

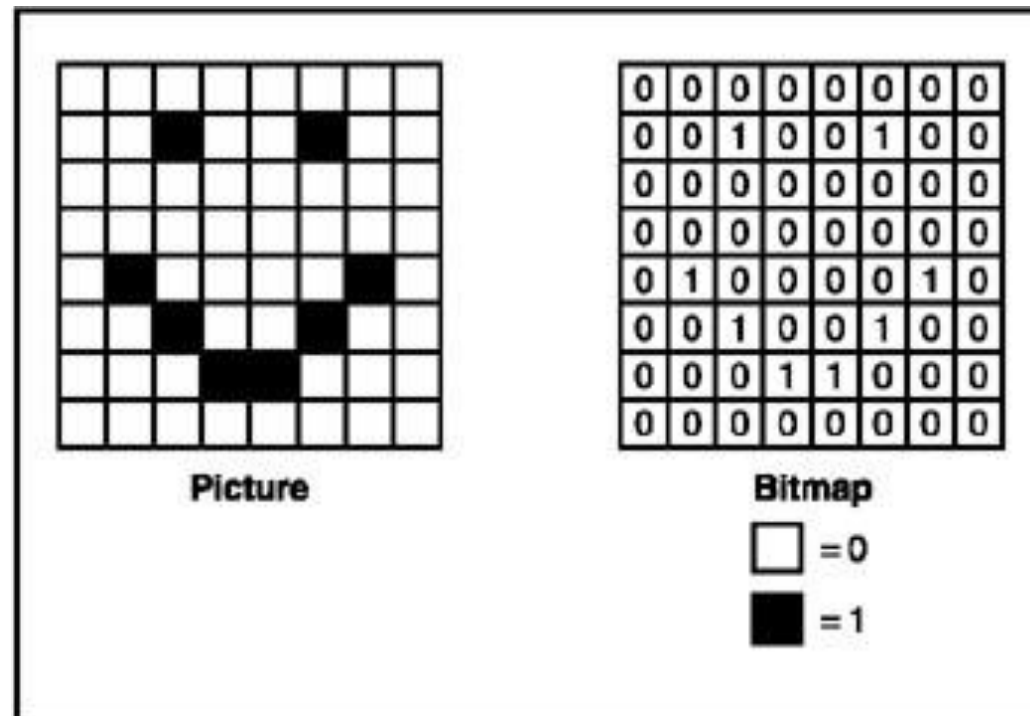
THRESHOLDING

Citra Biner



Citra Biner

- Citra biner adalah citra yang memiliki hanya dua nilai *graylevel*, 0 dan 1
- Hitam = 1, putih = 0, atau sebaliknya

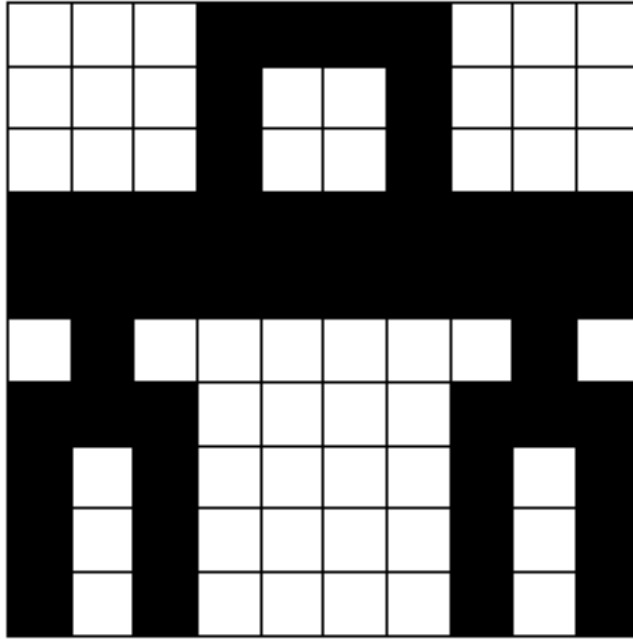


Picture

Bitmap

□ = 0

■ = 1



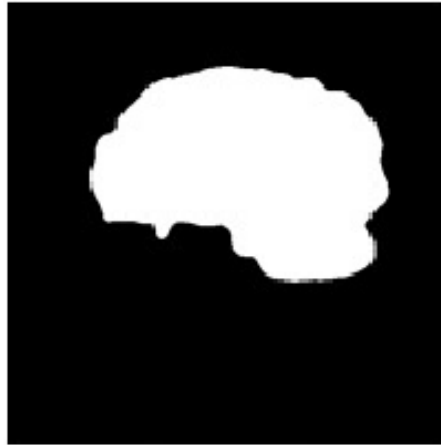
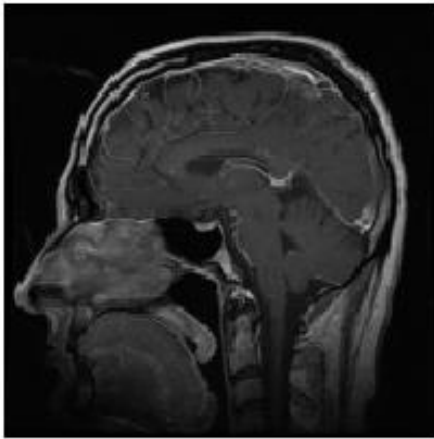
0	0	0	1	1	1	1	0	0	0
0	0	0	1	0	0	1	0	0	0
0	0	0	1	0	0	1	0	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
0	1	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	1	1	1
1	0	1	0	0	0	0	1	0	1
1	0	1	0	0	0	0	1	0	1
1	0	1	0	0	0	0	1	0	1

Mengapa perlu citra biner?

1. Kebutuhan memori untuk setiap pixel sedikit (hanya 1 bit/pixel)
2. Dapat menggunakan operasi logika (AND, OR, NOT) sehingga waktu komputasinya kecil
3. Untuk merepresentasikan citra hasil pendeteksian tepi (*edge images*)

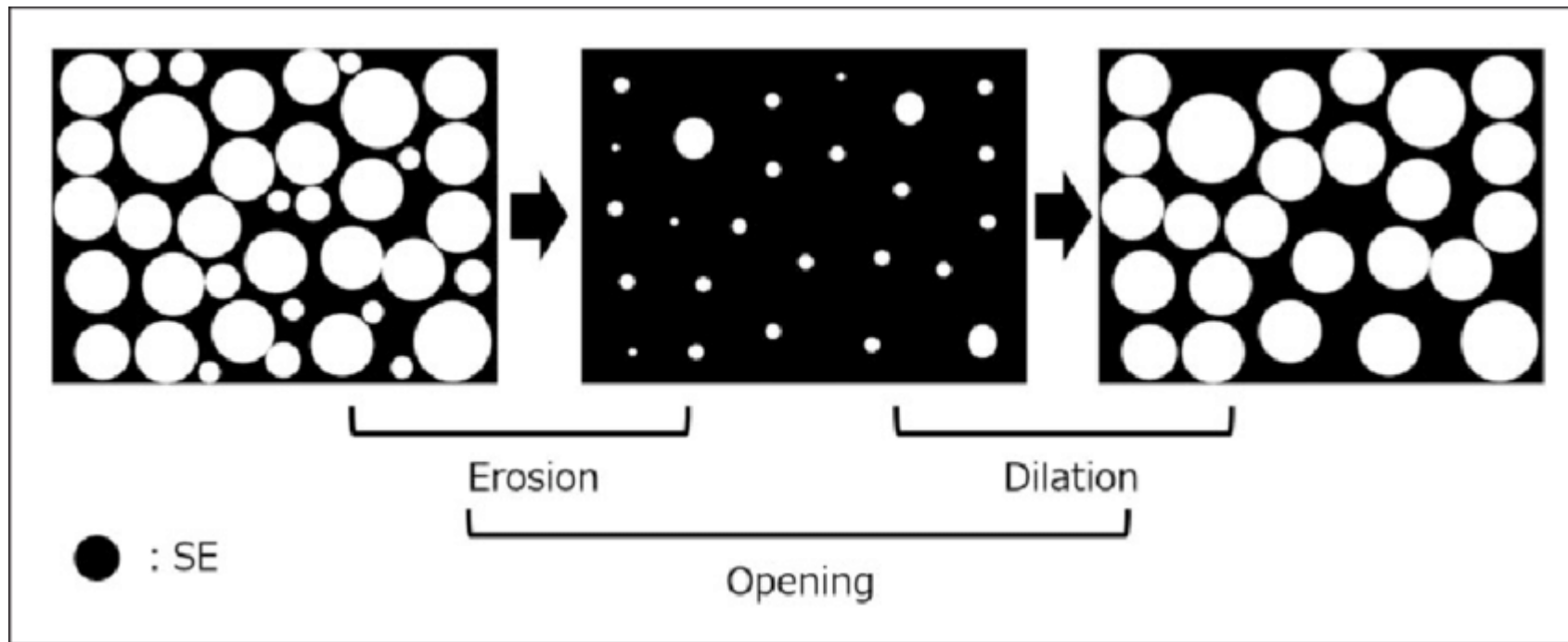


4. Untuk memisahkan (segmentasi) objek dari gambar latar belakangnya





5. Untuk lebih memfokuskan pada analisis bentuk morfologi citra. Setelah objek dipisahkan dari latar belakangnya, properti geometri dan morfologi objek dapat dihitung dari citra biner



Konversi Citra Grayscale ke Citra Biner

- Konversi dari citra grayscale ke citra biner dilakukan dengan operasi pengambangan (*thresholding*).
- Segmentasi yang paling sederhana dilaksanakan dengan menggunakan ambang intensitas.
- Nilai yang lebih kecil daripada nilai ambang diperlakukan sebagai area pertama dan yang lebih besar daripada atau sama dengan nilai ambang dikelompokkan sebagai area yang kedua
- Cara seperti itulah yang disebut peng-ambangan dwi-aras (*bi-level thresholding*) atau terkadang dinamakan peng-ambangan intensitas
- Operasi pengambangan mengelompokkan nilai derajat keabuan setiap *pixel* ke dalam 2 kelas, hitam dan putih.

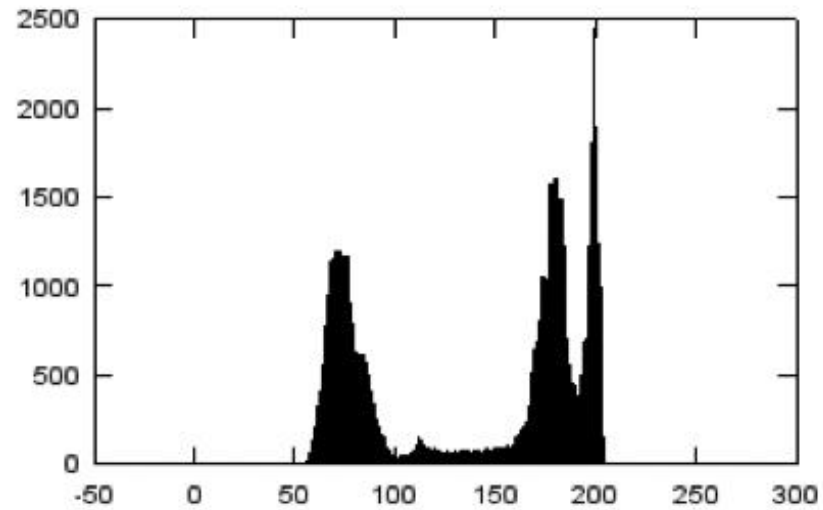
$$b(y, x) = f(x) = \begin{cases} 1, & \text{untuk } (y, x) \geq T \\ 0, & \text{untuk } (y, x) < T \end{cases}$$

- Peng-ambangan intensitas biasa digunakan untuk memisahkan tulisan hitam yang berada di atas secarik kertas putih. Namun, perlu diketahui, peng-ambangan ini mempunyai kelemahan, yaitu:

- 1) tidak memperlihatkan hubungan spasial antarpiksel;
- 2) sensitif terhadap pencahayaan yang tidak seragam;
- 3) hanya berlaku untuk keadaan yang ideal (misalnya, latarbelakang hitam dan objek berwarna putih).



(a) ipomoea.png



(b) Histogram citra

- Salah satu cara untuk menentukan nilai ambang adalah dengan memperhatikan histogram citra.
- Berdasarkan histogram, pemisahan dapat dilakukan dengan memilih nilai ambang pada bagian lembah.
- Sebagai contoh, nilai di sekitar 100 dapat digunakan sebagai nilai ambang.

Penggunaan 1-G dalam imshow ditujukan untuk membalik nilai 1 dan 0. Dengan ungkapan tersebut, bagian daun akan diberi nilai 1 (putih).



(a) ipomoea.tif



(b) Hasil pengambangan

Contoh peng-ambangan dengan nilai ambang sebesar 100:

```
>> Img = imread('C:\Image\ipomoea.png');  
>> G = ambang(Img, 100);  
>> imshow(1-G)
```

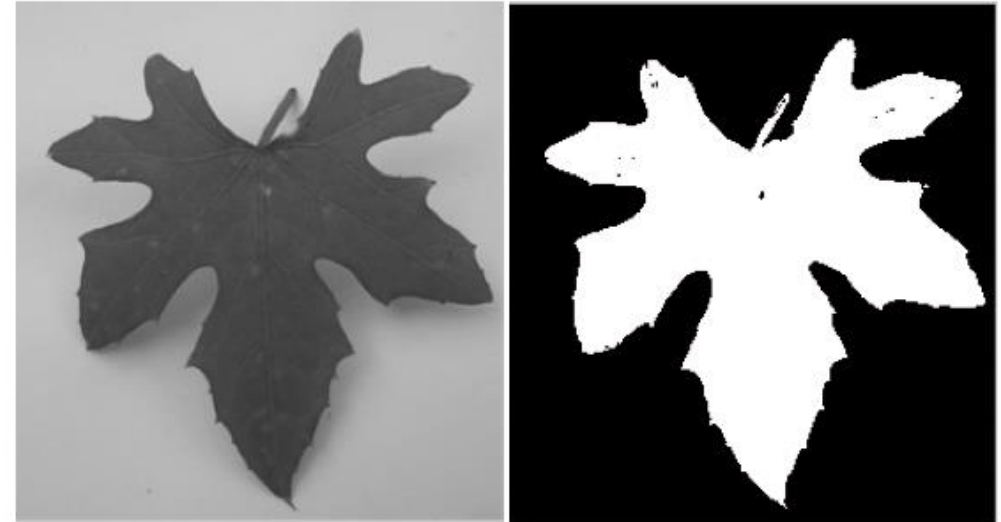
- Persoalan utama dalam peng-ambangan intensitas terletak pada penentuan nilai ambang (T) secara otomatis.



(a) ipomoea.tif

(b) Hasil peng-ambangan

Hasil di atas menyatakan bahwa nilai ambang yang digunakan adalah 130



(a) ipomoea.tif

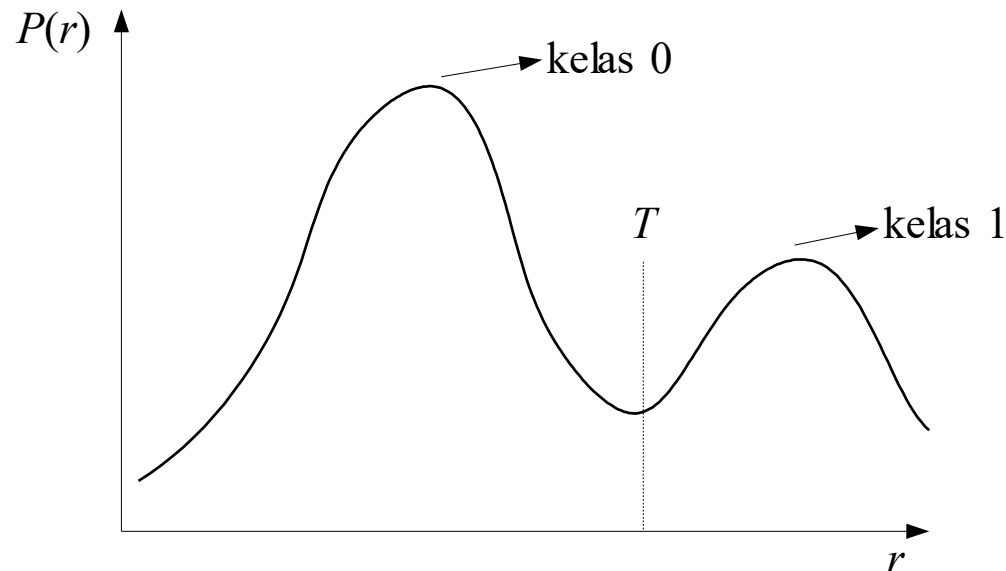
(b) Hasil pengambangan

Hasil di atas menyatakan bahwa nilai ambang yang digunakan adalah 100

Macam operasi pengembangan:

1. Pengembangan secara global

- Nilai T berlaku untuk seluruh bagian di dalam citra
- Nilai T dipilih sedemikian sehingga galat sekecil mungkin
- Jika citra mengandung satu atau lebih objek dan latar belakang dengan intensitas yang homogen, maka histogramnya *bimodal* (memiliki dua puncak) → sekaligus melakukan segmentasi objek



Contoh pengembangan Global → Binary Thresholding

- Metode segmentasi yang sangat sederhana terdiri dari mengasosiasikan dengan setiap piksel gambar sebuah bilangan biner yang bergantung pada intensitas piksel dan pada ambang

$$g(m, n) = \begin{cases} 1 & \text{if } f(m, n) \geq T, \\ 0 & \text{if } f(m, n) < T \end{cases}$$

Misalkan terdapat citra grayscale berukuran 4×4 :

$$I = \begin{bmatrix} 40 & 50 & 60 & 70 \\ 80 & 90 & 110 & 120 \\ 130 & 140 & 150 & 160 \\ 170 & 180 & 190 & 200 \end{bmatrix}$$

Misalkan ditentukan:

$$T = 120$$

Aturannya:

- Jika nilai piksel ≥ 120 , maka menjadi **255 (putih)**
- Jika nilai piksel < 120 , maka menjadi **0 (hitam)**

Sehingga hasil akhirnya:

$$I_{binary} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 255 \\ 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 \end{bmatrix}$$

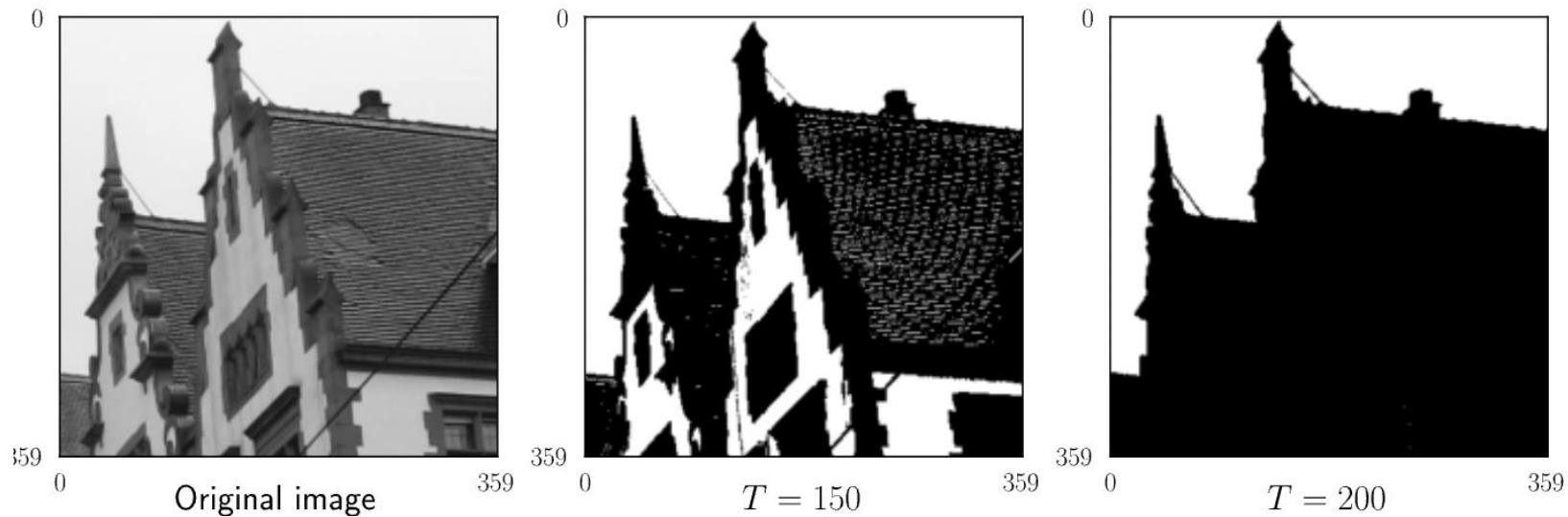
Intinya

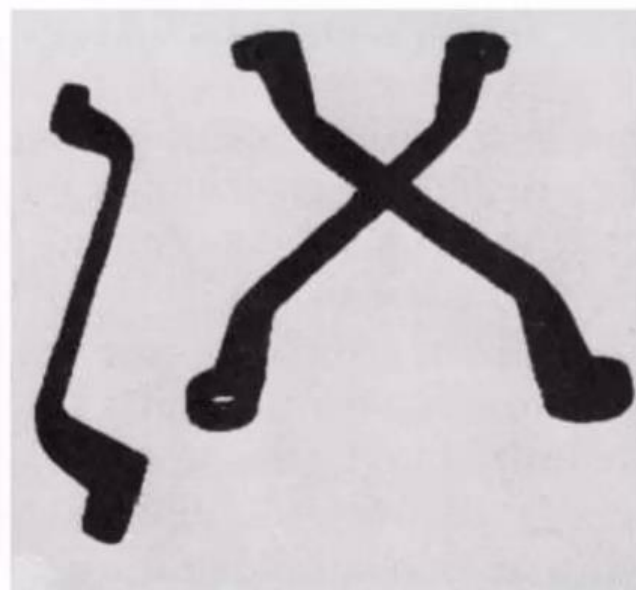
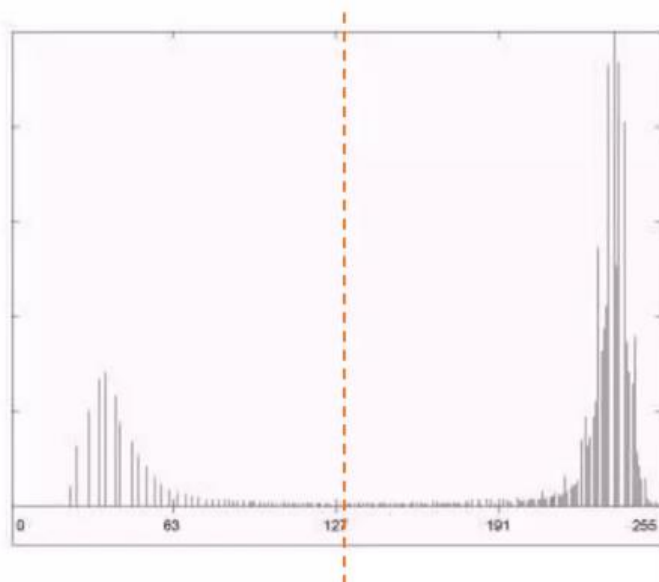
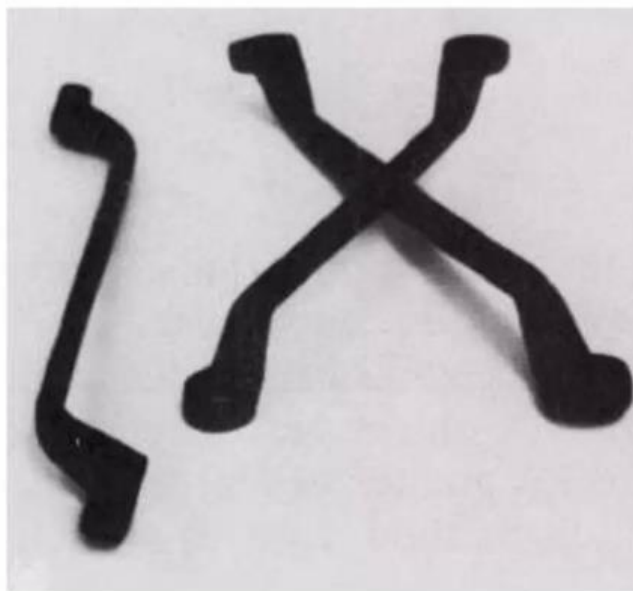
Pada **Global Thresholding**, nilai $T = 120$ diterapkan **sama untuk seluruh citra**.

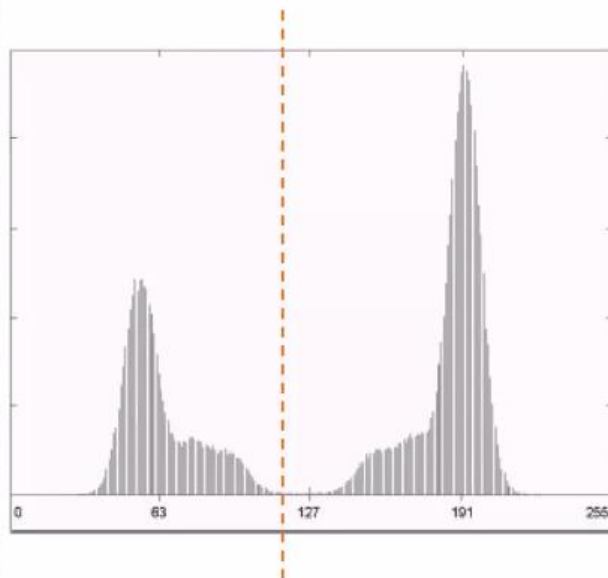
Kelebihan: sederhana dan cepat.

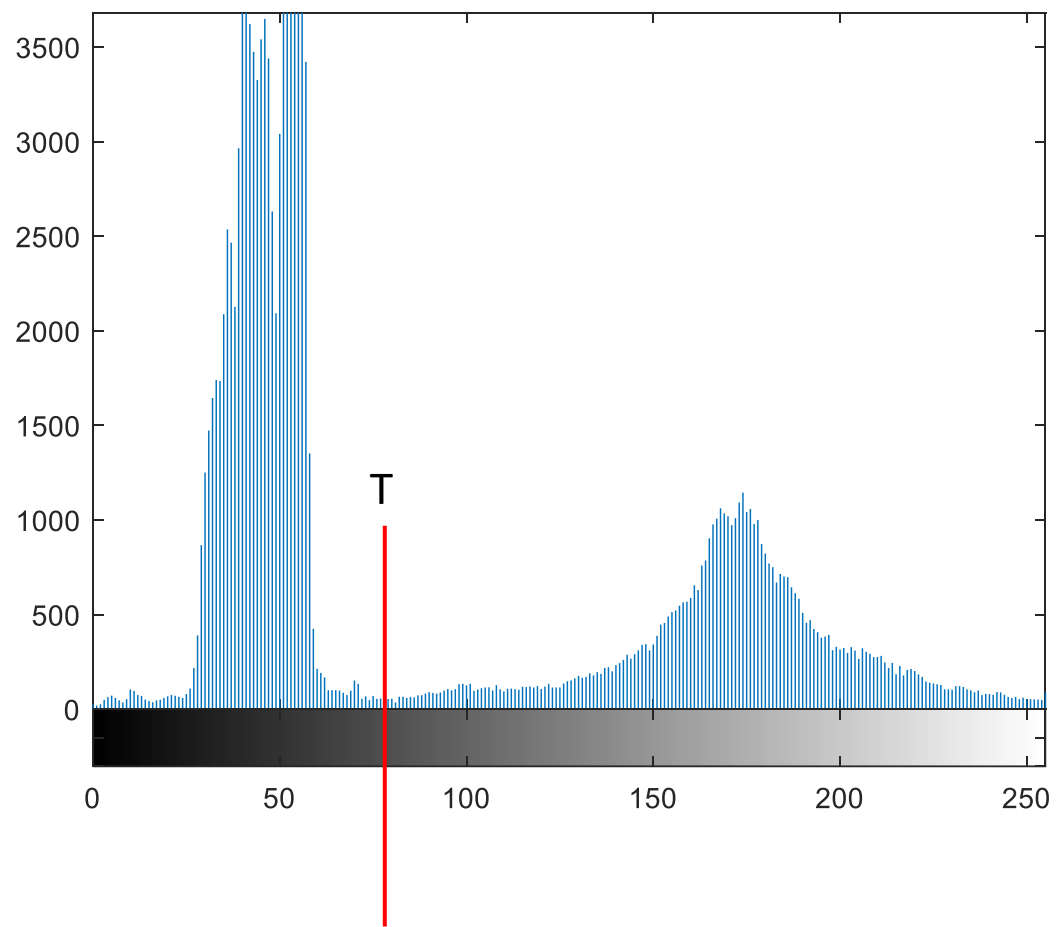
Kekurangan: kurang efektif apabila pencahayaan tidak merata.

- Metode ini disebut “binarisasi”. Metode ini memberikan segmentasi menjadi dua kelas, tergantung pada intensitas piksel dari gambar skala abu-abu



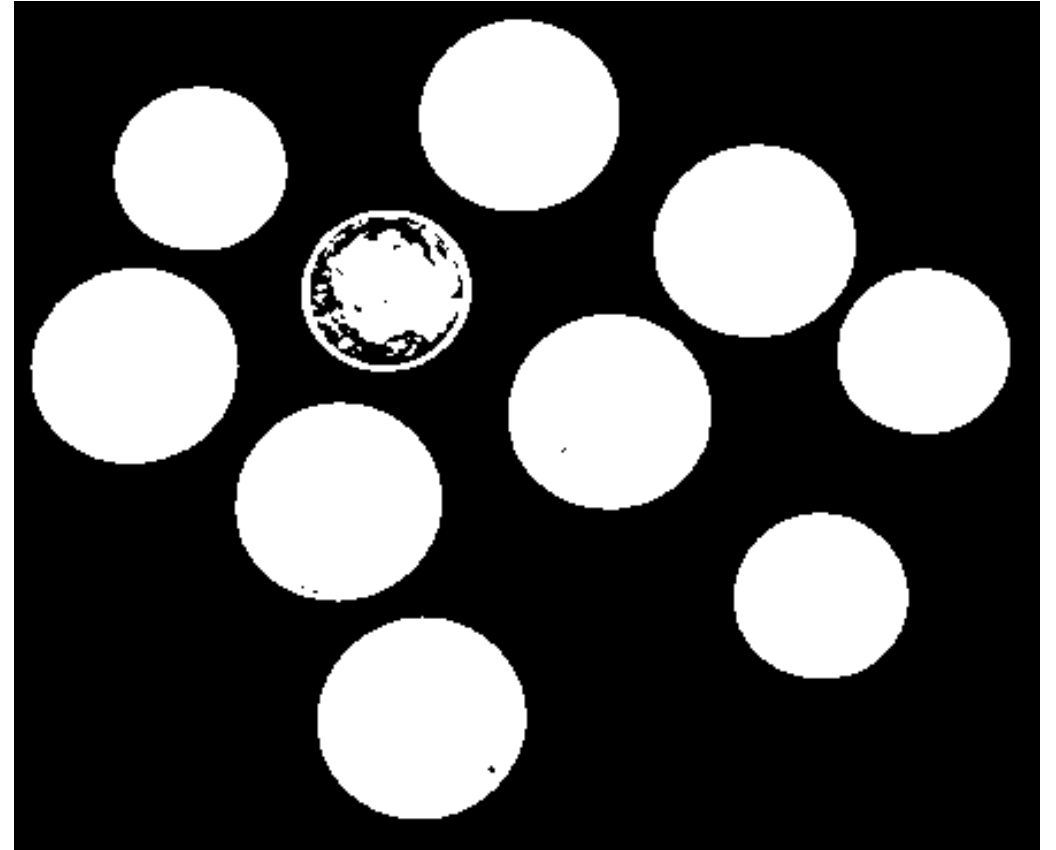






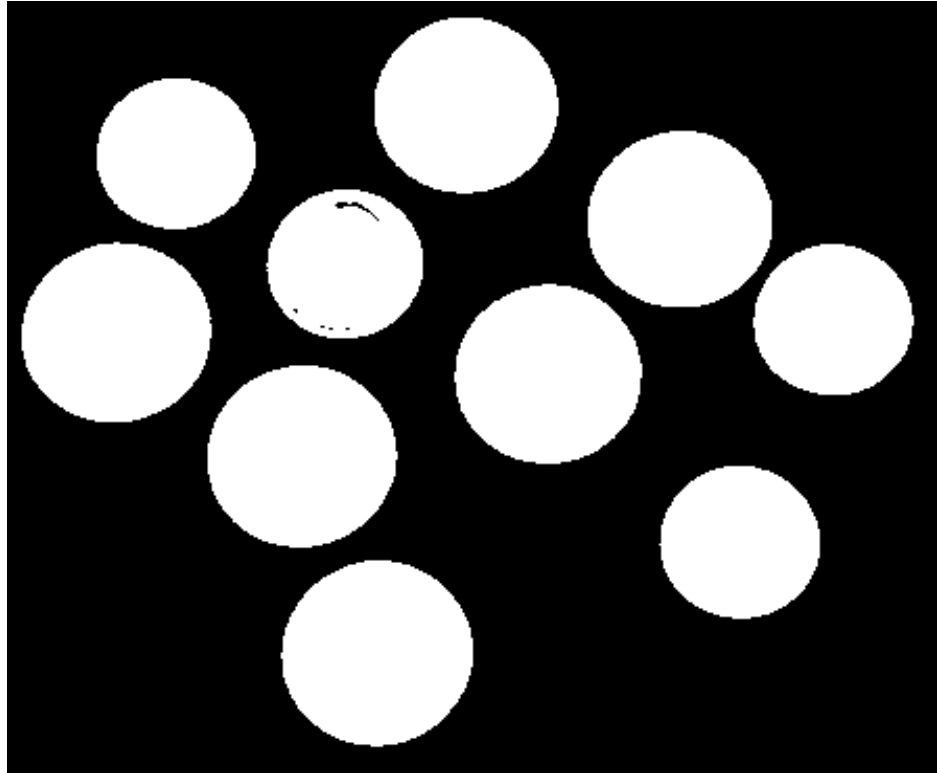
T = 100

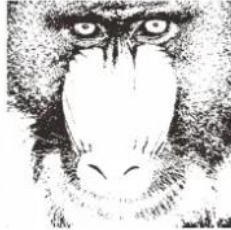
```
>> I = imread('coins.bmp');  
>> imhist(I)  
>> BW = im2bw(I, 100/255);  
>> figure, imshow(BW)
```



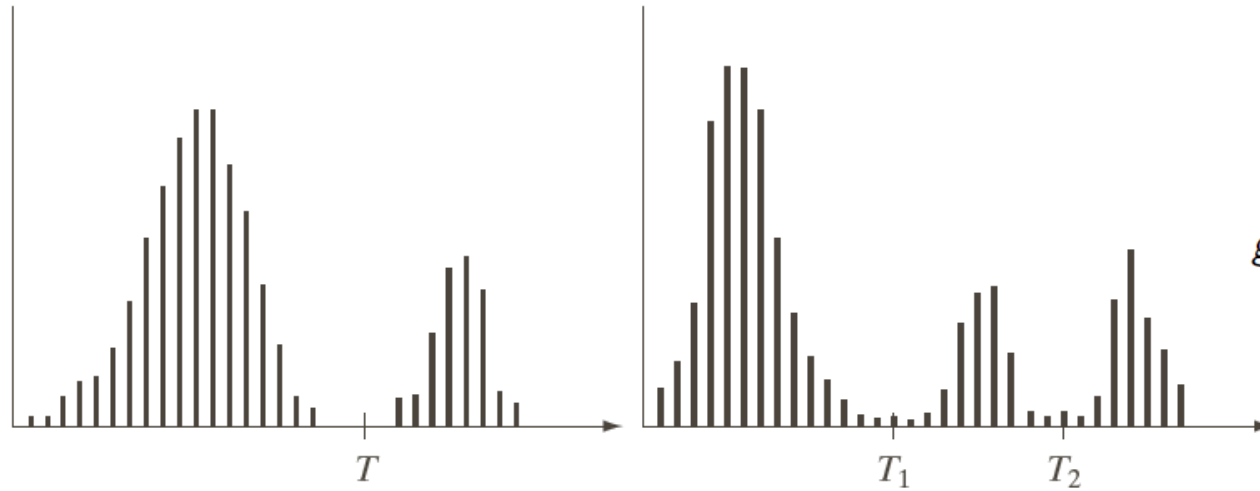
T = 75

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



Nama Citra	Citra Asli	T = 50	T = 100	T = 150	T = 200
cameraman					
lena					
mandril					
girl					
bicycle					

- Kasus jika terdapat lebih dari satu lembah dan dua bukit $\rightarrow$ lebih dari satu T :



$$g(x, y) = \begin{cases} a, & \text{if } f(x, y) > T_2 \\ b, & \text{if } T_1 < f(x, y) \leq T_2 \\ c, & \text{if } f(x, y) \leq T_1 \end{cases}$$

2. Pengambangan secara local/ **Adaptive Thresholding**

- Pengambangan secara lokal dilakukan terhadap daerah-daerah 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 daerah citra yang berukuran 5×5 atau 8×8 *pixel*. Nilai ambangnya ditentukan sebagai fungsi rata-rata derajat keabuan di dalam daerah citra tersebut.

Pada Local atau Adaptive Thresholding, nilai threshold tidak sama untuk seluruh citra. Threshold dihitung berdasarkan kondisi lokal di sekitar setiap piksel.

Jadi, bukan:

$$T = \text{satu nilai untuk seluruh citra}$$

tetapi:

$$T = T(x, y)$$

Artinya, setiap bagian citra dapat mempunyai nilai threshold yang berbeda.

Metode ini cocok untuk citra yang memiliki pencahayaan tidak merata, misalnya dokumen yang sebagian terang dan sebagian gelap.

Contoh metode:

- Mean Adaptive → ambang = rata-rata nilai piksel di sekitar titik.
- Gaussian Adaptive → ambang = rata-rata tertimbang Gaussian di sekitar titik.

Misalkan citra grayscale: (Mean Adaptif)

$$I = \begin{bmatrix} 40 & 50 & 60 & 70 \\ 50 & 80 & 90 & 100 \\ 60 & 90 & 150 & 160 \\ 70 & 100 & 160 & 180 \end{bmatrix}$$

Kita gunakan window 3×3 dan metode sederhana berupa **rata-rata lokal**.
Untuk setiap piksel, threshold dihitung berdasarkan rata-rata nilai piksel pada area 3×3di sekitarnya:

$$T(x,y) = \frac{\sum I_{window}}{N}$$

Contoh menghitung piksel tengah P(2, 2)

Area 3×3di sekitar P(2, 2):

$$\begin{bmatrix} 40 & 50 & 60 \\ 50 & 80 & 90 \\ 60 & 90 & 150 \end{bmatrix}$$

Jumlah seluruh nilai:

$$40 + 50 + 60 + 50 + 80 + 90 + 60 + 90 + 150 = 670$$

Jumlah piksel = 9, sehingga:

$$T(2,2) = \frac{670}{9} = 74,44$$

Nilai piksel yang diproses:

$$I(2,2) = 80$$

Karena:

$$80 > 74,44$$

maka:
g(2,2)=255

Contoh piksel lain P(3, 3)

Window 3×3:

$$\begin{bmatrix} 80 & 90 & 100 \\ 90 & 150 & 160 \\ 100 & 160 & 180 \end{bmatrix}$$

Jumlah:

$$80 + 90 + 100 + 90 + 150 + 160 + 100 + 160 + 180 = 1110$$

Threshold lokal:

$$T(3,3) = \frac{1110}{9} = 123,33$$

Nilai piksel:

$$I(3,3) = 150$$

Karena:

$$150 > 123,33$$

maka:
g(3,3)=255

Perbedaannya, Gaussian Adaptive tidak memberikan bobot yang sama kepada semua piksel dalam window. Piksel yang lebih dekat dengan titik yang sedang diproses diberi bobot lebih besar, sedangkan piksel yang lebih jauh diberi bobot lebih kecil.

Misalnya digunakan kernel Gaussian sederhana 3×3 :

$$G = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Jumlah bobot:

$$1+2+1+2+4+2+1+2+1=16$$

Misalnya citra:

$$I = \begin{bmatrix} 40 & 50 & 60 & 50 & 40 \\ 50 & 60 & 70 & 60 & 50 \\ 60 & 70 & 100 & 70 & 60 \\ 50 & 60 & 70 & 60 & 50 \\ 40 & 50 & 60 & 50 & 40 \end{bmatrix}$$

Gunakan window citra :

$$\begin{bmatrix} 60 & 70 & 60 \\ 70 & 100 & 70 \\ 60 & 70 & 60 \end{bmatrix}$$

Langkah 1 — Kalikan setiap piksel dengan bobot Gaussian

$$\begin{bmatrix} 60(1) & 70(2) & 60(1) \\ 70(2) & 100(4) & 70(2) \\ 60(1) & 70(2) & 60(1) \end{bmatrix}$$

Sehingga:

$$\begin{bmatrix} 60 & 140 & 60 \\ 140 & 400 & 140 \\ 60 & 140 & 60 \end{bmatrix}$$

Langkah 2 — Jumlahkan hasil perkalian

$$60 + 140 + 60 + 140 + 400 + 140 + 60 + 140 + 60 = 1200$$

Langkah 3 — Bagi dengan total bobot

$$T = \frac{1200}{16} = 75$$

Jadi threshold Gaussian lokal:

$$T = 75$$

Langkah 4 — Bandingkan dengan piksel tengah

Nilai piksel tengah:

$$I(3, 3) = 100$$

Karena:

$$100 > 75$$

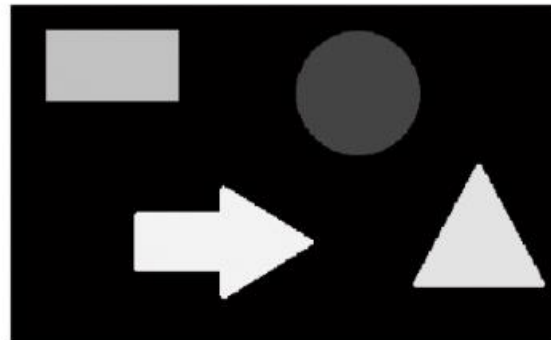
maka:

$$g(3, 3) = 255$$

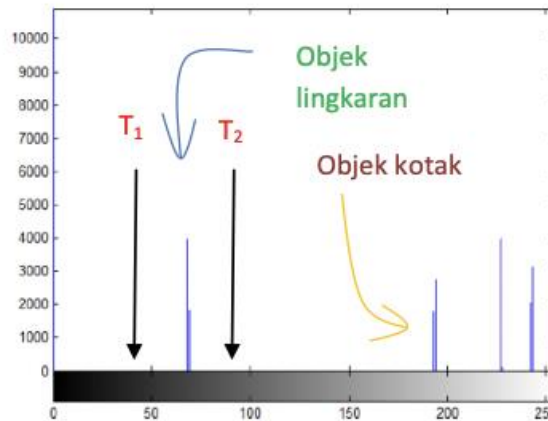
Piksel tersebut menjadi **putih/objek**.

3. Pengembangan multilevel/ Multilevel Thresholding

- Digunakan jika citra memiliki lebih dari dua objek dengan tingkat keabuan berbeda. Misalnya, membagi citra menjadi tiga atau lebih kelas: background, objek 1, objek 2, dll.
- citra dibagi menjadi beberapa bagian dengan menggunakan beberapa nilai ambang, Cara seperti itu dilakukan kalau pada histogram terdapat puncak-puncak yang membedakan antara satu objek terhadap yang lain.



(a) benda.png



(b) Histogram

Contoh citra dengan beberapa puncak dan lembah yang terpisah

Misalnya kita menggunakan dua threshold:

$$T_1 = 80, T_2 = 160$$

Maka citra dibagi menjadi tiga kelompok:

$$g(x, y) = \begin{cases} 0, & f(x, y) < 80 \\ 127, & 80 \leq f(x, y) < 160 \\ 255, & f(x, y) \geq 160 \end{cases}$$

Artinya:

- Intensitas 0–79 → kelompok 1 → 0
- Intensitas 80–159 → kelompok 2 → 127
- Intensitas 160–255 → kelompok 3 → 255

Jadi, berbeda dengan thresholding biasa yang menghasilkan 2 kelas, multilevel thresholding dapat menghasilkan 3 kelas atau lebih.

Misalkan terdapat citra grayscale 4×4:

$$I = \begin{bmatrix} 30 & 50 & 70 & 90 \\ 80 & 100 & 120 & 140 \\ 150 & 160 & 180 & 190 \\ 200 & 210 & 220 & 240 \end{bmatrix}$$

Kita tentukan:

$T_1 = 80$ dan $T_2=160$

Langkah 1 — Tentukan aturan

Kita buat tiga kelas:

Rentang intensitas	Kelas	Nilai output
0 – 79	Background/kelas 1	0
80 – 159	Kelas 2	127
160 – 255	Kelas 3	255

Langkah 2 — Proses setiap piksel

Contoh:

Piksel 30

$$30 < 80$$

maka:

$$30 \rightarrow 0$$

Piksel 100

$$80 \leq 100 < 160$$

maka:

$$100 \rightarrow 127$$

Piksel 180

$$180 \geq 160$$

maka:

180 → 255

Langkah 3 — Hasil akhir

Maka:

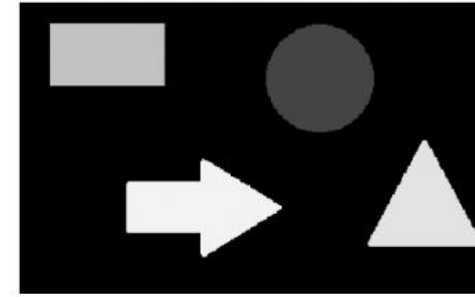
30	50	70	90
80	100	120	140
150	160	180	190
200	210	220	240

menjadi:

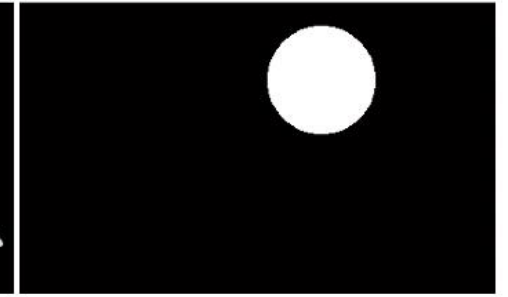
0	0	0	127
127	127	127	127
127	255	255	255
255	255	255	255

- Multilevel Thresholding adalah teknik pengembangan yang menggunakan dua atau lebih nilai threshold untuk membagi citra grayscale menjadi lebih dari dua kelompok intensitas. Semakin banyak threshold yang digunakan, semakin banyak kelas intensitas yang dapat dibentuk.
- Teknik ini berguna ketika citra tidak cukup direpresentasikan hanya sebagai objek dan background, tetapi membutuhkan beberapa kelompok intensitas, misalnya background, objek utama, dan objek dengan intensitas berbeda.

```
>> Img = imread('C:\Image\benda.png');  
>> G = multi_thres(Img, 50, 100); imshow(G);
```



(a) benda.png



(b) Nilai ambang $t_1=50$ dan $t_2=100$



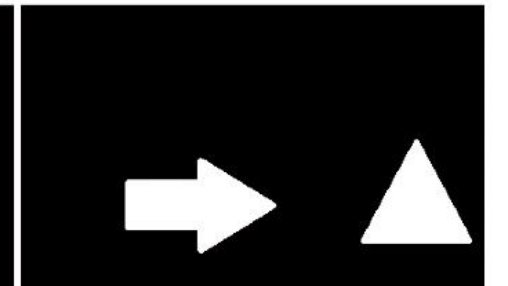
(c) Nilai ambang $t_1=175$ dan $t_2=200$



(d) Nilai ambang $t_1=220$ dan $t_2=230$

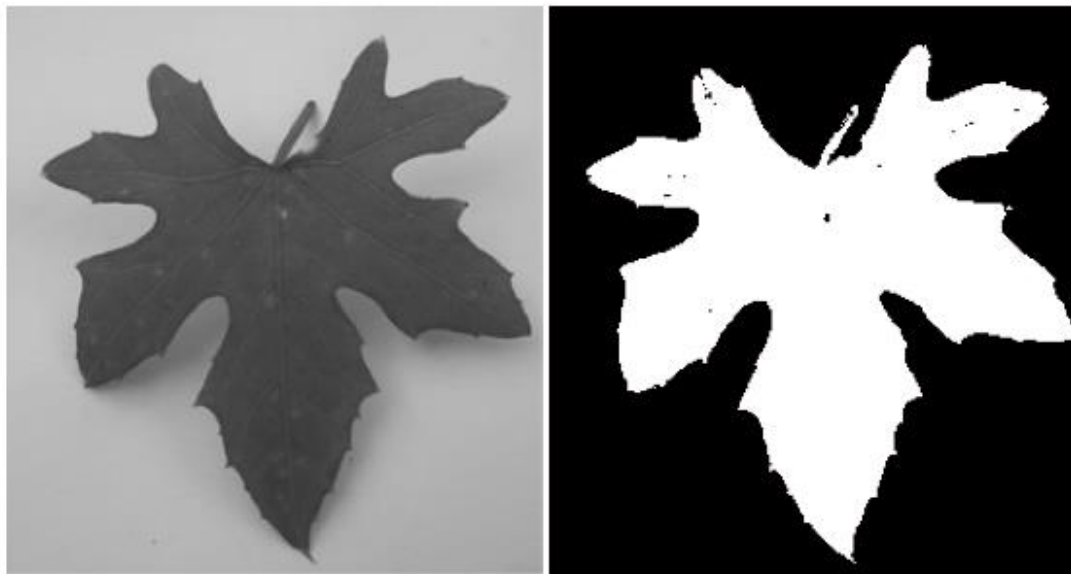


(e) Nilai ambang $t_1=235$ dan $t_2=250$



(f) Nilai ambang $t_1=220$ dan $t_2=250$

Hasil penerapan nilai ambang jamak



(a) ipomoea.png

(b) Pengambangan $t_1=50$, $t_2=100$

1. Original Grayscale



Citra asli dalam bentuk grayscale
(nilai intensitas 0–255).

2. Binary Threshold



Citra hasil threshold biner dengan nilai ambang
(misalnya 128).

3. Adaptive Threshold



Citra hasil adaptive threshold, di mana nilai ambang
d disesuaikan secara lokal pada setiap area.

4. Multilevel Threshold

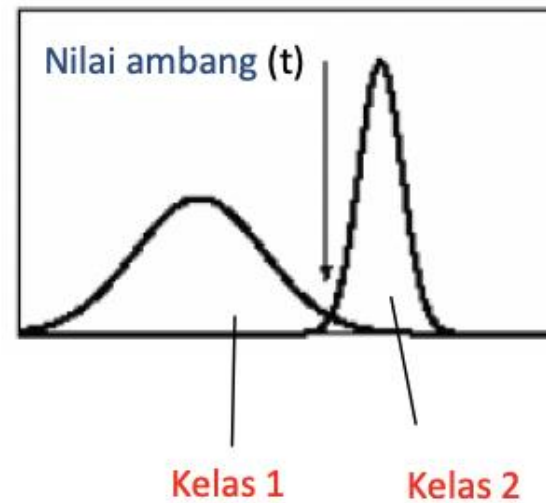


	Level 1 (0–63)
	Level 2 (64–127)
	Level 3 (128–191)
	Level 4 (192–255)

Citra hasil multilevel threshold dengan beberapa
tingkat ambang (misalnya 4 level).

Otsu Thresholding

- Metode Otsu dipublikasikan oleh Nobuyuki Otsu pada tahun 1979. Metode ini menentukan nilai ambang dengan cara membedakan dua kelompok, yaitu objek dan latar belakang, yang memiliki bagian yang saling bertumpukan, berdasarkan histogram

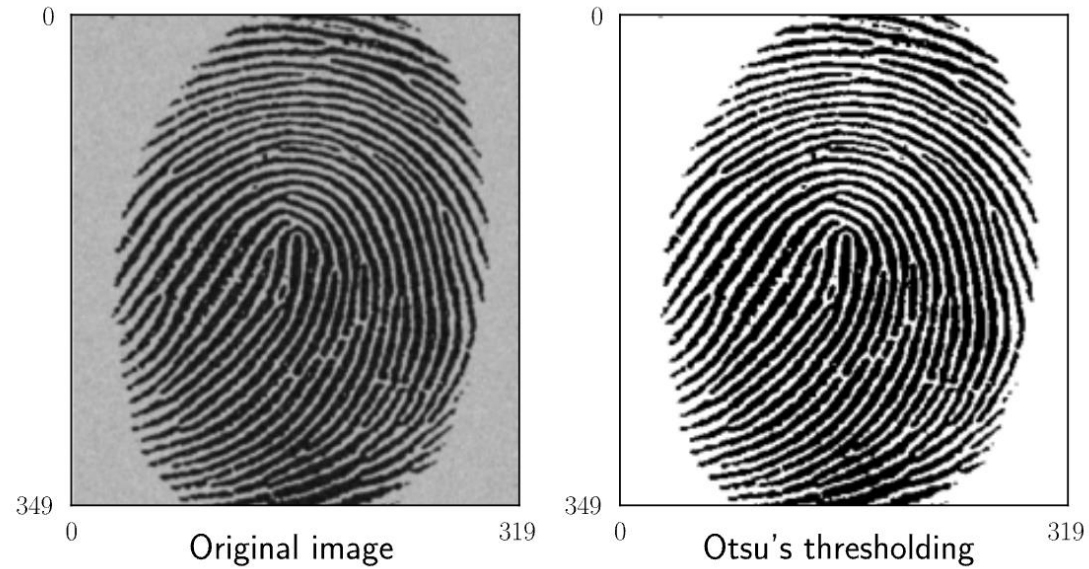


Penentuan nilai ambang

untuk memperoleh hasil yang optimal

- Otsu Thresholding adalah metode otomatis untuk menentukan nilai threshold (ambang batas) terbaik pada citra grayscale sehingga citra dapat dipisahkan menjadi dua kelompok, yaitu foreground (objek) dan background (latar belakang).
- Berbeda dengan binary threshold biasa, pada Otsu kita tidak menentukan nilai threshold secara manual, misalnya 128. Nilai threshold dicari secara otomatis berdasarkan distribusi intensitas piksel pada citra.
- Metode otsu merupakan metode untuk menentukan titik ambang batas optimal untuk melakukan thresholding.
- Hal ini dilakukan dengan meminimalkan variansi atau memaksimalkan variansi antarkelas dari kelas-kelas yang dipisahkan dengan threshold.
- Untuk melakukannya, algoritma ini menghitung variansi antarkelas untuk setiap nilai threshold yang mungkin.

- `>> Img = imread('C:\Image\ipomoea.png');`
- `>> t = otsu(Img)`
- `t = 130`

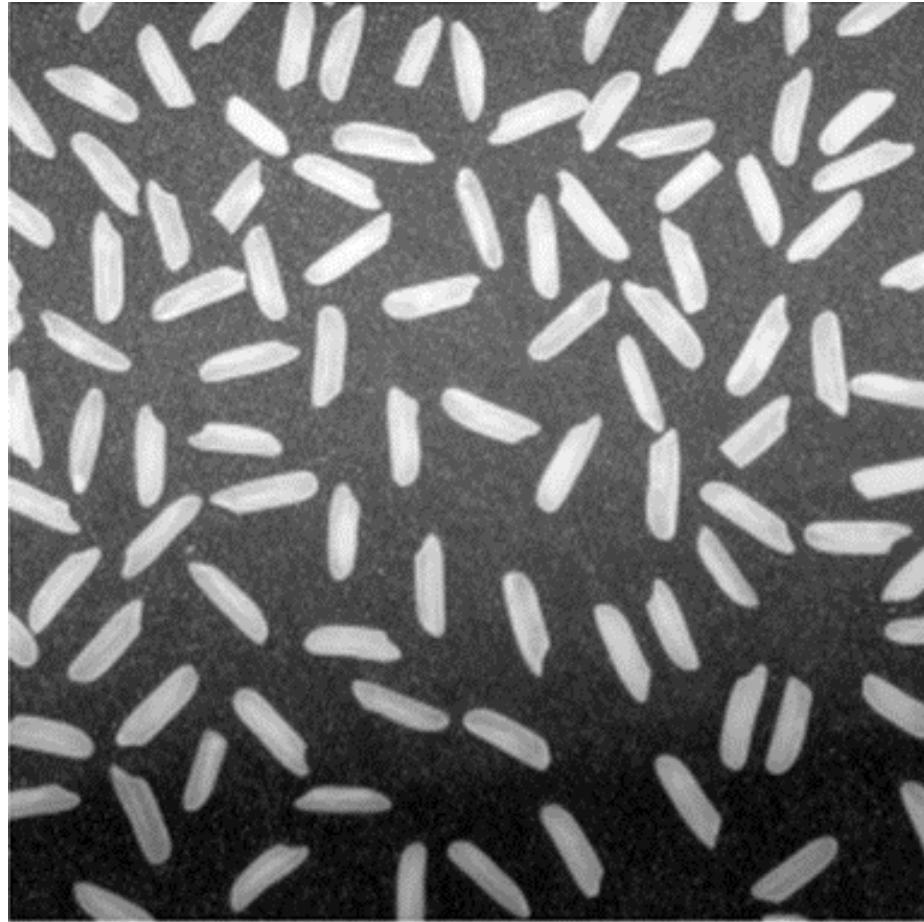


Kegunaan Citra Biner Hasil Pengambangan

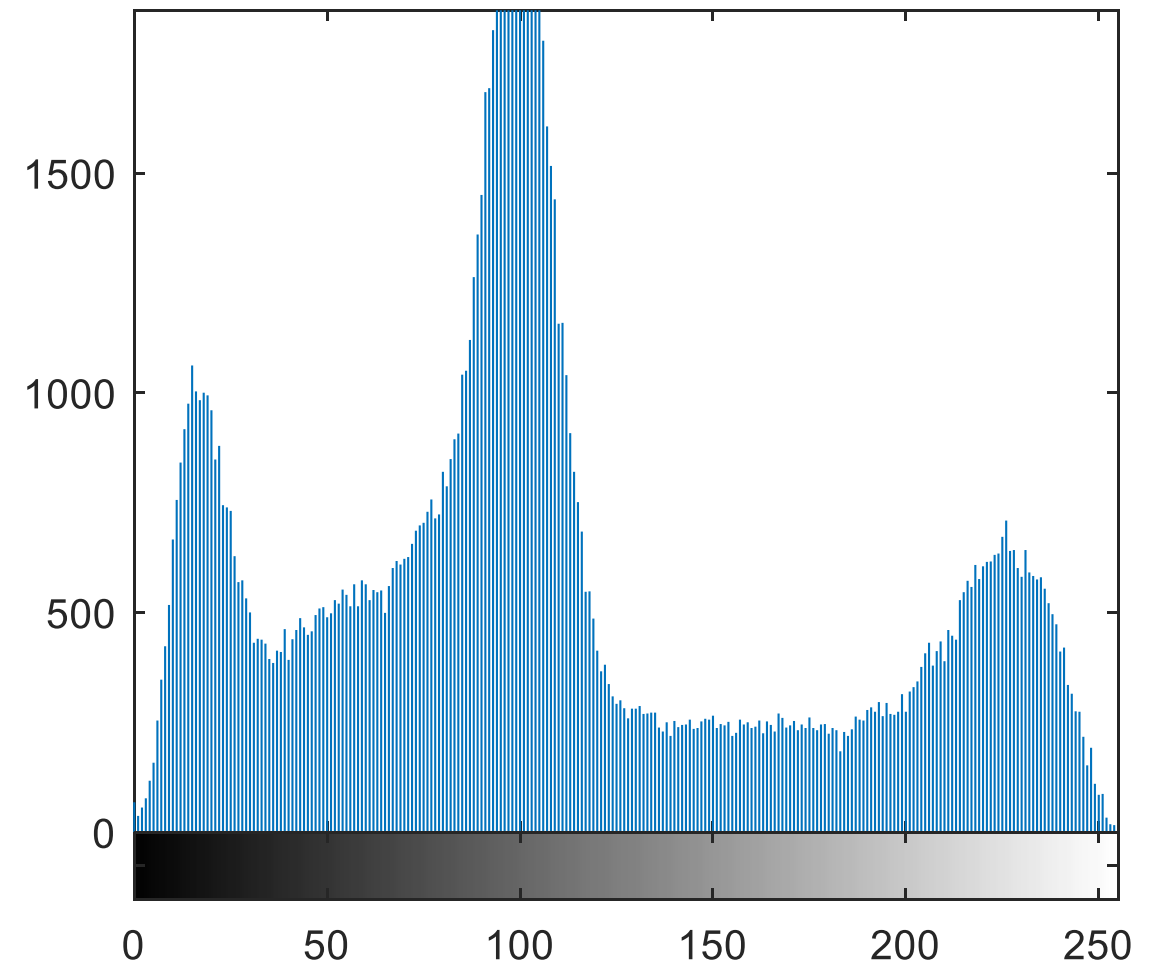
- Citra biner hasil pengambangan berguna untuk memisahkan objek dengan latar belakang pada citra asalnya.
- Citra biner menjadi *template* untuk melakukan segmentasi



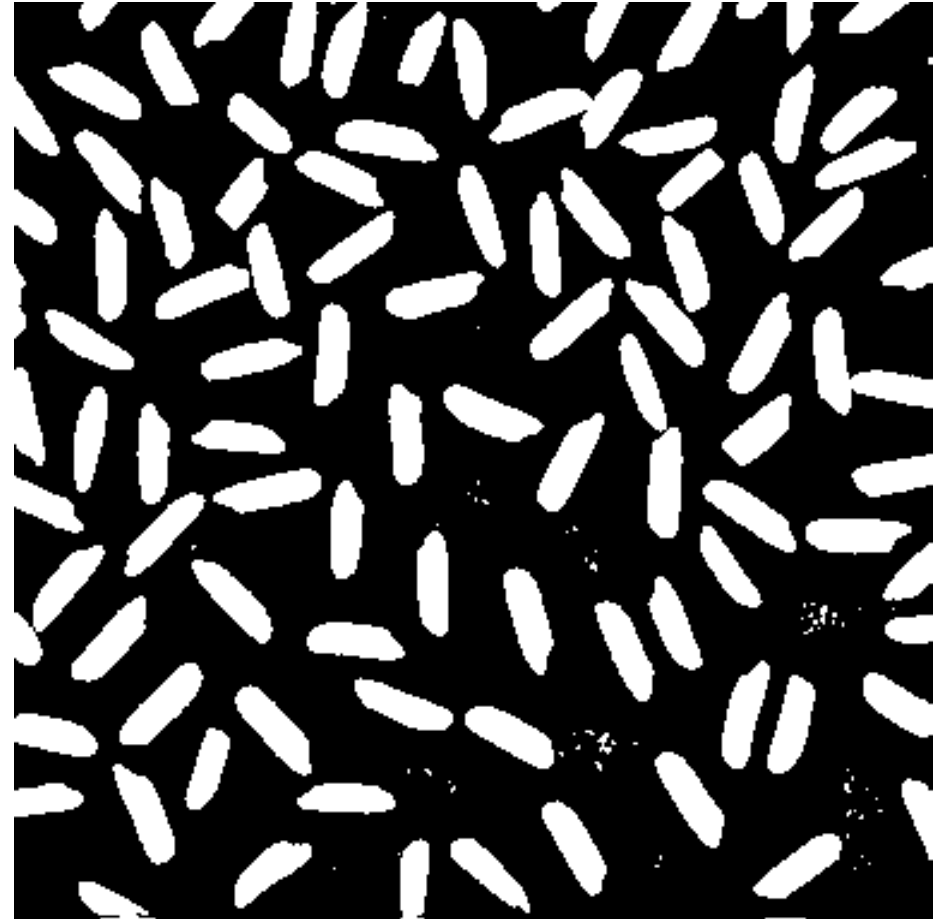
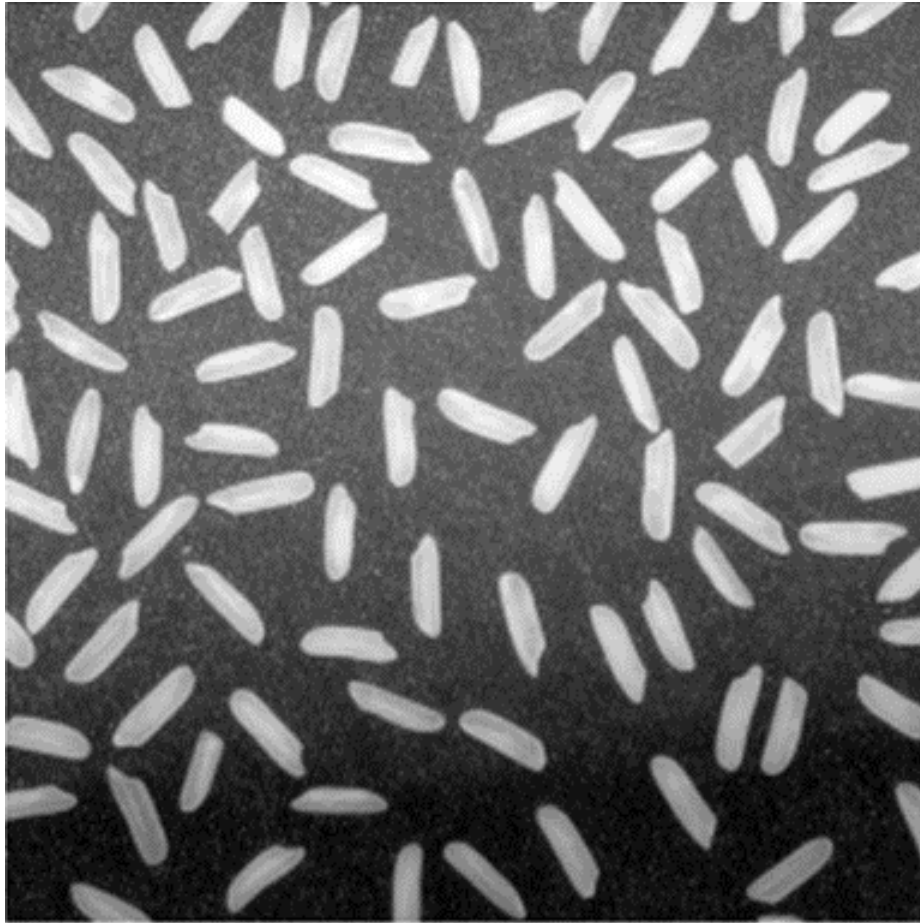
```
I = imread('rice.bmp');
```



```
imhist(I)
```



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

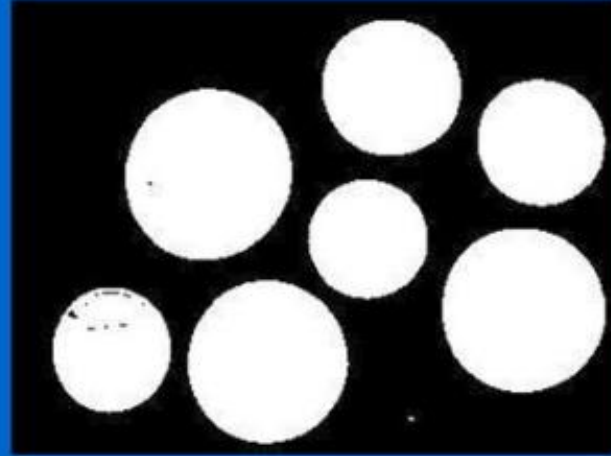


How to choose the threshold?

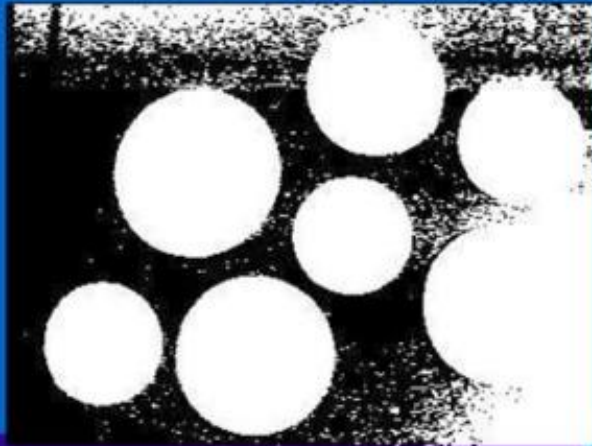
original



good threshold



low threshold

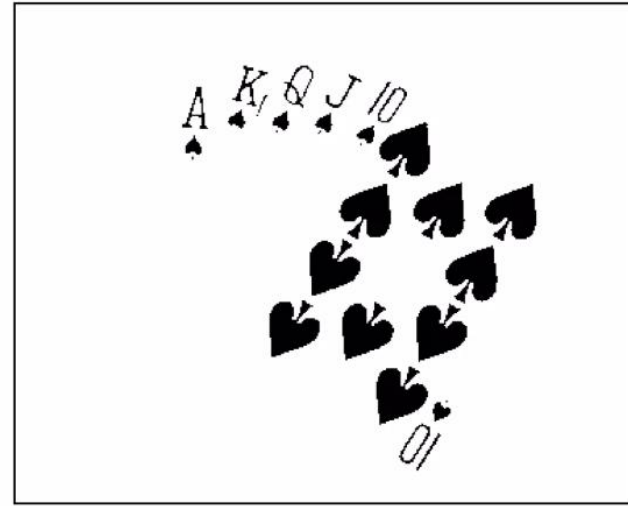


high threshold

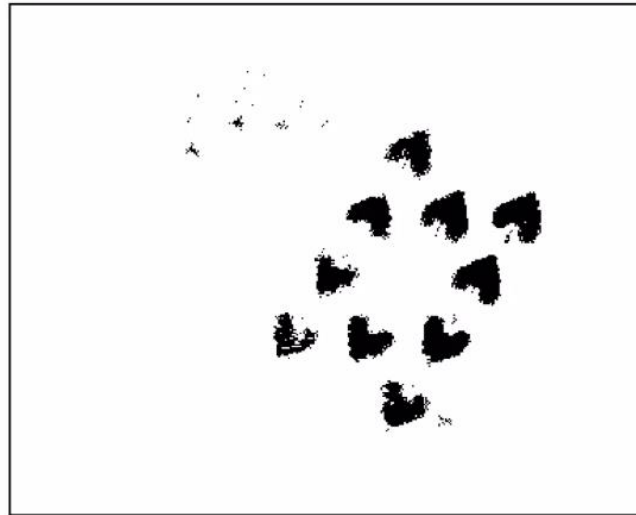




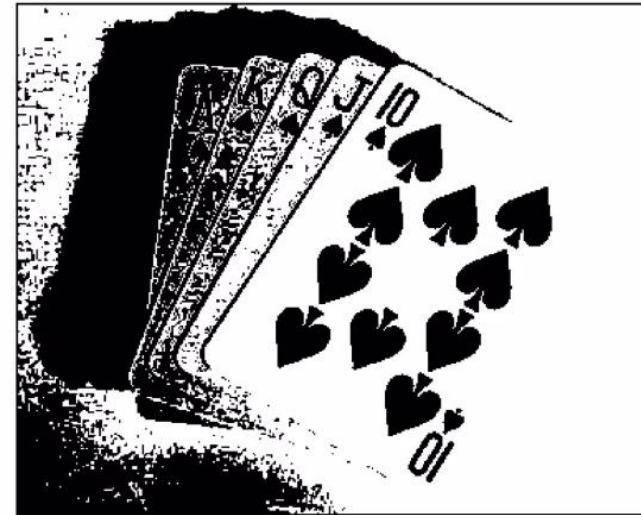
Original Image



Thresholded Image



Threshold Too Low



Threshold Too High