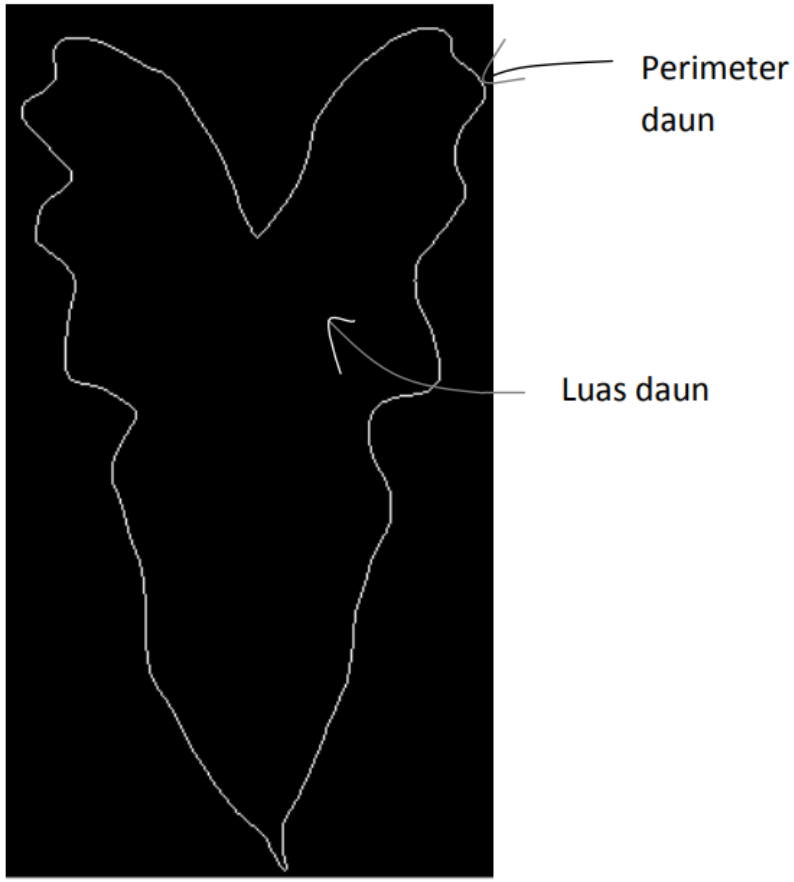
LANJUTAN OPERASI PADA CITRA BINER

PERTEMUAN KE 14

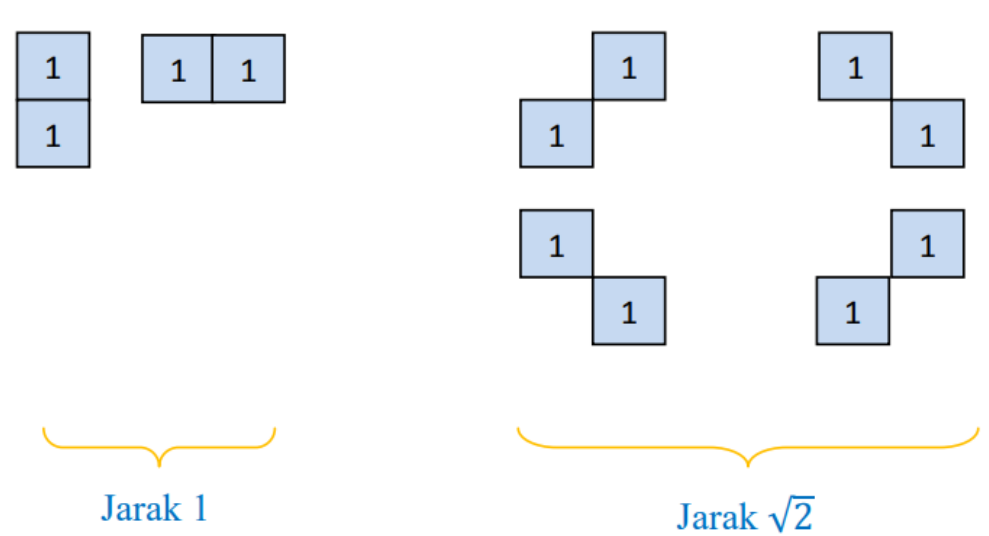
PERIMETER

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



Gambar 11.13 Perimeter dan luas daun

Menurut Costa & Cesar, tahun 2001, algoritma estimasi perimeter memberikan hasil yang baik ketika tepi objek terhubung dengan 4-ketetanggaan, tetapi tidak tepat kalau terhubung menurut 8-ketetanggaan.



Gambar 11.14 Jarak antarpiksel pada 8-ketetanggaan

Hal itu terjadi karena pada 8 ketetanggaan, jarak antara dua piksel tidak bersifat konstan (dapat berupa 1 atau √2), sedangkan pada 4 ketetanggaan jarak selalu 1 .

Ilustrasi mengenai jarak antarpiksel dapat dilihat pada Gambar 11.14.

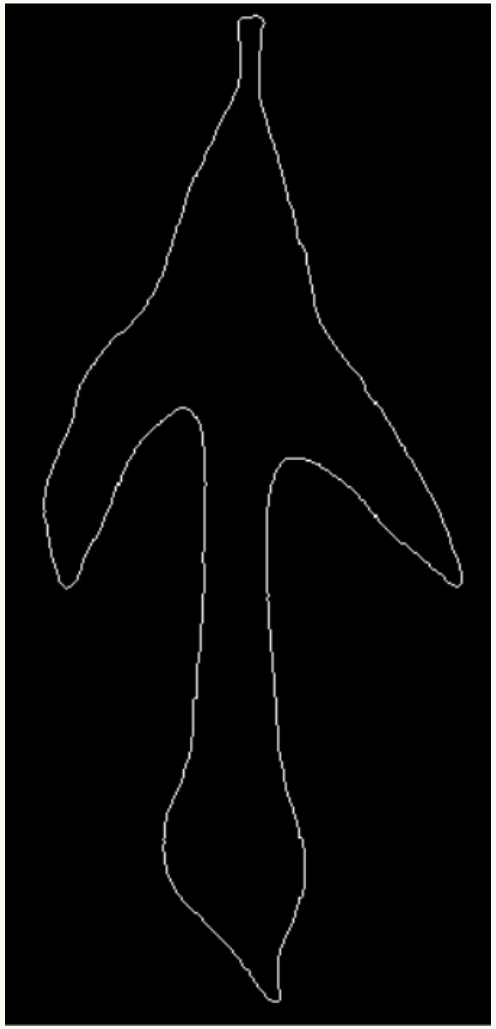
```
function hasil = perim1(BW)
% PERIM1 Untuk menghitung perimeter
% suatu objek pada BW (citra biner)
% hasil menyatakan hasil perhitungan perimeter

U = inbound_tracing(BW);
hasil = length(U) - 1;
```

Catatan:

- Pada skrip di atas, -1 diberikan mengingat elemen pertama dan terakhir U sebenarnya berisi nilai yang sama.
- Itulah sebabnya, jumlah piksel pada kontur perlu dikurangi satu.

Contoh pengujian fungsi perim1:

```
>> Img = imread('C:\Image\daun_bin.png'); ⏎
>> perim1(Img) ⏎
ans = 1409
>>
```



Gambar 11.15 Gambar file daun_bin.png yang diambil perimeternya

LUAS

Cara sederhana untuk menghitung luas suatu objek, adalah dengan cara menghitung jumlah piksel pada objek tersebut. Hitunglah luas pada obyek contoh dibawah:

```
>> 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

Sebagai contoh, hitunglah luas pada obyek dibawah:

```
>> 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 ];
```

```
function hasil = luas(BW)
% LUAS Untuk menghitung luas citra BW (citra biner)

[tinggi, lebar] = size(BW);
hasil = 0;
for p = 1 : tinggi
    for q = 1 : lebar
        if BW(p, q) == 1
        hasil = hasil + 1;
        end
    end
end
```

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

>>

DIAMETER

Diameter adalah jarak terpanjang antara dua titik dalam tepi objek. Hal ini dapat dihitung dengan menggunakan metode "**Brute force**" (Costa dan Cesar, 2001).

Brute force adalah sebuah pendekatan memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (*straight forward*), dengan cara membandingkan, mencocokan karakter per karakter patern dengan karakter yang sesuai.

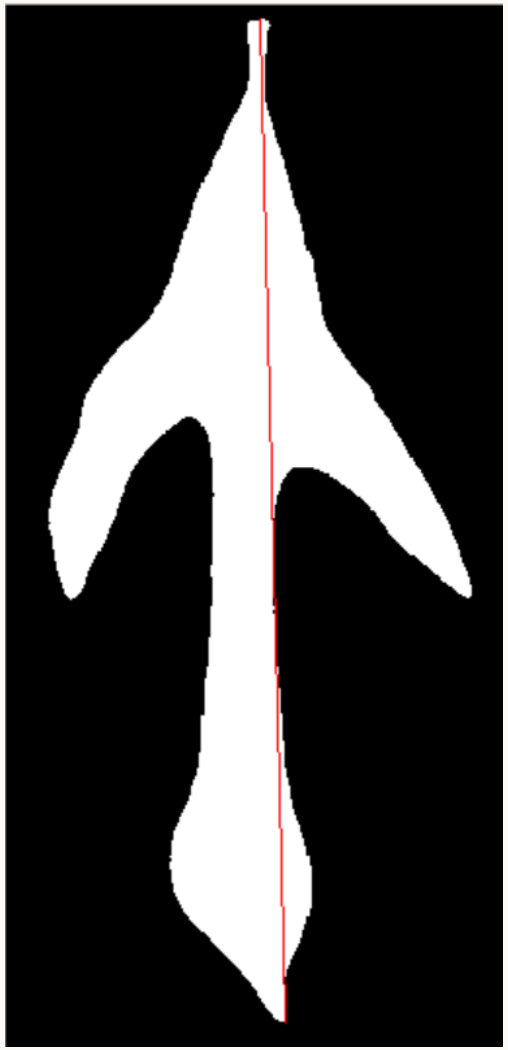
```
function [diameter, x1, y1, x2, y2] = peroleh_diameter(BW)
% PEROLEH_DIAMETER Digunakan untuk menghitung panjang objek
% pada citra BW (citra biner).
% Hasil:
% diameter : panjang objek
% x1, y1, x2, y2 : menyatakan dua titik yang
% mewakili panjang tersebut
U = get_contour(BW);
n = length(U);
jarak_maks = 0;
piksel1 = 0;
piksel2 = 0;
```

```
for p=1 : n-1
    for q=p+1 : n
        jarak = sqrt((U(p,1)-U(q,1)) ^ 2 + (U(p,2)-U(q,2)) ^ 2);
        if jarak > jarak_maks
            jarak_maks = jarak;
            piksel1 = p;
            piksel2 = q;
        end
    end
end
y1 = U(piksel1, 1);
x1 = U(piksel1, 2);
y2 = U(piksel2, 1);
x2 = U(piksel2, 2);
diameter = jarak_maks;
```

Contoh penggunaan fungsi peroleh_diameter diberikan di bawah ini:


```
>> Daun = imread('c:\image\daun_bin.png'); ⏎
>> [d,x1,y1,x2,y2] = peroleh_diameter(Daun); ⏎
>> d ⏎
d = 515.1641
>> X = [x1,x2] ⏎
X =

 144 131

>> Y = [y1,y2]; ⏎
>> line(X,Y, 'Color','r') ⏎
>>
```

Hasil dalam bentuk gambar diperlihatkan pada Gambar 11.17.



Gambar 11.17 Garis merah menyatakan diameter daun

PUSAT MASSA DAN FITUR MENGGUNAKAN PUSAT MASSA

Pusat massa atau sentroid (*centroid*) lazim ditemukan dengan menggunakan nilai rerata koordinat setiap piksel yang menyusun objek.

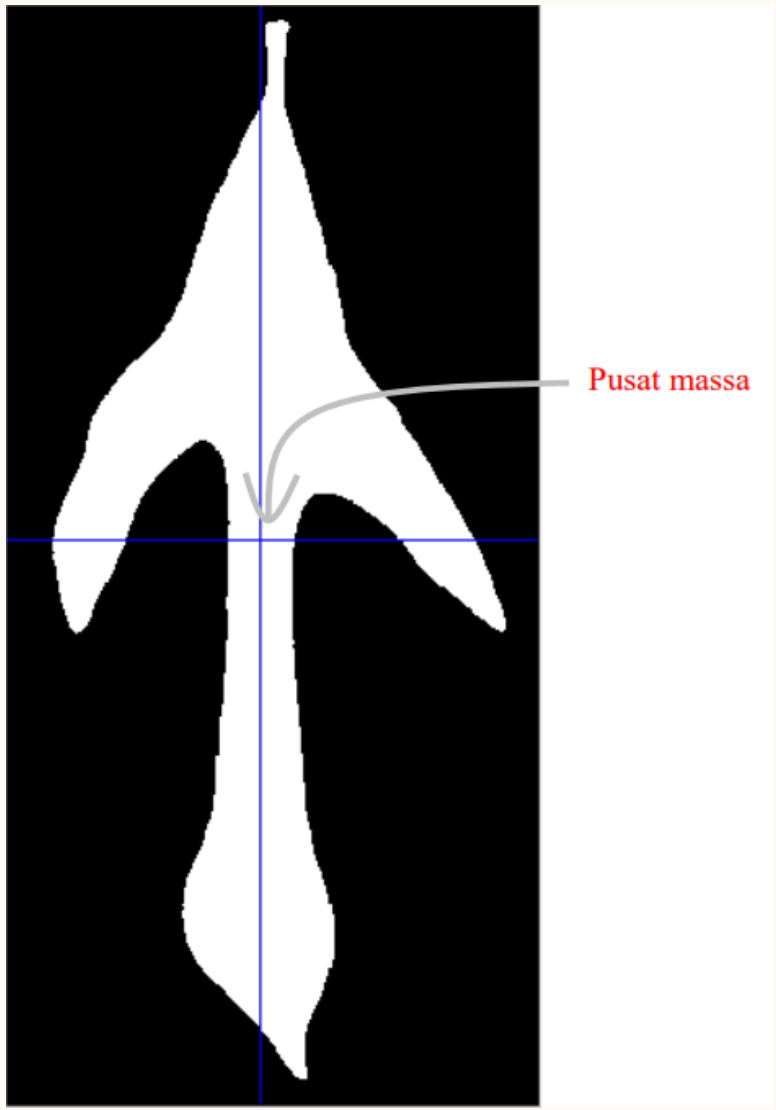
```
function [pusat_x, pusat_y] = centroid(BW)
% CENTROID Untuk memperoleh pusat massa sebuah objek
% yang terletak pada citra biner BW
[tinggi, lebar] = size(BW);
pusat_x = 0;
pusat_y = 0;
luas = 0;
for q = 1 : tinggi
 for p = 1 : lebar
```

```
if BW(q, p) == 1
 luas = luas + 1;
 pusat_x = pusat_x + p;
 pusat_y = pusat_y + q;
 end
 end
end
pusat_x = pusat_x / luas;
pusat_y = pusat_y / luas;
```

Contoh penggunaan fungsi centroid:


```
>> Daun = imread('C:\Image\daun_bin.png'); ⏎
>> [x, y] = centroid(Daun); ⏎
>> imshow(Daun); ⏎
>> [panjang, lebar] = size(Daun); ⏎
>> line([0 lebar], [round(y) round(y)],'Color','b') ⏎   % Buat garis tegak, berwarna biru.
>> line([round(x) round(x)], [0 panjang],'Color','b') ⏎  % Buat garis datar, berwarna biru.
>>
```

Hasil dalam bentuk gambar diperlihatkan pada Gambar 11.18.



Gambar 11.18 Contoh untuk menunjukkan centroid

Pada contoh di atas, **line** digunakan untuk membuat garis tegak dan garis datar, yang melewati pusat massa dan berwarna biru. Hasilnya dapat dilihat pada Gambar 11.18

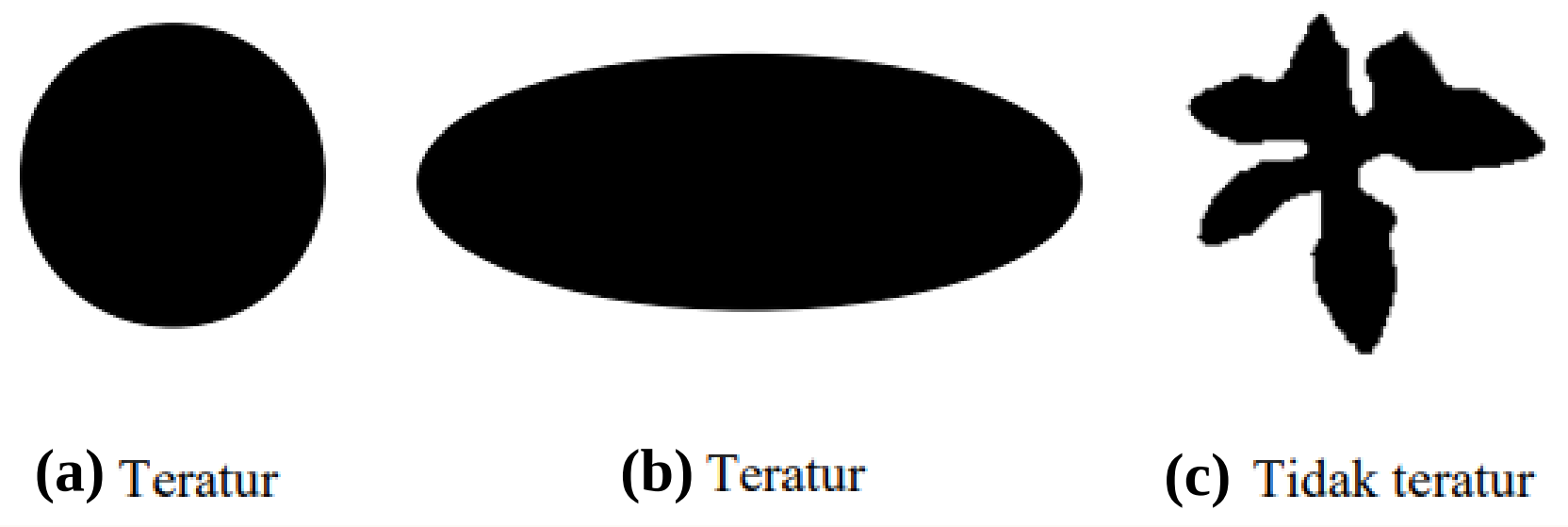
FITUR DISPRESI

Untuk bentuk yang tidak teratur (atau biasa disebut bentuk tidak kompak), Nixon dan Aguado (2002) menyarankan penggunaan fitur dispersi.

Berdasarkan definisi Chen di tahun 1995 (Nixon dan Aguado, 2002), Mereka menyarankan penggunaan dispersi pada bentuk yang tidak teratur, karena dispersi sangat tepat untuk bentuk seperti itu.

FITUR DISPRESI

Sebagai contoh, terdapat tiga bentuk seperti terlihat pada Gambar 11.19. Dispersi (atau juga disebut tidak teraturan) diukur sebagai perbandingan panjang chord utama terhadap area objek.



Gambar 11.19 Kekompakan objek pada berbagai bentuk

Penggunaan kekompakan bentuk untuk objek pada Gambar 11.19(c) sebagai diskriminator tidak tepat.

```
function [d1, d2] = dispersi(BW)
% DISPERSI Contoh untuk menguji beberapa fitur yang
% menggunakan pusat massa. BW = Citra biner
[px, py] = centroid(BW);
U = inbound_tracing(BW);
U(length(U),:) = [];                    % Hapus elemen terakhir
rerata = 0;
terkecil = 99999999;
terbesar = 0;
jum_piksel = length(U);
for j = 1 : jum_piksel
    panjang = sqrt((U(j,1)-py)^2 + (U(j,2)-px)^2);
    rerata = rerata + panjang;
```

```
        if panjang > terbesar
            terbesar = panjang;
        end

        if panjang < terkecil
            terkecil = panjang;
        end
    end
    a = perim2(BW);
    d1 = pi * terbesar / a;
    d2 = terbesar / terkecil;
```

Contoh penggunaan fungsi dispersi :


```
>> Daun1 = imread('C:\Image\adv.png'); ⏎
>> Daun2 = imread('C:\Image\aw.png'); ⏎
>> [d1, d2] = dispersi(Daun1) ⏎

d1 = 0.78285
d2 = 7.7401

>> [d1, d2] = dispersi(Daun2) ⏎

d1 = 0.57999
d2 = 1.7292 >> ⏎
>>
```

TERIMAKASIH

Ada yang ditanyakan?