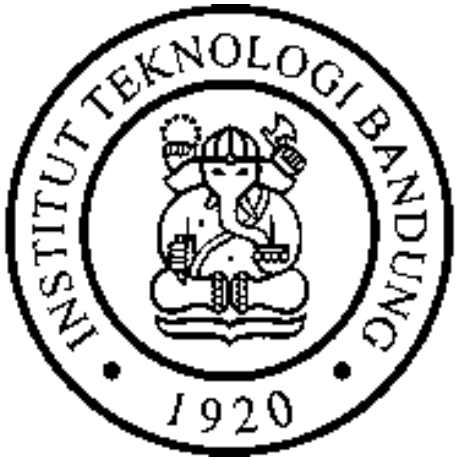
16 - Citra Biner

IF4073 Interpretasi dan Pengolahan Citra

Oleh: Rinaldi Munir

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

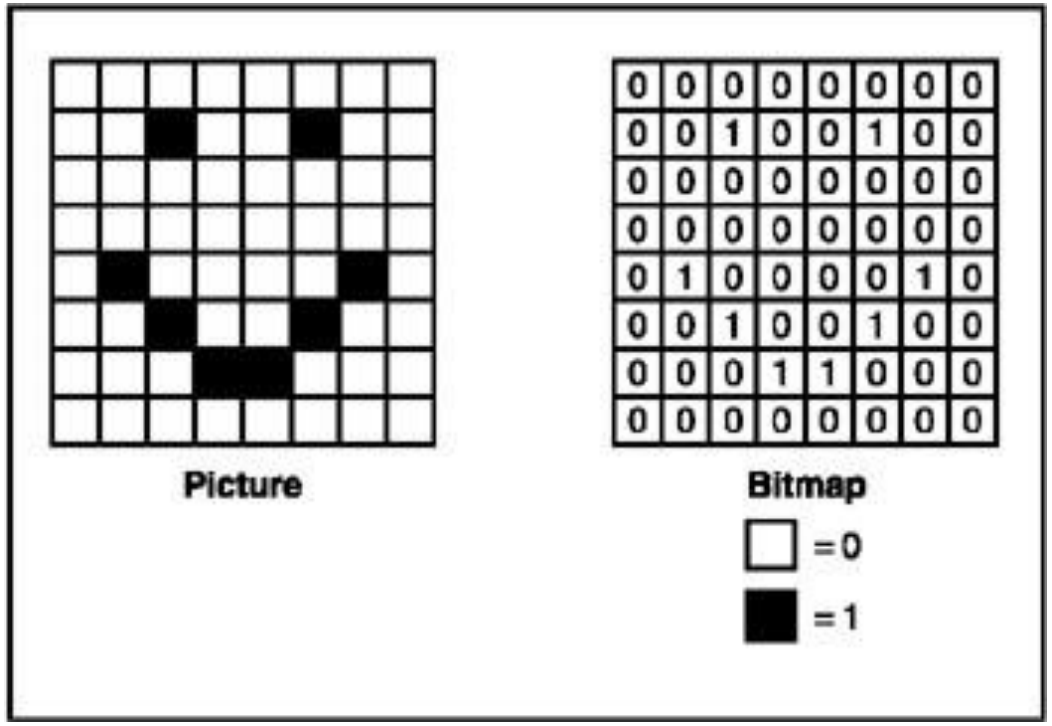
Citra Biner

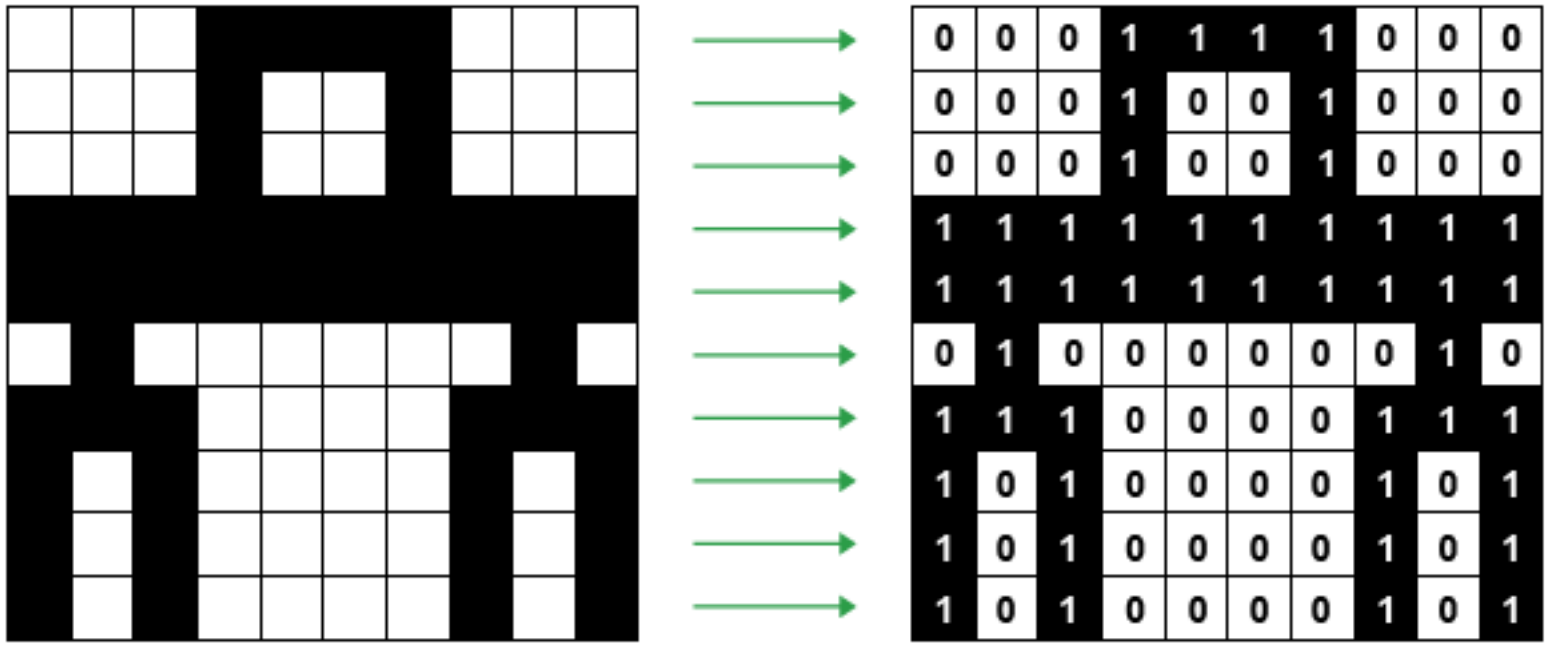



788679 912077

Citra Biner

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

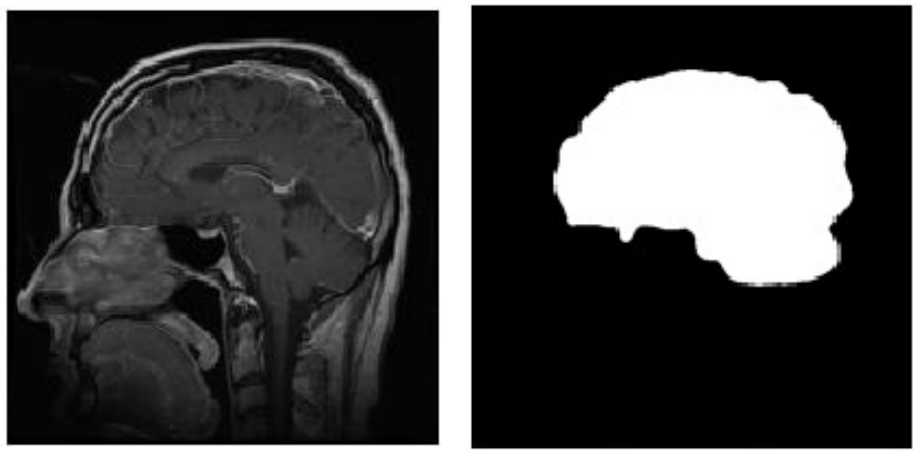


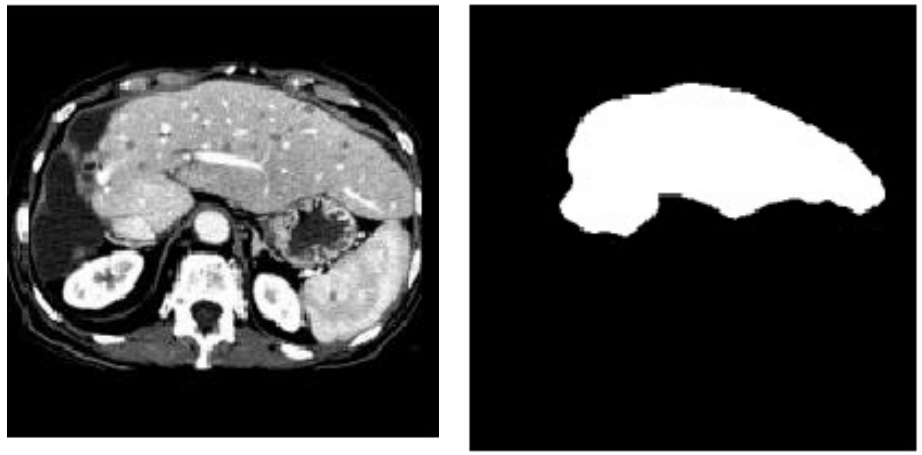

0 0 0 1 1 1 1 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 1 0 0 1 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
0 1 0 0 0 0 0 0 1 0
1 1 1 0 0 0 0 1 1 1
1 0 1 0 0 0 0 1 0 1
1 0 1 0 0 0 0 1 0 1
1 0 1 0 0 0 0 1 0 1

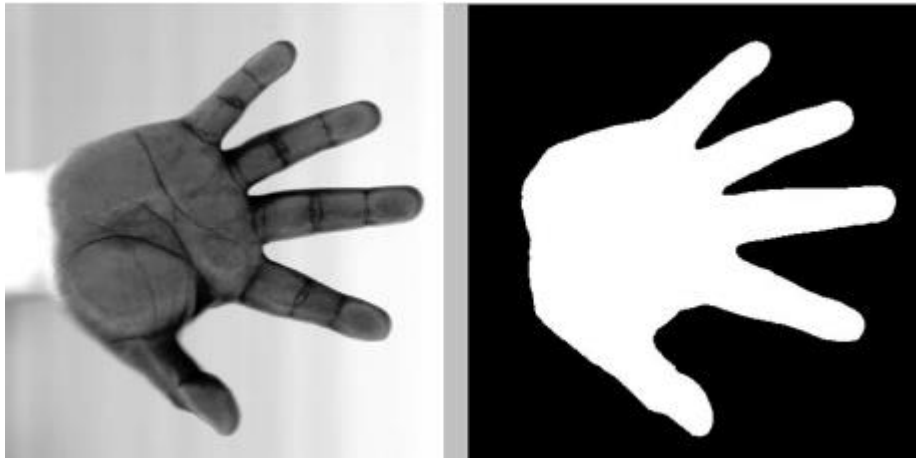
Mengapa perlu citra biner?

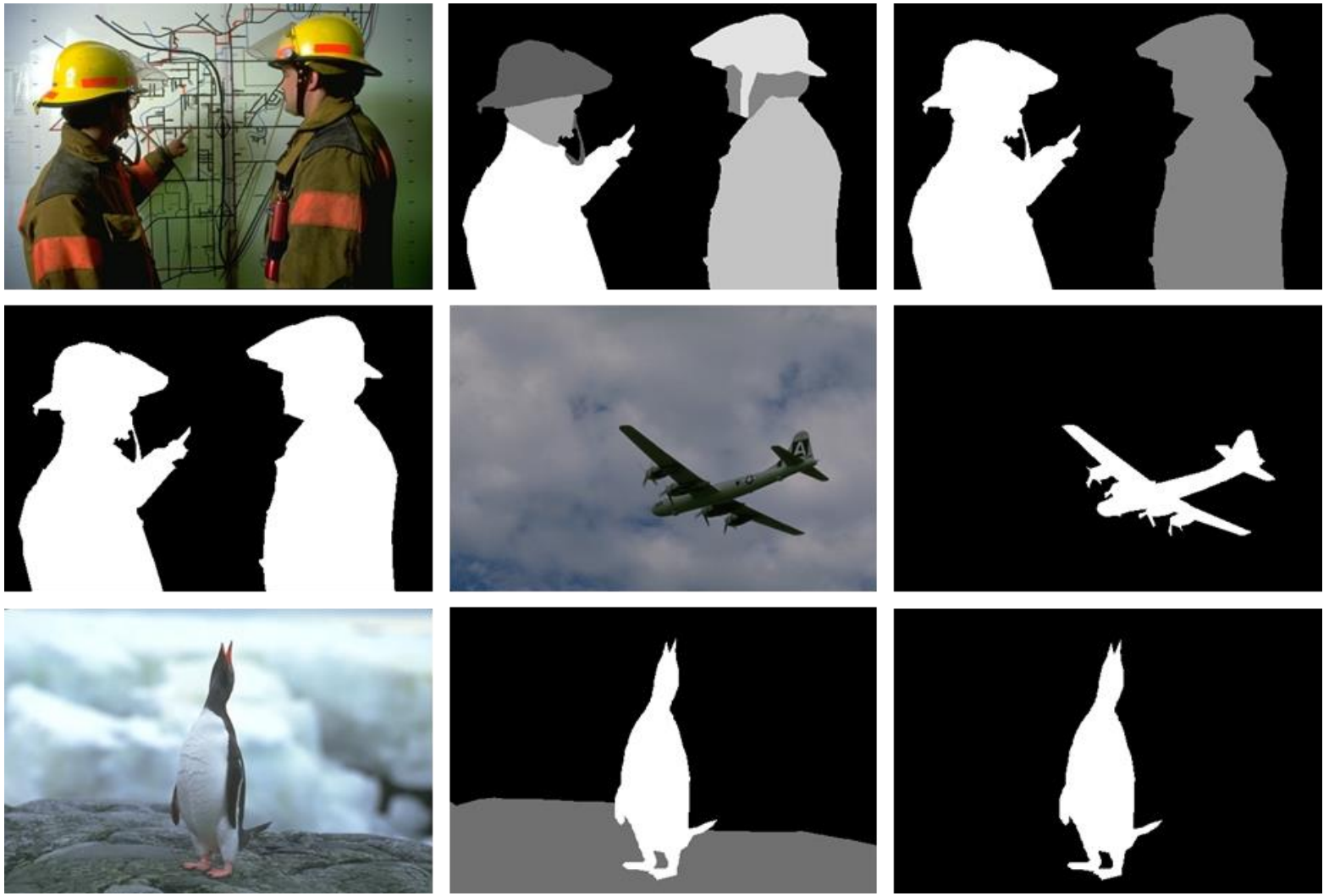
1. Kebutuhan memori untuk setiap pixel sedikit (hanya 1 bit/pixel)
2. Dapat menggunakan operasi logika (AND, OR, NOT) sehingga waktu komputasinya kecil
3. Untuk merepresentasikan citra hasil pendeteksian tepi (*edge images*)

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

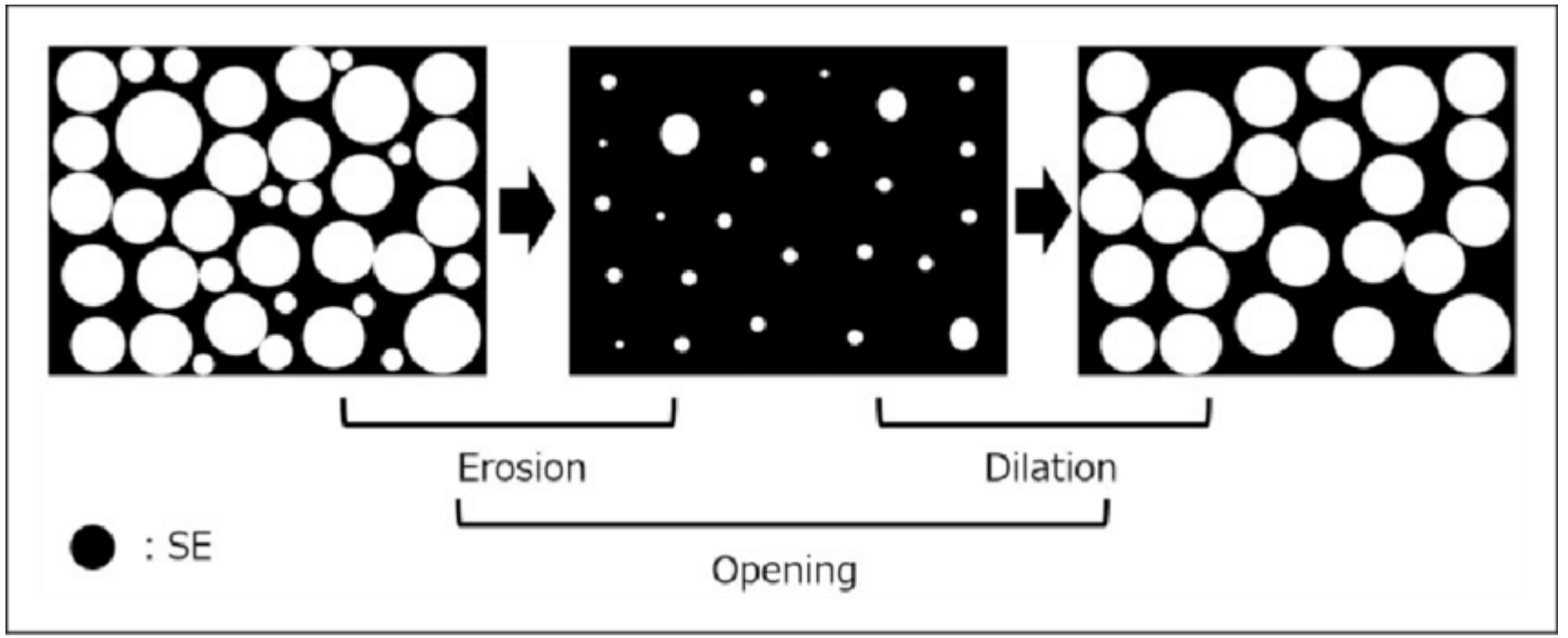








5. Untuk lebih memfokuskan pada analisis bentuk morfologi, yang dalam hal ini intensitas *pixel* tidak terlalu penting dibandingkan bentuknya. Setelah objek dipisahkan dari latar belakangnya, properti geometri dan morfologi objek dapat dihitung dari citra biner



6. Untuk menampilkan citra pada piranti luaran yang hanya mempunyai resolusi intensitas satu bit, misalnya pencetak (*printer*).

Konversi Citra Grayscale ke Citra Biner

- Konversi dari citra hitam-putih ke citra biner dilakukan dengan operasi pengambangan (*thresholding*).

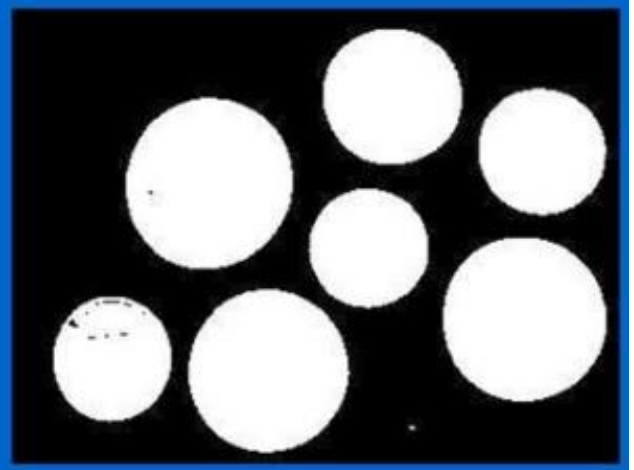
- Operasi pengambangan mengelompokkan nilai derajat keabuan setiap *pixel* ke dalam 2 kelas, hitam dan putih.

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

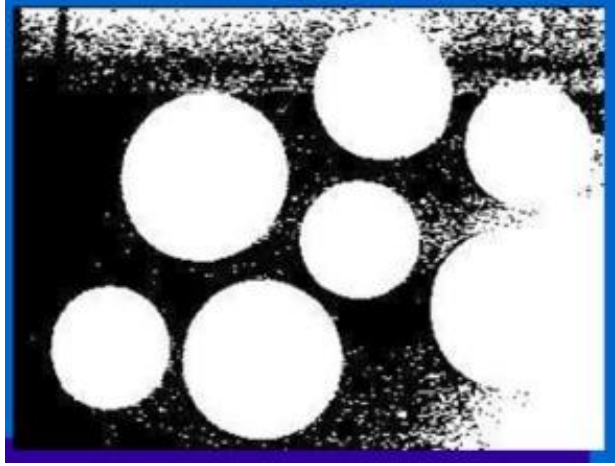
How to choose the threshold?

original

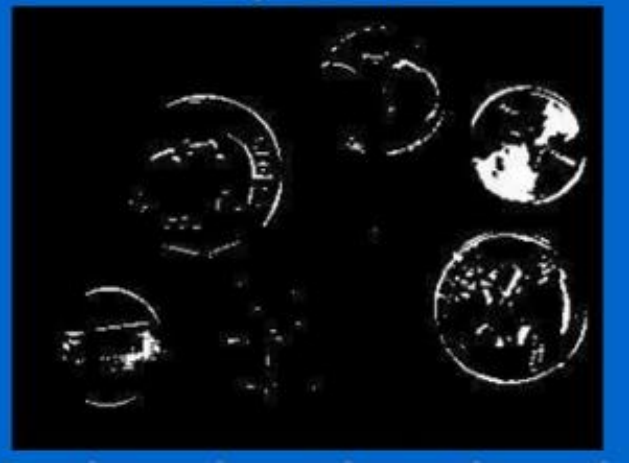
good threshold



low threshold

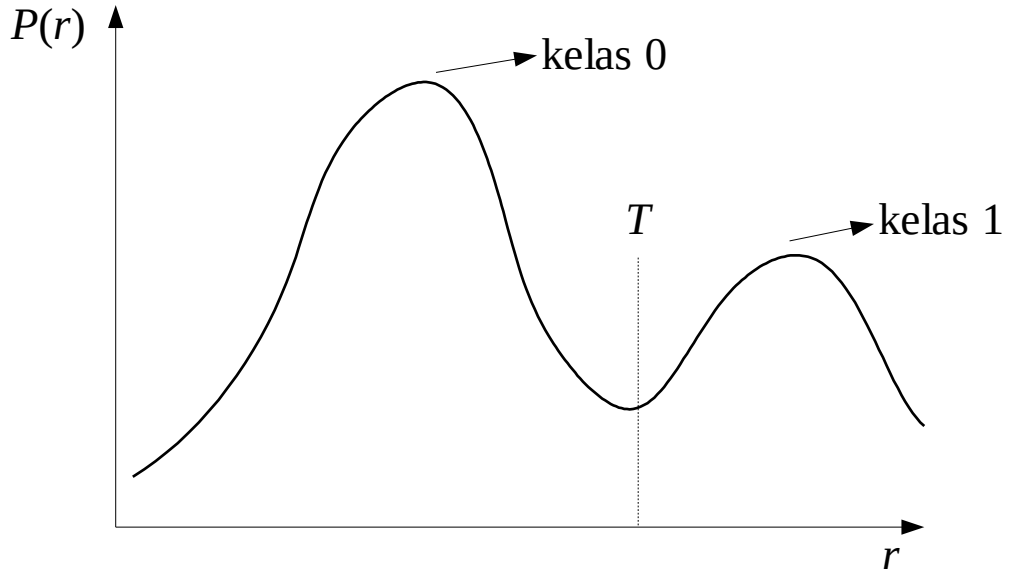


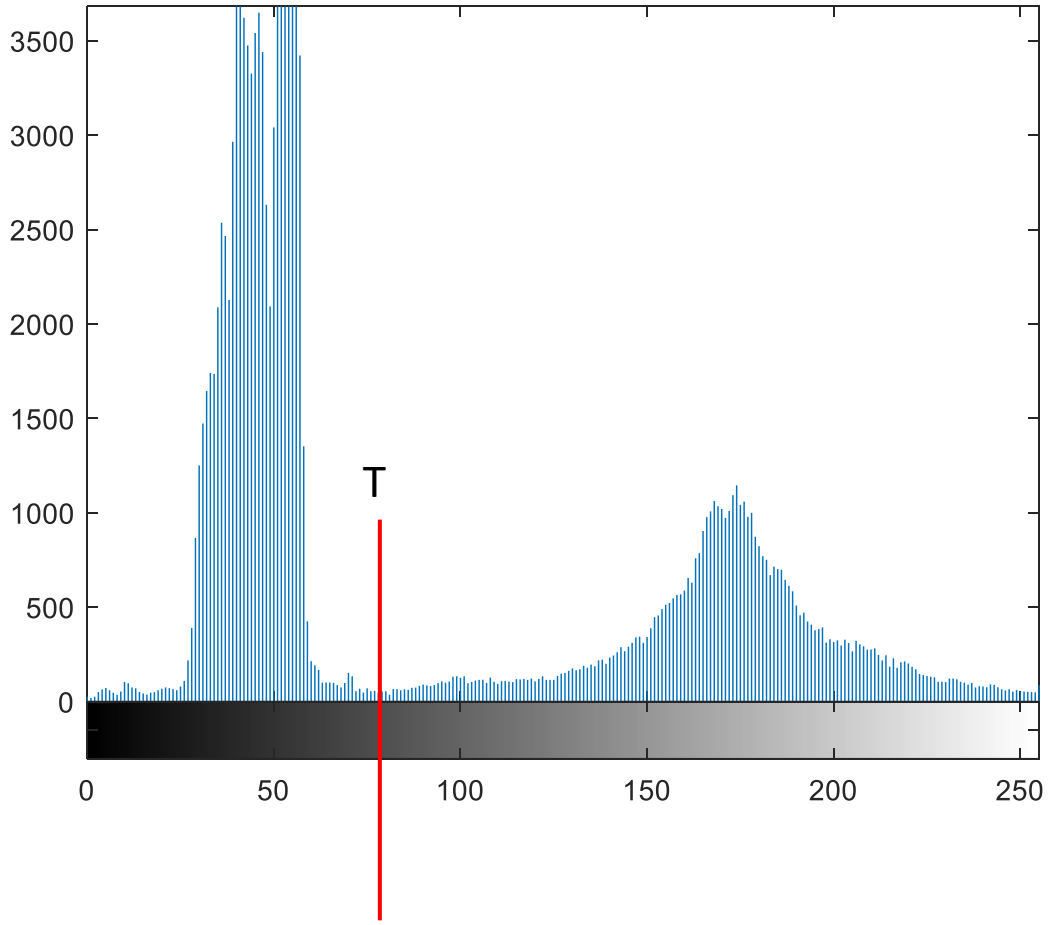
high threshold



Tiga macam operasi pengambangan:

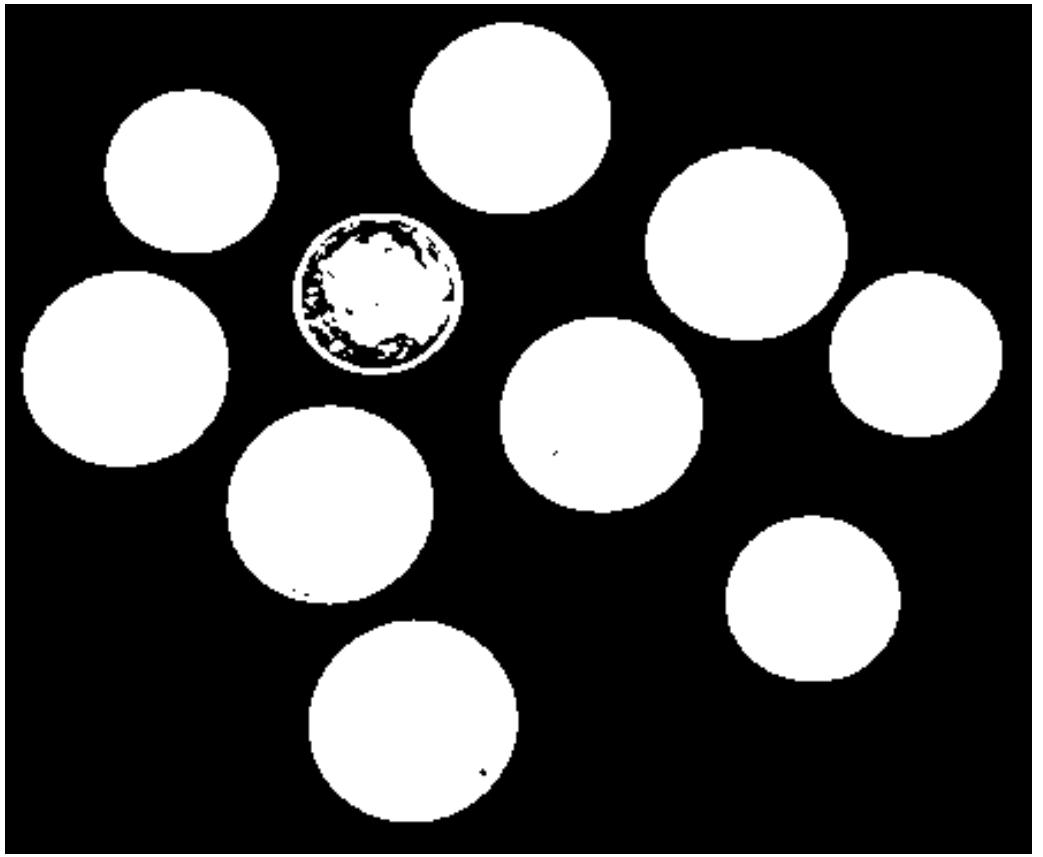
1. Pengambangan secara global
   - Nilai *T* berlaku untuk seluruh bagian di dalam citra
   - Nilai *T* dipilih sedemikian sehingga galat sekecil mungkin
   - Jika citra mengandung satu atau lebih objek dan latar belakang dengan intensitas yang homogen, maka histogramnya *bimodal* (memiliki dua puncak)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




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

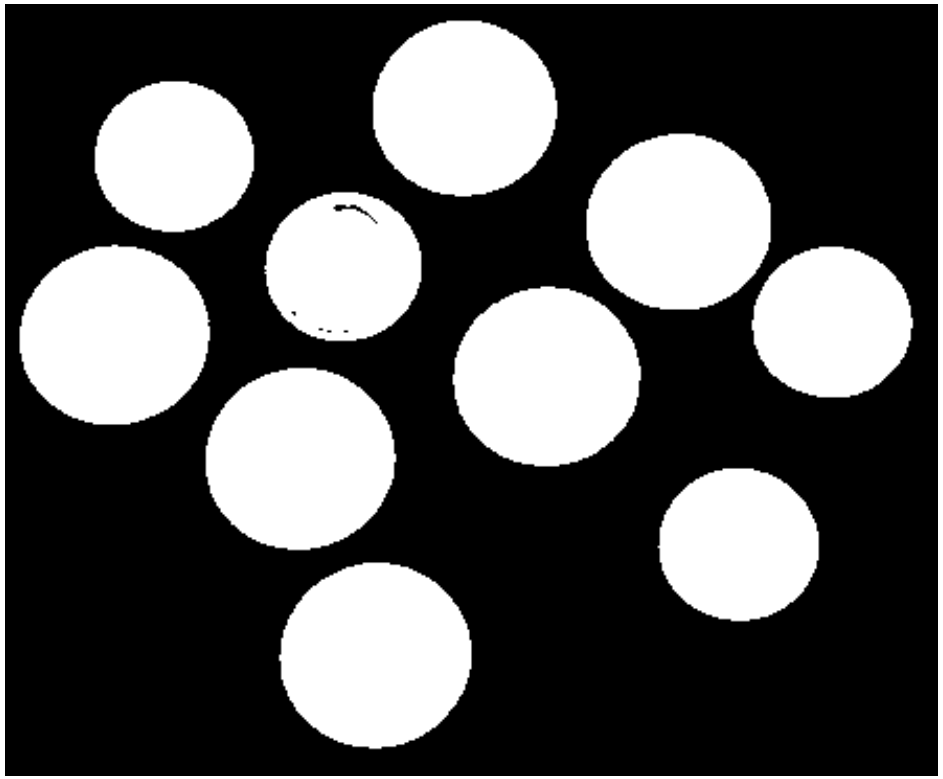
T = 100

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



T = 75

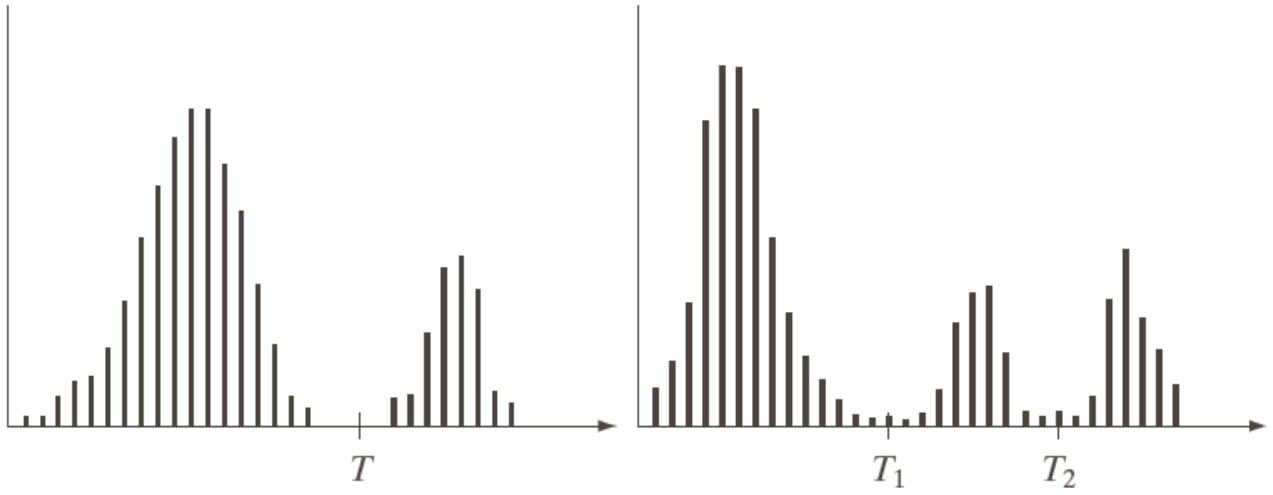
```
>> I = imread('coins.bmp');
>> imhist(I)
>> BW = im2bw(I, 75/255);
>> figure, imshow(BW)
```



- Jika nilai intensitas objek diketahui dalam selang $[T_1, T_2]$, maka kita dapat menggunakan fungsi pengambangan:

$$f_B(i,j) = \begin{cases} 1, & T_1 \le f_g(i,j) \le T_2 \\ 0, & \text{lainnya} \end{cases}$$

- Kasus jika terdapat lebih dari satu lembah dan dua bukit → lebih dari satu *T*:



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

- Menentukan nilai ambang T secara otomatis (tanpa inspeksi secara visual pada histogramnya):

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.

2. Pengambangan secara lokal

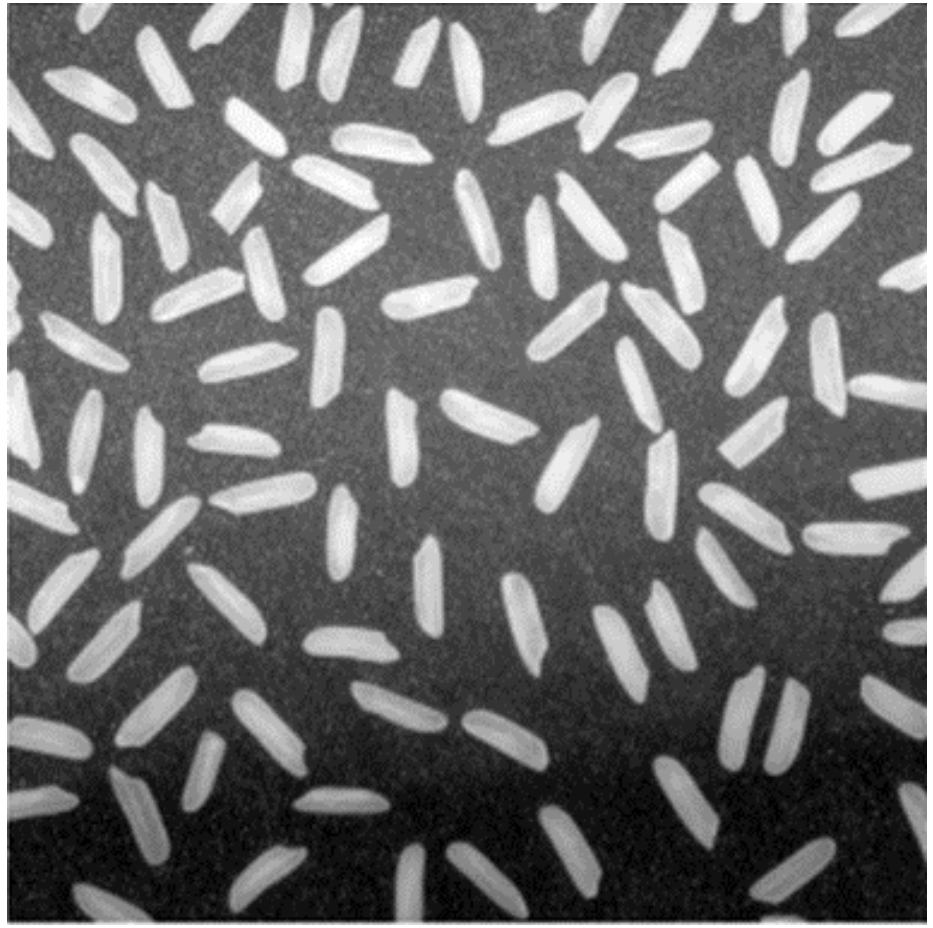
- Pengambangan secara lokal dilakukan terhadap daerah-daerah di dalam citra.
- Dalam hal ini citra dipecah menjadi bagian-bagian kecil, kemudian proses pengambangan dilakukan secara lokal.
- Nilai ambang untuk setiap bagian belum tentu sama dengan bagian lain.
- Sebagai contoh, pengambangan dilakukan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

3. Pengambangan secara adaptif

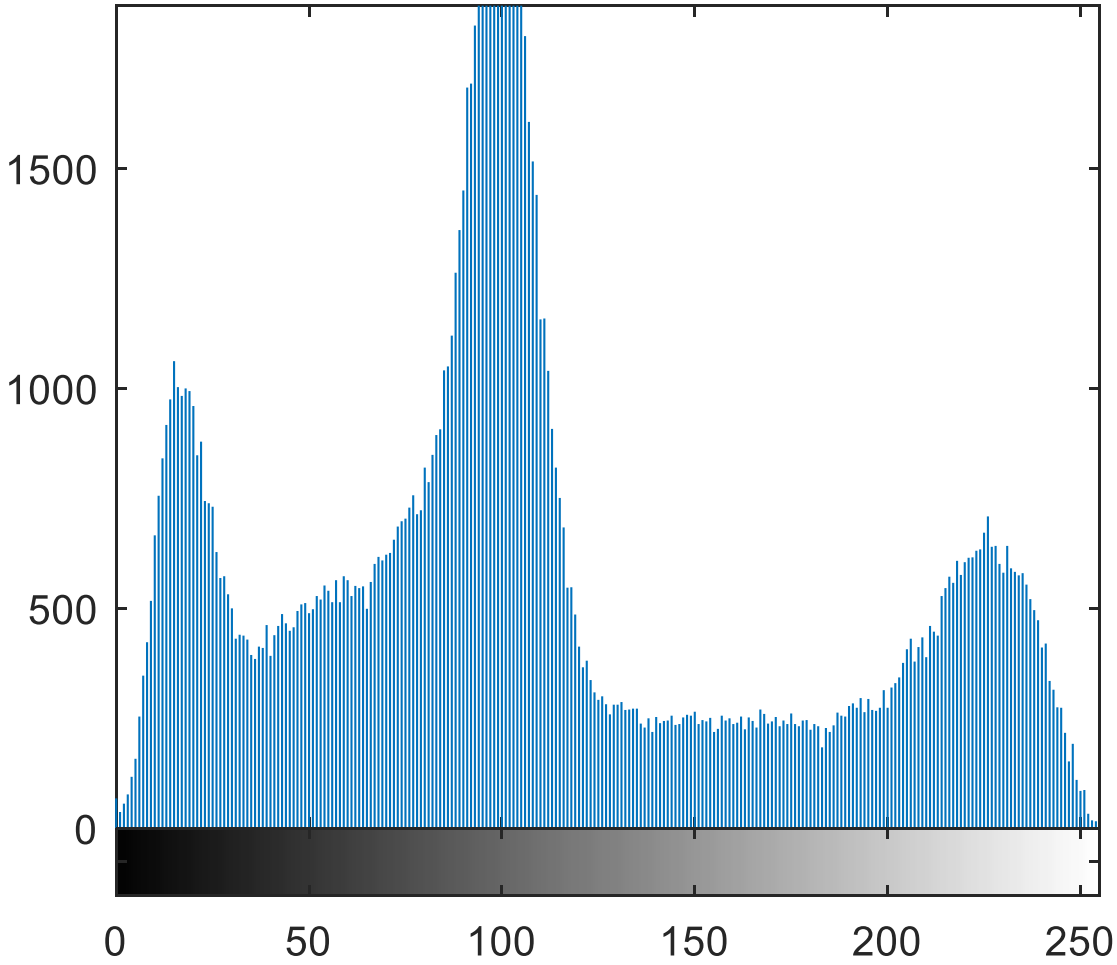
- Pengambangan secara adaptif mengubah nilai ambang secara dinamis pada citra.

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

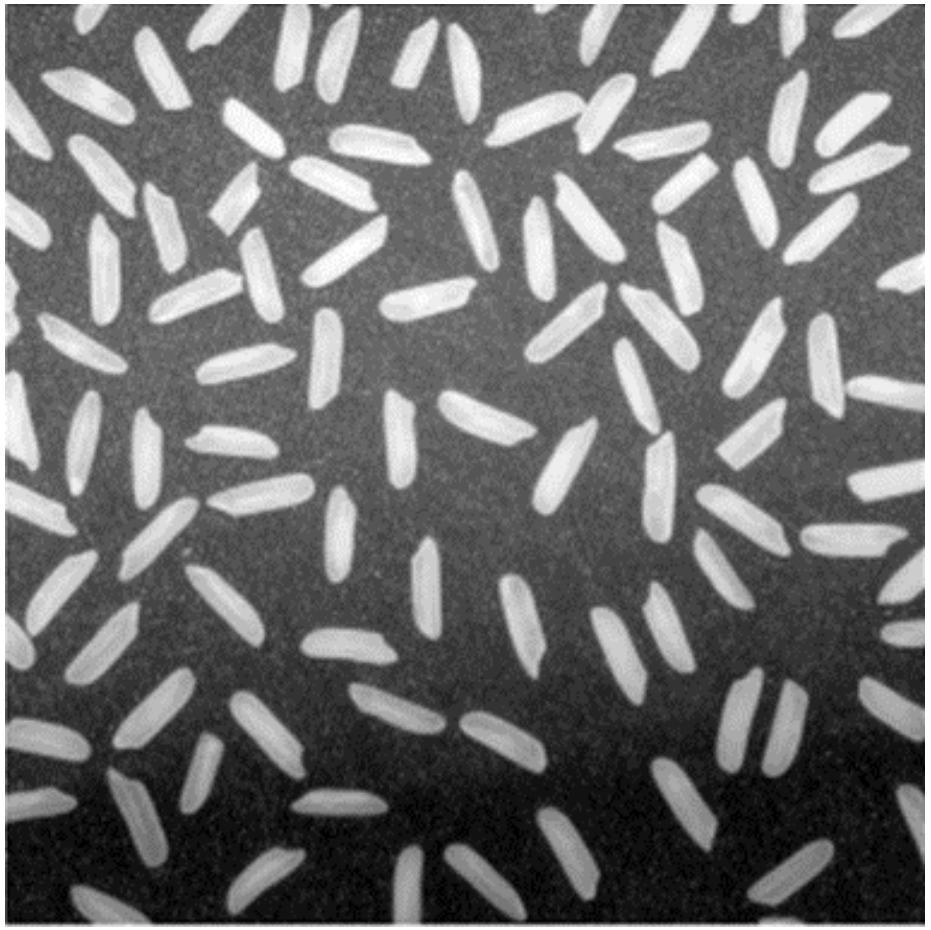
```
I = imread('rice.bmp');
```

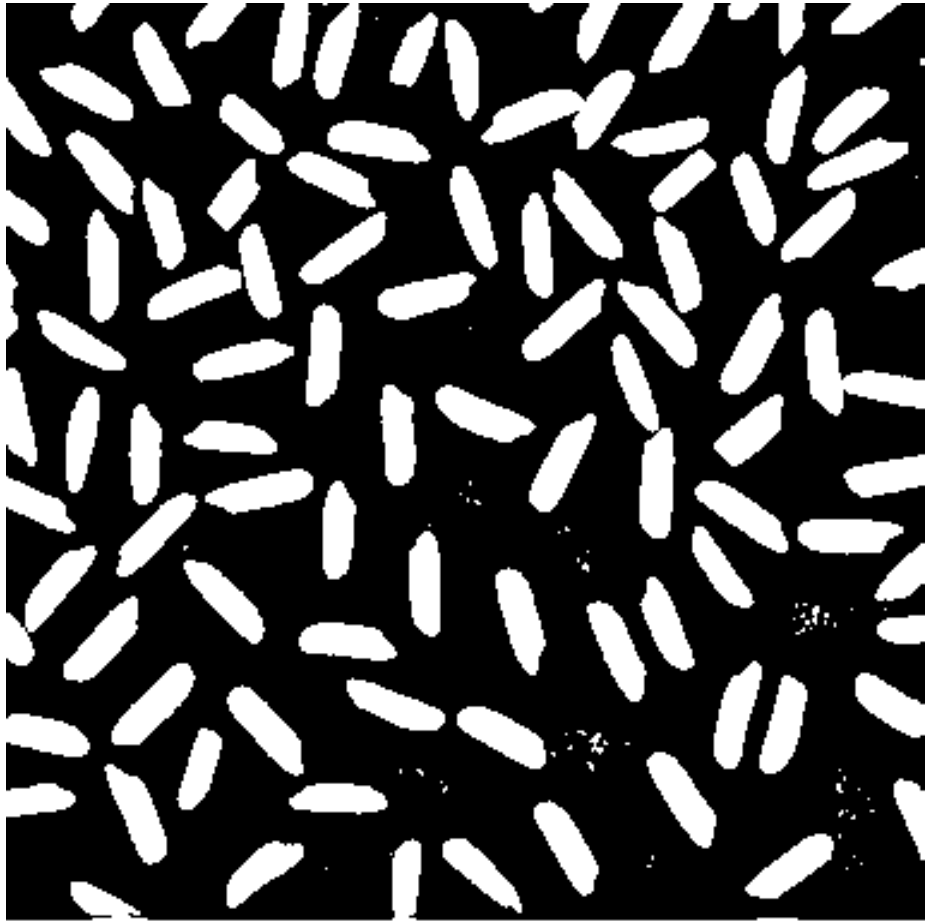


```
imhist(I)
```



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





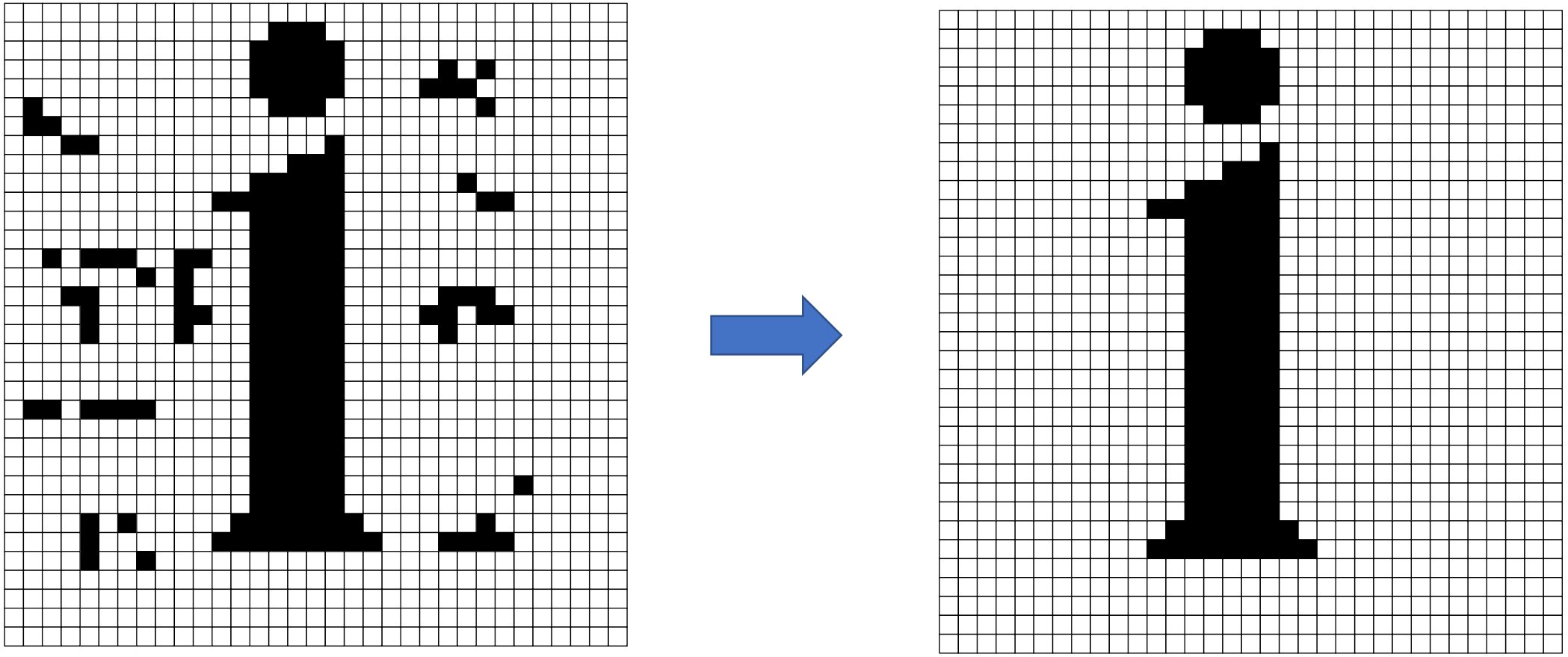
Kegunaan Citra Biner Hasil Pengambangan

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

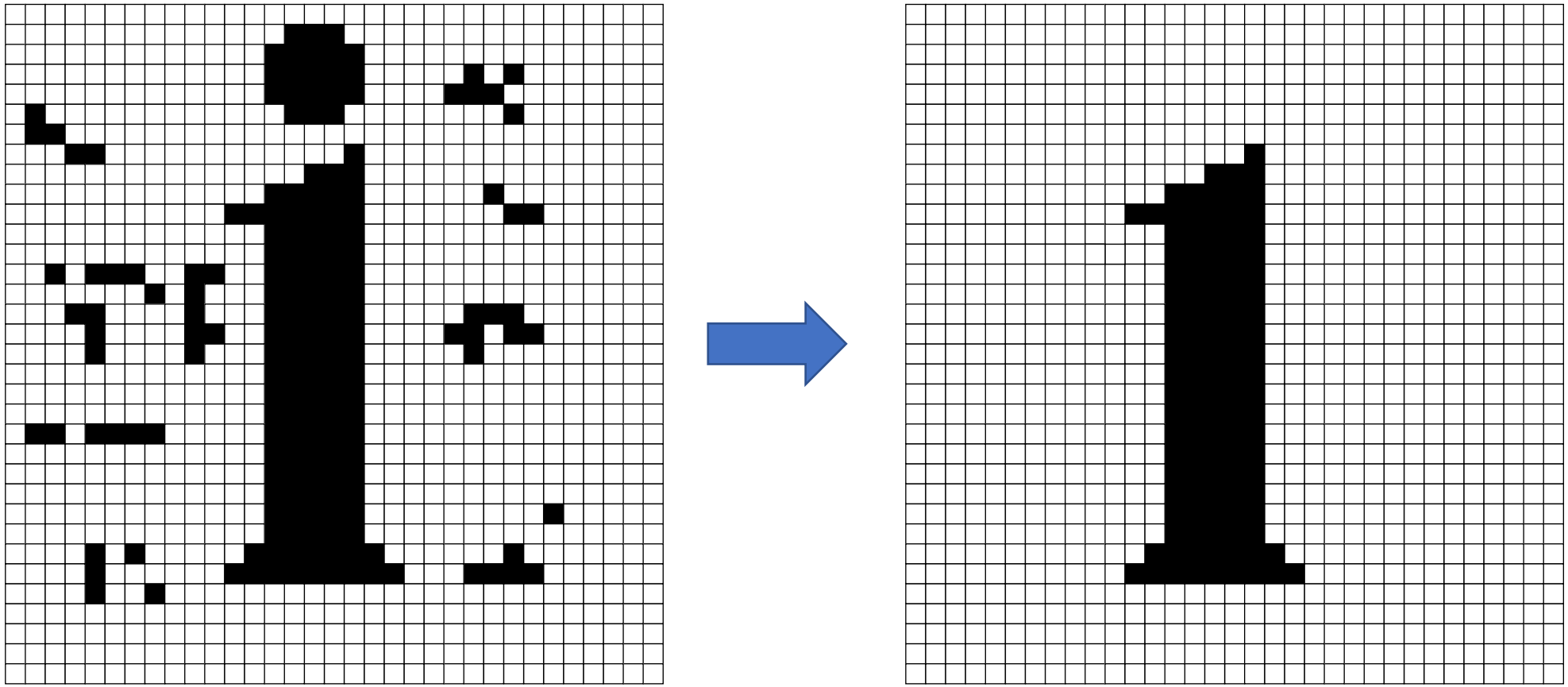
```
I = imread('coins.bmp');
BW = im2bw(I, 75/255);
figure, imshow(BW)
s = size(BW);
m = s(1);
n = s(2);
for i=1:m
    for j=1:n
       if BW(i,j) == 0
          C(i,j) = 255;
       else
          C(i,j) = I(i,j);
       end
    end
end
C = uint8(C);
figure, imshow(C);
```

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”

Run-Length Encoding (RLE)

- RLE merupakan metode pengkodean citra biner untuk menghasilkan representasi citra yang mampat.

- Dua pendekatan yang digunakan dalam penerapan *RLE* pada citra biner:
  1. Posisi awal kelompok nilai 1 dan panjangnya (*length of runs*)
  2. Panjang *run*, dimulai dengan panjang *run* 1.

- Contoh: Misalkan citra biner adalah sebagai berikut:

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

Hasil pengkodean dengan metode RLE:

(1) pendekatan pertama:

(1, 3) (7, 2) (12, 4) (17, 2) (20, 3)
(5, 13) (19, 4)
(1, 3) (17, 6)

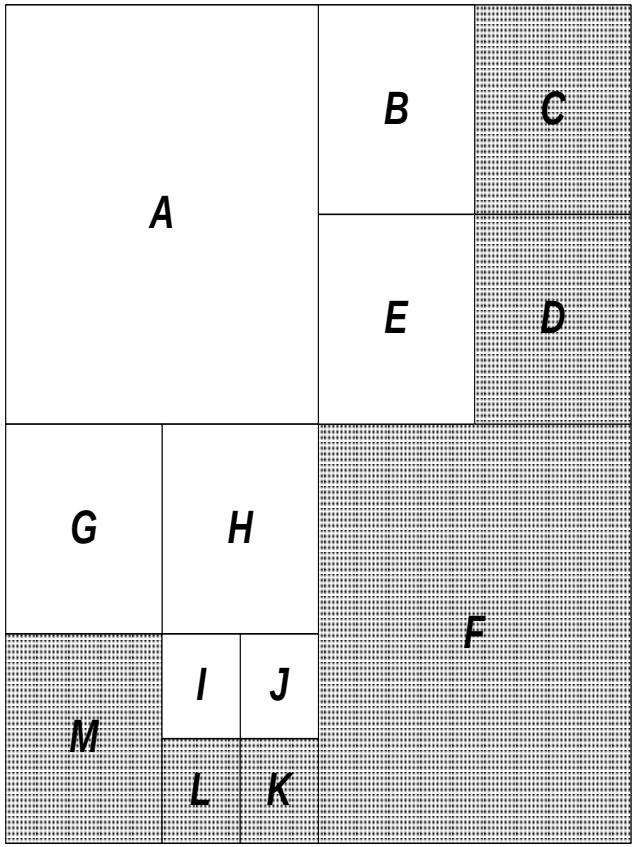
(2) pendekatan kedua

3, 3, 2, 3, 4, 1, 2, 1, 3
0, 4, 13, 1, 4
3, 13, 6

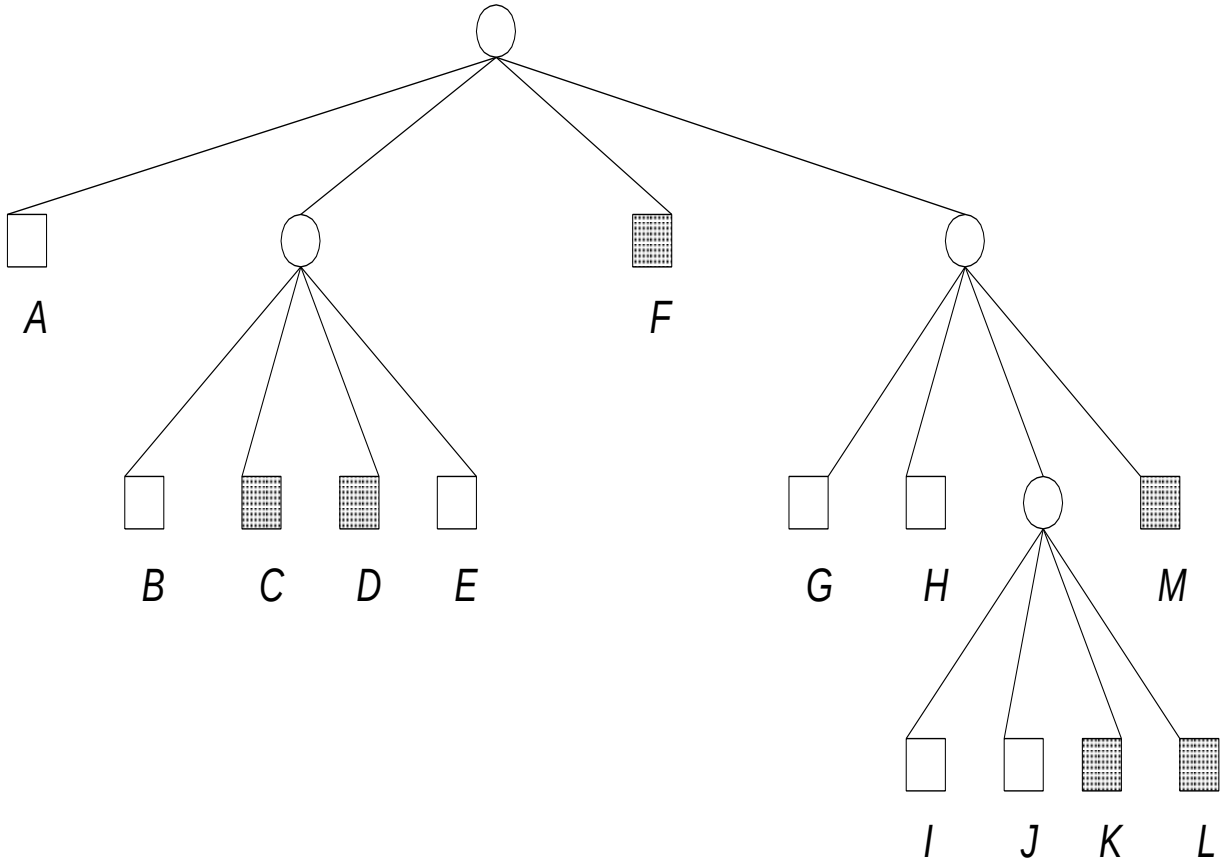
Pohon empatan (*quadtree*)

- Pohon empatan merupakan representasi wilayah (*region*) di dalam citra biner.
- Setiap simpul di dalam pohon-empatan merupakan salah satu dari tiga ketagori: putih, hitam, dan abu-abu.
- Algoritma pohon-empatan:

1. Bagi citra secara rekursif menjadi empat buah upa-wilayah yang berukuran sama.
2. Untuk setiap upa-wilayah, bila *pixel-pixel* di dalam wilayah tersebut semuanya hitam atau semuanya putih, maka proses pembagian dihentikan.
3. Sebaliknya, bila *pixel-pixel* di dalam upa-wilayah mengandung baik *pixel* hitam maupun *pixel* putih (kategori abu-abu), maka upa-wilayah tersebut dibagi lagi mejadi empat bagian.
4. Demikian seterusnya sampai diperoleh upa-wilayah yang semua *pixel*-nya hitam atau semua *pixel*-nya putih



Citra biner



Kode: gwgwbbwbgwwgwwgwwbbb

Di-*decode* sebagai: *g*(*wg*(*wbbw*)*bg*(*wwg*(*wwbb*)*b*))

Keterangan: *b* = *black*, *w* = *white*, *g* = *gray*

Penelusuran Batas Objek (*Boundary tracing*)

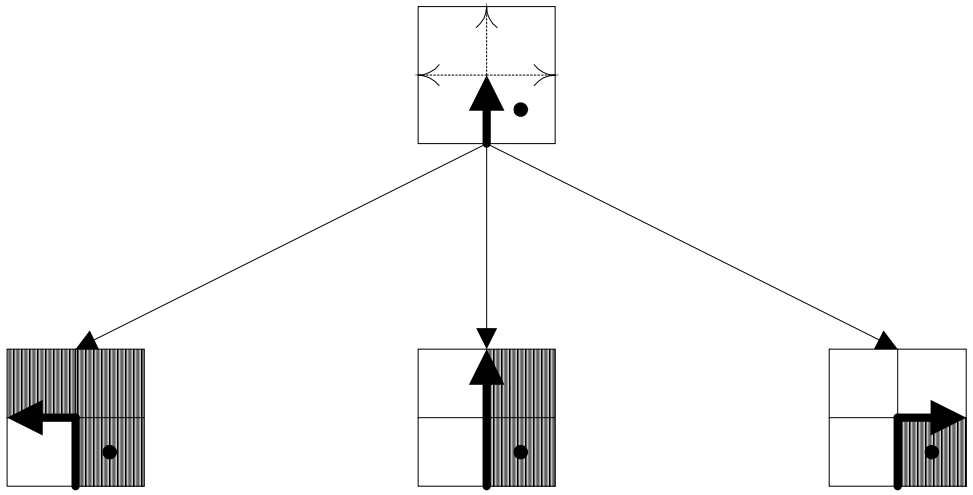
- Setelah citra *grayscale* disegmentasi menjadi citra biner, maka perbedaan antara objek dengan latar belakang terlihat dengan jelas.

- *Pixel* objek berwarna putih sedangkan *pixel* latar belakang berwarna hitam.

- Rangkaian *pixel* yang menjadi batas (*boundary*) antara objek dengan latar belakang dapat diketahui secara keseluruhan (algoritma *boundary following*).

Algoritma *boundary following* sebagai berikut:

- Pertemuan antara *pixel* putih dan hitam dimodelkan sebagai segmen garis.
- Penelusuran batas wilayah dianggap sebagai pembuatan rangkaian keputusan untuk bergerak lurus, belok kiri, atau belok kanan.



Pixel yang bertanda • menyatakan *pixel* yang sedang ditelaah. Penelusur harus menentukan arah *pixel* batas berikutnya bergantung pada *pixel-pixel* sekitarnya:

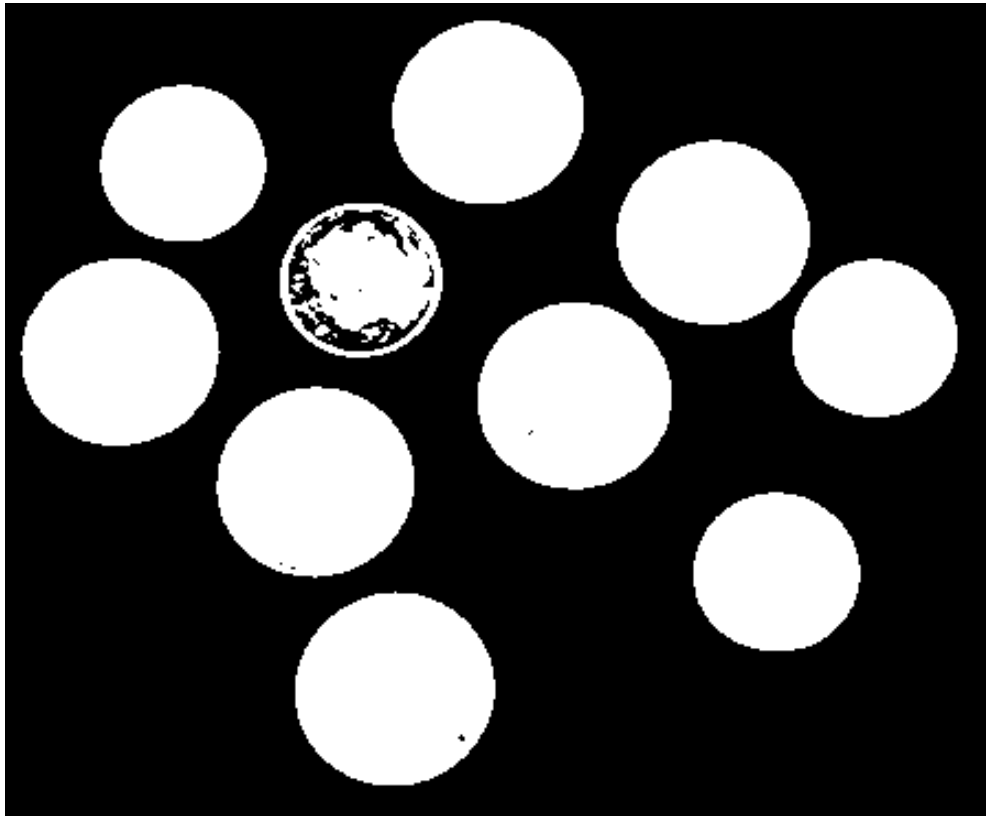
```
if DepanTidakSama(arah,x,y) then
   Belok kanan (arah,x,y)
else
   if SilangSama(arah,x,y) then
      Belok kiri (arah,x,y)
   else
      Lurus (arah,x,y)
   endif
endif
```

Step 1: Baca citra, tampilkan

```
I = imread('coins.bmp');
imshow(I)
```

Step 2: Ubah ke citra biner sekaligus segmentasi

```
BW = im2bw(I, 100/255);
imshow(BW)
```



Step 3: Tentukan koordinat awal untuk melakukan deteksi batas

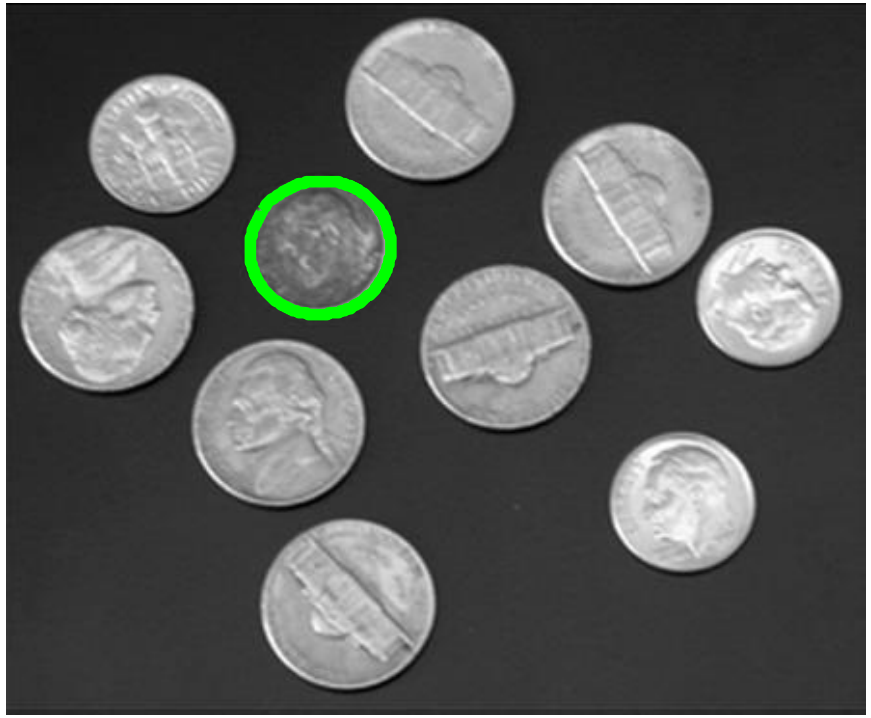
```
dim = size(BW)
col = round(dim(2)/2)-90;
row = min(find(BW(:,col)))

dim =

   369   446


row =

   108
```

Step 5: Panggil fungsi bwtraceboundary untuk menelusuri batas objek dari titik awal yang dispesifikasikan

```
boundary = bwtraceboundary(BW,[row, col],'N');
```

Step 6: Tampilkan citra awal dan gunakan koordinat yang dihasilkan oleh *bwtraceboundary* untuk mem-plot batas objek di dalam citra

```
boundary = bwtraceboundary(BW,[row, col],'N');
imshow(I)
hold on;
plot(boundary(:,2),
boundary(:,1),'g','LineWidth',3);
```



Untuk menelusuri batas-batas semua koin lain, gunakan fungsi *bwtraceboundary* secara berulang-ulang

- Di dalam Matlab, untuk menemukan semua batas objek digunakan fungsi *bwboundaries.*

- Pada contoh di atas, beberapa koin mengandung area hitam yang diinterpretasikan oleh fungsi *bwboundaries* sebagai objek terpisah.

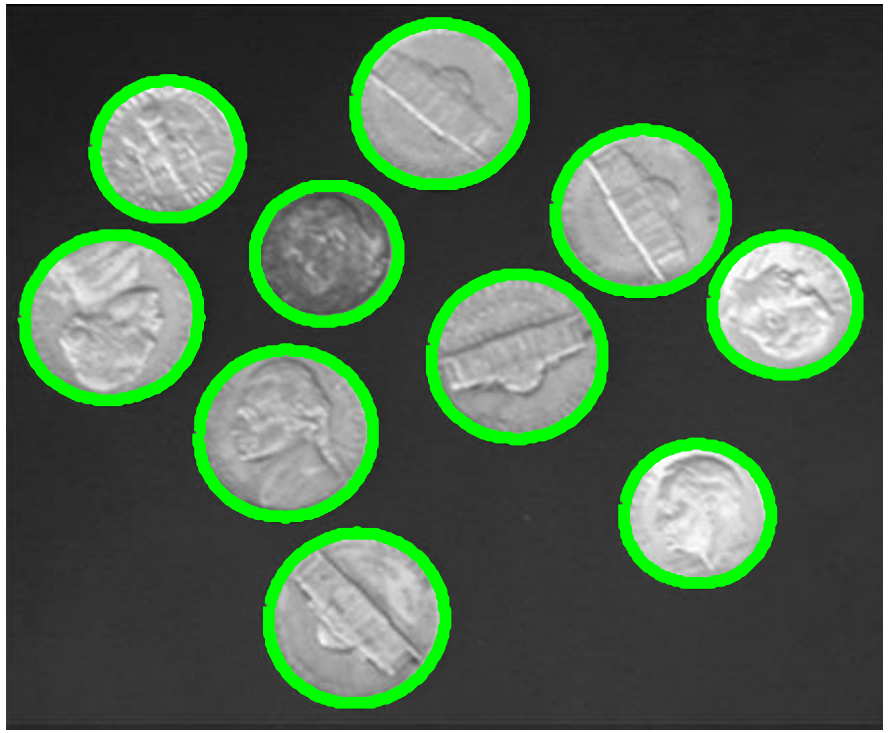
- Untuk memastikan bahwa *bwboundaries* hanya menelusuri koin, gunakan fungsi *imfill* untuk mengisi area di dalam setiap koin.

```
BW_filled = imfill(BW,'holes');
boundaries = bwboundaries(BW_filled);
```

- *bwboundaries* mengembalikan array sel, setiap sel mengandung koordinat baris/kolom objek di dalam citra.

Plot batas objek semua koin pada citra semula dengan menggunakan koordinat yang dihasilkan oleh *bwboundaries* .

```
for k=1:10
    b = boundaries{k};
    plot(b(:,2),b(:,1),'g','LineWidth',3);
end
```

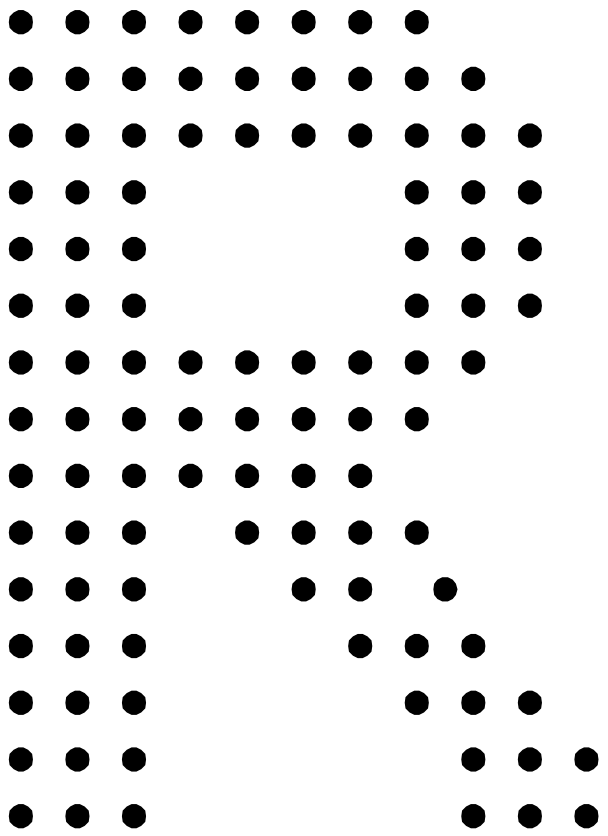


Penipisan (*thinning*)

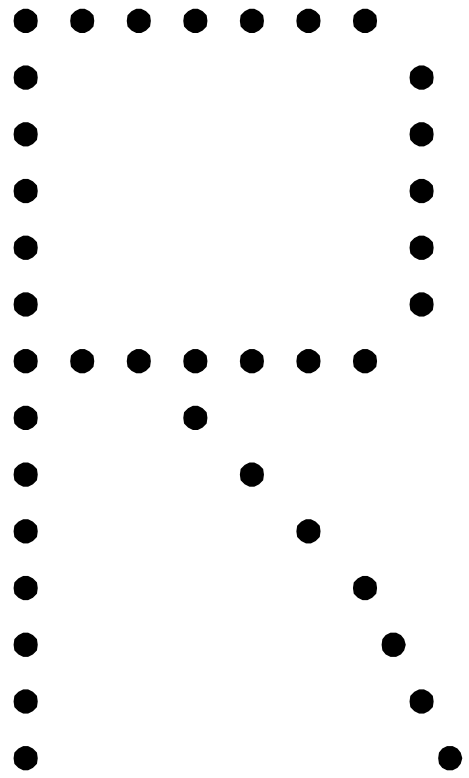
- Penipisan (*thinning*) adalah operasi pemrosesan citra biner yang dalam hal ini objek (*region*) direduksi menjadi rangka (skeleton) yang menghampiri garis sumbu objek.

- Tujuan penipisan adalah mengurangi bagian yang tidak perlu (*redundant*) sehingga hanya dihasilkan informasi yang esensial saja.

- Pola hasil penipisan harus tetap mempunyai bentuk yang menyerupai pola asalnya.



(a) Huruf “R”



(b) Hasil penipisan huruf “R

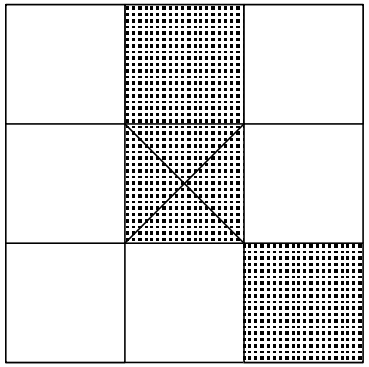
- Selanjutnya, pola hasil penipisan digunakan di dalam proses pencocokan pola (*pattern recognition*).

- Penipisan pola merupakan proses yang iteratif yang menghilangkan *pixel-pixel* hitam (mengubahnya menjadi *pixel* putih) pada tepi-tepi pola.

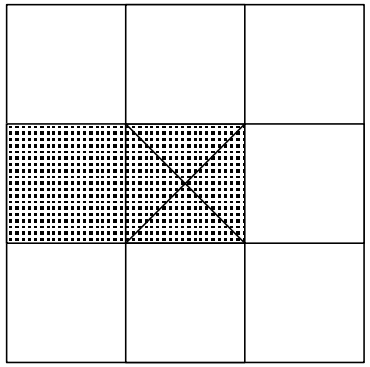
- Jadi, algoritma penipisan mengelupas *pixel-pixel* pinggir objek, yaitu *pixel-pixel* yang terdapat pada peralihan $0 \rightarrow 1$

Algoritma penipisan pola harus memenuhi persyaratan sebagai berikut:

1. Mempertahankan keterhubungan *pixel-pixel* objek pada setiap lelaran. Dengan kata lain, tidak menyebabkan bentuk objek menjadi terputus
2. Tidak memperpendek ujung lengan dari bentuk yang ditipiskan



(a)



(b)

p_8	p_1	p_2
p_7	p_0	p_3
p_6	p_5	p_4

(c)

(a) Penghapusan pixel pinggir menyebabkan ketidakterhubungan,
(b) penghapusan pixel pinggir memperpendek lengan objek,
(c) notasi pixel yang digunakan untuk memeriksa keterhubungan.

- Algoritma penipisan yang umum adalah memeriksa *pixel-pixel* di dalam jendela yang berukuran 3×3 *pixel* dan mengelupas satu *pixel* pada pinggiran (batas) objek pada setiap lelaran, sampai objek berkurang menjadi garis tipis.

Algoritmanya adalah sebagai berikut:

1. Mula-mula diperiksa jumlah *pixel* objek (yang bernilai 1), *N*, di dalam jendela 3×3 *pixel*.
2. Jika *N* kurang atau sama dengan 2, tidak ada aksi yang dilakukan karena di dalam jendela terdapat ujung lengan objek.
3. Jika *N* lebih besar dari 7, tidak ada aksi yang dilakukan karena dapat menyebabkan pengikisan (*erosion*) objek.

4. Jika *N* lebih besar dari 2, periksa apakah penghilangan *pixel* tengah menyebabkan objek tidak terhubung.

   Ini dilakukan dengan membentuk barisan $p_1p_2p_3...p_8p_1$.

   Jika jumlah peralihan $0 \rightarrow 1$ di dalam barisan tersebut sama dengan 1, berarti hanya terdapat satu komponen terhubung di dalam jendela 3×3.

   Pada kasus ini, dibolehkan menghapus *pixel* tengah yang bernilai 1 karena penghapusan tersebut tidak mempengaruhi keterhubungan.

```
void penipisan(citra f, int N1, int M1, int N2, int M2)
/* Prosedur yang mengimplementasikan penipisan pola
   Masukan :
      f      : citra biner
      N1, M1 : koordinat awal (sudut kiri atas)
      N2, M2 : koordinat akhir (sudut kanan bawah)
   Luaran: citra bner
*/
{
  int k, l, i, j, count=0, y[9], trans=0, m, OK=1

  do
   {
    OK=1;
    for(k=N1+1;k<N2-1;k++)
       for(l=M1+1;l<M2-1;l++)
         if (f[k][l]==1)
```

```
                {/* hitung jumlah 1 di dalam jendela 3 x 3 */
                 count=0;
                for(i=-1;i<=1;i++)
                  for(j=-1;j<=1;j++)
                    if(f[k+i][j+l]==1) count++;
                if((count>2)&&(count<8))
                { /* hitung jumlah peralihan 0->1 */
                  y[0]=f[k-1][l-11]; y[1]=f[k-1][l];y[2]=f[k-1][l+1];
                  y[3]=f[k][l+1];    y[4]=f[k+1]l+1];y[5]=f[k+1][l];
                  y[6]=f[k+1][l-1];  y[7]=f[k][l-1];y[8]=f[k-1][l-1];
                  trans=0;
                  for(m=0;m<=7;m++)
                    if (y[m]==0 && y[m+1]==1) trans++;
                    /* jika jumlah peralihan sama dengan 1, hapus pixel yang
                       sedang diacu (current) */
                    if (trans==1) {f[k][l]=0; OK=0;}
                }
            }
    } while (OK=0);
}
```