OPERASI PADA CITRA BINER

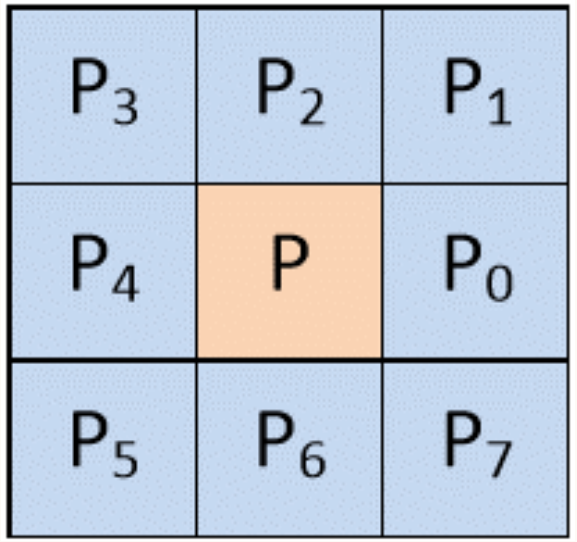
PERTEMUAN KE 13

MANFAAT

Beberapa pemrosesan citra biner, dengan menggunakan teknik perbandingan panjang dan lebar obyek pada citra. Dapat di gunakan antara lain untuk representasi bentuk, ekstrasi tepi obyek, mengikuti kontur, luas, diameter, fitur depresi dan pelabelan obyek.

EKSTRASI TEPI OBYEK

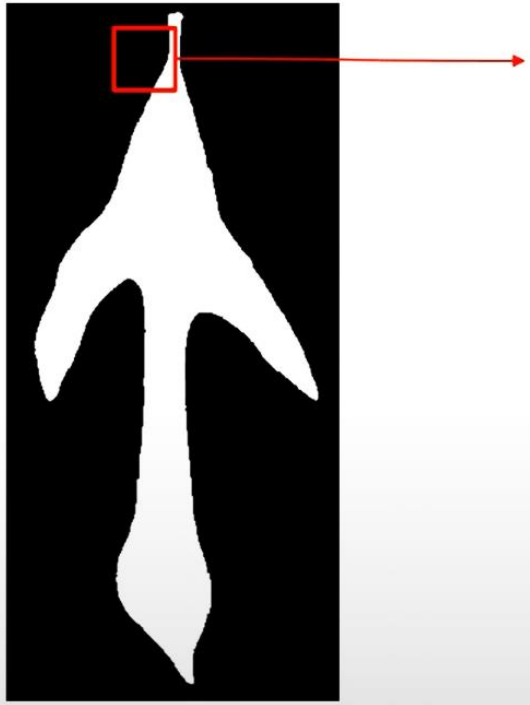
Tepi obyek pada citra biner dapat diperoleh dengan melakukan pemrosesan menggunakan 8 ketetanggan, yang dinyatakan dengan P0 hingga P7.



Gambar 11.1 Piksel dan 8 piksel tetangga

EKSTRASI TEPI OBYEK

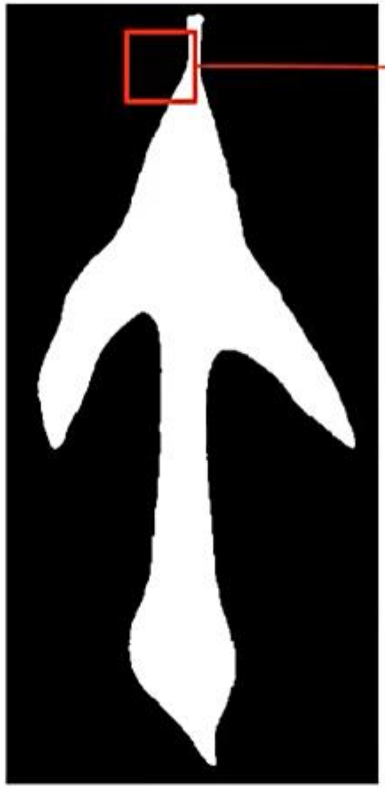
Jika sekeliling piksel P bernilai sama (**semua = 0 atau semua = 1**), maka diasumsikan bahwa piksel **P tidak berada di tepi obyek.**



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

Gambar 11.2 Deteksi tepi obyek

EKSTRASI TEPI OBYEK



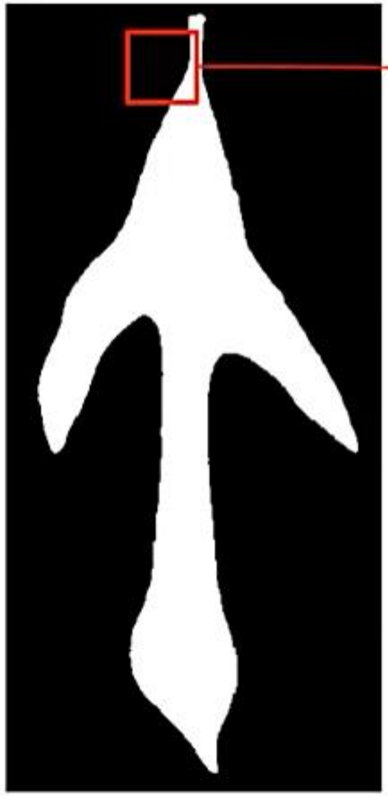
Bukan tepi, karena jumlah seluruh pixel tetangganya sama, pixel tidak diubah

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

Lakukan windowing dengan 8-ketetanggaan

Gambar 11.2 Deteksi tepi obyek

EKSTRASI TEPI OBYEK



Bukan tepi, karena semua pixel tetangga nilainya sama, ada pengubahan nilai pixel karena jumlah pixel tetangga == 8

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

Lakukan windowing dengan 8-ketetanggaan

Gambar 11.2 Deteksi tepi obyek

EKSTRASI TEPI OBYEK

ALGORITMA MEMPEROLEH TEPI OBYEK

```
function [G] = tepibiner(F)
% TEPIBINER Berguna untuk mendapatkan tepi objek
%     pada citra biner

[jum_baris, jum_kolom] = size(F);

G = zeros(jum_baris, jum_kolom);

for q = 2 : jum_baris - 1
   for p = 2 : jum_kolom - 1
      p0 = F(q, p+1);
      p1 = F(q-1, p+1);
      p2 = F(q-1, p);
      p3 = F(q-1, p-1);
      p4 = F(q, p-1);
      p5 = F(q+1, p-1);
      p6 = F(q+1, p);
      p7 = F(q+1, p+1);
      sigma = p0 + p1 + p2 + p3 + p4 + p5 + p6 + p7;
      if sigma == 8
         G(q, p) = 0;
      else
         G(q, p) = F(q, p);
      end
   end
end
```

EKSTRASI TEPI OBYEK

PENJELASAN PADA ALGORITMA MEMPEROLEH TEPI OBYEK

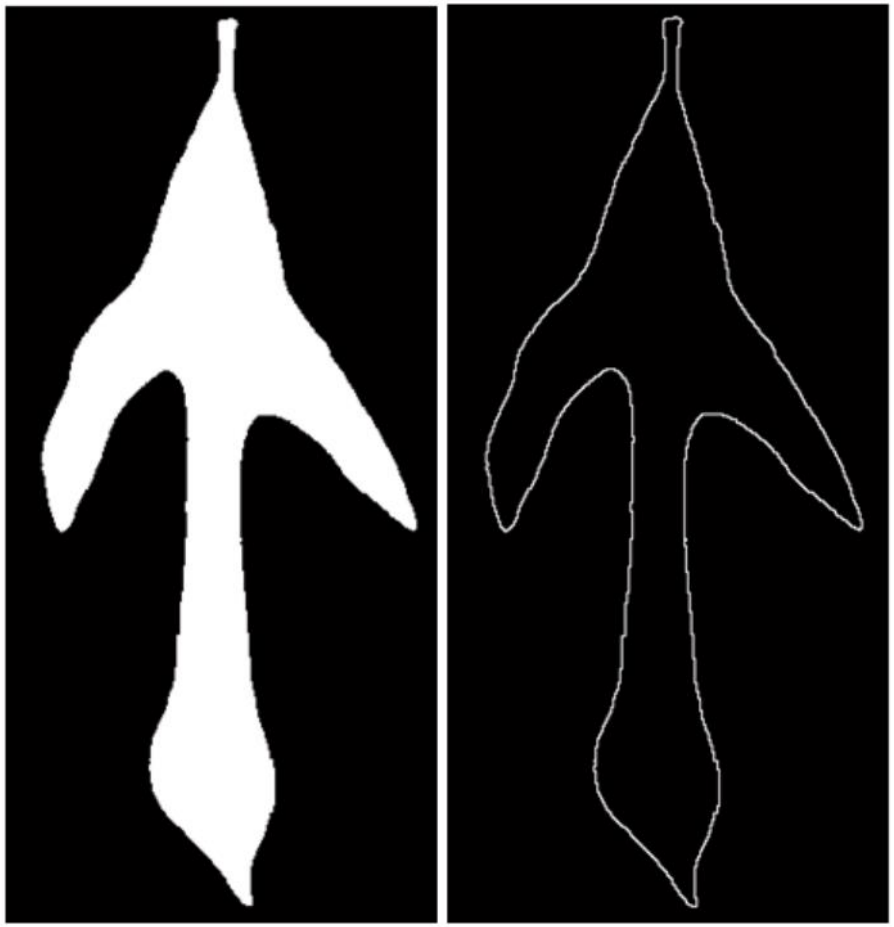
Pada algoritma memperoleh tepi obyek mengasumsikan bahwa semua piksel pada kolom pertama, kolom, terakhir, baris pertama, dan baris terakhir tidak ada yang bernilai 1. Apabila ada kemungkinan bahwa piksel pada posisi tersebut ada yang bernilai satu, perlu dibentuk larik baru yang berukuran (m+2) x (n+2), yang mencakup seluruh nilai f dan dengan bagian tepi larik berisi 0.

EKSTRASI TEPI OBYEK

HASIL PROSES PADA ALGORITMA MEMPEROLEH TEPI OBYEK

Dengan menggunakan perintah dibawah, untuk menampilkan hasil keluaran, maka diperoleh hasil seperti pada Gambar 11.3.

```
>> Img = imread('C:\Image\daun_bin.png');
>> G = tepibiner(Img);
>> imshow(G)
```



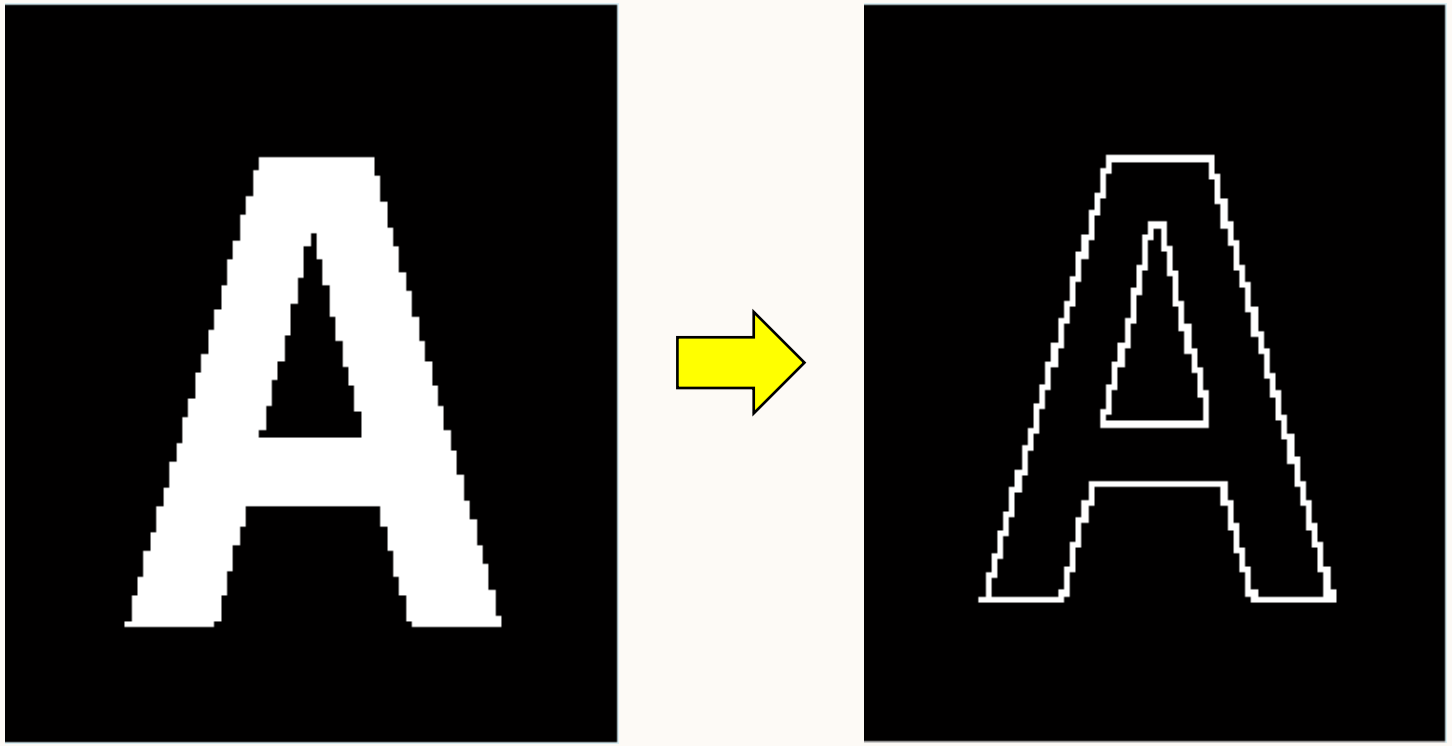
(a) Citra daun (b) Tepi citra daun

Gambar 11.3 Hasil pemrosesan tepi objek daun yang diperoleh

EKSTRASI TEPI OBYEK

HASIL PROSES PADA ALGORITMA MEMPEROLEH TEPI OBYEK

Jika obyek berlubang , kontur bagian dalam juga akan dibuat oleh fungsi tepi biner, seperti ditunjukan Gambar 8.4



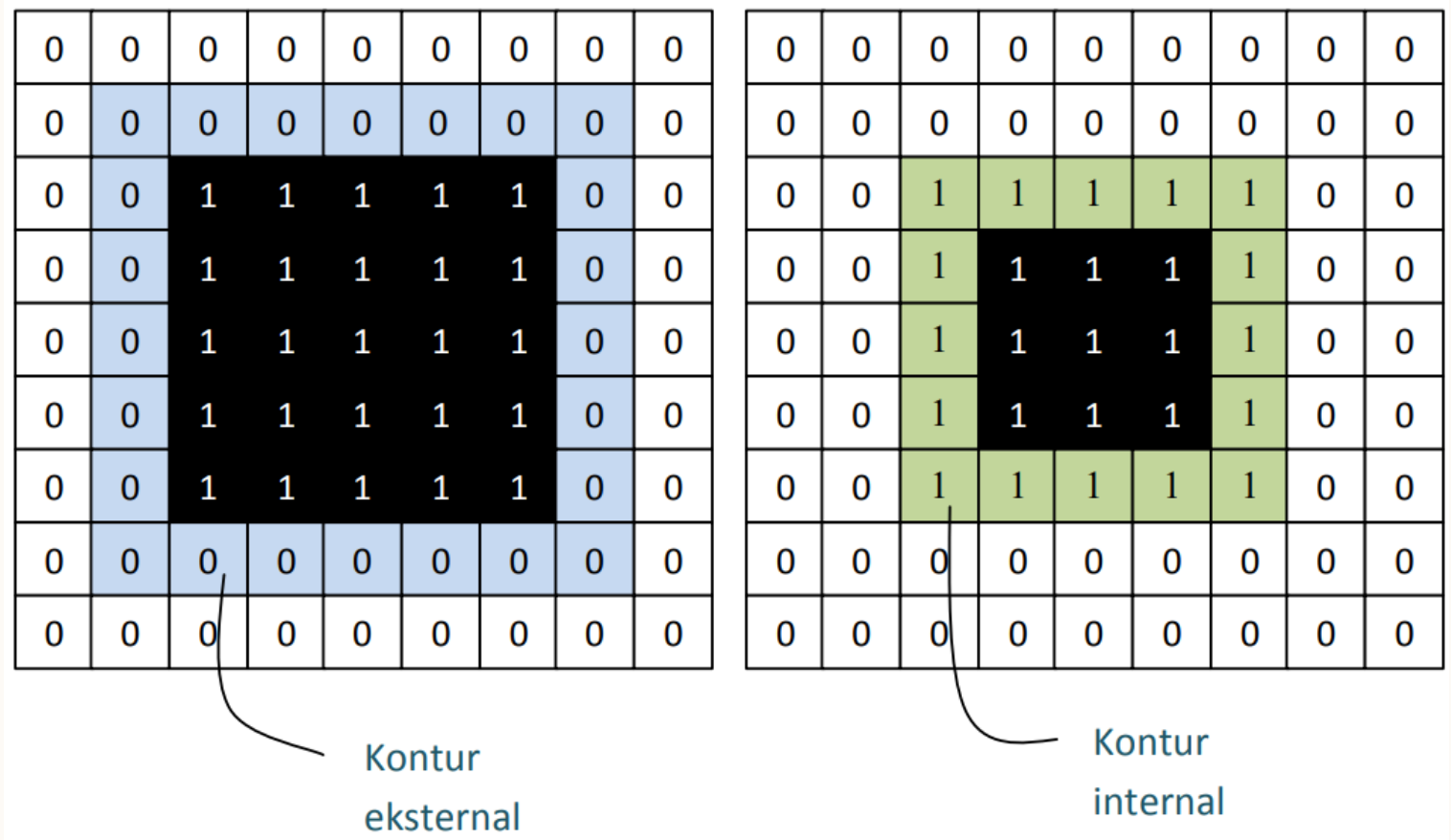
(a) Citra dengan obyek berlubang (b) Hasil perolehan tepi citra

Gambar 11.4 Hasil pemrosesan tepi objek pada obyek berlubang

MENGIKUTI KONTUR OBYEK

Mengikuti kontur (*Contour Following*) merupakan suatu metode yang digunakan untuk mendapatkan tepi objek.

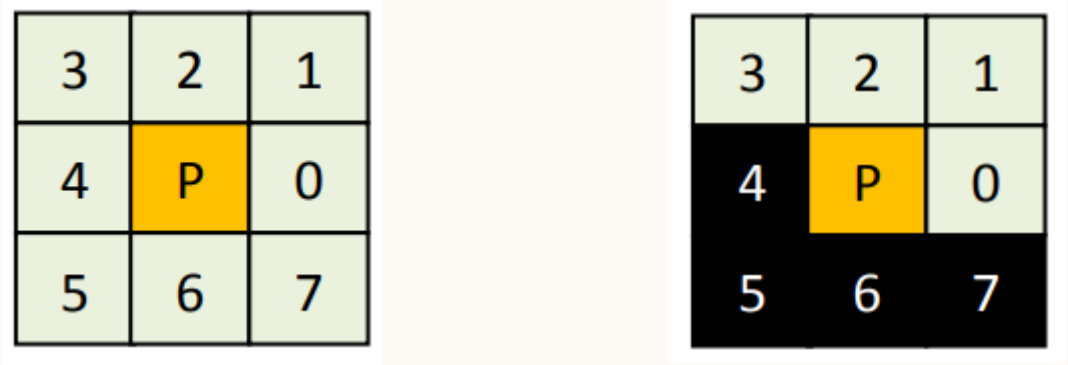
Namun terkait dengan hal itu, terdapat istilah kontur eksternal dan kontur internal. Gambar 11.5 memberikan ilustrasi tentang perbedaan kedua jenis kontur tersebut.



Gambar 11.5 Perbedaan kontur eksternal dan internal

Terlihat bahwa piksel yang menjadi bagian kontur eksternal terletak di luar objek, sedangkan piksel yang menjadi bagian kontur internal terletak di dalam objek itu sendiri.

Label yang digunakan untuk penelusuran kontur, ditunjukan pada Gambar 11.6



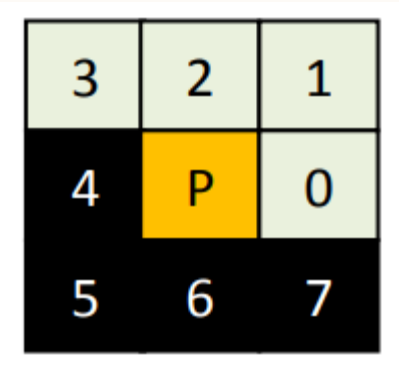
(a) Label tetangga untuk penelusuran kontur

(b) Tetangga berwarna hitam Sebagai basis pencarian titik kedua

Gambar 11.6 Label posisi tetangga dan pencarian tetangga untuk menentukan bagian kedua pada kontur

Piksel tetangga yang diberi latarbelakang hitam merupakan tetangga yang dijadikan acuan untuk mencari titik kedua, yang akan menjadi bagian kontur. Dengan cara seperti itu, piksel yang berada di atas piksel titik awal, ataupun yang berada di kanannya tidak mungkin menjadi piksel kedua yang merupakan bagian kontur.

Pedoman Untuk Memilih Arah



Gambar 11.7 Pencarian tetangga untuk menentukan bagian kedua pada kontur

Pedoman untuk memilih arah 4, 5, 6, atau 7 untuk piksel kedua adalah sebagai berikut:

Pertama-tama, piksel pada arah 4 adalah sebagai piksel ke dua, jika piksel tersebut bernilai 0, piksel arah 5 harus bernilai 1. Jika keadaan tersebut tidak tercapai, maka piksel arah 5 adalah piksel kedua, jika piksel tersebut bernilai 0, maka piksel arah 6 harus bernilai 1. Namun jika hal itu tidak tercapai, piksel arah 6 adalah piksel kedua, kalau piksel tersebut bernilai 0, piksel arah 7 harus bernilai 1. Apabila keadaan terakhir ini tidak tercapai, piksel kedua akan berupa piksel pada arah 7.

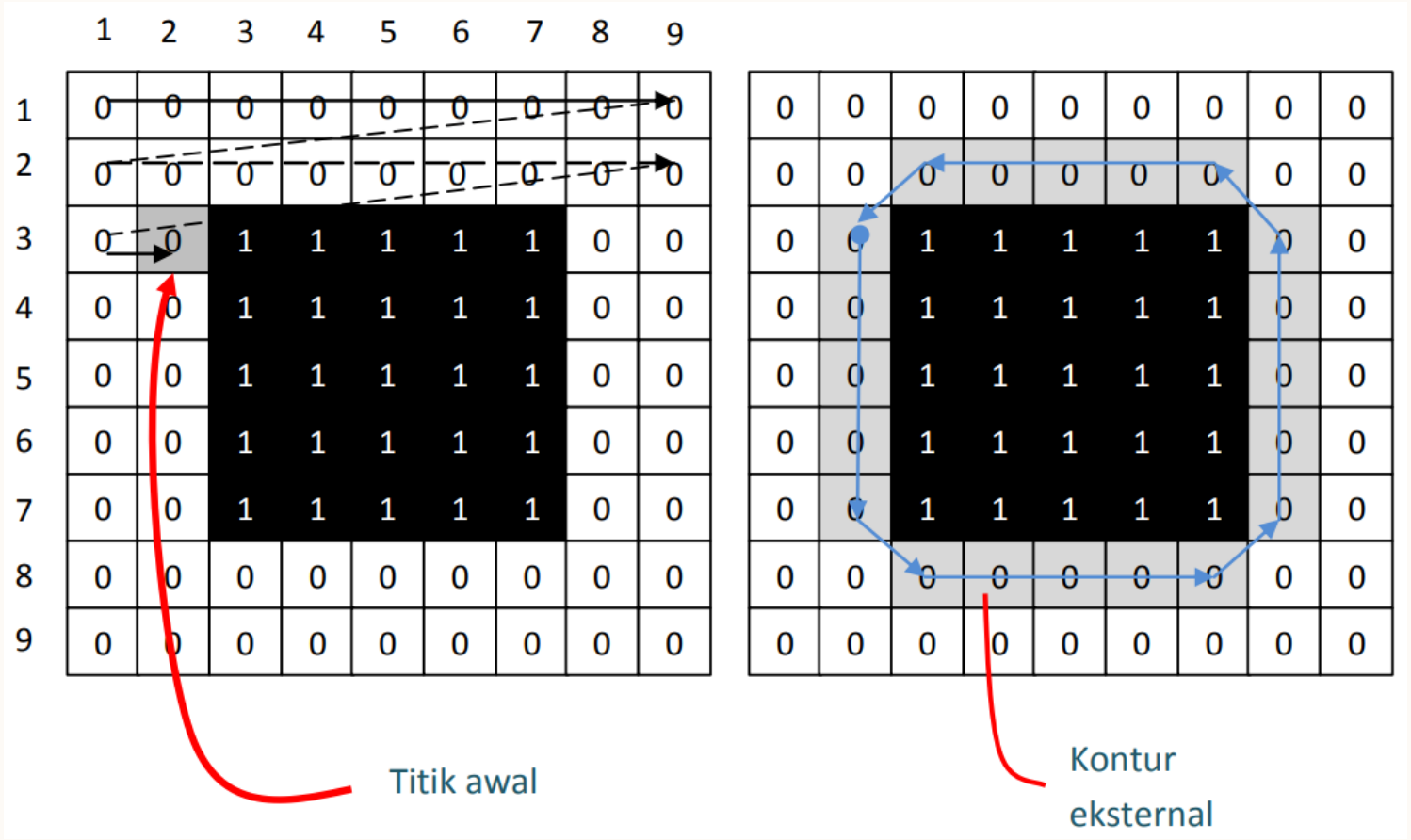
Untuk Memperoleh Kontur Eksternal

Contoh cara untuk memperoleh kontur eksternal, seperti ditunjukan pada Gambar 11. Dengan menggunakan pendekatan 8-ketetanggaan, diperoleh hasil sebagai berikut:

(3,2), (4,2), (5,2), (6,2), (7,2), (8,3), (8,4), (8,5), (8,6), (8,7), (7,8), (6,8), (5,8), (4,8), (3,8), (2,7), (2,6), (2,4), (2,3)

terakhir ini tidak tercapai, piksel kedua akan berupa piksel pada arah 7.

Proses untuk mendapatkan titik awal (yaitu (3,1)) dilakukan dengan melakukan pemindaian seperti yang diilustrasikan dalam Gambar 11.8. Setelah titik awal ditemukan, penelusuran dilakukan seperti terlihat pada Gambar 11.8(b). Penelusuran kontur berakhir setelah bertemu kembali dengan titik awal.



(a) (b)

Gambar 11.8 Perbedaan kontur eksternal dan internal

Terlihat bahwa piksel yang menjadi bagian kontur eksternal terletak di luar objek, sedangkan piksel yang menjadi bagian kontur internal terletak di dalam objek itu sendiri.

Untuk penelusuran piksel-piksel berikutnya dilakukan dengan cara khusus. Untuk kepentingan ini, diperlukan suatu pencatatan untuk mengetahui arah posisi sekarang terhadap posisi sebelumnya dan posisi berikutnya.

Sebagai contoh, **dcp** digunakan untuk mencatat arah posisi sekarang terhadap piksel sebelumnya, **dpc** untuk mencatat arah posisi sebelum terhadap posisi sekarang, dan **dcn** untuk mencatat arah posisi sekarang terhadap piksel berikutnya. Berdasarkan keadaan pada Gambar 11.6(a), hubungan antara **dcp** dan **dpc** adalah berkebalikan, seperti ditunjukan pada Tabel 11.1.

Tabel 11.1 Hubungan antara dpc dan dcp

dpc	dcp = kebalikan(dpc)
0	4
1	5
2	6
3	7
4	0
5	1
6	2
7	3

Listing Program Fungsi Kontur


```
function [Kontur] = get_contour(BW)
%   GET_CONTOUR Berfungsi untuk memperoleh kontur eksternal
%    dari suatu citra biner BW
%    Hasil berupa Kontur yang berisi pasangan X dan Y dari setiap
%    piksel yang menyusun kontur. Kolom 1 menyatakan Y dan
%    kolom 2 menyatakan X

% Peroleh kontur

% Proses rantai kode
DPC = [0, 1, 2, 3, 4, 5, 6, 7];          % Arah sebelumnya ke sekarang
DCP = [4, 5, 6, 7, 0, 1, 2, 3];          % Arah sekarang ke sebelumnya

% Arah  0   1   2   3   4   5  6  7  terhadap posisi sekarang
XP =   [1,  1,  0, -1, -1, -1, 0, 1];
YP =   [0, -1, -1, -1,  0,  1, 1, 1];

% Peroleh titik awal
[tinggi, lebar] = size(BW);
```

Listing Program Fungsi Kontur

```
% Cari titik awal
x1 = 1;
y1 = 1;
selesai = false;
for baris = 1 : tinggi
    for kolom = 1 :lebar
        if BW(baris, kolom) == 1
            y1 = baris;
            x1 = kolom-1;
            selesai = true;

            Kontur(1,1) = y1;
            Kontur(1,2) = x1;
            break;
        end
    end

    if selesai
        break;
    end
end
```

Listing Program Fungsi Kontur


```
% Proses piksel kedua
for i = 4 : 7
   if BW(y1+YP(i+1), x1+XP(i+1)) == 0
      dcn = i;          % Arah sekarang ke sesudahnya
      break;
   end
end

yberikut = y1 + YP(dcn+1);
xberikut = x1 + XP(dcn+1);

indeks = 2;             % Indeks kedua

% Proses peletakan piksel kedua dan seterusnya ke array Kontur
while (yberikut ~= Kontur(1,1)) || (xberikut ~= Kontur(1,2))
   Kontur(indeks,1) = yberikut;
   Kontur(indeks,2) = xberikut;
   dpc = dcn;           % Arah sebelum ke sekarang diisi
                        % dengan arah sekarang ke berikutnya
```

Listing Program Fungsi Kontur

```
% Cari piksel berikutnya
   for r = 0 : 7
      dcp = DCP(dpc+1);
      de = rem(dcp+r, 8);
      di = rem(dcp+r+1, 8);

      cxe = Kontur(indeks,2) + XP(de+1);
      cye = Kontur(indeks,1) + YP(de+1);
      cxi = Kontur(indeks,2) + XP(di+1);
      cyi = Kontur(indeks,1) + YP(di+1);

      if (BW(cye, cxe) == 0) && (BW(cyi, cxi) == 1)
         yberikut = cye;
         xberikut = cxe;
         break;
      end
   end

   % Naikkan indeks
   indeks = indeks + 1;
end
```

Menguji Fungsi Kontur

Untuk menguji fungsi Kontur yang sudah dibuat, maka digunakan perintah sebagai berikut:

```
>> Daun = imread('c:\image\daun_bin.png');
>> C = get_contour(Daun);
>>
```

Dengan cara seperti itu, C berisi data piksel yang menjadi kontur citra biner daun_bin.png.

Untuk membuktikan bahwa C berisi kontur daun, berikan kode seperti berikut:

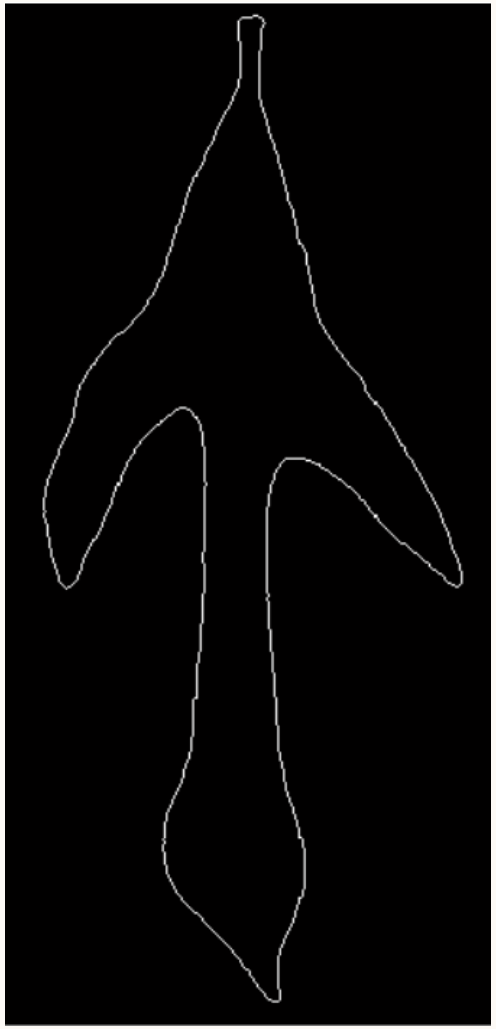
```
>> D = zeros(size(Daun));
>> for p=1:length(C)
    D(C(p,1), C(p,2)) = 1;
    end
>> imshow(D)
```

Untuk membentuk matriks berukuran sama dengan citra Daun dan seluruhnya diisi dengan nol, digunakan

```
>> D = zeros(size(Daun));
```

Selanjutnya, untuk membuat matriks D yang sesuai dengan nilai-nilai koordinat pada larik C diisi dengan angka 1. Dengan demikian, D merekam kontur yang tercatat pada C. Gambar 8.7 menunjukkan hasil imshow(D).

```
>> for p=1:length(C)
    D(C(p,1), C(p,2)) = 1;
    end
>> imshow(D)
```



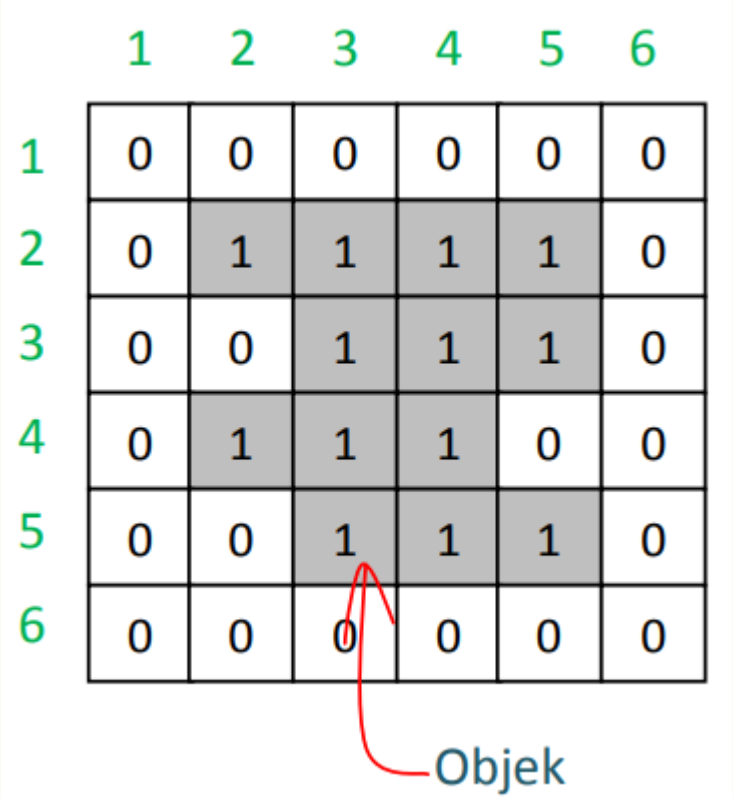
Gambar 11.9 Gambar kontur yang diperoleh melalui pengambilan kontur obyek

KONTUR INTERNAL

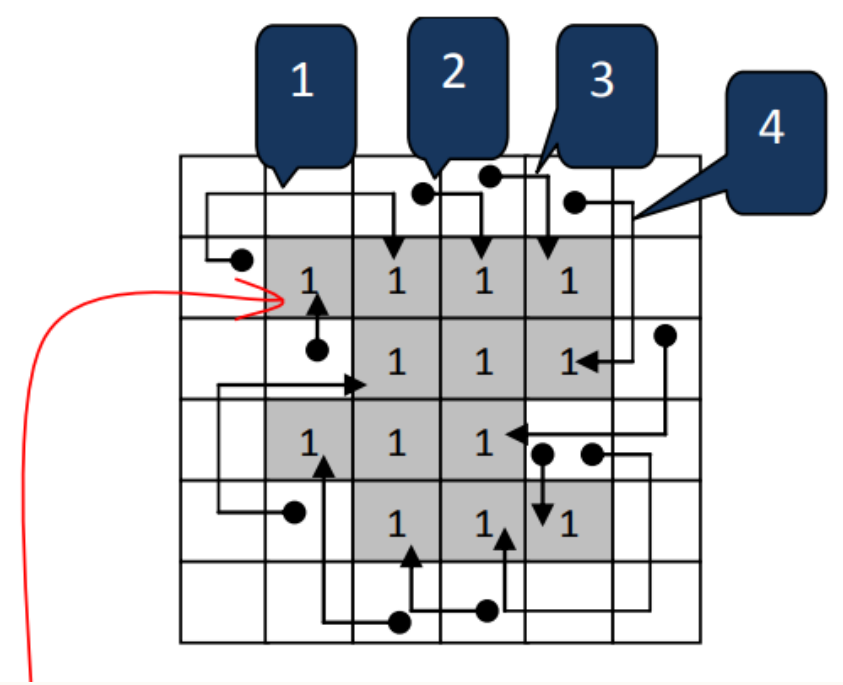
Salah satu cara untuk mendapatkan kontur internal yang telah diurutkan menurut letak piksel, yaitu dengan memanfaatkan algoritma pelacakan kontur Moore.

Algoritma ini antara lain digunakan pada peta topografik digital (Pradha, dkk., 2010).

Untuk memahami proses kerja pada algoritma Moore, perhatikan 11.10. Pada Gambar 11.10 menyatakan keadaan objek pada citra. Piksel yang bernilai 1 menyatakan bagian objek dan yang bernilai 0 adalah bagian latarbelakang.

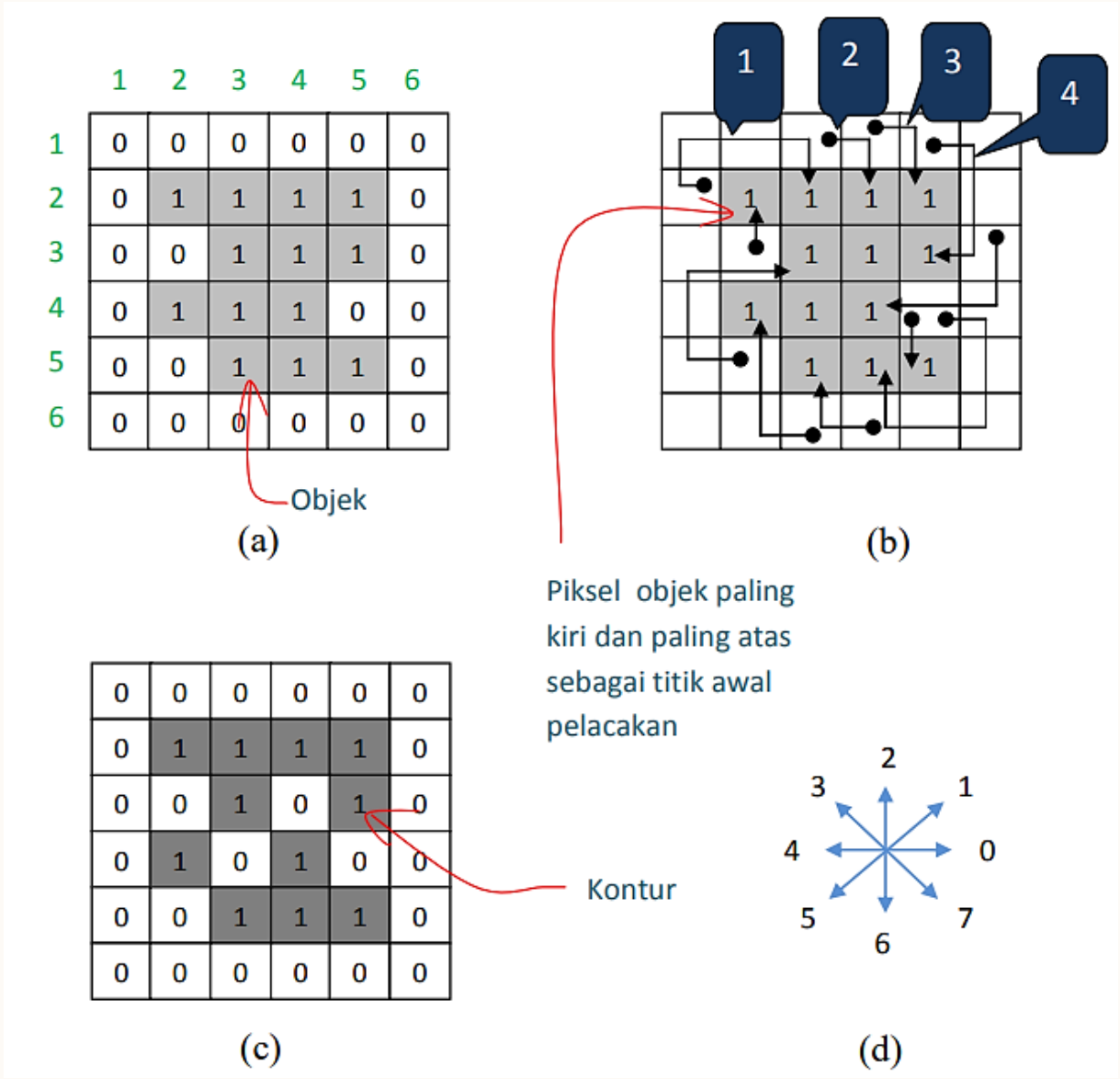


Gambar 11.10 Keadaan objek pada citra



Gambar 11.11 pelacakan kontur

Pada contoh tersebut, pelacakan akan dimulai pada posisi (2,2), yaitu piksel bagian objek yang terletak paling kiri dan paling atas. Adapun titik pencarian untuk piksel kedua dimulai dari arah kiri piksel (2,2) tersebut. Pencarian dilakukan searah jarum jam. Pada langkah pertama, diperoleh piksel pada posisi (2,3). Pencarian berikutnya akan dimulai di posisi (1,3), yaitu yang ditandai dengan bulatan. Pada pencarian kedua, piksel yang didapat, yaitu posisi (2,4), dengan titik pencarian berikutnya dimulai di posisi (1,4). Pada pencarian ketiga, piksel yang didapat adalah pada (2,5), dengan titik pencarian berikutnya dimulai di posisi (1,5). Pada pencarian keempat, piksel yang didapat yaitu (3,5), dengan titik pencarian berikutnya dimulai di posisi (3,6). Jika langkah seperti itu terus diulang, suatu ketika akan diperoleh piksel yang sama dengan piksel yang pertama kali menjadi bagian kontur. Saat itulah proses untuk melacak kontur diakhiri.



Gambar 11.12 Penjelasan pelacakan kontur dengan menggunakan Algoritma Moore

Semua langkah yang terjadi untuk contoh Gambar 11.12(a) ditunjukkan pada Gambar 11.12(b). Penomoran arah pencarian ditunjukkan pada Gambar 11.12(d), sedangkan hasil kontur diperlihatkan pada Gambar 11.12(c)

Listing Program Fungsi Kontur Internal


```
function [Kontur] = inbound_tracing(BW)
% INBOUND_TRACING Memperoleh kontur yang telah
terurutkan
%     dengan menggunakan algoritma pelacakan kontur
Moore

[jum_baris, jum_kolom] = size(BW);

% Peroleh piksel awal
selesai = false;
for p = 1 : jum_baris
   for q = 1 : jum_kolom
      if BW(p, q) == 1
         b0.y = p;
         b0.x = q;

         selesai = true;
         break;
      end
   end
```

Listing Program Fungsi Kontur Internal


```
if selesai
        break;
    end
end

c0 = 4; % Arah barat

% Periksa 8 tetangga dan cari piksel pertama yang bernilai 1
for p = 1 : 8
    [dy, dx] = delta_piksel(c0);
    if BW(b0.y + dy, b0.x + dx) == 1
        b1.y = b0.y + dy;
        b1.x = b0.x + dx;

        c1 = sebelum(c0);
        break;
    else
        c0 = berikut(c0);
    end
end
```

Listing Program Fungsi Kontur Internal


```
Kontur=[ ];
Kontur(1, 1) = b0.y;
Kontur(1, 2) = b0.x;
Kontur(2, 1) = b1.y;
Kontur(2, 2) = b1.x;

%Kontur

n = 2; % Jumlah piksel dalam kontur

b = b1;
c = c1;

% Ulang sampai berakhir
while true
   for p = 1 : 8
      [dy, dx] = delta_piksel(c);
      if BW(b.y + dy, b.x + dx) == 1
         b.y = b.y + dy;
         b.x = b.x + dx;
```

Listing Program Fungsi Kontur Internal


```
            c = sebelum(c);

            n = n + 1;
            Kontur(n, 1) = b.y;
            Kontur(n, 2) = b.x;

            break;
        else
            c = berikut(c);
        end
    end

    % Kondisi pengakhir pengulangan

    if (b.y == b0.y) && (b.x == b0.x)
        break;
    end
end
```

Listing Program Fungsi Kontur Internal


```
function [b] = berikut(x)
if x == 0
   b = 7;
else
   b = x - 1;
end

function [s] = sebelum(x)
if x == 7
   s = 0;
else
   s = x + 1;
end

if s < 2
   s = 2;
elseif s < 4
   s = 4;
elseif s < 6
   s = 6;
else
   s = 0;
end
```

Listing Program Fungsi Kontur Internal


```
function [dy, dx] = delta_piksel(id)
if id == 0
    dx = 1; dy = 0;
elseif id == 1
    dx = 1; dy = -1;
elseif id == 2
    dx = 0; dy = -1;
elseif id == 3
    dx = -1; dy = -1;
elseif id == 4
    dx = -1; dy = 0;
elseif id == 5
    dx = -1; dy = 1;
elseif id == 6
    dx = 0; dy = 1;
elseif id == 7
    dx = 1; dy = 1;
end
```

Contoh penggunaan inbound_tracing :

```
>> D = [ 0 0 0 0 0 0
 0 1 1 1 1 0
 0 0 1 1 1 0
 0 1 1 1 0 0
 0 0 1 1 1 0
 0 0 0 0 0 0 ]; ⏎
```

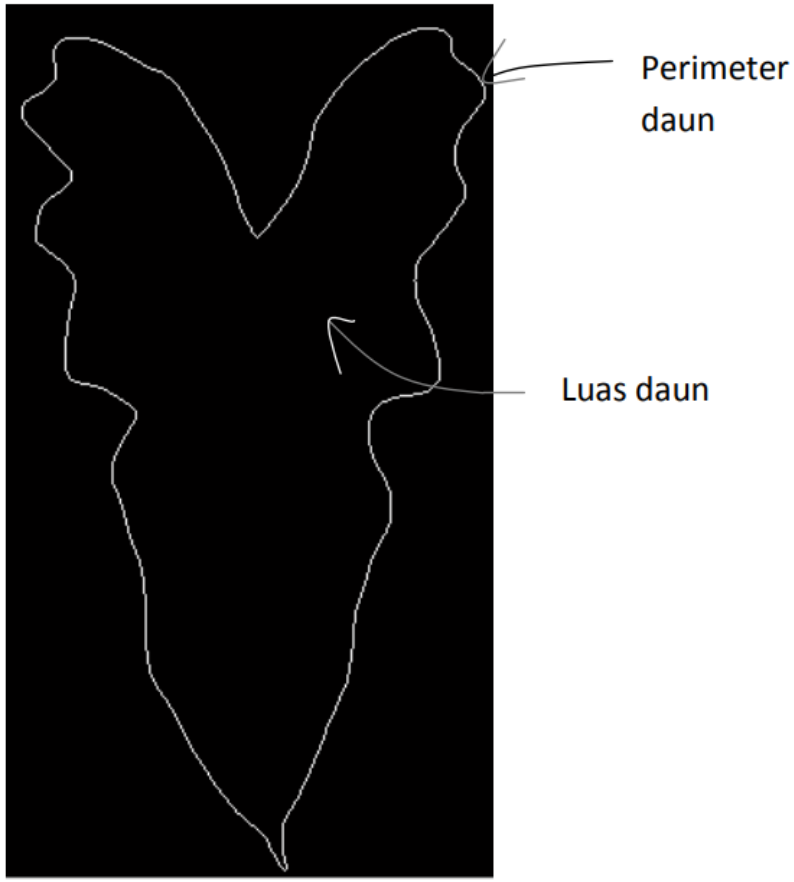
Saat dikketikan perintah seperti dibawah ini :

```
>> P = inbound_tracing(D) ⏎
P =
 2 2
 2 3
 2 4
 2 5
 3 5
 4 4
 5 5
 5 4
 5 3
 4 2
 3 3
 2 2
>>
```

Perhatikan, elemen **pertama** dan **terakhir** pada P sama.

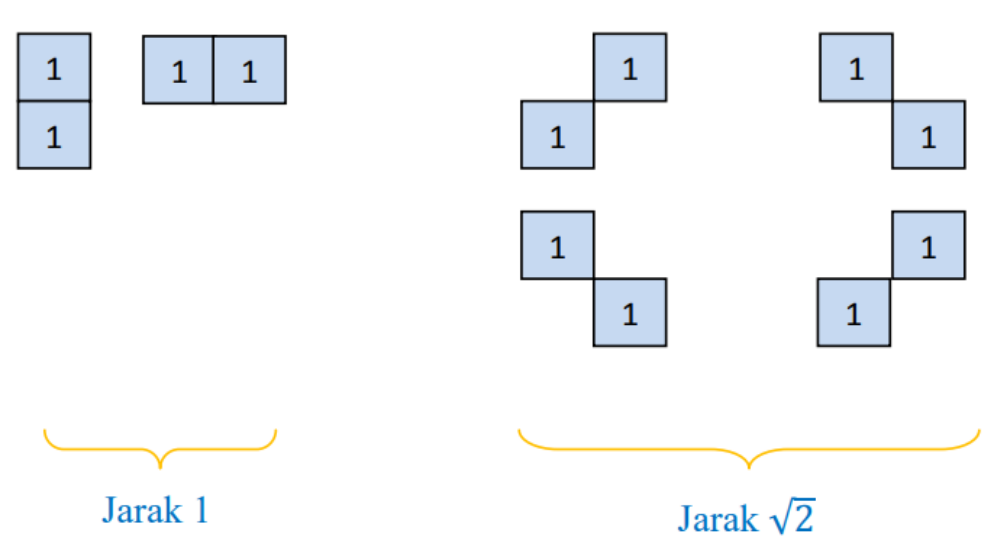
PERIMETER

Perimeter atau keliling menyatakan panjang tepi suatu objek. Ilustrasinya dapat dilihat pada Gambar 11.13.



Gambar 11.13 Perimeter dan luas daun

Algoritma estimasi perimeter di depan memberikan hasil yang baik ketika tepi objek terhubung dengan 4-ketetanggaan, tetapi tidak tepat kalau terhubung menurut 8-ketetanggaan (Costa & Cesar, 2001).



Gambar 11.11 Jarak antarpiksel pada 8-ketetanggaan

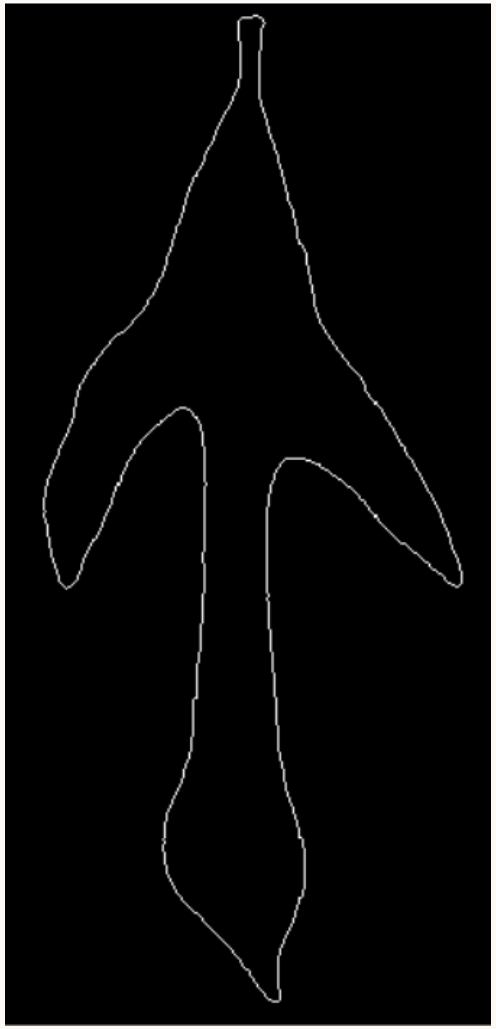
Hal itu terjadi karena jarak antara dua piksel tidak bersifat konstan (dapat berupa 1 atau √2) pada 8-ketetanggaan, sedangkan jarak selalu 1 pada 4-ketetanggaan. Ilustrasi mengenai jarak antarpiksel dapat dilihat pada Gambar 11.11.

Untuk membentuk matriks berukuran sama dengan citra Daun dan seluruhnya diisi dengan nol, digunakan

```
>> D = zeros(size(Daun));
```

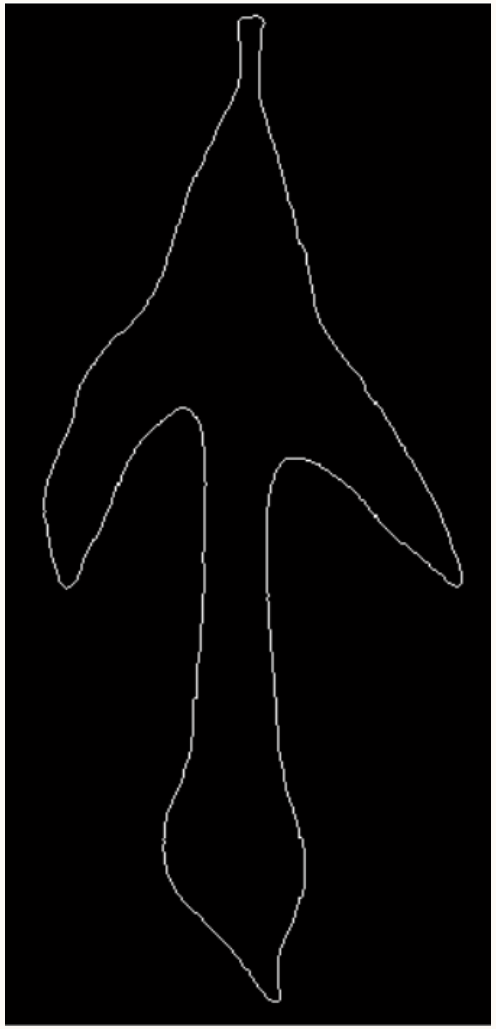
Selanjutnya, untuk membuat matriks D yang sesuai dengan nilai-nilai koordinat pada larik C diisi dengan angka 1. Dengan demikian, D merekam kontur yang tercatat pada C. Gambar 8.7 menunjukkan hasil imshow(D).

```
>> for p=1:length(C)
    D(C(p,1), C(p,2)) = 1;
    end
>> imshow(D)
```



Gambar 11.9 Gambar kontur yang diperoleh melalui pengambilan kontur obyek

```
Contoh: >> D = [ 0 0 0 0 0 0
                 0 1 1 1 1 0
                 0 0 1 1 1 0
                 0 1 1 1 0 0
                 0 0 1 1 1 0
                 0 0 0 0 0 0 ]; ⏎
>> luas(D) ⏎
ans = 13
>>
>> Daun = imread('c:\image\daun_bin.png'); ⏎
>> luas(Daun) ⏎
ans = 31862
>>
```



Gambar 11.9 Gambar kontur yang diperoleh melalui pengambilan kontur obyek

TERIMAKASIH

Ada yang ditanyakan?