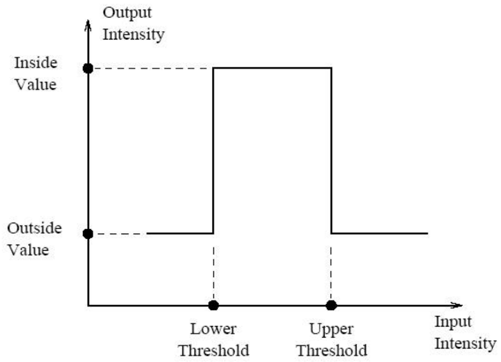
Image Segmentation

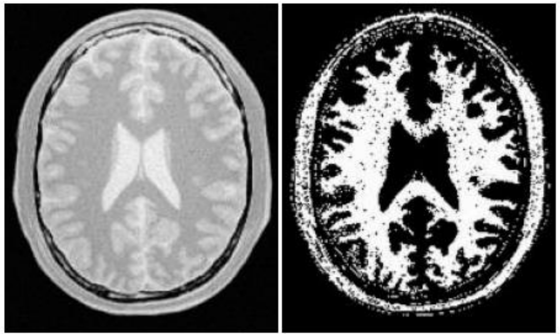
- Edge-based method
  - Edge detectors
  - Classical / geodesic active contours
  - Active shape models
- Region-based method
  - Region growing
  - Active contour models without edge
  - Statistical/clustering techniques
  - MRF-based techniques
  - Active appearance models
- Hybrid method
  - Other active contour models
  - Graph-based techniques
  - Level set methods

Thresholding

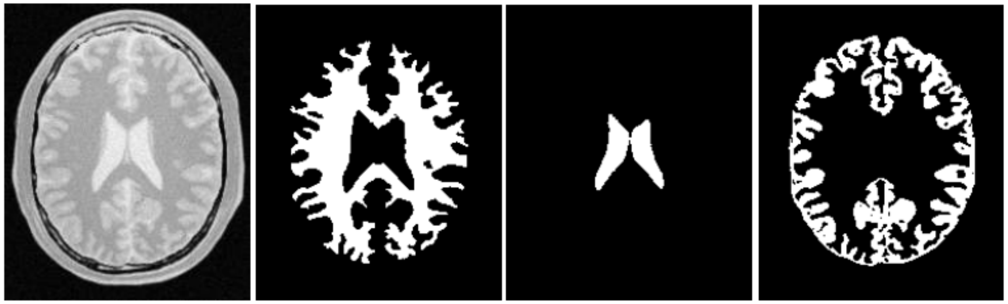
- Connected threshold
  - Evaluate intensity value inside a specific interval

$$I(\mathbf{X}) \in [\text{lower}, \text{upper}]$$





- Results for various interval



Thresholding

- Otsu threshold
  - Minimize the error of misclassification
  - Find a threshold that classifies the image into two clusters
  - Minimize the within class variance
  - Maximize the between class variance
  - The weighted within class variance

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

  - The class probabilities

$$q_1(t) = \sum_{i=1}^{t} P(i) \qquad q_2(t) = \sum_{i=t+1}^{I} P(i)$$

Thresholding

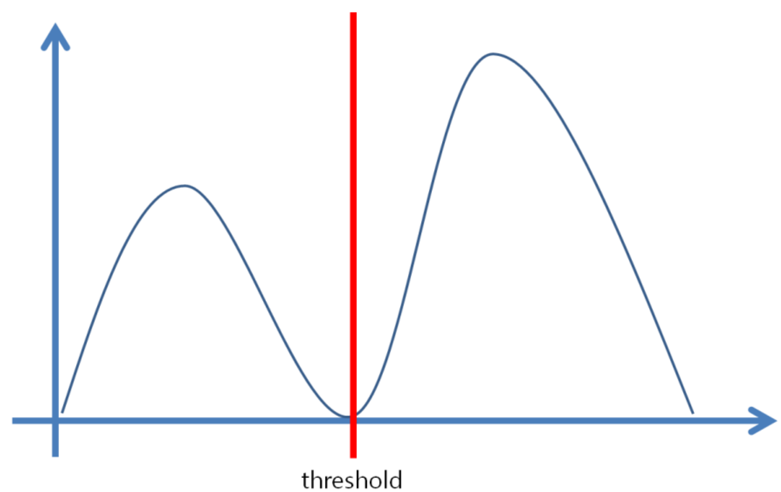
- Otsu threshold
  - Class means

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \qquad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

  - Individual class variances

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \qquad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

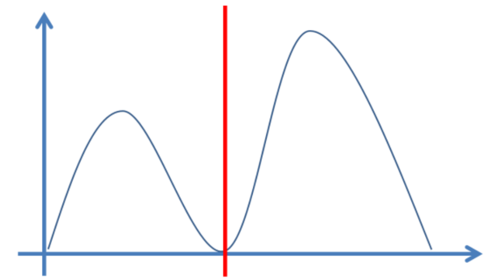
  - Minimize $\sigma_w^2(t)$



Thresholding

- Otsu threshold
  - Class means

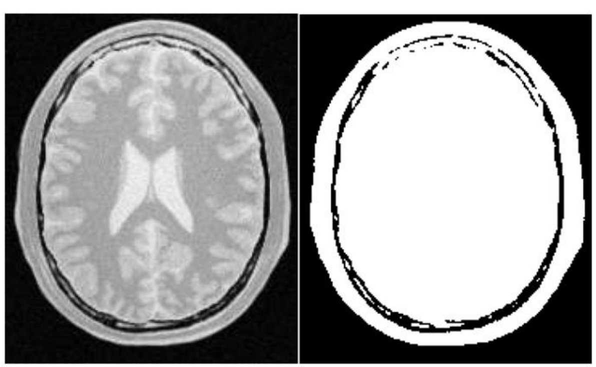
$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \qquad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$



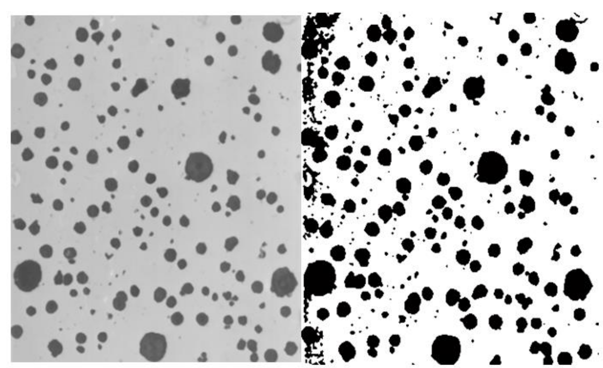
  - Individual class variances

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \qquad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

  - Minimize $\sigma_w^2(t)$



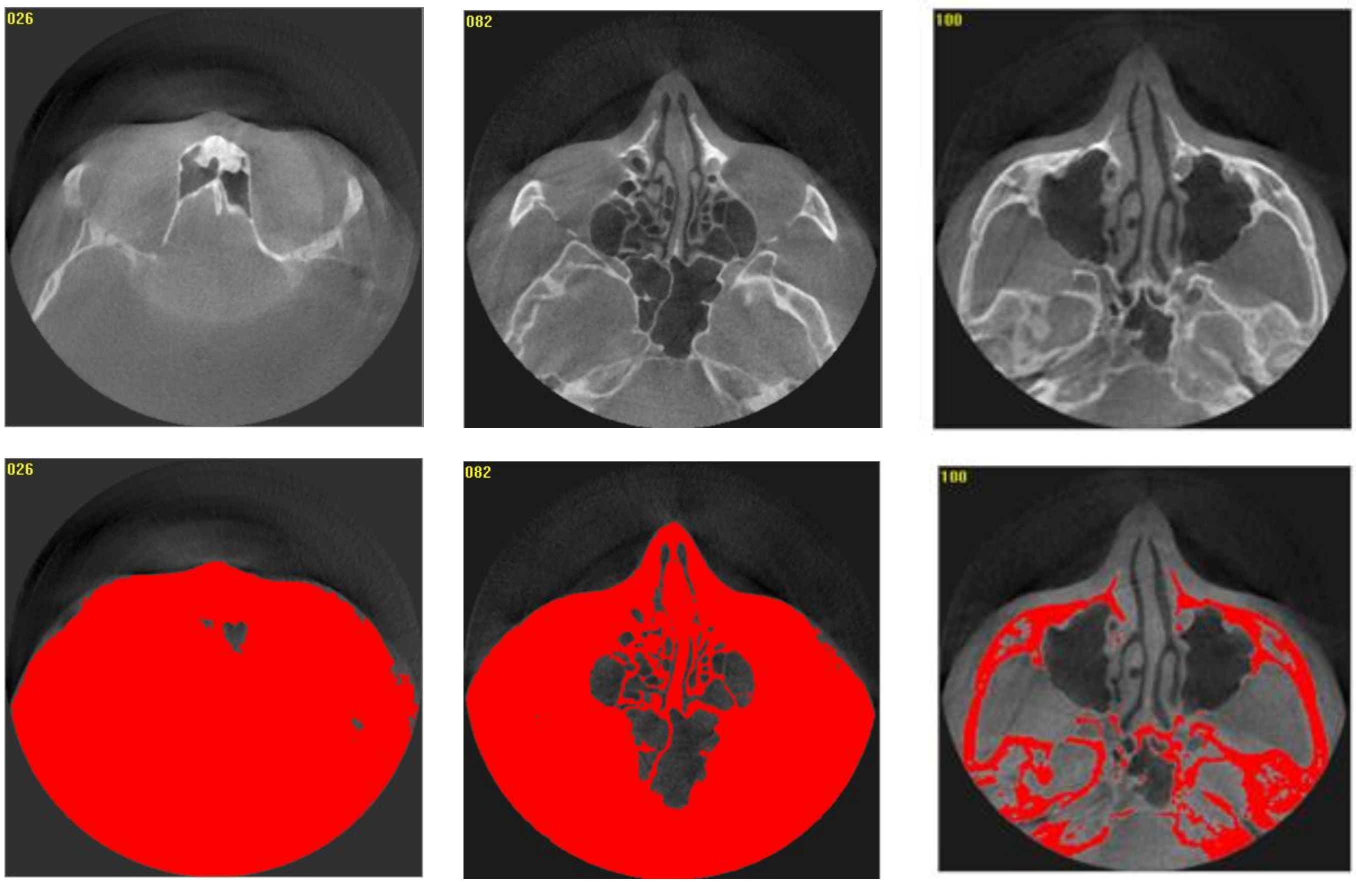
Original image   Result image



Original image   Result image

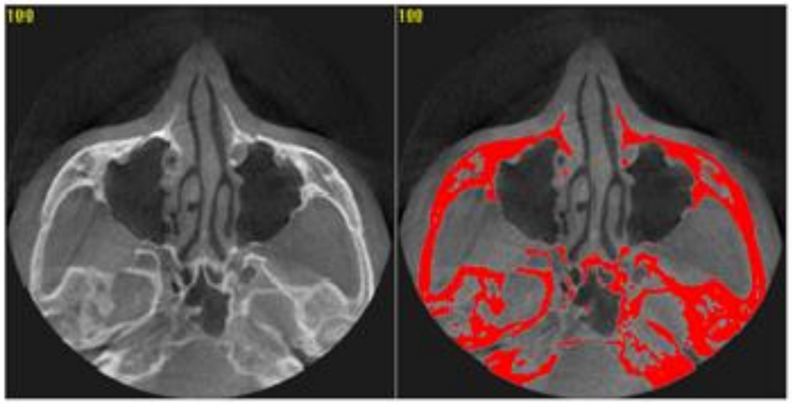
Thresholding

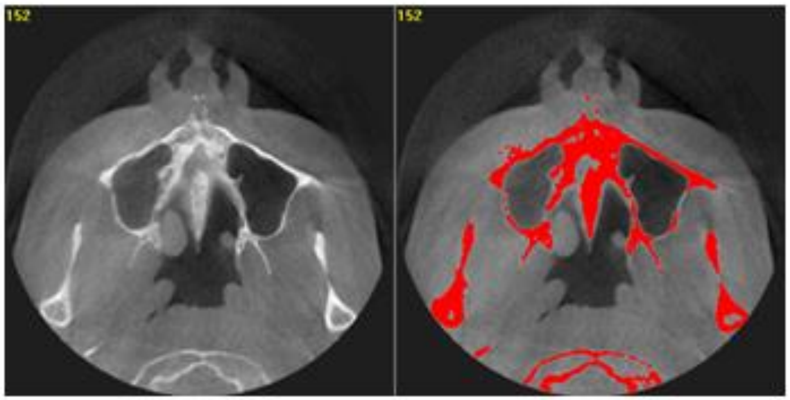
- Otsu threshold
  - Apply to Dental CT

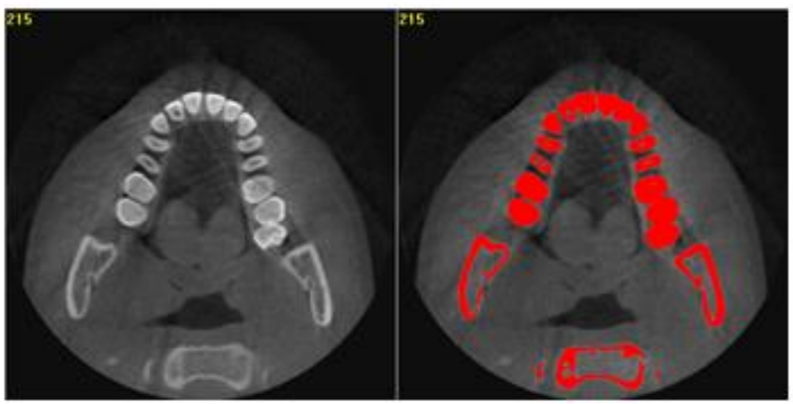


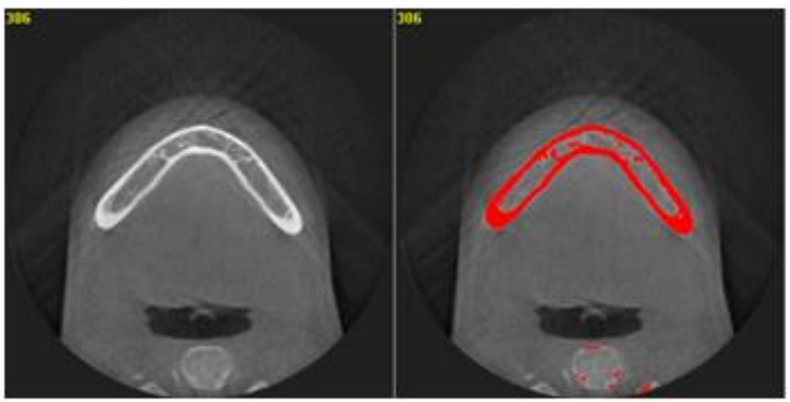
Thresholding

- Iterative Otsu threshold
  - Apply to Dental CT







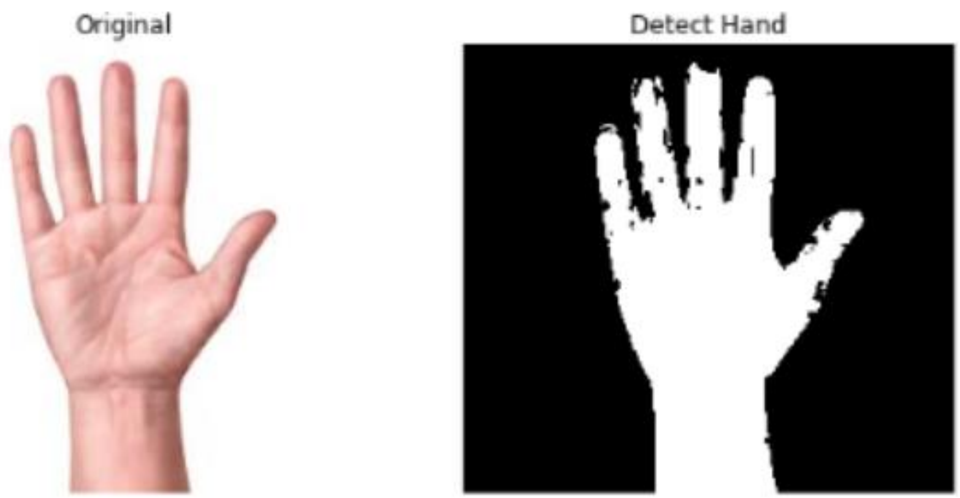


Thresholding for Color Image

- 색상 범위 지정에 의한 영역 분할
  - RGB → HSV
  - cv2.inRange(src, lowerb, upperb[,dst])

$$\mathrm{dst}(x,y)=\begin{cases}255 & lowerb(x,y)\le src(x,y)\le upperb(x,y) \text{ 일 때} \\ 0 & \text{그 외}\end{cases}$$

  - 손 얼굴 등 피부 검출 분할에 유용함



출처: https://wjddyd66.github.io/opencv/OpenCV(6)/#%EC%BB%AC%EB%9F%AC-%EB%B2%94%EC%9C%84%EC%97%90-%EC%9D%98%ED%95%9C-%EC%98%81%EC%97%AD-%EB%B6%84%ED%95%A0

Region Growing

- Seed Region Growing
  - Algorithm
    - T : the set of all pixels which are on the borders of the regions

$$T = \left\{ x \notin \bigcup_{i=1}^{n} A_i \mid N(x) \bigcap \bigcup_{i=1}^{n} A_i \neq 0 \right\}$$

    - Distance : the measure which says how far the intensity of the regarde $\sigma_w^2(t)$l is from the intensity mean value of those regions

$$\delta(x) = g(x) - mean_{y \in A_i(x)}[g(y)]$$

    - Its minimal distance form the neighboring regions as :

$$\delta(z) = \min_{x \in T} \{\delta(x)\}$$

Region Growing

- Seed Region Growing
  - Example

- Start with a single pixel (**seed**) and add new pixels slowly

(1) Choose the seed pixel (2) Check the neighboring pixels and add them to the region if they are similar to the seed (3) Repeat step 2 for each of the newly added pixels; stop if no more pixels can be added.

Seed point

10	11	10	9	11
18	20	19	18	12
14	15	15	19	13
2	2	1	14	14
1	3	2	10	16
2	2	1	11	18
3	1	2	12	19

Region Growing

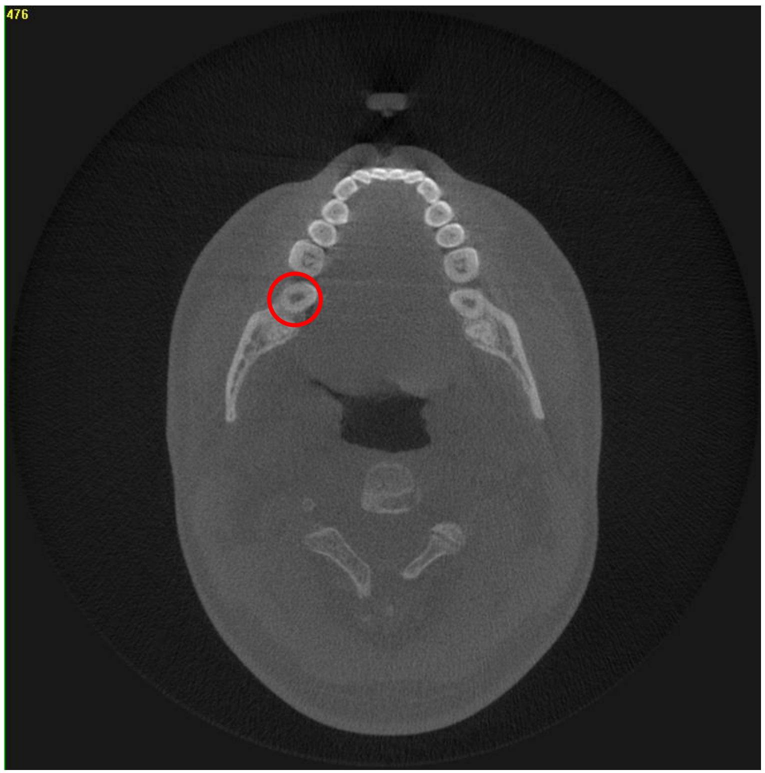
- Seed Region Growing
  - Example
    - Add pixels whose distance is smaller than threshold
    - Add pixels whose mean of region is smaller than threshold

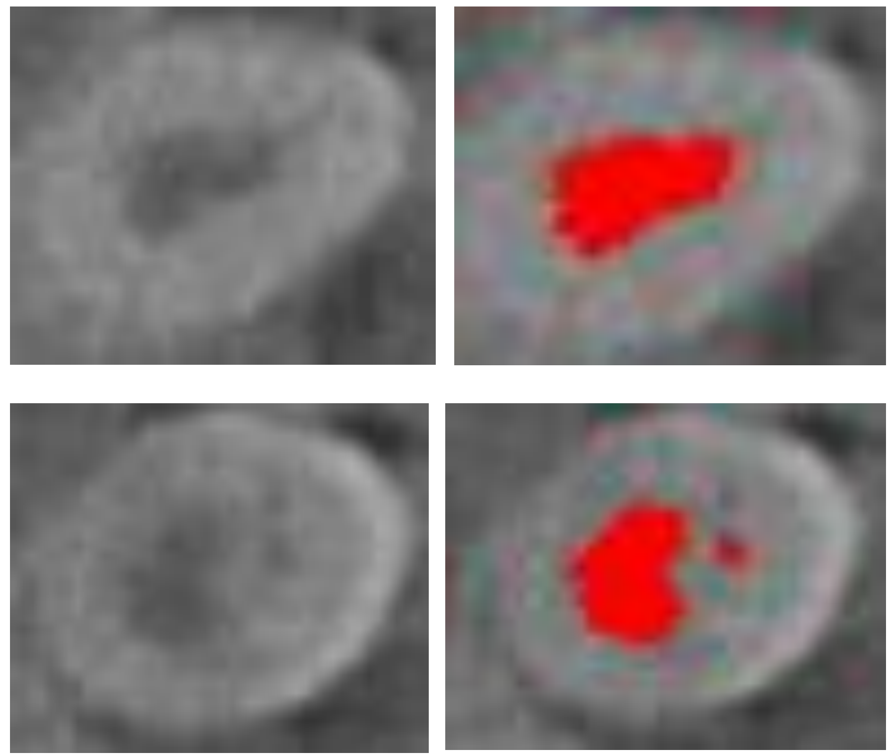
10	11	10	9	11
18	20	19	18	12
14	15	15	19	13
2	2	1	14	14
1	3	2	10	16
2	2	1	11	18
3	1	2	12	19

10	11	10	9	11
18	20	19	18	12
14	15	15	19	13
2	2	1	14	14
1	3	2	10	16
2	2	1	11	18
3	1	2	12	19

Region Growing

- Seed Region Growing
  - Apply to Dental CT





Region Growing

- Seed Region Growing
  - cv2.floodFill
    - (image, mask, seedPoint, newVal[, loDiff[, upDiff[, flags]]])

      parameter

      - image: Input
      - seedPoint: starting Point
      - newVal: 안을 채울 새로운 Pixel 값
      - loDiff: 현재 Pixel과의 차이의 최소값
      - upDiff: 현재 Pixel과의 차이의 최대값

      return

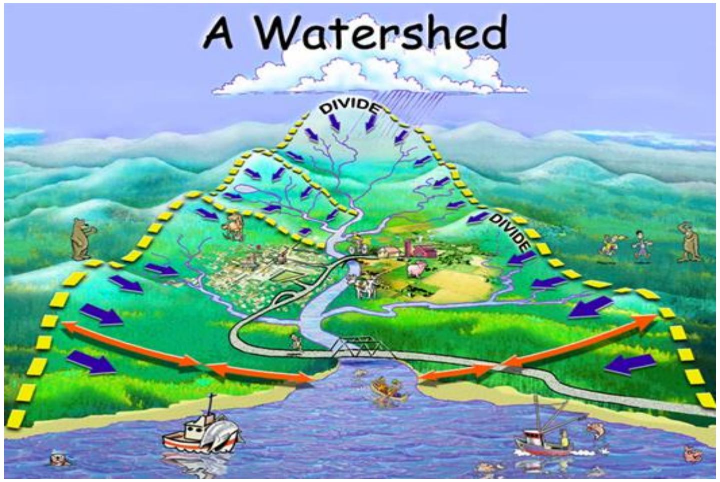
      - mask: 지정한 조건으로 이미지를 채운값
      - rect: mask의 바운딩 사각형

    - seed point로 부터 픽셀 채우기

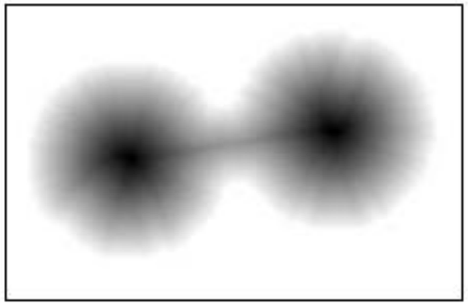
$$src(x', y') - loDiff \leq src(x, y) \leq src(x', y') + upDiff$$

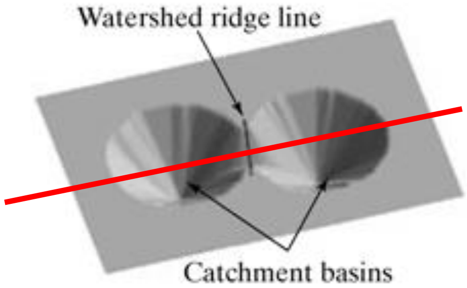
Watershed

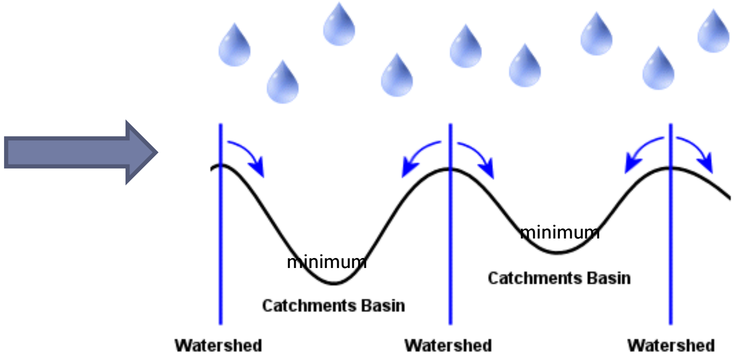
- **In Topography**
  - The area of land where all of the water that is under it of drains off of it goes into the same place.



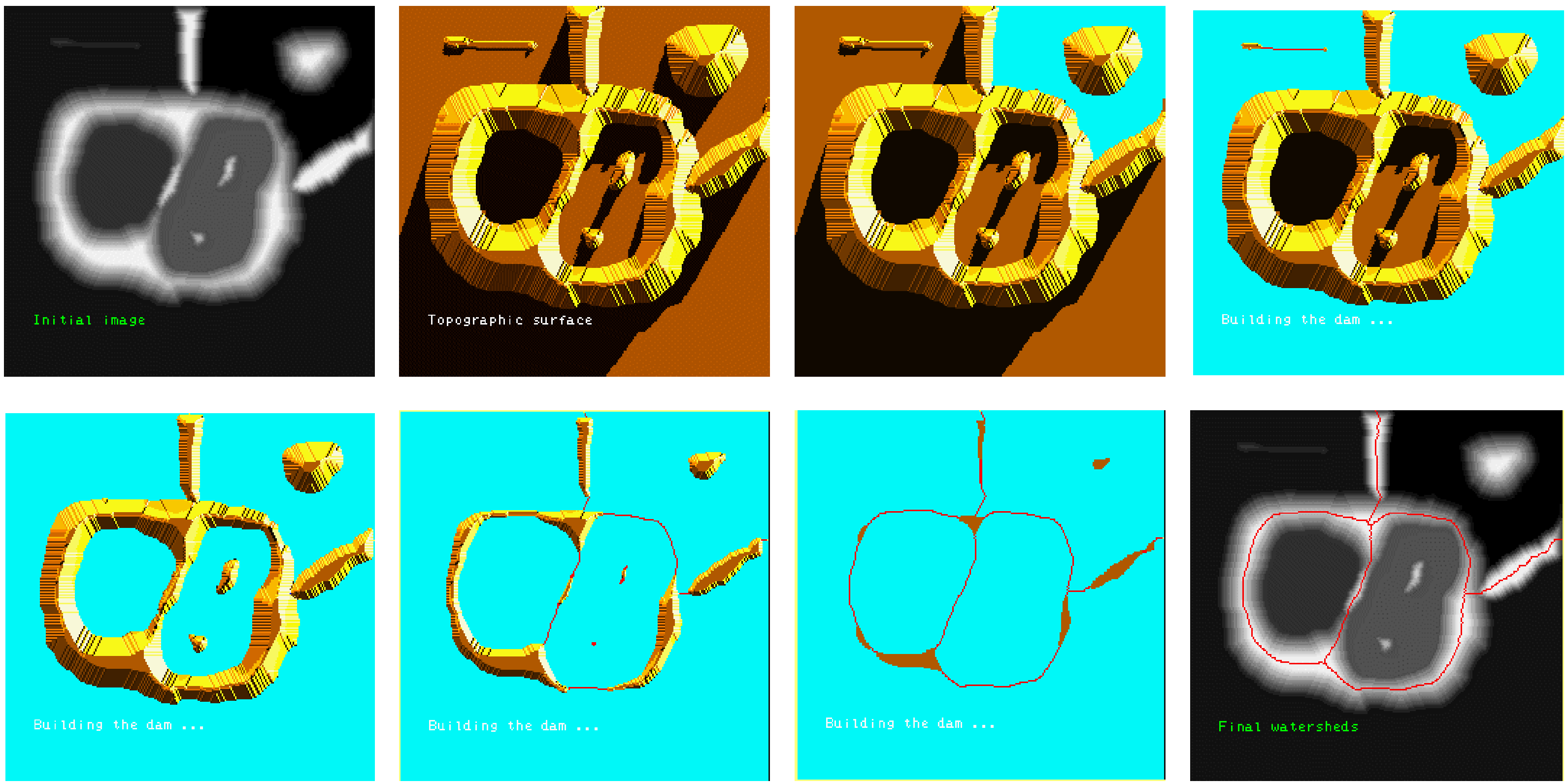
- **In Image**







Watershed



The sequence of Watershed Segmentation

Watershed

- **Advantage**
  - Fast, Simple, Intuitive
  - Produce a complete division of the image in separated regions
  - Able to use when the contrast is poor
  - Provide closed contours

- **Disadvantage**
  - Sensitivity to noise
  - Oversegmentation

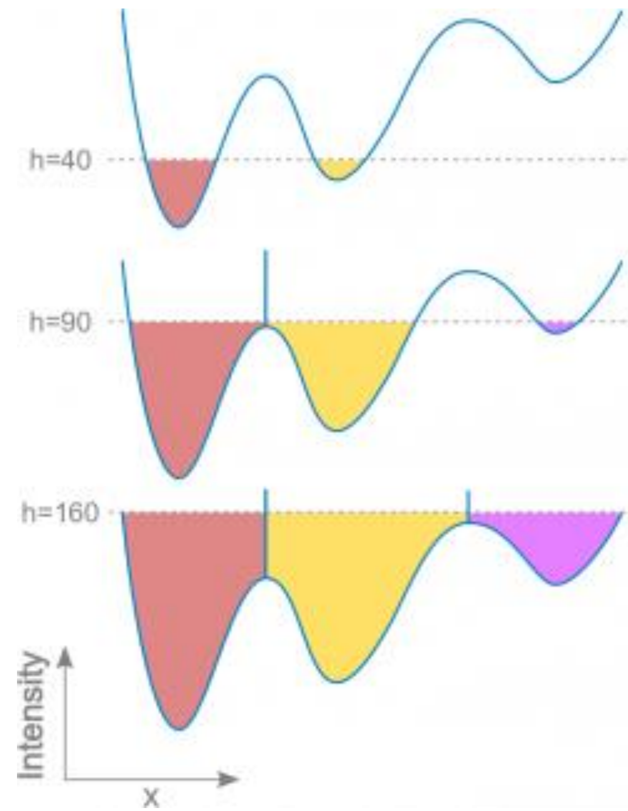
Watershed

- **Advantage**
  - Fast, Simple, Intuitive
  - Produce a complete division of the image in separated regions
  - Able to use when the contrast is poor
  - Provide closed contours

- **Disadvantage**
  - Sensitivity to noise
    → need to pre-processing (Noise reduction)
  - Oversegmentation
    → need to post-processing (Region merging)

Watershed

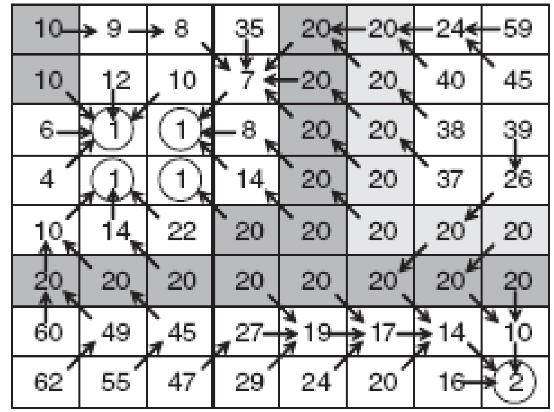
- Immersion simulation
  - 영상을 gray scale로 변환
    - 각 픽셀의 밝기 값을 높고 낮음으로 구분할 수 있음
  - 지형의 높낮이로 가정하여 높은 부분을 봉우리, 낮은 부분을 계곡으로 표현
  - 지형 간 섞이지 않도록 댐을 세움
  - cv2.watershed(images, markers)



Rain Simulation

10	9	8	35	20	20	24	59
10	12	10	7	20	20	40	45
6	1	1	8	20	20	38	39
4	1	1	14	20	20	37	26
10	14	22	20	20	20	20	20
20	20	20	20	20	20	20	20
60	49	45	27	19	17	14	10
62	55	47	29	24	20	16	2

(a)



(b)

-4	-4	-5	-6	-7	0	0	0
-5	-6	-7	-7	0	-1	-1	-1
-4	1	1	0	-1	-1	-1	-6
-3	1	1	-1	-1	-1	-1	-7
-3	-2	-1	-1	-1	-1	-7	-7
-2	-1	-1	-5	-5	-5	-5	-6
-2	-1	-1	-4	-4	-4	-5	-6
-3	-3	-3	-3	-3	-3	-4	2

(c)

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	2
1	1	1	1	1	1	2	2
1	1	1	2	2	2	2	2
1	1	1	2	2	2	2	2
1	1	2	2	2	2	2	2

(d)

Fig. 9. From left to right, top to down (a, b, c, d): (a) Original image, (b) detection of minima and steepest descending paths carried out in step 1 of the algorithm, (c) output matrix after step 1, (d) output matrix after step 2.

Watershed

- Immersion sim. vs. Rain sim.

10	10	12	14	20	20	15	13	11	11
10	10	12	14	20	20	15	13	11	11
12	12	12	14	20	20	15	13	13	13
14	14	14	14	20	20	15	15	15	15
20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20
18	18	18	18	20	20	19	19	19	19
17	17	17	18	20	20	19	18	18	18
15	15	17	18	20	20	19	18	17	17
15	15	17	18	20	20	19	18	17	17

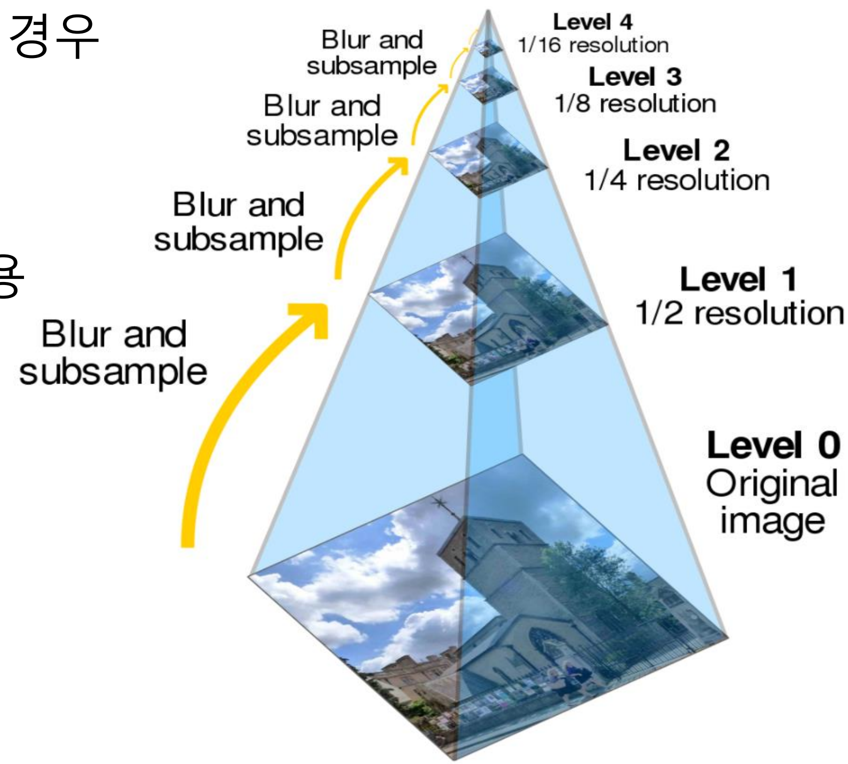
Immersion

10	10	12	14	20	20	15	13	11	11
10	10	12	14	20	20	15	13	11	11
12	12	12	14	20	20	15	13	13	13
14	14	14	14	20	20	15	15	15	15
20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20
18	18	18	18	20	20	19	19	19	19
17	17	17	18	20	20	19	18	18	18
15	15	17	18	20	20	19	18	17	17
15	15	17	18	20	20	19	18	17	17

Rainfall

피라미드 기반 분할

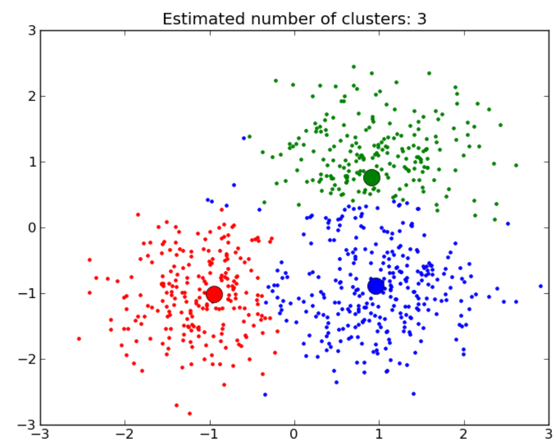
- 피라미드 영상
  - 원본 영상을 단계적으로 축소하여 피라미드 형성
    - 템플릿 매칭 시 스케일이 다른 경우 용이하게 사용
    - 물체 비교 및 매칭에 사용
  - 대표적으로 가우시안 피라미드 사용
    - cv2.pyrDown
    - cv2.pyrUp



출처: https://en.wikipedia.org/wiki/Pyramid_(image_processing)

피라미드 기반 분할

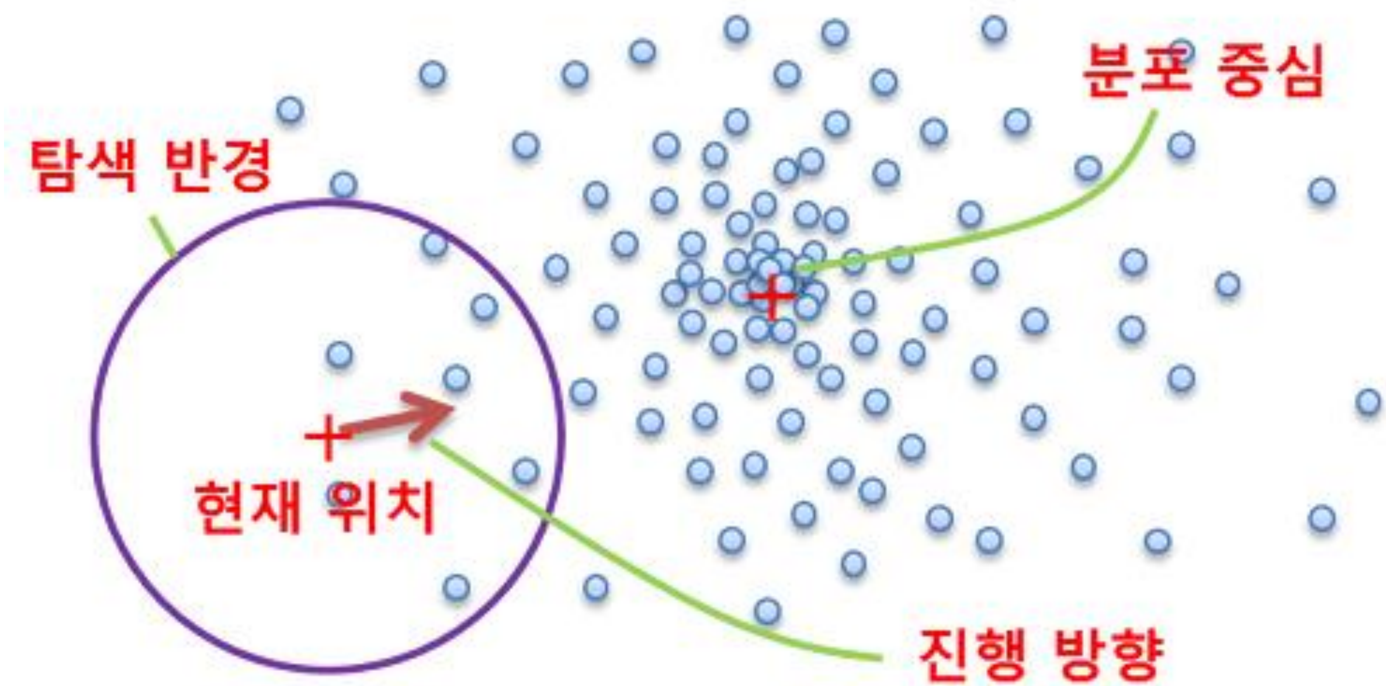
- MeanShift
  - 데이터 분포의 무게중심을 찾는 방법
    - 자신의 주변에서 가장 데이터가 밀집된 방향으로 이동
      - ROI 설정 (radius)
      - ROI에서 가장 밀도가 큰 곳을 찾고 중심으로 설정
      - 중심을 기준으로 ROI 다시 설정
      - 중심 위치의 변화가 없을 때 까지 위의 단계 반복
  - 영상 분할 뿐만 아니라 잡음 제거에도 사용
  - cv2.pyrMeanShiftFiltering
    - (src, sp, sr[, dst[, maxLevel[, termcrit]]]) → dst
      - **src** – The source 8-bit, 3-channel image.
      - **dst** – The destination image of the same format and the same size as the source.
      - **sp** – The spatial window radius.
      - **sr** – The color window radius.
      - **maxLevel** – Maximum level of the pyramid for the segmentation.
      - **termcrit** – Termination criteria: when to stop meanshift iterations.



출처: http://scikit-learn.sourceforge.net/0.5/auto_examples/cluster/plot_mean_shift.html

피라미드 기반 분할

- MeanShift



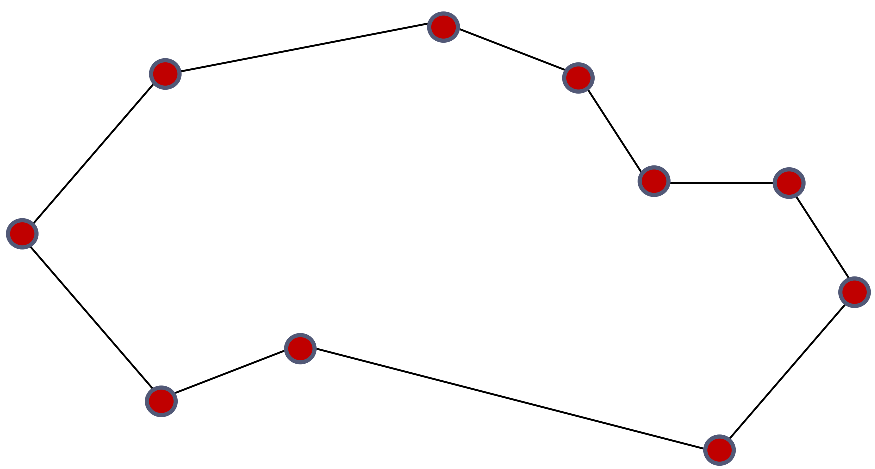
$$(x, y) : X - sp \le x \le X + sp, Y - sp \le y \le Y + sp, \|(R, G, B) - (r, g, b)\| \le sr$$

$$(X, Y)\ (X', Y'), (R, G, B)\ (R', G', B').$$

$$I(X, Y) < -(R*, G*, B*)$$

General Curve Evolution

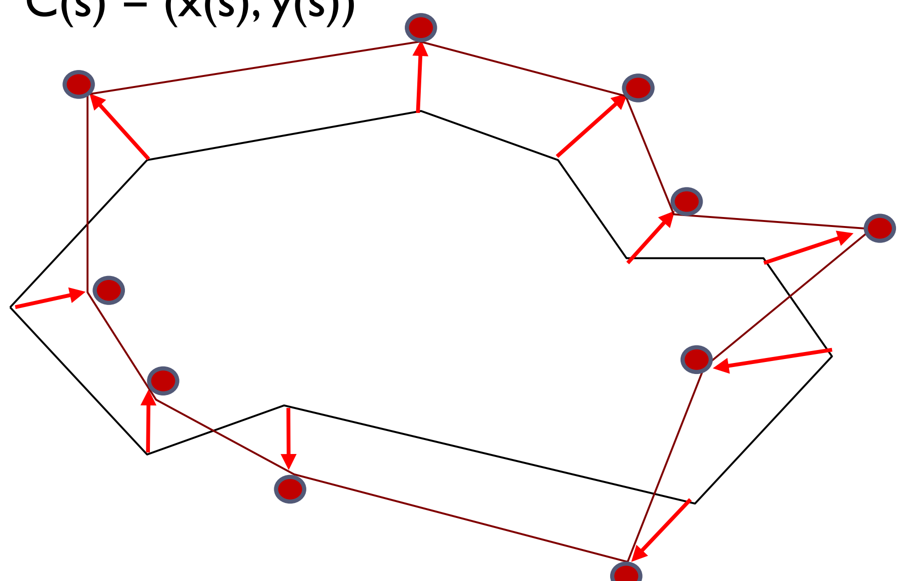
- Curve Define
  - Explicit
    - Representation one explicitly writes down the points that belong to the interface
    - C = { Pi = (xi, yi) , i = 1, 2, 3, …. N } or
    - C(s) = (x(s), y(s))



Curve (interface)

General Curve Evolution

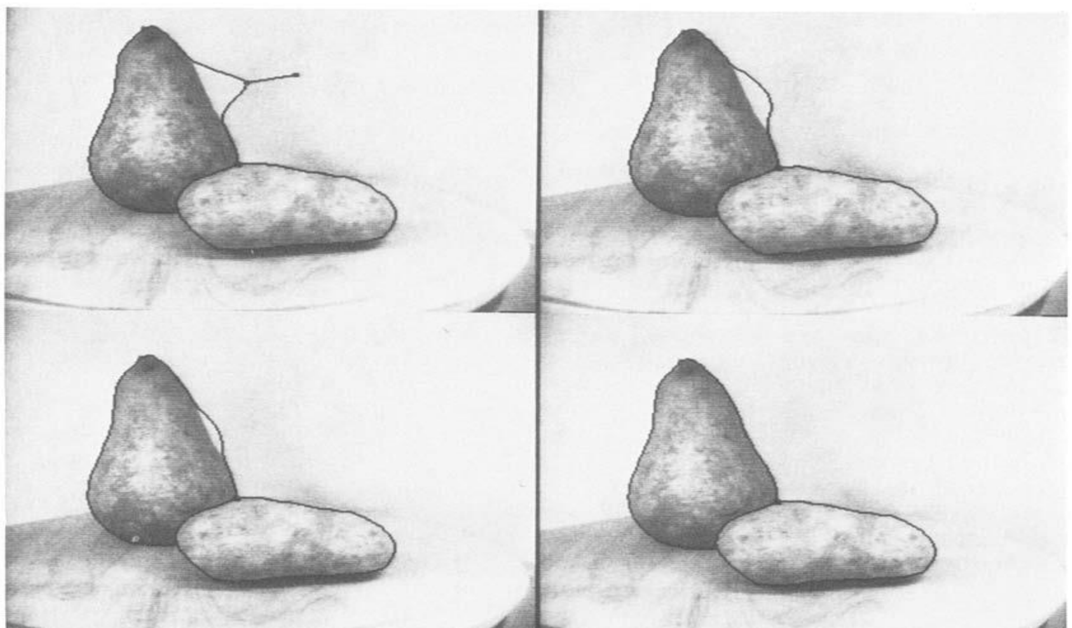
- Curve Define
  - Explicit
    - Representation one explicitly writes down the points that belong to the interface
    - $C = \{ P_i = (x_i, y_i) \ , i = 1, 2, 3, \ldots. N \}$ or
    - $C(s) = (x(s), y(s))$



Curve (interface)

Active Contour Model

- Explicit curve model
  - Kass et al, IJCV, 1987
  - Concept
    - Giving an image $u_0 : \Omega \rightarrow \Re$
    - Evolve a curve C to detect objects in u_0



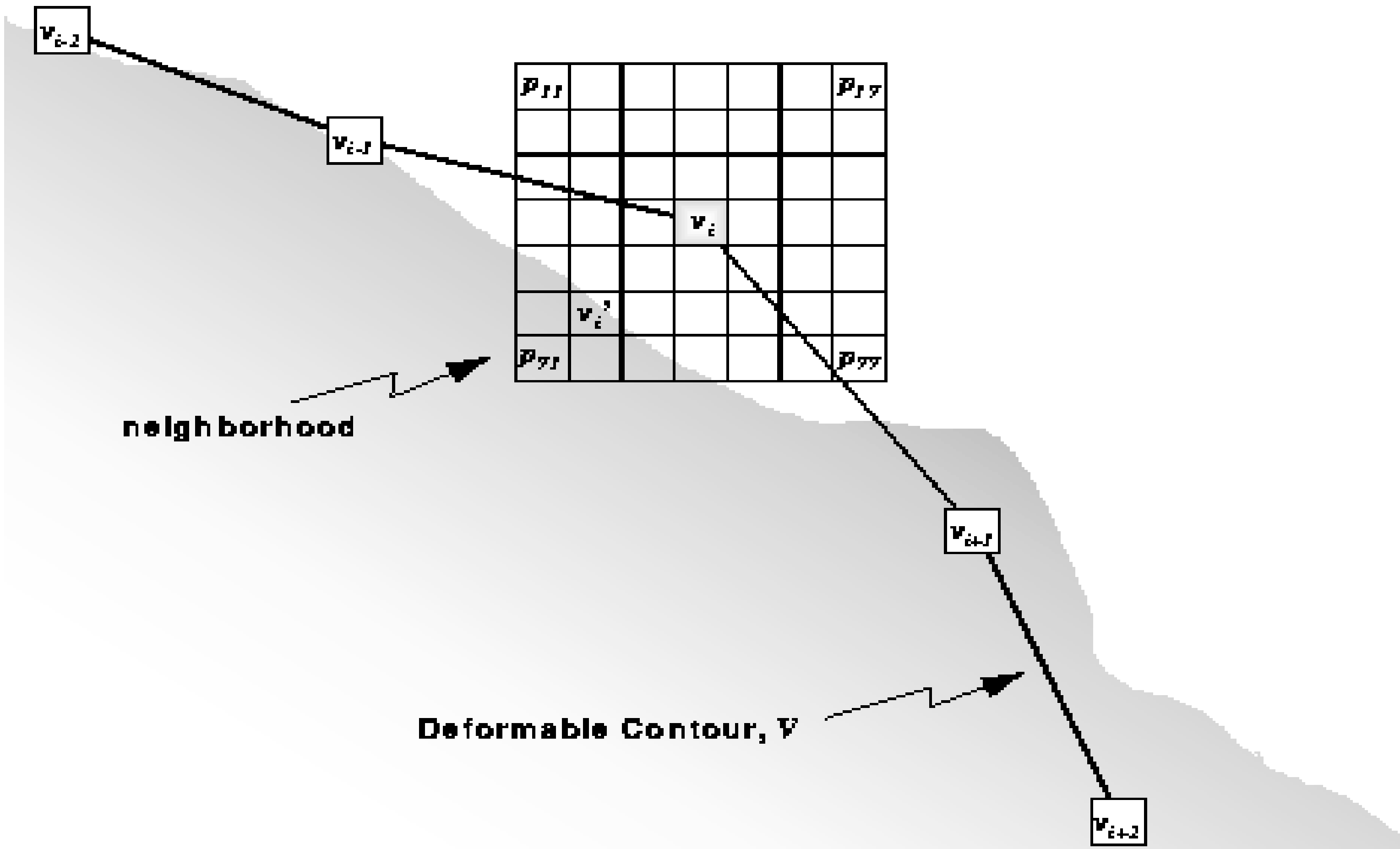
Active Contour Model

- Explicit curve model
  - Kass et al, IJCV, 1987
  - Concept
    - Giving an image $u_0 : \Omega \rightarrow \Re$
    - Evolve a curve C to detect objects in u_0
  - Snake model
    - Energy function

$$E[(C)(p)] = \underbrace{\alpha \int_0^1 E_{int}(C(p))dp}_{\text{internal energy}} + \underbrace{\beta \int_0^1 E_{img}(C(p))dp}_{\text{external energy}} + \gamma \int_0^1 E_{con}(C(p))dp$$

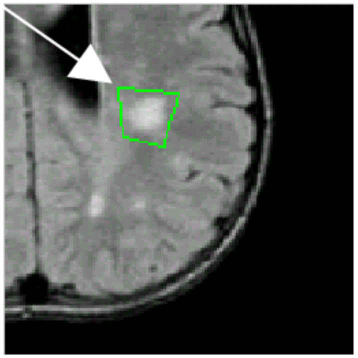
Active Contour Model

- Explicit curve model

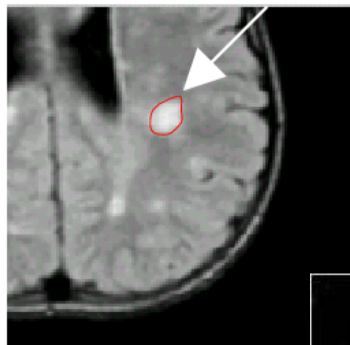


Active Contour Model

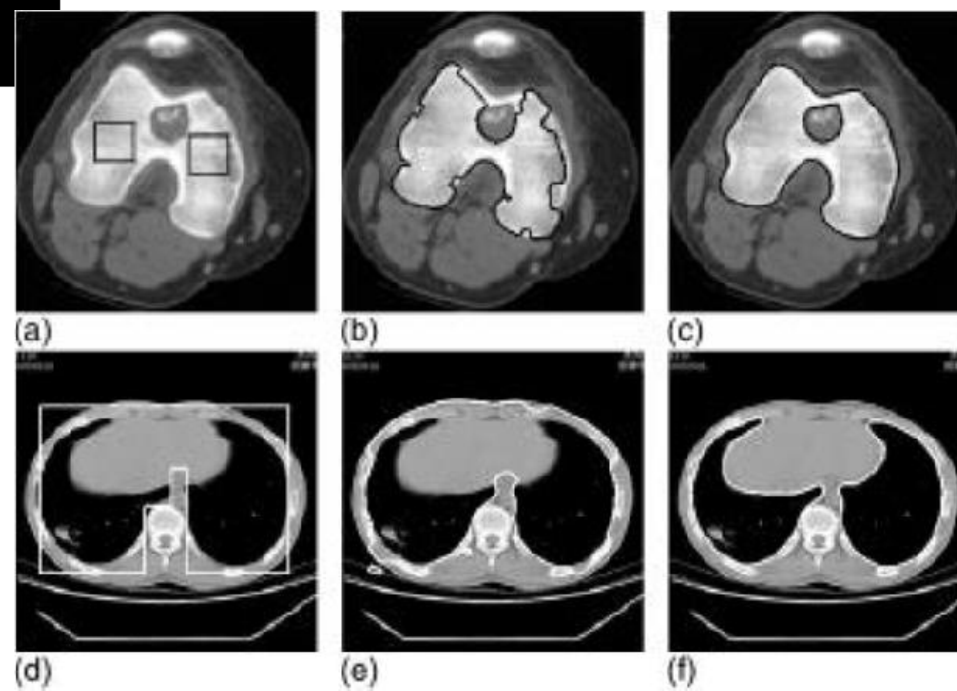
- Result



Initial contour



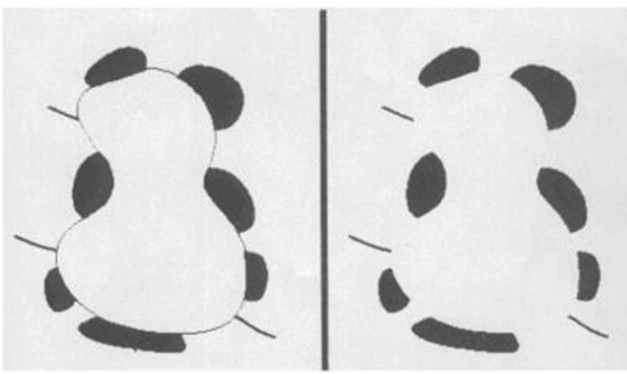
Final contour



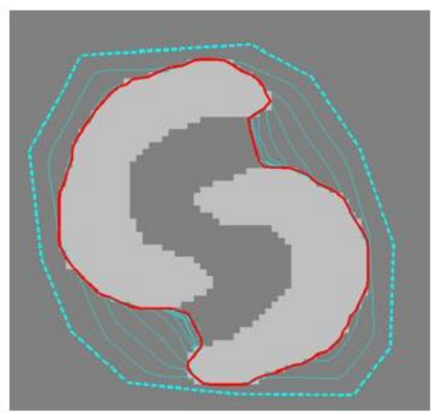
Active Contour Model

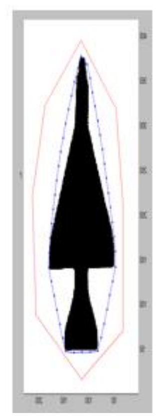
- Pros.
  - Low complexity, fast
  - Can account for open as well as closed structures
  - Flexible energy function
  - Easy to implementation using Euler method

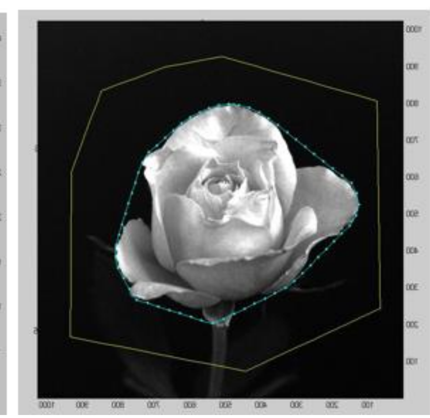
$$x(t + \Delta t) = x(t) + \Delta t \dot{x}(t)$$



- Cons.
  - Sensitive to the initial conditions
  - Sensitive to noise
  - Difficult to handle topology changing
  - Difficult to segment a concave region



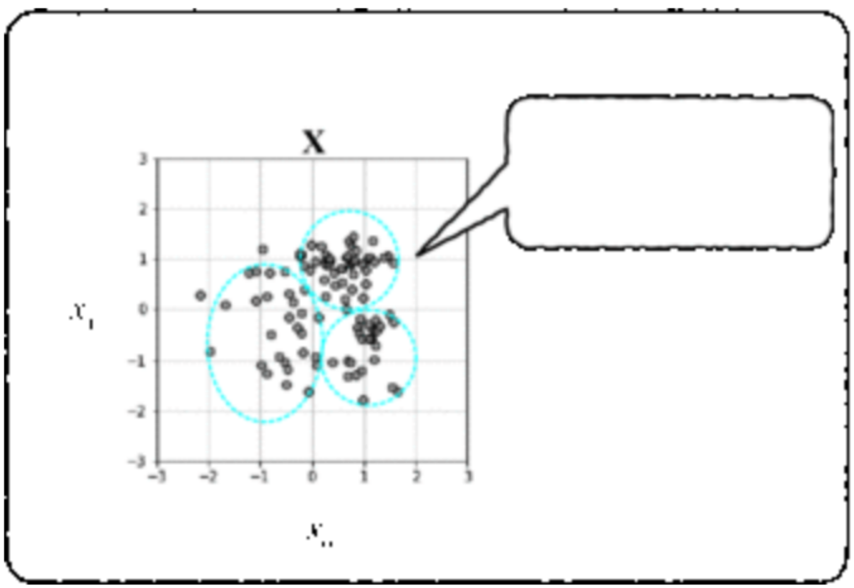


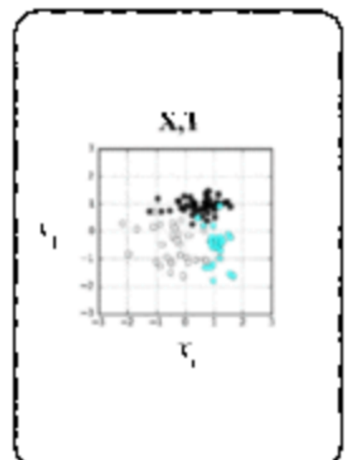


Clustering

- 개념
  - 클래스 정보 없이 입력 데이터가 비슷한 것끼리 클래스로 나누는 작업

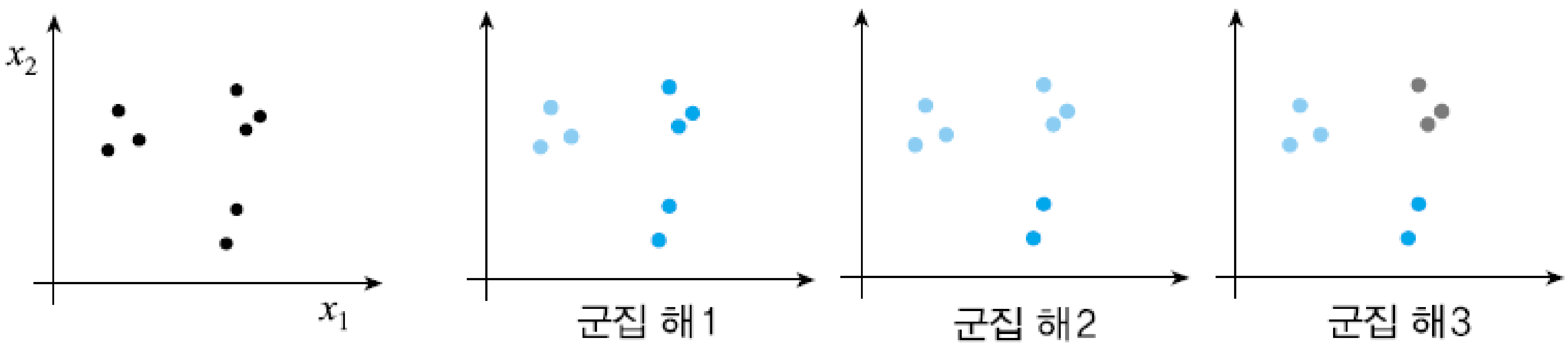
- 2차원 입력 데이터
  - 입력 데이터 X 사용
  - 클래스 데이터 T는 사용하지 않음





Clustering

- 특성
  - 주관적인 판단에 따라 결과가 달라짐
    - 클러스터링 결과의 품질은 응용이 처한 상황과 요구사항에 따라 다름



(a) 샘플 집합 (b) 세 가지 군집화 결과를 어떻게 평가할까?

K-means Clustering

- 알고리즘

알고리즘 [10.4] *k*-means 알고리즘

입력: 샘플 집합 $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, 군집의 개수 k

출력: 군집 해 C

알고리즘:

1. k 개의 군집 중심 $Z = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k\}$를 초기화 한다.
2. **while** (TRUE) {
3.   **for** ($i = 1$ **to** N) $\mathbf{x}_i$를 가장 가까운 군집 중심에 배정한다.
4.   **if** (이 배정이 이전 루프의 배정과 같음) **break**;
5.   **for** ($j = 1$ **to** k) $\mathbf{z}_j$에 배정된 샘플의 평균으로 $\mathbf{z}_j$를 대치한다.
6. }

K-means Clustering

- 예제

7개 샘플을 k=3개의 군집으로 만드는 상황

$\mathbf{x}_1 = (18,5)^T,\ \mathbf{x}_2 = (20,9)^T,\ \mathbf{x}_3 = (20,14)^T,\ \mathbf{x}_4 = (20,17)^T,\ \mathbf{x}_5 = (5,15)^T,\ \mathbf{x}_6 = (9,15)^T,$

$\mathbf{x}_7 = (6,20)^T$

초기화에 의해 (그림 10.12(a)), $\{\mathbf{x}_1\}$은 $\mathbf{z}_1$

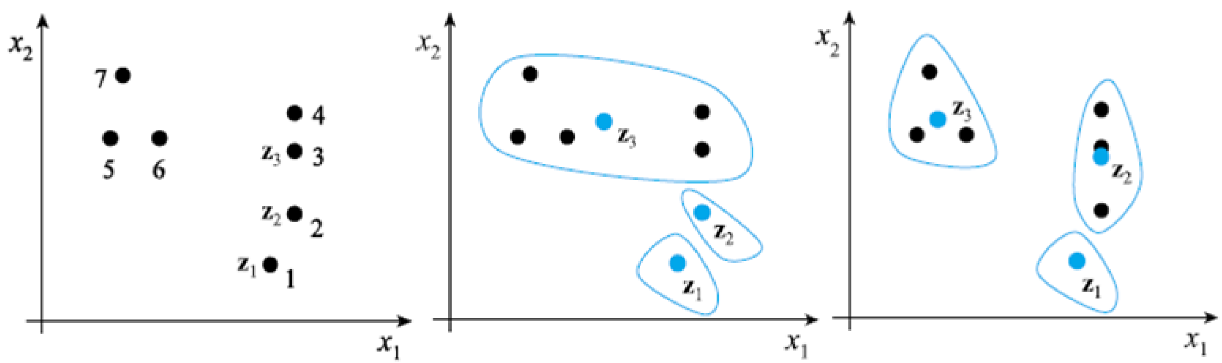
$\{\mathbf{x}_2\}$은 $\mathbf{z}_2$

$\{\mathbf{x}_3,\mathbf{x}_4,\mathbf{x}_5,\mathbf{x}_6,\mathbf{x}_7\}$은 $\mathbf{z}_3$

라인 5에 의해 (그림 10.12(b)), $\mathbf{z}_1 = \mathbf{x}_1 = (18,5)^T$

$\mathbf{z}_2 = \mathbf{x}_2 = (20,9)^T$

$\mathbf{z}_3 = (\mathbf{x}_3+\mathbf{x}_4+\mathbf{x}_5+\mathbf{x}_6+\mathbf{x}_7)/5 = (12,16.2)^T$



(a) 초기 군집 중심 (b) 첫 번째 루프 후 (c) 두 번째 루프 후

그림 10.12 k-means의 동작 예

K-means Clustering

- 예제

두 번째 루프를 실행하면 (그림 10.12(c)),

$\{\mathbf{x}_1\}$은 $\mathbf{z}_1$

$\{\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4\}$은 $\mathbf{z}_2$

$\{\mathbf{x}_5,\mathbf{x}_6,\mathbf{x}_7\}$은 $\mathbf{z}_3$

$$\mathbf{z}_1 = \mathbf{x}_1 = (18,5)^T$$

$$\mathbf{z}_2 = (\mathbf{x}_2+\mathbf{x}_3+\mathbf{x}_4)/3 = (20,13.333)^T$$

$$\mathbf{z}_3 = (\mathbf{x}_5+\mathbf{x}_6+\mathbf{x}_7)/3 = (6.667,16.667)^T$$

세 번째 루프는 그 이전과 결과가 같다. 따라서 멈춘다.

결국 출력은

$$C = \{\{\mathbf{x}_1\},\ \{\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4\},\ \{\mathbf{x}_5,\mathbf{x}_6,\mathbf{x}_7\}\}$$

K-means Clustering

- Apply to image
  - K개의 object로 구별함
  - 입력 데이터 : $u=[I_R(x,y),I_G(x,y),I_B(x,y)]$
  - L_2 norm를 이용하여 픽셀과 픽셀 사이 거리 측정

    $D(u_i,u_j|L^2)=||u^i-u^j||^2$

- 알고리즘
  - 랜덤으로 K개의 centroids 설정
  - 각 픽셀과 centroid와 거리 계산 및 cluster 설정

    $j=argmin_j||u^i-C||_{j2}$

  - 새로운 centroids 계산 $C_j=\sum_{u^i\in S_j} u^i$
  - 위의 과정 반복

K-means Clustering

- 구현
  - cv2.kmeans(data, K, bestLabels, criteria, attemps, flags[,centers]

parameter

- src: input
- K: 클러스터 개수
- criteria: 반복 회수 종료 시점 정의
- ateempts: 알고리즘 시도하는 회수, 서로다른 시도 횟수 중 최적을 레이블링 경과를 bestLabels에 저장하여 반환
- flags: K개의 클러스터 중심을 초기화하는 방법을 명시한다.
- cv2.KMEANS_RANDOM_CENTERS: 난수를 사용하여 설정
- cv2.KMEANS_PP_CENTERS: Arthur and Vassivitskii에 의해 제안 방법
- cv2.KMEANS_USE_INITIAL_LABELS: 처음 시도에는 사용자가 제공한 레이블을 사용하고, 다음 시도부터는 난수를 이용하여 임의로 설정

K-means Clustering

- Result

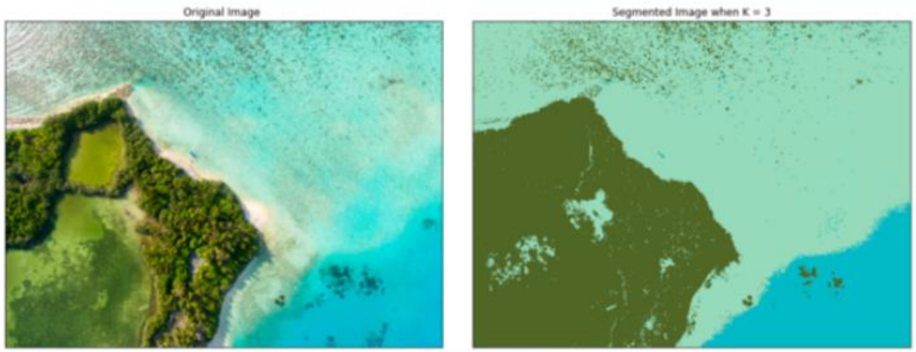


Image Segmentation when K=3

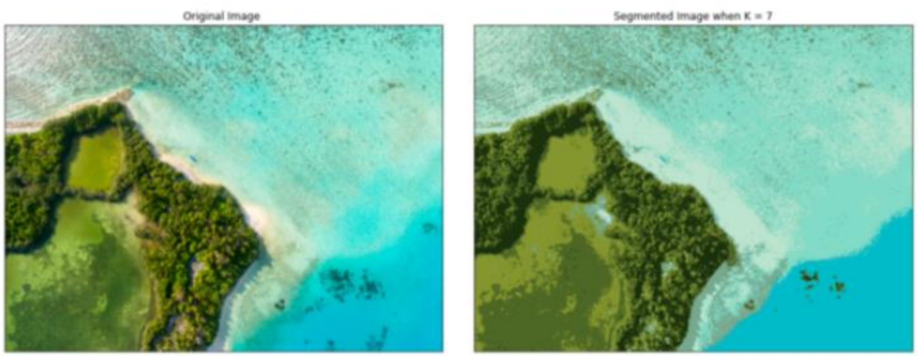


Image Segmentation when K=7

Image Segmentation when K=6

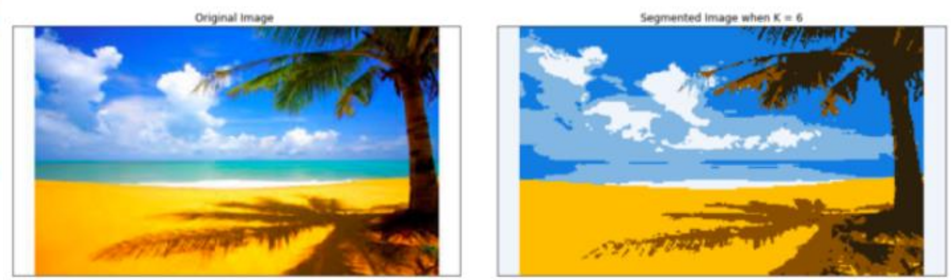


Image Segmentation when K=6

출처: https://www.kdnuggets.com/2019/08/introduction-image-segmentation-k-means-clustering.html

참고자료

- OpenCV4로 배우는 컴퓨터 비전과 머신러닝
  - 황선규 지음
  - 길벗출판사, 2019
- Python으로 배우는 OpenCV 프로그래밍
  - 김동근 지음
  - 가메출판사, 2018
- 파이썬으로 배우는 머신러닝의 교과서
  - 이토마코토 지음, 박광수 옮김
  - 한빛미디어, 2018
- 패턴인식
  - 오일석 지음
  - 교보문고, 2008