Bab 11

Citra Biner

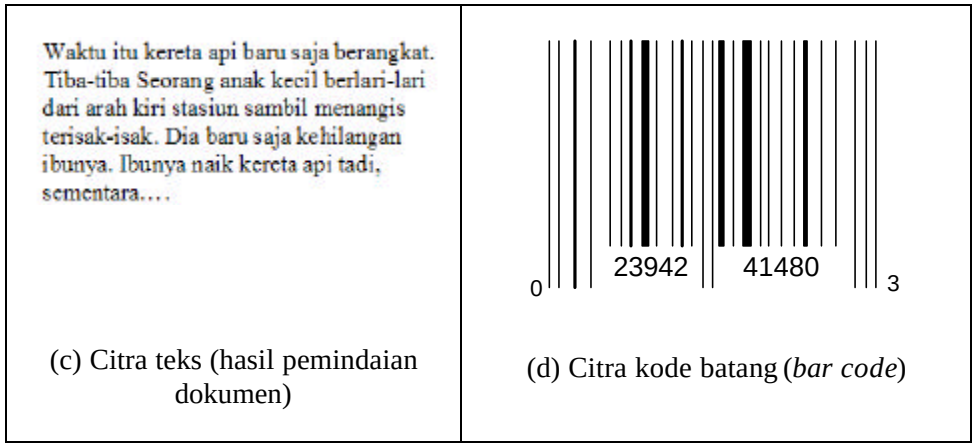
Citra biner (*binary image*) adalah citra yang hanya mempunyai dua nilai derajat keabuan: hitam dan putih. Meskipun saat ini citra berwarna lebih disukai karena memberi kesan yang lebih kaya daripada citra biner, namun tidak membuat citra biner mati. Pada beberapa aplikasi citra biner masih tetap dibutuhkan, misalnya citra logo instansi (yang hanya terdiri atas warna hitam dan putih), citra kode batang (*bar code*) yang tertera pada label barang, citra hasil pemindaian dokumen teks, dan sebagainya. Bab 10 ini akan memaparkan beberapa konsep dan teknik pengolahan citra biner.

11.1 Pendahuluan

Seperti yang sudah disebutkan di awal Bab, citra biner hanya mempunyai dua nilai derajat keabuan: hitam dan putih. *Pixel-pixel* objek bernilai 1 dan *pixel*-pixel latar belakang bernilai 0. Pada waktu menampilkan gambar, 0 adalah putih dan 1 adalah hitam. Jadi, pada citra biner, latar belakang berwarna putih sedangkan objek berwarna hitam. Gambar 11.1 memperlihatkan beberapa contoh citra biner, sedangkan Gambar 11.2 adalah contoh pengkodean citra biner.

(a) Citra logo

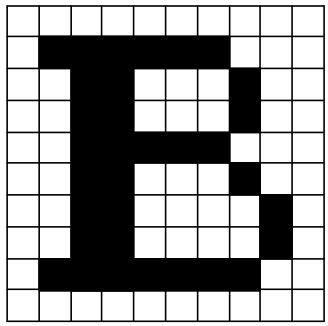
(b) Citra lukisan mobil



(c) Citra teks (hasil pemindaian dokumen)

(d) Citra kode batang (*bar code*)

Gambar 11.1 *Beberapa contoh citra biner*



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

Gambar 11.2 *Huruf "B" dan representasi biner dari derajat keabuannya.*

Meskipun komputer saat ini dapat memproses citra hitam-putih (*greyscale*) maupun citra berwarna, namun citra biner masih tetap dipertahankan keberadaannya. Alasan penggunaan citra biner adalah karena ia memiliki sejumlah keuntungan sebagai berikut:

1. Kebutuhan memori kecil karena nilai derajat keabuan hanya membutuhkan representasi 1 bit. Kebutuhan memori untuk citra biner masih dapat berkurang secara berarti dengan metode pemampatan *run-length encoding* (*RLE*). Metode *RLE* akan dijelaskan kemudian.
2. Waktu pemrosesan lebih cepat dibandingkan dengan citra hitam-putih karena banyak operasi pada citra biner yang dilakukan sebagai operasi logika (*AND*, *OR*, *NOT*, dll) ketimbang operasi aritmetika bilangan bulat.

Aplikasi yang menggunakan citra biner sebagai masukan untuk pemrosesan pengenalan objek, misalnya pengenalan karakter secara optik, analisis kromosom, pengenalan *sparepart* komponen industri, dan sebagainya.

11.2 Konversi Citra hitam-putih ke Citra Biner

Pengkonversian citra hitam-putih (*greyscale*) menjadi citra biner dilakukan untuk alasan-alasan sebagai berikut:

1. Untuk mengidentifikasi keberadaan objek, yang direpresentasikan sebagai daerah (region) di dalam citra. Misalnya kita ingin memisahkan (segmentasi) objek dari gambar latar belakangnya. *Pixel-pixel* objek dinyatakan dengan nilai 1 sedangkan *pixel* lainnya dengan 0. Objek ditampilkan seperti gambar siluet. Untuk memperoleh siluet yang bagus, objek harus dapat dipisahkan dengan mudah dari gambar latar belakangnya.
2. 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/ topologi objek dapat dihitung dari citra biner. Hal ini berguna untuk pengambilan keputusan.
3. Untuk menampilkan citra pada piranti keluaran yang hanya mempunyai resolusi intensitas satu bit, yaitu piranti penampil dua-aras atau biner seperti pencetak (*printer*).
4. Mengkonversi citra yang telah ditingkatkan kualitas tepinya (*edge enhancement*) ke penggambaran garis-garis tepi. Ini perlu untuk membedakan tepi yang kuat yang berkoresponden dengan batas-batas objek dengan tepi lemah yang berkoresponden dengan perubahan *illumination*, bayangan, dll.

Pengambangan

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.

Dua pendekatan yang digunakan dalam operasi pengambangan adalah pengambangan secara global dan pengambangan secara lokal.

a. Pengambangan secara global (*global image thresholding*)

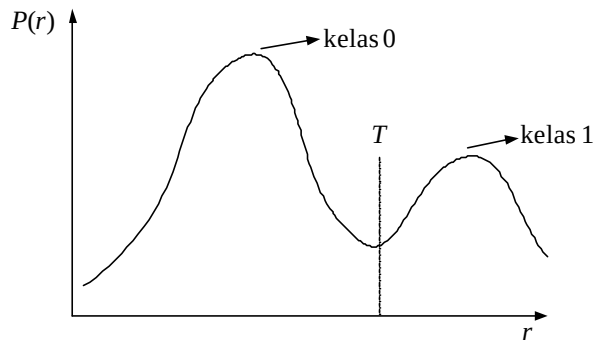
Setiap *pixel* di dalam citra dipetakan ke dua nilai, 1 atau 0 dengan fungsi pengambangan:

$$f_B(i,j) = \begin{cases} 1, & f_g(i,j) \le T \\ 0, & \text{lainnya} \end{cases} \qquad (11.1)$$

yang dalam hal ini, $f_g(i, j)$ adalah citra hitam-putih, $f_B(i, j)$ adalah citra biner, dan T adalah nilai ambang yang dispesifikasikan. Dengan operasi pengambangan tersebut, objek dibuat berwarna gelap (1 atau hitam) sedangkan latar belakang berwarna terang (0 atau putih).

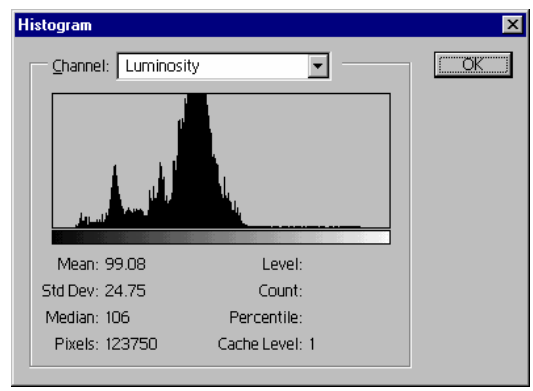
Nilai ambang T dipilih sedemikian sehingga galat yang diperoleh sekecil mungkin. Cara yang umum menentukan nilai T adalah dengan membuat histogram citra. Jika citra mengandung satu buah objek dan latar belakang mempunyai nilai intensitas yang homogen, maka citra tersebut umumnya mempunyai histogram *bimodal* (mempunyai dua puncak atau dua buah maksimum lokal) seperti yang ditunjukkan pada Gambar 11.3. Nilai T dipilih pada nilai minimum lokal yang terdapat di antara dua puncak. Dengan cara seperti ini, kita tidak hanya mengkonversi citra hitam-putih ke citra biner, tetapi sekaligus melakukan segmentasi objek dari latar belakangnya.

Gambar 11.4 memperlihatkan segmentasi objek (botol dan apel) dari latar belakangnya dengan cara mengkonversikan citra hitam-putihnya menjadi citra biner dengan menggunakan nilai ambang T = 90 dan T = 100. Gambar 11.5 memperlihatkan konversi citra Lena menjadi citra biner dengan T = 128 dan T = 150.



Gambar 11.3 *Penentuan nilai ambang T*

(a)

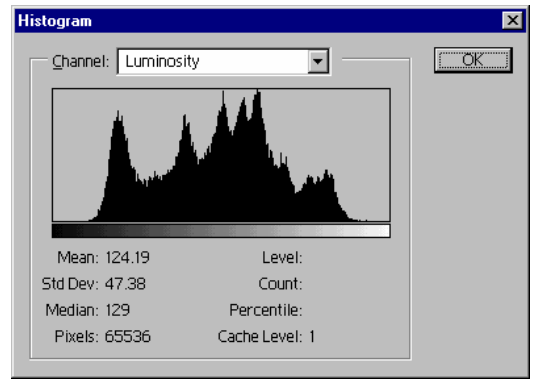


(b)

(c) (d)

Gambar 11.4 *(a) Citra botol, (b) histogram, (c) T = 90, dan (d) T = 100*

(a) Citra Lena



(b) Histogram citra Lena

(c) T = 128

(d) T = 150

Gambar 11.5 *Operasi pengambangan pada citra Lena*

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

$$f_B(i,j) = \begin{cases} 1, & T_1 \le f_g(i,j) \le T_2 \\ 0, & \text{lainnya} \end{cases} \qquad (11.2)$$

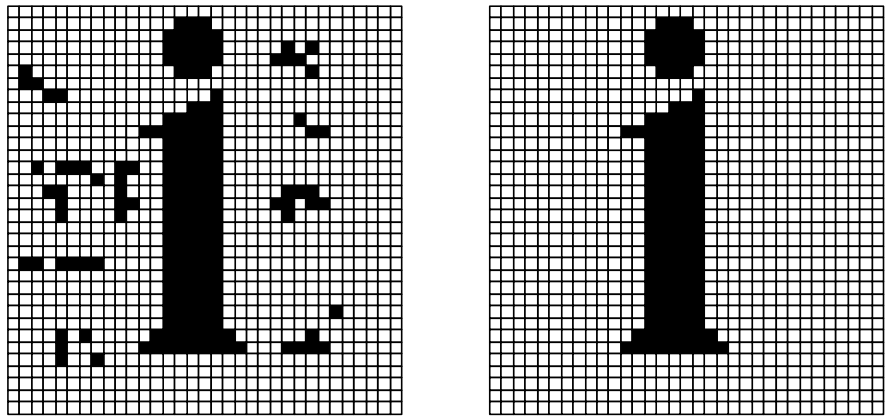
b. Pengambangan secara lokal adaptif (*locally adaptive image thresholding*)
Pengambangan secara global tidak selalu tepat untuk seluruh macam gambar. Beberapa informasi penting di dalam gambar mungkin hilang karena pengambangan global ini. Lagipula, tidak ada harga nilai ambang yang berlaku secara global untuk seluruh daerah citra (misalnya pada citra kedokteran, citra pemandangan alam, dsb).

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 3×3 atau 5×5 *pixel*. Nilai ambangnya ditentukan sebagai fungsi rata-rata derajat keabuan di dalam dearah citra tersebut. Intensitas *pixel* yang berbeda secara signifikan dari nilai rata-rata tersebut dianggap mengandung informasi kontras dan ini harus dipertahankan di dalam citra biner.

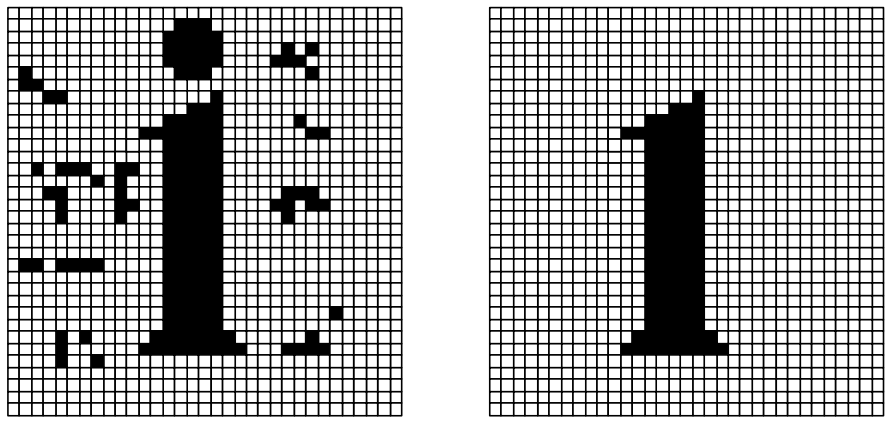
Dengan pengambangan secara lokal adaptif, secara subjektif citra biner yang dihasilkan terlihat lebih menyenangkan dan sedikit informasi yang hilang.

11.3 Penapis Luas

Proses pengambangan menghasilkan citra biner. Seringkali citra biner yang dihasilkan mengandung beberapa daerah yang dianggap sebagai gangguan. Biasanya daerah gangguan itu berukuran kecil. Penapis luas dapat digunakan untuk menghilangan daerah gangguan tersebut [JAI95]. 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 (Gambar 11.6 dan 11.7).



Gambar 11.6 *Kiri: gangguan pada citra biner yang mengandung huruf "i"; Kanan: citra yang dihasilkan setelah dilakukan penapisan (T = 10) [JAI95].*



Gambar 11.7 *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" [JAI95].*

11.4 Pengkodean Citra Biner

Citra biner umumnya dikodekan dengan metode *run-length encoding* (*RLE*). Metode pengkodean ini menghasilkan representasi citra yang mampat.

Dua pendekatan yang digunakan dalam penerapan *RLE* pada citra biner:

a. Posisi awal kelompok nilai 1 dan panjangnya (*length of runs*)
b. Panjang *run*, dimulai dengan panjang *run* 1.

Contoh 11.1. Misalkan citra binernya 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	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:

(i) pendekatan pertama:
(1, 3) (7, 2) (12, 4) (17, 2) (20, 3)
(5, 13) (19, 4)
(1, 3) (17, 6)

(ii) pendekatan kedua
3, 3, 2, 3, 4, 1, 2, 1, 3
0, 4, 13, 1, 4
3, 13, 6 ■

11.5 Segmentasi Citra Biner

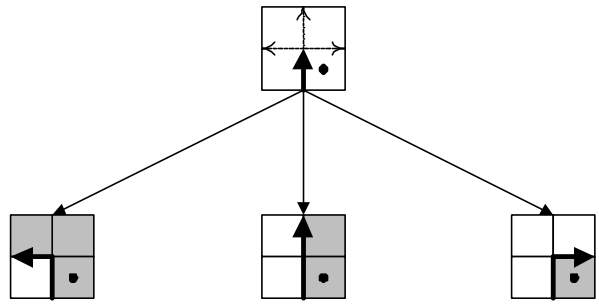
Proses awal yang dilakukan dalam menganalisis objek di dalam citra biner adalah segmentasi objek. Proses segmentasi bertujuan mengelompokkan *pixel-pixel* objek menjadi wilayah (*region*) yang merepresentasikan objek.

Ada dua pendekatan yang digunakan dalam segmentasi objek:

1. Segmentasi berdasarkan batas wilayah (tepi dari objek).
   Pixel-pixel tepi ditelusuri sehingga rangkaian *pixel* yang menjadi batas (*boundary*) antara objek dengan latar belakang dapat diketahui secara keseluruhan (algoritma *boundary following*).
2. Segmentasi ke bentuk-bentuk dasar (misalnya segmentasi huruf menjadi garis-garis vertikal dan horizontal, segmentasi objek menjadi bentuk lingkaran, elips, dan sebagainya).

Kita hanya akan membahas pendekatan pertama.

Segmentasi berdasarkan batas wilayah.
Pada citra biner, batas antara objek dengan latar belakang terlihat jelas. *Pixel* objek berwarna hitam sedangkan *pixel* latar belakang berwarna putih. Pertemuan antara *pixel* hitam dan putih dimodelkan sebagai segmen garis. Penelusuran batas wilayah dianggap sebagai pembuatan rangkaian keputusan untuk bergerak lurus, belok kiri, atau belok kanan seperti yang diperlihatkan pada Gambar 11.8.



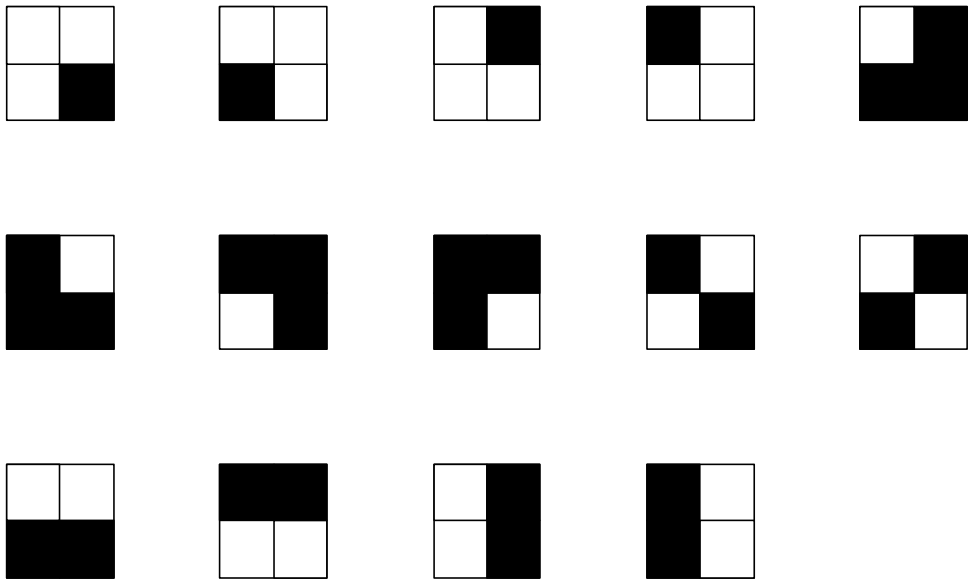
Gambar 11.8 *Proses penelusuran batas wilayah dalam citra biner [DUL97].*

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

Algoritma menentukan arah berikutnya:

```
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
```

Metode pendeteksian batas wilayah yang lain adalah pendeteksian secara **topologi**. Pada metode topologi, setiap kelompok 4-*pixel* bertetangga diperiksa, dan bila kelompok tersebut sama dengan salah satu bentuk pada Gambar 11.9, maka pada titik tengah dari kelompok *pixel* tersebut terdapat tepi.

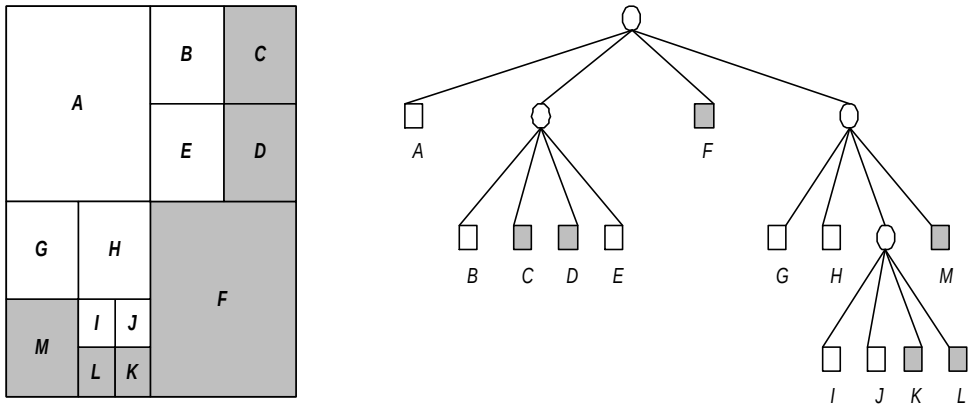


Gambar 11.9 *Bentuk-bentuk yang menghasilkan titik tepi [MEN89].*

Titik tepi yang dideteksi selanjutnya dihubungkan oleh garis-garis penghubung. Arah garis penghubung dikodekan dengan kode rantai (*chain code*).

11.6 Representasi Wilayah

Wilayah (*region*) di dalam citra biner dapat direpresentasikan dalam beberapa cara. Salah satu cara yang populer adalah representasi wilayah dengan **pohon-empatan** (*quadtree*). Setiap simpul di dalam pohon-empatan merupakan salah satu dari tiga ketagori: putih, hitam, dan abu-abu. Pohon-empatan diperoleh dengan membagi citra secara rekursif. Wilayah di dalam citra dibagi menjadi empat buah upa-wilayah yang berukuran sama. Untuk setiap upa-wilayah, bila *pixel-pixel* di dalam wilayah tersebut semuanya hitam atau semuanya putih, maka proses pembagian dihentikan. Sebaliknya, bila *pixel-pixel* di dalam upa-wilayah mengandung baik *pixel* hitam mapupun *pixel* putih (kategori abu-abu), maka upa-wilayah tersebut dibagi lagi mejadi empat bagian. Demikian seterusnya sampai diperoleh upa-wilayah yang semua *pixel*-nya hitam atau semua *pixel*-nya putih. Proses pembagian tersebut digambarkan dengan pohon-empatan. Dinamakan pohon-empatan karena setiap simpul mempunyai tepat empat anak, kecuali simpul daun. Gambar 10.10 memperlihatkan contoh representasi wilayah dengan pohon empatan.



Kode: gwgwbbwbgwwgwwgwwbbb
Di-*decode* sebagai: *g*(*wg*(*wbbw*)*bg*(*wwg*(*wwbb*)*b*))
Keterangan: *b* = *black*, *w* = *white*, *g* = *gray*

(a) Citra biner (b) Pohon-empatan

Gambar 11.10 *Representasi wilayah dengan pohon-empatan*

11.7 Properti Geometri

Setelah proses segmentasi objek selesai dilakukan, maka proses berikutnya adalah menganalisis objek untuk mengenali objek tersebut. Analisis objek didasarkan pada ciri khas (*feature*) geometri pada objek tersebut. Kita asumsikan di dalam citra biner hanya terdapat 1 buah objek.

Ada dua kelompok ciri khas pada objek [JAI95]:
a. *Global feature*, yaitu ciri khas keseluruhan objek.
b. *Local feature*, yaitu ciri khas bagian tertentu dari objek.

Besaran yang termasuk *global feature*:
(i) Luas atau ukuran objek (*A*)

$$A = \sum_{i=1}^{n} \sum_{j=1}^{m} f(i, j) \qquad (11.3)$$

Catatan: $f(i, j) = 1$ jika (i, j) adalah *pixel* objek

(ii) Pusat massa
Berguna untuk menentukan posisi objek.

$$\bar{x} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} j.f(i,j)}{A} \tag{11.4}$$

$$\bar{y} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} i.f(i,j)}{A} \tag{11.5}$$

(iii) Momen inersia (M)

$$M_x = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} j^2.f(i,j)}{A} \tag{11.6}$$

$$M_y = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} i^2.f(i,j)}{A} \tag{11.7}$$

(iv) Keliling objek (K)
Menghitung panjang batas wilayah. *Pixel* dalam batas wilayah horizontal atau vertikal dianggap satu satuan panjang, sedangkan *pixel* pada arah diagonal panjangnya $\sqrt{2}$ satuan.

(v) Tinggi (T)
Dihitung dari jarak vertikal dari *pixel* tertinggi dan terendah dari objek. Jarak antara *pixel* (i_1, j_1) dan *pixel* (i_2, j_2) dapat dihitung dengan bermacam-macam rumus:

- *Euclidean*

$$d_{\text{Euclidean}} = \sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2} \tag{11.8}$$

- *City-block*

$$d_{\text{city}} = |i_1 - i_2| + |j_1 - j_2| \tag{11.9}$$

- *Chessboard*

$$d_{\text{chess}} = \max\left(|i_1 - i_2|, |j_1 - j_2|\right) \quad (11.10)$$

(vi) Lebar (*L*)
Dihitung dari jarak horizontal dari *pixel* tertinggi dan terendah dari objek.

(vii) Diameter
Dihitung dari jarak paling jauh dari dua titik pada objek.

(viii) Kompleksitas bentuk
Menyatakan seberapa rumitnya suatu bentuk. Didefinisikan sebagai K^2/A, yang dalam hal ini K = keliling, A = luas.

(ix) Proyeksi
Menyatakan bentuk yang diperoleh dari hasil proyeksi objek terhadap garis sumbu.
Proyeksi citra biner terhadap garis horizontal dan garis vertikal dihitung dengan rumus:

$$H(i) = \sum_{j=1}^{m} f(i, j) \quad (11.11)$$

$$V(i) = \sum_{i=1}^{n} f(i, j) \quad (11.12)$$

Sedangkan besaran yang termasuk *local feature* antara lain:
(i) Arah dan panjang segmen garis lurus
Arah garis dinyatakan dengan kode Freeman, sedangkan panjang garis dihitung sebagai jarak antara ujung-ujung garis.

(ii) Sudut antar garis
Menyatakan besar sudut antara dua garis lurus yang berpotongan.

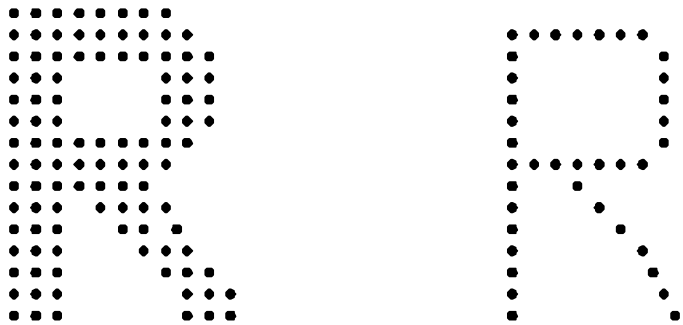
(iii) Jarak relatif
Dihitung sebagai jarak antara dua titik.

(iv) *Object signature*
Menyatakan jarak dari pusat massa ke tepi suatu objek pada arah 0 sampai 360 derajat.

11.8 Penipisan Pola

Pada aplikasi pencocokan pola, banyak bentuk terutama bentuk yang mengulur/memanjang yang dapat dinyatakan dalam versi yang lebih tipis. Bentuk yang lebih tipis terdiri dari garis-garis terhubung yang disebut rangka (*skeleton*) atau tulang atau garis inti. Idealnya, rangka tersebut membentang sepanjang garis sumbu objek.

Penipisan (*thinning*) adalah operasi pemrosesan citra biner yang dalam hal ini objek (*region*) direduksi menjadi rangka 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. Sebagai contoh, Gambar 11.11 adalah huruf "R" dan hasil penipisan polanya menjadi rangka "R".

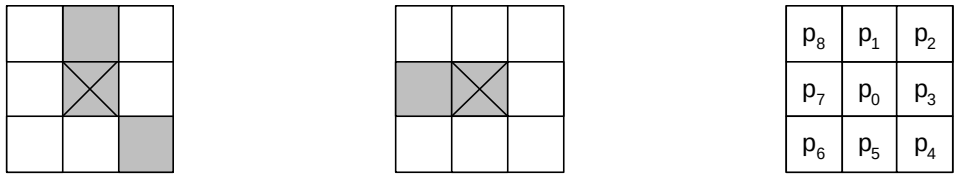


(a) Huruf "R" (b) Hasil penipisan huruf "R

Gambar 11.11 *Penpisan pola huruf "R"*

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 [PIT93]:

1. Mempertahankan keterhubungan *pixel-pixel* objek pada setiap lelaran. Dengan kata lain, tidak menyebabkan bentuk objek menjadi terputus (Gambar 11.12(a)).
2. Tidak memperpendek ujung lengan dari bentuk yang ditipiskan (Gambar 11.12(b)).



Gambar 11.12 *(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. Notasi *pixel* di dalam jendela 3×3 diperlihatkan pada Gambar 11.12(c). Algoritma bekerja secara iteratif, pada setiap lelaran dilakukan premrosesan pada jendela yang berukuran 3×3 *pixel*.

Algoritmanya adalah sebagai berikut [PIT93]:

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 $p_1p_2p_3 \ldots p_8p_1$. 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.

Algoritma penipisan pola dalam bahasa *C* diperlihatkan pada Algoritma 11.1.

```
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)
   Keluaran: citra bienrf
*/
{
  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 ´ 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);
}
```

Algoritma 11.1 *Penipisan pola.*