

Segmentasi Citra

PCD-10

- Segmentasi citra merupakan proses yang ditujukan untuk mendapatkan objek-objek yang terkandung di dalam citra atau membagi citra ke dalam beberapa daerah dengan setiap objek atau daerah memiliki kemiripan atribut
- Pada citra yang mengandung hanya satu objek, objek dibedakan dari latarbelakangnya.



(b) Citra daun



(a) Hasil segmentasi dalam bentuk biner

Pemisahan objek daun terhadap latarbelakang

- Contoh penerapan segmentasi yaitu untuk membuat “Magic Wand”, yang biasa terdapat pada perangkat pengedit foto seperti Adobe Photoshop



Pemilihan citra berdasarkan warna,
yang intinya diperoleh melalui segmentasi.
Bagian terpilih ditandai dengan garis terputus-putus

- segmentasi dilakukan untuk mendapatkan objek yang menjadi perhatian.
- Segmentasi juga biasa dilakukan sebagai langkah awal untuk melaksanakan klasifikasi objek

Objek	Citra	Kegunaan Segmentasi	Acuan yang Digunakan
Mobil	Mobil, jalan, dan latarbelakang	Pelacakan mobil	Gerakan dan warna
Struktur permukaan bumi	Foto satelit	Pengklasifikasian area	Tekstur dan warna
Wajah orang	Kerumunan orang di pasar	Pengenalan wajah	Warna, bentuk, dan tekstur
Apel	Kumpulan apel pada ban berjalan	Pemilahan buah apel berdasarkan ukuran	Bentuk, warna, ukuran

- Setelah segmentasi citra dilaksanakan, fitur yang terdapat pada objek diambil.
- Fitur objek dapat berupa perbandingan lebar dan panjang objek, warna rata-rata objek, atau bahkan tekstur pada objek. Selanjutnya, melalui pengklasifikasi, jenis objek dapat ditentukan.
- Sebagai contoh, pengklasifikasi menyatakan bahwa daun termasuk golongan *Aglaonema*.



Citra masukan

Segmentasi
Citra

Objek daun

Ekstraksi
Fitur

*Fitur-fitur
pada daun*

Pengklasifikasi

Jenis tanaman

**Segmentasi sebagai langkah awal
sistem klasifikasi**

Deteksi Garis

- Deteksi garis pada citra dapat diperoleh melalui penggunaan *mask*

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$

(a)

$$\begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$$

(b)

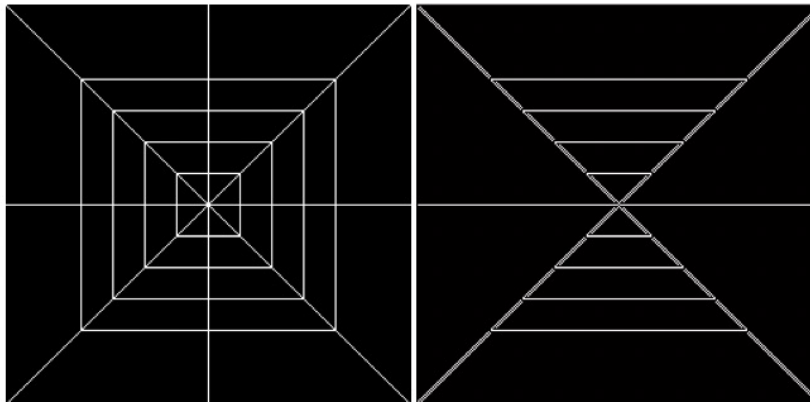
$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

(c)

$$\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

(d)

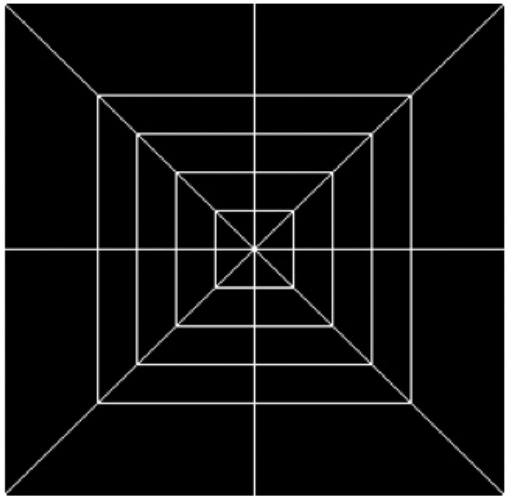
- Mask (a) berguna untuk memperoleh garis horizontal, Mask (b) untuk mendapatkan garis yang berorientasi 45°, Mask (c) untuk memperoleh garis tegak, dan Mask (d) untuk mendapatkan garis yang berorientasi -45°.



(a) Citra jaring.png

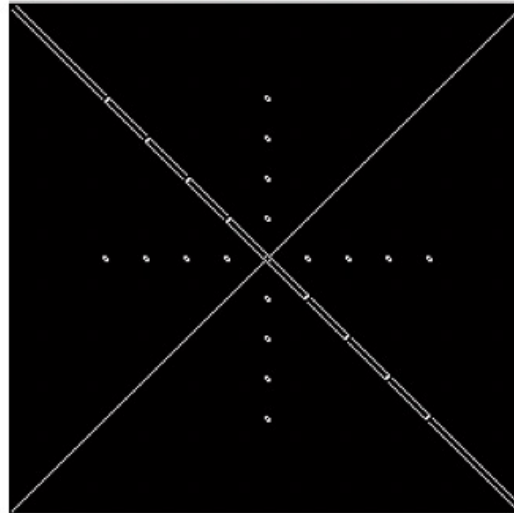
(b) Hasil deteksi garis horizontal

Contoh deteksi garis horizontal

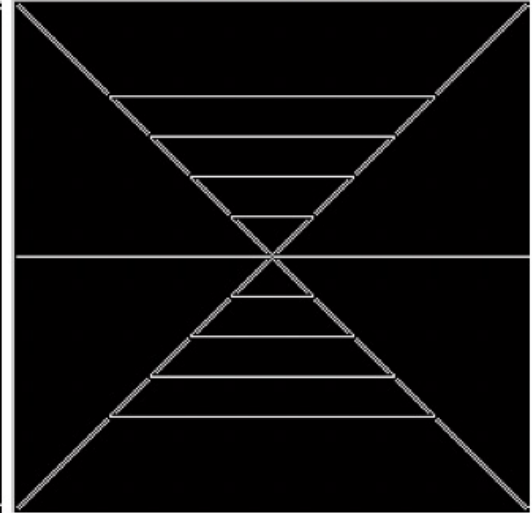


(a) Citra jaring.png

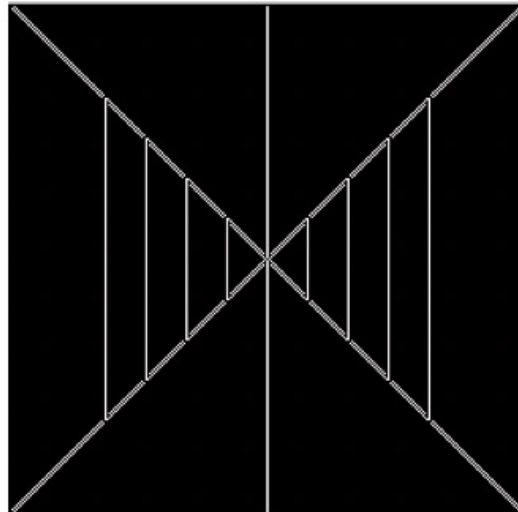
$$\begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \Rightarrow$$



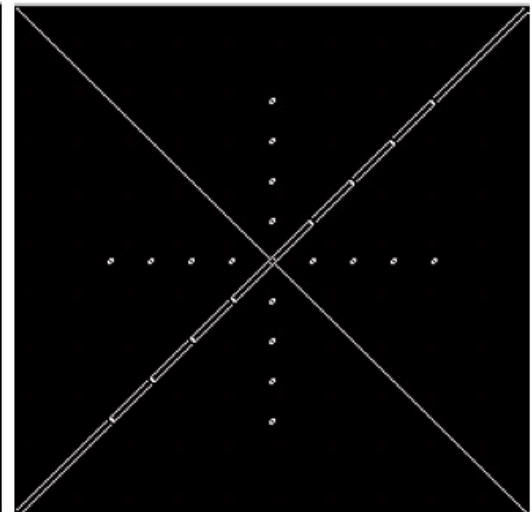
$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \Rightarrow$$



**Contoh hasil deteksi
garis untuk empat arah
yang berbeda**



$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \Rightarrow$$



$$\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \Rightarrow$$

Deteksi Tepi

- Deteksi tepi berfungsi untuk memperoleh tepi objek.
- Deteksi tepi memanfaatkan perubahan nilai intensitas yang drastis pada batas dua area
- Tepi obyek sesungguhnya mengandung informasi yang sangat penting. Informasi yang diperoleh dapat berupa bentuk maupun ukuran objek.
- contoh yang tergolong jenis pendeteksi Tepi adalaah operator *Roberts*, *Prewitt*, *Sobel*, dan *Frei-Chen*.

Deteksi tepi dapat dibagi menjadi dua golongan

- Golongan pertama disebut deteksi tepi orde pertama, yang bekerja dengan menggunakan turunan atau diferensial orde pertama. Termasuk kelompok ini adalah operator *Roberts*, *Prewitt*, dan *Sobel*.
- Golongan kedua dinamakan deteksi tepi orde kedua, yang menggunakan turunan orde kedua. Contoh yang termasuk kelompok ini adalah *Laplacian of Gaussian* (LoG).

Operator Roberts

- Operator *Roberts* terdiri atas dua filter berukuran 2×2 .
- Ukuran filter yang kecil membuat komputasi sangat cepat. Namun, kelebihan ini sekaligus menimbulkan kelemahan, yakni sangat terpengaruh oleh derau.
- Selain itu, operator *Roberts* memberikan tanggapan yang lemah terhadap tepi, kecuali kalau tepi sangat tajam

	x	x+1
y	z_1	z_2
y+1	z_3	z_4

(a) Posisi pada citra f

1	0
0	-1

(b) G_x

0	-1
1	0

(c) G_y

Operator *Roberts* (b) dan (c) serta posisi pada citra f

Misalkan, f adalah citra yang akan dikenai operator *Roberts*. Maka, nilai operator *Roberts* pada (y, x) didefinisikan sebagai

$$r(y, x) = \sqrt{(z_1 - z_4)^2 + (z_3 - z_2)^2}$$

Dalam hal ini, $z_1 = f(y, x)$, $z_2 = f(y, x+1)$, $z_3 = f(y+1, x)$, dan $z_4 = f(y+1, x+1)$.

- function [G] = roberts(F)
- % ROBERTS Pemerolehan tepi objek pada citra F
- % melalui operator Roberts
- % Hasil: citra G
-
- [m, n] = size(F);
-
- F=double(F);
- for y=1 : m-1
- for x=1 : n-1
-
- $G(y, x) = \sqrt{(F(y, x) - F(y+1, x+1))^2 + \dots}$
- $(F(y+1, x) - F(y, x+1))^2)$;
- end
- end
-

Pengenalan operator *Roberts* pada citra

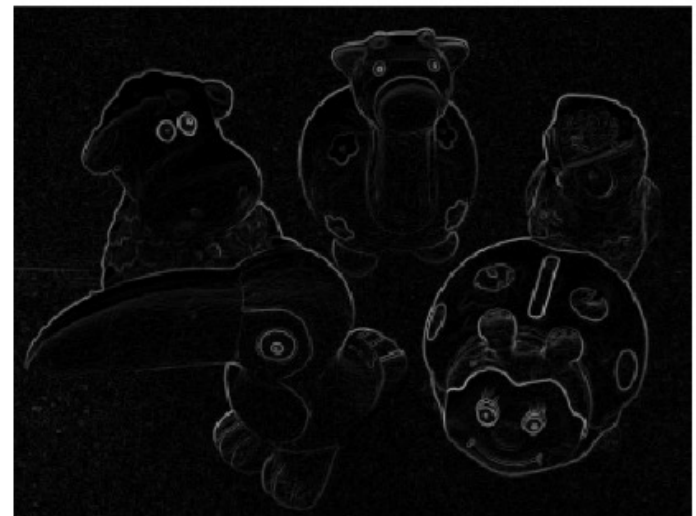
mainan.png

Contoh penggunaan fungsi
`roberts`:

```
>>                               Img                               =  
rgb2gray(imread('C:\Image\ma  
inan.png')) ;  
  
>> G = roberts(Img) ;  
  
>> imshow(G)
```



(a) Citra `mainan.png`



(b) Hasil deteksi tepi dengan operator *Roberts*

Operator Prewitt

- Operator *Prewitt* → Untuk mempercepat komputasi, bagian yang bernilai nol tidak perlu diproses. Oleh karena itu, perhitungan dengan operator *Prewitt* ditulis menjadi

$$r(y, x) = \text{sqrt}((f(y-1, x-1) + f(y, x-1) + f(y+1, x-1) - f(y-1, x+1) - f(y, x+1) - f(y+1, x+1))^2 + (f(y+1, x-1) + f(y+1, x) + f(y+1, x+1) - f(y-1, x-1) - f(y-1, x) - f(y-1, x+1))^2))$$

	x-1	x	x+1
y-1	z ₁	z ₂	z ₃
y	z ₄	z ₅	z ₆
y+1	z ₇	z ₈	z ₉

(a) Posisi pada citra f

1	0	-1
1	0	-1
1	0	-1

(b) G_x

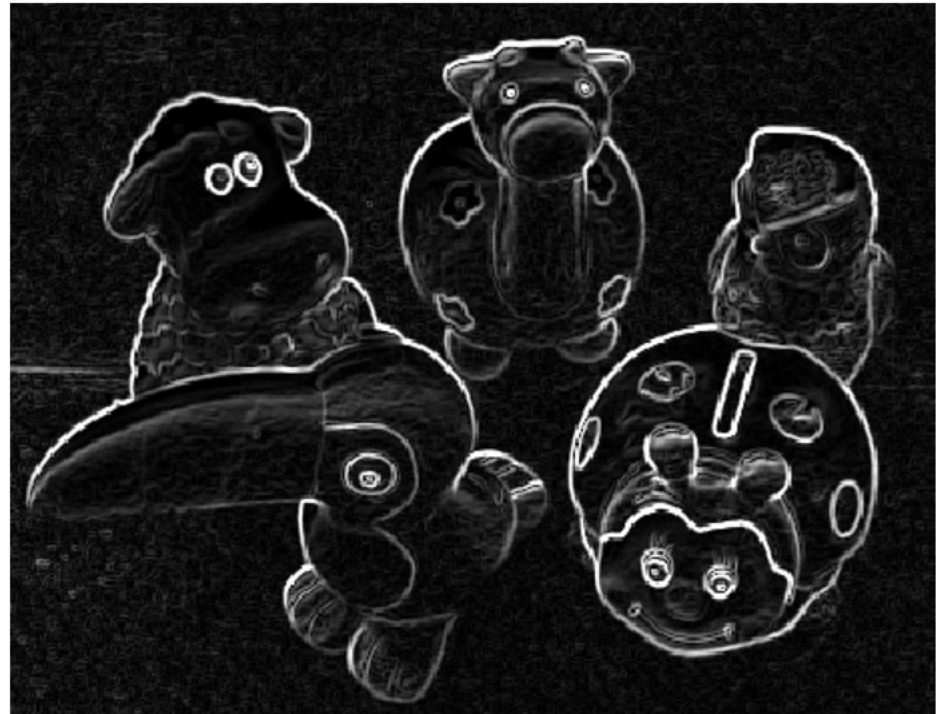
-1	-1	-1
0	0	0
1	1	1

(c) G_y

- function [G] = prewitt(F)
- % PREWITT Pemerolehan tepi objek pada citra F
- % melalui operator Prewitt
- % Hasil: citra G
-
- [m, n] = size(F);
-
- F=double(F);
- G=zeros(m,n);
- for y=2 : m-1
- for x=2 : n-1
- G(y, x) = sqrt((F(y-1,x-1) + F(y,x-1) + F(y+1,x-
- 1) - ...
- F(y,x) - F(y,x+1) - F(y+1,x+1))^2 + ...
- (F(y+1,x-1)+ F(y+1,x) + F(y+1,x+1) - ...
- F(y-1,x-1) - F(y-1,x) - F(y-1,x+1))^2) ;
- end
- end
-

Contoh penggunaan
fungsi `prewitt`:

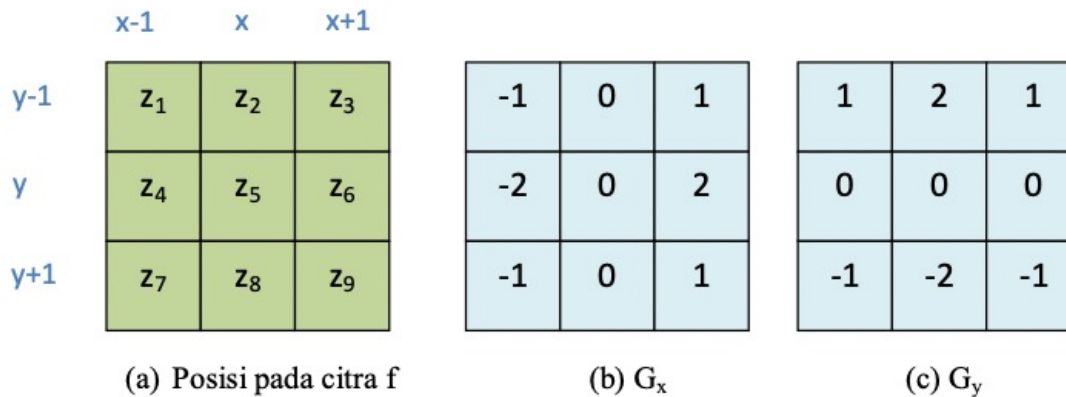
```
>> Img =  
rgb2gray(imread('C:\Image\mainan.png'));  
  
>> G = prewitt(Img);  
  
>> imshow(G)
```



**Hasil deteksi tepi dengan operator *Prewitt*.
Hasil tepi lebih tegas daripada hasil dengan
operator *Roberts***

Operator Sobel

- Operator *Sobel* lebih sensitif terhadap tepi diagonal daripada tepi vertikal dan horizontal



- function [G] = sobel(F)
- % SOBEL Pemerolehan tepi objek pada citra F
- % melalui operator Sobel
- % Hasil: citra G
-
- [m, n] = size(F);
-
- F=double(F);
- G=zeros(m,n);
- for y=2 : m-1
- for x=2 : n-1
- G(y, x) = sqrt(...
- (F(y-1,x+1)+2*F(y,x+1)+F(y+1,x+1) - ...
- F(y-1,x-1)-F(y,x-1)-F(y+1,x-1))^2 + ...
- (F(y-1,x-1)+2*F(y-1,x)+F(y-1,x+1) - ...
- F(y+1,x-1)-2*F(y+1,x)-F(y+1,x+1))^2) ;
- end
- end

Contoh penggunaan fungsi sobel:

```
>> Img =  
rgb2gray(imread('C:\Image\mainan.png'));  
  
>> G = sobel(Img);  
  
>> imshow(G)
```



Operator Frei-Chen

- Operator *Frei-Chen* (kadang disebut operator isotropik)
- Operator ini mirip dengan operator *Sobel*, dengan setiap angka 2 diganti menjadi $\sqrt{2}$.

	x-1	x	x+1
y-1	z_1	z_2	z_3
y	z_4	z_5	z_6
y+1	z_7	z_8	z_9

(a) Posisi pada citra f

-1	0	1
$-\sqrt{2}$	0	$\sqrt{2}$
-1	0	1

(b) G_x

1	$\sqrt{2}$	1
0	0	0
-1	$-\sqrt{2}$	-1

(c) G_y

- function [G] = freichen(F)
- % FREICHEN Pemerolehan tepi objek pada citra F
- % melalui operator Frei-Chen
- % Hasil: citra G
-
- [m, n] = size(F);
-
- akar2 = sqrt(2);
- F=double(F);
- G=zeros(m,n);
- for y=2 : m-1
- for x=2 : n-1
- G(y, x) = sqrt(...
- (F(y-1,x+1)+akar2*F(y,x+1)+F(y+1,x+1) - ...
- F(y-1,x-1)-F(y,x-1)-F(y+1,x-1))^2 + ...
- (F(y-1,x-1)+akar2*F(y-1,x)+F(y-1,x+1) - ...
- F(y+1,x-1)-akar2*F(y+1,x)-F(y+1,x+1))^2) ;
- end
- end

Contoh penggunaan

fungsi freichen:>>

```
Img =
```

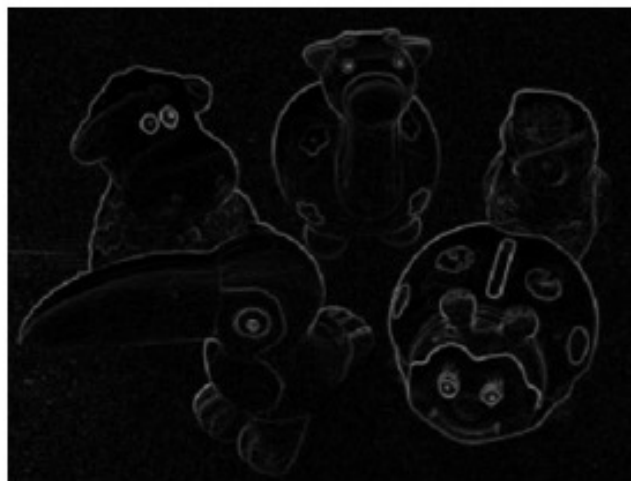
```
rgb2gray(imread('C:\  
Image\mainan.png'));
```

```
>> G =
```

```
freichen(Img);
```

```
>> imshow(G)
```





(a) Operator *Roberts*



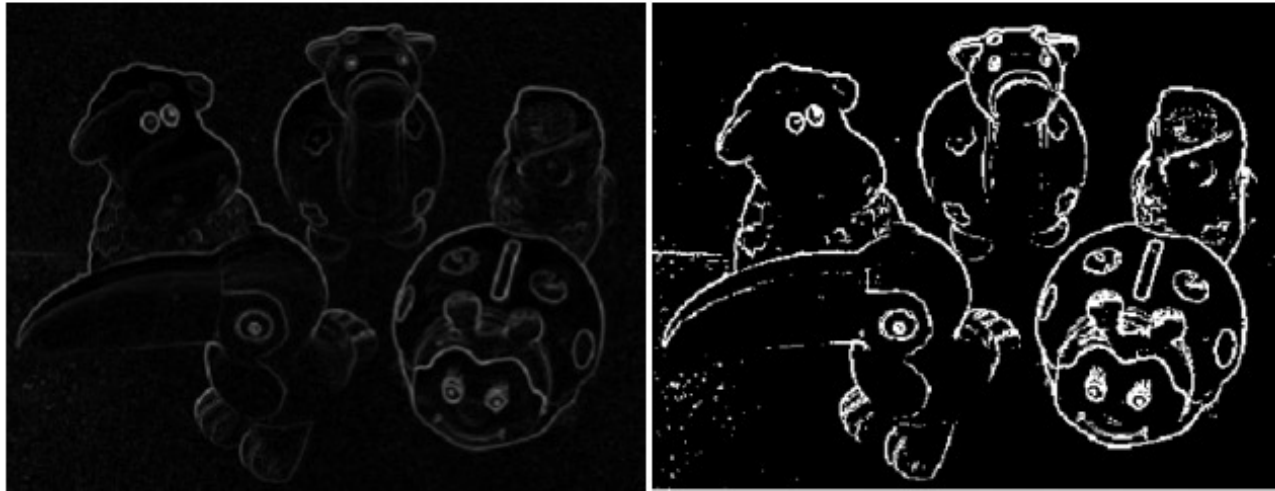
(b) Operator *Prewitt*



(c) Operator *Sobel*



(d) Operator *Frei-Chen*



(a) Hasil deteksi tepi dengan operator *Roberts*

(b) Hasil setelah peng-ambangan dengan nilai ambang 20

Contoh hasil deteksi tepi dan peng-ambangan

Peng-ambangan Intensitas (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
- salah satu area tersebut berkedudukan sebagai latarbelakang.
- Cara seperti itulah yang disebut peng-ambangan dwi-aras (*bi-level thresholding*) atau terkadang dinamakan peng-ambangan intensitas

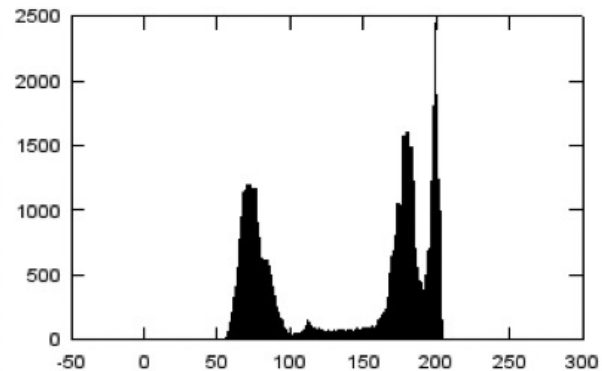
$$b(y, x) = f(x) = \begin{cases} 1, & \text{untuk } (y, x) \geq T \\ 0, & \text{untuk } (y, x) < T \end{cases}$$

T menyatakan ambang intensitas

- 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.

Contoh peng-ambangan dengan nilai ambang sebesar 100:

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



(a) ipomoea.tif



(b) Hasil pengambangan

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

- 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

Thresholding Global dan Lokal

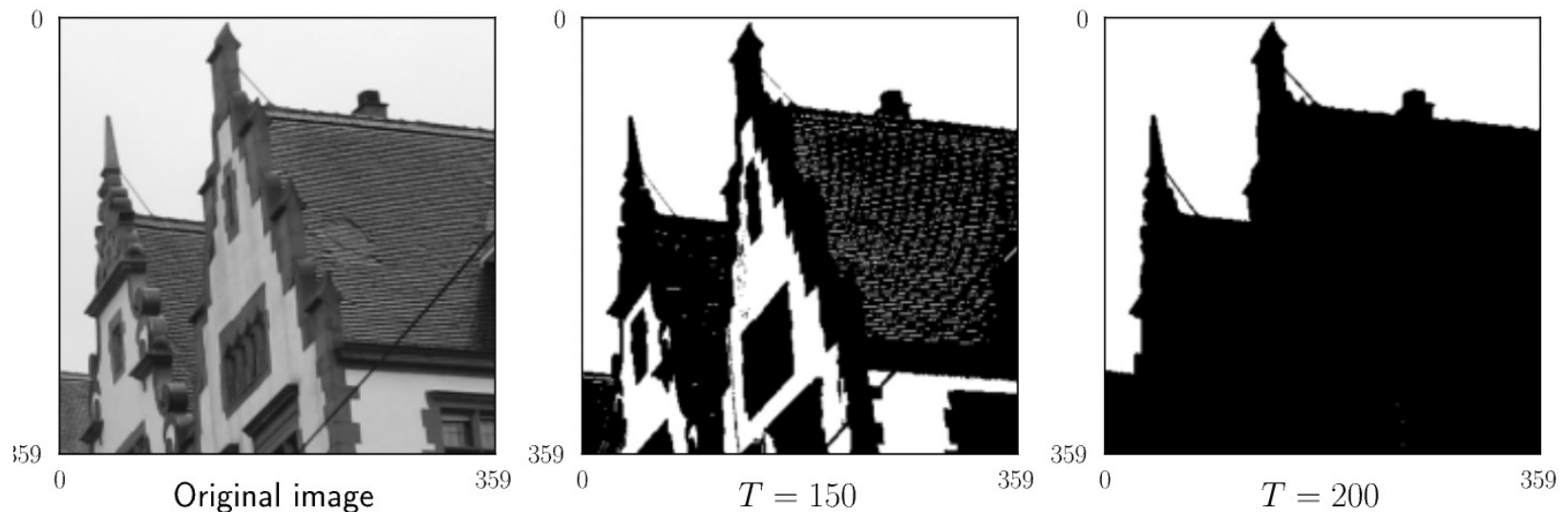
- Terkait dengan nilai ambang yang digunakan pada segmentasi citra, terdapat istilah peng-ambangan global dan peng-ambangan lokal. Pengertiannya sebagai berikut.
- ❖ Apabila nilai ambang t bergantung hanya pada satu nilai keabuan $f(y, x)$, peng-ambangan disebut sebagai global. Dalam hal ini, semua piksel dalam citra akan ditentukan oleh satu nilai ambang t .
- ❖ Peng-ambangan disebut lokal kalau nilai ambang t bergantung pada $f(y, x)$ dan $g(y, x)$ dengan $g(y, x)$ menyatakan properti citra lokal pada titik (y, x) . Dalam hal ini, properti citra lokal dapat diperoleh melalui statistik (misalnya rerata tetangga di sekitar titik (y, x)). Dengan kata lain, nilai ambang untuk setiap piksel ditentukan oleh nilai piksel tetangga. Dengan demikian, nilai ambang untuk piksel masing-masing belum tentu sama.

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

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

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



segmentasi tergantung pada nilai threshold . Oleh karena itu, histogram sangat berguna untuk memilih ambang batas

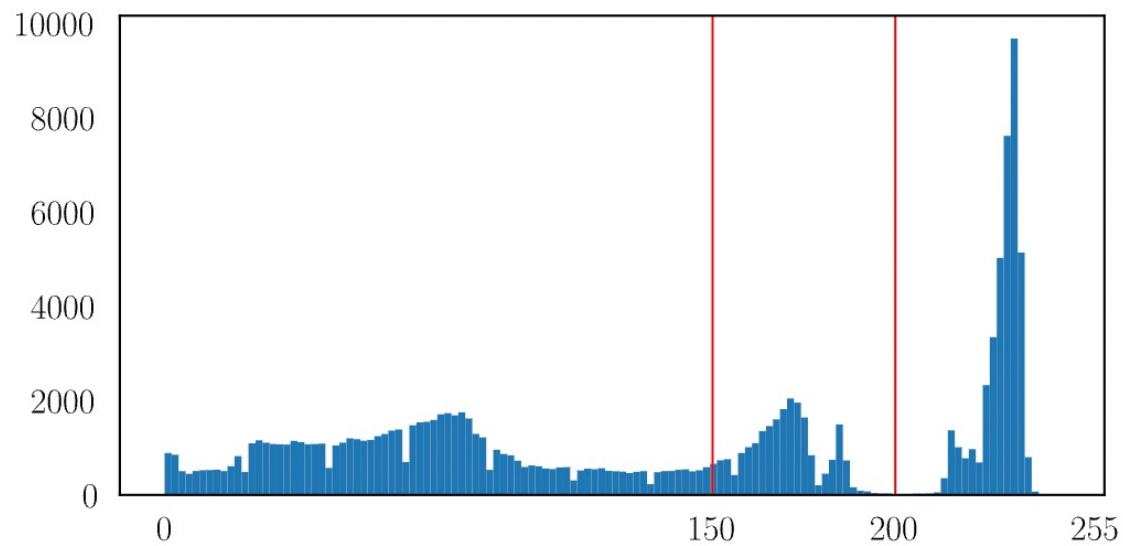

























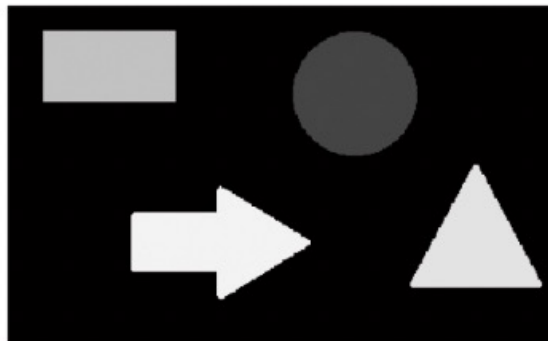


Fig. 43 Histogram of [Fig. 42](#) with the two thresholds.

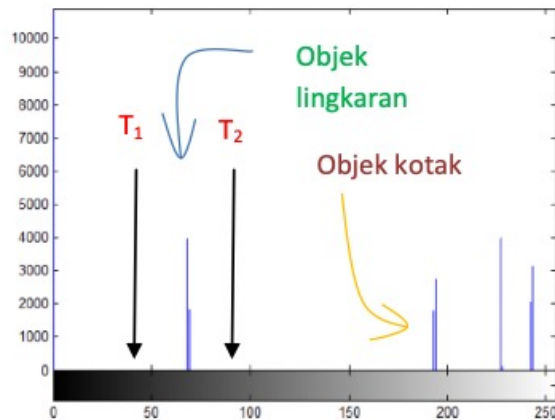
Nama Citra	Citra Asli	T = 50	T = 100	T = 150	T = 200
cameraman					
lena					
mandril					
girl					
bicycle					

Multilevel Thresholding

- 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



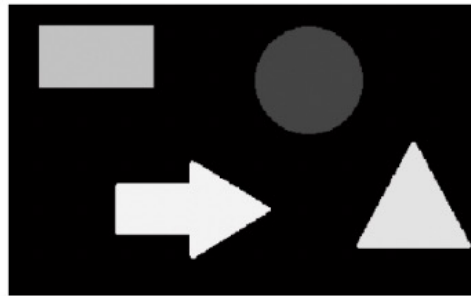
(b) Histogram

Contoh citra dengan beberapa puncak dan lembah yang terpisah

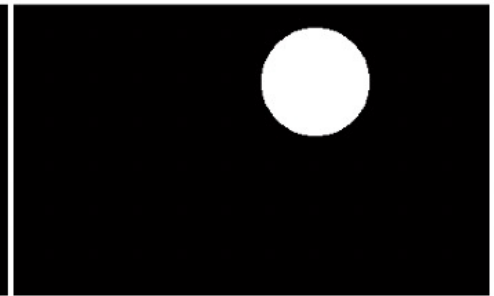
- function [G] = multi_thres(F, t1, t2)
- % Pengambilan dengan dua nilai ambang
- % F = Citra berskala keabuan
- % t1 = nilai ambang bawah
- % t2 = nilai ambang atas
- %
- % Keluaran: G = Citra biner
-
- [m, n] = size(F);
- for i=1 : m
- for j=1:n
- if F(i,j) <= t1 || F(i,j) >= t2
- G(i,j) = 0;
- else
- G(i,j) = 1;
- end
- end
- end

```
>>      Img      =
imread('C:\Image\b
enda.png'); ↵

>> G =
multi_thres(Img, 50,
100); imshow(G);
```



(a) benda.png



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



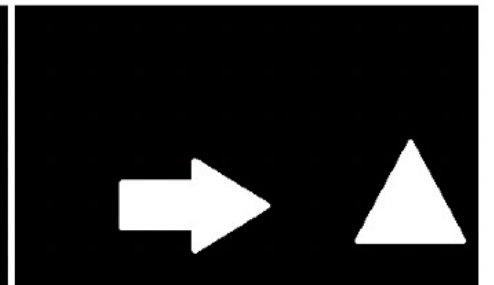
(c) Nilai ambang $t_1=175$ dan $t_2=200$



(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



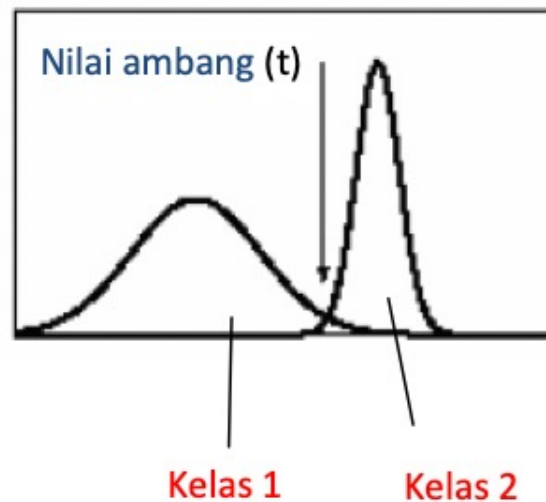
(a) ipomoea.png



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

Otsu Thresholding

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



**Penentuan nilai ambang
untuk memperoleh hasil yang optimal**

- 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.

- Algoritma dari metode Otsu adalah sebagai berikut:
 1. Hitung histogram dari setiap level intensitas
 2. Inisialisasi class probability dan *mean*
 3. Untuk setiap threshold t yang mungkin,
 - a. Update Probability dan mean
 - b. Hitung variansi antarkelas
 4. Threshold yang dipilih adalah yang memberikan nilai variansi antarkelas terbesar.

1. Prinsip metode *Otsu* dijelaskan berikut ini. Pertama-tama, probabilitas nilai intensitas i dalam histogram dihitung melalui

$$p(i) = \frac{n_i}{N}, p(i) \geq 0, \sum_1^{256} p(i) = 1$$

2. dengan n_i menyatakan jumlah piksel berintensitas i dan N menyatakan jumlah semua piksel dalam citra. Jika histogram dibagi menjadi dua kelas (objek dan latarbelakang), pembobotan pada kedua kelas dinyatakan sebagai berikut:

$$w_1(t) = \sum_{i=1}^t p(i)$$

$$w_2(t) = \sum_{i=t+1}^L p(i) = 1 - w_1(t)$$

Dalam hal ini, L menyatakan jumlah aras keabuan. Rerata kedua kelas dihitung melalui:

$$m_1(t) = \sum_{i=1}^t i \cdot p(i) / W_1(t)$$

$$m_2(t) = \sum_{i=1}^t i \cdot p(i) / W_2(t)$$

3. Varians kedua kelas dinyatakan dengan rumus:

$$\sigma_1^2(t) = \sum_{i=1}^t (1 - m_1)^2 \cdot \frac{p(i)}{W_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^L (1 - m_2)^2 \cdot \frac{p(i)}{W_2(t)}$$

Varians total dapat dinyatakan dengan

$$\sigma^2(t) = \sigma_W^2(t) + \sigma_B^2(t)$$

Dalam hal ini, σ_W^2 dinamakan sebagai *within-class variance* (WCV) dan σ_B^2 disebut sebagai *between-class variance* (BCV). WCV dapat dinyatakan dengan

$$\sigma_W^2(t) = W_1(t) \cdot \sigma_1(t)^2 + W_2(t) \cdot \sigma_2(t)^2$$

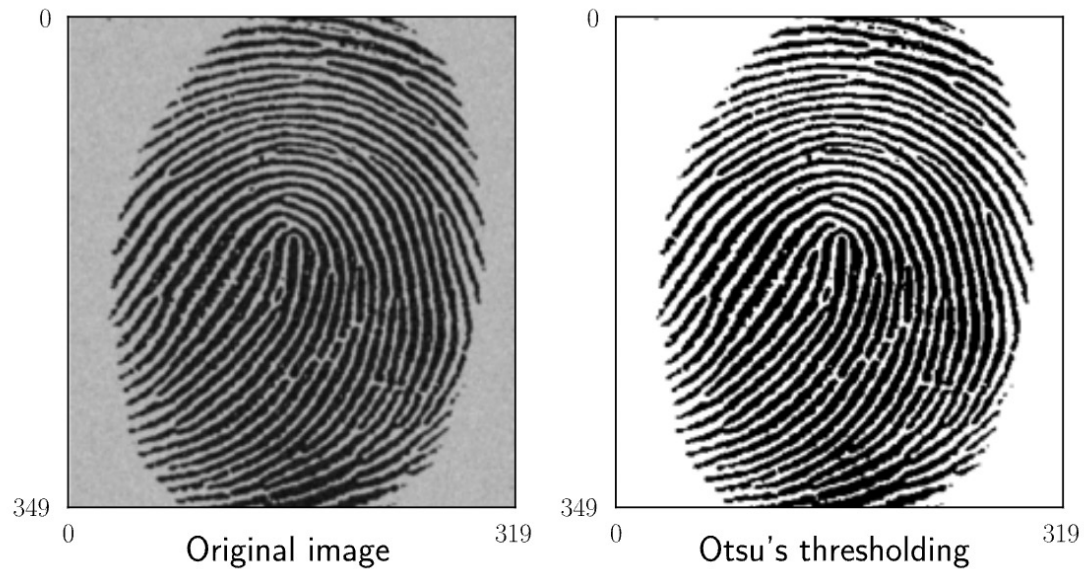
Rumus di atas menunjukkan bahwa WCV adalah jumlah varians kelas secara individual yang telah diboboti dengan probabilitas kelas masing-masing. Adapun BCV dinyatakan dengan

$$\sigma_B^2(t) = W_1 \cdot [m_1(t) - m_T]^2 + W_2 \cdot [m_2(t) - m_T]^2$$

Dalam hal ini, m_T adalah rerata total ($m_T = \sum_{i=1}^N i \cdot p(i)$).

4. Nilai ambang optimum dapat diperoleh dengan dua cara. Cara pertama dilaksanakan dengan meminimumkan WCV. Cara kedua dilaksanakan dengan memaksimumkan BCV. Namun, berdasarkan kedua cara tersebut, cara yang kedua lebih menghemat komputasi.

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



TUGAS

Gunakan dataset yang telah anda punya:

1. Lakukan deteksi tepi : sobel, prewit, Robert dan Laplacian of Gaussian → dan silahkan di Analisa hasilnya
2. Lakukan thresholding citra dengan metode Otsu (sebagai Referensi silahkan cari dan baca paper tentang Otus Thresholding)
3. Dokumentasikan tugas anda dalam file pdf dan Tugas dikumpulkan di ilmu