22 - Segmentasi Citra (Bagian 1)

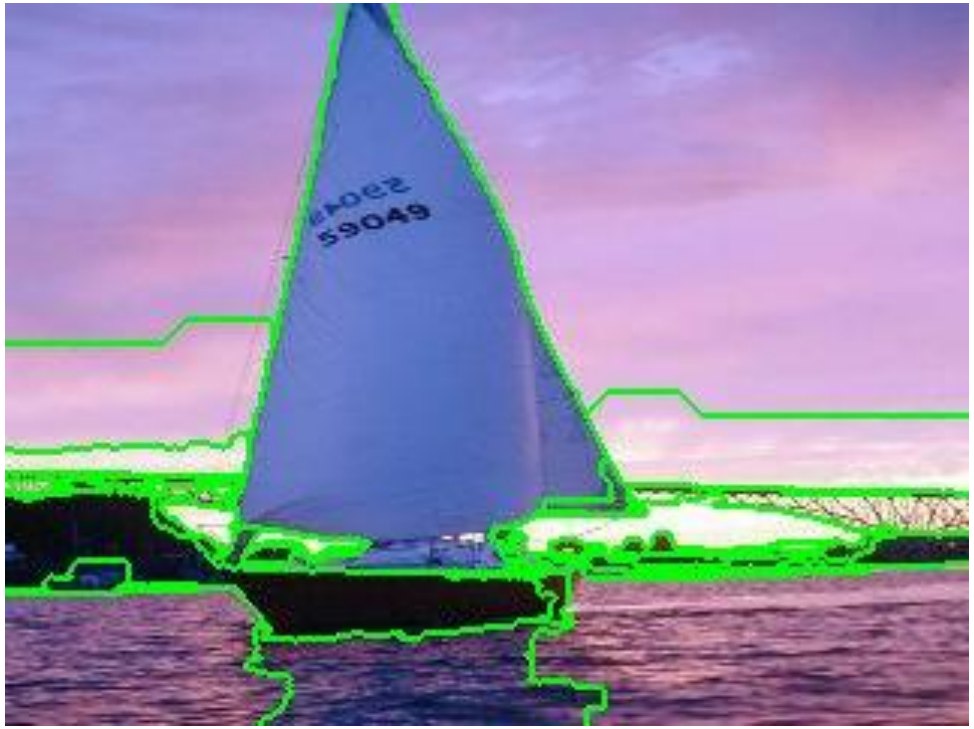
IF4073 Interpretasi dan Pengolahan Citra

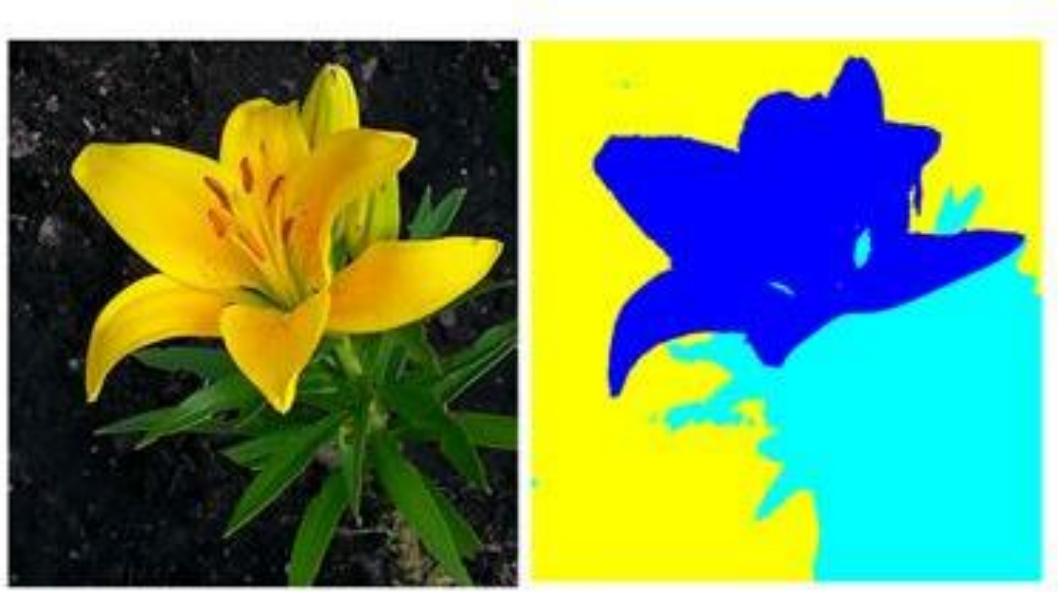
Oleh: Rinaldi Munir

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2023

Segmentasi Citra

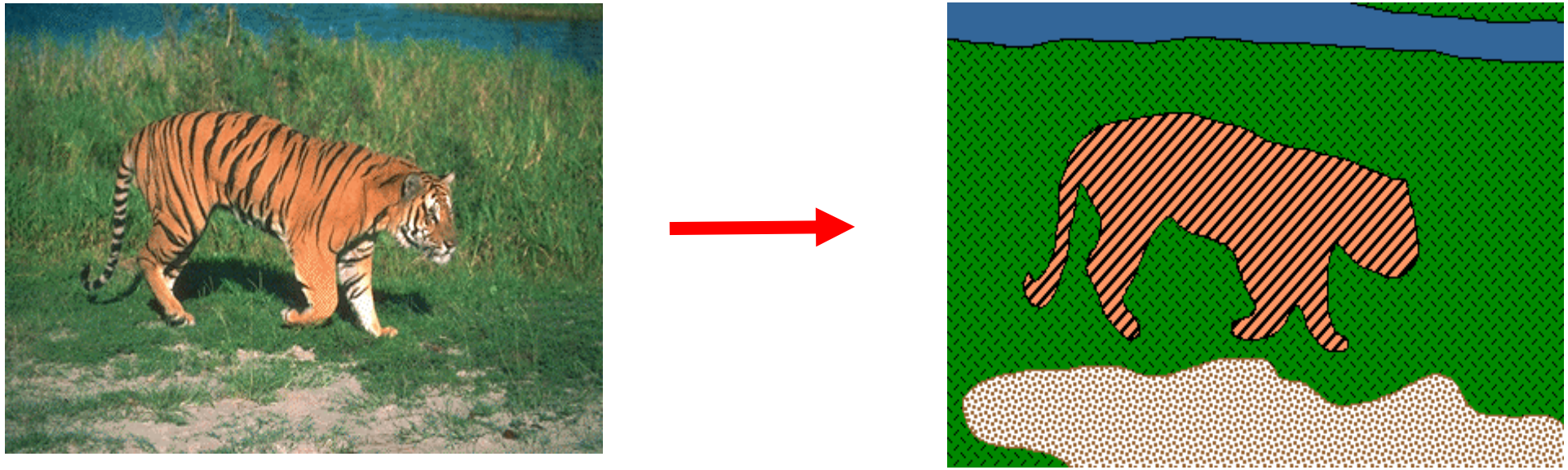
- Segmentasi citra adalah prosesi mempartisi citra menjadi sejumlah region atau objek, setiap region terdiri dari sekumpulan pixel yang terhubung satu sama lain.





- Segmentasi citra (*image segmentation*) menjadi sejumlah region bertujuan untuk:

  1. membagi citra menjadi segmen-segmen atau objek-objek yang berbeda.
  2. memisahkan objek dengan latar belakang



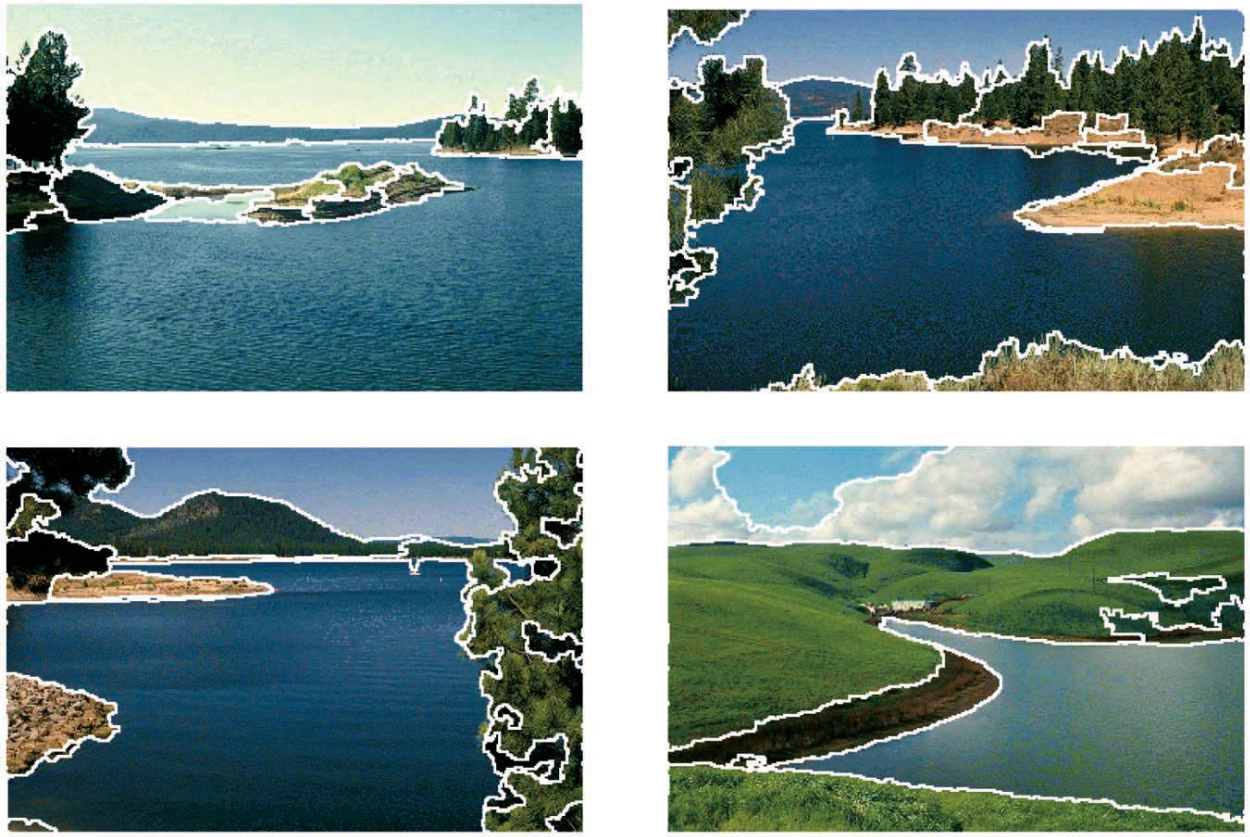
- Dengan membagi citra menjadi sejumlah segmen, kita dapat memproses hanya segmen penting atau segmen tertentu di dalam citra daripada memproses seluruh bagian citra

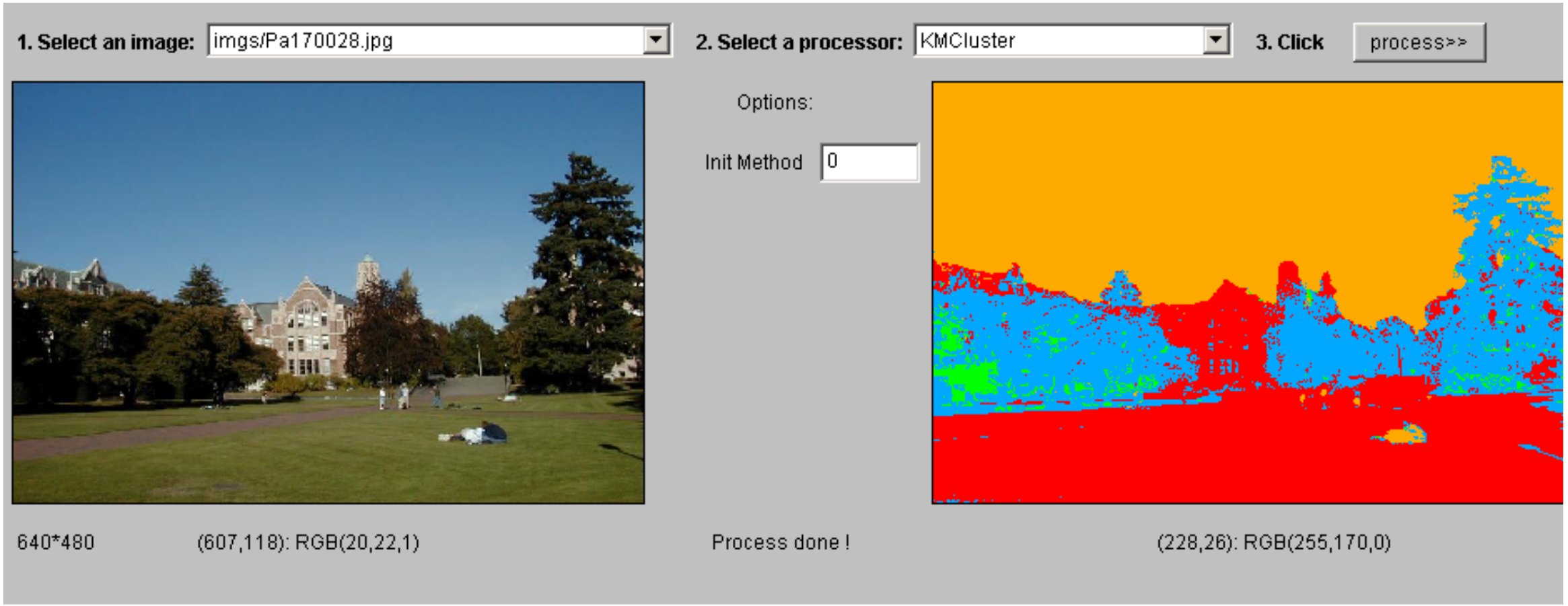
- Goal segmentasi citra adalah menemukan bagian citra yang koheren atau objek spesifik.

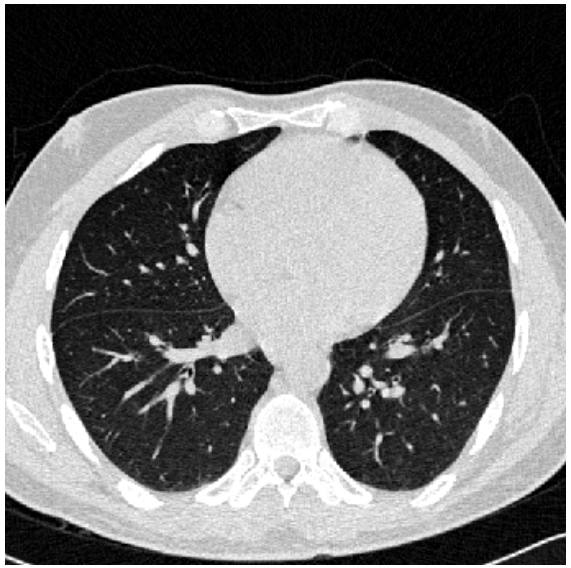
- Citra disegmentasi berdasarkan properti yang dipilih seperti kecerahan, warna, tekstur, dan sebagainya.

- Segmentasi membagi citra menjadi sejumlah segmen yang terhubung, tiap segmen 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)



Citra medis

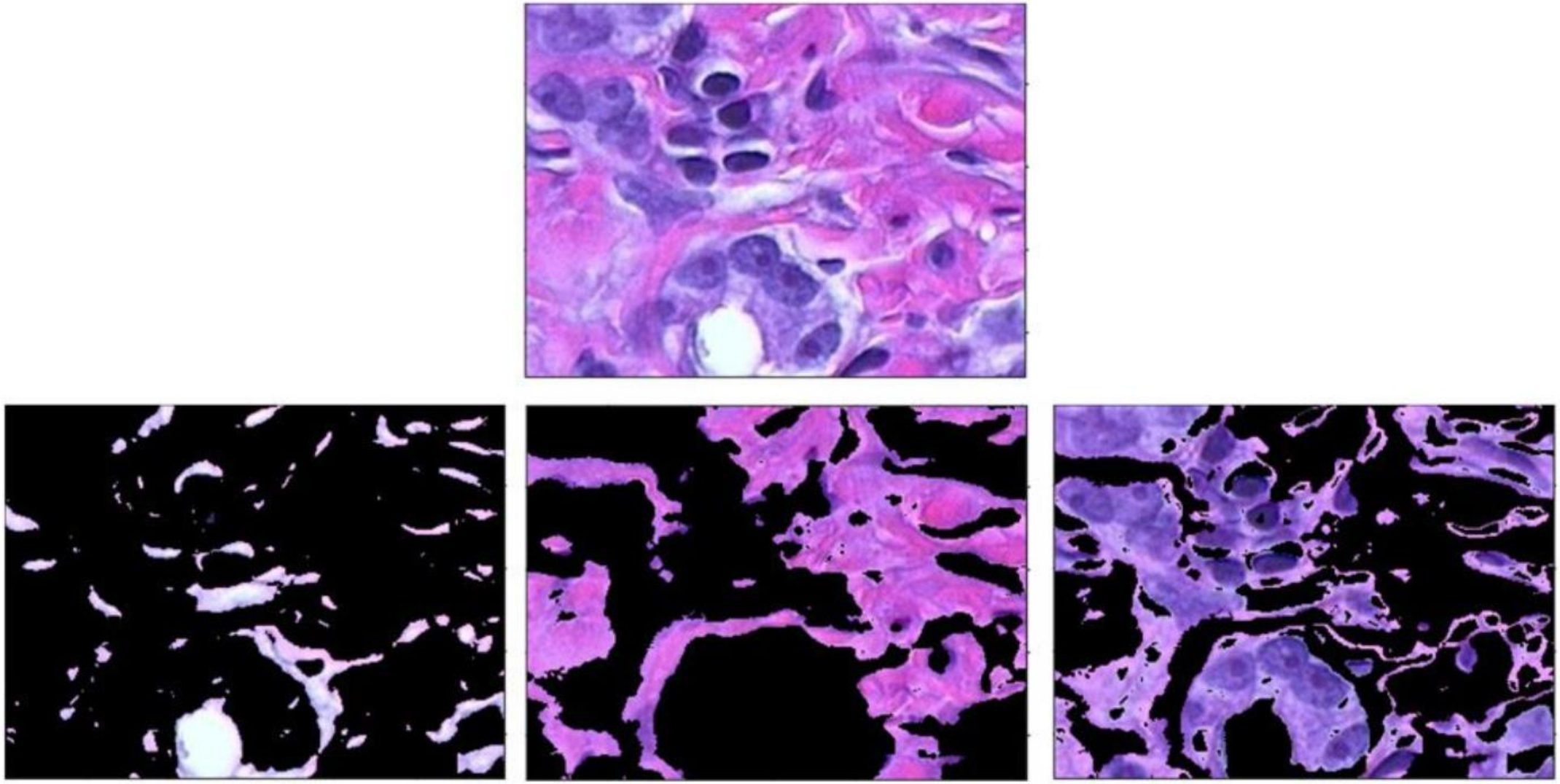
Hasil segmentasi

Beberapa aplikasi segmentasi citra

1. **Medical imaging**

   Selama diagnosis medis untuk kanker, ahli patologi menginjeksi jaringan tubuh dengan *hematoxylin* dan *eosin* (H&E) untuk membedakan jenis jaringan.

   Mereka kemudian menggunakan teknik segmentasi gambar yang disebut *clustering* untuk mengidentifikasi jenis jaringan tersebut dalam gambar mereka.

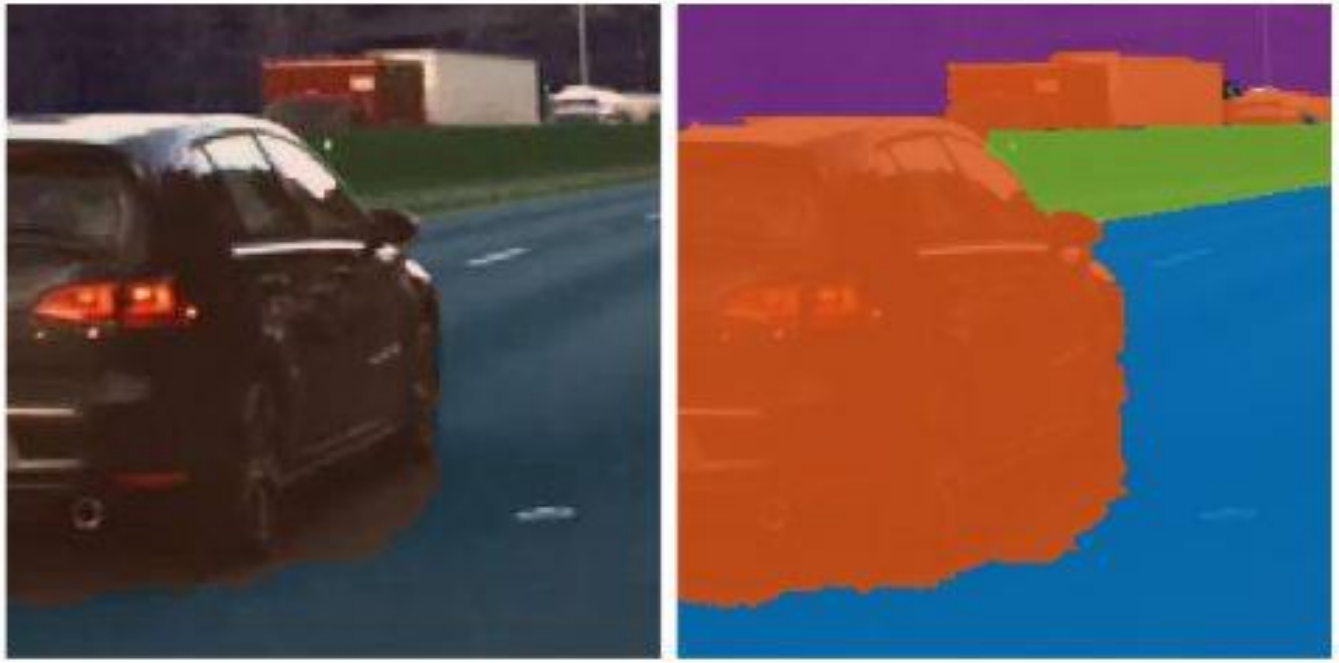


Menggunakan clustering untuk membedakan jenis jaringan (bawah) pada citra jaringan tubuh (atas) yang diwarnai dengan hematoxylin dan eosin (H&E).

Sumber: https://www.mathworks.com/discovery/image-segmentation.html

2. Autonomous Vehicle

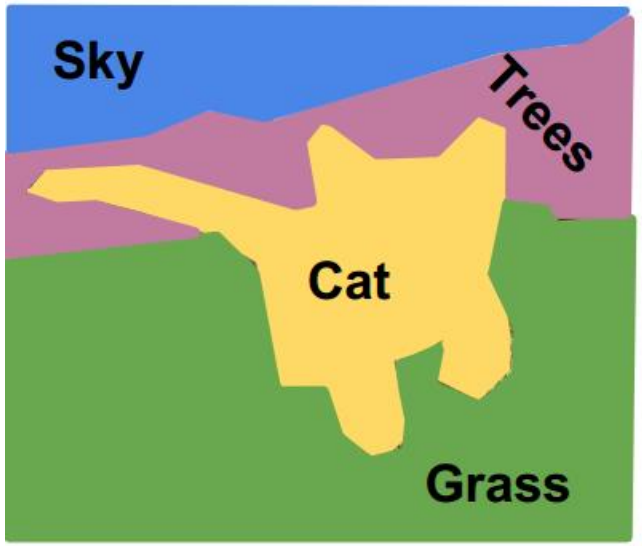
Saat merancang persepsi untuk kendaraan otonom, seperti mobil *self-driving*, segmentasi citra digunakan untuk membantu sistem mengidentifikasi dan menemukan kendaraan dan objek lain di jalan.

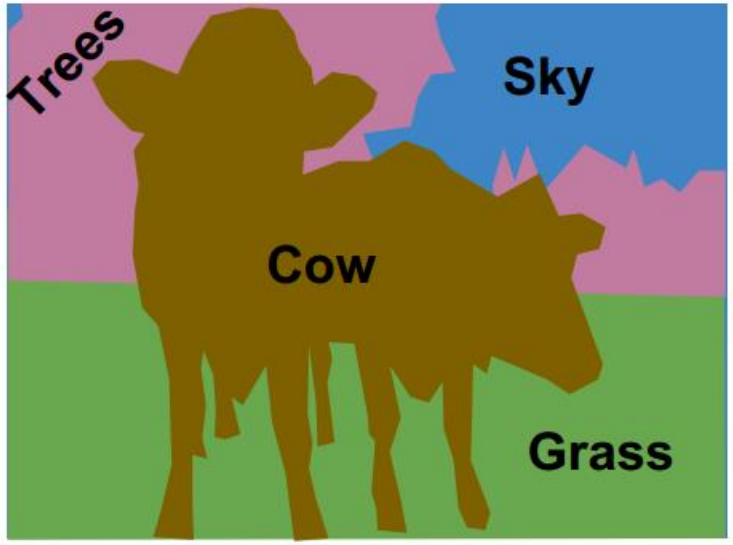


Menggunakan segmentasi citra untuk mengaitkan setiap piksel gambar dengan label kelas (seperti mobil, jalan, langit, pejalan kaki, atau sepeda).

Sumber: https://www.mathworks.com/discovery/image-segmentation.html

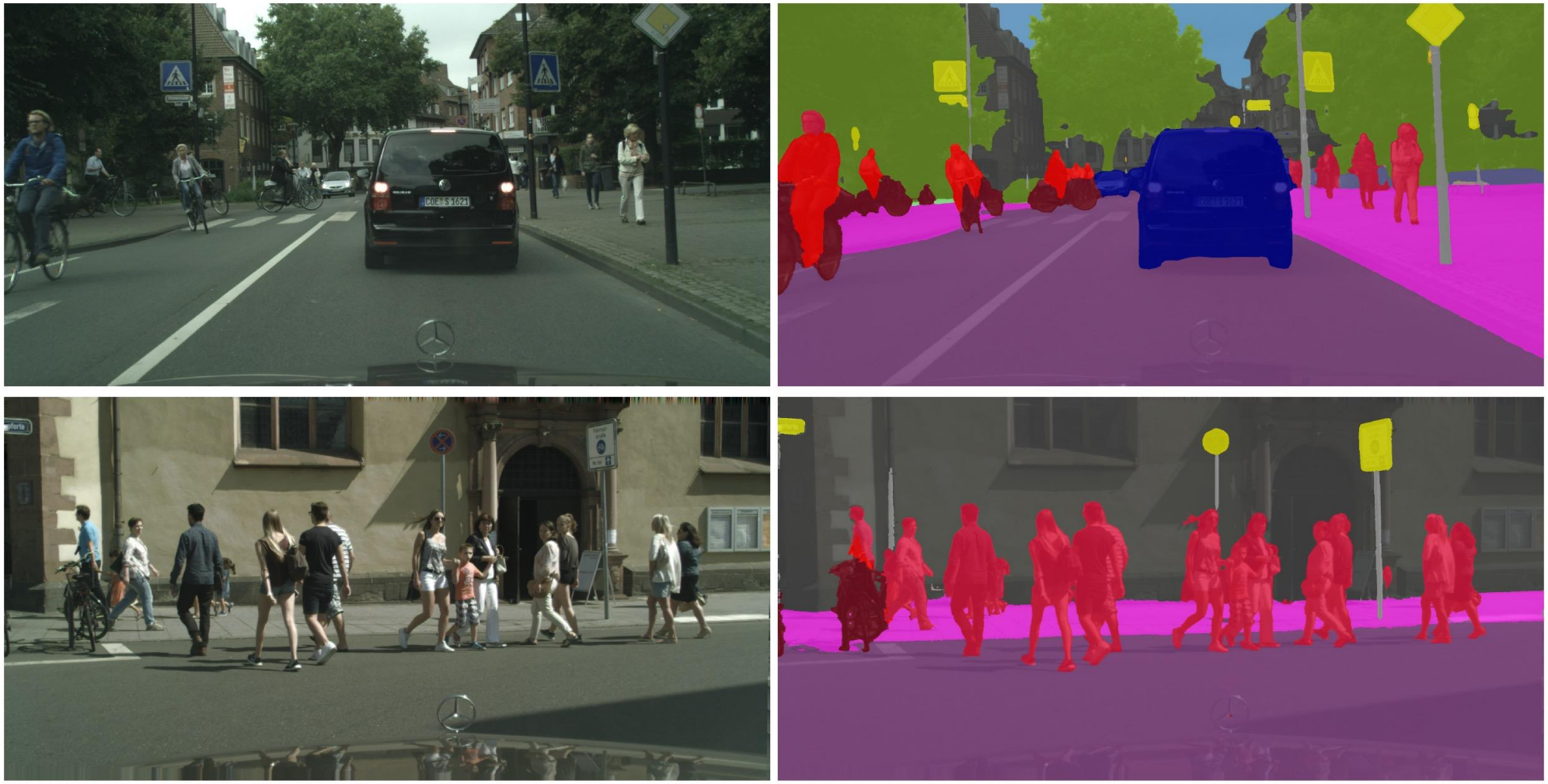
3. Object recognition





3. Scene understanding

Mesin dapat memahami apa yang “dilihatnya”



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 P

- 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)

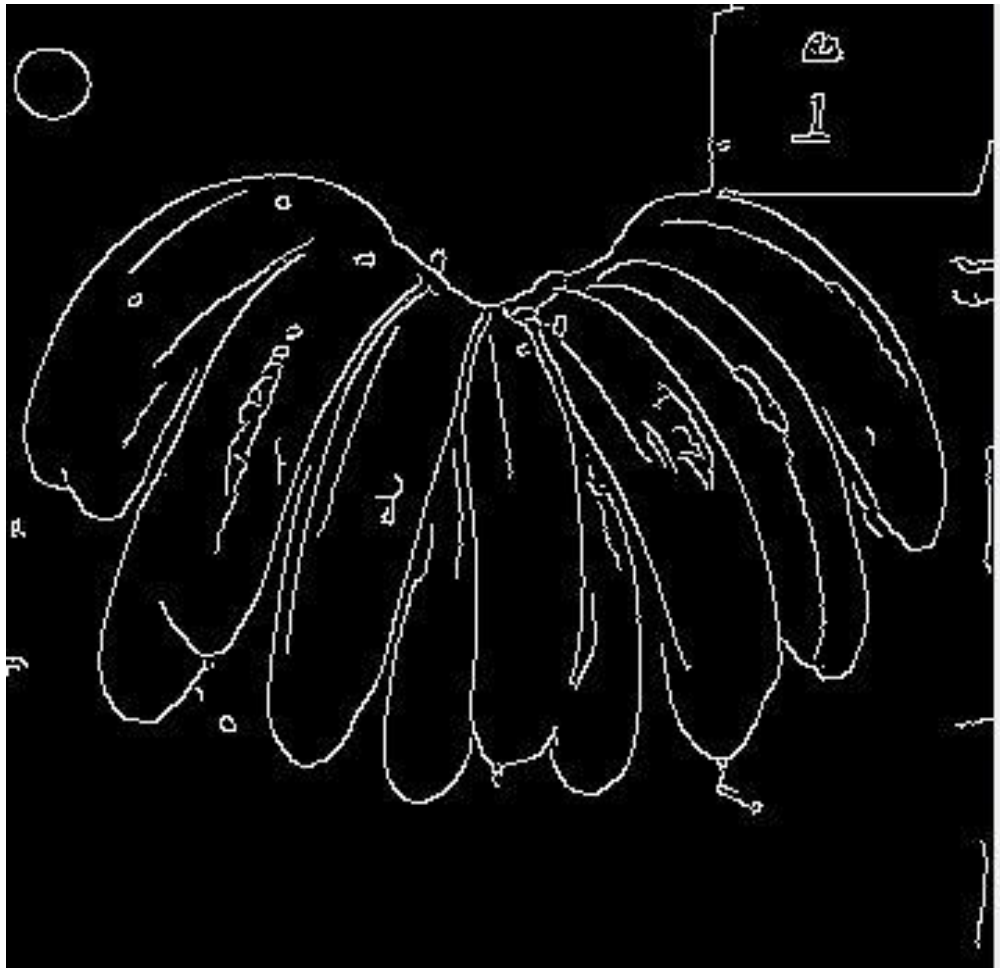
a b
c d

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

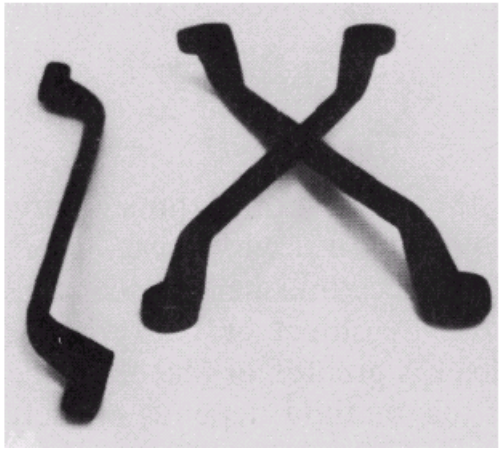
Figure 3. Complemented Image and Edged Image


1


1

Segmentasi citra berdasarkan *similarity*

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



Perkakas menjadi bagian yang koheren

Rumah, rumput, dan langit membentuk bagian koheren yang berbeda

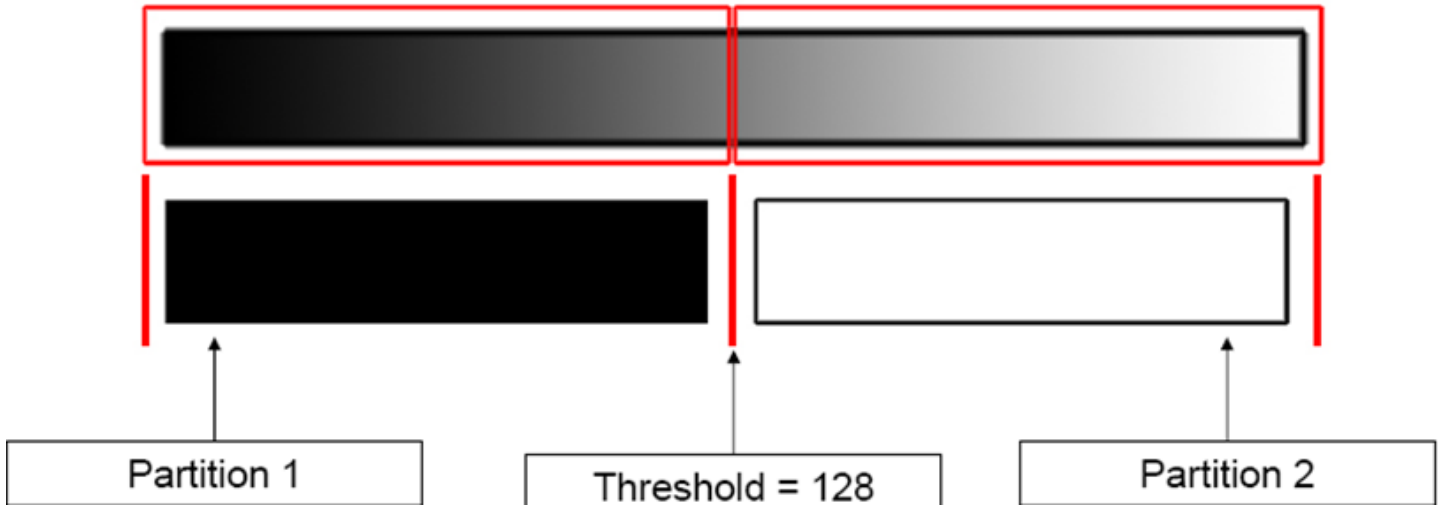
- Metode segmentasi berbasis *similarity*: pengambangan (*thresholding*), *region growing*, *split and merge*, dan *clustering*.

Sumber: CS 4487/9587 Algorithms for Image Analysis

1. Pengambangan

- Segmentasi citra dengan pengambangan (*thresholding*) dapat digunakan untuk memisahkan objek dengan latar belakangnya.
- Pengambangan dilakukan berdasarkan pada nilai intensitas pixel-pixel dan sebuah nilai ambang T.
- Setiap pixel pada posisi (i, j) pada citra di mana f(i, j)> T disebut titik objek, jika tidak maka akan disebut latar belakang.
- Hasil segmentasi adalah segmen *foreground* (objek) dan segmen *background*

$$f_B(i,j) = \begin{cases} A, & f_g(i,j) \leq T \\ B, & \text{lainnya} \end{cases}$$



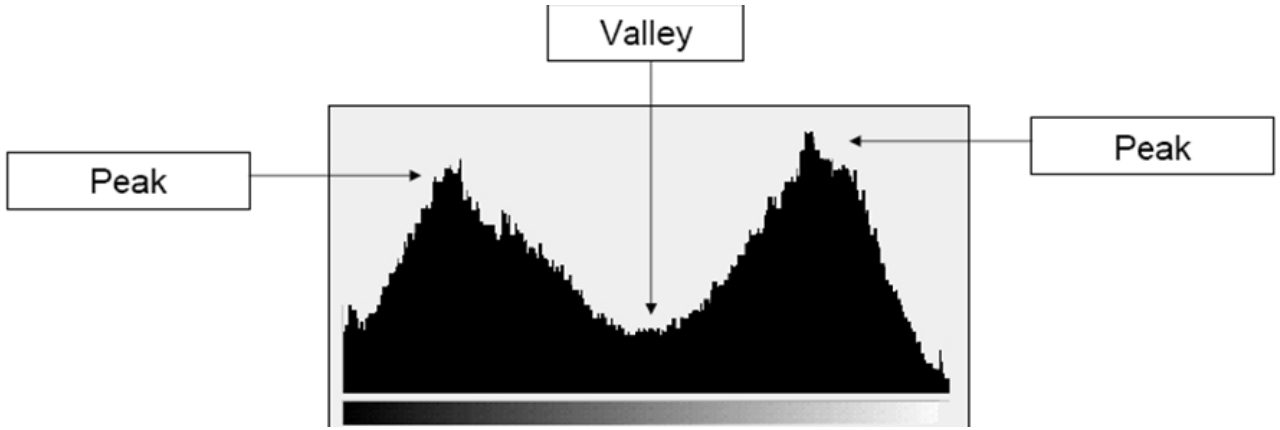
Pilih nilai ambang T

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

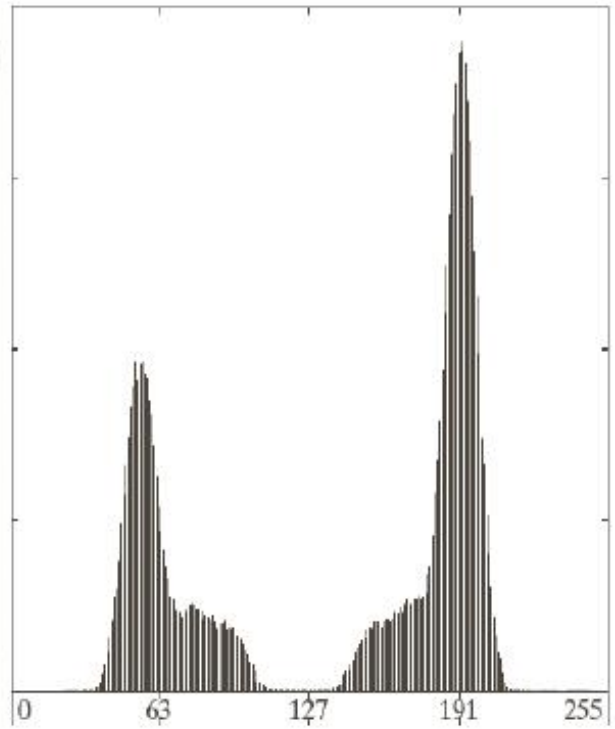
Jika A = 1 dan B = 0, maka hasil pengambangan meghasilkan citra biner:

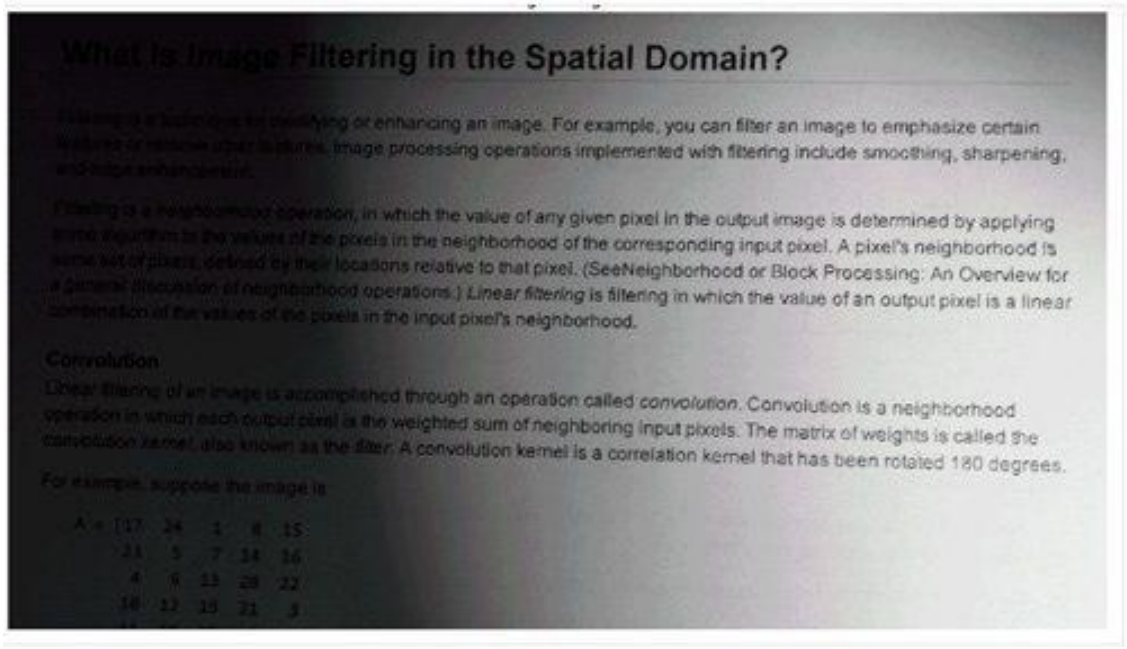
$$f_B(i,j) = \begin{cases} 1, & f_g(i,j) \leq T \\ 0, & \text{lainnya} \end{cases}$$

- 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



What Is Image Filtering in the Spatial Domain?

Filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement.

Filtering is a *neighborhood operation*, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel. (See Neighborhood or Block Processing: An Overview for a general discussion of neighborhood operations.) *Linear filtering* is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood.

Convolution

Linear filtering of an image is accomplished through an operation called *convolution*. Convolution is a neighborhood operation in which each output pixel is the weighted sum of neighboring input pixels. The matrix of weights is called the *convolution kernel*, also known as the *filter*. A convolution kernel is a correlation kernel that has been rotated 180 degrees.

For example, suppose the image is

```
A = [17  24   1   8  15
     23   5   7  14  16
      4   6  13  20  22
     10  12  19  21   3
```

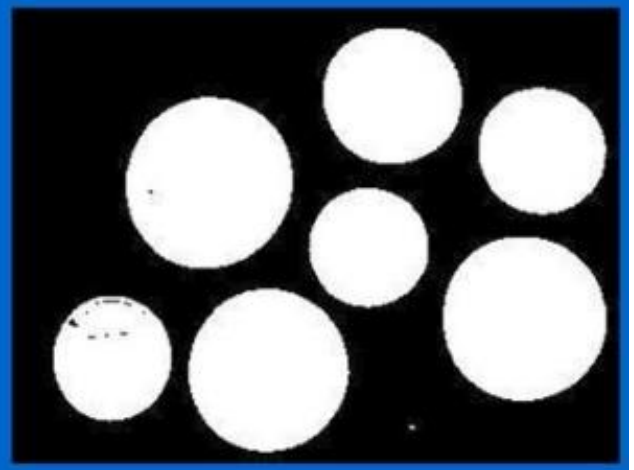
Menggunakan pengambangan untuk mengonversi ke gambar biner untuk meningkatkan keterbacaan teks dalam gambar.

Sumber: https://www.mathworks.com/discovery/image-segmentation.html

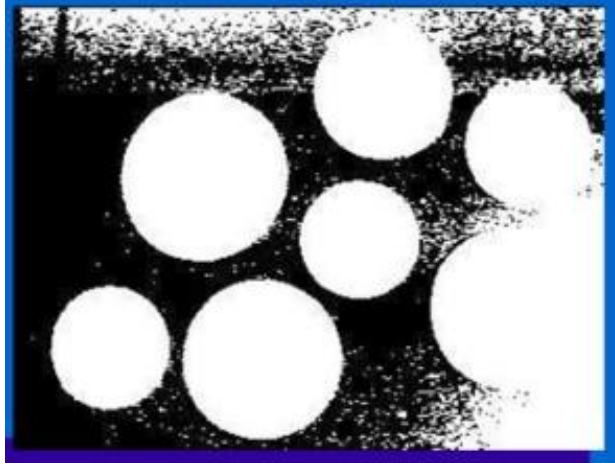
How to choose the threshold?

original

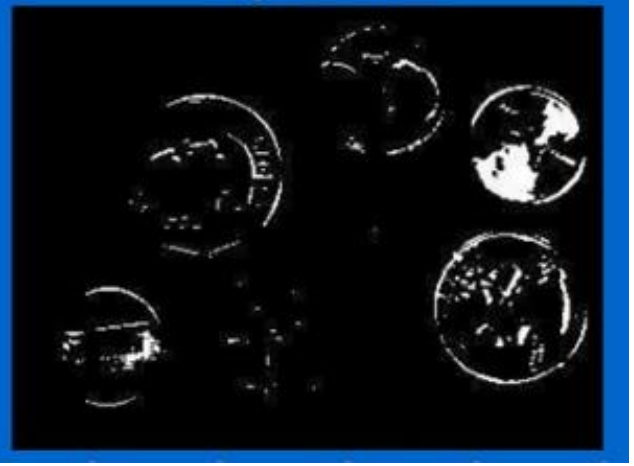
good threshold



low threshold



high threshold



- 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*

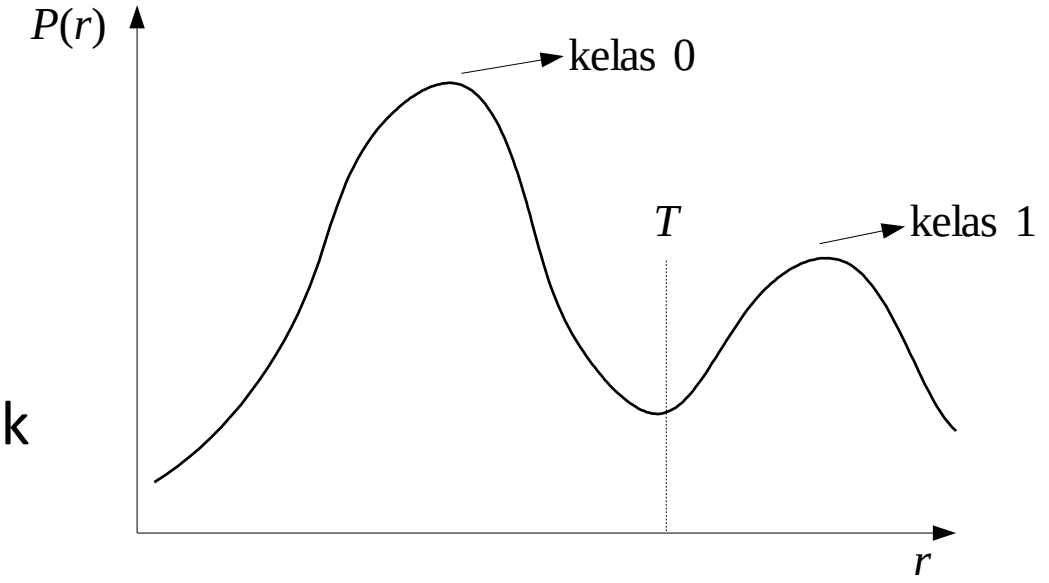
     Nilai ambang berubah secara dinamis bergantung pada perubahan pencahayaan di dalam citra

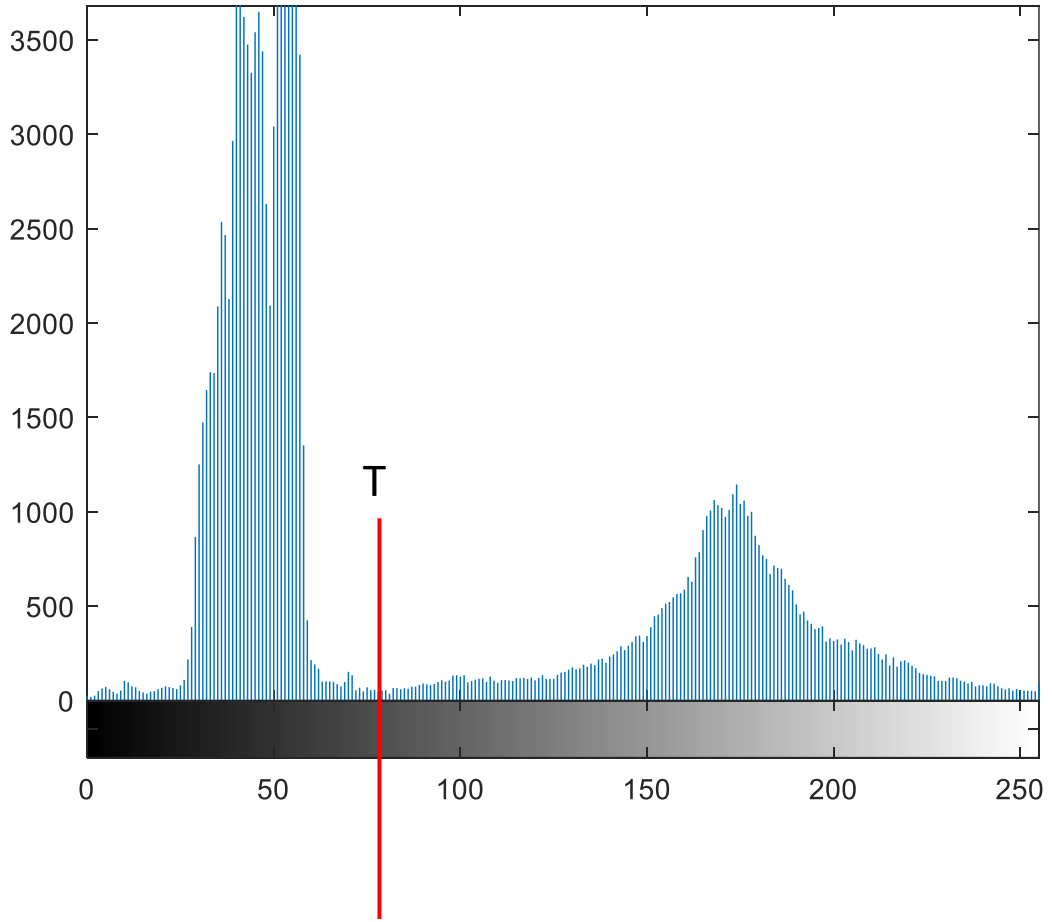
1. Pengambangan secara global

- Nilai *T* berlaku untuk seluruh bagian di dalam citra

- Nilai *T* dipilih sedemikian sehingga galat sekecil mungkin

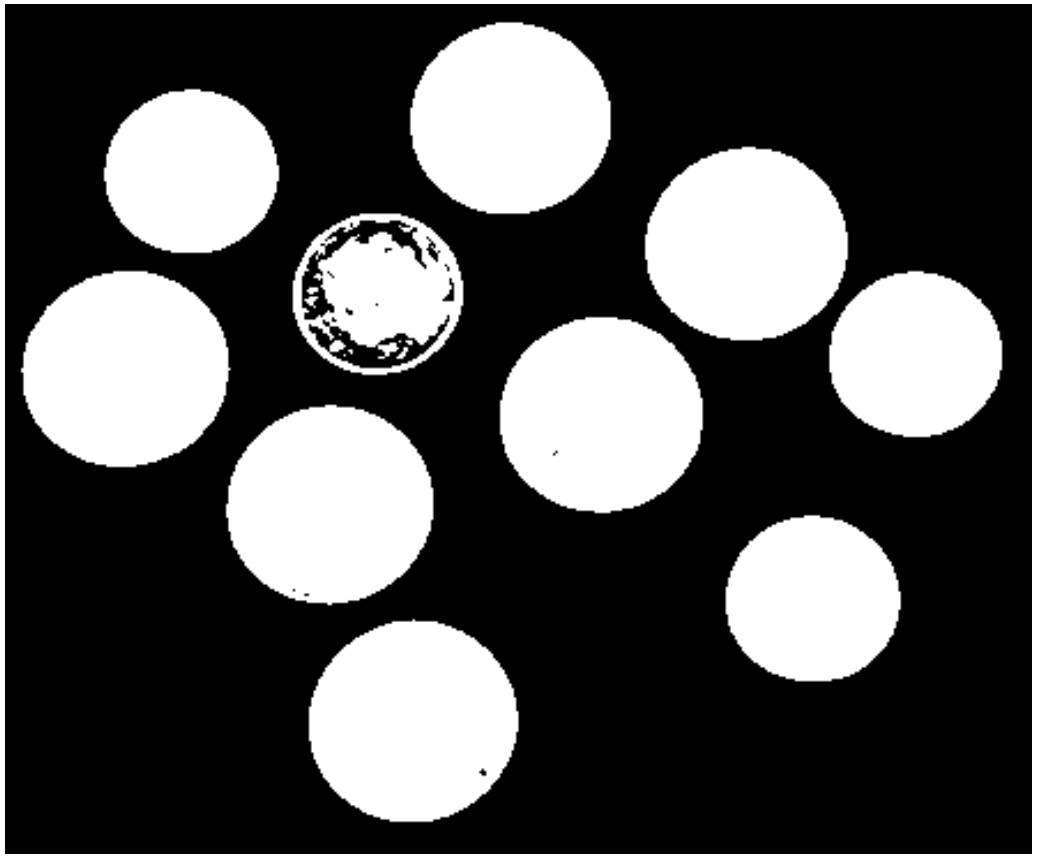
- Cara menentukan nilai T:
  - gambarkan histogram citra.
  - Jika citra mengandung satu atau lebih objek dan latar belakang dengan intensitas yang homogen, maka histogramnya *bimodal* (memiliki dua puncak)
  - nilai T dipilih di antara dua puncak tersebut




3500
3000
2500
2000
1500
1000
500
0
T
0
50
100
150
200
250

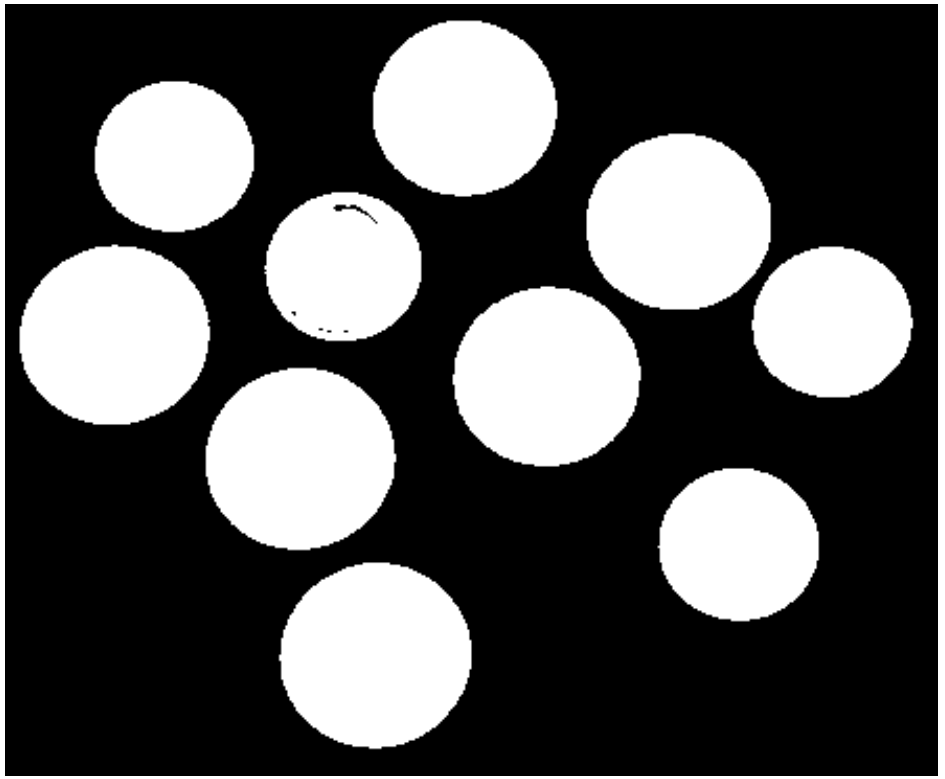
T = 100

```
I = imread('coins.bmp');
imhist(I)
BW = im2bw(I, 100/255);
figure, imshow(BW)
```



T = 75

```
I = imread('coins.bmp');
imhist(I)
BW = im2bw(I, 75/255);
figure, imshow(BW)
```



Menghitung T secara otomatis dengan algoritma *global thresholding:*

Global thresholding

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:

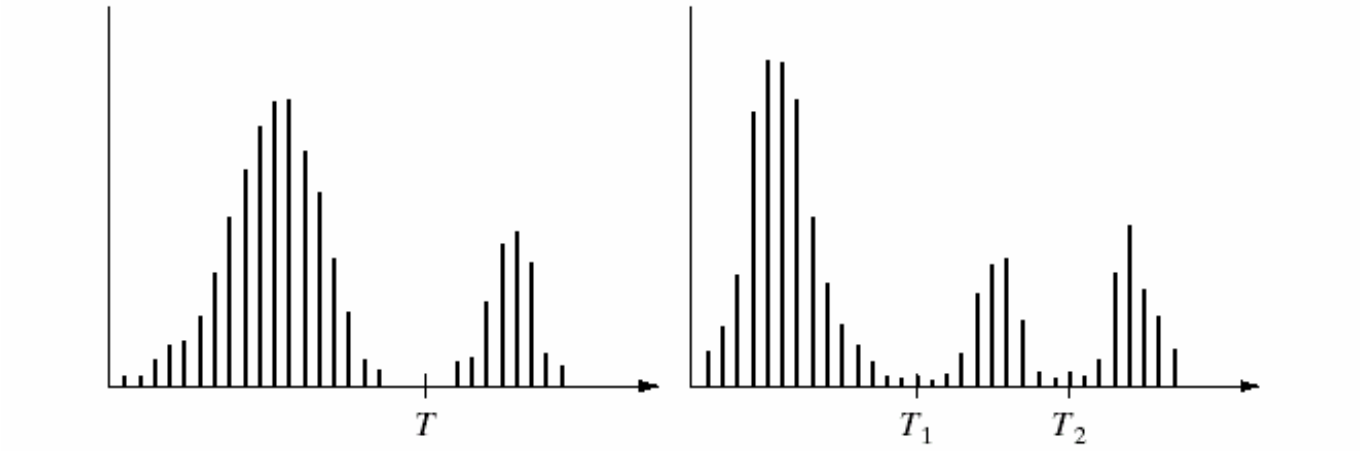
$$T_{\text{new}} = \frac{m_1 + m_2}{2}$$

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

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

- Mencari nilai T dengan algoritma 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

Menentukan nilai T dengan 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:

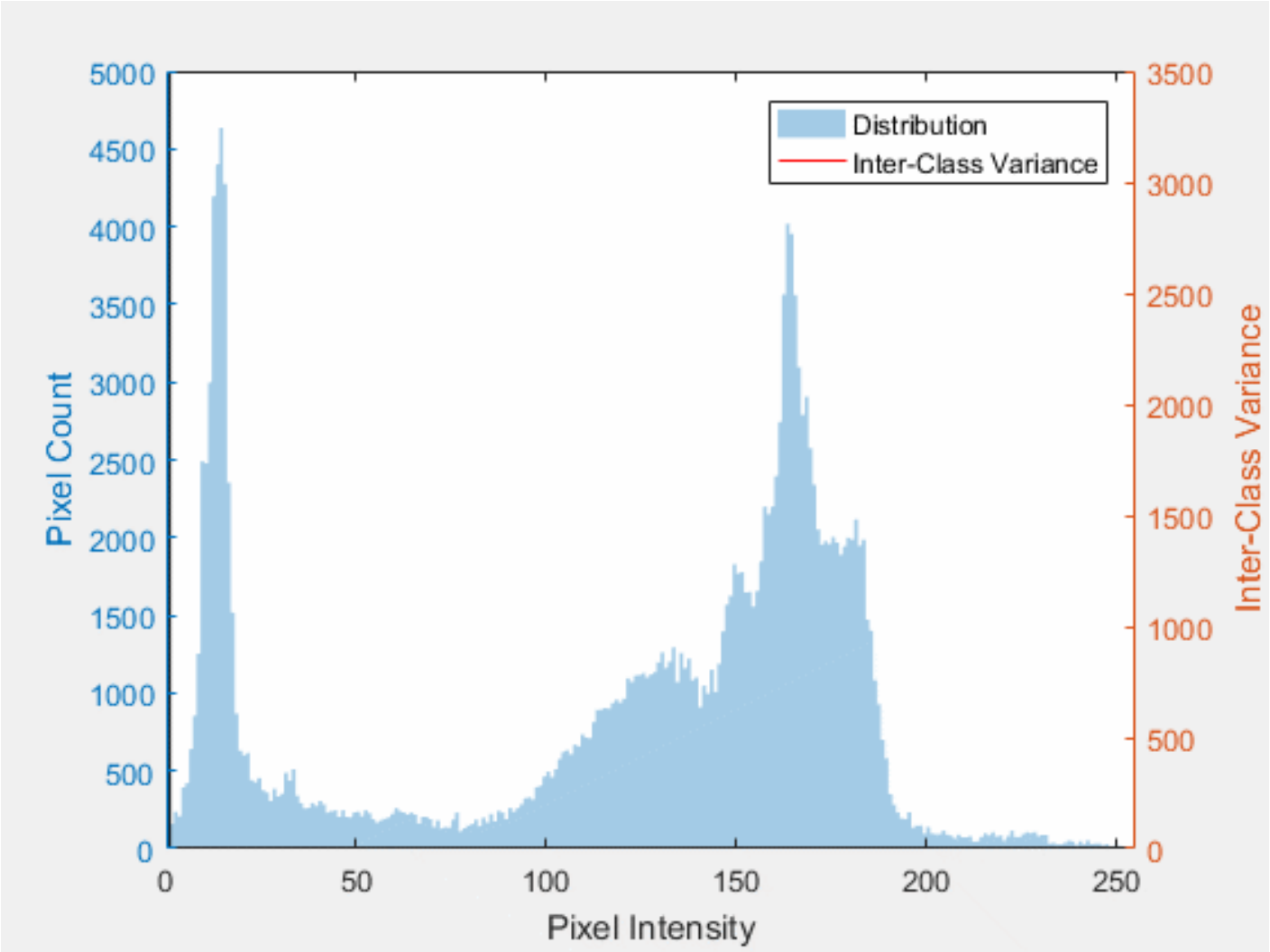
$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2}$$

where

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

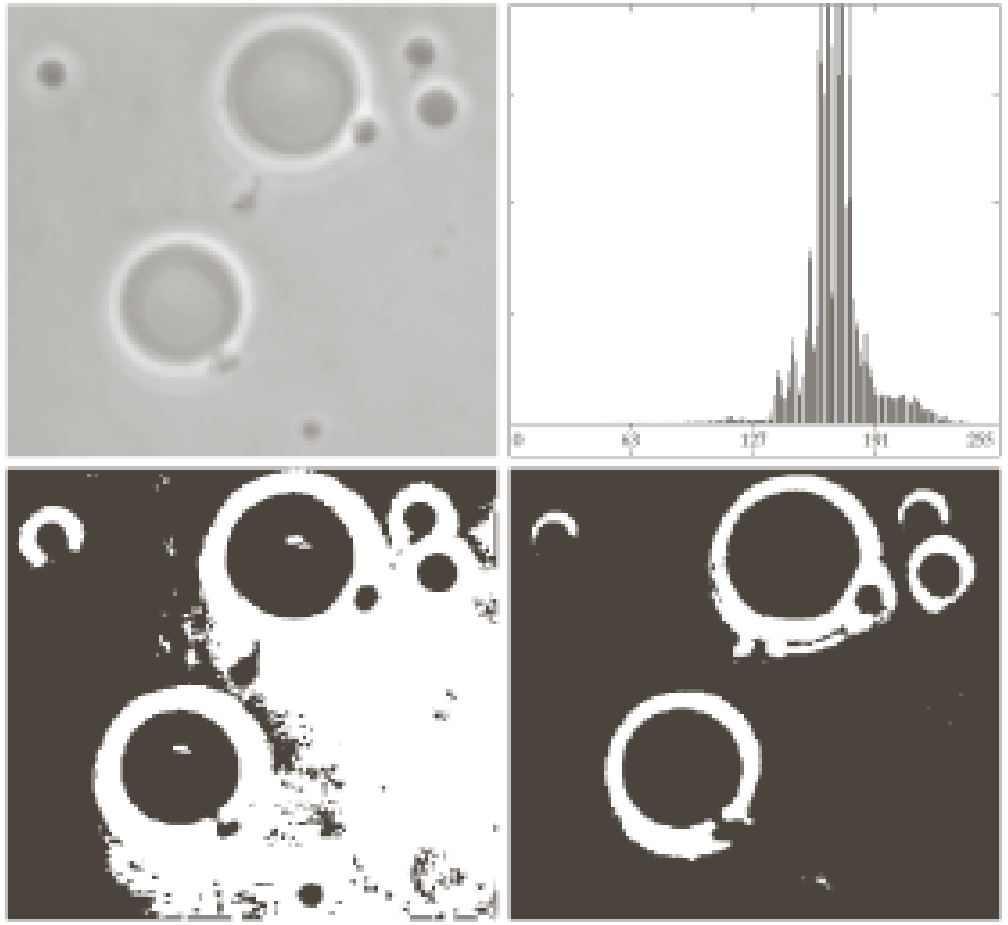
- The optimal threshold value, k^*, satisfies:

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



Visualisasi metode Otsu (Sumber: Wikipedia)

Otsu's method: an example



a	b
c	d

(a) original image;

(b) histogram of (a);

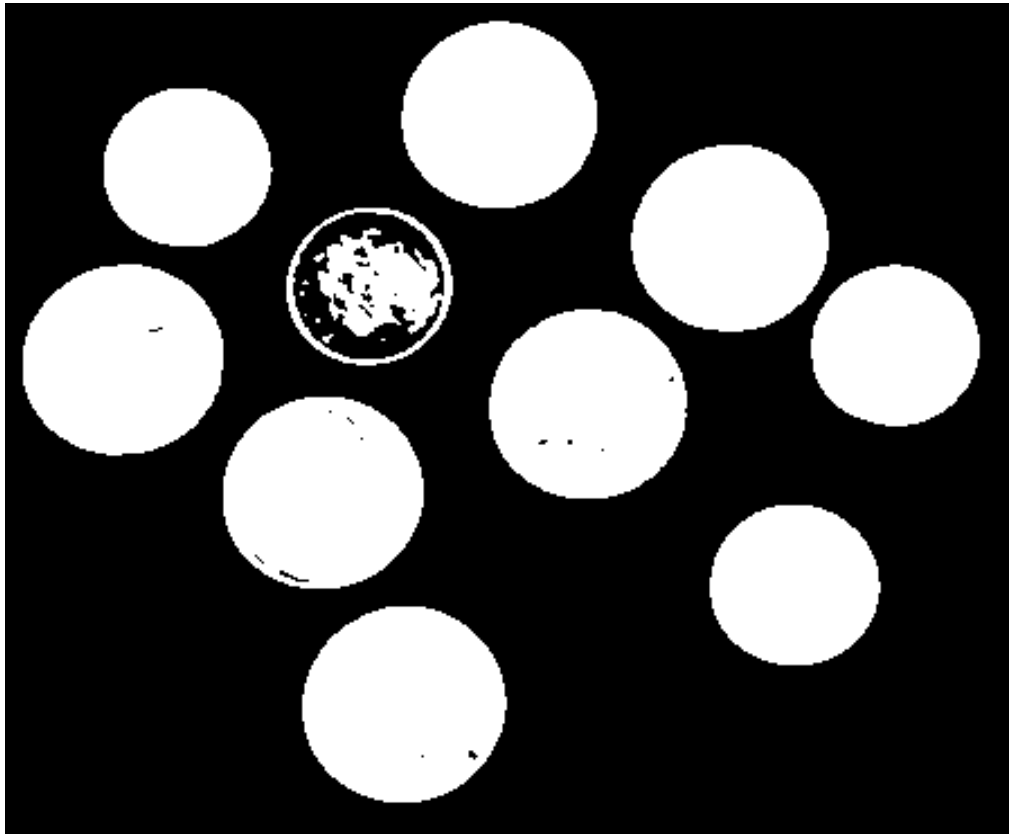
(c) global threshold: $T = 169$, $\eta = 0.467$;

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

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

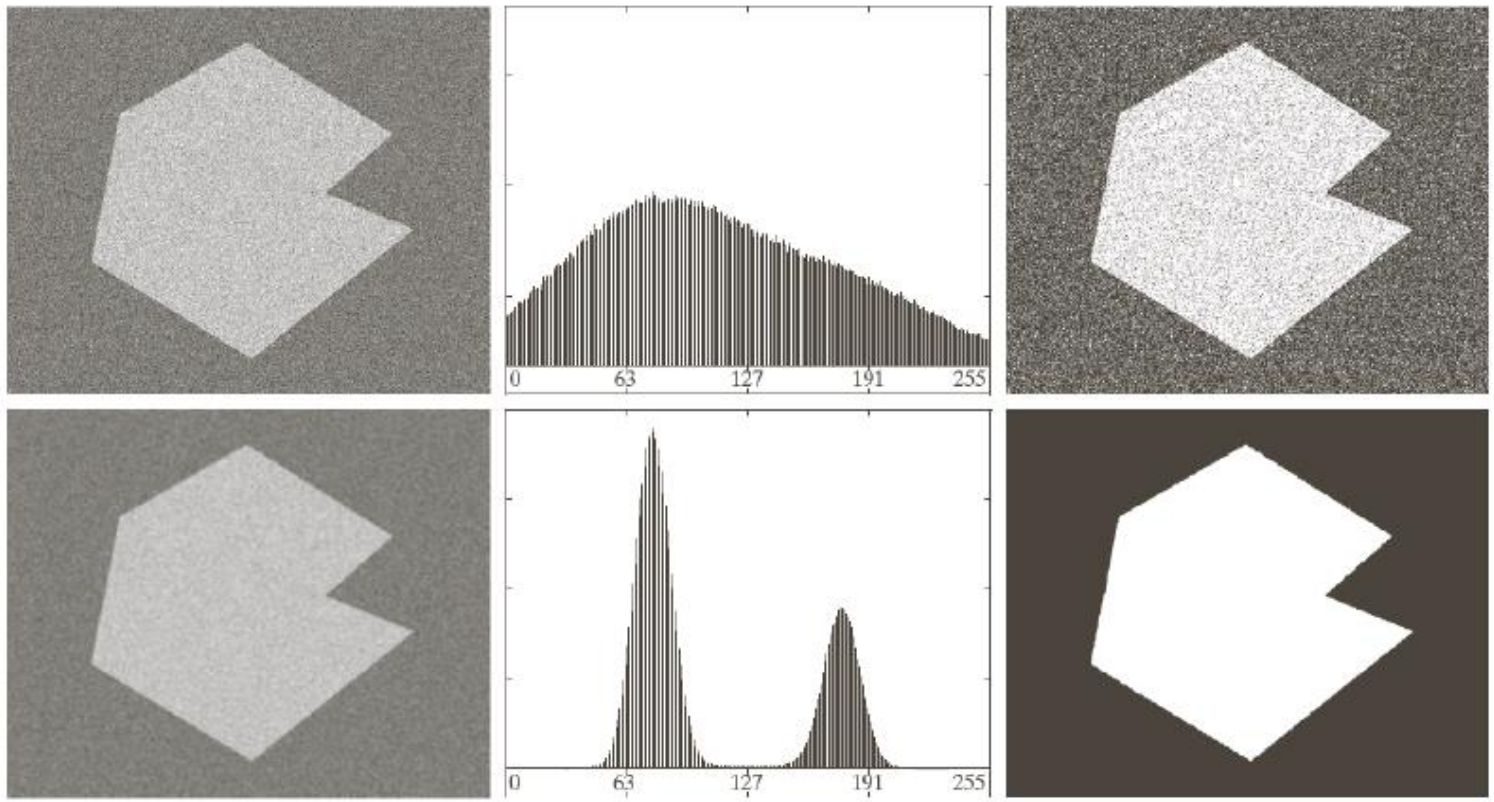
```
I = imread('house.jpg');
T = graythresh(I);
BW = im2bw(I, T);
imshow(I);
figure; imshow(BW)
```

```
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:

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

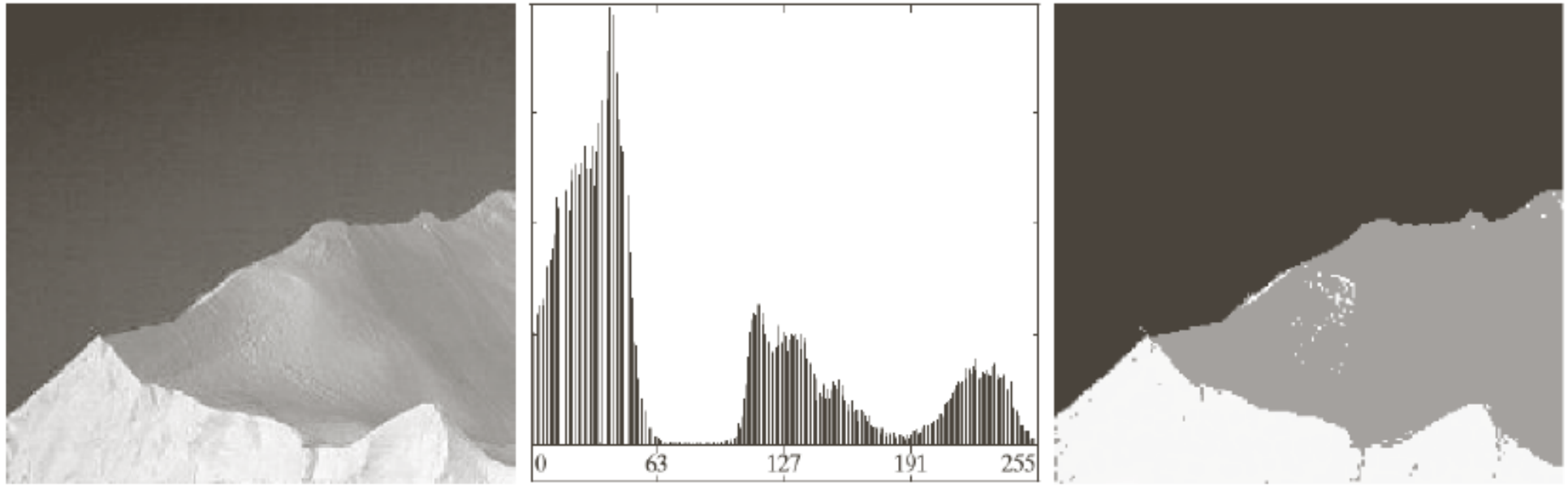
- The optimal thresholds k_1^* and k_2^* can be computed as:

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

- The separability degree can be measured as:

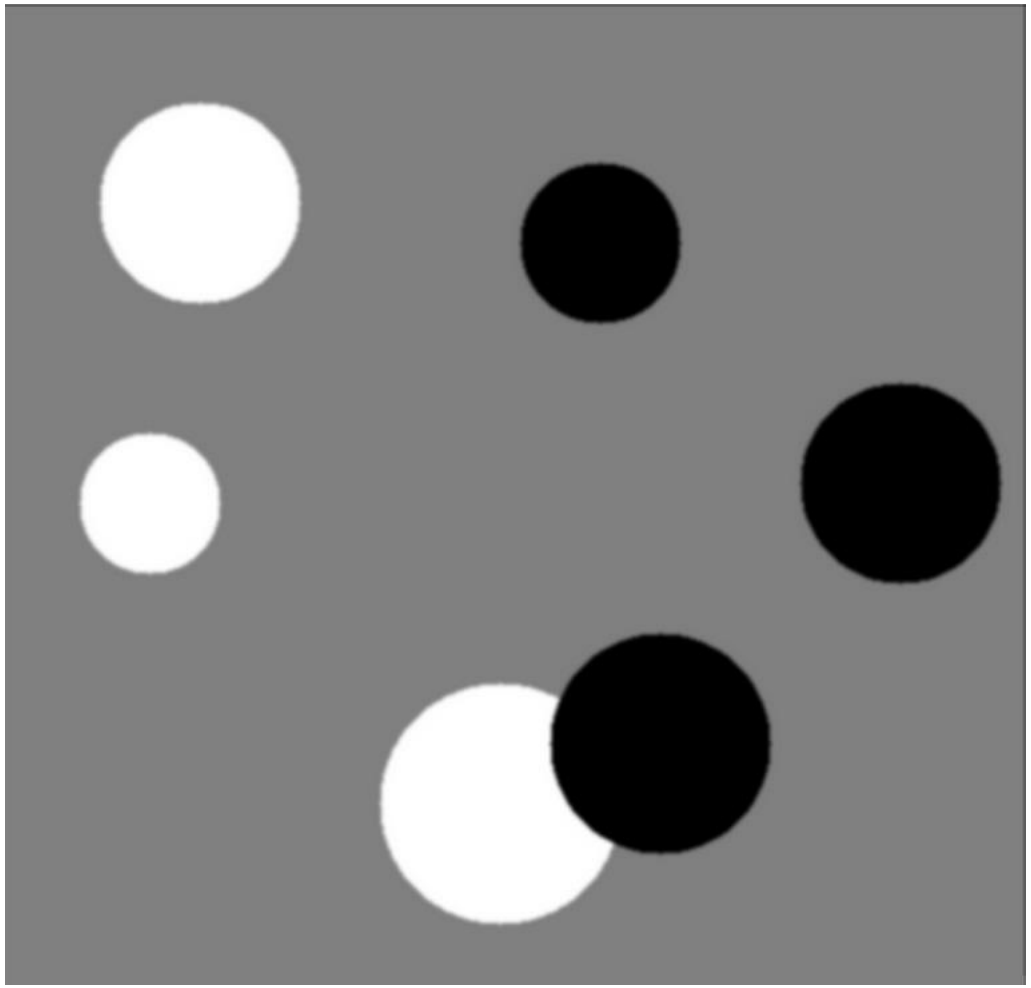
$$\eta(k_1^*,\, k_2^*) = \frac{\sigma_B^2(k_1^*,\, k_2^*)}{\sigma_G^2}$$

Multiple thresholds Otsu's method: an example

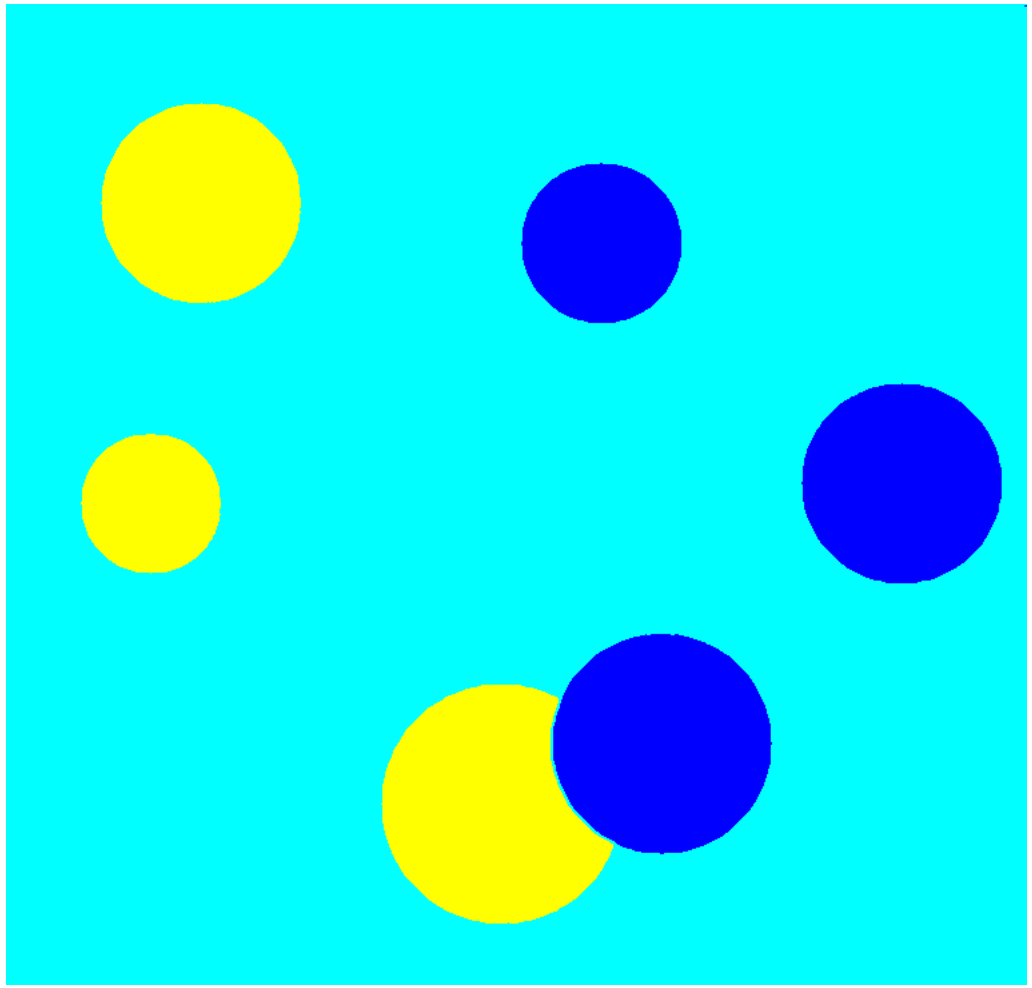


- Di dalam Matlab, fungsi `multithresh()` digunakan untuk melakukan *multiple threshold* dengan metode Otsu.

```
% 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. Pengambangan secara lokal

- Pengambangan secara lokal dilakukan terhadap bagian-bagian tertentu di dalam citra.
- Dalam hal ini citra dipecah menjadi bagian-bagian kecil, kemudian proses pengambangan dilakukan secara lokal.
- Nilai ambang untuk setiap bagian belum tentu sama dengan bagian lain.
- Sebagai contoh, pengambangan dilakukan terhadap bagian citra a yang berukuran 5×5 atau 8×8 *pixel*. Nilai ambangnya ditentukan sebagai fungsi rata-rata derajat keabuan di dalam dearah citra tersebut.

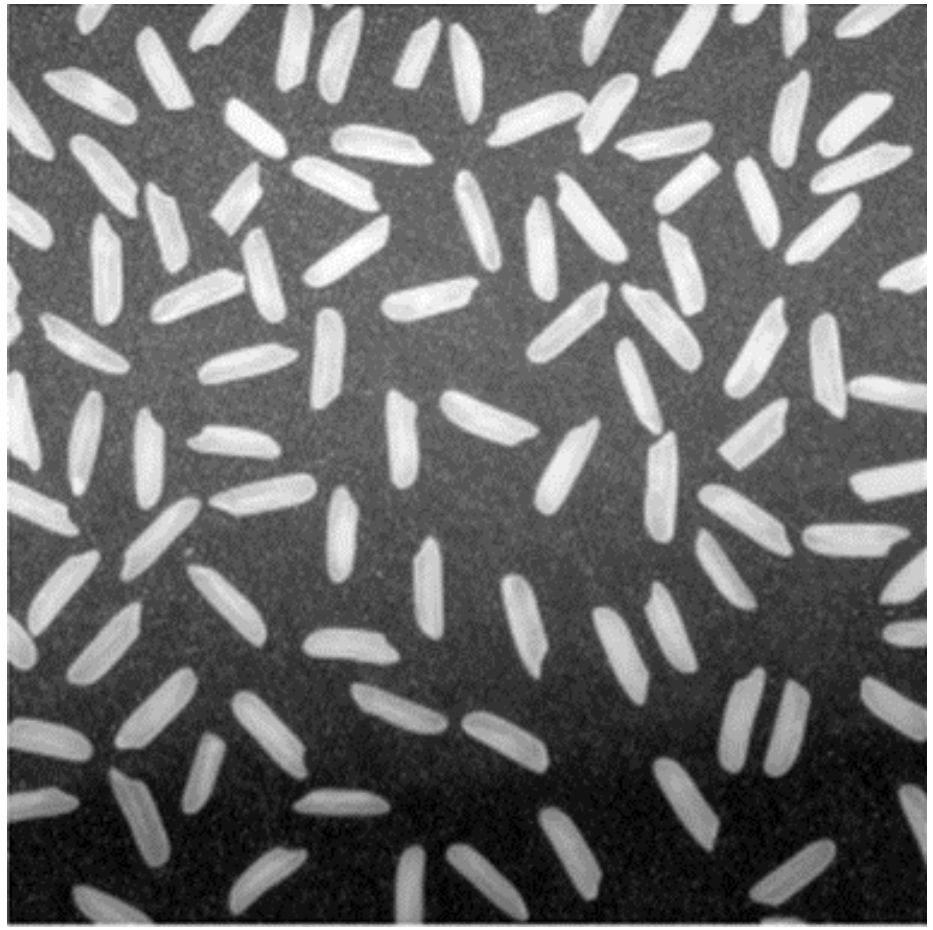
3. **Pengambangan secara adaptif**

- Pengambangan secara adaptif mengubah nilai ambang secara dinamis pada citra.

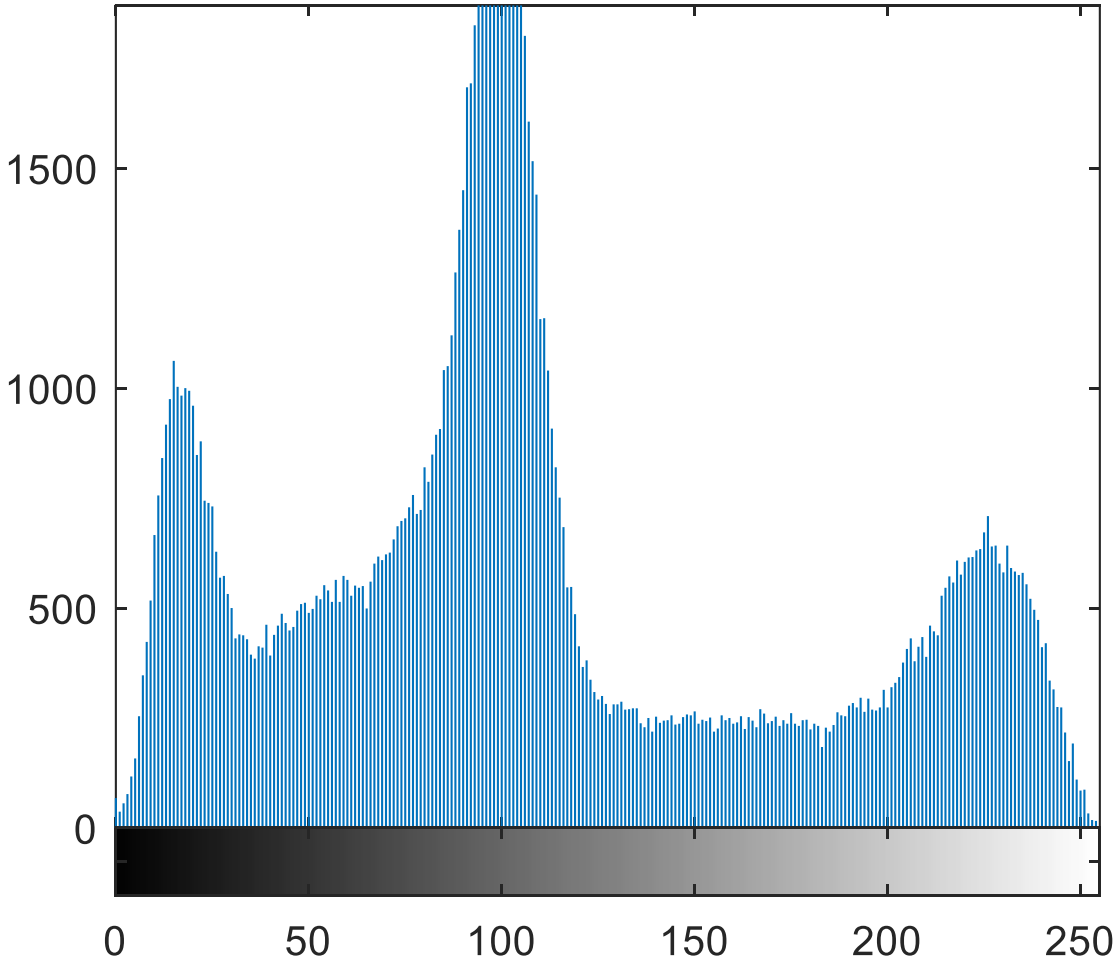
- Versi pengambangan yang lebih canggih ini dapat mengakomodasi perubahan kondisi cahaya pada gambar, misalkan yang terjadi akibat gradien iluminasi yang kuat atau ada bayangan.

Pengambangan adaptif di dalam Matlab

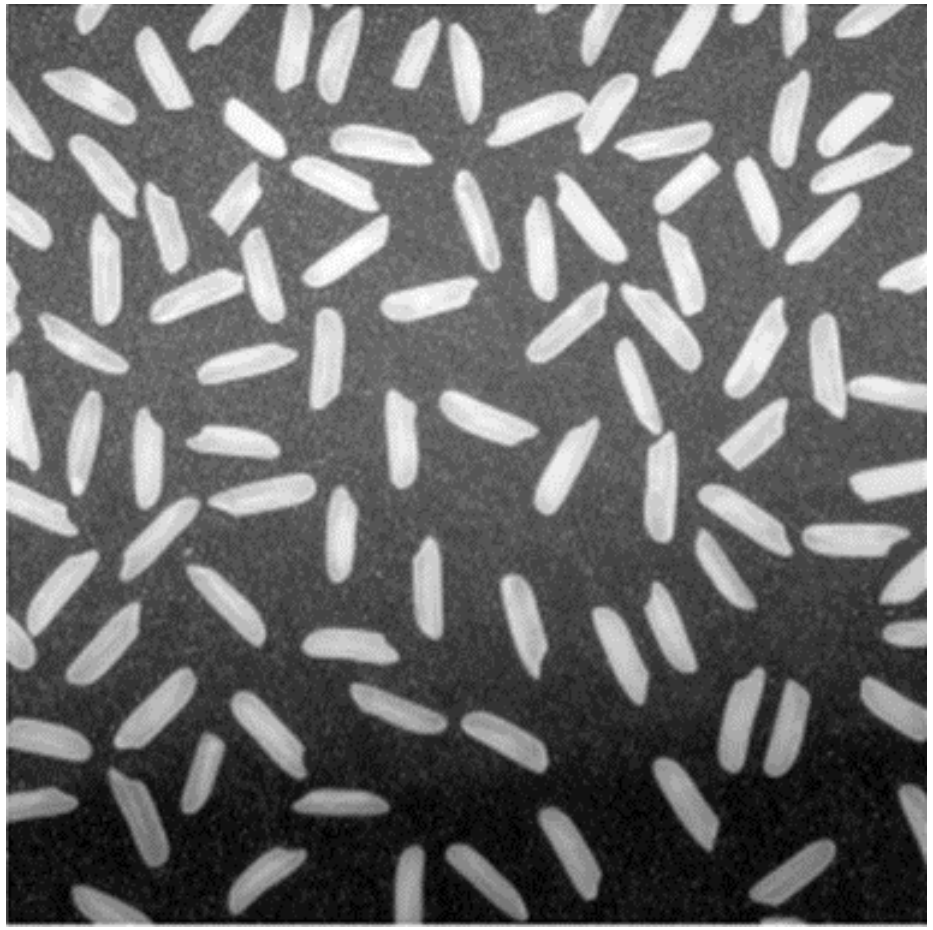
```
I = imread('rice.bmp');
```

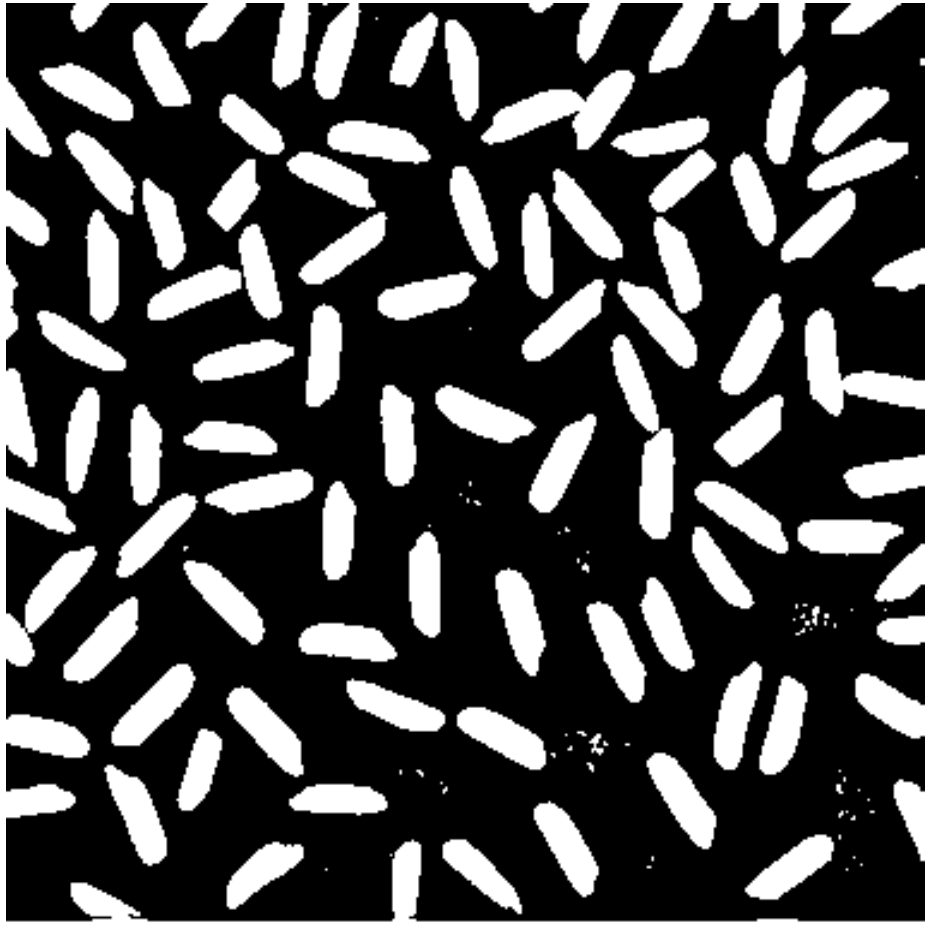


```
imhist(I)
```



```
BW = imbinarize(I,'adaptive');
figure, imshow(BW)
```





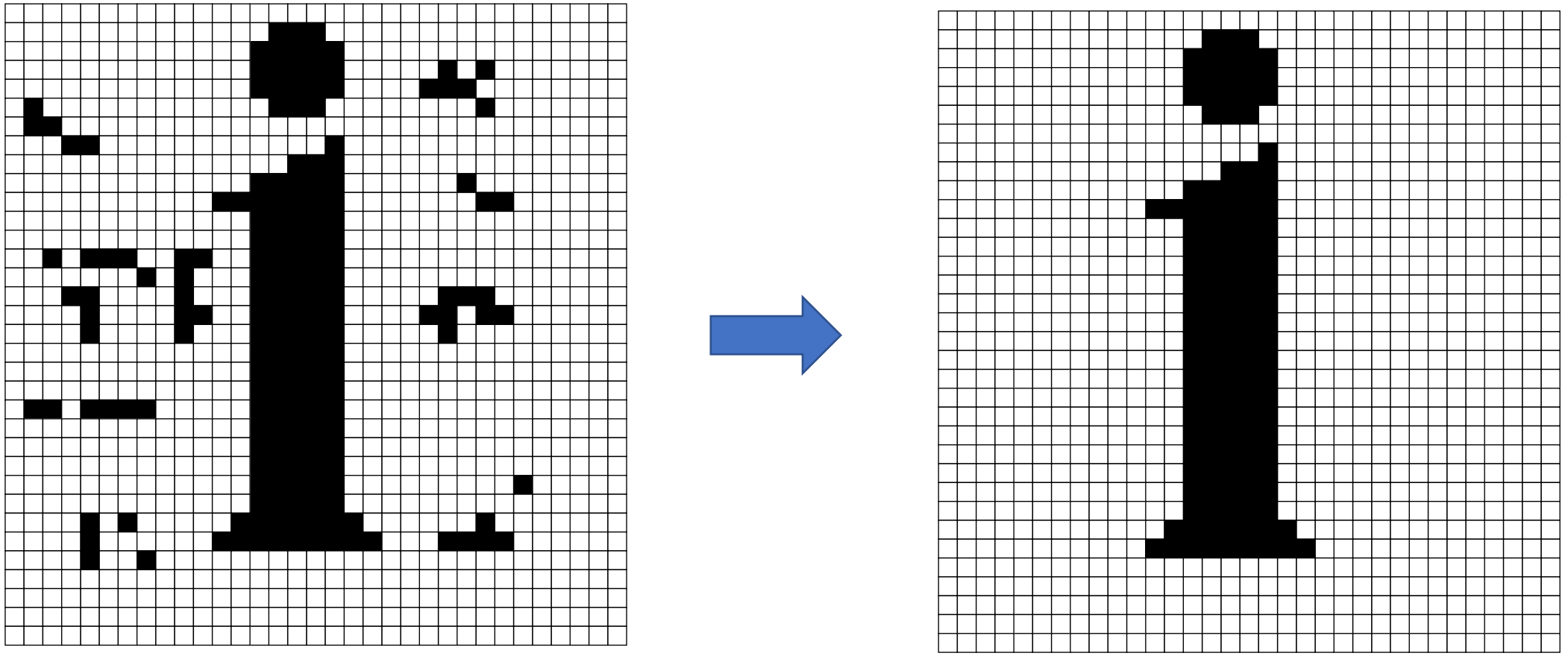
Kegunaan Citra Biner Hasil Pengambangan

- Citra biner hasil pengambangan berguna untuk memisahkan objek dengan latar belakang pada citra asalnya.
- Citra biner menjadi *template* untuk melakukan segmentasi

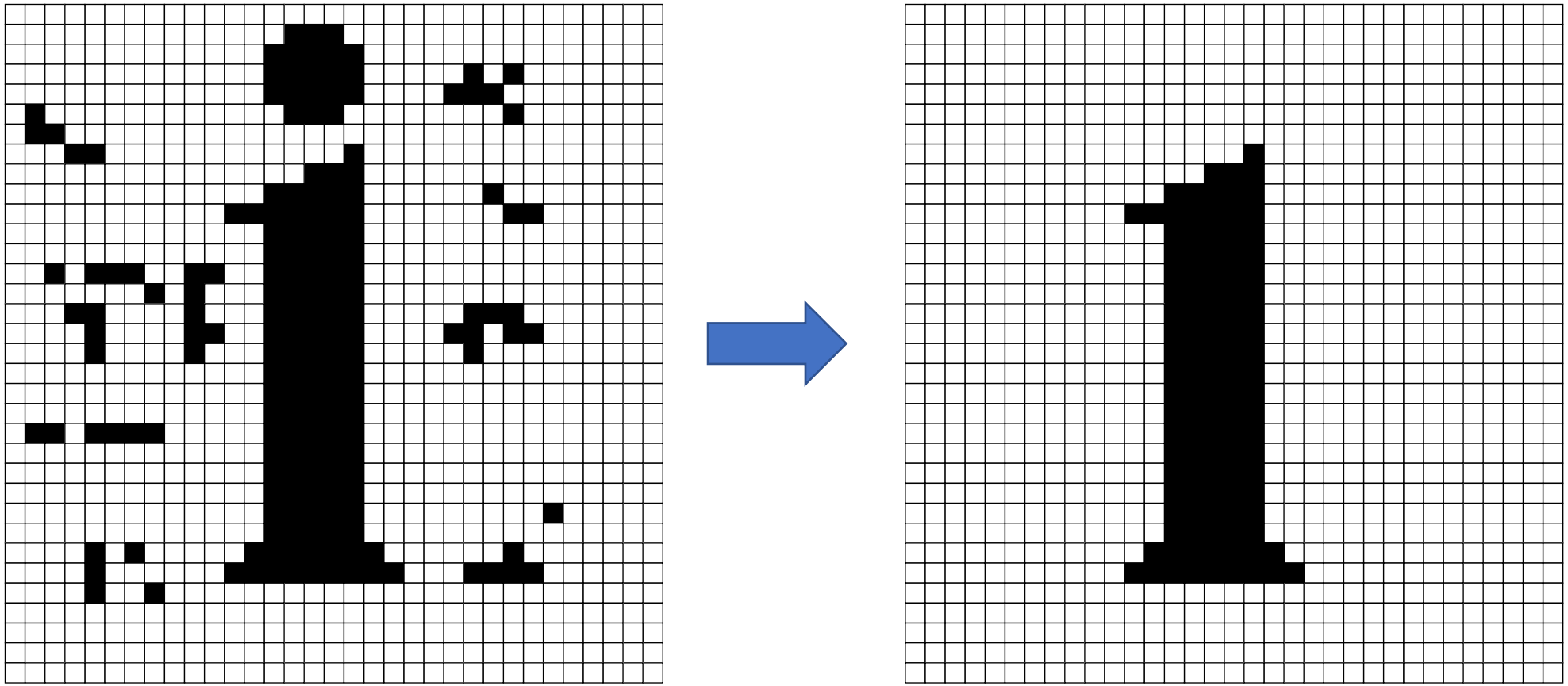
```
I = imread('coins.bmp');
BW = im2bw(I, 75/255);
figure, imshow(BW)
s = size(BW);
m = s(1);
n = s(2);
for i=1:m
    for j=1:n
       if BW(i,j) == 0
          C(i,j) = 255;
       else
          C(i,j) = I(i,j);
       end
    end
end
C = uint8(C);
figure, imshow(C);
```

Penapis Luas

- Seringkali citra biner hasil pengambangan mengandung beberapa daerah yang dianggap sebagai gangguan. Biasanya daerah gangguan itu berukuran kecil.
- Penapis luas dapat digunakan untuk menghilangan daerah gangguan tersebut.
- Misalkan objek yang dianalisis diketahui mempunyai luas yang lebih besar dari *T*.
- Maka, *pixel-pixel* dari daerah yang luasnya di bawah *T* dinyatakan dengan 0. Dengan cara ini, daerah yang berupa gangguan dapat dihilangkan



Kiri: gangguan pada citra biner yang mengandung huruf “i”;
Kanan: citra yang dihasilkan setelah dilakukan penapisan (T = 10)



Kesalahan yang diperoleh dari pengambilan nilai T_0 yang tidak tepat (T = 25). Perhatikan bahwa “titik” di atas huruf “i” hilang karena luasnya, sehingga huruf “i” terlihat seperti angka “1”

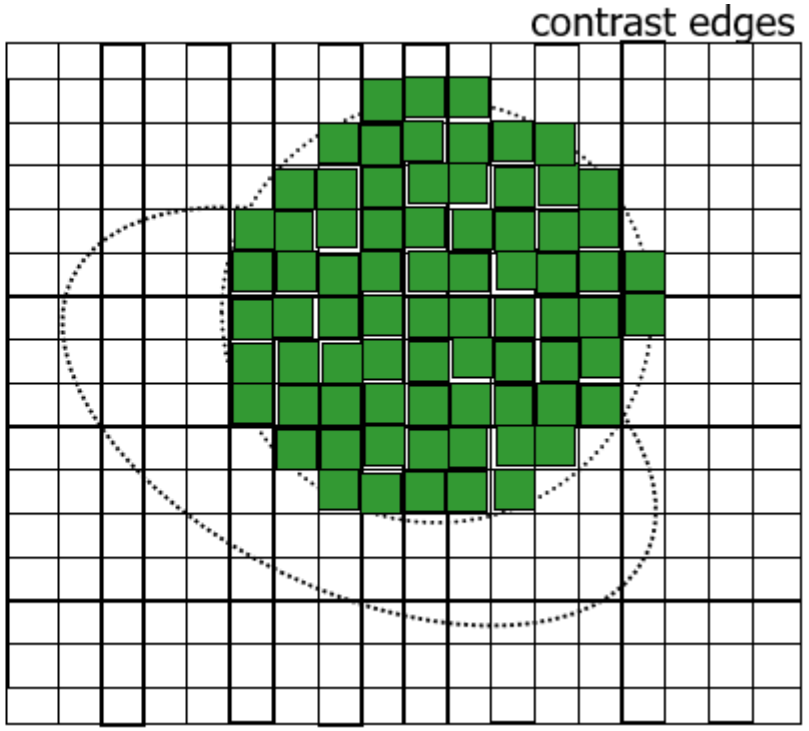
2. Region Growing

- *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.

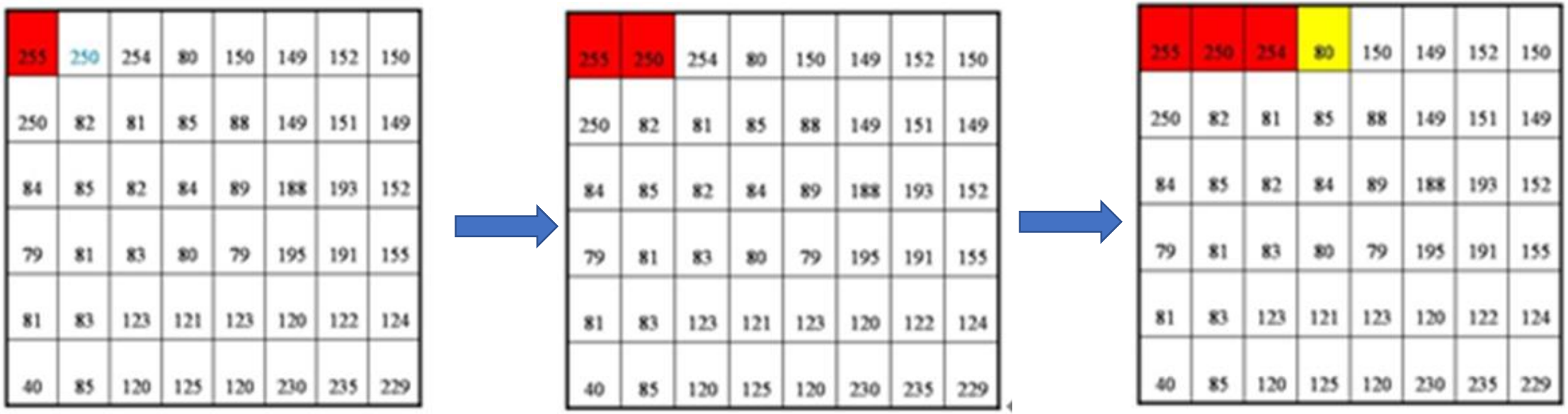
- Keuntungan: memiliki keterhubungan yang bagus antar pixel di dalam region
- Kelemahan: - pemilihan umpan yang tepat
  - kriteria berhenti
  - mengkonsumsi waktu yang lama



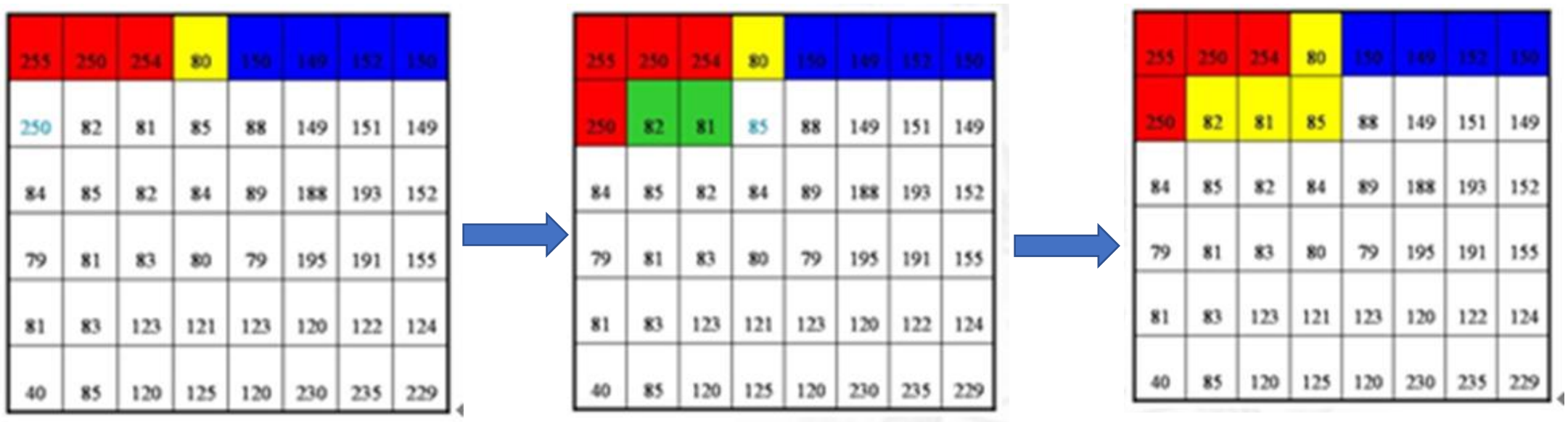
Sumber: CS 4487/9587 Algorithms for Image Analysis: Basic Image Segmentation

Unseeded Region Growing

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

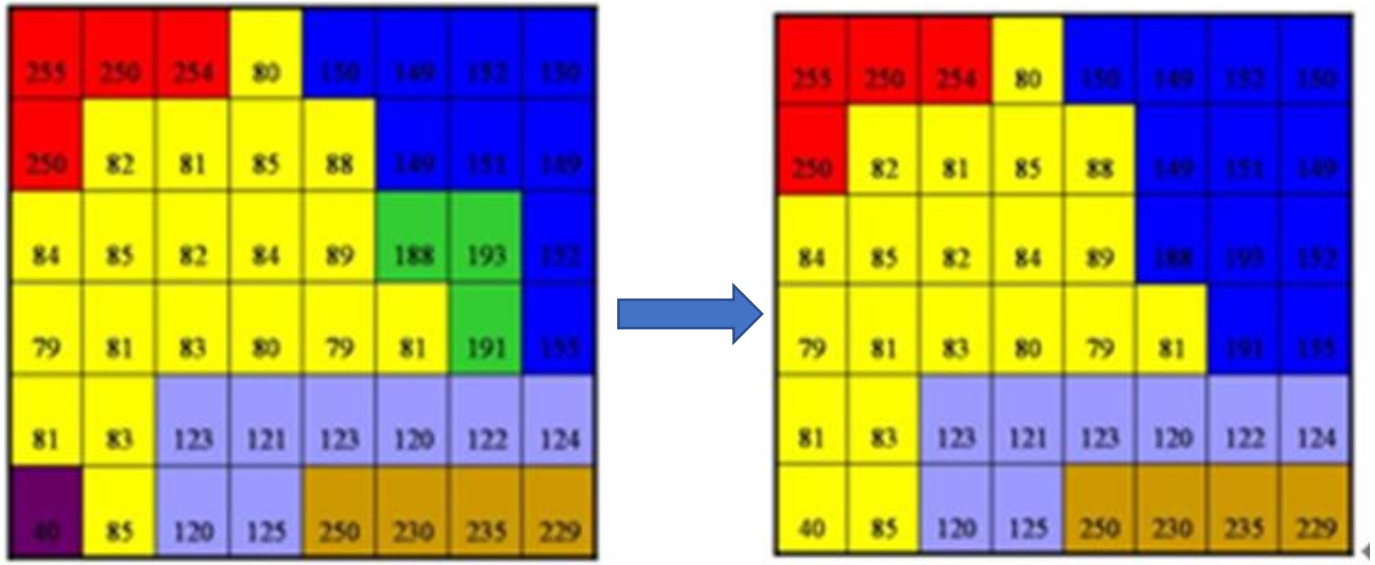


Sumber:主講人:張緯德, Image segmentation, Representation, and Description



Langkah terakhir:

Gabungkan (*merge*) region kecil menjadi region besar



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

Seeded Region Growing (segmentasi.m)

```
% read image
reg_maxdist = 0.2;
I = im2double(imread('lada-gray.bmp'));
subplot(121);
imshow(I);
% let the user pick one point
[x,y] = ginput(1);
% round to integer to match required input by regiongrowing function
x = round(x);
y = round(y);
% plot point on original image
hold on;
plot(x,y,'xg','MarkerSize',20,'LineWidth',2);
hold off;
% get region from seed point
J = regiongrowing(I,y,x,reg_maxdist);
% plot region
subplot(122);
imshow(J);
```

Sumber: https://stackoverflow.com/questions/44234856/region-growing-in-matlab

(regiongrowing.m)

```
function J=regiongrowing(I,x,y,reg_maxdist)
% This function performs "region growing" in an image from a specified
% seedpoint (x,y)
%
% J = regiongrowing(I,x,y,t)
% % I : input image
% J : logical output image of region
% x,y : the position of the seedpoint (if not given uses function getpts)
% t : maximum intensity distance (defaults to 0.2)
%
% The region is iteratively grown by comparing all unallocated neighbouring pixels to
the region.
% The difference between a pixel's intensity value and the region's mean,
% is used as a measure of similarity. The pixel with the smallest difference
% measured this way is allocated to the respective region.
% This process stops when the intensity difference between region mean and
% new pixel become larger than a certain treshold (t)
%
% Example:
%
% I = im2double(imread('medtest.png'));
% x=198; y=359;
% J = regiongrowing(I,x,y,0.2);
% figure, imshow(I+J);
%
% Author: D. Kroon, University of Twente
```

```
if(exist('reg_maxdist','var')==0), reg_maxdist=0.2; end
if(exist('y','var')==0), figure, imshow(I,[]); [y,x]=getpts;
y=round(y(1)); x=round(x(1)); end
J = zeros(size(I)); % Output
Isizes = size(I); % Dimensions of input image
reg_mean = I(x,y); % The mean of the segmented region
reg_size = 1; % Number of pixels in region
% Free memory to store neighbours of the (segmented) region
neg_free = 10000; neg_pos=0;
neg_list = zeros(neg_free,3);
pixdist=0; % Distance of the region newest pixel to the regio mean
% Neighbor locations (footprint)
neigb=[-1 0; 1 0; 0 -1;0 1];
% Start regiogrowing until distance between regio and posible new
pixels become
% higher than a certain treshold
```

```
while(pixdist<reg_maxdist&&reg_size<numel(I))
    % Add new neighbors pixels
    for j=1:4,
        % Calculate the neighbour coordinate
        xn = x +neigb(j,1); yn = y +neigb(j,2);

        % Check if neighbour is inside or outside the image
        ins=(xn>=1)&&(yn>=1)&&(xn<=Isizes(1))&&(yn<=Isizes(2));

        % Add neighbor if inside and not already part of the segmented area
        if(ins&&(J(xn,yn)==0))
                neg_pos = neg_pos+1;
                neg_list(neg_pos,:) = [xn yn I(xn,yn)]; J(xn,yn)=1;
        end
    end
    % Add a new block of free memory
    if(neg_pos+10>neg_free), neg_free=neg_free+10000;
neg_list((neg_pos+1):neg_free,:)=0; end
```

```
% Add pixel with intensity nearest to the mean of the region, to the region
    dist = abs(neg_list(1:neg_pos,3)-reg_mean);
    [pixdist, index] = min(dist);
    J(x,y)=2; reg_size=reg_size+1;

    % Calculate the new mean of the region
    reg_mean= (reg_mean*reg_size + neg_list(index,3))/(reg_size+1);

    % Save the x and y coordinates of the pixel (for the neighbour add
proccess)
    x = neg_list(index,1); y = neg_list(index,2);

    % Remove the pixel from the neighbour (check) list
    neg_list(index,:)=neg_list(neg_pos,:); neg_pos=neg_pos-1;
end
% Return the segmented area as logical matrix
J=J>1;
```

Sumber: https://www.mathworks.com/matlabcentral/fileexchange/19084-region-growing

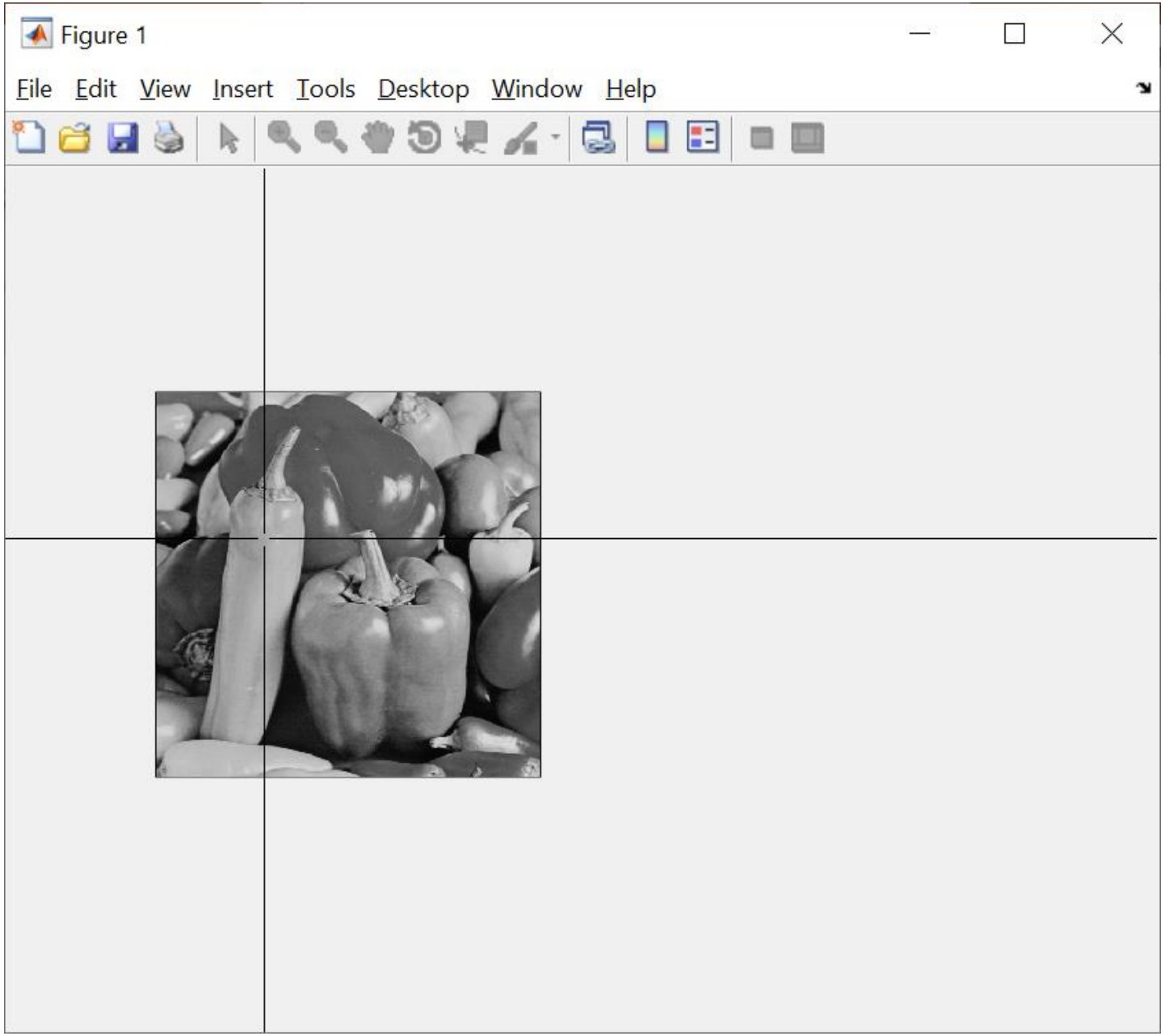


Figure 1
File Edit View Insert Tools Desktop Window Help

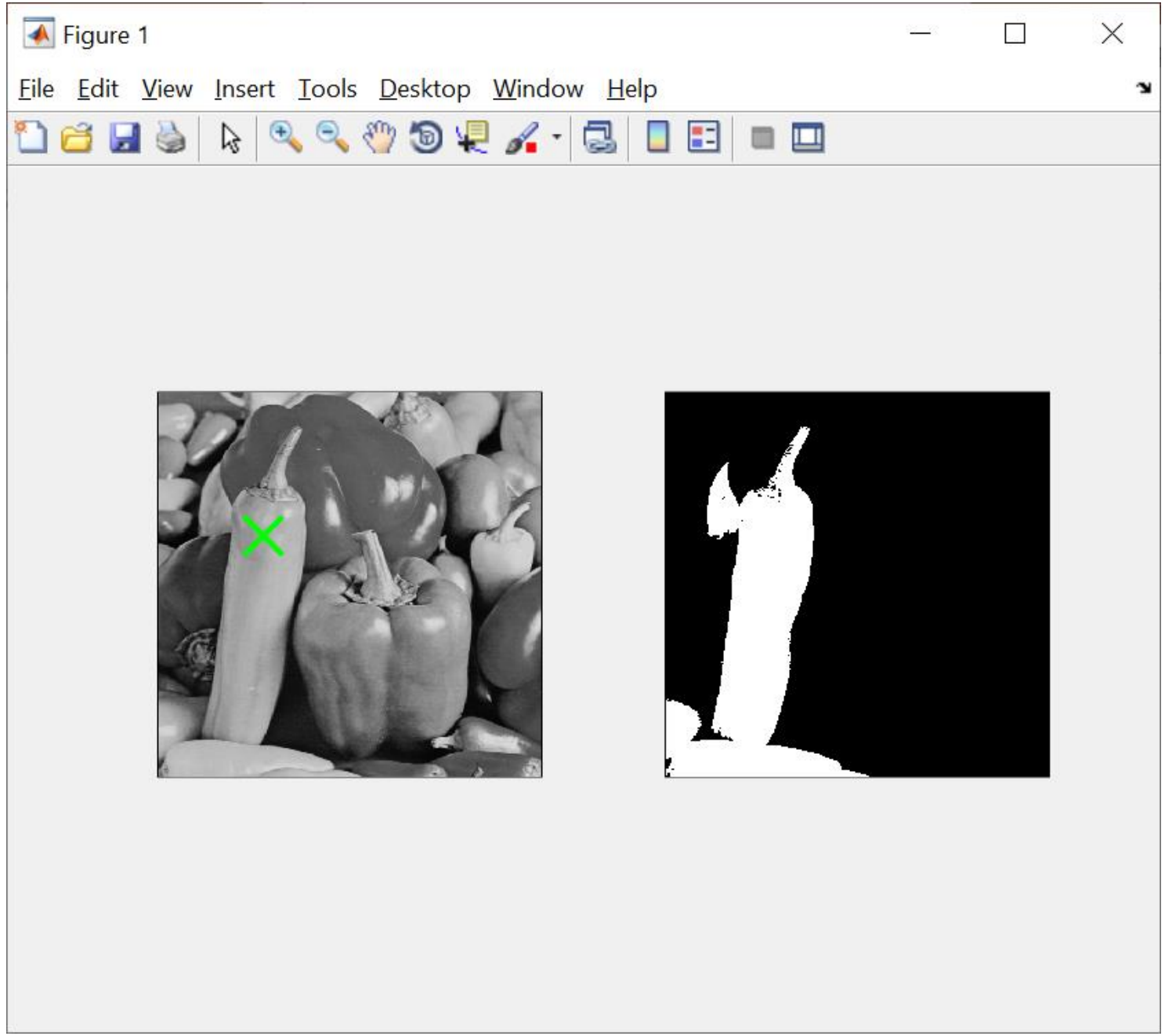
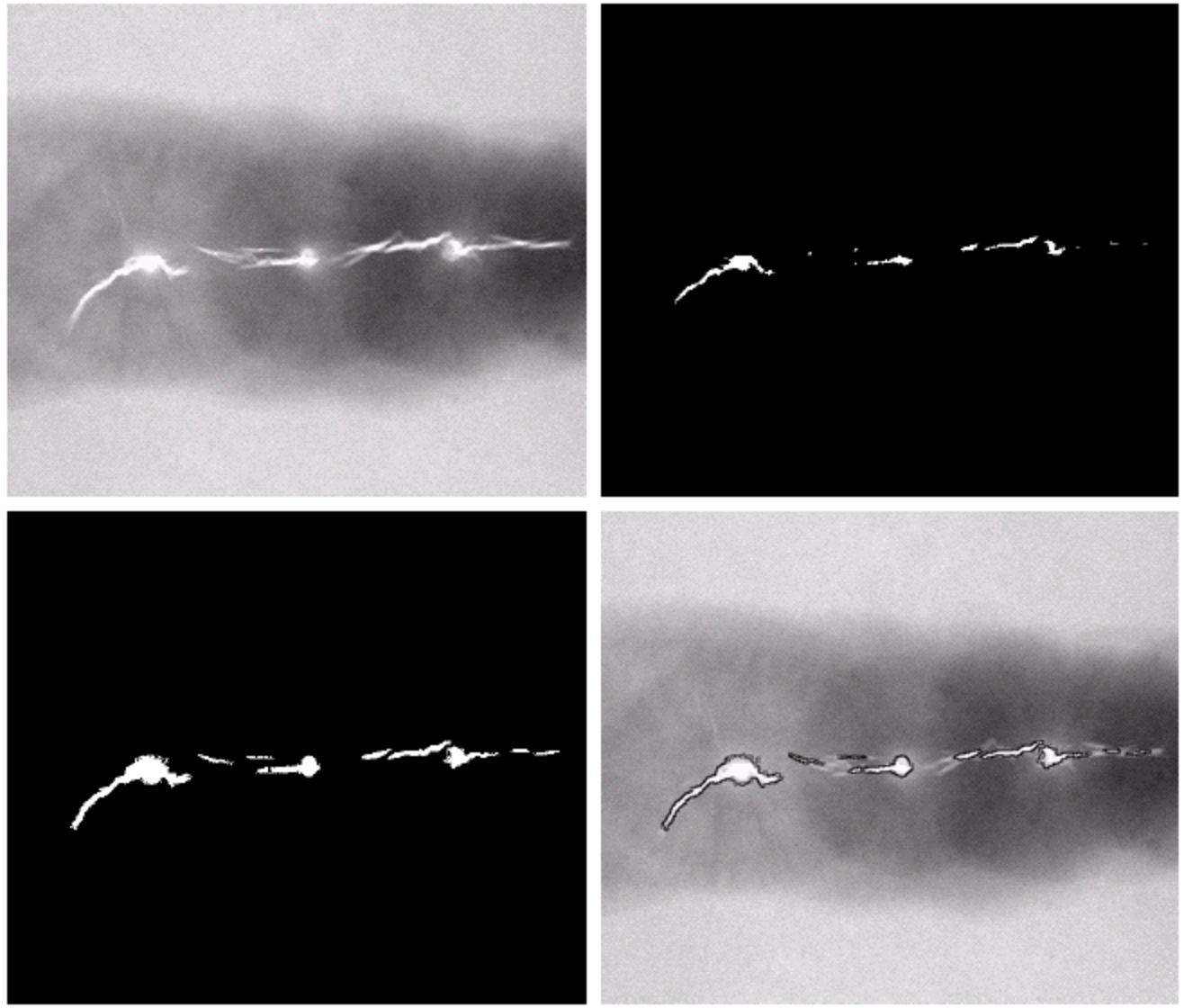


Figure 1
File Edit View Insert Tools Desktop Window Help

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.).



BERSAMBUNG