17 - Segmentasi Citra

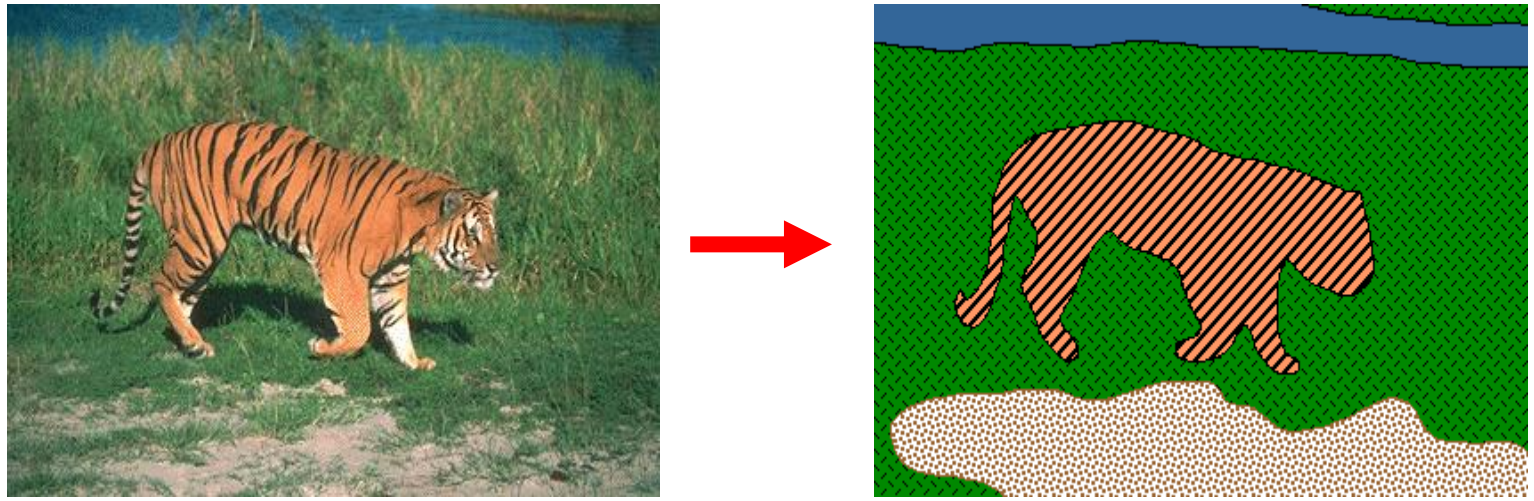
IF4073 Interpretasi dan Pengolahan Citra

Oleh: Rinaldi Munir

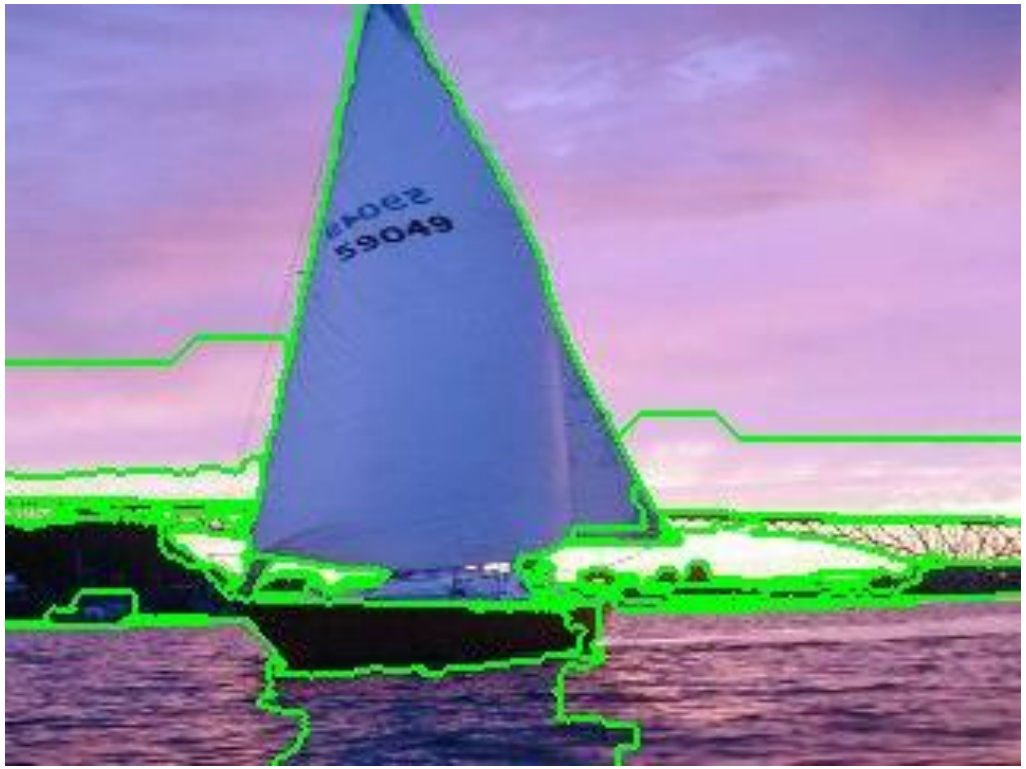
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2021

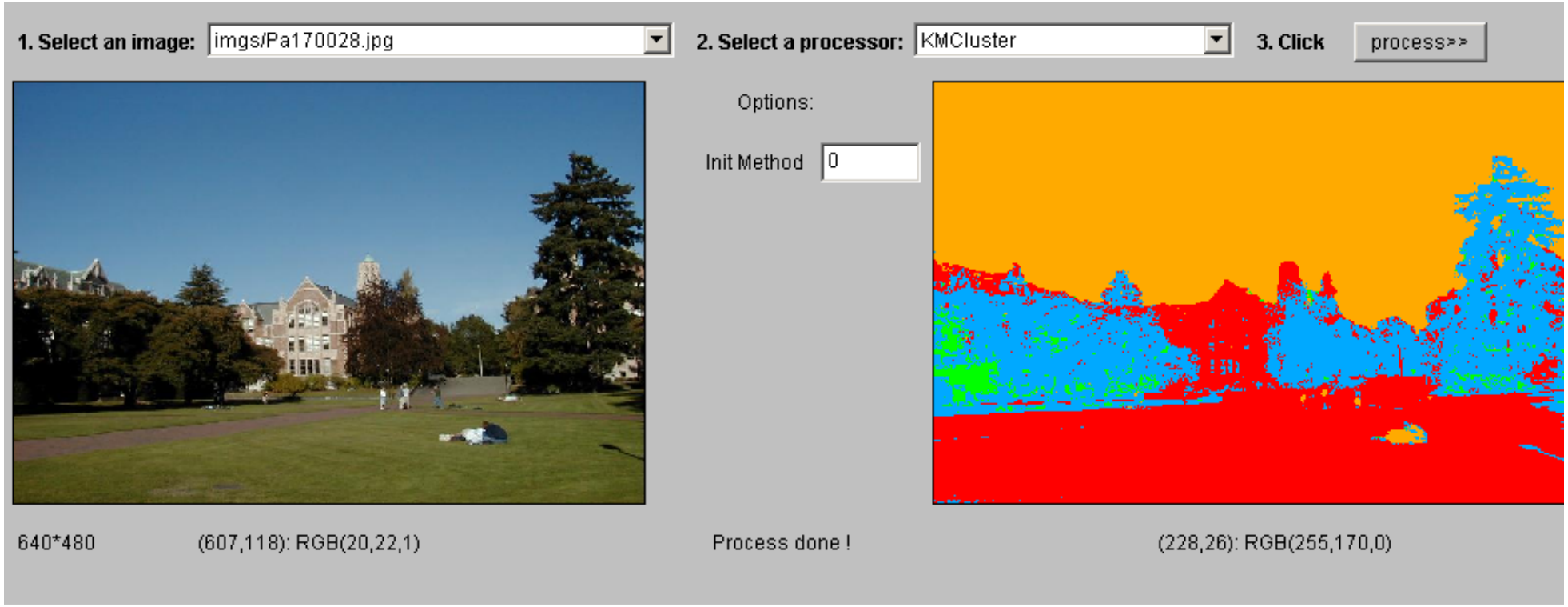
Tujuan Segmentasi Citra

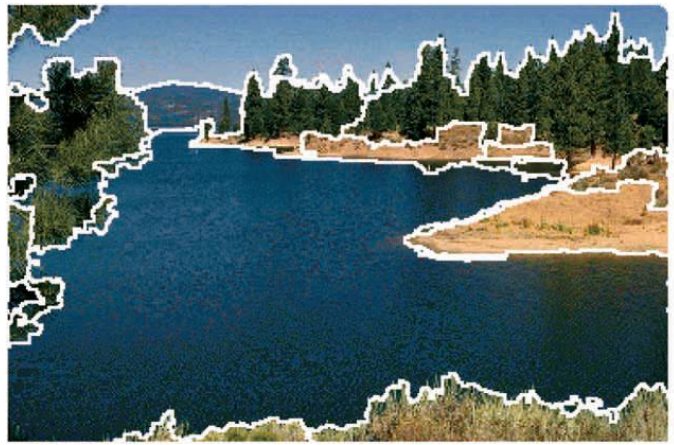
- Segmentasi citra (*image segmentation*) bertujuan untuk:
  1. membagi citra menjadi region-region atau objek-objek.
  2. memisahkan objek dengan latar belakang



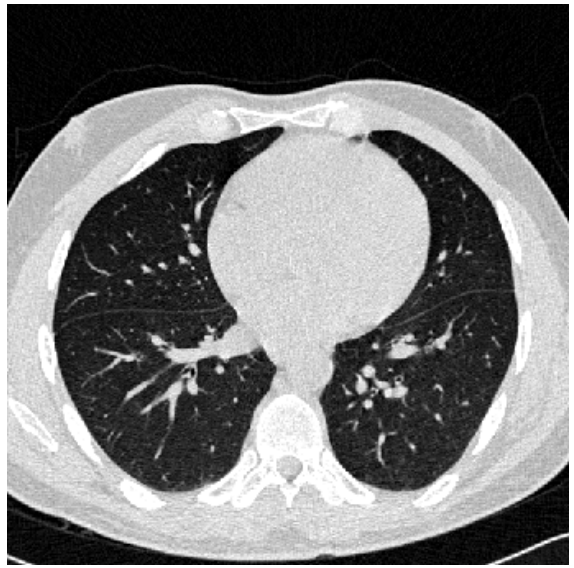
- Goal segmentasi citra adalah menemukan bagian citra yang koheren atau objek spesifik.
- Citra disegmentasi berdasarkan properti yang dipilih seperti kecerahan, warna, tekstur, dan sebagainya.
- Segmentasi membagi citra menjadi sejumlah region yang terhubung, tiap region bersifat homogen berdasarkan properti yang dipilih.
- Segmentasi citra merupakan tahapan sebelum melakukan *image/object recognition*, *image understanding*, dll.




1. Select an image:
imgs/Pa170028.jpg
2. Select a processor:
KMCluster
3. Click
process>>
Options:
Init Method
0
640*480
(607,118): RGB(20,22,1)
Process done !
(228,26): RGB(255,170,0)

Sumber: CS 4487/9587 Algorithms for Image Analysis: Basic Image Segmentation



Citra medis

Hasil segmentasi

Kriteria Segmentasi

- Menurut Pavlidis:

Segmentasi adalah partisi citra I menjadi sejumlah region S_1, S_2, … S_m yang memenuhi persyaratan:

1. $\cup S_i = S$ — Partisi mencakup keseluruhan *pixel* di dalam citra.
2. $S_i \cap S_j = \phi$, $i \neq j$ — Tidak ada region yang beririsan.
3. $\forall S_i$, $P(S_i) = true$ — P = Predikat homogenitas, dipenuhi oleh setiap region
4. $P(S_i \cup S_j) = false$, $i \neq j$, S_i adjacent S_j — Gabungan region bertetangga tidak memenuhi predikat

- Jadi, yang harus dilakukan adalah mendefinisikan dan mengimplementasikan predikat *similarity.*
- Misalnya, *similarity* didasarkan pada pixel-pixel di dalam rentang nilai yang sama.

Metode segmentasi citra

Metode segmentasi citra umumnya dikelompokkan berdasarkan dua pendekatan:

1. *Diskontinuitas*

   Mempartisi citra berdasarkan perubahan nilai intensitas *pixel* yang cepat seperti tepi (*edge detection*)

2. *Similarity*

   Mempartisi citra berdasarkan kemiripan area menurut properti yang ditentukan

   Metode segmentasi citra yang termasuk ke dalam pendekatan ini:

   a) Pengambangan (*thresholding*)

   b) *Region growing*

   c) *Split and merge*

   d) *Clustering*

- Pendeteksian tepi dapat digunakan untuk melakukan segmentasi citra.
- Metode-metode deteksi tepi sudah dibahas pada materi sebelumnya, seperti metode berbasis gradien (Sobel, Prewit, Canny, Roberts, Laplacian, LoG, dll)

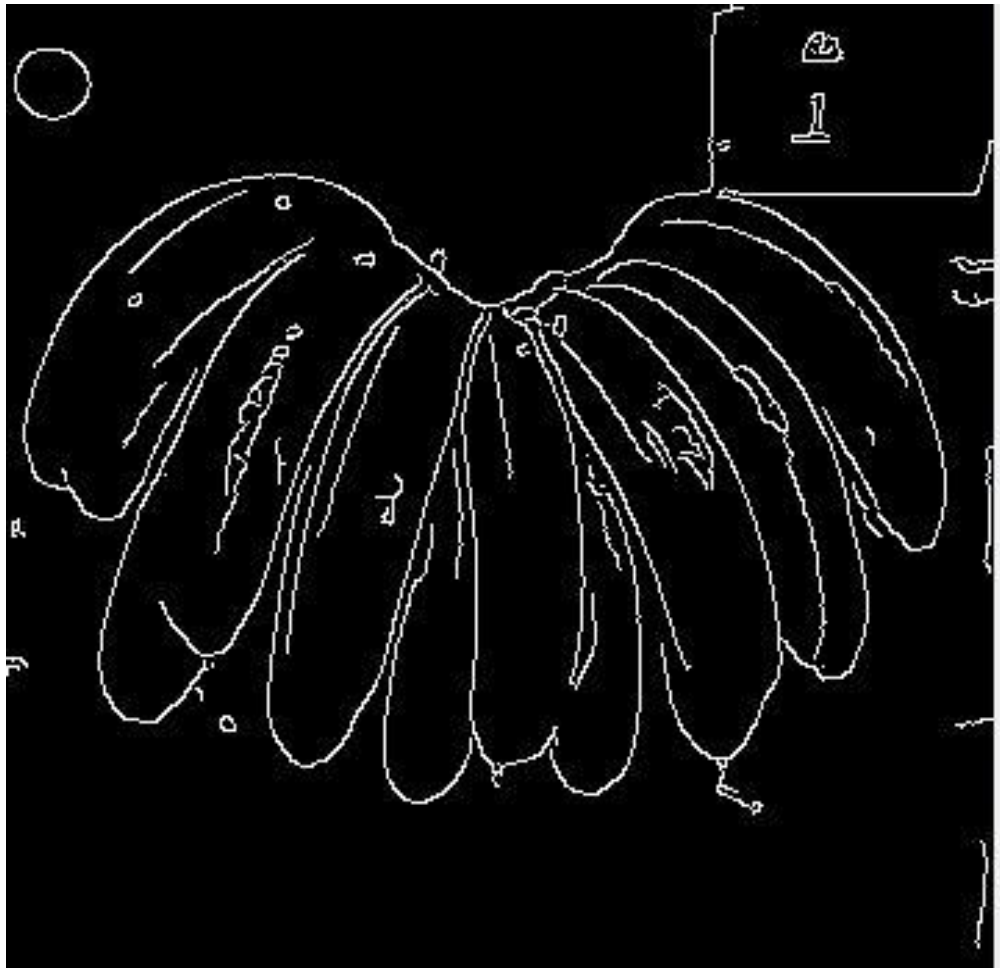
a b
c d

FIGURE 10.10
(a) Original image. (b) $|G_x|$, component of the gradient in the x-direction. (c) $|G_y|$, component in the y-direction. (d) Gradient image, $|G_x| + |G_y|$.

a) Complemented Image

b) Edged Image

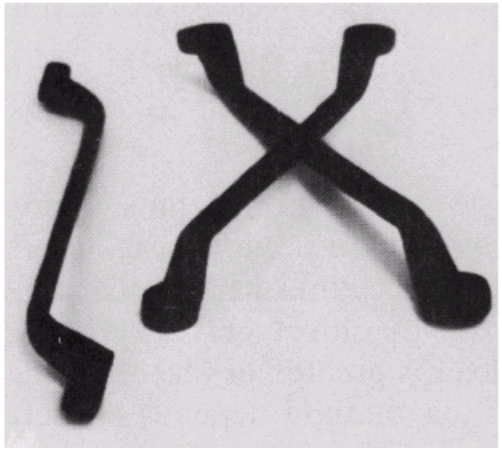
Figure 3. Complemented Image and Edged Image


1


1

Segmentasi citra berdasarkan *similarity*

- Cara paling sederhana menemukan bagian citra yang koheren adalah berdasarkan nilai intensitas pixel atau warna



Perkakas menjadi bagian yang koheren

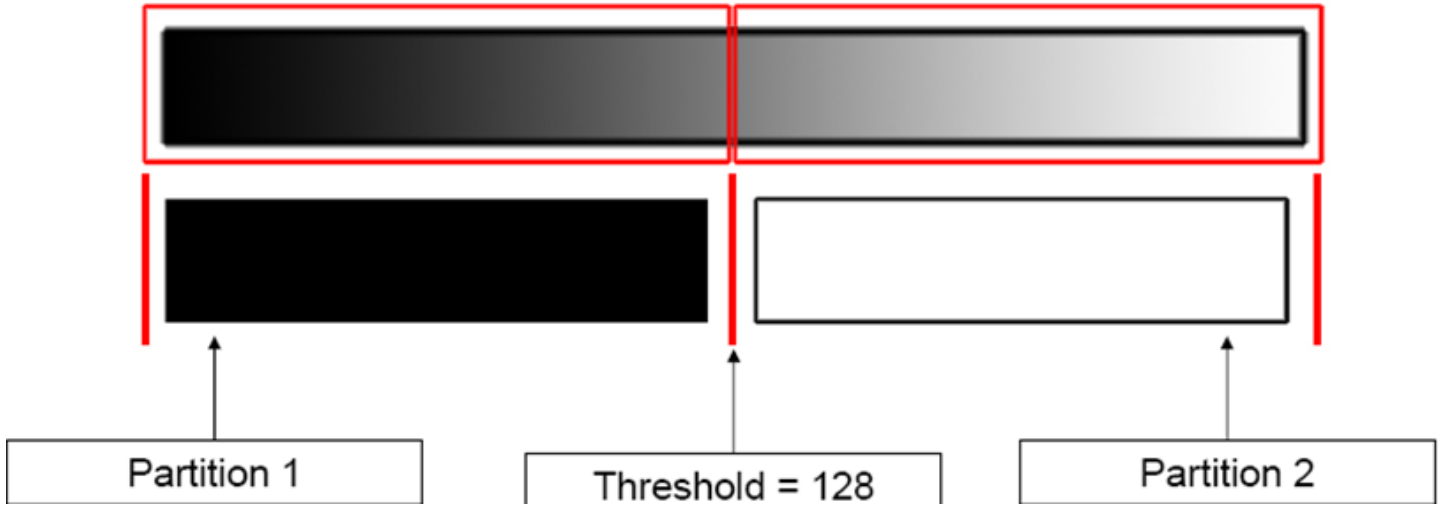
Rumah, rumput, dan langit membentuk bagian koheren yang berbeda

- Metode segmentasi berbasis *similarity*: pengambangan, *region growing*, *split and merge*, dan *clustering*.

Sumber: CS 4487/9587 Algorithms for Image Analysis

1. Pengambangan

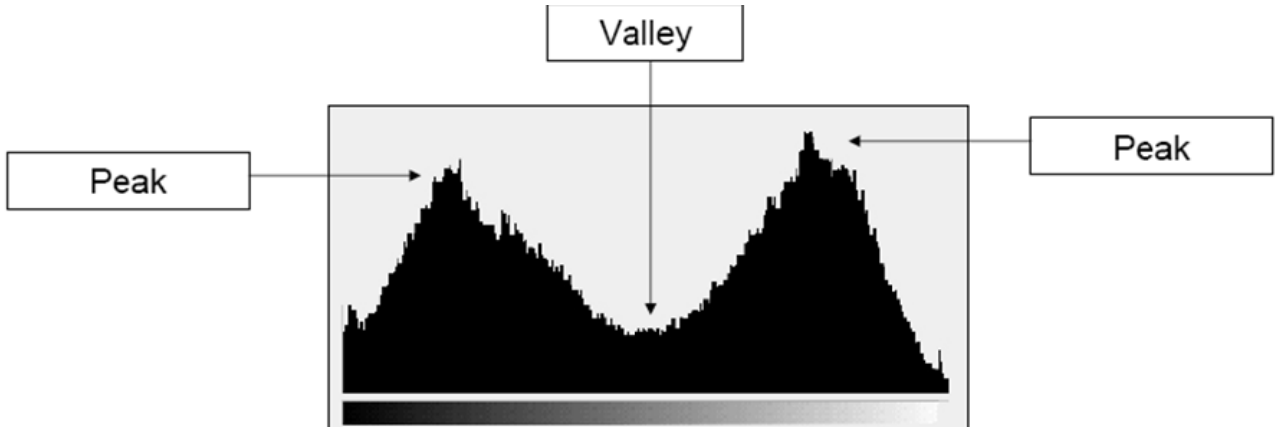
- Sudah dijelaskan pada materi sebelumnya (lihat materi Citra Biner)
- Segmentasi citra didasarkan pada nilai intensitas pixel-pixel dan nilai ambang T.
- Salah satu cara untuk mengekstrak objek dari latar belakang adalah dengan memilih ambang T.
- Setiap pixel (x, y) pada citra di mana f (x, y)> T disebut titik objek, jika tidak maka akan disebut latar belakang.
- Hasil segmentasi adalah berupa citra biner



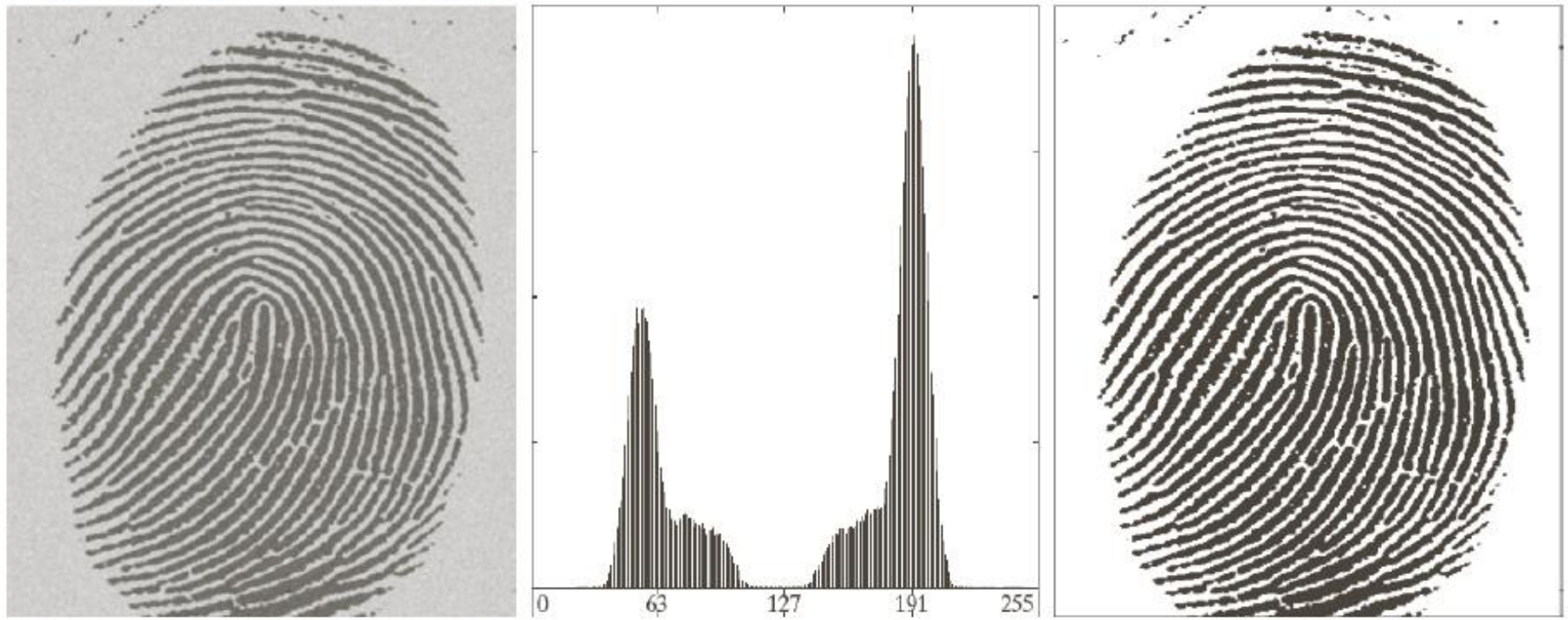
Pilih nilai ambang T

1. Pixel-pixel di atas nilai ambang mendapatkan intensitas baru A.
2. Pixels di bawah nilai ambang mendapatkan intensitas baru B.

- Untuk mendapatkan nilai ambang T, analisis histogram citra lalu identifikasi puncak dan lembah.

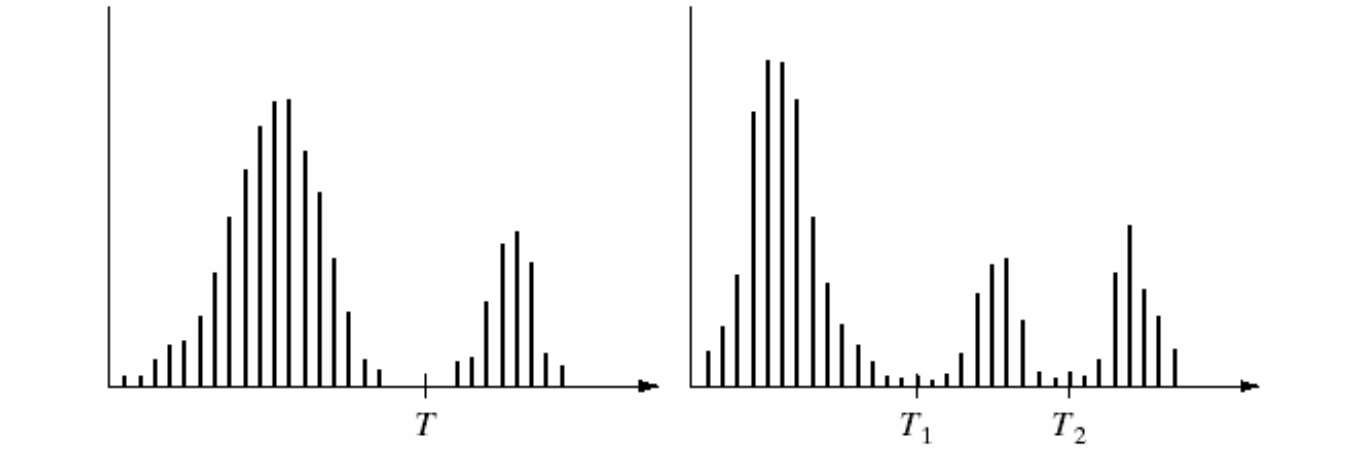


- Nilai *grayscale* pada lembah terdalam di antara dua bukit menyatakan nilai T.


0
63
127
191
255

- Mencari nilai T dengan cara sederhana di atas hanya tepat jika histogram bersifat bimodal (mempunyai dua puncak dan satu lembah). Misalnya segmentasi teks dengan latar belakangnya.

- Jika terdapat multimodal di dalam citra, maka diperlukan beberapa nilai ambang.



bimodal multimodal

- Teknik pengambangan dibagi menjadi:
  1. *Global thresholding*

     Nilai ambang bergantung pada keseluruhan nilai-nilai *pixel*

  2. *Local thresholding*

     Nilai ambang bergantung pada *pixel-pixe*l bertetangga, hanya untuk sekelompok *pixel* saja.

  3. *Adaptive thresholding*

     Nilai ambang berubah secara dinamis bergantung pada perubahan pencahayaan di dalam citra

Global thresholding

Sumber: Image segmentation
Stefano Ferrari
Universit`a degli Studi di Milano
stefano.ferrari@unimi.it

A simple algorithm:

1. Initial estimate of T
2. Segmentation using T:
   - G_1, pixels brighter than T;
   - G_2, pixels darker than (or equal to) T.
3. Computation of the average intensities m_1 and m_2 of G_1 and G_2.
4. New threshold value:

$$T_{\text{new}} = \frac{m_1 + m_2}{2}$$

5. If $|T - T_{\text{new}}| > \Delta T$, back to step 2, otherwise stop.

Pengambangan dengan Metode Otsu

Otsu's method

- Otsu's method is aimed in finding the optimal value for the global threshold.
- It is based on the interclass variance maximization.
  - Well thresholded classes have well discriminated intensity values.
- $M \times N$ image histogram:
  - L intensity levels, $[0, \ldots, L-1]$;
  - n_i #pixels of intensity i:

$$MN = \sum_{i=0}^{L-1} n_i$$

- Normalized histogram:

$$p_i = \frac{n_i}{MN}$$

$$\sum_{i=0}^{L-1} p_i = 1, \quad p_i \geq 0$$

Sumber: *Image Segmentation*, by Stefano Ferrari

Otsu's method (2)

- Using k, $0 < k < L-1$, as threshold, $T = k$:
  - two classes: C_1 (pixels in $[0, k]$) and C_2 (pixels in $[k+1, L-1]$)
  - $P_1 = P(C_1) = \sum_{i=0}^{k} p_i$, probability of the class C_1
    - $P_2 = P(C_2) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1$, probability of the class C_2
  - m_1, mean intensity of the pixels in C_1:

$$\begin{aligned} m_1 &= \sum_{i=0}^{k} i \cdot P(i|C_1) \\ &= \sum_{i=0}^{k} i \frac{P(C_1|i)P(i)}{P(C_1)} \\ &= \frac{1}{P_1} \sum_{i=0}^{k} i \cdot p_i \end{aligned}$$

where $P(C_1|i) = 1$, $P(i) = p_i$ e $P(C_1) = P_1$.

Otsu's method (3)

- Similarly, m_2, mean intensity of the pixels in C_2:

$$m_2 = \frac{1}{P_2} \sum_{i=k+1}^{L-1} i \cdot p_i$$

- Mean global intensity, m_G:

$$m_G = \sum_{i=0}^{L-1} i \cdot p_i$$

- while the mean intensity up to the k level, m:

$$m = \sum_{i=0}^{k} i \cdot p_i$$

- Hence:

$$P_1 m_1 + P_2 m_2 = m_G$$

$$P_1 + P_2 = 1$$

Otsu's method (4)

- The global variance σ_G^2:

$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 \cdot p_i$$

- The *between-class variance*, σ_B, can be defined as:

$$\begin{aligned} \sigma_B^2 &= P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 \\ &= P_1 P_2 (m_1 - m_2)^2 \\ &= \frac{(m_G P_1 - m)^2}{P_1(1 - P_1)} x \end{aligned}$$

- The *goodness* of the choice $T = k$ can be estimated as the ratio η:

$$\eta = \frac{\sigma_B^2}{\sigma_G^2}$$

Otsu's method (5)

- The quantities required for the computation of η, can be obtained from the histogram:
- Hence, for each value of k, $\eta(k)$ can be computed:

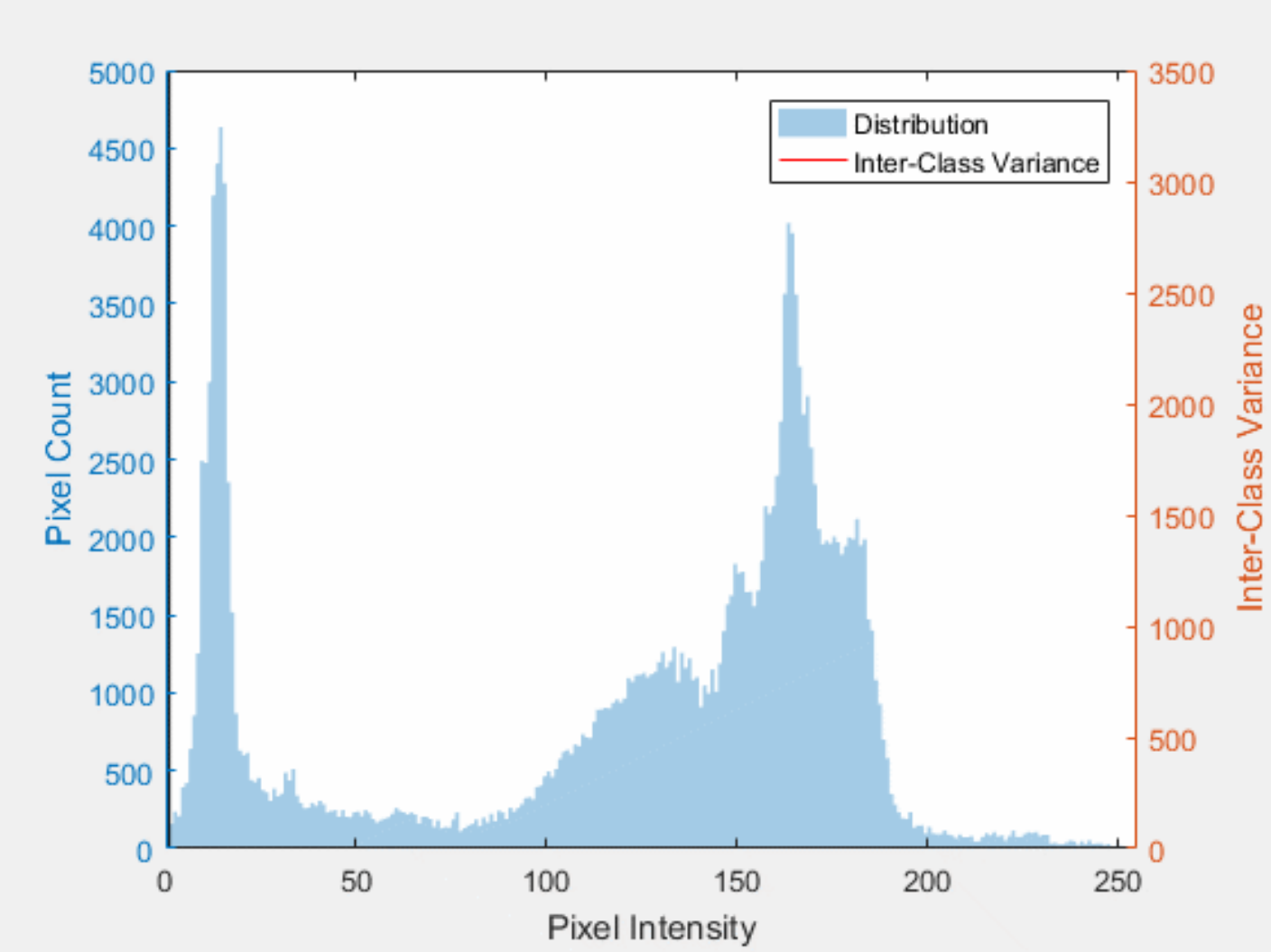
$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2}$$

where

$$\sigma_B^2(k) = \frac{(m_G P_1(k) - m(k))^2}{P_1(k)(1 - P_1(k))}$$

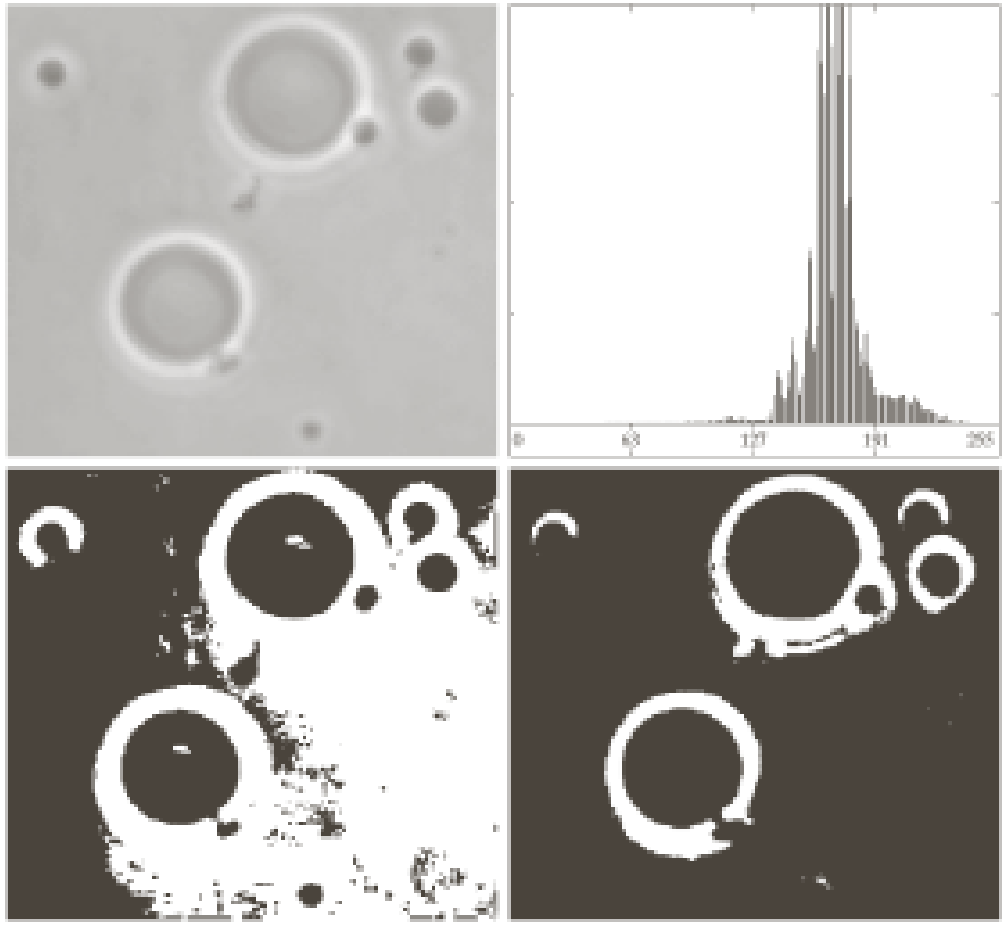
- The optimal threshold value, k^*, satisfies:

$$\sigma_B^2(k^*) = \max_{0<k<L-1} \sigma_B^2(k)$$



Visualisasi metode Otsu (Sumber: Wikipedia)

Otsu's method: an example



a	b
c	d

(a) original image;

(b) histogram of (a);

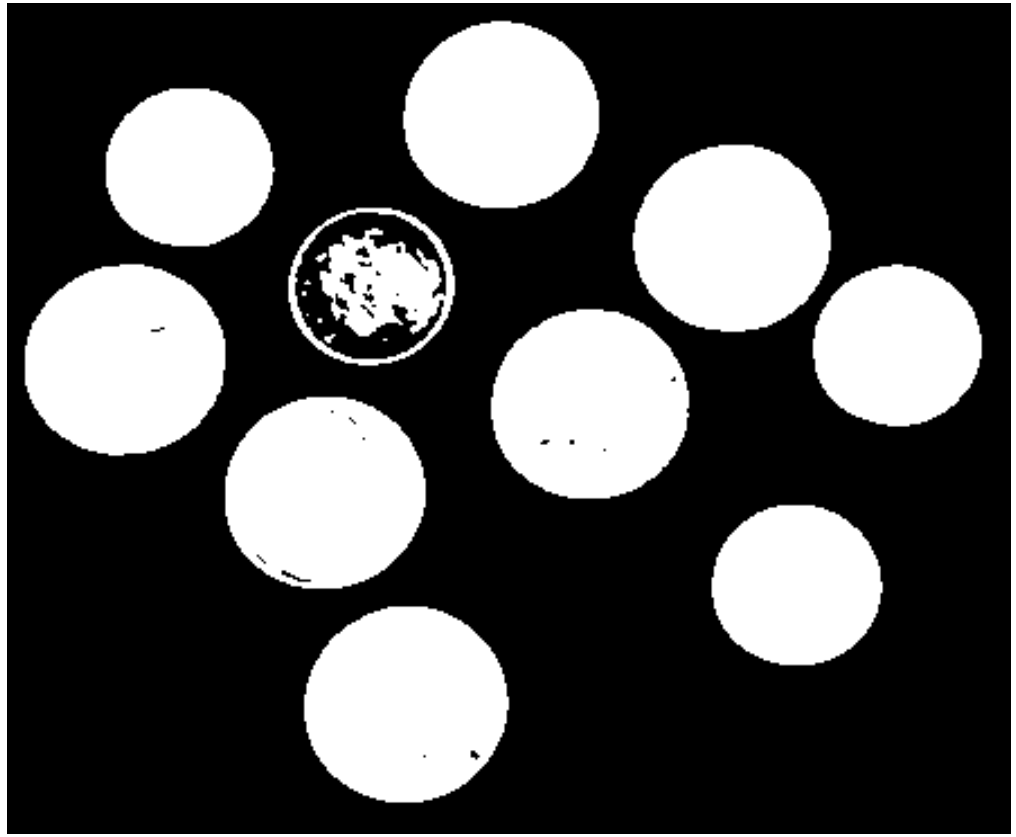
(c) global threshold: $T = 169$, $\eta = 0.467$;

(d) Otsu's method: $T = 181$, $\eta = 0.944$.

- Matlab memiliki fungsi `graythresh()` untuk melakukan pengambangan dengan metode Otsu.

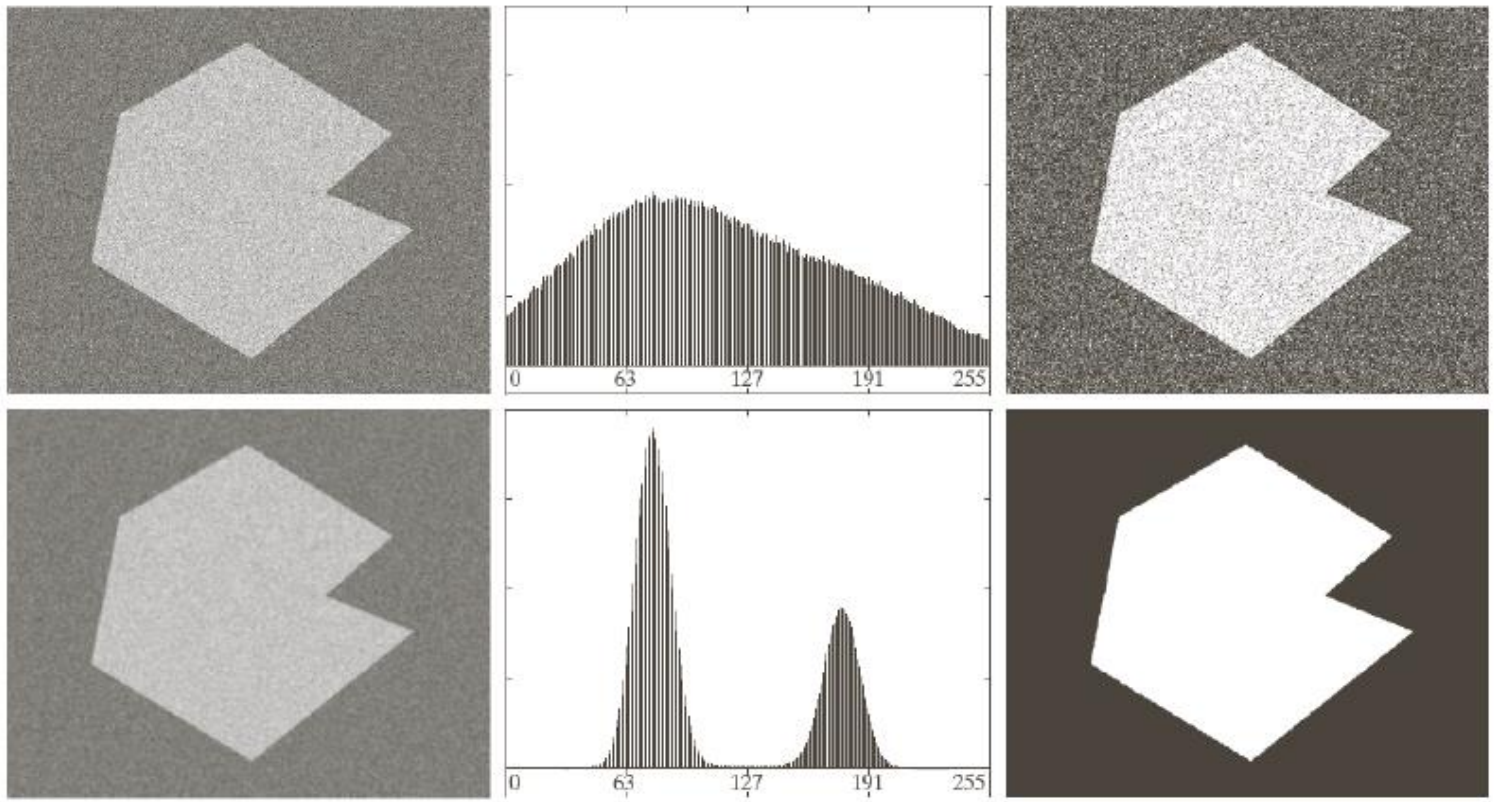
```
I = imread('house.jpg');
T = graythresh(I);
BW = im2bw(I, T);
imshow(I);
figure; imshow(BW)
```

```
I = imread('coins.bmp');
T = graythresh(I);
BW = im2bw(I, T);
imshow(I);
figure; imshow(BW)
```



Hasil pengambangan dengan metode Otsu

Smoothing



- Otsu's method may not work in presence of noise.
- Smoothing can produce a histogram with separated peaks.

Multiple thresholds Otsu's method

- The Otsu's method can be applied also for the multiple thresholds segmentation (generally, double threshold).
- Between-class variance:

$$\sigma_B^2(k_1,\, k_2) = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 + P_3(m_3 - m_G)^2$$

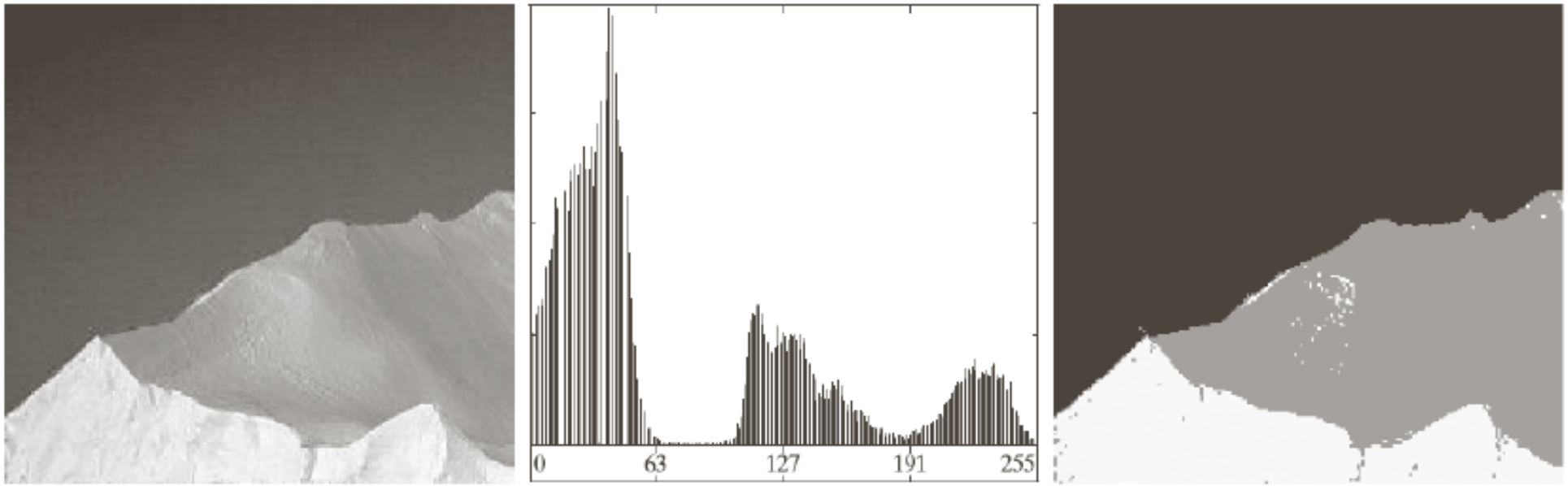
- The optimal thresholds k_1^* and k_2^* can be computed as:

$$\sigma_B^2(k_1^*,\, k_2^*) = \max_{0<k_1<k_2<L-1} \sigma_B^2(k_1,\, k_2)$$

- The separability degree can be measured as:

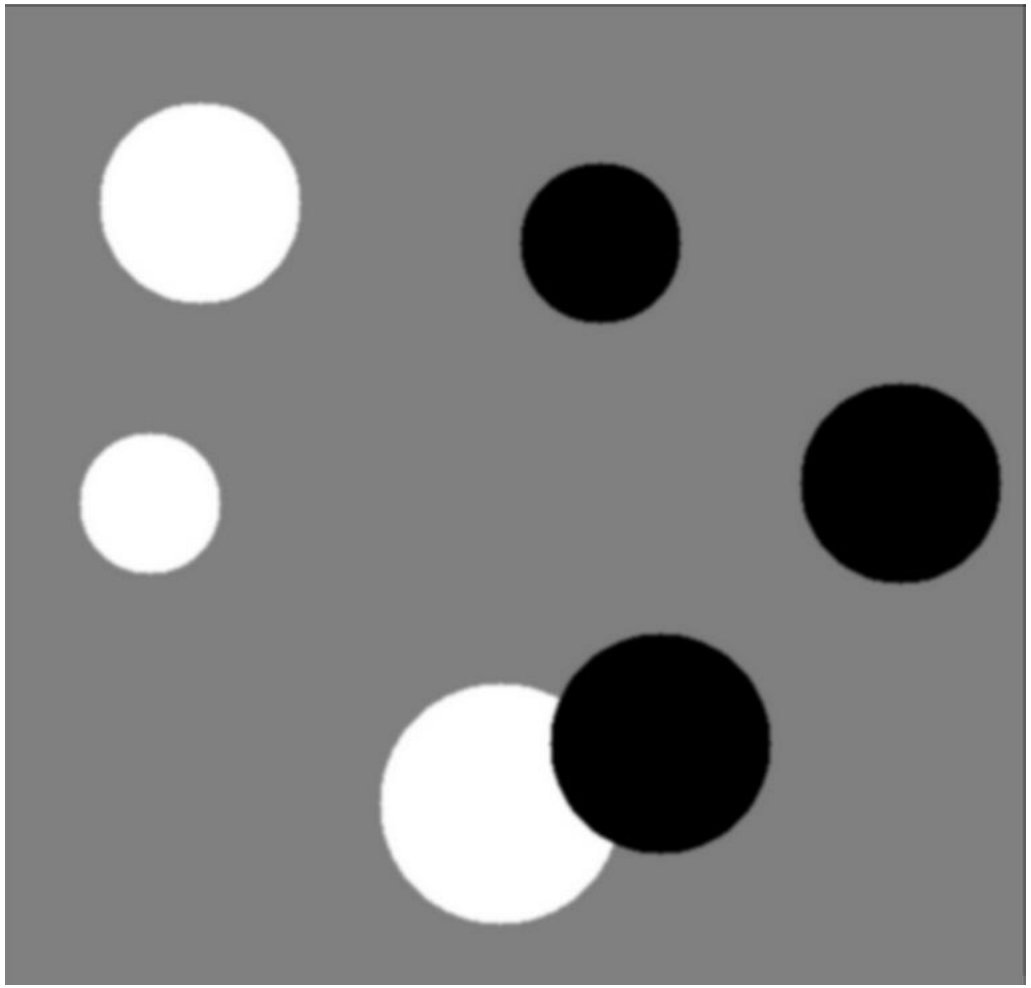
$$\eta(k_1^*,\, k_2^*) = \frac{\sigma_B^2(k_1^*,\, k_2^*)}{\sigma_G^2}$$

Multiple thresholds Otsu's method: an example

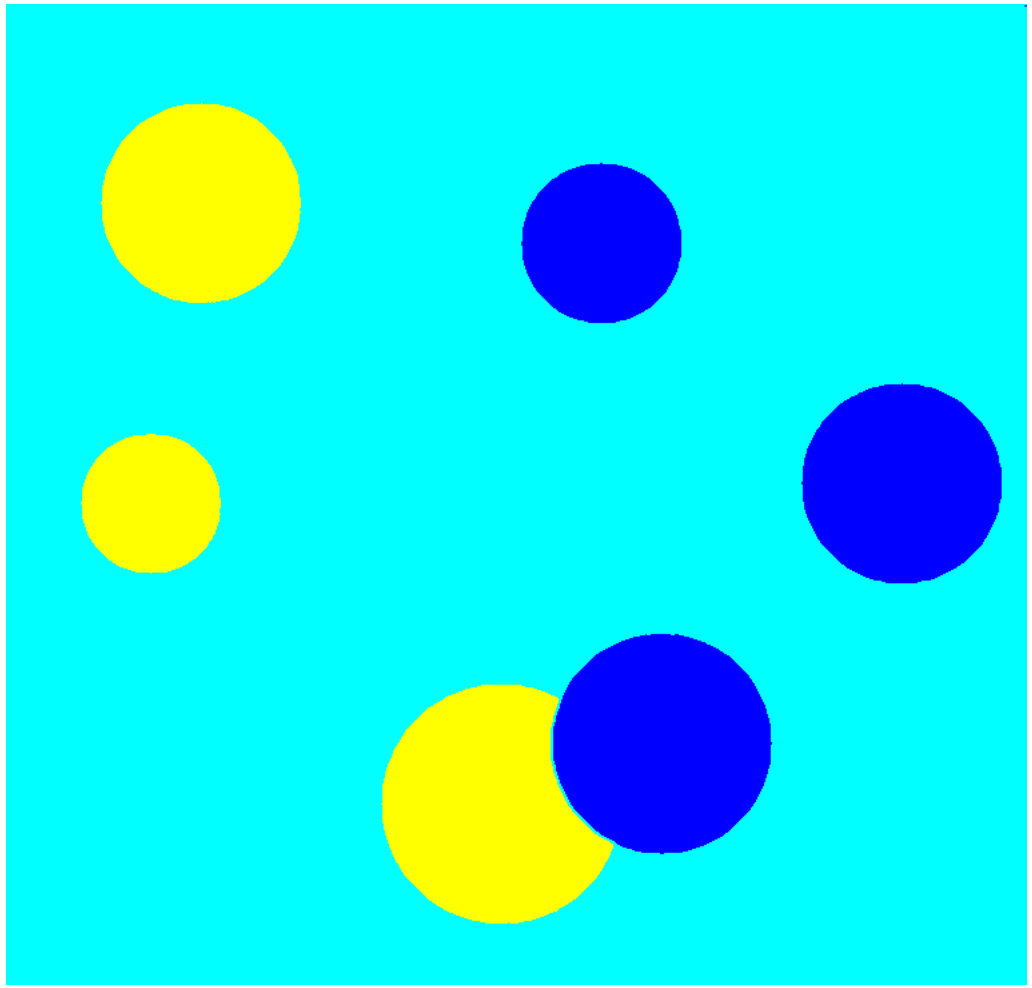


- Di dalam Matlab, fungsi `multithresh()` digunakan untuk melakukan *multiple threshold* dengan metode Otsu.

```
% Baca citra
I = imread('circle.jpg');

% Tampilkan citra
imshow(I);

% Hitung dua buah nilai ambang
thresh = multithresh(I, 2);

%Segmentasi citra menjadi tiga level dengan fungsi imquantize
seg_I = imquantize(I,thresh);

% Konversi citra yang disegmentasi menjadi citra berwarna dengan
% menggunakan fungsi label2rgb dan tampilkan
RGB = label2rgb(seg_I);
figure;  imshow(RGB)
axis off
title('RGB Segmented Image')
```



RGB Segmented Image



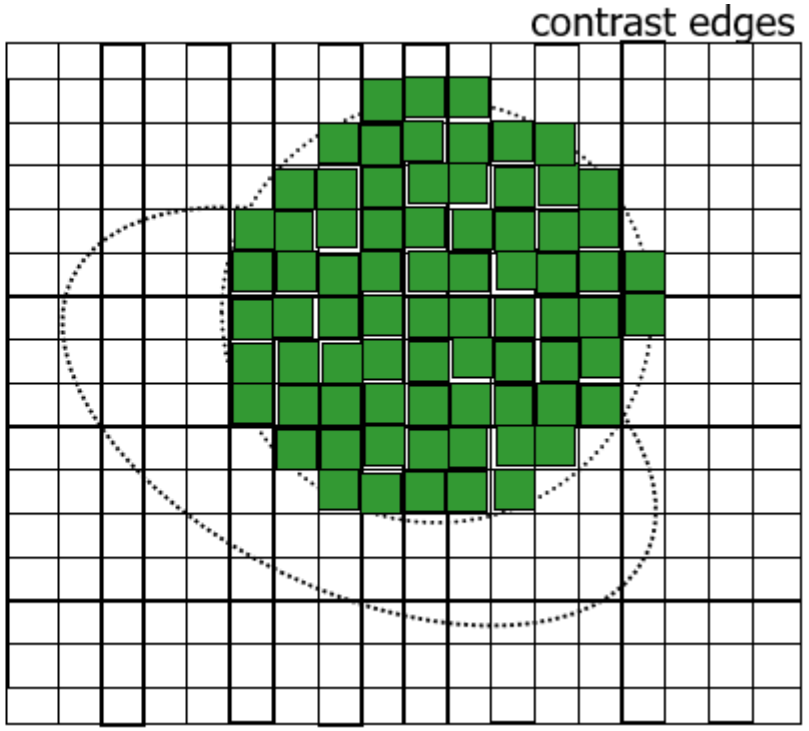
2. Region Growing

- *Region growing*: kelompok *pixel* atau sub-region yang tumbuh menjadi region yang lebih besar.

- Algoritma: Mulai dengan “umpan (*seed*)” yang berisi himpunan beranggota satu atau lebih *pixel* dari region yang potensial, dan dari sini *region* berkembang dengan menambahkan pada umpan *pixel-pixel* tetangga yang memiliki properti yang mirip dengan umpan, lalu berhenti jika *pixel-pixel* tetangga tidak mirip lagi.

- Biasanya uji statistik digunakan untuk memutuskan apakah sebuah *pixel* dapat digabungkan ke dalam region atau tidak.

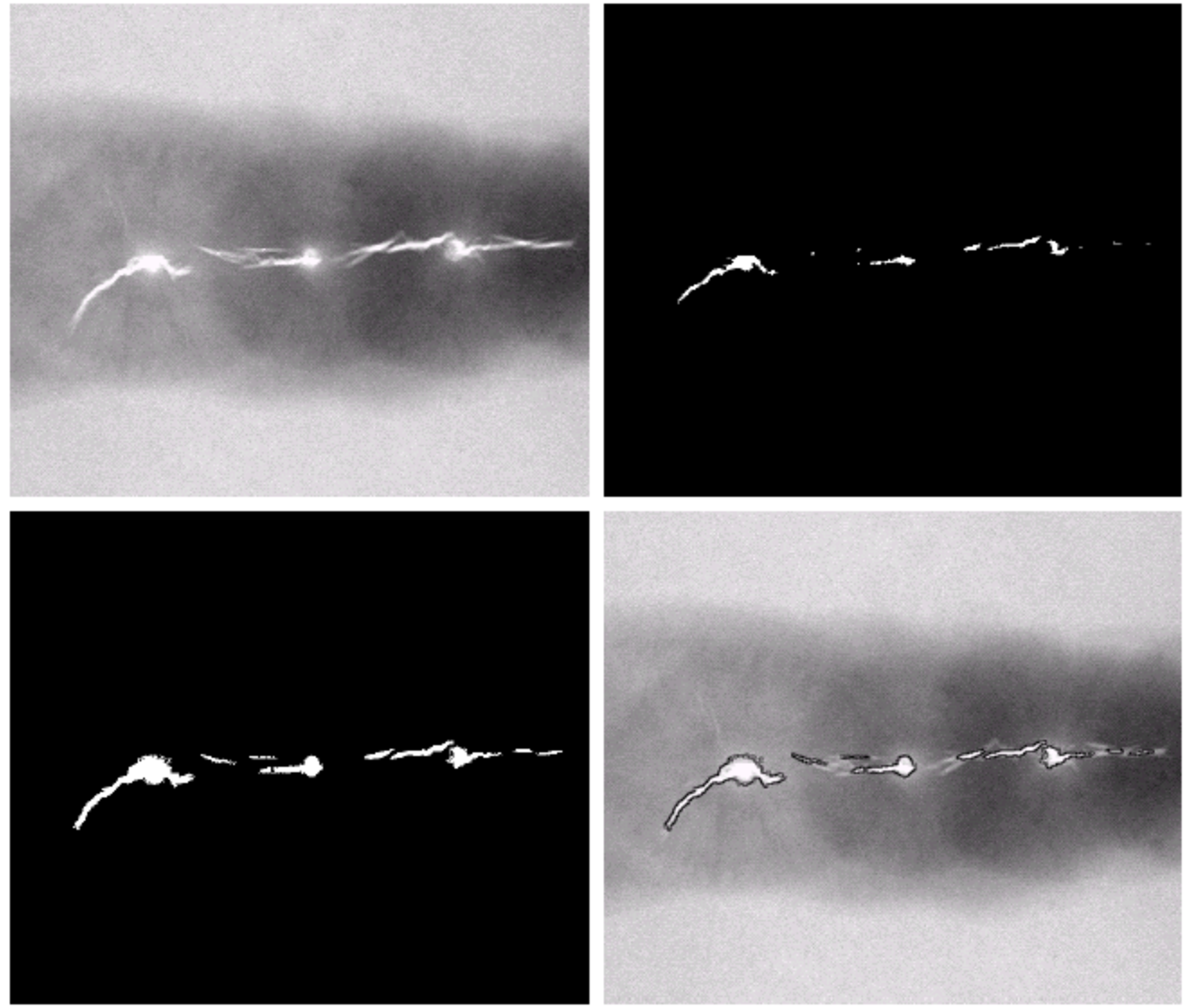
- Keuntungan: memiliki keterhubungan yang bagus antar pixel di dalam region
- Kelemahan: - pemilihan umpan yang tepat
  - kriteria berhenti
  - mengkonsumsi waktu yang lama



Sumber: CS 4487/9587 Algorithms for Image Analysis: Basic Image Segmentation

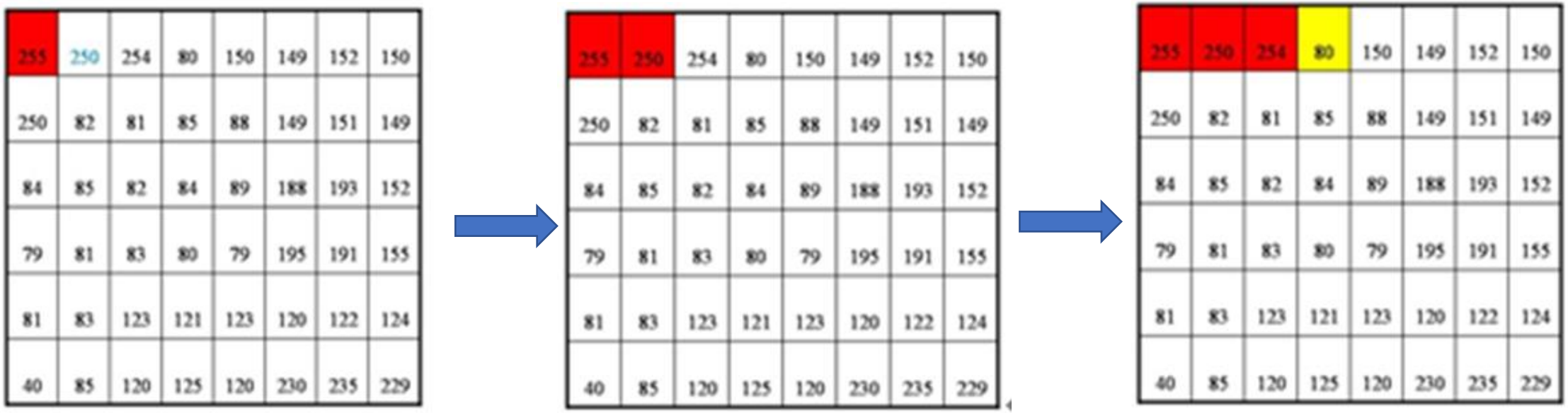
a b
c d

FIGURE 10.40
(a) Image showing defective welds. (b) Seed points. (c) Result of region growing. (d) Boundaries of segmented defective welds (in black). (Original image courtesy of X-TEK Systems, Ltd.).

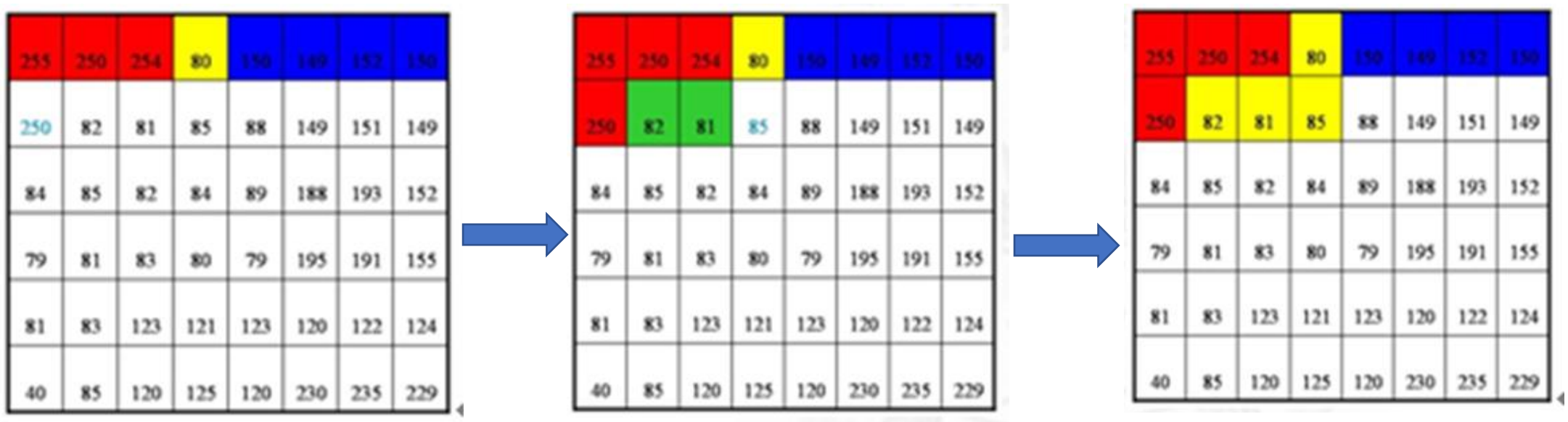


Unseeded Region Growing

- Metode *region growing* tanpa spesifikasi umpan
- Menggunakan algoritma *fast scanning*

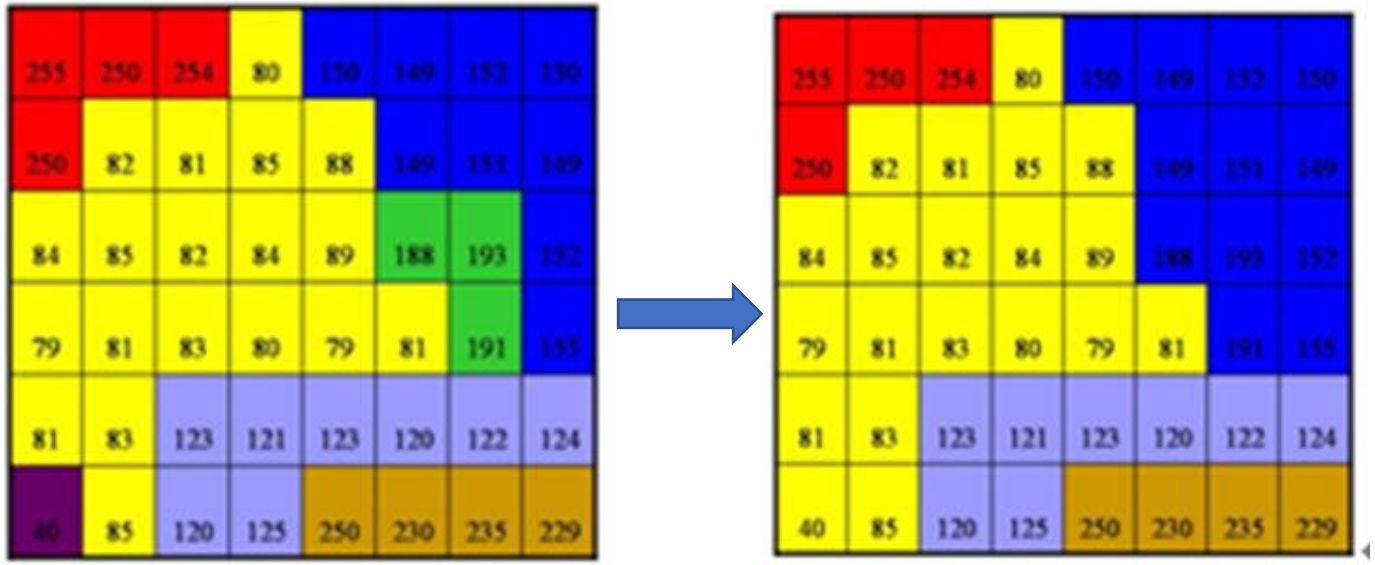


Sumber:主講人:張緯德, Image segmentation, Representation, and Description



Langkah terakhir:

Gabungkan (*merge*) region kecil menjadi region besar

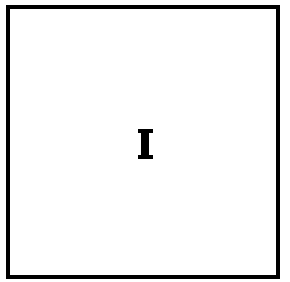


Sumber:主講人:張緯德, Image segmentation, Representation, and Description

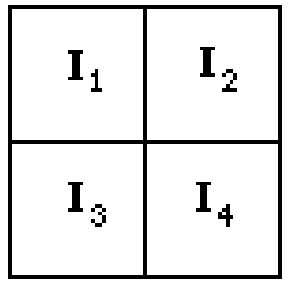
3. Split and Merge

- Mengggunakan algoritma *divide and conquer*
- Citra dibagi (split) menjadi sejumlah region yang *disjoint*

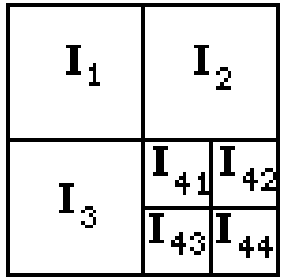
- Gabung (*merge*) region-region bertetangga yang homogen



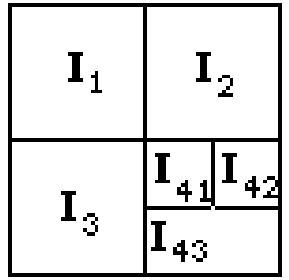
(a) Whole Image



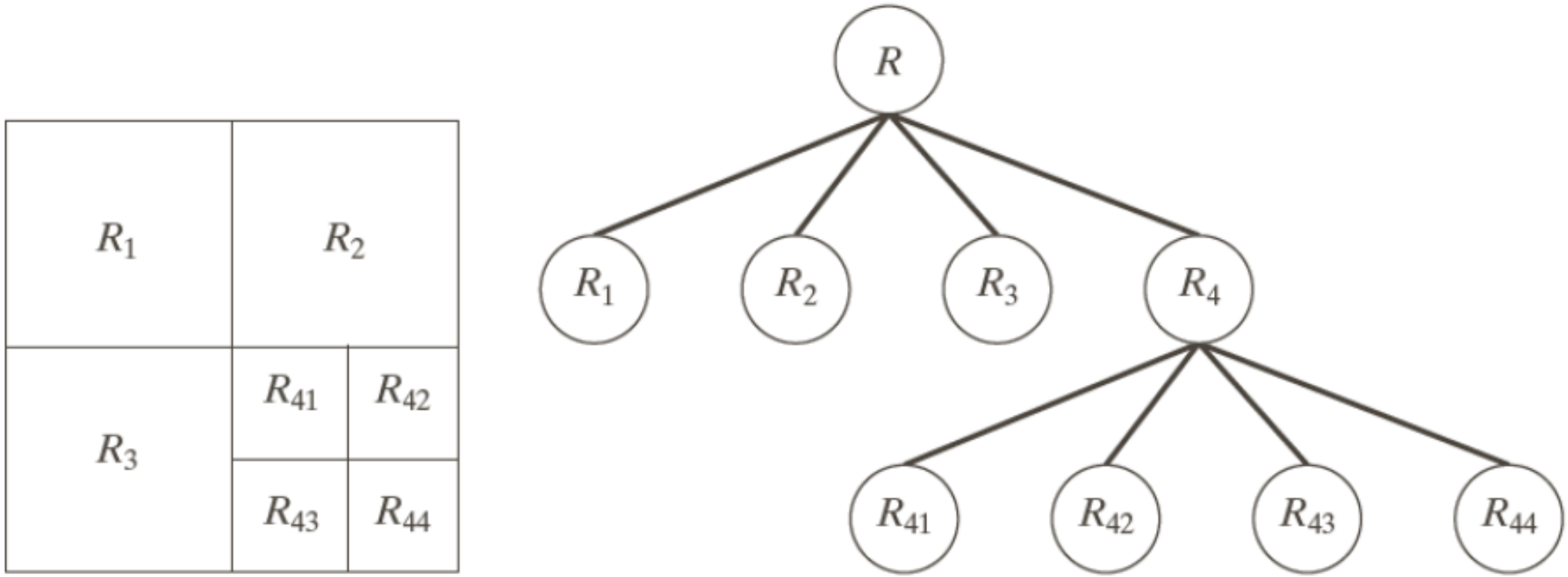
(b) First Split



(c) Second Split



(d) Merge

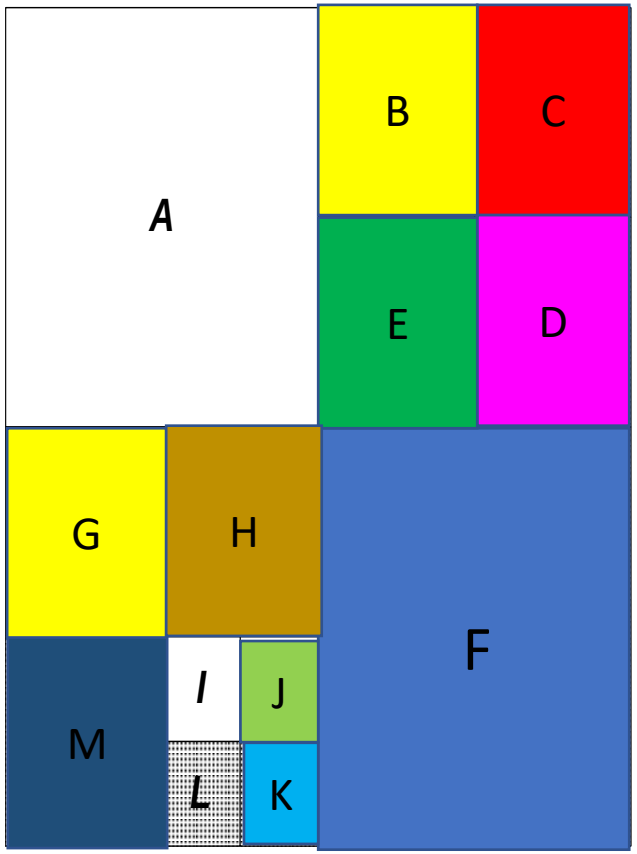

R_1
R_2
R_3
R_{41}
R_{42}
R_{43}
R_{44}
R
R_1
R_2
R_3
R_4
R_{41}
R_{42}
R_{43}
R_{44}

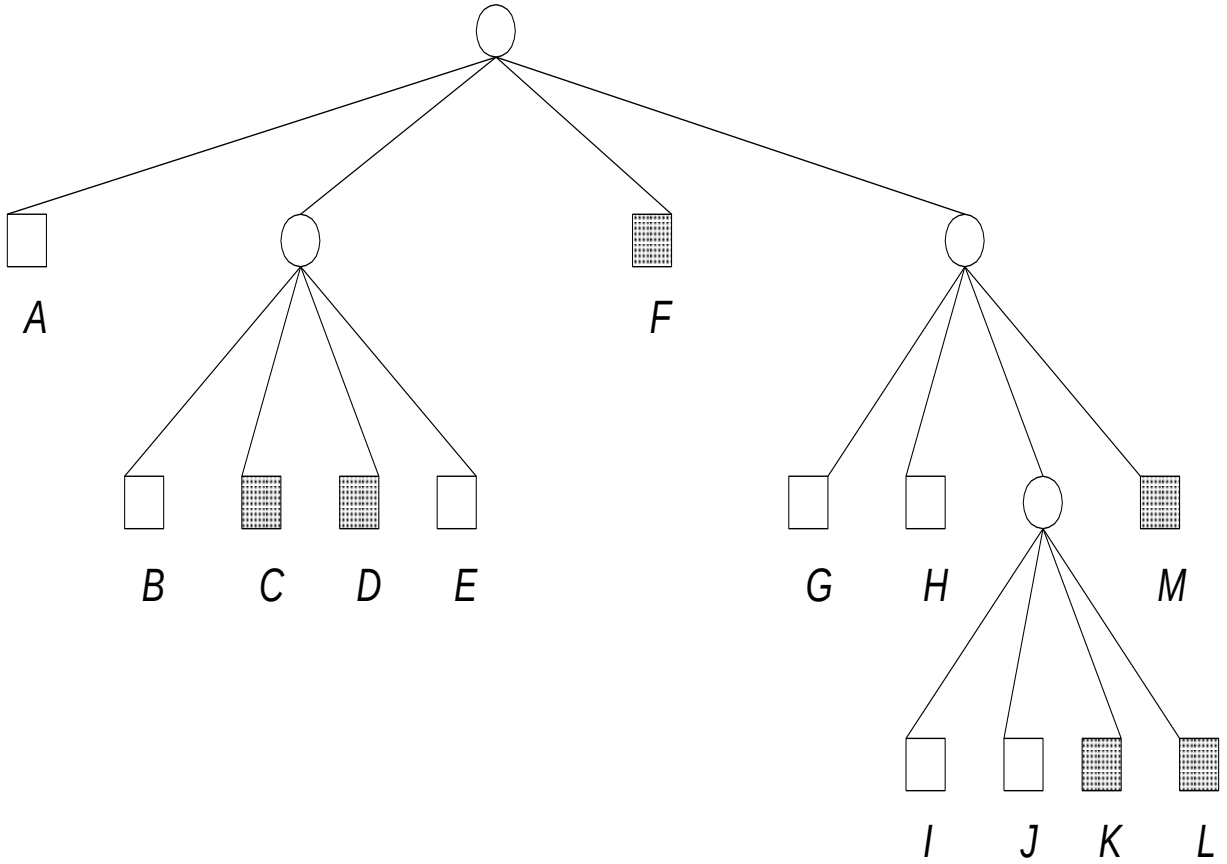
Algoritma *Split & Merge*

Given an image f and a predicate Q, the basic algorithm is:

1. $R_1 = f$
2. Subdivision in quadrants of each region R_i for which $Q(R_i) = \text{FALSE}$.
3. If $Q(R_i) = \text{TRUE}$ for every regions, merge those adjacent regions R_i and R_j such that $Q(R_i \cup R_j) = \text{TRUE}$; otherwise, repeat step 2.
4. Repeat the step 3 until no merging is possible.

Sumber: Image segmentation
Stefano Ferrari
Universit`a degli Studi di Milano
stefano.ferrari@unimi.it


A
B
C
E
D
G
H
F
M
I
J
L
K

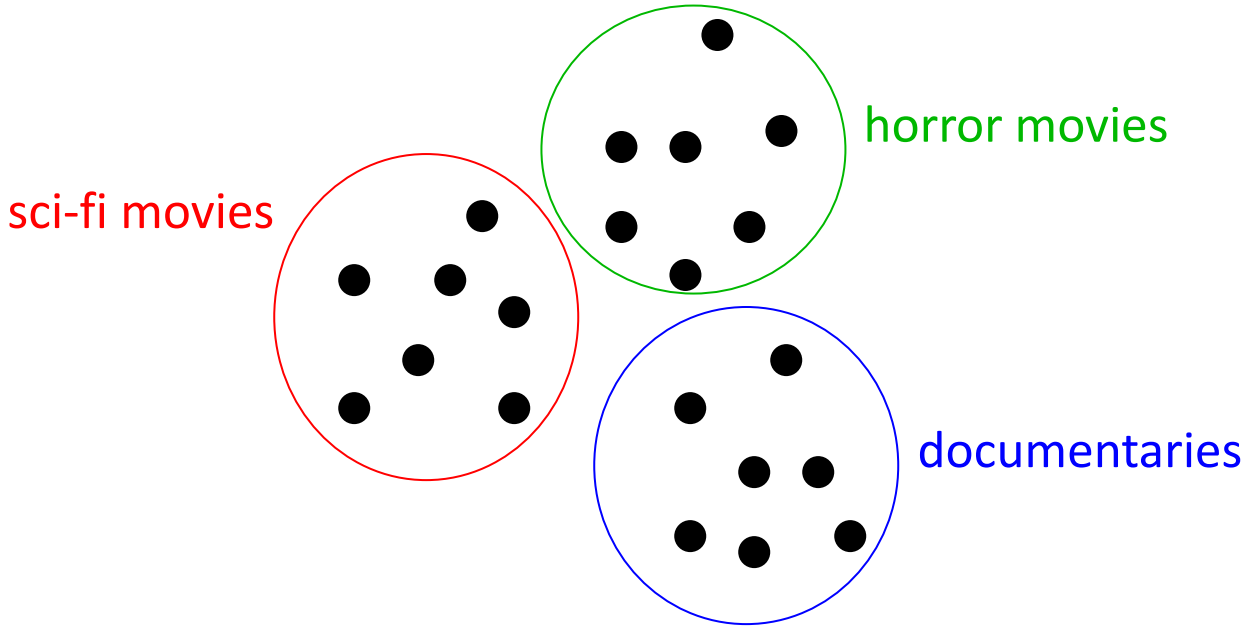

A
F
B
C
D
E
G
H
M
I
J
K
L

Sumber: Image Segmentation, by Dr. Rajeev Srivastava

4. Clustering

Prinsip *clustering* secara umum

- Misalkan terdapat N buah titik data (terokan, vektor fitur, dll), $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_N$
- Kelompokkan (*cluster*) titik-titik yang mirip dalam kelompok yang sama



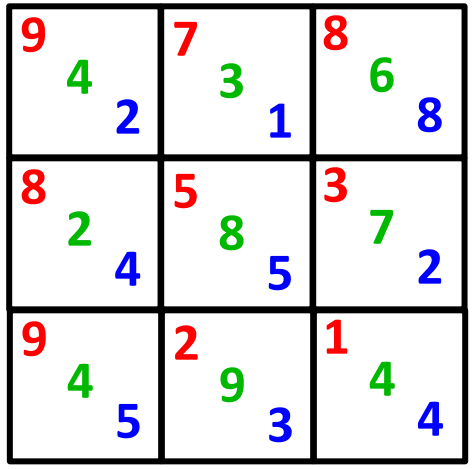
Bagaimana kaitan *clustering* pada segmentasi citra?

- Nyatakan citra sebagai vektor fitur $\boldsymbol{x}_1,\ldots,\mathbf{x}_n$
  - Sebagai contoh, setiap *pixel* dapat dinyatakan sebagai vektor:
    - Intensitas → menghasilkan vektor dimensi satu
    - Warna → menghasilkan vektor berdimensi tiga (R, G, B)
    - Warna + koordinat, → menghasilkan vektor berdimensi lima

- Kelompokkan vektor-vektor fitur ke dalam ***k*** kluster

Sumber: CS 4487/9587 Algorithms for Image Analysis: Basic Image Segmentation

citra input



Vektor fitur untuk clustering berdasarkan warna

[9 4 2] [7 3 1] [8 6 8]

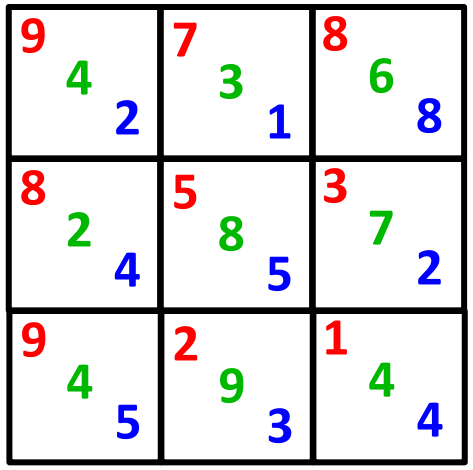
[8 2 4] [5 8 5] [3 7 2]

[9 4 5] [2 9 3] [1 4 4]

RGB (or LUV) space clustering

Sumber: CS 4487/9587 Algorithms for Image Analysis: Basic Image Segmentation

citra input



Vektor fitur untuk clustering berdasarkan warna dan koordinat pixel

[9 4 2 0 0] [7 3 1 0 1] [8 6 8 0 2]

[8 2 4 1 0] [5 8 5 1 1] [3 7 2 1 2]

[9 4 5 2 0] [2 9 3 2 1] [1 4 4 2 2]

RGBXY (or LUVXY) space clustering

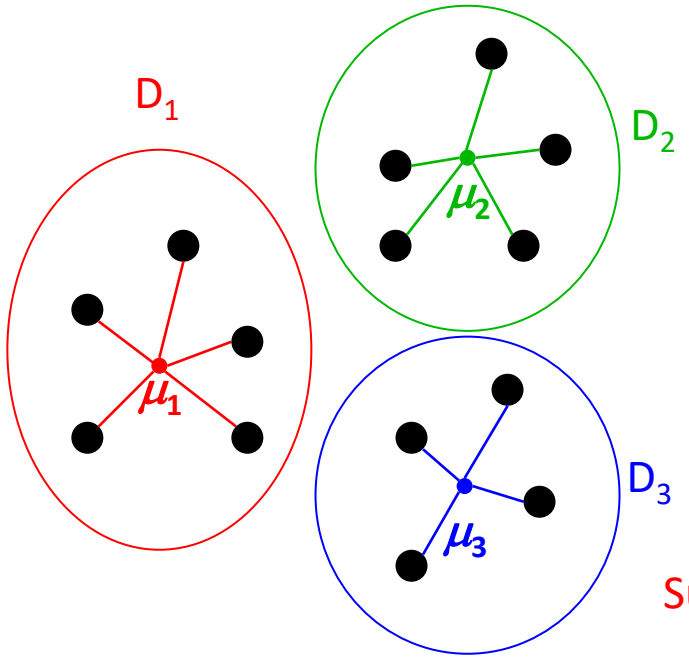
Sumber: CS 4487/9587 Algorithms for Image Analysis: Basic Image Segmentation

K-Means Clustering

- *K-means clustering* merupakan algoritma *clustering* yang paling populer
- Asumsikan jumlah cluster adalah *k*
- Mengooptimalkan (secara hampiran) fungsi objektif berikut untuk variabel D_i dan μ_i

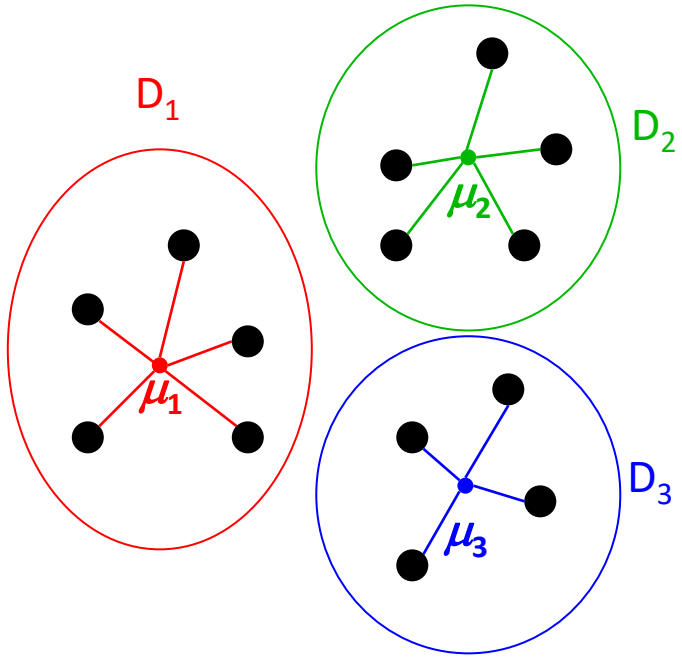
$$E_k = SSE = \sum_{i=1}^{k} \sum_{x \in D_i} \|x - \mu_i\|^2$$

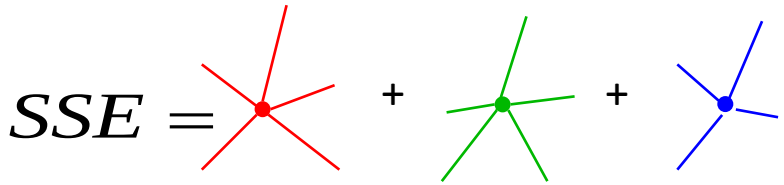
sum of squared errors dari kluster dengan pusat μ_i



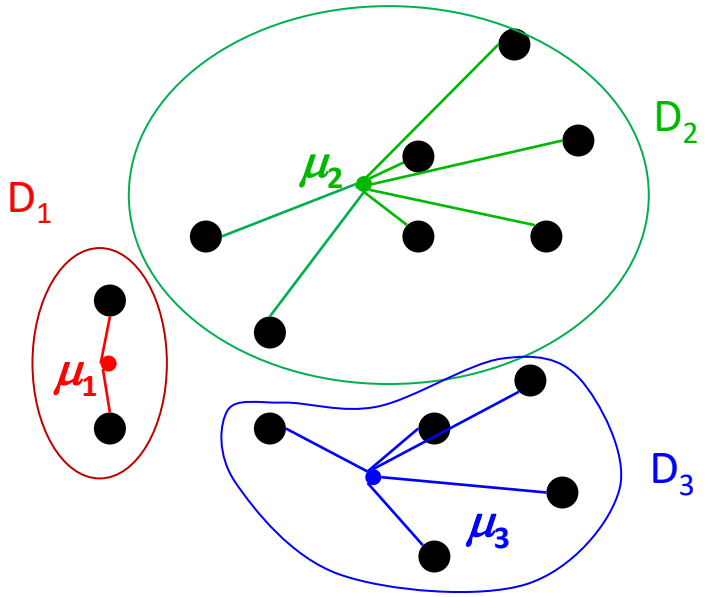
$$SSE = \quad + \quad +$$

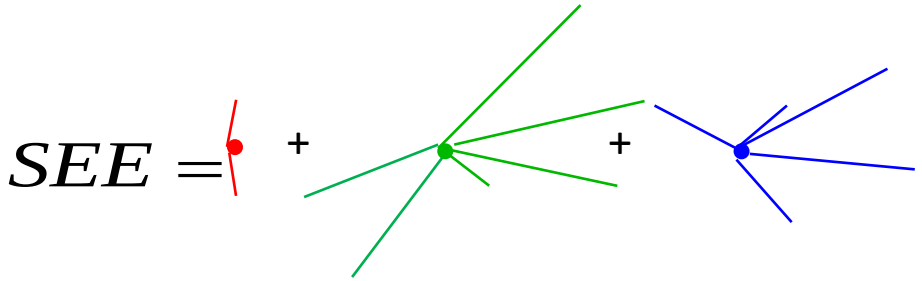
Sumber: CS 4487/9587 Algorithms for Image Analysis: Basic Image Segmentation





Good (tight) clustering
smaller value of SSE

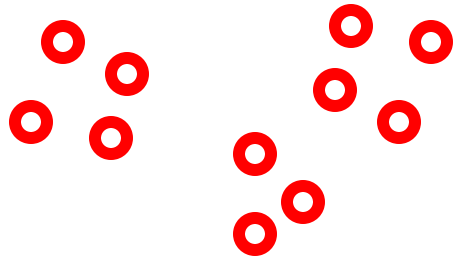




Bad (loose) clustering
larger value of *SSE*

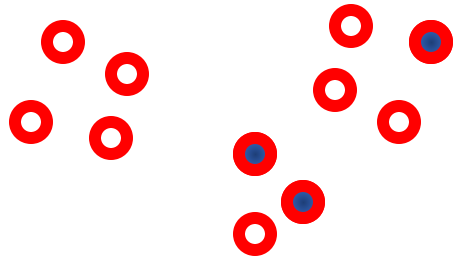
Algoritma K-means Clustering

- Initialization step
  1. pick ***k*** cluster centers randomly



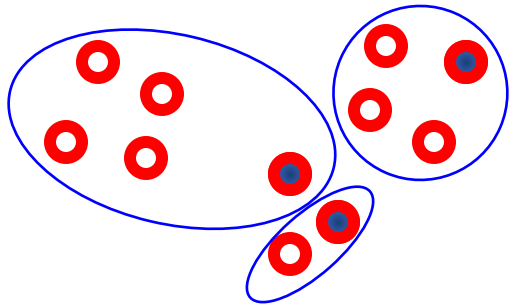
Algoritma K-means Clustering

- Initialization step
  1. pick ***k*** cluster centers randomly



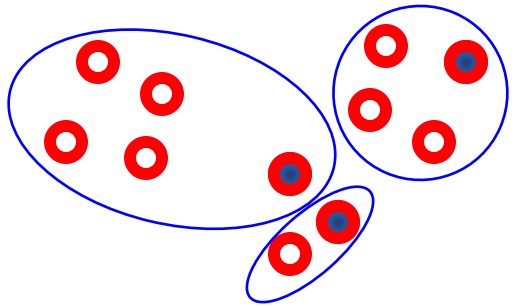
Algoritma K-means Clustering

- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center



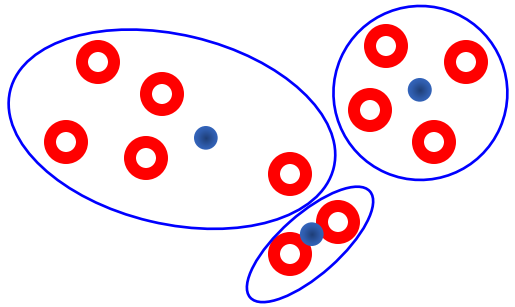
Algoritma K-means Clustering

- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center



Algoritma K-means Clustering

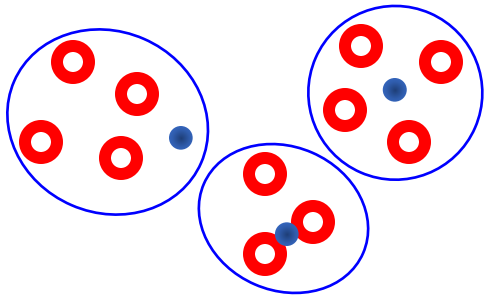
- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center



- Iteration steps
  1. compute means in each cluster $\mu_i = \frac{1}{|D_i|} \sum_{x \in D_i} x$

Algoritma K-means Clustering

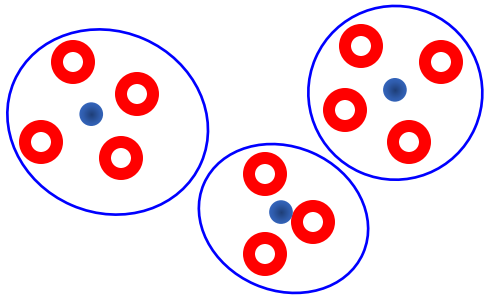
- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center



- Iteration steps
  1. compute means in each cluster $\mu_i = \frac{1}{|D_i|} \sum_{x \in D_i} x$
  2. re-assign each sample to the closest mean

Algoritma K-means Clustering

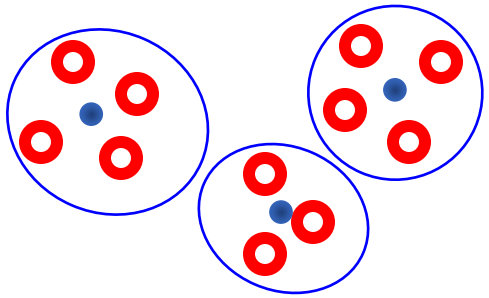
- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center



- Iteration steps
  1. compute means in each cluster $\mu_i = \frac{1}{|D_i|} \sum_{x \in D_i} x$
  2. re-assign each sample to the closest mean
- Iterate until clusters stop changing

Algoritma K-means Clustering

- Initialization step
  1. pick ***k*** cluster centers randomly
  2. assign each sample to closest center



- Iteration steps
  1. compute means in each cluster $\mu_i = \frac{1}{|D_i|} \sum_{x \in D_i} x$
  2. re-assign each sample to the closest mean
- Iterate until clusters stop changing

- This procedure decreases the value of the objective function

$$E_k(D, \mu) = \sum_{i=1}^{k} \sum_{x \in D_i} \|x - \mu_i\|^2$$

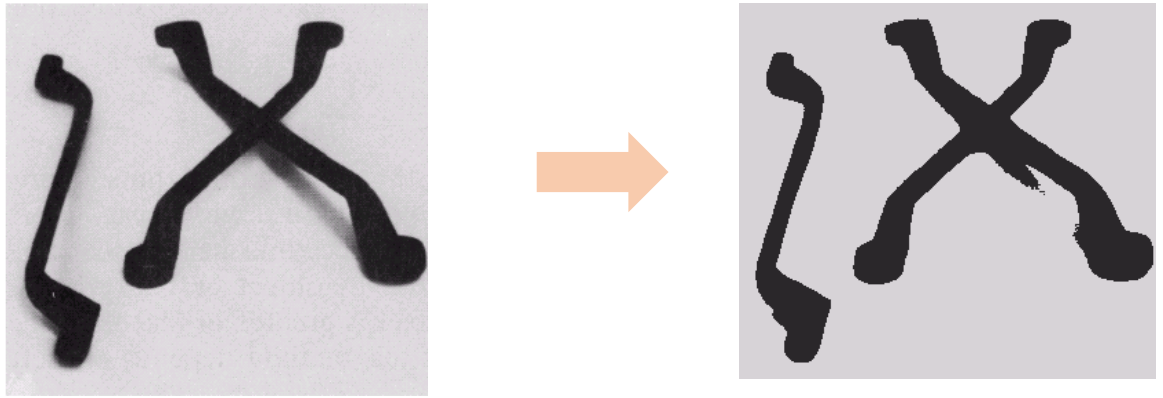
optimization variables

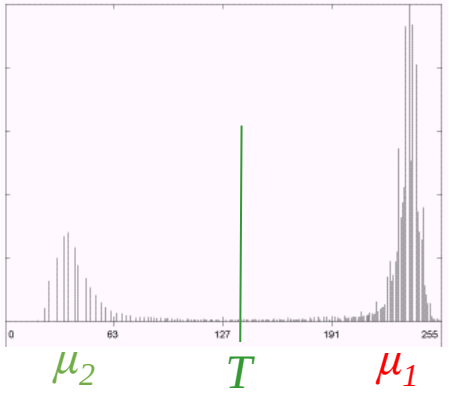
$$D = (D_1, \ldots, D_k)$$

$$\mu = (\mu_1, \ldots, \mu_k)$$

block-coordinate descent: step 1 optimizes μ, step 2 optimizes D

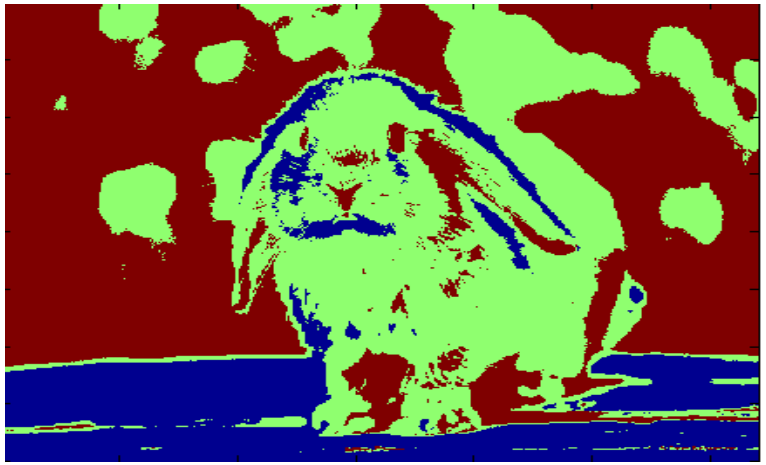
Contoh hasil *K-means clustering*





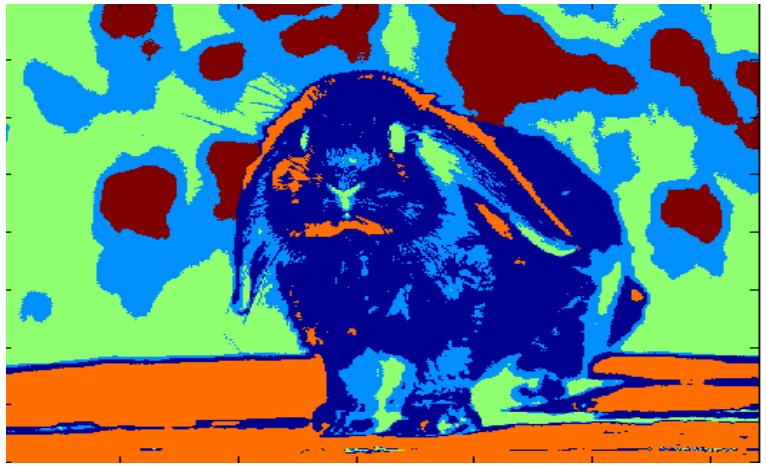
K-means menghasilkan
Pengelompokan yang kompak

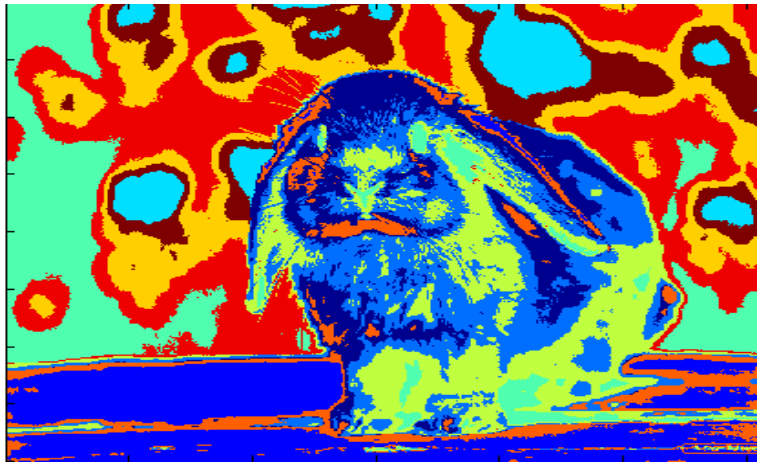
Pada kasus ini, K-means (K=2) secara otomatis menemukan nilai ambang yang bagus (antara 2 cluster

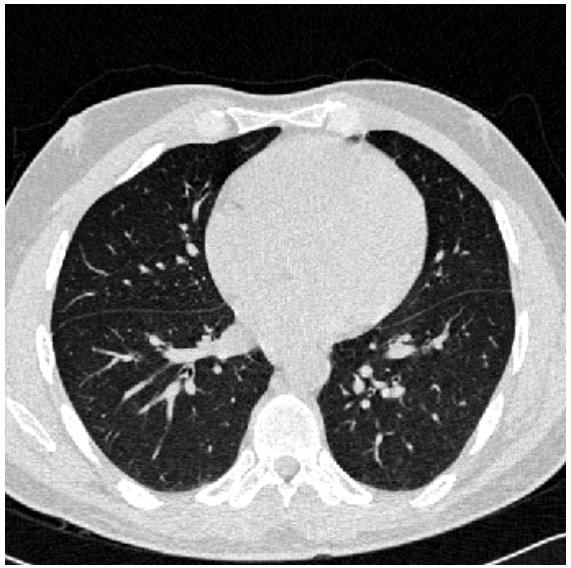


***k* = 3**

(random colors are used to better show segments/clusters)

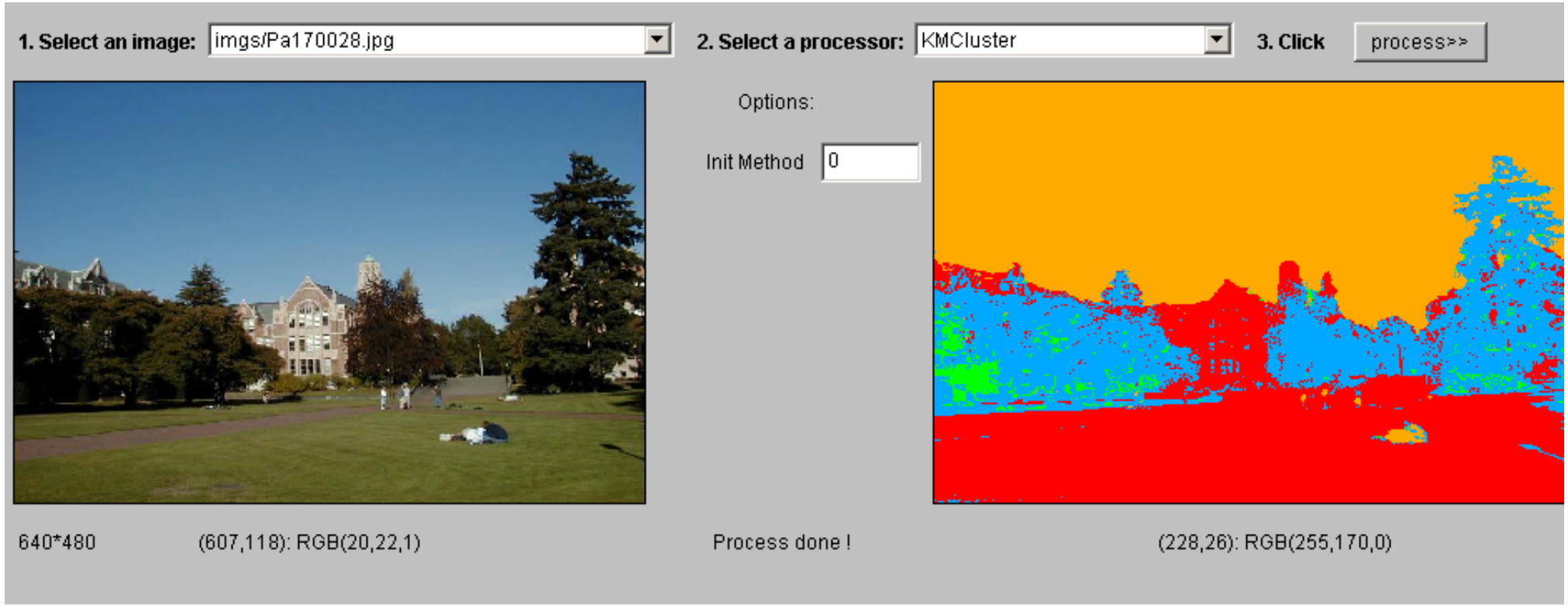


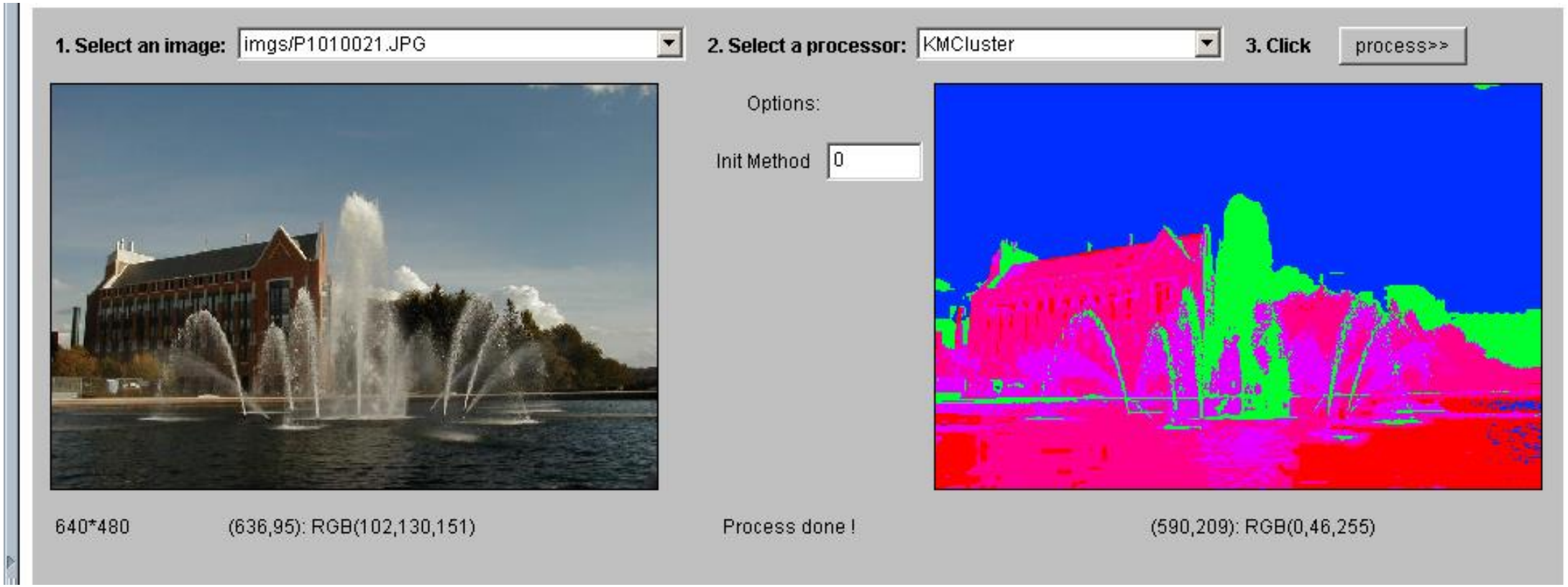




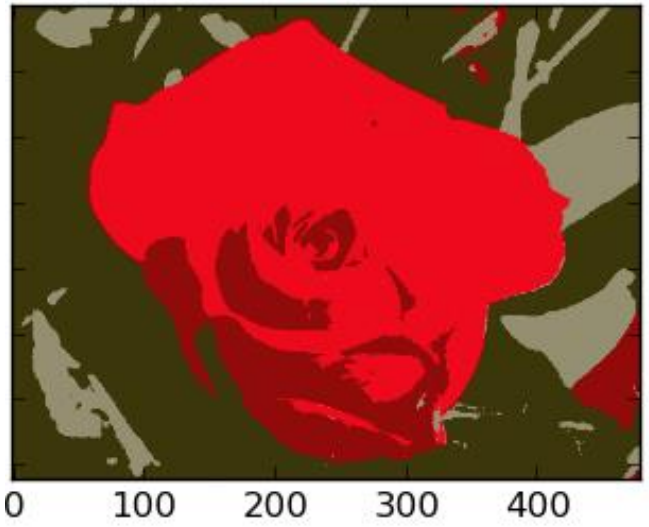
An image(I)

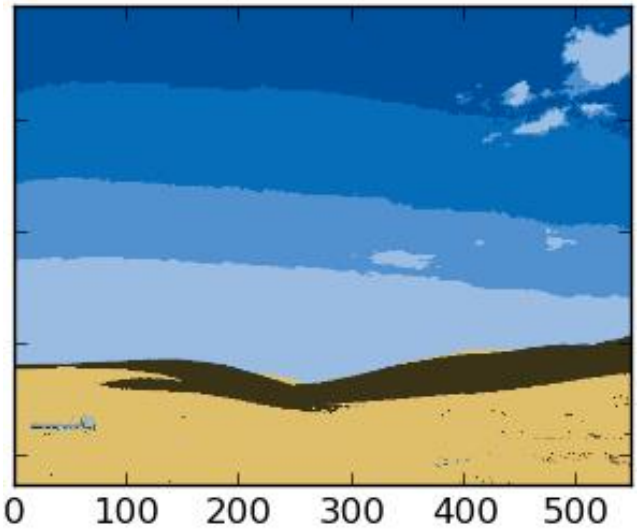
Three cluster image (J)on gray values of I


1. Select an image:
imgs/Pa170028.jpg
2. Select a processor:
KMCluster
3. Click
process>>
Options:
Init Method
0
640*480
(607,118): RGB(20,22,1)
Process done !
(228,26): RGB(255,170,0)


1. Select an image:
imgs/P1010021.JPG
2. Select a processor:
KMCluster
3. Click
process>>
Options:
Init Method
0
640*480
(636,95): RGB(102,130,151)
Process done !
(590,209): RGB(0,46,255)

Contoh hasil *K-means clustering (berdasarkan warna)*





Contoh hasil *K-means clustering (berdasarkan warna + koordinat)*

color quantization

RGB features

superpixels

RGBXY features

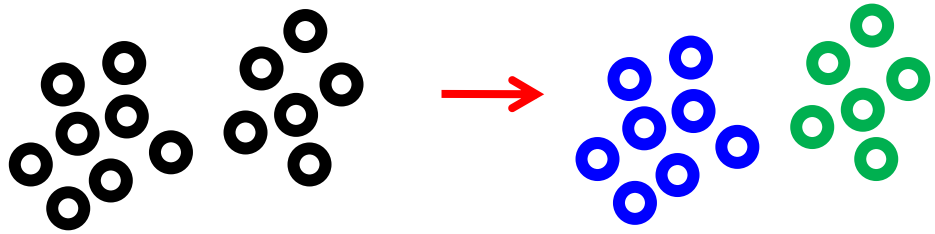
Voronoi cells

XY features only

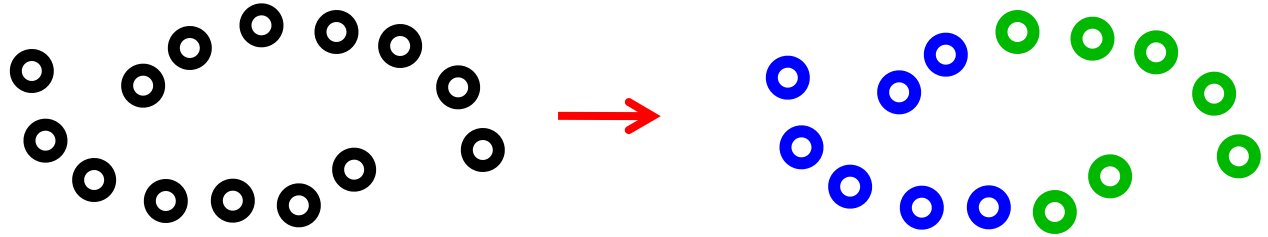
Sumber: CS 4487/9587 Algorithms for Image Analysis: Basic Image Segmentation

Sifat-sifat K-means

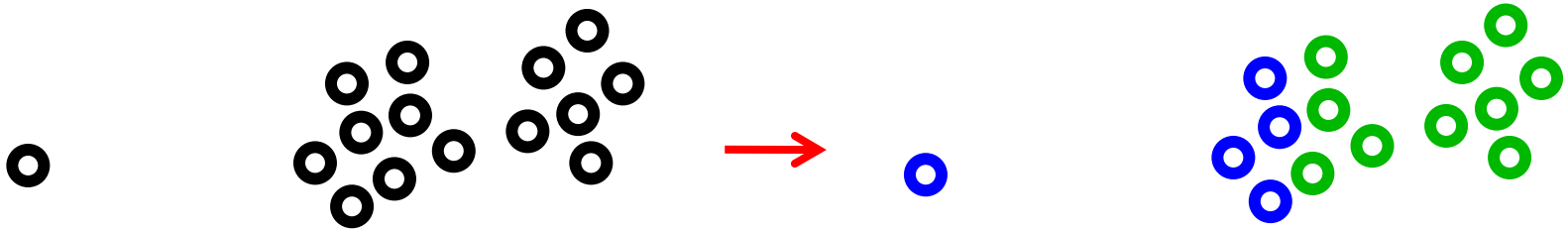
- Works best when clusters are spherical (blob like)



- Fails for elongated clusters
  - SSE is not an appropriate objective function in this case



- Sensitive to outliers



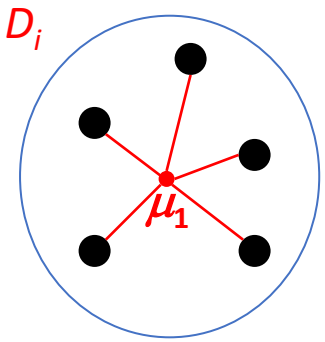
maximum likelihood (ML) fitting
of parameters μ_i (means) of Gaussian distributions

$$E_k = \sum_{i=1}^{k} \sum_{x \in D_i} \|x - \mu_i\|^2$$

equivalent (easy to check)

$$E_k \sim -\sum_{i=1}^{k} \sum_{x \in D_i} \log P(x \mid \mu_i) + const$$

Gaussian distribution $P(x \mid \mu_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\| x - \mu_i \|^2}{2\sigma^2} \right)$

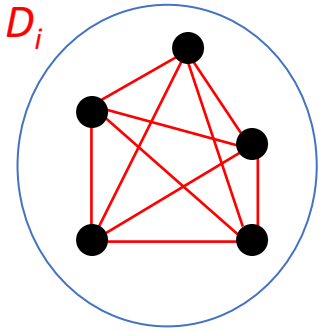


just plug-in expression

$$\mu_i = \frac{1}{|D_i|} \sum_{y \in D_i} y$$

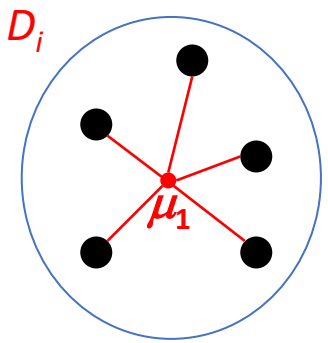
$$E_k = \sum_{i=1}^{k} \sum_{x \in D_i} \| x - \mu_i \|^2$$

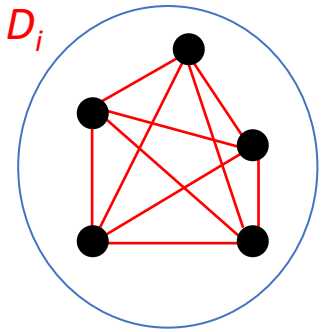
equivalent (easy to check)



$$E_k = \sum_{i=1}^{k} \sum_{x,y \in D_i} \frac{\| x - y \|^2}{2 \cdot | D_i |}$$

sample variance: $\mathrm{var}(D_i) = \frac{1}{|D_i|} \sum_{x \in D_i} \| x - \mu_i \|^2 = \frac{1}{2|D_i|^2} \sum_{x,y \in D_i} \| x - y \|^2$





both formulas can be written as

$$E_k = \sum_{i=1}^{k} |D_i| \cdot \mathrm{var}(D_i)$$

sample variance: $\mathrm{var}(D_i) = \frac{1}{|D_i|} \sum_{x \in D_i} \| x - \mu_i \|^2 = \frac{1}{2|D_i|^2} \sum_{x,y \in D_i} \| x - y \|^2$

Rangkuman K-means

- Advantages
  - Principled (objective function) approach to clustering
  - Simple to implement (the approximate iterative optimization)
  - Fast
- Disadvantages
  - Only a local minimum is found (sensitive to initialization)
  - May fail for non-blob like clusters
  - Sensitive to outliers
  - Sensitive to choice of ***k***

K-means fits Gaussian models

Quadratic errors are such

Can add sparsity term and make *k* an additional variable

$$E = \sum_{i=1}^{k} \sum_{x \in D_i} \|x - \mu_i\|^2 \quad + \quad \gamma \cdot |\, k \,|$$

Akaike Information Criterion (AIC) or
Bayesian Information Criterion (BIC)