

# Pengenalan Citra Digital

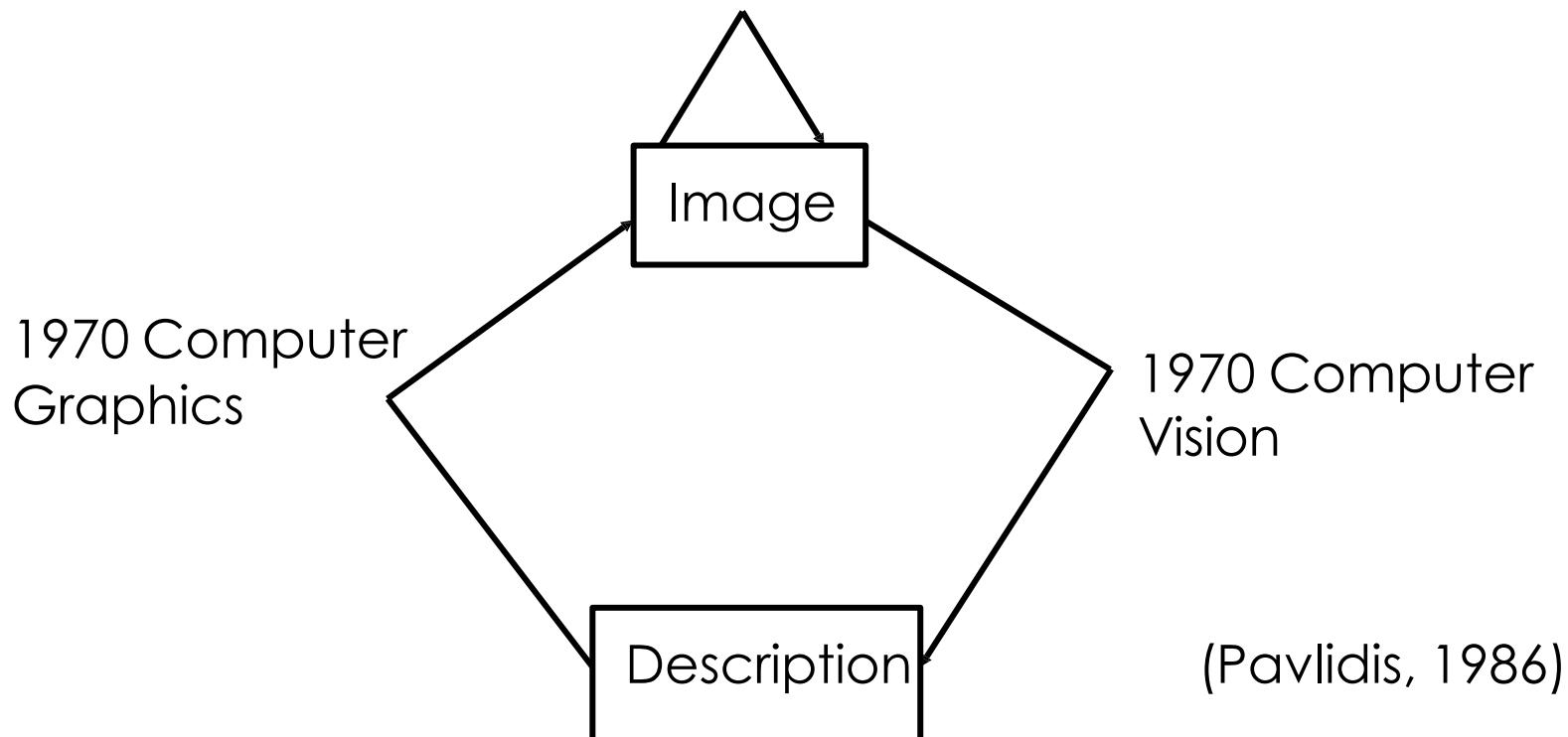


# Pengolahan Citra

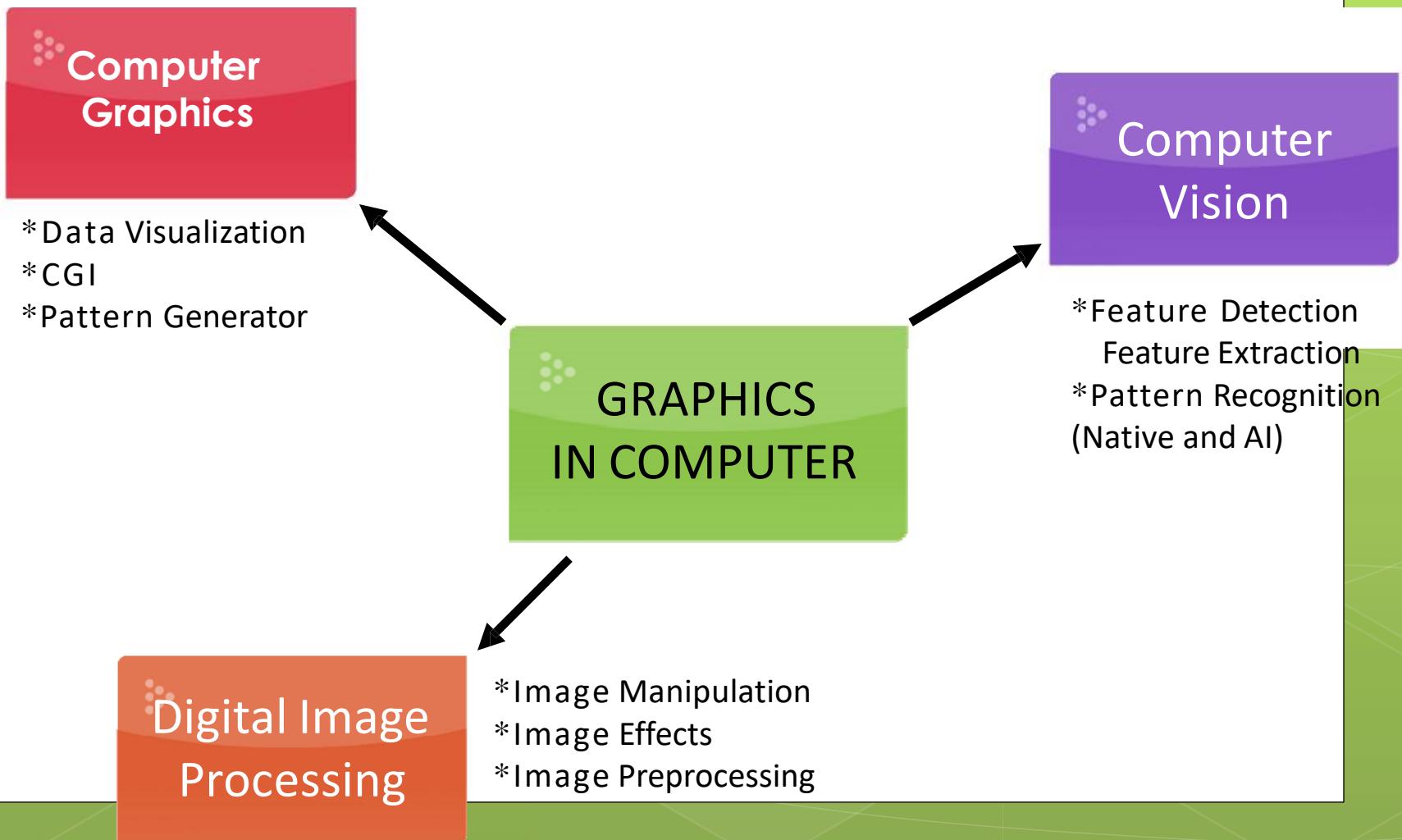
Pengolahan Citra adalah kegiatan memperbaiki kualitas citra agar mudah diinterpretasikan oleh manusia/mesin (komputer). Input adalah citra dan keluarannya adalah citra dengan kualitas lebih baik

# Tiga Bidang Berkaitan dengan Citra

1950 Image Processing



# Tiga Bidang...(lanjutan)



# Hal yang dilakukan di PCD

Image Processing/  
Manipulation



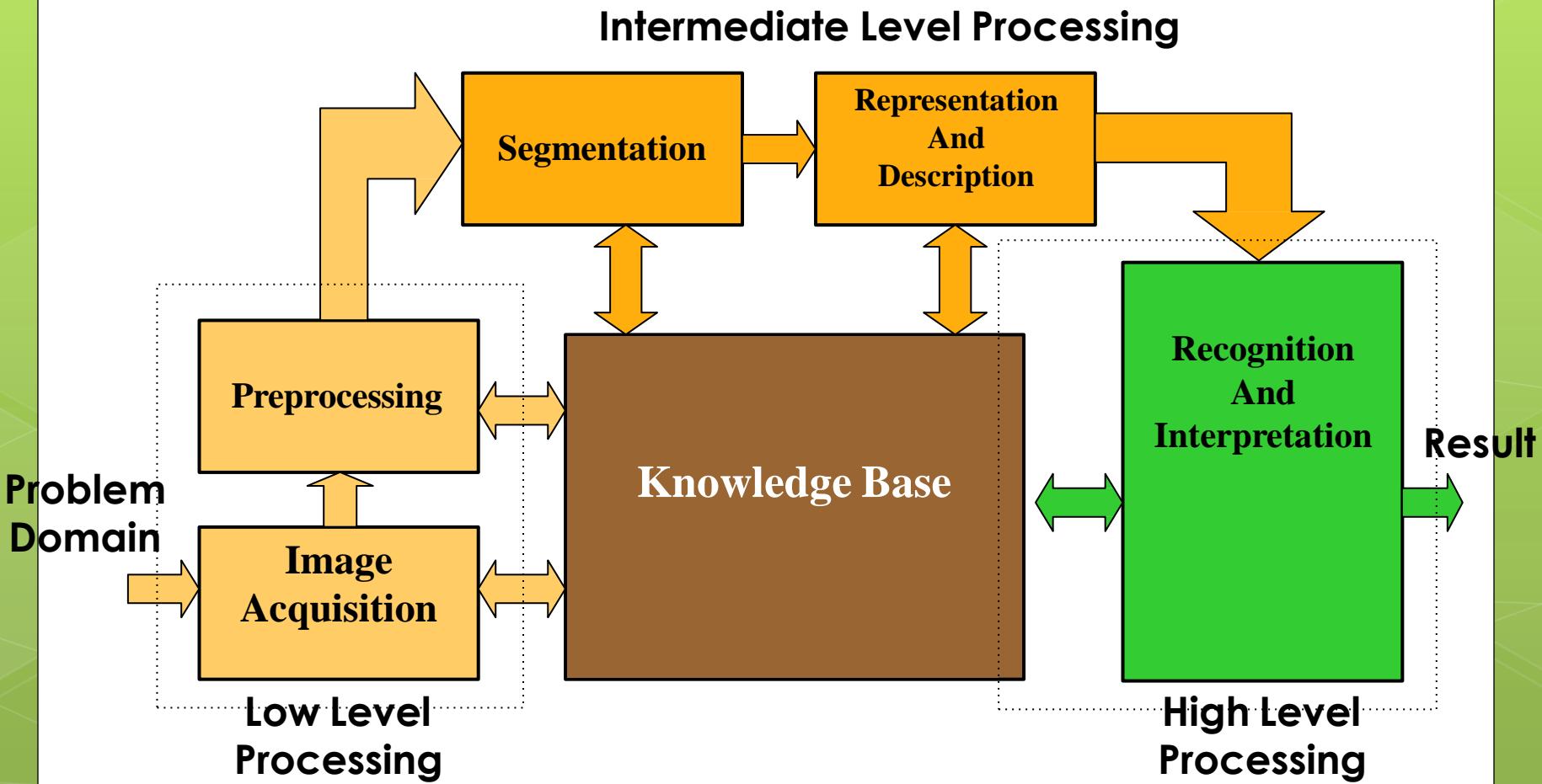
Digital Image  
Processing



Image Analysis/  
Interpretation

Image Coding/  
Communication

# Alur Diagram PCD



# JENIS CITRA

Terdapat 2 jenis citra :

- Citra Kontinu yang diperoleh dari sistem optik yg menerima sinyal analog, seperti mata manusia dan kamera analog
- Citra Diskrit (disebut kemudian dengan citra digital) adalah dihasilkan melalui proses digitalisasi terhadap citra kontinu

# Pengertian Citra Digital

## ■ Citra Digital

- Citra digital merupakan **fungsi intensitas cahaya**  $f(x,y)$ , dimana harga  $x$  dan  $y$  merupakan koordinat spasial dan nilai fungsi pada setiap titik  $(x,y)$  tersebut merupakan tingkat keabuan citra pada titik yang bersangkutan;
- Citra digital adalah citra  $f(x,y)$  dimana dilakukan **diskritisasi koordinat spasial** (sampling) dan **diskritisasi tingkat keabuan** (kuantisasi);
- Citra digital merupakan **suatu matriks** dimana indeks baris dan kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya (yang disebut sebagai elemen gambar / piksel / pixel / picture element / pels) menyatakan tingkat keabuan pada titik tersebut.

- Citra digital dinyatakan dengan matriks NXM

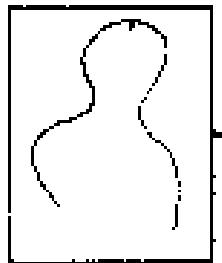
$$N = \text{jumlah baris} \quad 0 \leq x \leq N - 1$$

$$M = \text{jumlah kolom} \quad 0 \leq y \leq M - 1$$

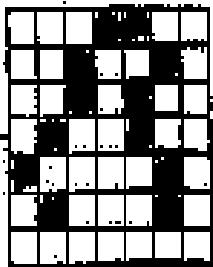
$L$  = maksimal warna intensitas (derajat keabuan/gray level)

$$0 \leq f(x,y) \leq L - 1$$

$$f(x, y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$



Citra kontinue



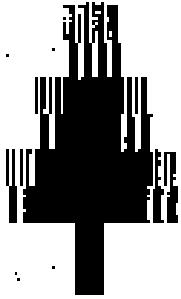
Citra digital



Matriks citra dengan obyek angka 5



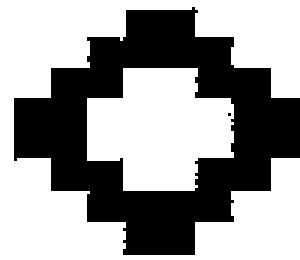
Resolusi spasial :  
Tinggi (16 x 16)



Rendah (8 x 8)



Resolusi keabuan :  
Tinggi (4)



Rendah (2)

# Citra Digital



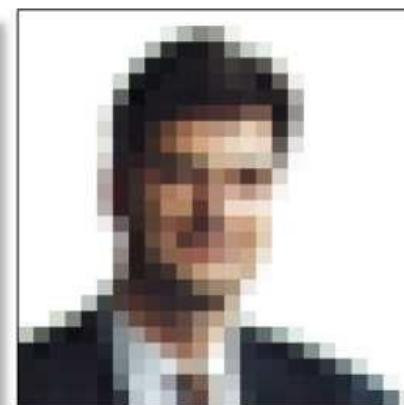
200x200



100x100



50x50



25x25

Citra digital diperoleh dari proses digitalisasi. Ada 2 proses digitalisasi yakni:

1. sampling merupakan proses pengambilan nilai diskrit koordinat ruang (x,y) dengan melewatkannya melalui grid (celah)
2. kuantisasi merupakan proses pengelompokkan nilai tingkat keabuan citra kontinu ke dalam beberapa level atau merupakan proses membagi skala keabuan (0,L) menjadi G buah level yg dinyatakan dengan suatu harga bilangan bulat (integer), dinyatakan sebagai

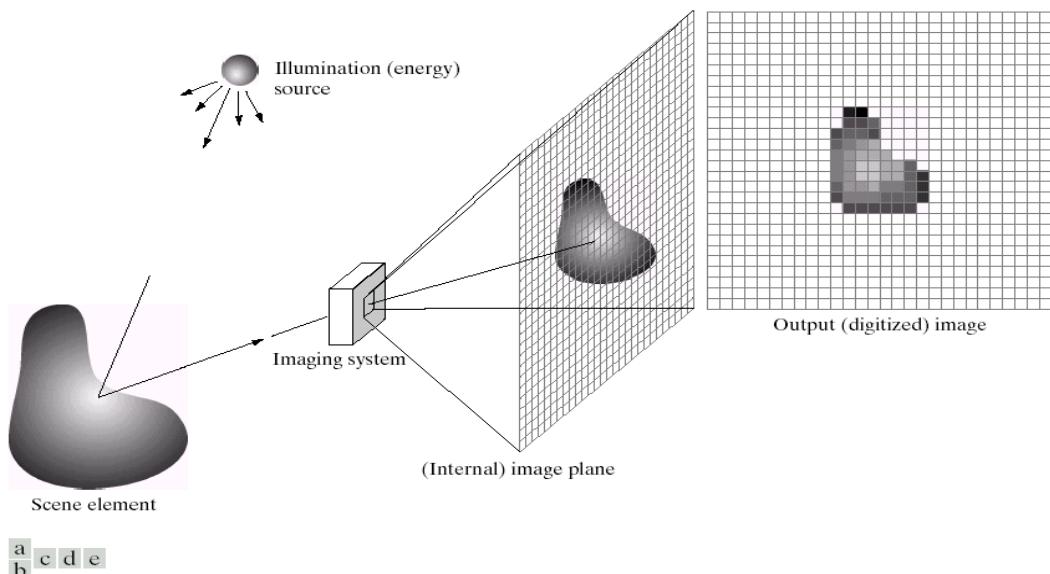
$$G = 2^m$$

G : derajat keabuan,

m : bil bulat positif

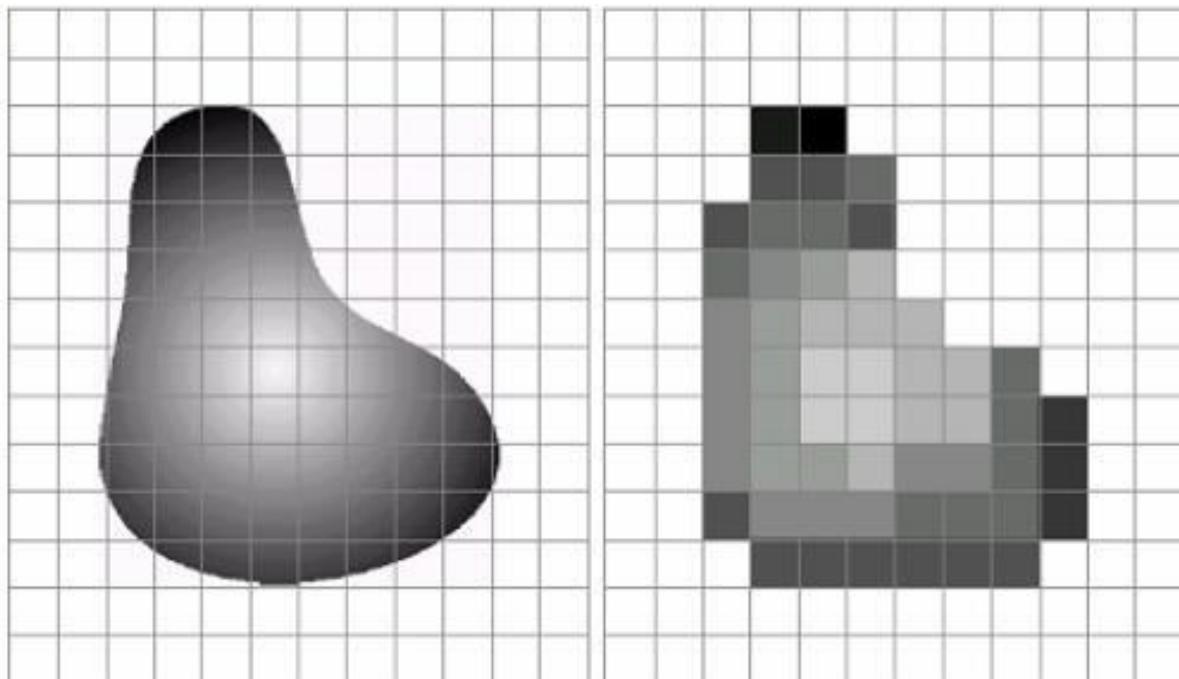
# Image Acquisition / Formation

- Kamera, scanner, sensor (obtained)
- Bitmap drawing software (created)
- Mainstream formats:  
BMP, JPG, PNG,  
GIF, TIFF, RAZ

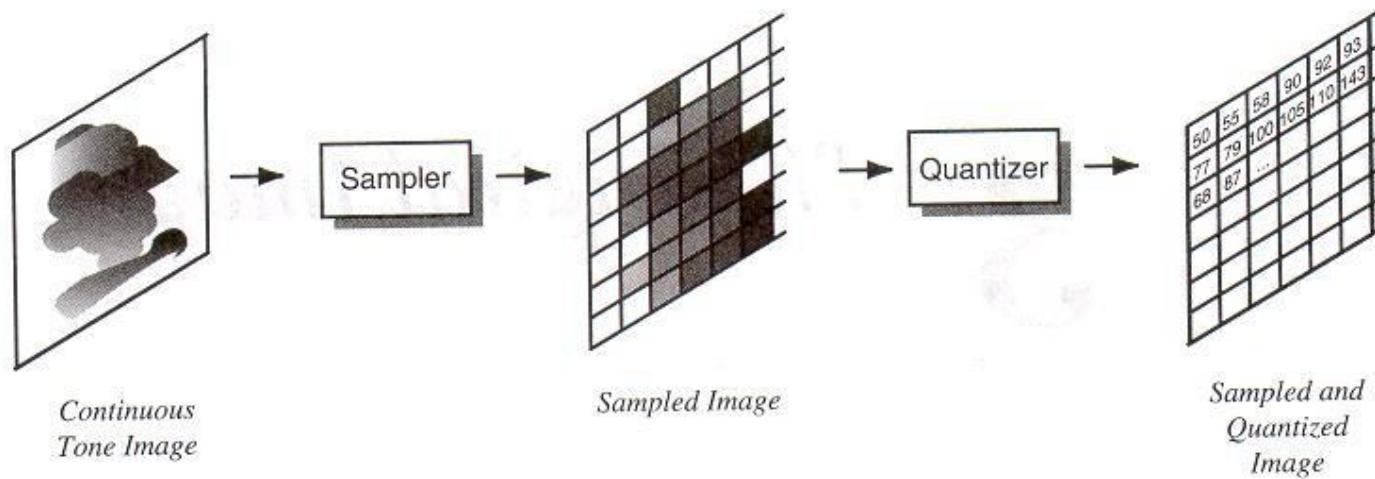


**FIGURE 2.15** An example of the digital image acquisition process. (a) Energy (“illumination”) source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

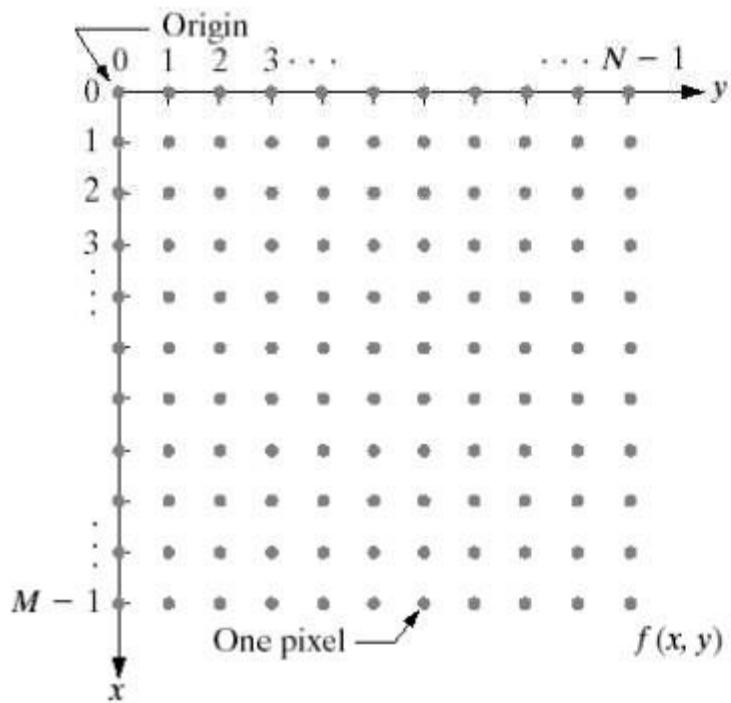
# Image Acquisition / Formation



# Image Acquisition / Formation



# Representasi Citra



**FIGURE 2.18**  
Coordinate  
convention used  
in this book to  
represent digital  
images.

# Representasi Matriks



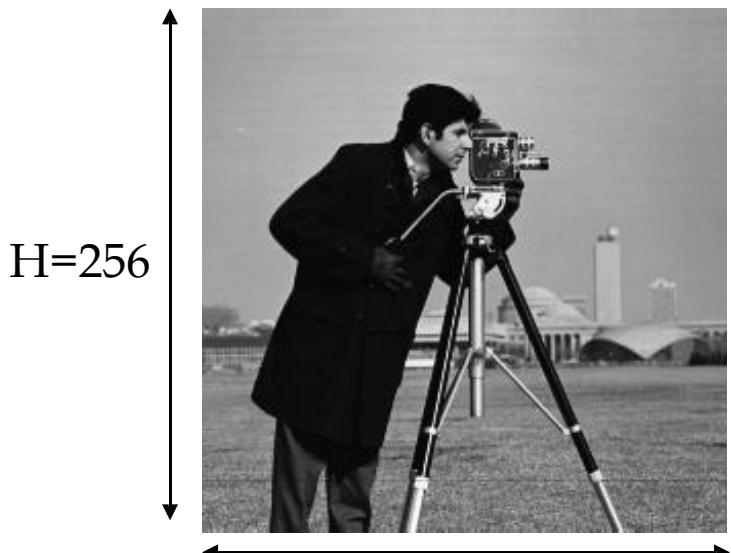
Nilai matriks pada bagian citra yang ditandai

99	71	61	51	49	40	35	53	86	99
83	74	53	56	48	46	48	72	85	102
101	69	57	53	54	52	64	82	88	101
107	82	64	53	59	60	81	90	93	100
114	93	76	69	72	85	94	99	95	99
117	108	94	92	97	101	100	108	105	99
116	114	109	106	105	108	108	102	107	110
115	113	109	114	111	111	113	108	111	115
110	119	111	108	106	108	110	115	120	122
103	107	106	108	109	114	120	124	124	132

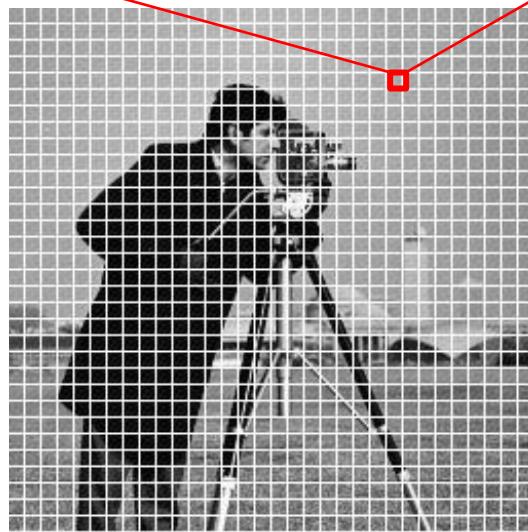
# Representasi Matriks

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

183	160	94	153	194	163	132	165
183	153	116	176	187	166	130	169
179	168	171	182	179	170	131	167
177	177	179	177	179	165	131	167
178	178	179	176	182	164	130	171
179	180	180	179	183	169	132	169
179	179	180	182	183	170	129	173
180	179	181	179	181	170	130	169



Divide into  
8x8 blocks



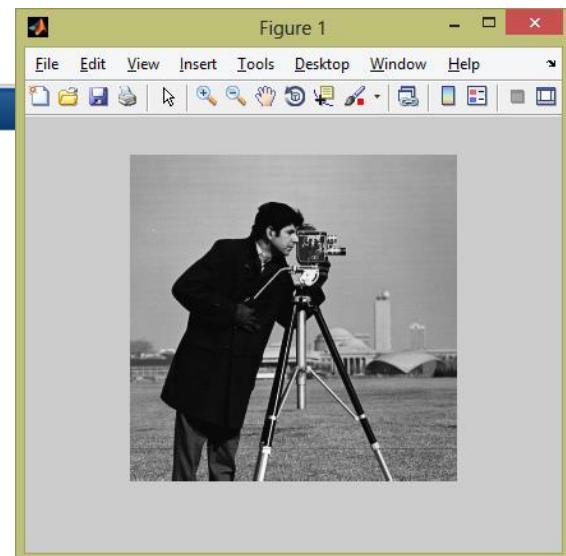
# Implementasi di matlab

Command Window

```
>> x=imread('cameraman.tif');  
>> imshow(x);  
>> x  
  
x =
```

Columns 1 through 15

```
156 159 158 155 158 156 159 158 157 158 158 159 160  
160 154 157 158 157 159 158 158 158 160 155 156 159  
156 159 158 155 158 156 159 158 157 158 158 159 160  
160 154 157 158 157 159 158 158 158 160 155 156 159  
156 153 155 159 159 155 156 155 155 157 155 154 154 158 162  
155 155 155 157 156 159 152 158 156 158 152 153 159 156 157  
156 153 157 156 153 155 154 155 157 156 155 156 155 157 158  
159 159 156 158 156 159 157 161 162 157 157 159 161 156 163  
158 155 158 154 156 160 162 155 159 161 156 161 160 155 158  
155 154 157 158 160 160 159 160 158 161 160 160 158 161 158  
154 157 157 157 156 155 159 154 159 158 161 158 158 160 159  
152 150 155 154 152 156 157 156 157 154 157 159 155 156 159  
157 153 156 155 157 160 160 157 159 159 160 161 160 160 158  
151 154 157 156 156 158 158 156 157 159 158 156 159 161 159  
156 157 157 160 159 159 156 158 159 162 161 160 160 161 162  
157 158 159 157 157 154 153 158 159 155 160 159 161 161 159
```



# Resolusi Citra



1024



512



256



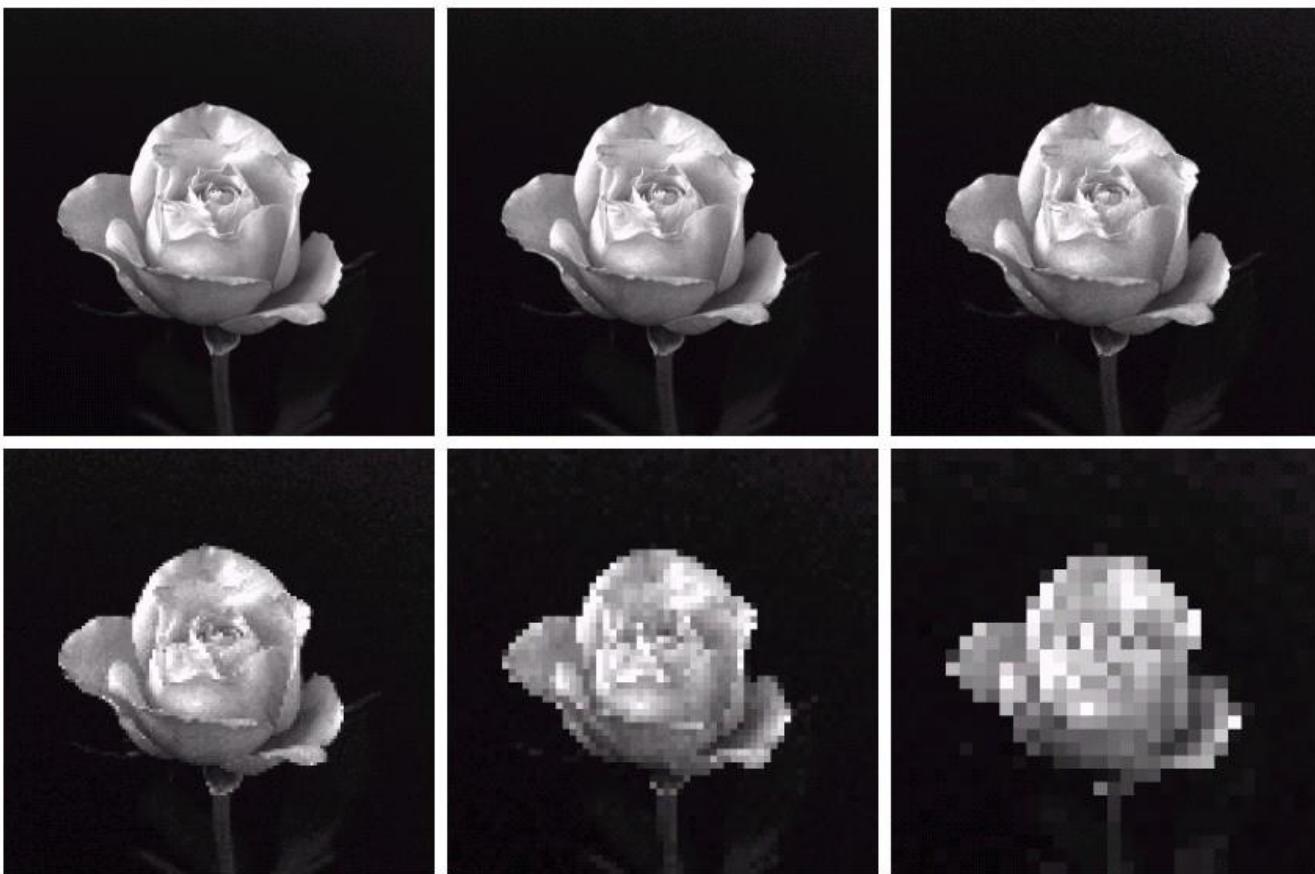
128



32

**FIGURE 2.19** A  $1024 \times 1024$ , 8-bit image subsampled down to size  $32 \times 32$  pixels. The number of allowable gray levels was kept at 256.

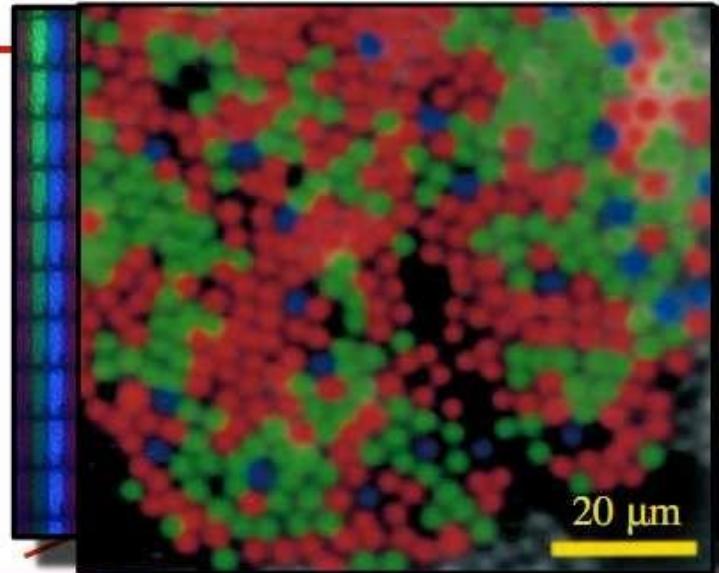
# Resolusi Citra



a	b	c
d	e	f

**FIGURE 2.20** (a)  $1024 \times 1024$ , 8-bit image. (b)  $512 \times 512$  image resampled into  $1024 \times 1024$  pixels by row and column duplication. (c) through (f)  $256 \times 256$ ,  $128 \times 128$ ,  $64 \times 64$ , and  $32 \times 32$  images resampled into  $1024 \times 1024$  pixels.

# Komponen Warna



Monochrome image



$$R[x,y] = G[x,y] = B[x,y]$$



Red  $R[x,y]$



Green  $G[x,y]$



Blue  $B[x,y]$

## PENGOLAHAN CITRA

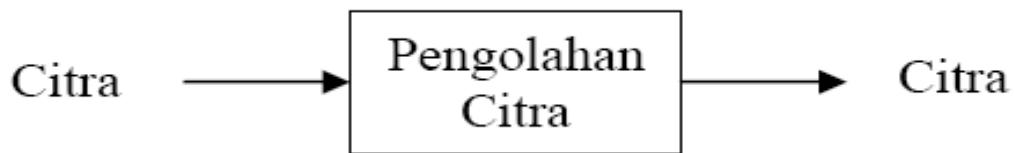
Operasi-operasi pada pengolahan citra diterapkan pada citra bila :

- Perbaikan atau memodifikasi citra dilakukan untuk meningkatkan kualitas penampakan citra/ menonjolkan beberapa aspek informasi yang terkandung dalam citra (image enhancement).  
Contoh : perbaikan kontras gelap/terang, perbaikan tepian obyek, penajaman, pemberian warna semu, dll.

- Adanya cacat pada citra sehingga perlu dihilangkan/diminimumkan (image restoration). Contoh : penghilangan kesamaran (deblurring); citra tampak kabur karena pengaturan fokus lensa tidak tepat/kamera goyang, penghilangan noise.
- Elemen dalam citra perlu dikelompokkan, dicocokkan atau diukur (image segmentation). Operasi ini berkaitan erat dengan pengenalan pola.

- Diperlukan ekstraksi ciri-ciri tertentu yang dimiliki citra untuk membantu dalam pengidentifikasiannya (image analysis). Proses segmentasi kadangkala diperlukan untuk melokalisasi obyek yang diinginkan dari sekelilingnya. Contoh : pendekripsi tepi obyek.
- Sebagian citra perlu digabung dengan bagian citra yang lain (image reconstruction). Contoh : beberapa foto rontgen digunakan untuk membentuk ulang gambar organ tubuh

- Citra perlu dimampatkan (image compression).  
Contoh : suatu file citra berbentuk bmp berukuran 258 kb dimampatkan dengan metode JPEG menjadi berukuran 49 kb.
- Menyembunyikan data rahasia (berupa teks/citra) pada citra sehingga keberadaan data rahasia tersebut tidak diketahui orang (steganografi & watermarking)



# Pengolahan Citra Digital

- Perbaikan kualitas citra (*Image Enhancement*)
- Pemugaran citra (*Image Restoration*)
- Segmentasi citra (*Image Segmentation*)
- Rekonstruksi citra (*Image Reconstruction*)
- Penambahan efek citra (*Image Stylization*)
- Pemampatan citra (*Image Compression*)
- Analisis citra (*Image Analysis*)

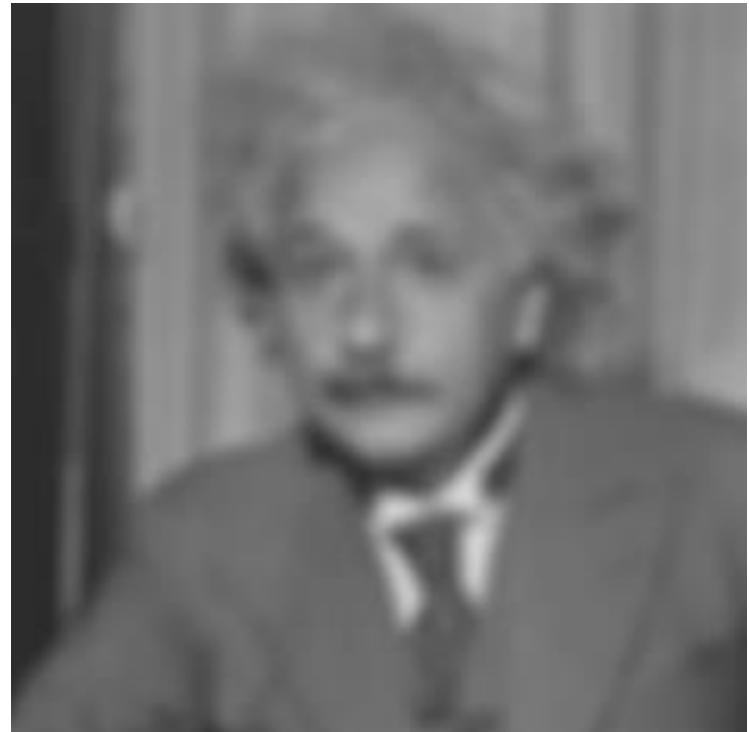
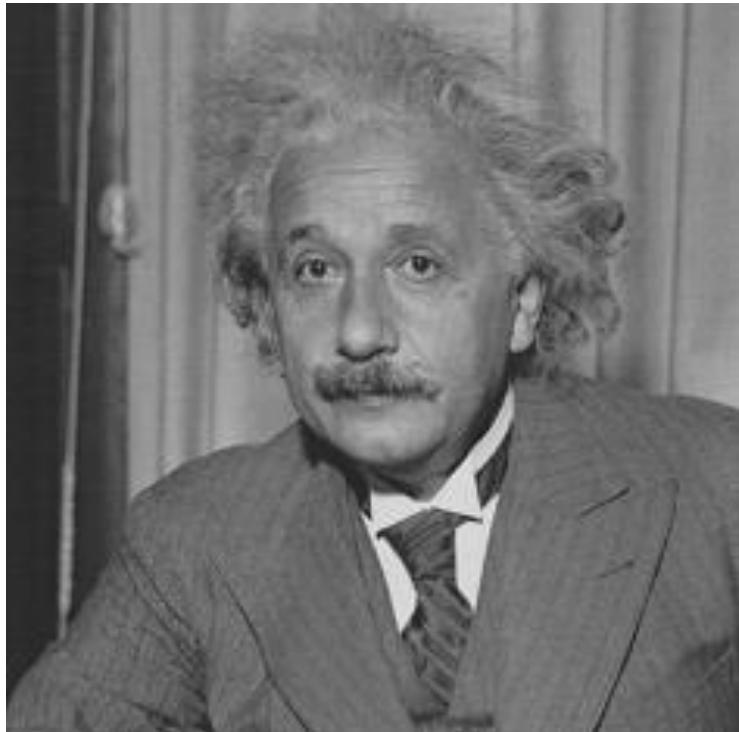
# Peningkatan Kontras



# Penajaman (*Sharpening*)



# Pengkaburan (*Bluring*)



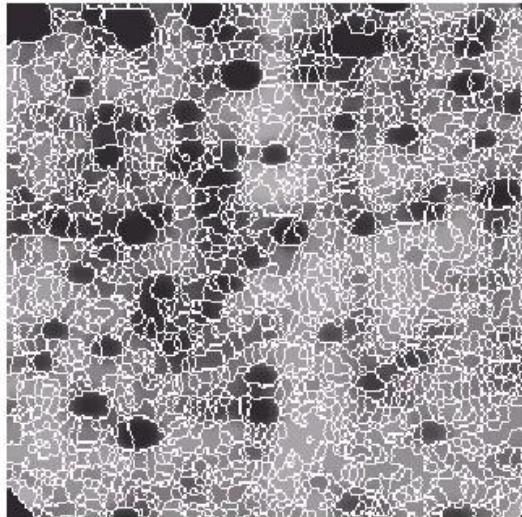
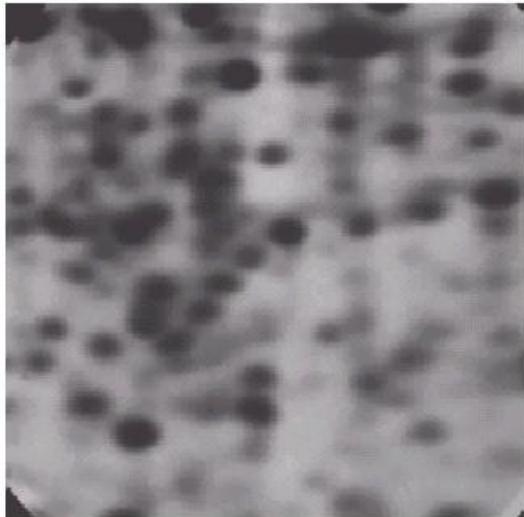
# Menghilangkan Noise



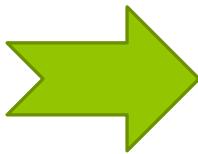
# Pemugaran Citra



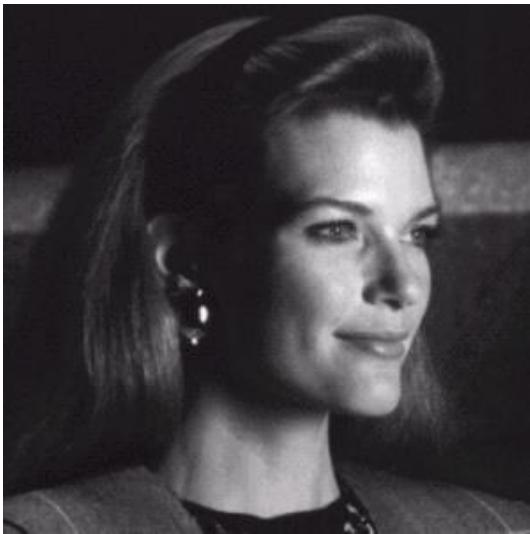
# Segmentasi Citra



# Rekonstruksi Citra



# Kompresi Citra



original image  
262144 Bytes

compression ratio (CR) = 108:1

**image  
encoder**

compressed bitstream  
00111000001001101...  
(2428 Bytes)

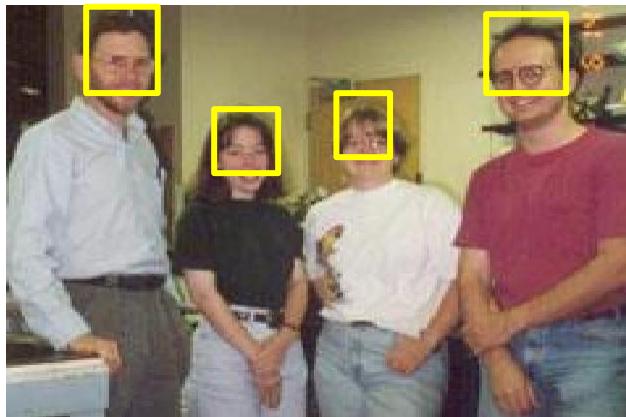
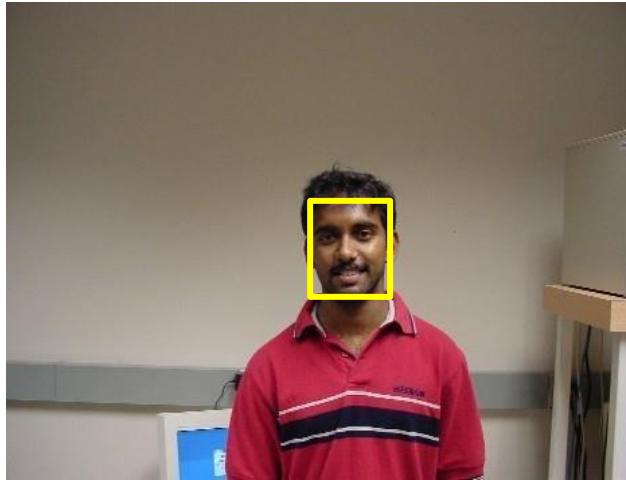
**image  
decoder**

# Image Analysis: Edge Detection



From [Gonzalez & Woods]

# Image Analysis: Face Detection



From Prof. Xin Li

# Image Analysis: Skin Detection



# Image Analysis: Image Matching



From Prof. Xin Li

**Thank You !**

# PENGOLAHAN CITRA DIGITAL

OPERASI DASAR  
GKV - IF 2020

# OPERASI DASAR CITRA

- LEVEL KOMPUTASI
- OPERASI ARITMATIKA
- OPERASI BOOLEAN
- OPERASI GEOMETRI

# ODC - LEVEL KOMPUTASI

- TITIK
- LOKAL
- GLOBAL

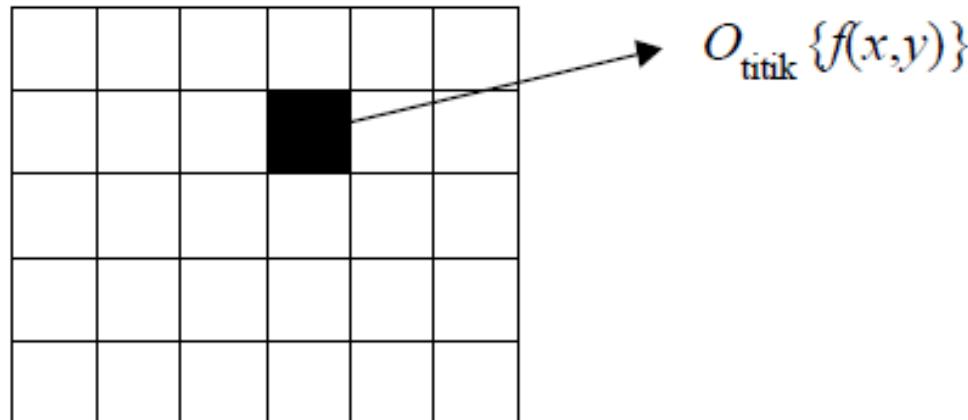
## LK-TITIK

- Operasi pada level titik hanya dilakukan pada *pixel tunggal di dalam citra*.
- *Operasi titik* dikenal juga dengan nama operasi *pointwise*.
- *Operasi ini terdiri dari pengaksesan pixel pada lokasi yang diberikan, memodifikasinya dengan operasi linear atau nonlinear, dan menempatkan nilai pixel baru pada lokasi yang bersesuaian di dalam citra yang baru.*
- Operasi ini diulangi untuk keseluruhan *pixel di dalam citra*.

## LK-TITIK (2)

- Secara matematis operasi titik dituliskan sebagai berikut :

$$f_B(x, y) = O_{\text{titik}} \{f_A(x, y)\}$$



## LK-TITIK (3)

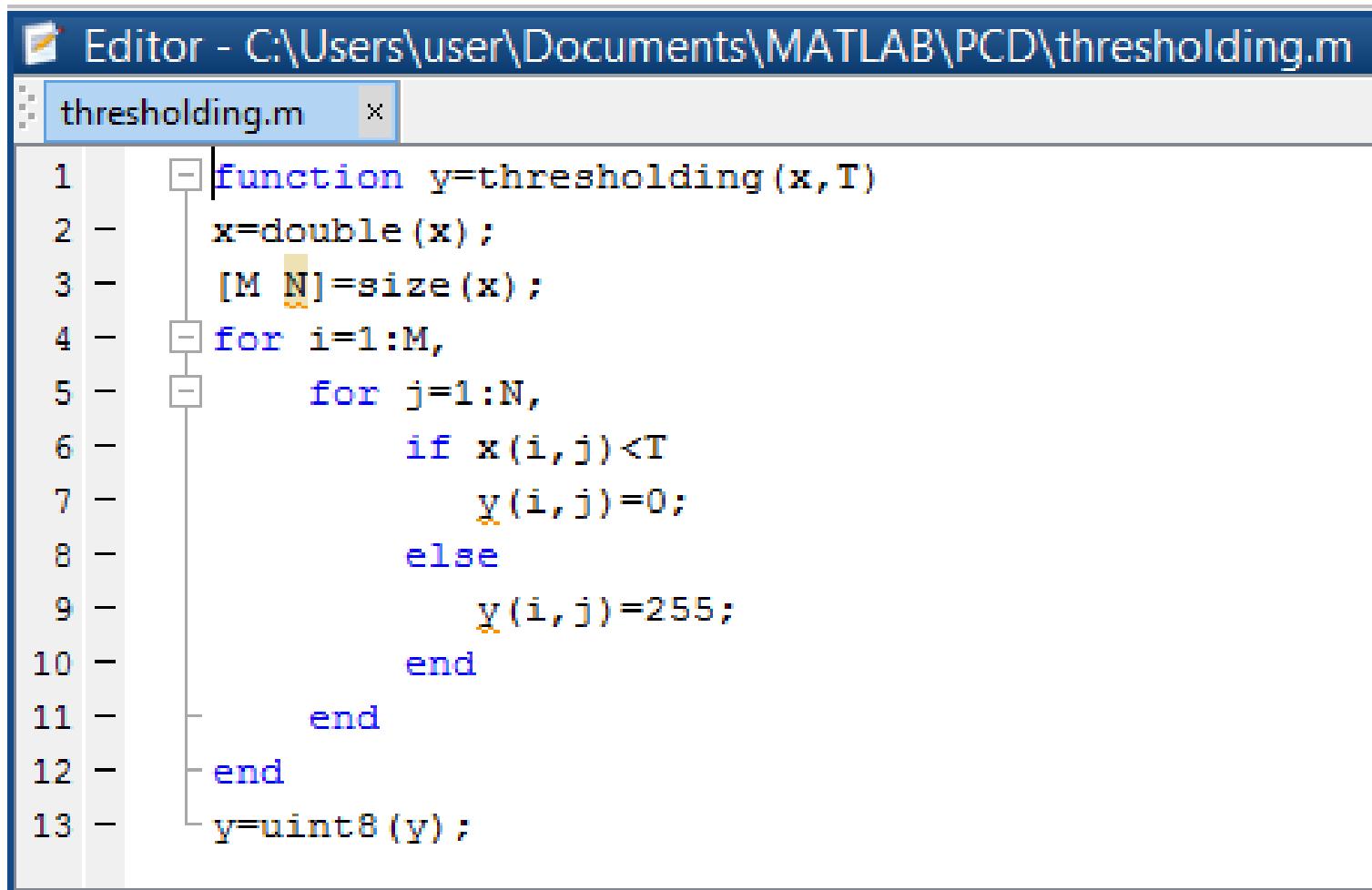
- Pengambangan (Thresholding)
- Negatif (Negative)
- Pencerahan (Brightening)
- Pemotongan (Clipping)

# Pengambangan (Thresholding-1)

- Pada operasi pengambangan, nilai intensitas *pixel dipetakan ke salah satu dari dua nilai,  $a_1$  atau  $a_2$ , berdasarkan nilai ambang (*threshold*)  $T$ :*

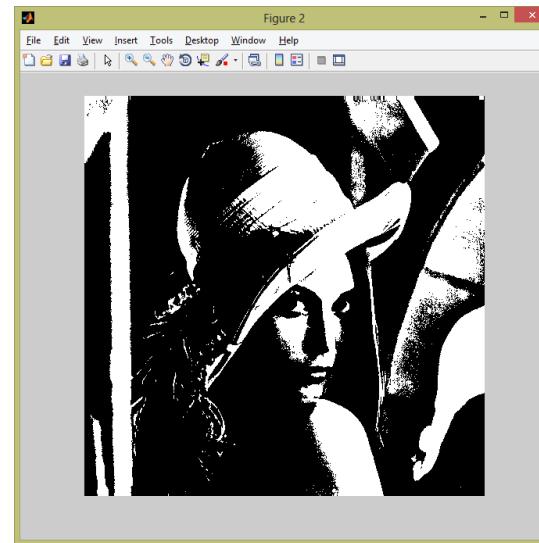
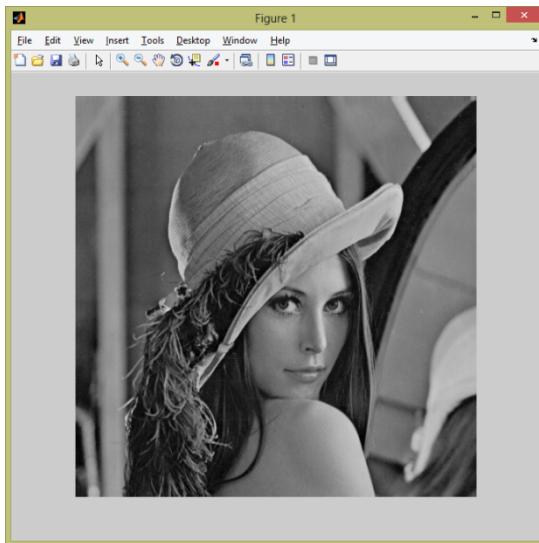
$$f(x, y)^\circ = \begin{cases} a_1, & f(x, y) < T \\ a_2, & f(x, y) \geq T \end{cases}$$

# Pengambangan (Thresholding-2)



```
Editor - C:\Users\user\Documents\MATLAB\PCD\thresholding.m
thresholding.m x
1 function y=thresholding (x,T)
2 x=double (x);
3 [M N]=size (x);
4 for i=1:M,
5     for j=1:N,
6         if x(i,j)<T
7             y(i,j)=0;
8         else
9             y(i,j)=255;
10        end
11    end
12 end
13 y=uint8 (y);
```

# Pengambangan (Thresholding-3)



Variables - x

x	x				
x <512x512 uint8>					
5	6	7	8	9	
6	136	134	134	135	136
7	135	137	134	131	134
8	133	134	131	132	134
9	134	135	132	132	138
10	130	129	130	132	136
11	131	130	130	127	134
12	134	130	125	123	135
13	132	128	134	131	136
14	135	132	131	131	134

Variables - y

y	y				
y <512x512 uint8>					
5	6	7	8	9	
6	255	255	255	255	255
7	255	255	255	255	255
8	255	255	255	255	255
9	255	255	255	255	255
10	255	255	255	255	255
11	255	255	255	0	255
12	255	255	0	0	255
13	255	255	255	255	255
14	255	255	255	255	255

## Negative Image -1

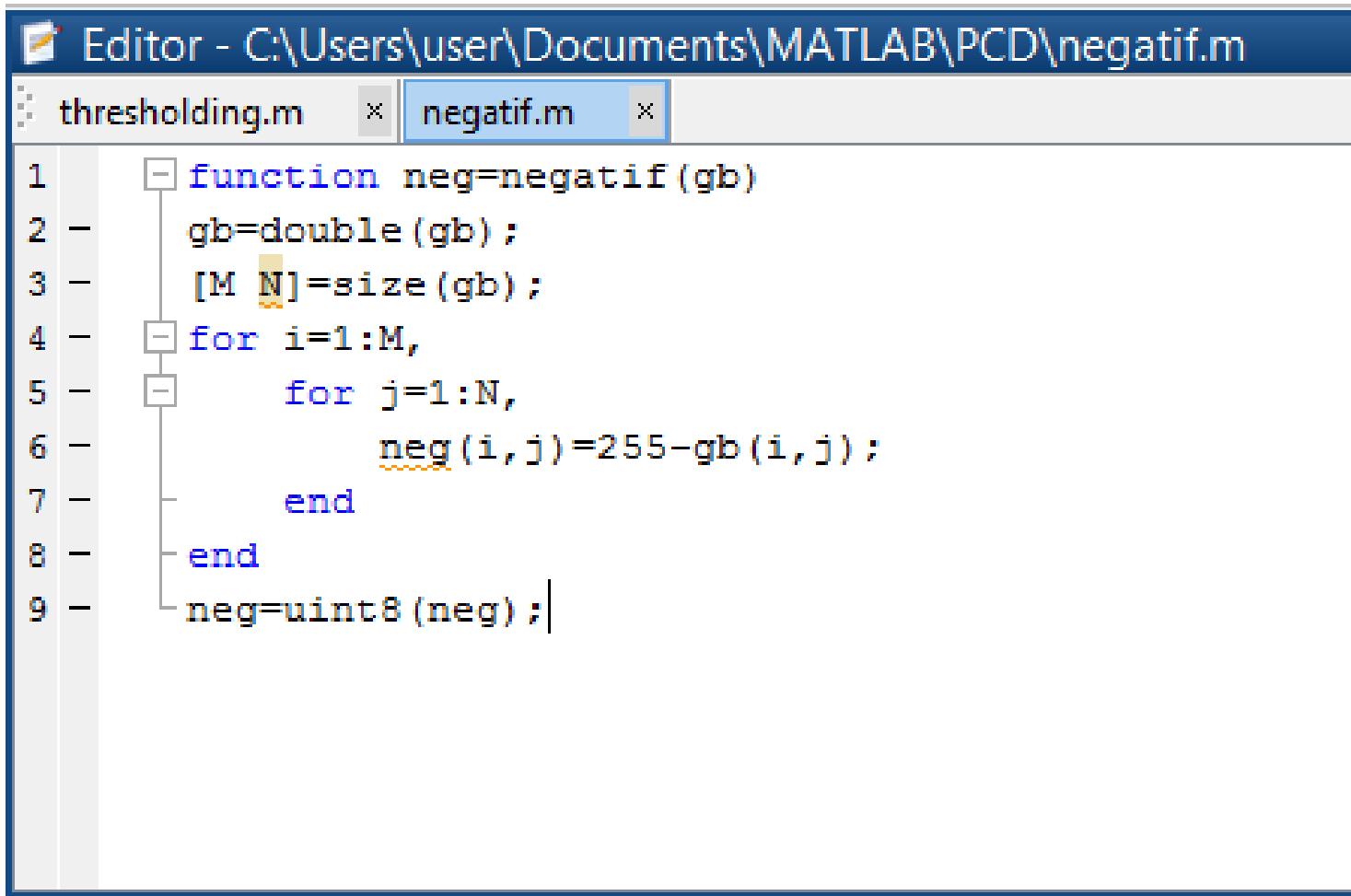
Operasi negatif digunakan untuk mendapatkan citra negatif (*negative image*) meniru film negatif pada fotografi dengan cara mengurangi nilai intensitas *pixel* dari nilai keabuan maksimum.

$$f'(x,y) = \text{max gray level} - f(x,y)$$

$$f'(x,y) = 255 - f(x,y), \text{ untuk 8 bit}$$

$$f'(x,y) = 127 - f(x,y), \text{ untuk 7 bit}$$

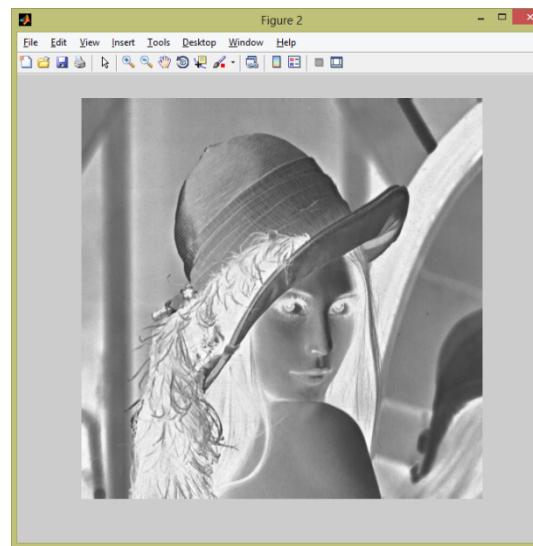
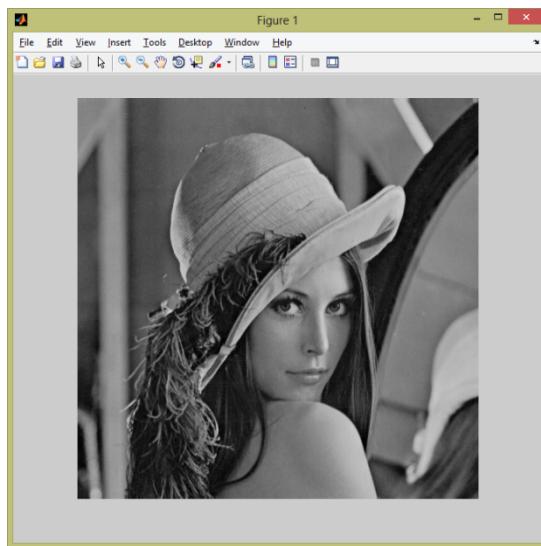
# Negative Image -2



The screenshot shows the MATLAB Editor window with two files open: 'thresholding.m' and 'negatif.m'. The 'negatif.m' file is active and contains the following code:

```
1 function neg=negatif(gb)
2 gb=double(gb);
3 [M N]=size(gb);
4 for i=1:M,
5     for j=1:N,
6         neg(i,j)=255-gb(i,j);
7     end
8 end
9 neg=uint8(neg);
```

# Negative Image -3



	5	6	7	8	9
6	136	134	134	135	136
7	135	137	134	131	134
8	133	134	131	132	134
9	134	135	132	132	138
10	130	129	130	132	136
11	131	130	130	127	134
12	134	130	125	123	135
13	132	128	134	131	136
14	135	132	131	131	134

	5	6	7	8	9
6	119	121	121	120	119
7	120	118	121	124	121
8	122	121	124	123	121
9	121	120	123	123	117
10	125	126	125	123	119
11	124	125	125	128	121
12	121	125	130	132	120
13	123	127	121	124	119
14	120	123	124	124	121

## Pencerahan - 1

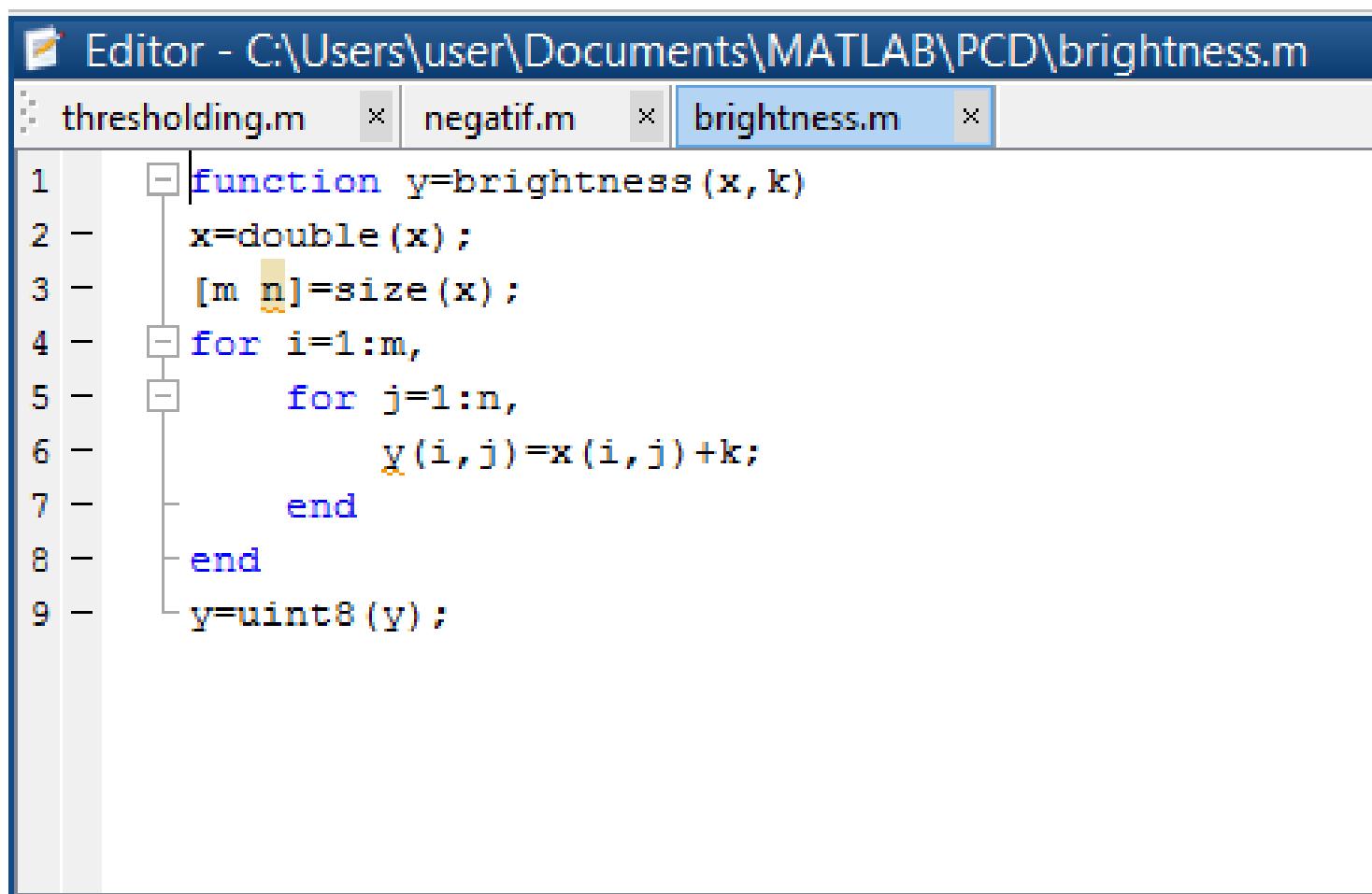
Kecerahan citra dapat diperbaiki dengan menambahkan sebuah konstanta kepada (atau dari) setiap *pixel di dalam citra*.

Secara matematis operasi ini ditulis sebagai :

$$f(x, y)' = f(x, y) + b$$

Jika  $b$  positif, kecerahan citra bertambah, sebaliknya jika  $b$  negatif kecerahan citra berkurang.

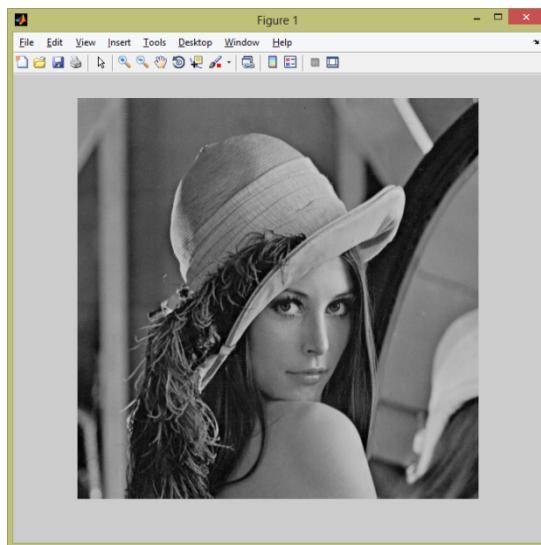
# Pencerahan - 2



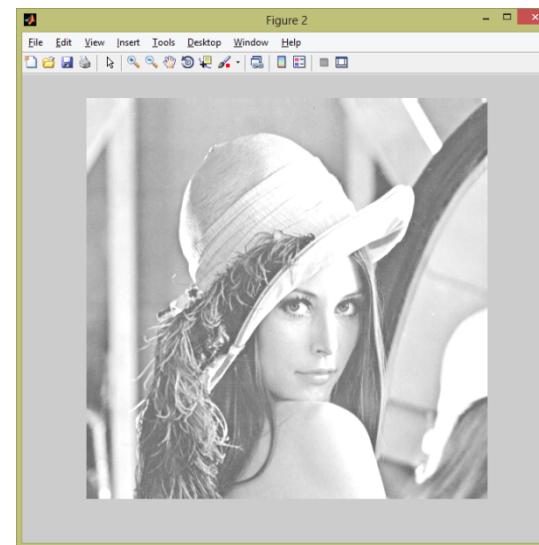
The screenshot shows the MATLAB Editor window with the title bar "Editor - C:\Users\user\Documents\MATLAB\PCD\brightness.m". The current file is "brightness.m", which is highlighted in the tab bar. Other files listed in the tab bar are "thresholding.m" and "negatif.m". The code editor displays the following MATLAB script:

```
function y=brightness(x, k)
x=double(x);
[m n]=size(x);
for i=1:m,
    for j=1:n,
        y(i,j)=x(i,j)+k;
    end
end
y=uint8(y);
```

## Pencerahan - 3



	5	6	7	8	9
6	136	134	134	135	136
7	135	137	134	131	134
8	133	134	131	132	134
9	134	135	132	132	138
10	130	129	130	132	136
11	131	130	130	127	134
12	134	130	125	123	135
13	132	128	134	131	136
14	135	132	131	131	134



	5	6	7	8	9
6	236	234	234	235	236
7	235	237	234	231	234
8	233	234	231	232	234
9	234	235	232	232	238
10	230	229	230	232	236
11	231	230	230	227	234
12	234	230	225	223	235
13	232	228	234	231	236
14	235	232	231	231	234

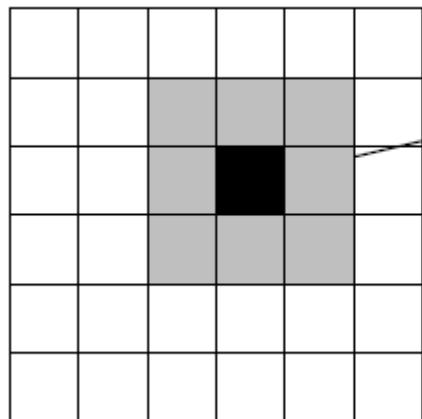
# Pemotongan (clipping)

Operasi ini dilakukan jika nilai intensitas *pixel hasil suatu operasi pengolahan citra* terletak di bawah nilai intensitas minimum atau di atas nilai intensitas maksimum:

$$f(x, y)' = \begin{cases} 255, & f(x, y) > 255 \\ f(x, y), & 0 \leq f(x, y) \leq 255 \\ 0, & f(x, y) < 0 \end{cases}$$

# LK - LOKAL

Operasi pada aras lokal menghasilkan citra keluaran yang intensitas suatu pixel bergantung pada intensitas pixel-pixel tetangganya.

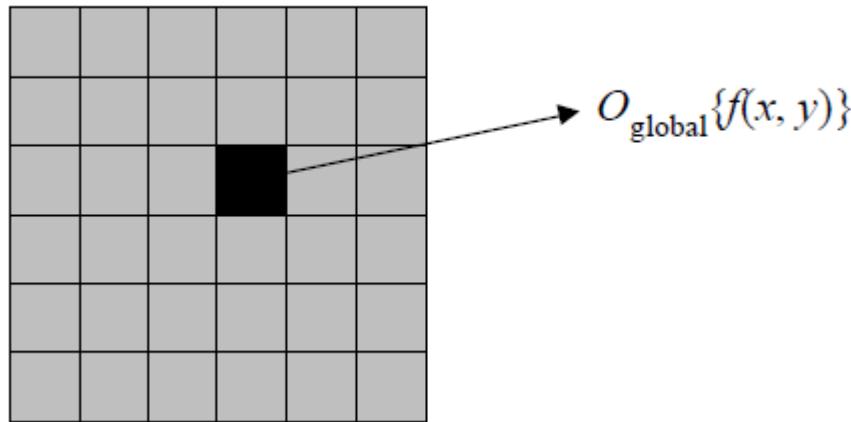


$O_{\text{lokal}} \{f(x_i, y_i), (x_i, y_i) \text{ pixel}$   
di sekitar  $(x, y)\}$



# LK - GLOBAL

Operasi pada aras global menghasilkan citra keluaran yang intensitas suatu *pixel* bergantung pada intensitas keseluruhan *pixel*.



Contoh operasi global : Perataan Histogram

# OPERASI ARITMATIKA

Karena citra digital adalah matriks, maka operasi-operasi aritmetika matriks juga berlaku pada citra. Operasi matriks yang dapat dilakukan adalah:

1. Penjumlahan atau pengurangan antara dua buah citra  $A$  dan  $B$ :

$$C(x, y) = A(x, y) \pm B(x, y),$$

2. Perkalian dua buah citra:

$$C(x, y) = A(x, y) B(x, y),$$

3. Penjumlahan/pengurangan citra  $A$  dengan skalar  $c$ :

$$B(x, y) = A(x, y) \pm c,$$

4. Perkalian/pembagian citra  $A$  dengan sebuah skalar  $c$ :

$$B(x, y) = c \times A(x, y)$$

# OPERASI BOOLEAN

Selain operasi aritmatika, pemrosesan citra digital juga melibatkan operasi Boolean (**and**, **or**, **dan not**) :

1.  $C(x, y) = A(x, y)$  **and**  $B(x, y)$ ,
2.  $C(x, y) = A(x, y)$  **or**  $B(x, y)$ ,
3.  $C(x, y) = \text{not } A(x, y)$ .



# OPERASI GEOMETRI

Pada operasi geometrik, koordinat *pixel berubah akibat transformasi, sedangkan intensitasnya tetap*. Ini berbeda dengan dengan operasi aritmetika yang mana koordinat *pixel tetap sedangkan intensitasnya berubah*.

$$f'(x', y') = f(g_1(x, y), g_2(x, y))$$

1. Translasi
2. Rotasi
3. Penskalaan
4. Flipping

# GEOMETRI-TRANSLASI

- Rumus translasi citra:

$$x' = x + m$$

$$y' = y + n$$

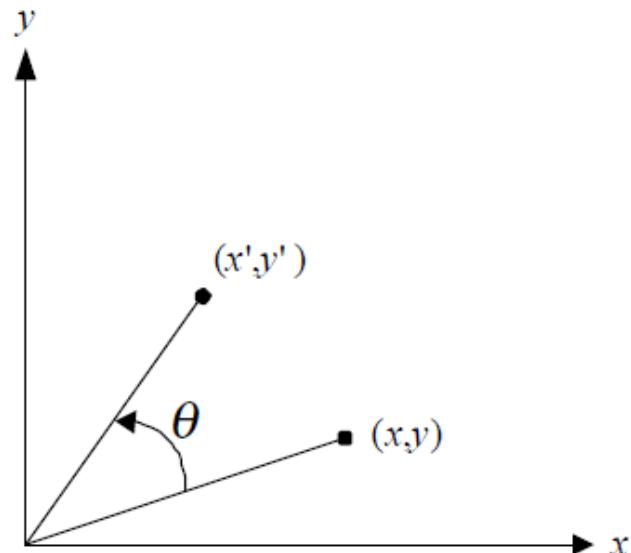


# GEOMETRI-ROTASI

- Rumus rotasi citra:

$$x' = x \cos(\theta) - y \sin(\theta)$$

$$y' = x \sin(\theta) + y \cos(\theta)$$



Rotasi 90 derajat CCW

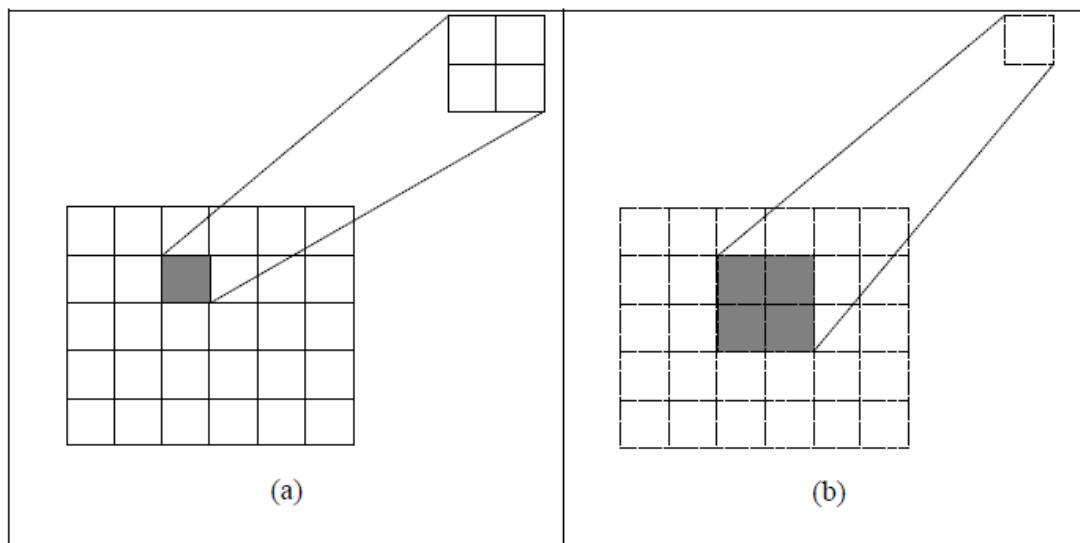
# GEOMETRI-PENSKALAAN

Penskalaan citra, disebut juga *image zooming*, yaitu pengubahan ukuran citra (membesar/zoom out atau mengecil/zoom in).

Rumus penskalaan citra:

$$x' = sx \times x$$

$$y' = sy \times y$$



# GEOMETRI-PENSKALAAN-2



Penskalaan 2 x

# GEOMETRI - FLIPPING

*Flipping adalah operasi geometri yang sama dengan pencerminan (image reflection). Ada dua macam flipping: horizontal dan vertikal.*

*Flipping Horisontal :*

$$f'(x,y) = f(N-x,y)$$

*Flipping Vertikal :*

$$f'(x,y)=f(x,M-y)$$

# GEOMETRI – FLIPPING-2



(a) citra



(b) *flip* horizontal



(c) *flip* vertikal

# PENGOLAHAN CITRA DIGITAL

IMAGES ENHANCED (SPATIAL DOMAIN)

GKV - IF 2020

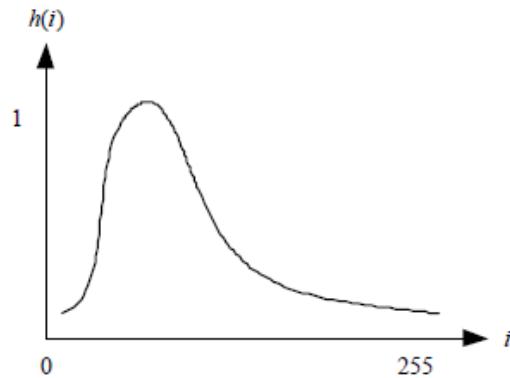
# OUTLINE

- HISTOGRAM
- KONVOLUSI
- PELEMBUTAN GAMBAR
- PENAJAMAN GAMBAR
- PEREGANGAN KONTRAS

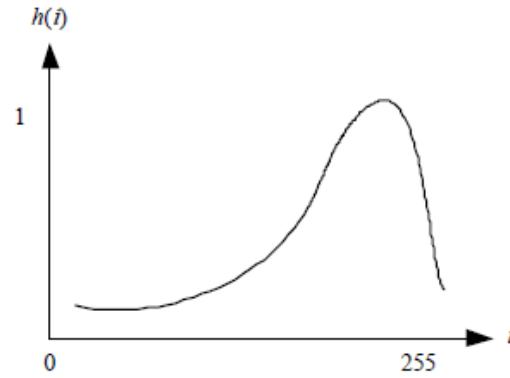
# HISTOGRAM-1

- Histogram sebuah citra digital merupakan grafik yang memvisualisasikan distribusi nilai intensitas pixel dari citra digital.
- Histogram merupakan sebuah tool yang sangat berguna untuk mengetahui kualitas sebuah citra digital.
- Berdasarkan penyebaran atau pengelompokan pixel dari histogram akan dapat diketahui apakah sebuah citra terlalu terang atau terlalu gelap atau dalam keadaan normal.

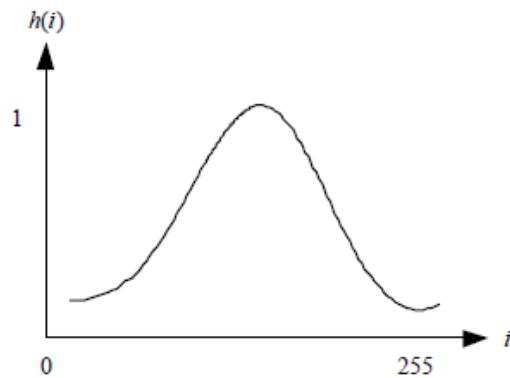
# HISTOGRAM-2



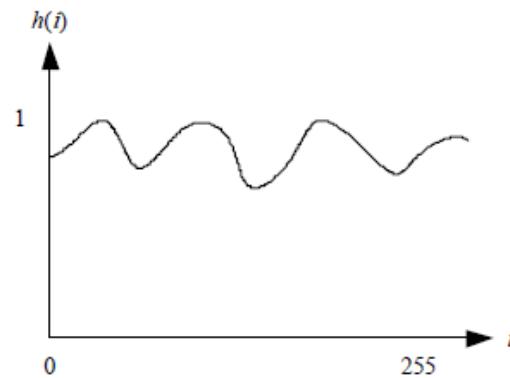
(a)



(b)



(c)



(d)

(a) citra gelap, (b) citra terang, (c) citra normal dan (d) citra normal dengan kontras tinggi

# MEMBANGUN HISTOGRAM-1

- Histogram citra digital dengan derajat keabuan  $[0, L-1]$  merupakan sebuah fungsi diskrit :

$$h(r_k) = n_k$$

dimana  $r_k$  merupakan derajat keabuan ke  $k$  dan  $n_k$  merupakan jumlah pixel dalam citra yang memiliki derajat keabuan  $r_k$ .

- Nilai  $L$  bergantung pada kedalaman bit yang digunakan. Untuk citra digital dengan kedalaman 8 bit maka nilai  $L$  memiliki rentang 0 – 255.

## MEMBANGUN HISTOGRAM-2

- Untuk menormalisasi histogram maka dilakukan dengan cara membagi masing – masing nilai  $n_k$  dengan jumlah pixel pada citra tersebut dengan rumus :

$$p(r_k) = n_k / n$$

dimana n merupakan jumlah pixel dari citra.

## MEMBANGUN HISTOGRAM-3

- Andaikan terdapat sebuah citra dengan kedalaman 3 bit, maka derajat keabuannya memiliki rentang nilai 0 – 7 dengan ukuran  $4 \times 4$  pixel dengan matriks :

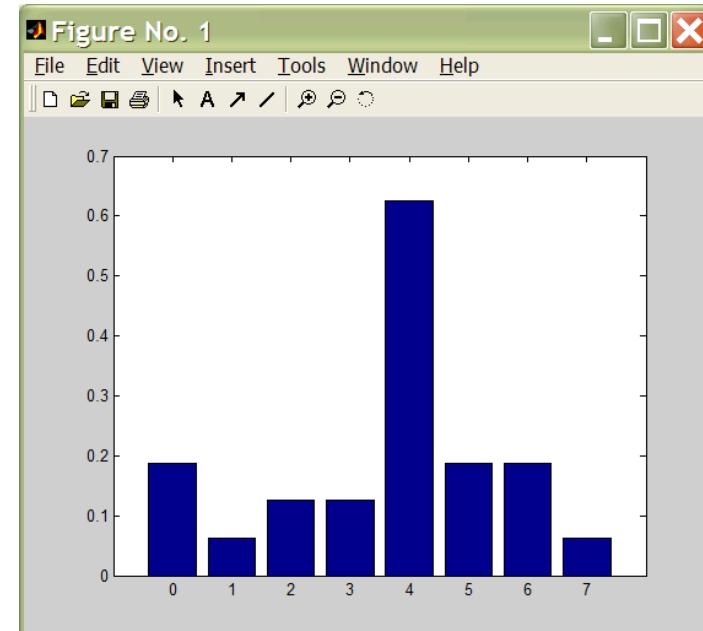
$$\begin{bmatrix} 1 & 7 & 0 & 5 \\ 6 & 6 & 0 & 2 \\ 0 & 5 & 6 & 5 \\ 3 & 2 & 4 & 3 \end{bmatrix}$$

# MEMBANGUN HISTOGRAM-4

Menghitung frekuensi kemunculan pixel dan mentabulasikannya

Derajat keabuan	Jumlah	$p(r_k) = n_k / n$
0	3	0,1875
1	1	0,0625
2	2	0,125
3	2	0,125
4	1	0,0625
5	3	0,1875
6	3	0,1875
7	1	0,0625

# MEMBANGUN HISTOGRAM-5

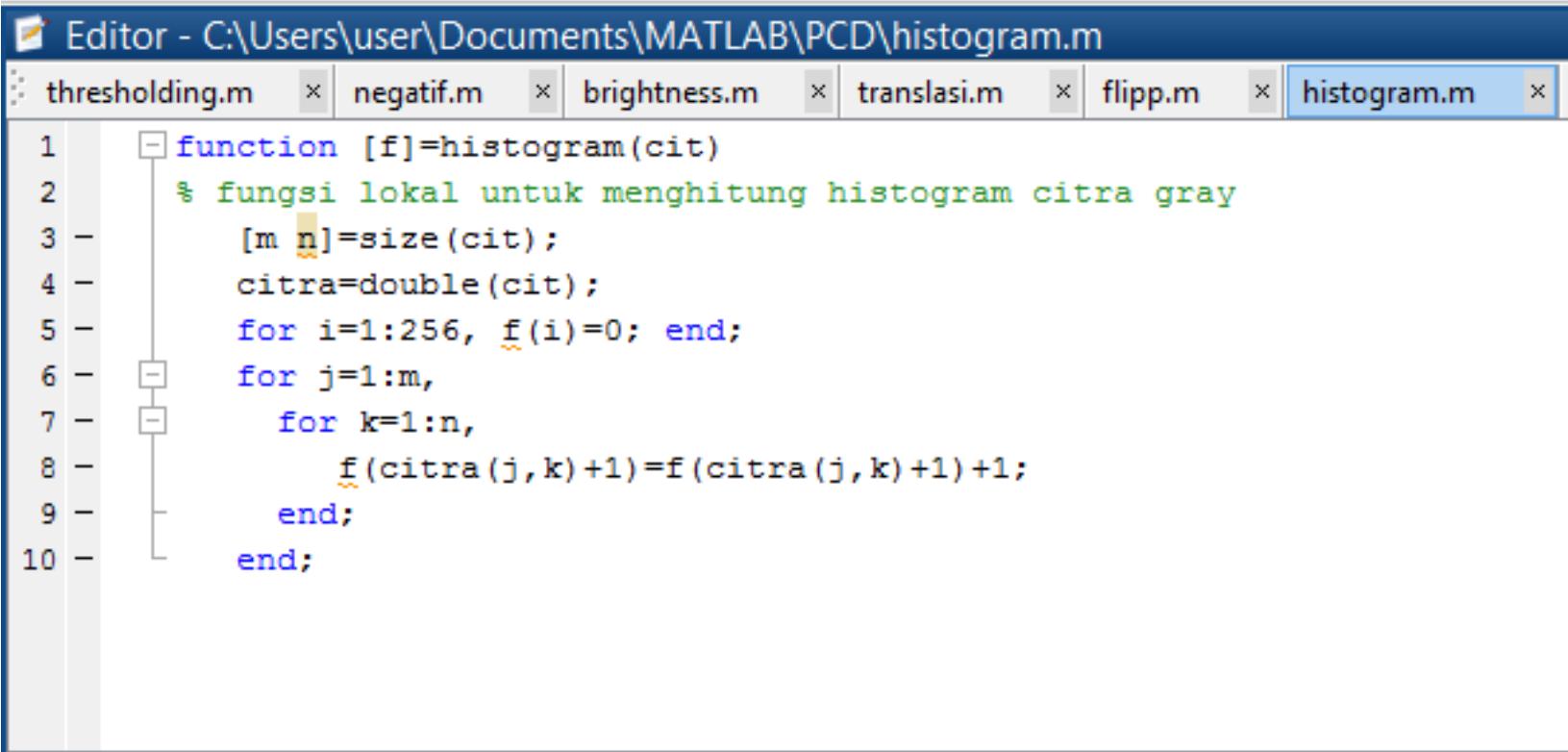
$$\begin{bmatrix} 1 & 7 & 0 & 5 \\ 6 & 6 & 0 & 2 \\ 0 & 5 & 6 & 5 \\ 3 & 2 & 4 & 3 \end{bmatrix}$$


# MEMBANGUN HISTOGRAM-6

Tahapan :

- Baca citra
- Hitung ukuran citra ( $M$  baris dan  $N$  kolom)
- Untuk  $i$  dari 1 sampai  $M$ 
  - Untuk  $j$  dari 1 sampai  $N$ 
    - Hitung kemunculan tiap-tiap derajat keabuan (Turus)

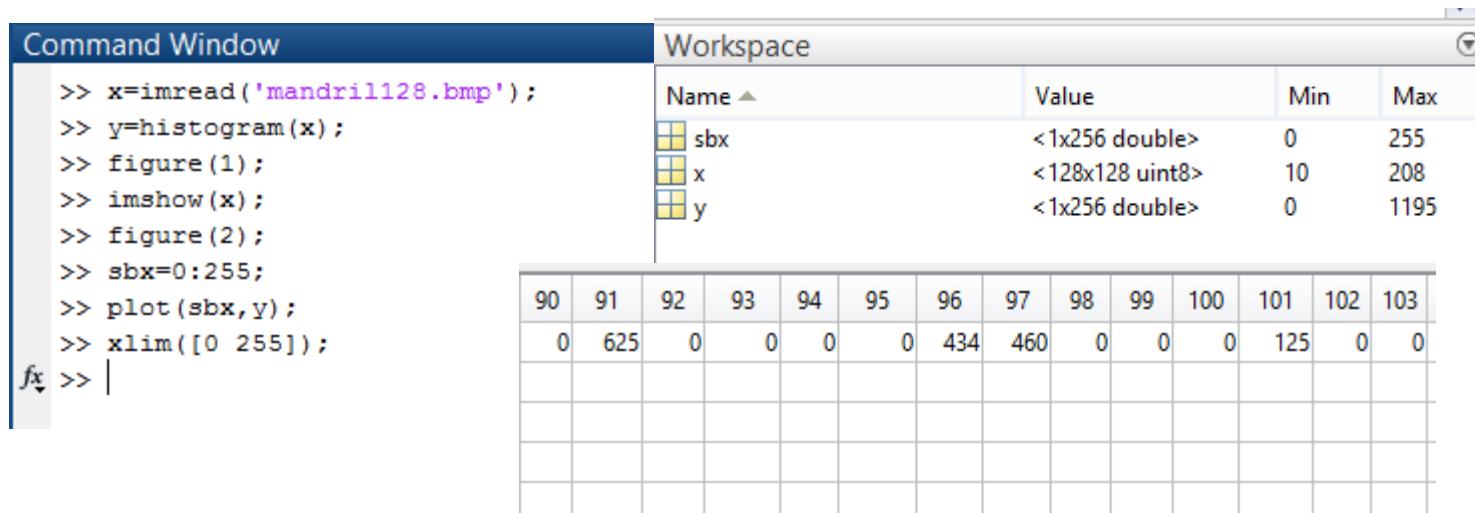
# MEMBANGUN HISTOGRAM-7



The screenshot shows the MATLAB Editor window with the title bar "Editor - C:\Users\user\Documents\MATLAB\PCD\histogram.m". The current tab is "histogram.m", which is highlighted in blue. The editor displays the following MATLAB code:

```
1 function [f]=histogram(cit)
2 % fungsi lokal untuk menghitung histogram citra gray
3 [m n]=size(cit);
4 citra=double(cit);
5 for i=1:256, f(i)=0; end;
6 for j=1:m,
7     for k=1:n,
8         f(citra(j,k)+1)=f(citra(j,k)+1)+1;
9     end;
10 end;
```

# MEMBANGUN HISTOGRAM-8



# KONVOLUSI-1

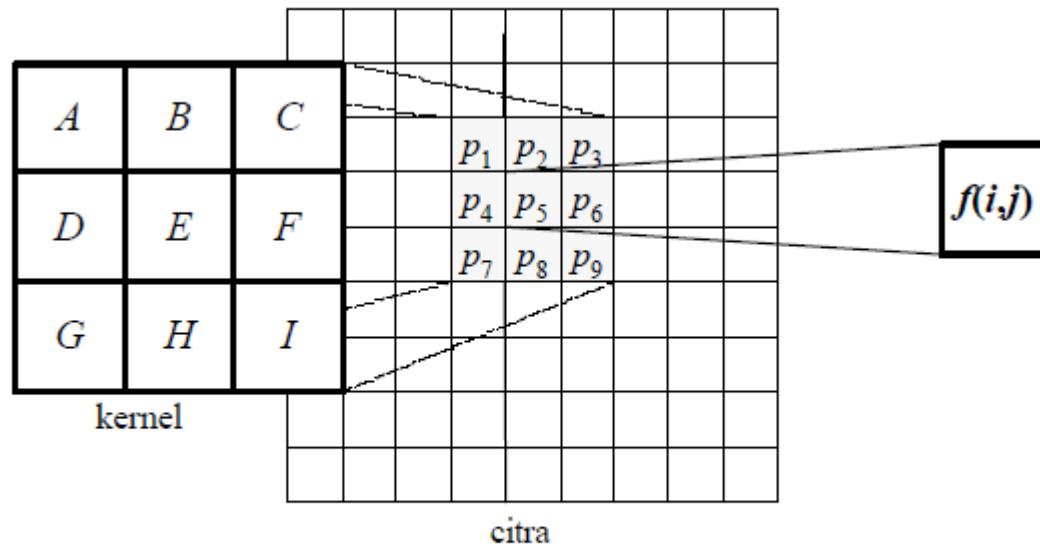
- Konvolusi merupakan operasi pengolahan citra yang mengalikan sebuah citra dengan sebuah *mask atau kernel*.
- Fungsi Konvolusi kontinu dan diskrit adalah :

$$h(x, y) = f(x, y) * g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(a, b)g(x-a, y-b) da db$$

$$h(x, y) = f(x, y) * g(x, y) = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f(a, b)g(x-a, y-b)$$

- Dimana  $f(x)$  adalah citra dan  $g(x)$  merupakan kernel atau filter.

# KONVOLUSI-2



$$f(i,j) = A p_1 + B p_2 + C p_3 + D p_4 + E p_5 + F p_6 + G p_7 + H p_8 + I p_9$$

- Kernel dapat berupa matriks berukuran 2x2, 3x3, 5x5 maupun yang lain dan umumnya ukuran kernel lebih kecil dibandingkan dengan ukuran citranya.

# KONVOLUIS-3

Citra (C)

Kernel (K)

K(1,1)	K(1,2)	K(1,3)
K(2,1)	K(2,2)	K(2,3)
K(3,1)	K(3,2)	K(3,3)

(1,1)	(1,2)	(1,3)	.....	.....	(1,N)
(2,1)	(2,2)	(2,3)	.....	.....	(2,N)
(3,1)	(3,2)	(3,3)	.....	.....	(3,N)
:	:	:	:	:	:
:	:	:	:	:	:
(M,1)	(M,2)	(M,3)	.....	.....	(M,N)

for i = 2 : M-1,

for j = 2 : N-1,

$$f(i,j) = C(i-1,j-1)*K(1,1) + C(i-1,j)*K(1,2) + C(i-1,j+1)*K(1,3) + \\ C(i,j-1)*K(2,1) + C(i,j)*K(2,2) + C(i,j+1)*K(2,3) + \\ C(i+1,j-1)*K(3,1) + C(i+1,j)*K(3,2) + C(i+1,j+1)*K(3,3)$$

end

end

# KONVOLUIS-4

$$f(x, y) = \begin{bmatrix} 4 & 4 & 3 & 5 & 4 \\ 6 & 6 & 5 & 5 & 2 \\ 5 & 6 & 6 & 6 & 2 \\ 6 & 7 & 5 & 5 & 3 \\ 3 & 5 & 2 & 4 & 4 \end{bmatrix} \quad g(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \bullet 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

———

		3		

$$(0 \times 4) + (-1 \times 4) + (0 \times 3) + (-1 \times 6) + (4 \times 6) + (-1 \times 5) + (0 \times 5) + (-1 \times 6) + (0 \times 6) = 3$$

# KONVOLUIS-5

4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

\_\_\_\_\_

	3	0		

4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

\_\_\_\_\_

	3	0	2	

# KONVOLUIS-6

4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

3	0	2		
0				

4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

4	0	8		
0	2			

# KONVOLUSI-7

4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

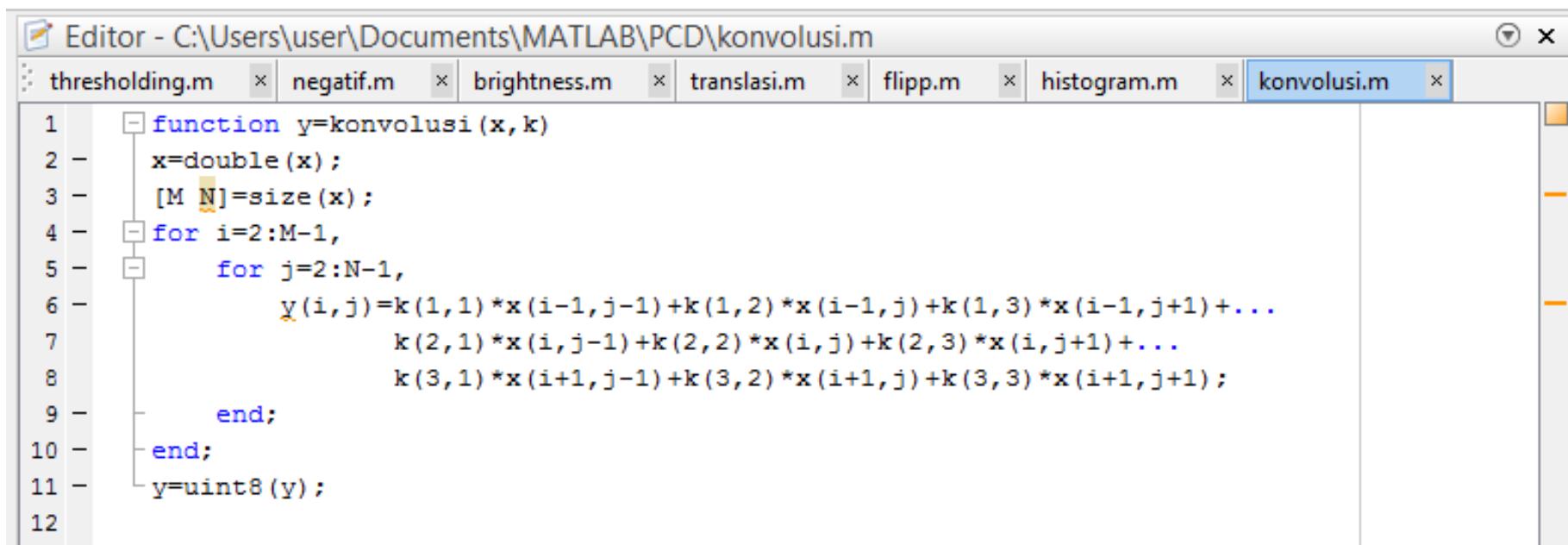
—————

4	0	8		
0	2	6		

Dan seterusnya, dengan cara yang sama akhirnya diperoleh hasil sebagai berikut :

4	0	8		
0	2	6		
6	0	2		

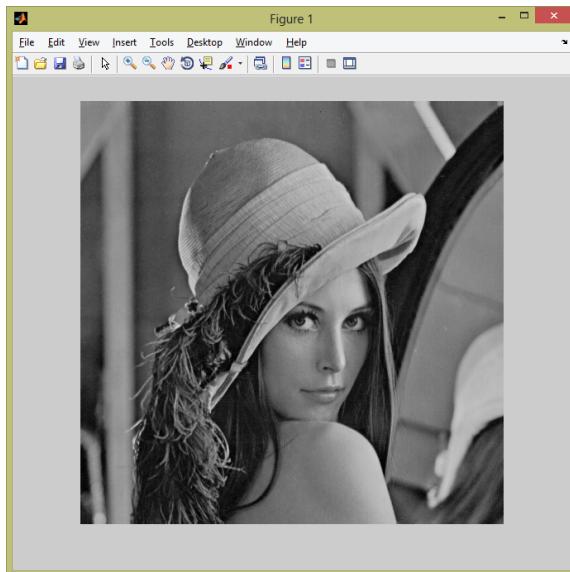
# KONVOLUSI-8



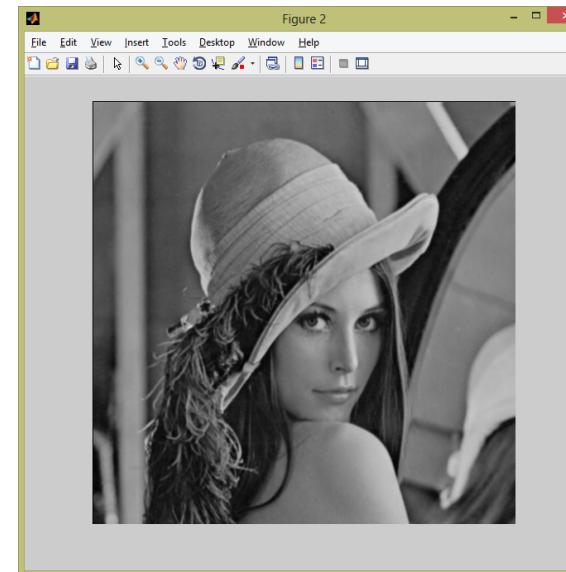
The screenshot shows the MATLAB Editor window with the title "Editor - C:\Users\user\Documents\MATLAB\PCD\konvolusi.m". The tab bar also lists other files: thresholding.m, negatif.m, brightness.m, translasi.m, flipp.m, histogram.m, and konvolusi.m. The code in the editor is as follows:

```
1 function y=konvolusi(x,k)
2 - x=double(x);
3 - [M N]=size(x);
4 - for i=2:M-1,
5 -     for j=2:N-1,
6 -         y(i,j)=k(1,1)*x(i-1,j-1)+k(1,2)*x(i-1,j)+k(1,3)*x(i-1,j+1)+...
7 -                 k(2,1)*x(i,j-1)+k(2,2)*x(i,j)+k(2,3)*x(i,j+1)+...
8 -                 k(3,1)*x(i+1,j-1)+k(3,2)*x(i+1,j)+k(3,3)*x(i+1,j+1);
9 -     end;
10 -    end;
11 -    y=uint8(y);
12
```

# KONVOLUSI-9



 
$$\begin{bmatrix} 0.0625 & 0.1250 & 0.0625 \\ 0.1250 & 0.2500 & 0.1250 \\ 0.0625 & 0.1250 & 0.0625 \end{bmatrix} =$$



Command Window

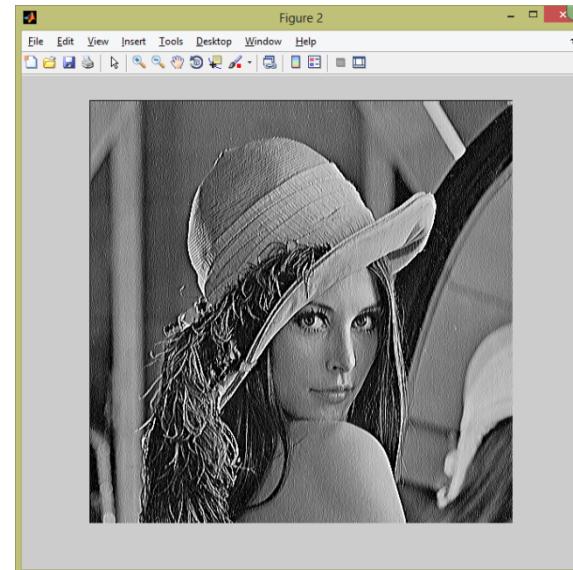
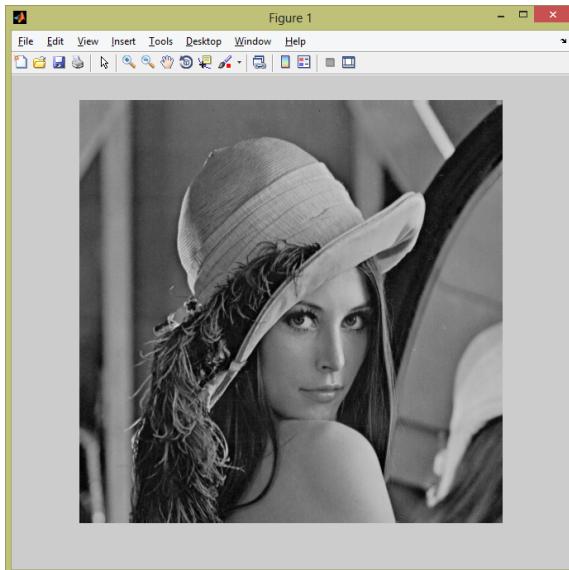
```
>> x=imread('lena.bmp');  
>> k=[1/16 1/8 1/16;1/8 1/4 1/8;1/16 1/8 1/16]
```

k =

```
0.0625 0.1250 0.0625  
0.1250 0.2500 0.1250  
0.0625 0.1250 0.0625
```

```
>> y=konvolusi(x,k);  
>> figure(1);  
>> imshow(x);  
>> figure(2);  
>> imshow(y);  
fx >> |
```

# KONVOLUSI-10



✗  $G = \begin{bmatrix} -1 & 2 & -1 \\ -2 & 5 & -2 \\ -1 & 2 & -1 \end{bmatrix} =$

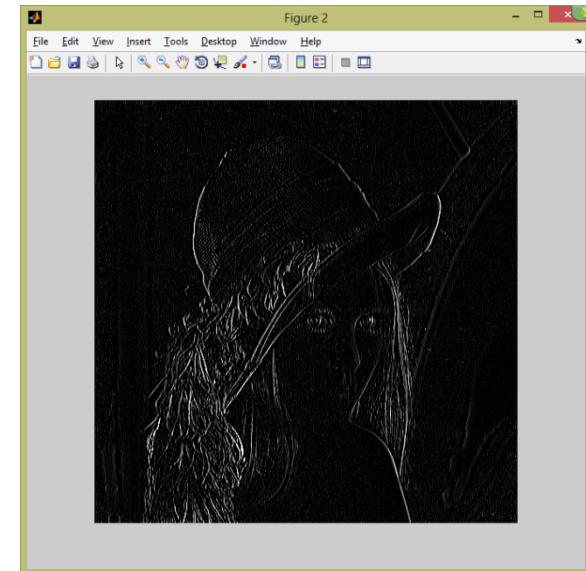
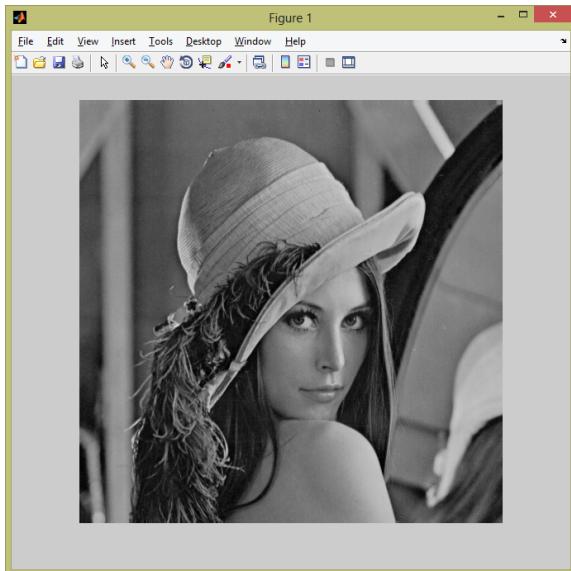
Command Window

```
>> x=imread('lena.bmp');
>> g=[-1 2 -1;-2 5 -2;-1 2 -1]

g =
    -1     2     -1
    -2     5     -2
    -1     2     -1

>> y=konvolusi(x,g);
>> figure(1);
>> imshow(x);
>> figure(2);
>> imshow(y);
fx >> |
```

# KONVOLUSI-11



✗  $G = \begin{bmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{bmatrix} =$

Command Window

```
>> x=imread('lena.bmp');
>> g=[-1 2 -1;-2 4 -2;-1 2 -1]

g =

    -1     2     -1
    -2     4     -2
    -1     2     -1

>> y=konvolusi(x,g);
>> figure(1);
>> imshow(x);
>> figure(2);
>> imshow(y);

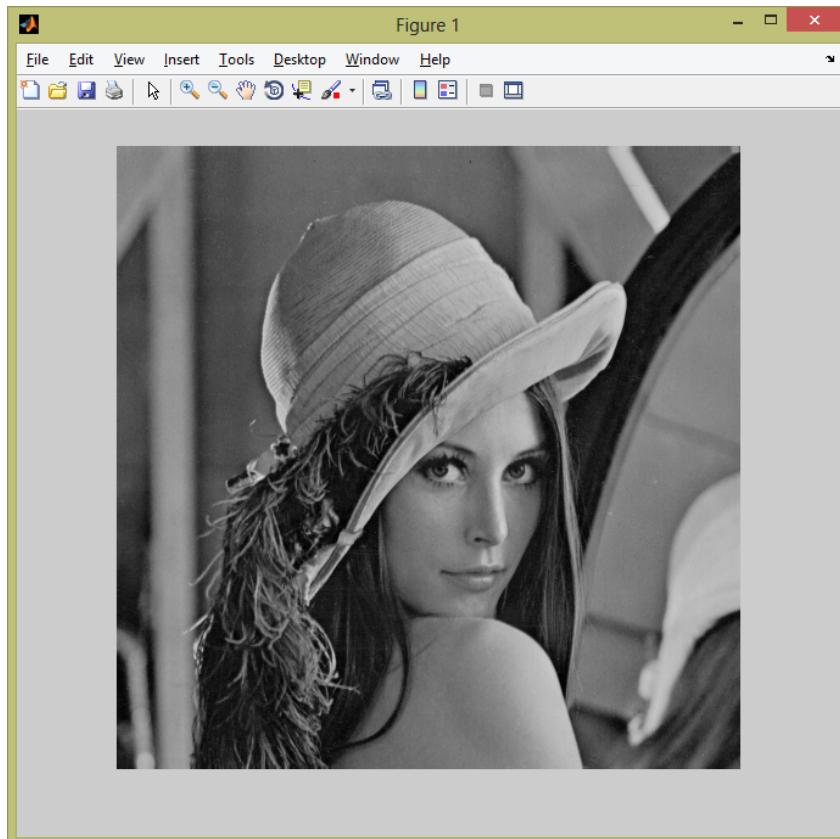
fx >>
```

$G - \begin{vmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{vmatrix}$  ?

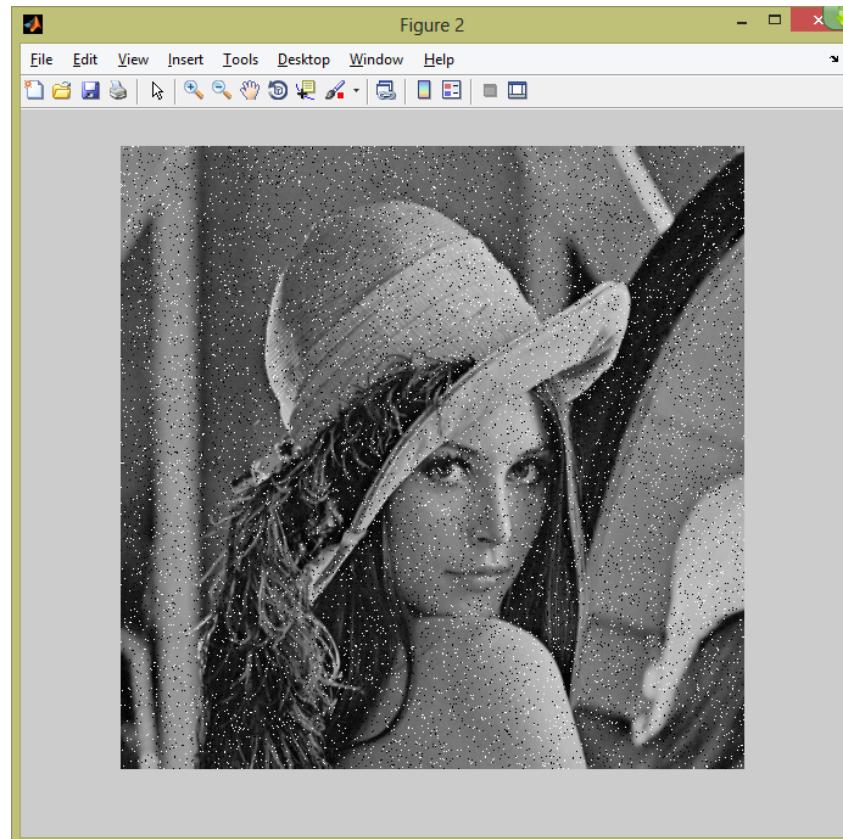
# PELEMBUTAN (SMOOTHING)-1

- Pelembutan citra (*image smoothing*) bertujuan untuk menekan gangguan (*noise*) pada citra.
- Gangguan tersebut biasanya muncul sebagai akibat dari hasil pengambilan gambar yang tidak bagus (*sensor noise, photographic grain noise*) atau akibat saluran transmisi (pada pengiriman data).
- Pada domain spasial, operasi pelembutan dilakukan dengan mengganti intensitas suatu *pixel* dengan rata-rata dari nilai *pixel* tersebut dengan nilai *pixel/pixel* tetangganya.

# PELEMBUTAN (SMOOTHING)-2

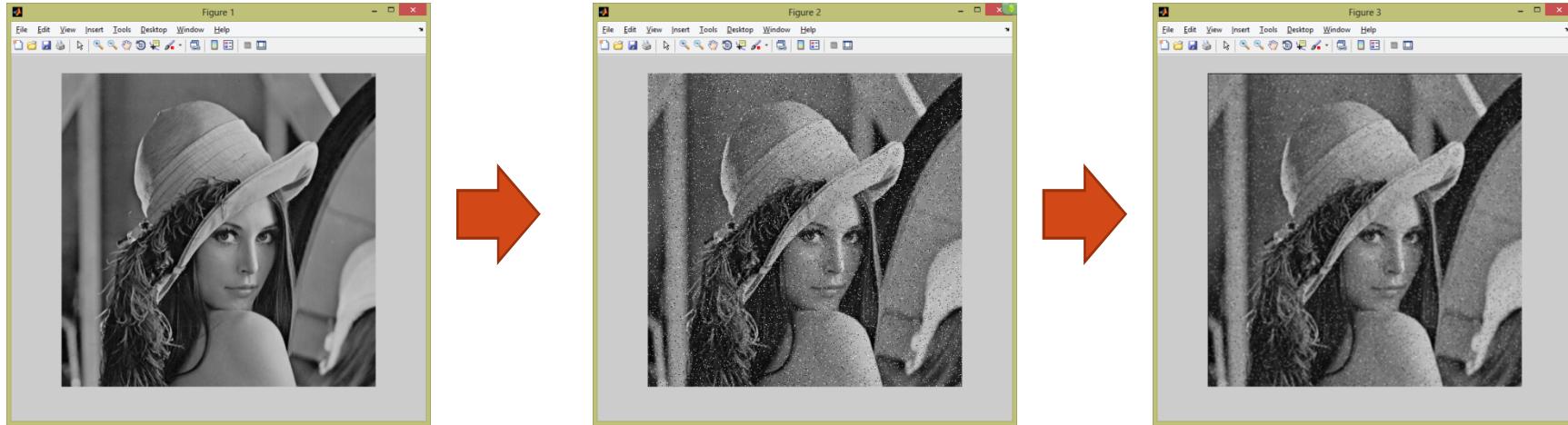


Lena-Normal



Lena-Noisy-'salt & pepper'

# PELEMBUTAN (SMOOTHING)-3



Command Window

```
>> x=imread('lena.bmp');
>> x_noise=imnoise(x,'salt & pepper');
>> g=[1/9 1/9 1/9;1/9 1/9 1/9;1/9 1/9 1/9]

g =

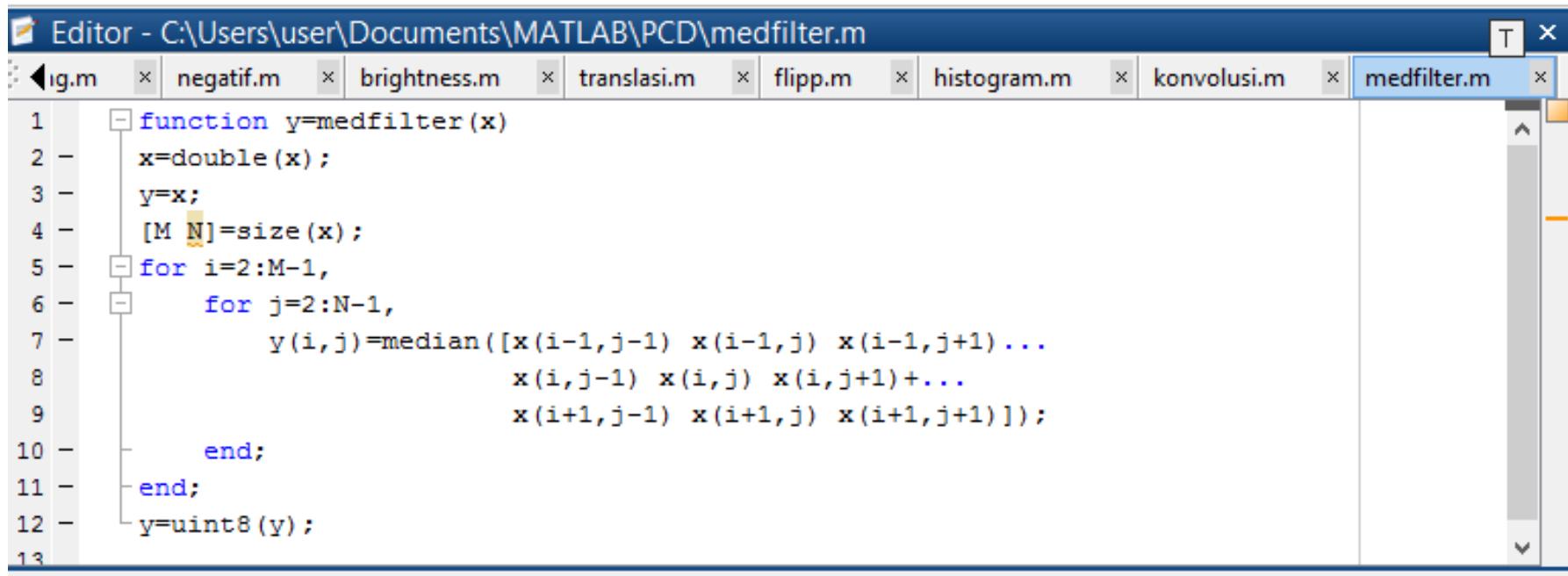
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111

>> y=konvolusi(x_noise,g);
>> figure(1);
>> imshow(x);
>> figure(2);
>> imshow(x_noise);
>> figure(3);
>> imshow(y);
fx >>
```

Kernel rata-rata

# PELEMBUTAN (SMOOTHING)-4

- Selain menggunakan teknik averaging pelembutan dapat juga dilakukan dengan menggunakan teknik median filtering.
- Teknik ini dilakukan dengan mengganti nilai pusat sliding window dengan menggunakan nilai median dari piksel-piksel tetanggannya.



The screenshot shows the MATLAB Editor window with the title bar "Editor - C:\Users\user\Documents\MATLAB\PCD\medfilter.m". The tab bar below the title bar lists several other files: ig.m, negatif.m, brightness.m, translasi.m, flipp.m, histogram.m, konvolusi.m, and medfilter.m. The "medfilter.m" tab is currently selected. The main editor area displays the following MATLAB code:

```
1 function y=medfilter(x)
2 x=double(x);
3 y=x;
4 [M N]=size(x);
5 for i=2:M-1,
6 for j=2:N-1,
7 y(i,j)=median([x(i-1,j-1) x(i-1,j) x(i-1,j+1) ...
8 x(i,j-1) x(i,j) x(i,j+1)+...
9 x(i+1,j-1) x(i+1,j) x(i+1,j+1)]);
10 end;
11 end;
12 y=uint8(y);
13
```

# PELEMBUTAN (SMOOTHING)-5



Averaging



Median Filtering



Lena-Noisy-'salt & pepper'

# PENAJAMAN GAMBAR-1

- Operasi penajaman citra bertujuan memperjelas tepi pada objek di dalam citra.
- Penajaman citra merupakan kebalikan dari operasi pelembutan citra karena operasi ini menghilangkan bagian citra yang lembut.
- Operasi penajaman dilakukan dengan melewatkannya citra pada **filter lolos-tinggi** (*high-pass filter*).

## PENAJAMAN GAMBAR-2

- *Penapis lolos-tinggi akan meloloskan (atau memperkuat) komponen yang berfrekuensi tinggi (misalnya tepi atau pinggiran objek) dan akan menurunkan komponen berfrekuensi rendah. Akibatnya, pinggiran objek telihat lebih tajam dibandingkan sekitarnya.*
- Karena penajaman citra lebih berpengaruh pada tepi (*edge*) objek, maka penajaman citra sering disebut juga **penajaman tepi (*edge sharpening*) atau** peningkatan kualitas tepi (*edge enhancement*).

# PENAJAMAN GAMBAR-3

Kriteria filter lolos tinggi :

1. koefisien filter boleh positif, negatif, atau nol
2. jumlah semua koefisien adalah 0 atau 1
  - Jika jumlah koefisien = 0, maka komponen berfrekuensi rendah akan turun nilainya, sedangkan jika jumlah koefisien sama dengan 1, maka komponen berfrekuensi rendah akan tetap sama dengan nilai semula.

# PENAJAMAN GAMBAR-4

- Filter lolos tinggi

$$(i) \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$
$$\Sigma = 0$$

$$(ii) \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$
$$\Sigma = 1$$

$$(iii) \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$
$$\Sigma = 1$$

$$(iv) \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$
$$\Sigma = 1$$

$$(v) \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$
$$\Sigma = 0$$

$$(vi) \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$
$$\Sigma = 0$$

# PENAJAMAN GAMBAR-5



**X** 
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$



```
Command Window
>> x=imread('mandril128.bmp');
>> K=[-1 -1 -1;-1 9 -1;-1 -1 -1]

K =
-1     -1     -1
-1      9     -1
-1     -1     -1

>> y=konvolusi(x,K);
>> figure(1)
>> imshow(x);
>> figure(2)
>> imshow(y);

fx >>
```

# PENAJAMAN GAMBAR-6



$\times$  
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$



```
Command Window
>> x=imread('mandrill128.bmp');
>> K2=[-1 -1 -1;-1 8 -1;-1 -1 -1]

K2 =
-1     -1     -1
-1      8     -1
-1     -1     -1

>> y=konvolusi(x,K2);
>> figure(1)
>> imshow(x);
>> figure(2)
>> imshow(y);

fx >> |
```

# PEREGANGAN KONTRAS-1

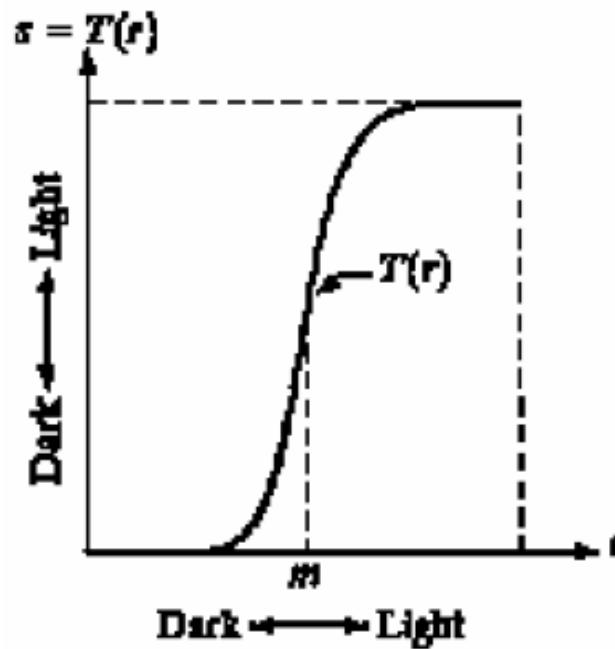
- Kontras menyatakan sebaran terang (*lightness*) dan gelap (*darkness*) *di dalam* sebuah gambar.
- Citra dapat dikelompokkan ke dalam tiga kategori kontras: citra kontras-rendah (*low contrast*), *citra kontras-bagus (good contrast atau normal contrast)*, dan *citra kontras-tinggi (high contrast)*.

## PEREGANGAN KONTRAS-2

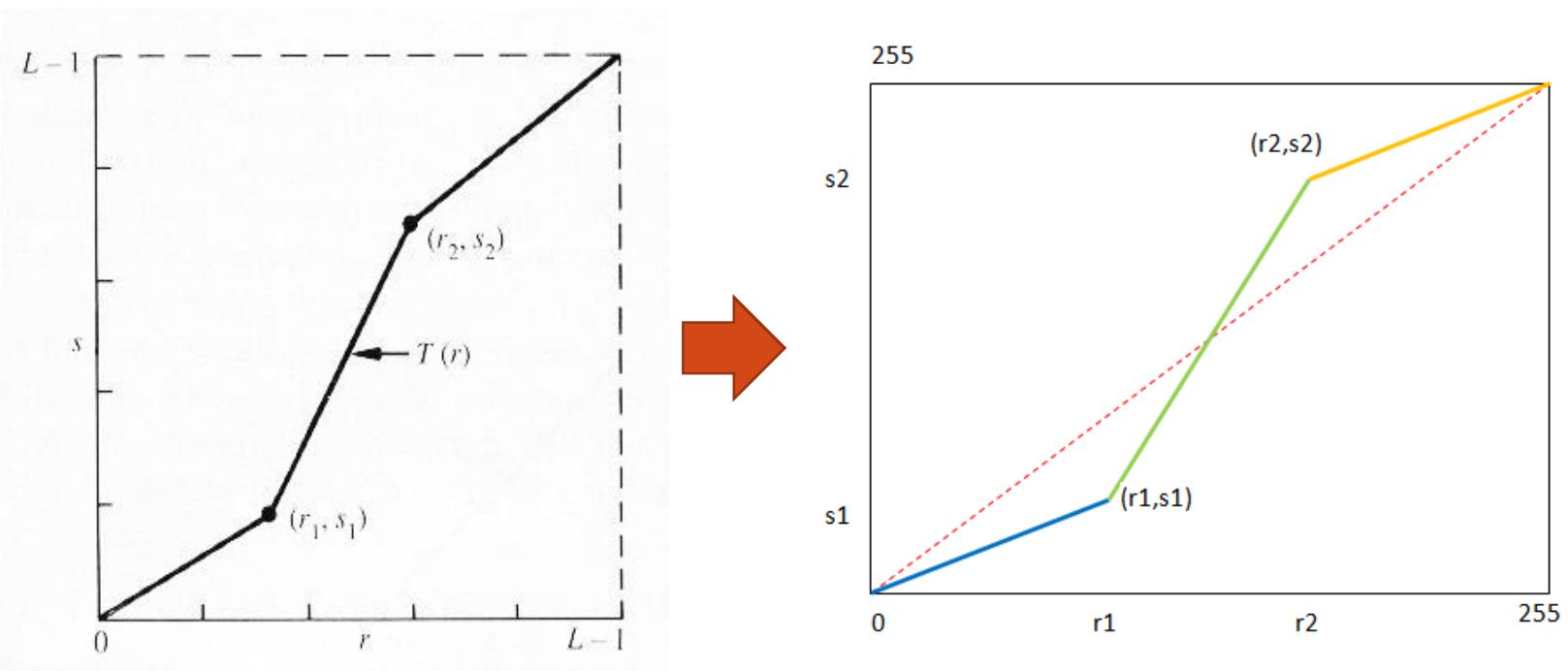
- Citra kontras-rendah dicirikan dengan sebagian besar komposisi citranya adalah terang atau sebagian besar gelap.
- Citra kontras-bagus memperlihatkan jangkauan nilai keabuan yang lebar tanpa ada suatu nilai keabuan yang mendominasi. Histogram citranya memperlihatkan sebaran nilai keabuan yang relatif seragam.
- Citra kontras-tinggi, seperti halnya citra kontras bagus, memiliki jangkauan nilai keabuan yang lebar, tetapi terdapat area yang lebar yang didominasi oleh warna gelap dan area yang lebar yang didominasi oleh warna terang.

# PEREGANGAN KONTRAS-3

- Menghasilkan nilai contrast yang lebih besar dari image original, dengan cara :
  - Menggelapkan (darkening) level dibawah m dari image asli.
  - Mencerahkan (Brightening) level atas m dari image asli.

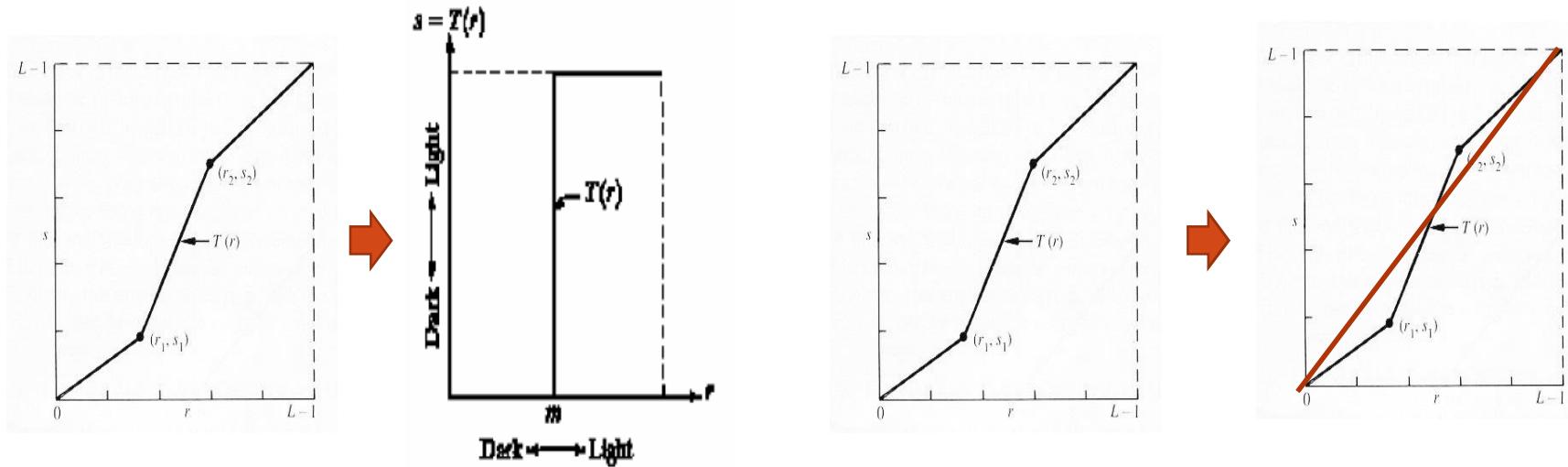


# PEREGANGAN KONTRAS-4



Lokasi  $(r_1, s_1)$  dan  $(r_2, s_2)$  menjadi kontrol dari bentuk fungsi transformasi.

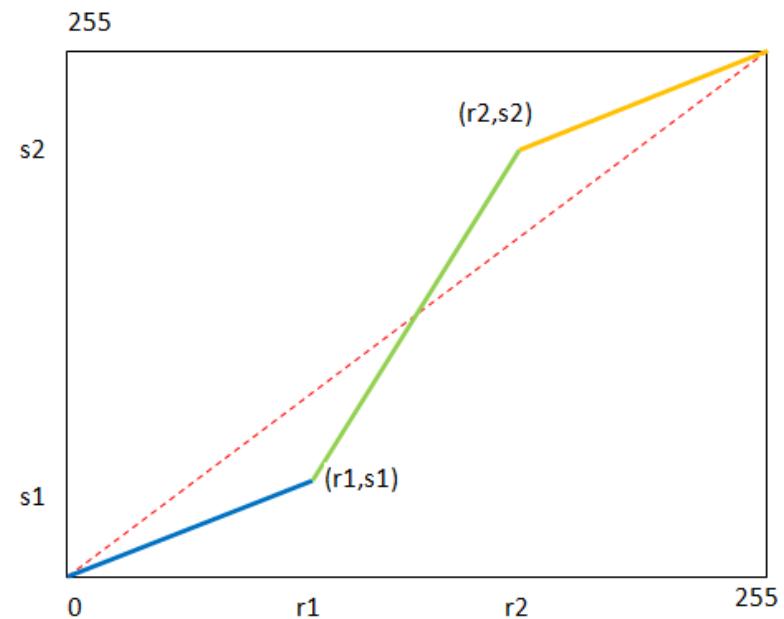
# PEREGANGAN KONTRAS-5



- Lokasi dari  $(r_1, s_1)$  dan  $(r_2, s_2)$  menjadi kontrol dari bentuk fungsi transformasi.
  - Bila  $r_1 = s_1$  dan  $r_2 = s_2$  transformasi adalah fungsi linear dan hasilnya adalah tidak ada perubahan image.
  - Bila  $r_1 = r_2$ ,  $s_1 = 0$  dan  $s_2 = L-1$ , transformasi akan berubah menjadi sebuah fungsi thresholding yang menghasilkan sebuah binary image.

# PEREGANGAN KONTRAS-6

- Untuk menyelesaikan peregangan kontras berdasarkan grafik disamping, maka perlu dicari 3 persamaan garis.
  - Garis biru, melalui titik  $(0,0)$  dan  $(r_1, s_1)$
  - Garis hijau, melalui titik  $(r_1, s_1)$  dan  $(r_2, s_2)$
  - Garis kuning, melalui titik  $(r_2, s_2)$  dan  $(255, 255)$



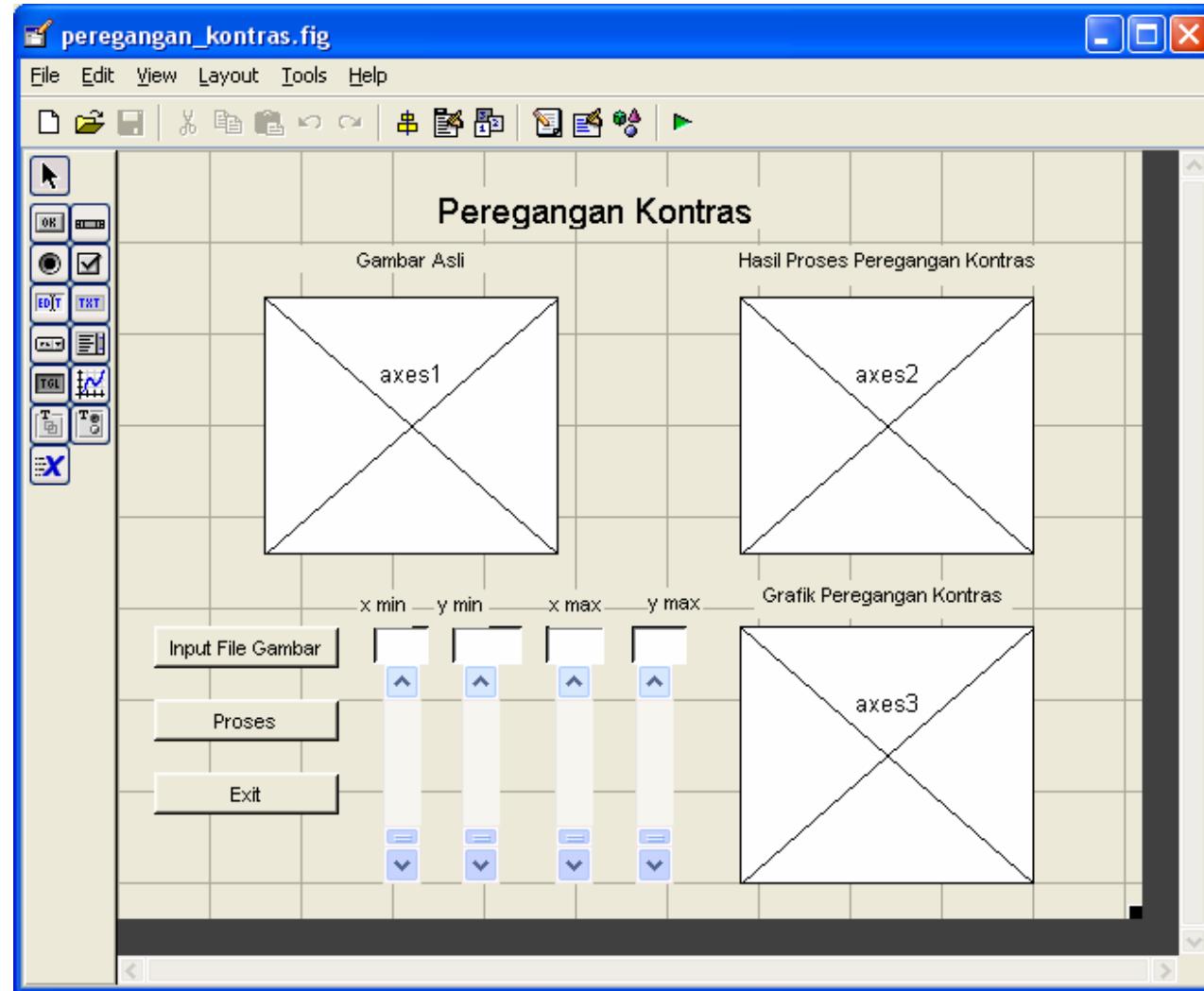
*Solusinya hanya persamaan garis !*  
$$(y-y_1)/(y_2-y_1)=(x-x_1)/(x_2-x_1)$$

# PEREGANGAN KONTRAS-7

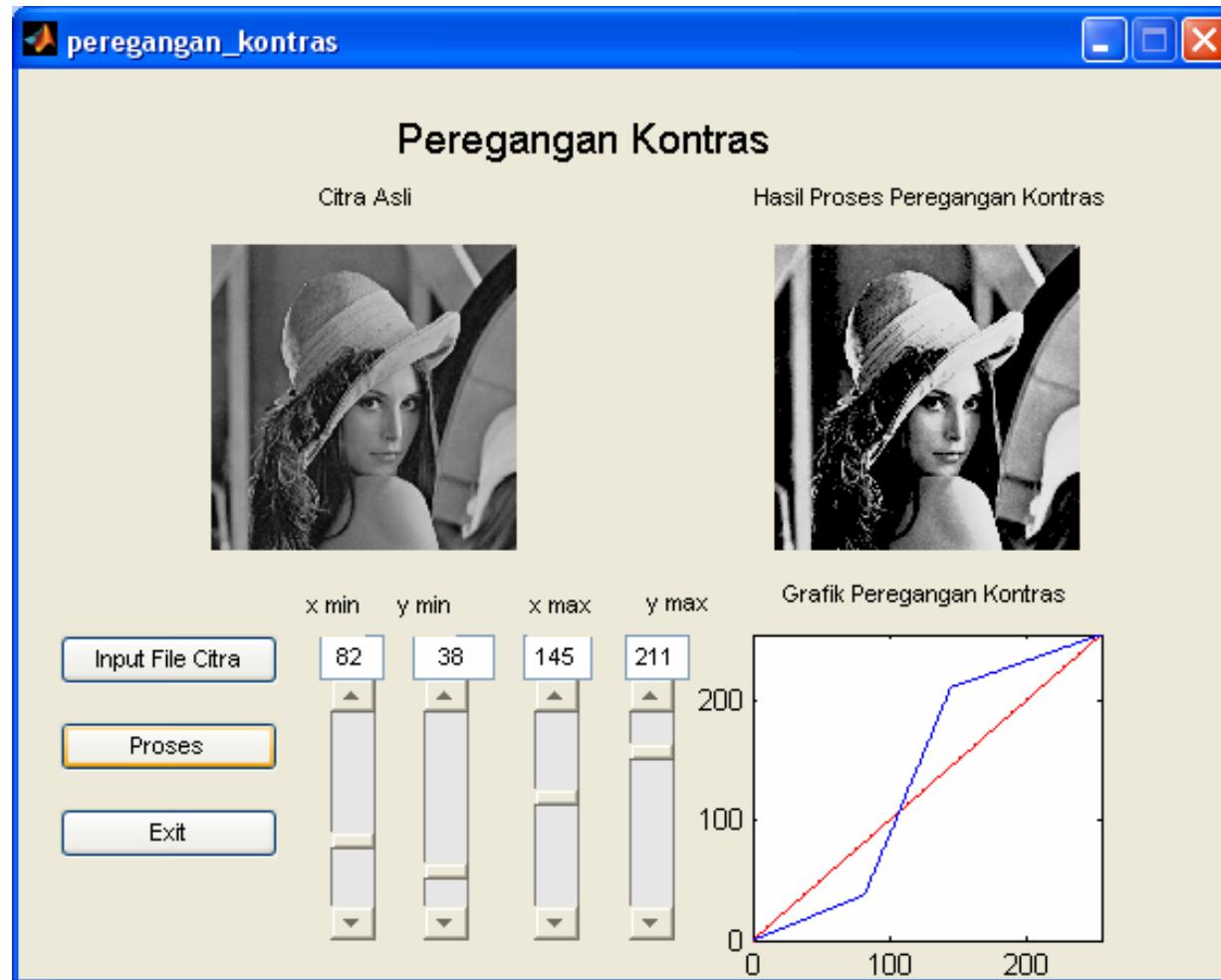
## Algorithma :

- Input citra
- Tentukan ukuran citra
- Untuk tiap-tiap piksel citra
  - Jika piksel citra  $< r1$  maka gunakan persamaan garis 1 (biru)
  - Jika piksel citra  $> r2$  maka gunakan persamaan garis 3 (kuning)
  - Selain itu gunakan persamaan garis 2 (hijau)

# PEREGANGAN KONTRAS-8



# PEREGANGAN KONTRAS-9



# PENGOLAHAN CITRA DIGITAL

EDGE DETECTION

GKV - IF 2020

# OUTLINE

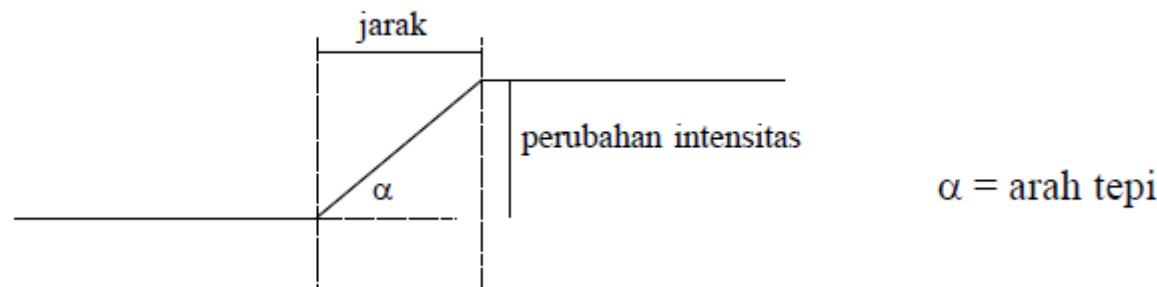
- LATAR BELAKANG
- PENGERTIAN TEPI
- OPERATOR
- ALGORITMA TEPI
- IMPLEMENTASI TEPI

# LATAR BELAKANG

- Faktor kunci dalam mengekstraksi ciri adalah kemampuan mendeteksi keberadaan tepi (*edge*) dari objek *di dalam citra*.
- *Setelah tepi objek diketahui, langkah selanjutnya dalam analisis citra adalah segmentasi*, yaitu mereduksi citra menjadi objek atau region, misalnya memisahkan objek-objek yang berbeda dengan mengekstraksi batas-batas objek (*boundary*).
- *Langkah terakhir dari analisis citra adalah klasifikasi*, yaitu memetakan segmen-segmen yang berbeda ke dalam kelas objek yang berbeda pula.

# PENGERTIAN TEPI (EDGE)

- tepi (*edge*) adalah perubahan nilai intensitas derajat keabuan yang mendadak (besar) dalam jarak yang singkat.
- Perbedaan intensitas inilah yang menampakkan rincian pada gambar.
- Tepi biasanya terdapat pada batas antara dua daerah berbeda pada suatu citra.
- Tepi dapat diorientasikan dengan suatu arah, dan arah ini berbeda-beda pada bergantung pada perubahan intensitas.



# TIPE TEPI - 1

- **Tepi curam**

Tepi dengan perubahan intensitas yang tajam. Arah tepi berkisar  $90^\circ$ .

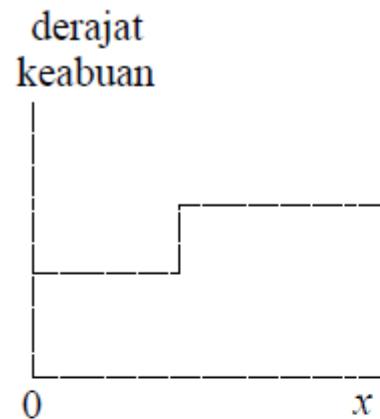
- **Tepi landai**

Disebut juga tepi lebar, yaitu tepi dengan sudut arah yang kecil. Tepi landai dapat dianggap terdiri dari sejumlah tepi-tepi lokal yang lokasinya berdekatan.

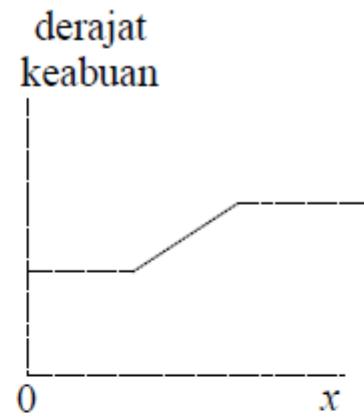
- **Tepi yang mengandung derau (*noise*)**

Umumnya tepi yang terdapat pada aplikasi *computer vision* mengandung derau. Operasi peningkatan kualitas citra (*image enhancement*) dapat dilakukan terlebih dahulu sebelum pendeksiian tepi.

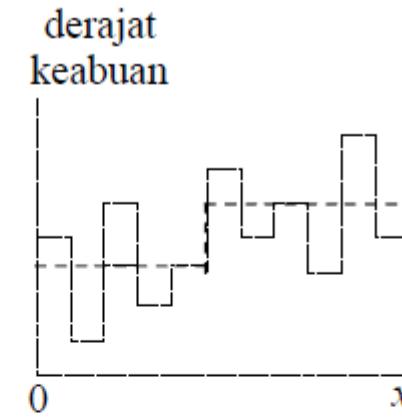
# TIPE TEPI - 2



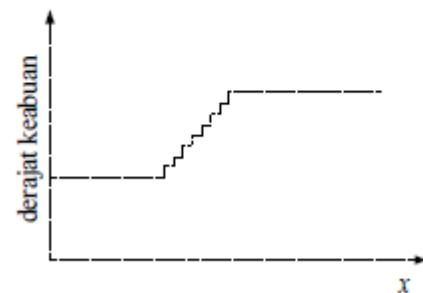
(a) Tepi curam



(b) tepi landai



(c) tepi curam dengan derau



(d) *break down* tepi landai

4	4	4	8	8	8	8	8	8
4	4	4	8	8	8	8	8	8
4	4	4	8	8	8	8	8	8
4	4	4	8	8	8	8	8	8
4	4	4	8	8	8	8	8	8

(e) citra dengan tepi curam

4	4	5	6	7	8	8	8	8
4	4	5	6	7	8	8	8	8
4	4	5	6	7	8	8	8	8
4	4	5	6	7	8	8	8	8
4	4	5	6	7	8	8	8	8

(f) citra dengan tepi landai

# TUJUAN

- Tujuan operasi deteksi tepi adalah untuk meningkatkan penampakan garis batas suatu daerah atau objek di dalam citra. Karena tepi termasuk ke dalam komponen berfrekuensi tinggi, maka pendektsian tepi dapat dilakukan dengan filter lolos-tinggi.

# TEKNIK DETEKSİ TEPI

- Terdapat beberapa teknik yang digunakan untuk mendeteksi tepi, antara lain:
  1. Operator gradien pertama (*differential gradient*)
  2. Operator turunan kedua (*Laplacian*)
  3. Operator kompas (*compass operator*)

# Operator gradien pertama

- Perubahan intensitas yang besar dalam jarak yang singkat dipandang sebagai fungsi yang memiliki kemiringan yang besar.
- Kemiringan fungsi biasanya dilakukan dengan menghitung turunan pertama (*gradient*).
- *Karena citra  $f(x,y)$  adalah fungsi 2 dimensi dalam bentuk diskrit, maka turunan pertamanya adalah secara parsial, masing-masing dalam arah- $x$  dan dalam arah- $y$ .*

# Operator gradien pertama-2

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} G_x \\ G_y \end{bmatrix}$$

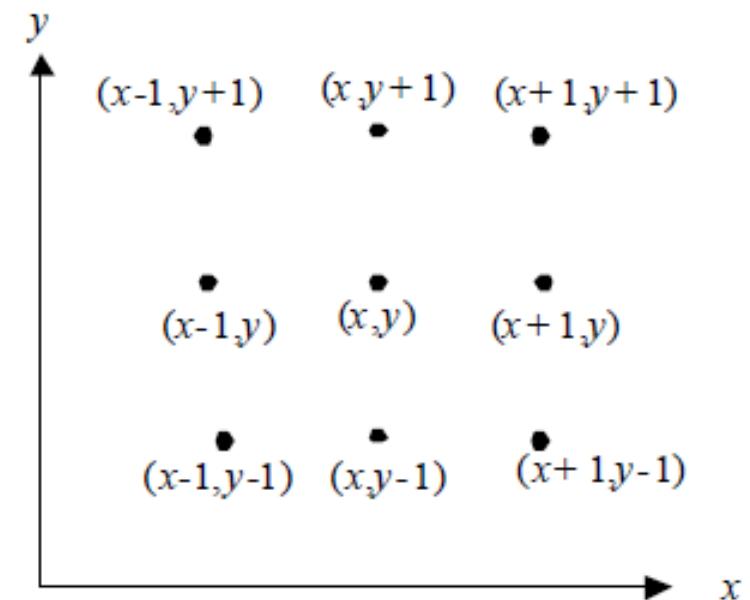
$$G_x = \frac{\partial f(x, y)}{\partial x} = \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x}$$

$$G_y = \frac{\partial f(x, y)}{\partial y} = \frac{f(x, y + \Delta y) - f(x, y)}{\Delta y}$$

$$\Delta x = \Delta y = 1$$

$$G_x = \frac{\partial f(x, y)}{\partial x} = f(x + 1, y) - f(x, y)$$

$$G_y = \frac{\partial f(x, y)}{\partial y} = f(x, y + 1) - f(x, y)$$



$$G_1(x) = \begin{bmatrix} -1 & 1 \end{bmatrix} \quad \text{dan} \quad G_1(y) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

# Operator gradien pertama-3

- Operator gradien selisih-terpusat (*center-difference*):

$$D_x(x, y) = \frac{\partial f(x, y)}{\partial x} = \frac{f(x+1, y) - f(x-1, y)}{2}$$

$$D_y(x, y) = \frac{\partial f(x, y)}{\partial y} = \frac{f(x, y+1) - f(x, y-1)}{2}$$

$$D_2(x) = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \text{dan} \quad D_2(y) = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

# Operator gradien pertama-4

- Operator Sobel

$$\begin{bmatrix} a_0 & a_1 & a_2 \\ a_7 & (x, y) & a_3 \\ a_6 & a_5 & a_4 \end{bmatrix}$$

$$S_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$

$$S_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4)$$

$$c = 2$$

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{dan} \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

# Operator gradien pertama-5

- Operator Prewitt

$$\begin{bmatrix} a_0 & a_1 & a_2 \\ a_7 & (x, y) & a_3 \\ a_6 & a_5 & a_4 \end{bmatrix}$$

$$s_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$

$$s_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4)$$

$$c = 1$$

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{dan} \quad P_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

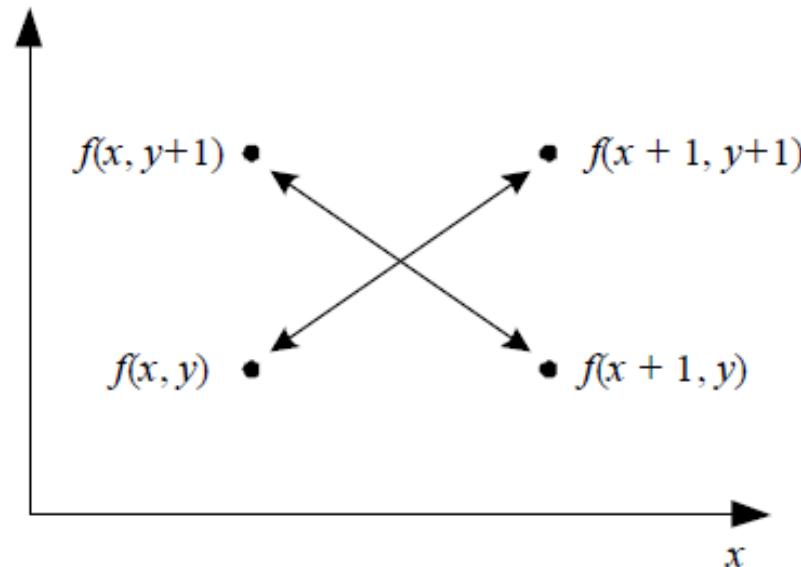
# Operator gradien pertama-5

- Operator Roberts
  - Sering disebut juga operator silang. Gradien Roberts dalam arah- $x$  dan arah- $y$  dihitung dengan rumus:

$$R_+(x, y) = f(x + 1, y + 1) - f(x, y)$$

$$R_-(x, y) = f(x, y + 1) - f(x + 1, y)$$

$$R_+ = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \text{ dan } R_- = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$



- Magnitude Tepi  $G[f(x,y)] = |R_+| + |R_-|$

# Operator Turunan Ke-dua

- Operator turunan kedua disebut juga operator Laplace. Operator Laplace mendeteksi lokasi tepi lebih akurat khususnya pada tepi yang curam.
- Pada tepi yang curam, turunan keduanya mempunyai persilangan nol (*zero-crossing*), *yaitu* titik di mana terdapat pergantian tanda nilai turunan kedua sedangkan pada tepi yang landai tidak terdapat persilangan nol.
- Persilangan nol merupakan lokasi tepi yang akurat.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

# Operator Turunan Ke-dua

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$= \frac{f(x + \Delta x, y) - 2f(x, y) + f(x - \Delta x, y)}{(\Delta x)^2} + \frac{f(x, y + \Delta y) - 2f(x, y) + f(x, y - \Delta y)}{(\Delta y)^2}$$

$$\Delta x = \Delta y = 1$$

$$\begin{aligned}\nabla^2 f(x, y) &= f(x+1, y) - 2f(x, y) + f(x-1, y) + f(x, y+1) - 2f(x, y) + f(x, y-1) \\ &= f(x, y-1) + f(x-1, y) - 4f(x, y) + f(x+1, y) + f(x, y+1)\end{aligned}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\text{ dan } \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

# Operator Kompas

- Operator kompas (*compass operator*) digunakan untuk mendeteksi semua tepi dari berbagai arah di dalam citra.
- Operator kompas yang dipakai untuk pendekslsian tepi menampilkan tepi dari 8 macam arah mata angin: Utara, Timur Laut, Timur, Tenggara, Selatan, Barat Daya, dan Barat Laut.
- Pendekslsian tepi dilakukan dengan mengkonvolusikan citra dengan berbagai *mask kompas*, lalu dicari nilai kekuatan tepi (*magnitude*) yang terbesar dan arahnya.

# Operator Kompas

- Operator Kompas berdasarkan arah mata angin

Utara

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

Timur Laut

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

Timur

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

Tenggara

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

Selatan

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

Barat Daya

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

Barat

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

Barat Laut

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

- Magnitude Tepi

$$G[f(x, y)] = \max_i \{G_i[f(x, y)] | i = 1, 2, \dots, p\}$$

# ALGORITMA DETEKSI TEPI

1. Inputkan citra ( $x$ ), kernel operator ( $k_x, k_y$ ), parameter ambang ( $T$ )
2. Tentukan ukuran citra ( $M, N$ )
3. Konvolusi citra dengan kernel  $k_x$  ( $D_x$ )
4. Konvolusi citra dengan kernel  $k_y$  ( $D_y$ )
5. Hitung kekuatan tepi citra  $K[f(x,y)] = |D_x| + |D_y|$
6. Lakukan Proses Thresholding terhadap  $K[f(x,y)]$

# IMPLEMENTASI - Prewitt

Editor - C:\Users\user\Documents\MATLAB\PCD\tepi.m

```
1 function y=tepi(x,kx,ky,T)
2 Dx=konvolusi(x,kx);
3 Dy=konvolusi(x,ky);
4 KT=abs(Dx)+abs(Dy);
5 y=thresholding(KT,T);
6
```

Command Window

```
>> x=imread('lena.bmp');
>> Px=[-1 0 1;-1 0 1;-1 0 1]

Px =
-1 0 1
-1 0 1
-1 0 1

>> Py=[1 1 1;0 0 0;-1 -1 -1]

Py =
1 1 1
0 0 0
-1 -1 -1

>> y=tepi(x,Px,Py,128);
>> figure(1)
>> imshow(x);
>> figure(2)
>> imshow(y);
>>
```

A grayscale image of a woman wearing a hat, used as the input for edge detection.The output image showing the edges detected by the Prewitt operator, appearing as white lines on a black background.

# IMPLEMENTASI - Sobel

Command Window

```
>> x=imread('lena.bmp');
>> Sx=[-1 0 1;-2 0 2;-1 0 1]

Sx =
-1     0     1
-2     0     2
-1     0     1

>> Sy=[1 2 1;0 0 0;-1 -2 -1]

Sy =
1     2     1
0     0     0
-1    -2    -1

>> y=tепи(x,Sx,Sy,128);
>> figure(1)
>> imshow(x);
>> figure(2)
>> imshow(y);

fx >>
```



Prewitt

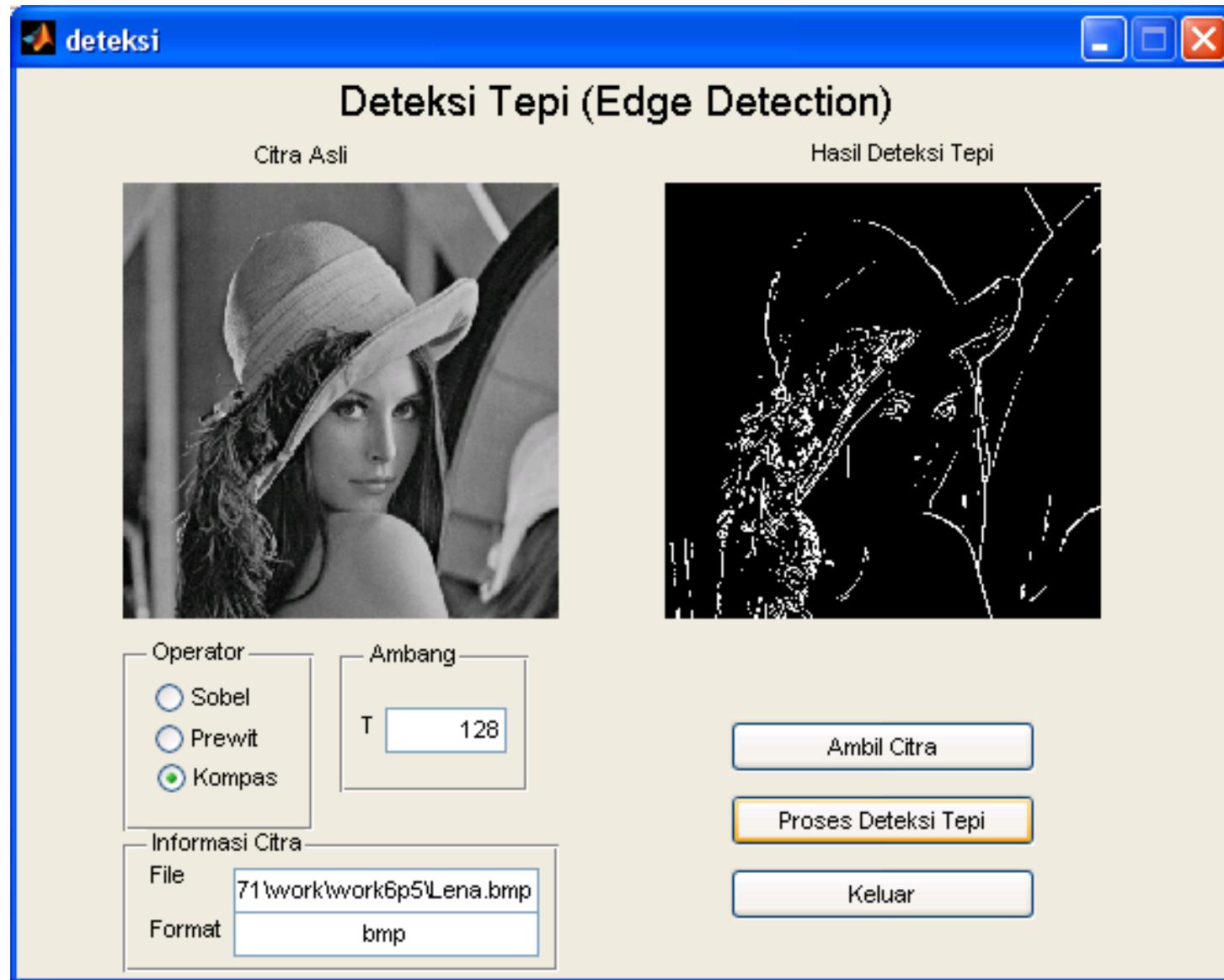


Sobel

# IMPLEMENTASI-Kompas

- Algoritma :
  1. Inputkan citra ( $x$ ), kernel operator ( $Tr, Tg, S, BD, B, BL, U, TL$ ), parameter ambang ( $T$ )
  2. Tentukan ukuran citra ( $M, N$ )
  3. Konvolusi citra dengan kernel  $Tr(K1), Tg(K2), S(K3), BD(K4), B(K5), BL(K6), U(K7), TL(K8)$ .
  4. Hitung kekuatan tepi citra  $K[f(x,y)] = \max(K1, K2, \dots, K8)$
  5. Lakukan Proses Thresholding terhadap  $K[f(x,y)]$

# IMPLEMENTASI-Kompas



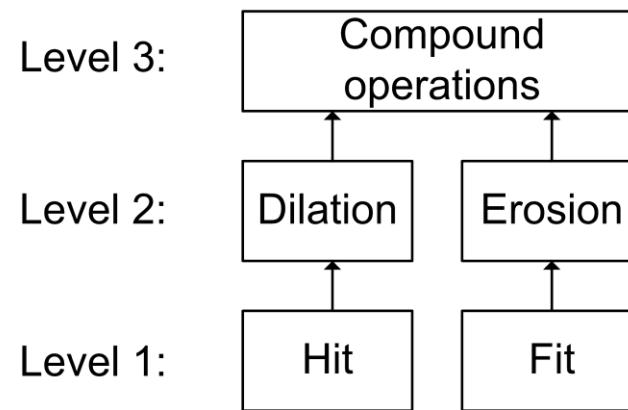
# PENGOLAHAN CITRA DIGITAL

IMAGES MORPHOLOGY

GKV - IF 2020

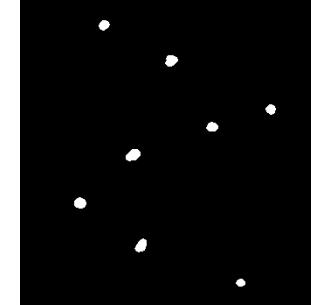
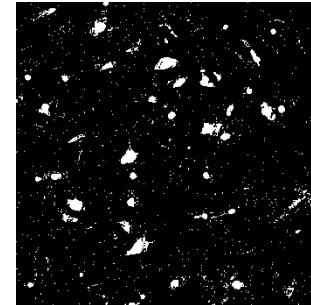
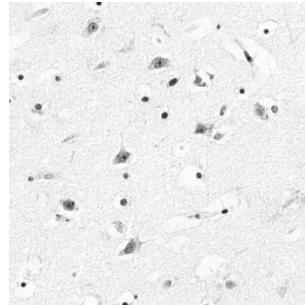
# Morfologi Citra

- Apa yang bisa dilakukan oleh morfologi citra ?
- Operasi morfologi :
  - Fit dan Hit
  - Erosi (Erosion)
  - Dilasi (Dilation)
  - Operasi Gabungan (Compound Operations)



# Manfaat Morfologi

- Remove Noise
  - Small Objects

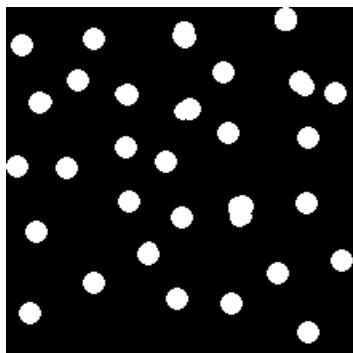
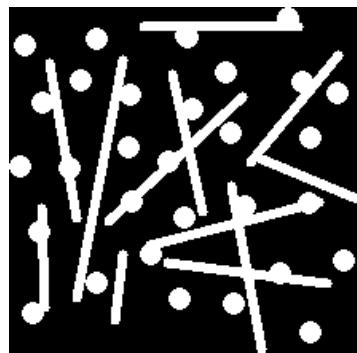
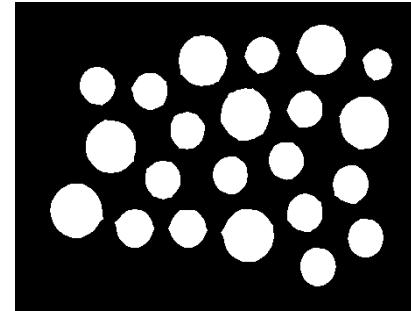
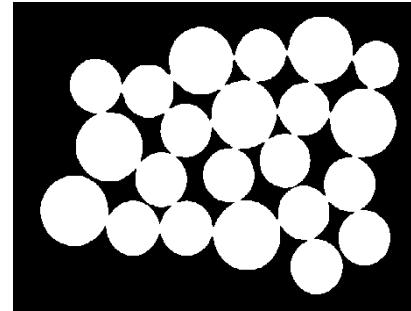
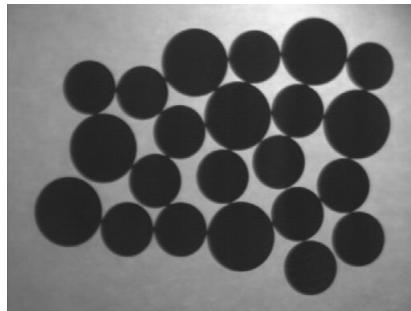


- Fill holes



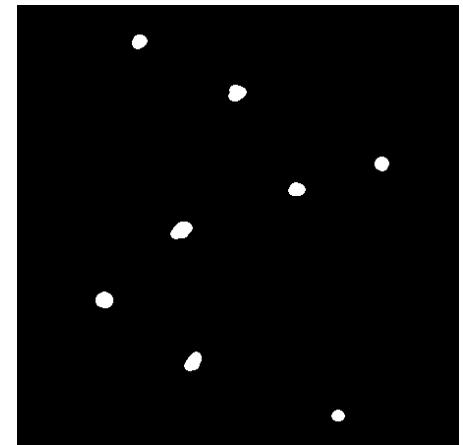
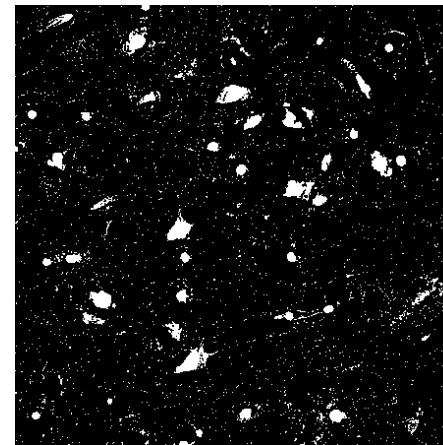
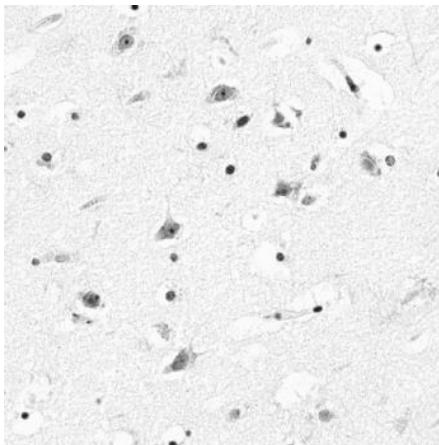
# Manfaat Morfologi

- Isolate Objects



# Cara Kerja MorfologiCitra

- Konversi citra ke dalam bentuk Grayscale
- Lakukan binerisasi citra → Thresholding
- Morfologi



- Dapat juga diterapkan pada citra grayscale

# Hit dan Fit untuk Citra1D / Vektor

Input image

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



Structuring  
Element (SE)

1	1	1
---	---	---

Desain SEsesuai  
keinginan



Output Image

		0/1							
--	--	-----	--	--	--	--	--	--	--

**Hit:** If just one of the '1's in the SE match with the input

⇒ output = 1, otherwise output = 0

**Fit :** If all '1's in the SE match with input

⇒ output = 1, otherwise output = 0

# Dilasi (Dilation) berdasarkan Operasi Hit

Input image

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



Structuring  
Element (SE)

1	1	1
---	---	---

$$g(x) = f(x) \oplus SE$$



Output Image

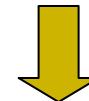
	1								
--	---	--	--	--	--	--	--	--	--

**Hit : If just one of the '1's in the SE match with the input => output =1, otherwise output =0**

# Contoh Dilasi

Input

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



SE

1	1	1
---	---	---



Output

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

# Contoh Dilasi

Input

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



S E

1	1	1
---	---	---



Output

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

# Contoh Dilasi

Input

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



S E

1	1	1
---	---	---



Output

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

# Contoh Dilasi

Input

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



S E

1	1	1
---	---	---



Output

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

# Contoh Dilasi

Input

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



S E

1	1	1
---	---	---



Output

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

# Contoh Dilasi

Input

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



S E

1	1	1
---	---	---



Output

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

## Contoh Dilasi

Input

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



S E

1	1	1
---	---	---



Output

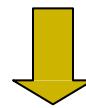
	1	0	1	1	1	1	1	1	
--	---	---	---	---	---	---	---	---	--

**Object (1) menjadi lebih besar dan holes (0) menjadi terisi dengan object atau hilang**

# Erosi (Erosion) berdasarkan Operasi Fit

Input image

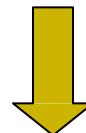
1	0	0	0	1	1	1	0	1	1
---	---	---	---	---	---	---	---	---	---



Structuring  
Element (SE)

1	1	1
---	---	---

$$g(x) = f(x) \text{O-SE}$$



Output Image

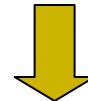
	0								
--	---	--	--	--	--	--	--	--	--

**Fit If all '1's in the SE match with input =>  
output =1, otherwise output =0**

# Contoh Erosi

Input

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



SE

1	1	1
---	---	---



Output

	0	0							
--	---	---	--	--	--	--	--	--	--

# Contoh Erosi

Input

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



S E

1	1	1
---	---	---



Output

	0	0	0						
--	---	---	---	--	--	--	--	--	--

# Contoh Erosi

Input

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



S E

1	1	1
---	---	---



Output

	0	0	0	0					
--	---	---	---	---	--	--	--	--	--

## Contoh Erosi

Input

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



S E

1	1	1
---	---	---



Output

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

# Contoh Erosi

Input

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



S E

1	1	1
---	---	---



Output

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

# Contoh Erosi

Input

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



S E

1	1	1
---	---	---



Output

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

# Contoh Erosi

Input

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



S E

1	1	1
---	---	---



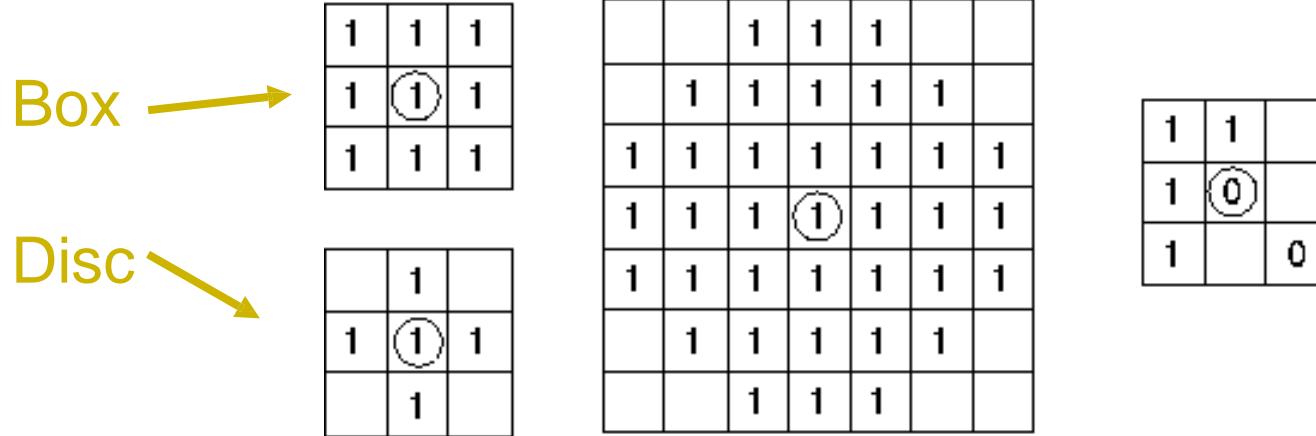
Output

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

**Object (1) menjadi lebih kecil**

# Morfologi Citra

- Structuring Elements (SE) dapat terdiri dari sebarang ukuran sesuai dengan kebutuhan
- Nilai dari elemen adalah **0**atau **1**, namun dimungkinkan memiliki nilai yang lain (termasuk tidak ada nilainya)
- Nilai kosong pada SE berarti bebas (*don't care*)

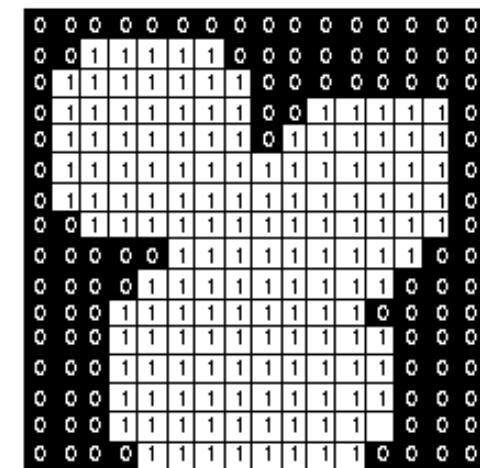
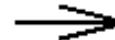
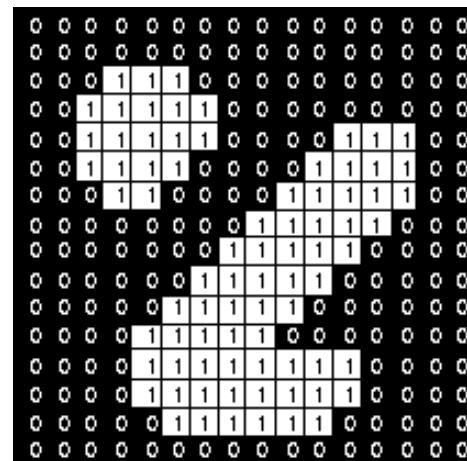


# Dilasi (2-Dimensi) ← Hit

$$g(x, y) = f(x, y) \oplus SE$$

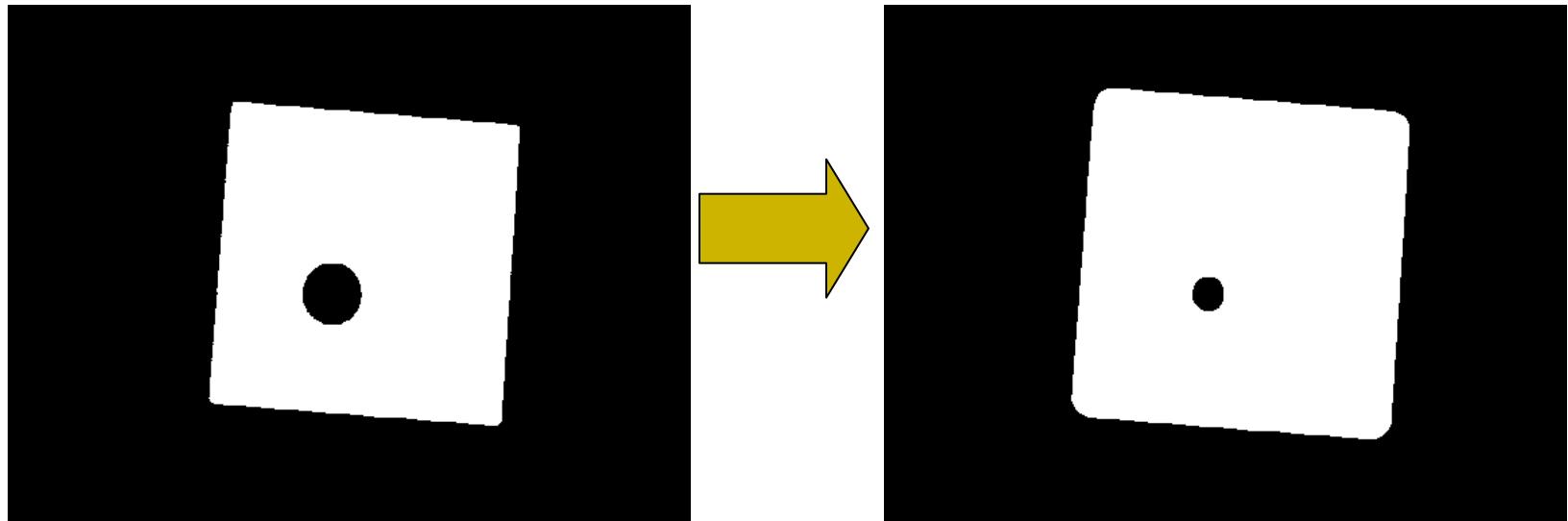
Structuring  
Element

1	1	1
1	1	1
1	1	1



- Objects tergabung (holes terisi object)
- Sudut yang tajam dihaluskan

# Contoh Dilasi



		1	1	1		
1	1	1	1	1		
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
	1	1	1	1		
	1	1	1			

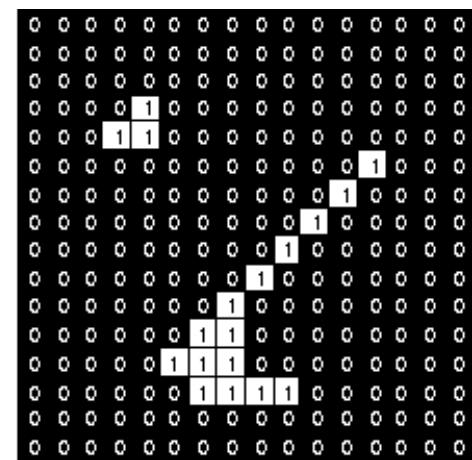
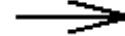
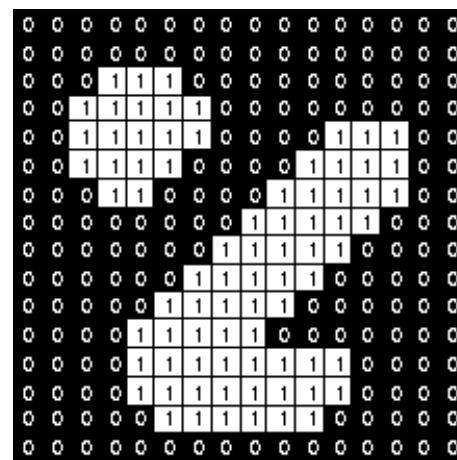
Structuring element:  
disc =>rounded corners

# Erosi (2-Dimensi) ← Fit

$$g(x, y) = f(x, y) \Theta SE$$

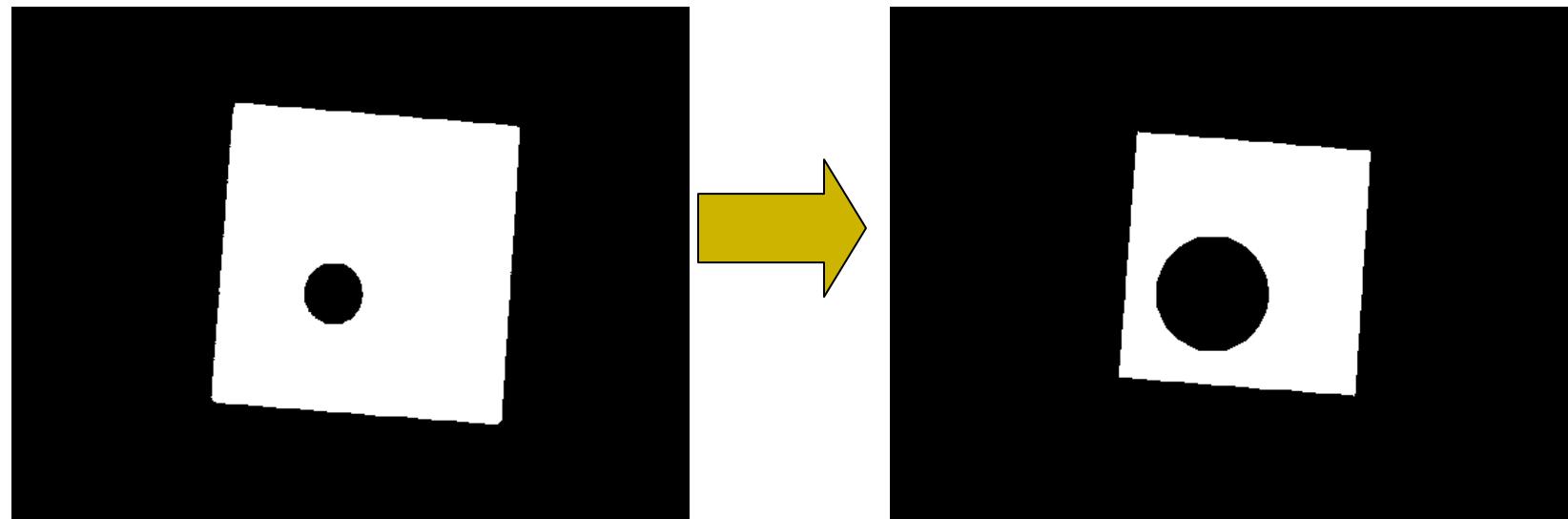
Structuring  
Element

1	1	1
1	1	1
1	1	1



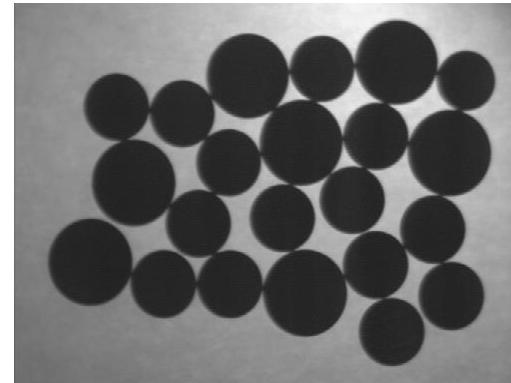
- Objects menjadi lebih kecil

# Contoh Erosi

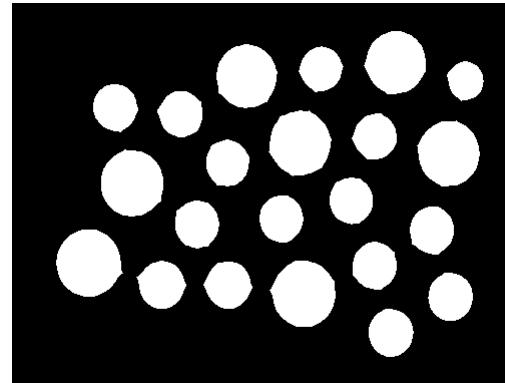
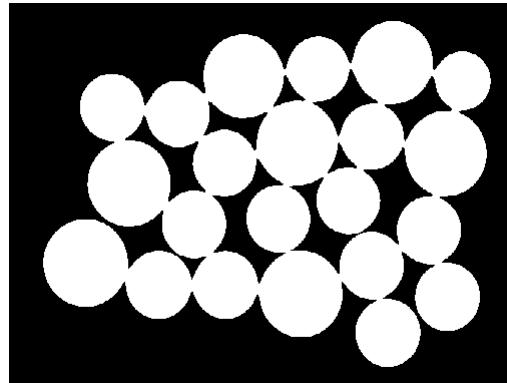


# Aplikasi Menghitung Koin

- Kesulitan menghitung koin pada gambar di bawah disebabkan tergabungnya object koin



- Solusi: Thresholding dan Erosi utk memisahkannya!

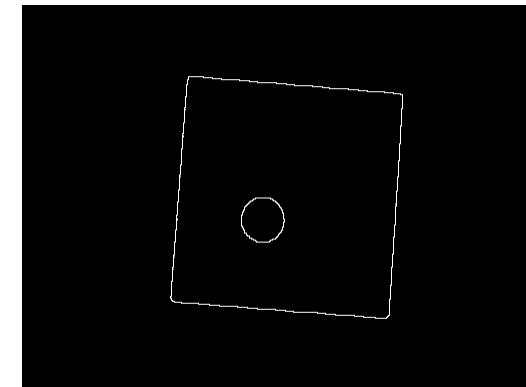
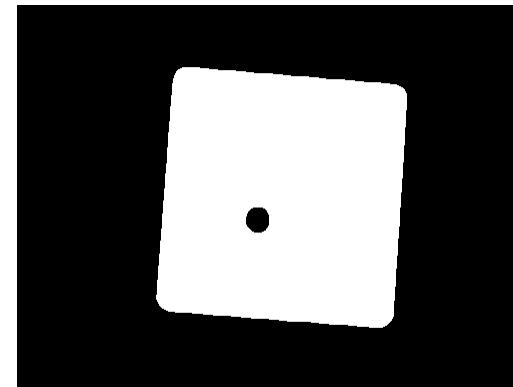
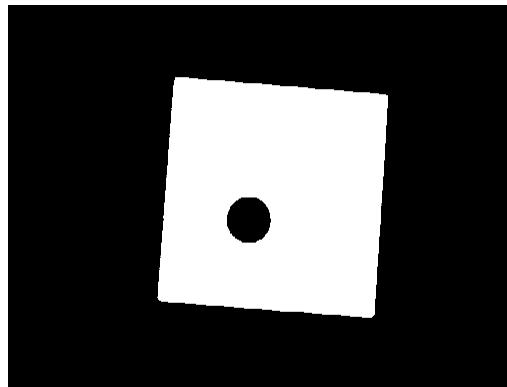


# Compound Operations

- Menggabungkan operasi Erosion dan Dilasi ke dalam level operasi yang lebih tinggi (more advanced)
  - Mencari garis tepi (*outline*)
  - *Opening*: mengisolasi objects dan menghilangkan object-object kecil (lebih baik daripada Erosi)
  - *Closing*: mengisi holes pada citra (lebih baik daripada Dilasi)

## Mencari garis tepi(outline)

- Operasi Dilasi (object menjadi lebih besar)
- Substraksi citra asal dengan citra hasil dilasi
- Didapatkan outline



# Opening

- Motivasi: menghilangkan object-object kecil TETAPI tetap mempertahankan ukurannya
- Opening =Erosion +Dilation
  - Gunakan SE yang sama
  - Hampir sama dengan erosi tetapi tidak terlalu *destructive*
- Math:
$$f(x, y) \circ SE = (f(x, y) \ominus SE) \oplus SE$$
- Opening adalah *idempotent*: operasi opening yang diulang-ulang tidak memberikan dampak yang berkelanjutan!

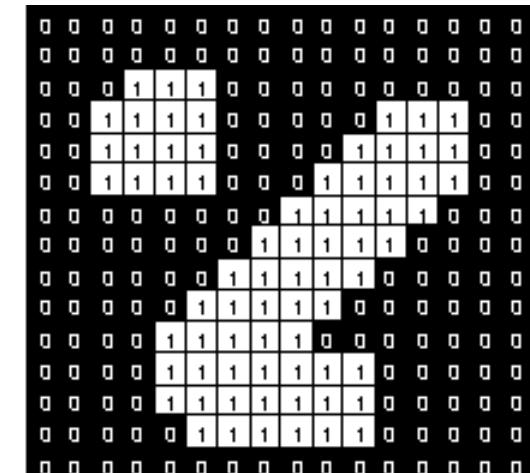
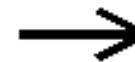
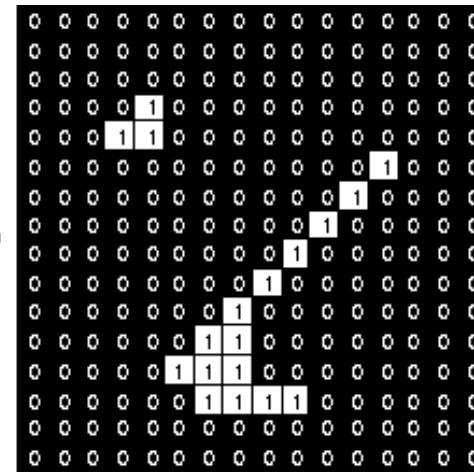
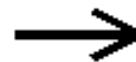
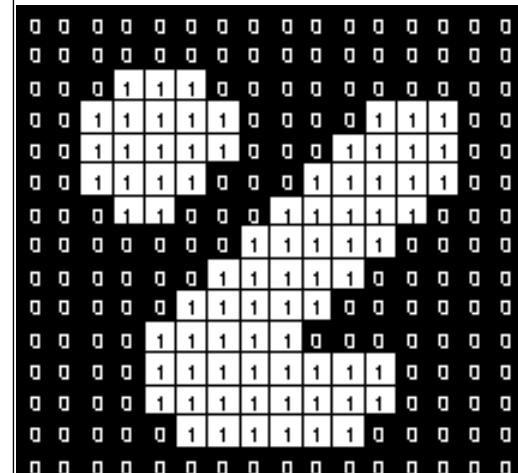
# Contoh Opening

SE

1	1	1
1	1	1
1	1	1

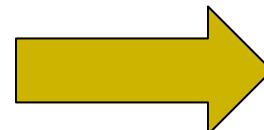
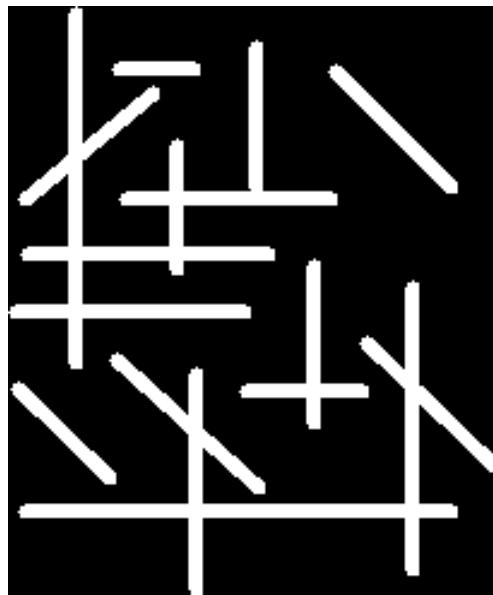
Erosion

Dilation



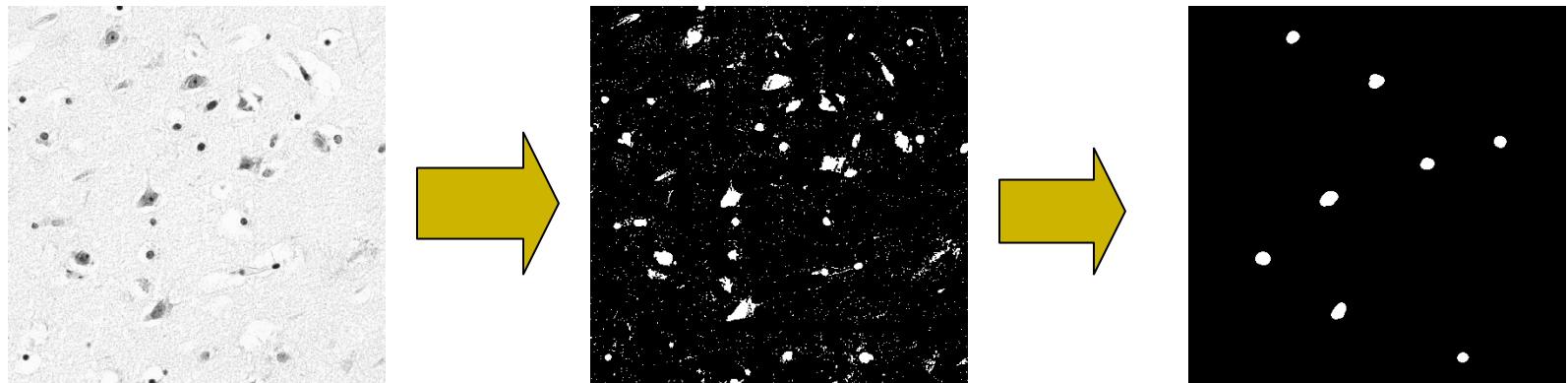
# Contoh Opening

- 9x3 and 3x9 Structuring Elements



# Contoh Opening

- Structuring Element: 11 pixel disc



(show: cell\_colony, 3 x erosion, 3 x dilation)

# Closing

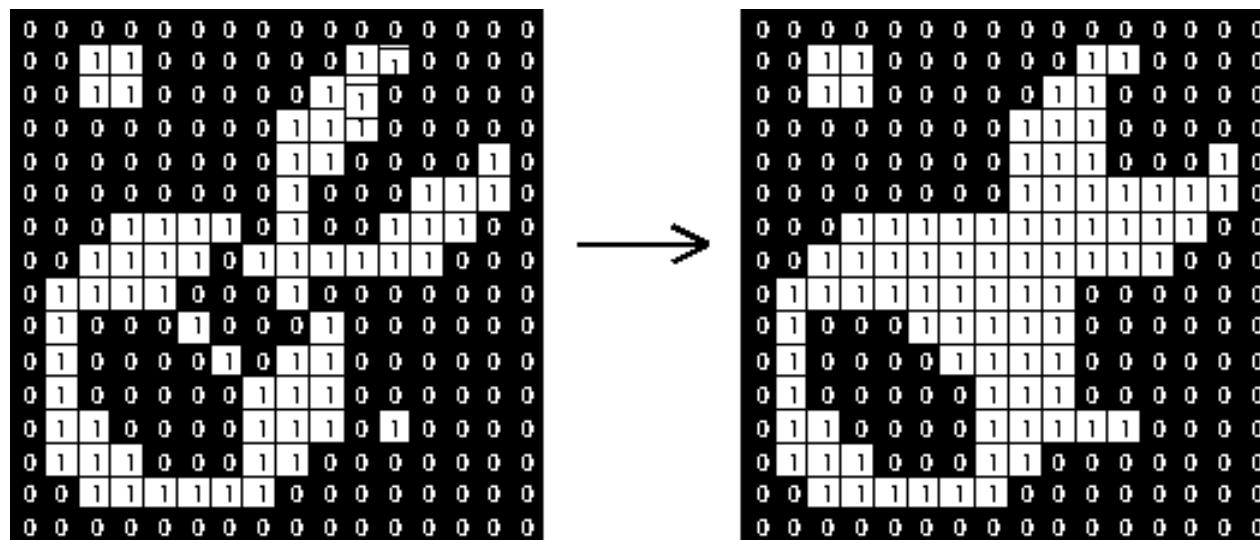
- Motivasi: Mengisi holes TETAPI tetap menjaga ukuran aslinya
- Opening = Dilation + Erosion
  - Gunakan SE yang sama
  - Hampir sama dengan dilasi tetapi tidak terlalu *destructive*
- Math:

$$f(x, y) \bullet SE = (f(x, y) \oplus SE) \ominus SE$$

- Closing adalah *idempotent*: operasi closing yang diulang-ulang tidak memberikan dampak yang berkelanjutan!

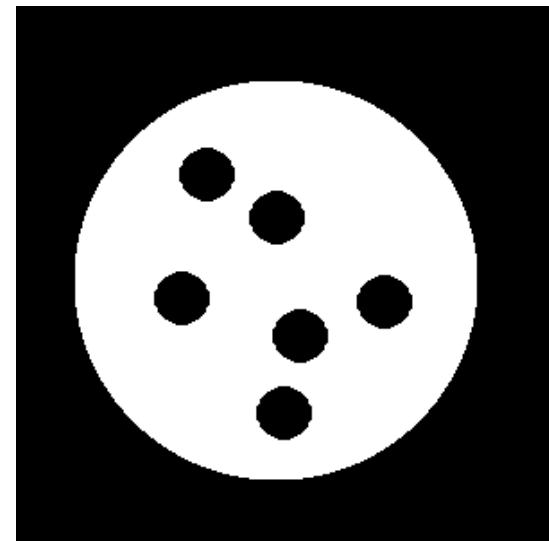
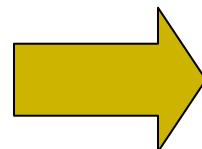
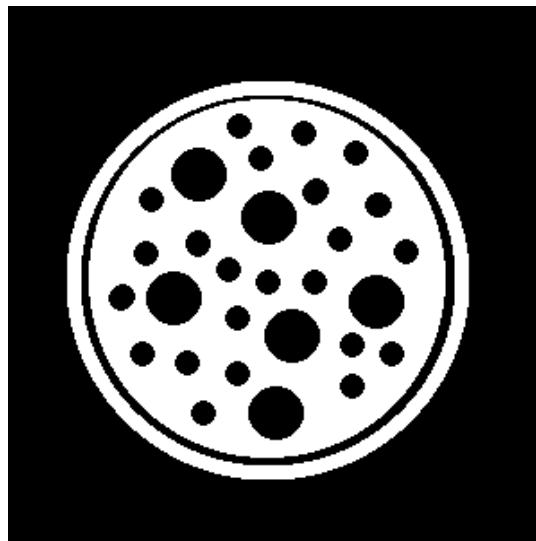
# Closing

- Structuring element: 3x3 square



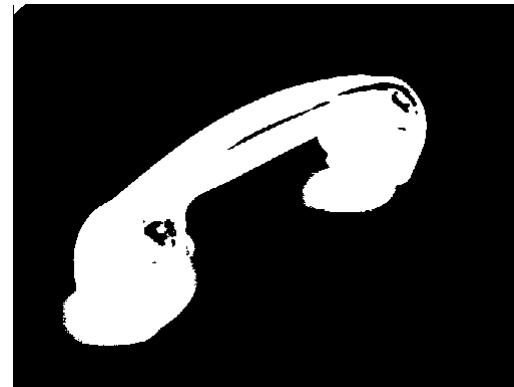
## Contoh Closing

- Operasi Closing dengan 22 piksel disc
- Menutupi holes yang kecil



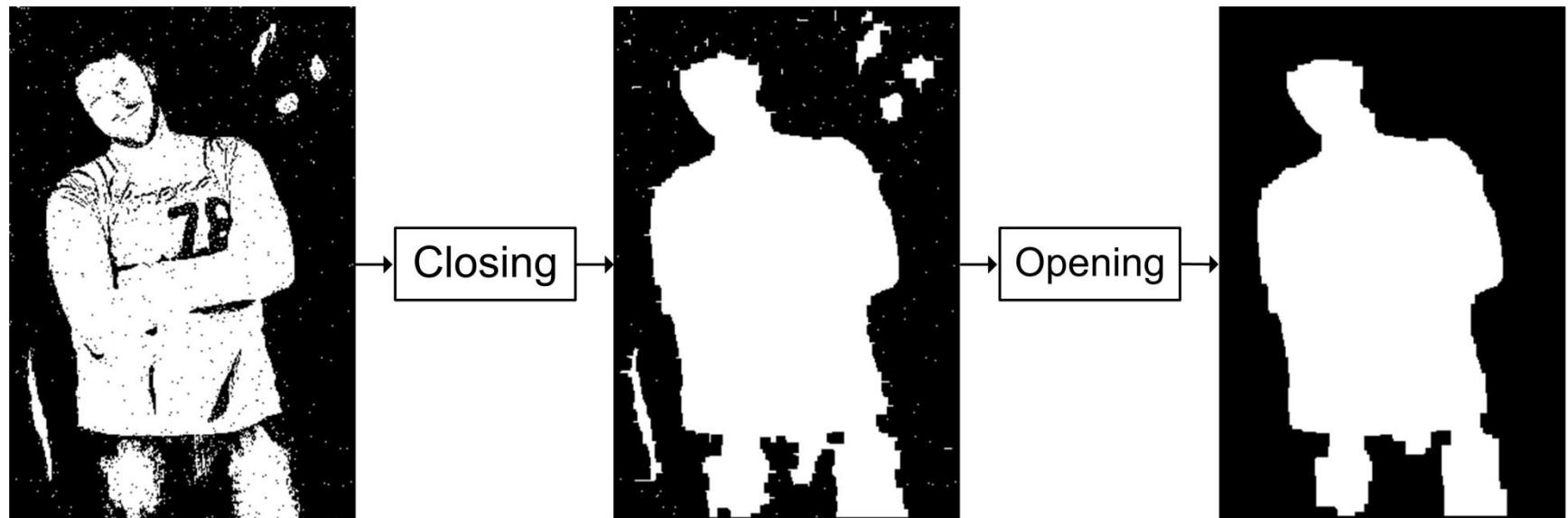
# Contoh Closing

- Improve segmentation
  - 1. Threshold
  - 2. Closing dengan ukuran 20 piksel disc

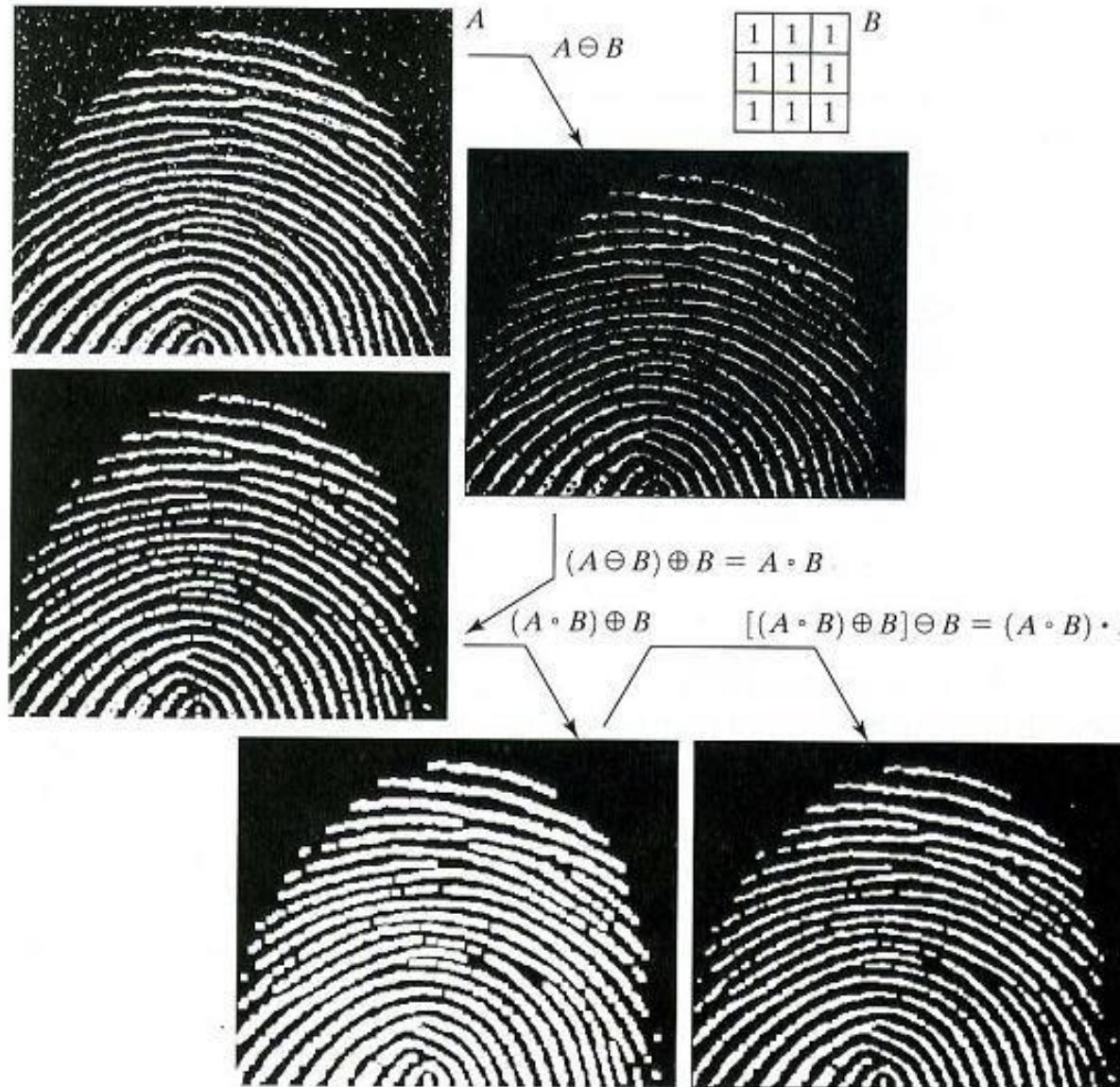


(show: blobs-holes, 1 x dilation, 1 x erosion)

# Kombinasi Closing + Opening



# Kombinasi Opening + Closing



**FIGURE 9.11**

- (a) Noisy image.  
(b) Structuring element.  
(c) Eroded image.  
(d) Opening of  $A$ .  
(e) Dilation of the opening.  
(f) Closing of the opening. (Original image courtesy of the National Institute of Standards and Technology.)

# Ringkasan

- Morphology
- Fit and Hit operations
- Erosion (based on Fit): Make objects smaller
  - Separate objects, remove small objects (noise)
- Dilation (based on Hit): Make objects bigger
  - Remove holes in objects
- Compound operations
  - Finding the outlines of the objects
  - Opening (Erosion +Dilation)
    - As Erosion but less destructive
  - Closing (Dilation +Erosion)
    - As Dilation but less destructive

# Latihan

- Diberikan citra biner:
    - Dilation
    - Erosion
    - Closing
    - Opening
  - Structuring element:

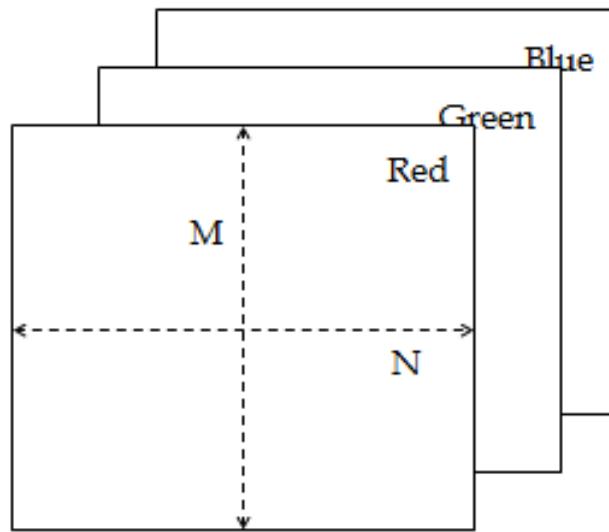
1	1	1
1	1	1
1	1	1

# Suplement-Color Space

- RGB
- GREY
- YCbCr
- YIQ
- HSV / HSI
- NTSC
- CIE Lab / Luv

# RGB (Red-Green-Blue)

- Citra berwarna (RGB) merupakan sebuah matriks piksel warna yang berukuran  $M \times N \times 3$ , yang mana setiap piksel warna dari citra terdiri dari komponen warna merah (Red), hijau (Green) dan biru (Blue).
- Secara khusus sebuah citra berwarna dapat pula dipandang sebagai sebuah tumpukan dari tiga buah matriks citra abu-abu yang masing – masing mewakili kanal merah, hijau dan biru.



# RGB

Command Window

```
>> x=imread('IMG_3333.jpg');
>> x_red=x(:,:,1);
>> x_green=x(:,:,2);
>> x_blue=x(:,:,3);
>> figure(1)
>> imshow(x)
Warning: Image is too big to fit on screen
> In imuitools\private\initSize at 72
  In imshow at 283
>> figure(2)
>> imshow(x_red)
Warning: Image is too big to fit on screen; displaying at 25%
> In imuitools\private\initSize at 72
  In imshow at 283
>> figure(3)
>> imshow(x_green)
Warning: Image is too big to fit on screen; displaying at 25%
> In imuitools\private\initSize at 72
  In imshow at 283
>> figure(4)
>> imshow(x_blue)
Warning: Image is too big to fit on screen; displaying at 25%
> In imuitools\private\initSize at 72
  In imshow at 283
```

Name	Value	Min	Max
x	<1880x2816x3 uint8>	<Too ...	<Too ...
x_blue	<1880x2816 uint8>	<Too ...	<Too ...
x_green	<1880x2816 uint8>	<Too ...	<Too ...
x_red	<1880x2816 uint8>	<Too ...	<Too ...

# RGB



RED



GREEN



BLUE



# Convert RGB to GRAY

- Formula bervariasi

1.  $\text{Gray} = (\text{R} + \text{G} + \text{B}) / 3$

2.  $\text{Gray} = ((0.3 * \text{R}) + (0.59 * \text{G}) + (0.11 * \text{B}))$

3. .....

# COLOR SPACE Formulas

RGB to YIQ

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

RGB to YCbCr

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65,481 & 128,553 & 24,966 \\ -37,797 & -74,203 & 112,000 \\ 112,000 & -93,786 & -18,214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

RGB to HSI

$$r = \frac{R}{(R + G + B)}, g = \frac{G}{(R + G + B)}, b = \frac{B}{(R + G + B)}$$

$$H = \begin{cases} 2\pi - \arccos\left(\frac{\frac{(r-b)+(r-g)}{2}}{\sqrt{(r-g)^2 + (r-b)(g-b)}}\right) & \text{for } b > g \\ \arccos\left(\frac{\frac{(r-b)+(r-g)}{2}}{\sqrt{(r-g)^2 + (r-b)(g-b)}}\right) & \text{for } b \leq g \end{cases}$$

$$S = 1 - 3 \min(r, g, b)$$

$$I = (R + G + B) / (3 * 255)$$

# RGB to HSI-1



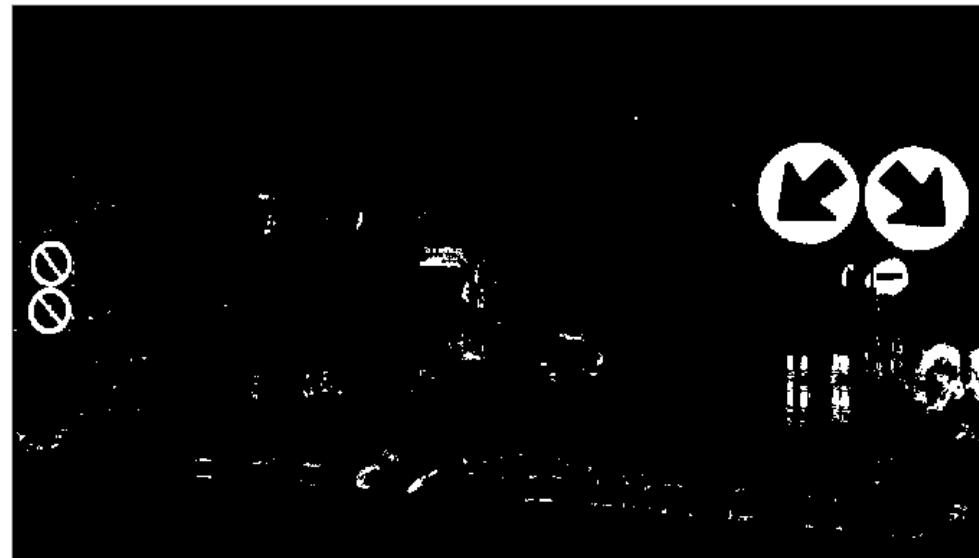
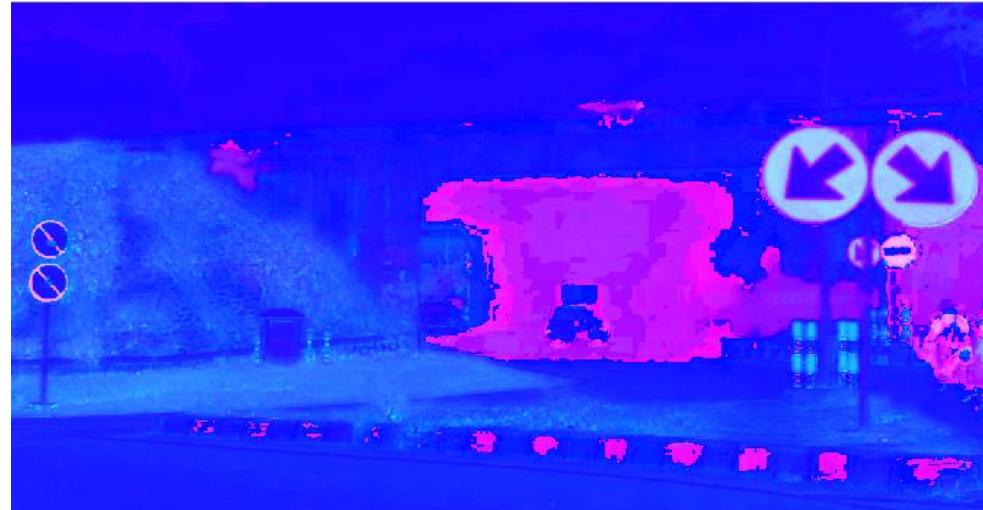
$$R = \begin{pmatrix} 224 & 224 & 224 & \dots & 228 \\ 224 & 224 & 224 & \dots & 243 \\ 224 & 224 & 224 & \dots & 236 \\ \vdots & & \ddots & & \vdots \\ 104 & 104 & 105 & \dots & 93 \end{pmatrix} \quad G = \begin{pmatrix} 224 & 224 & 224 & \dots & 224 \\ 224 & 224 & 224 & \dots & 239 \\ 224 & 224 & 224 & \dots & 232 \\ \vdots & & \ddots & & \vdots \\ 98 & 98 & 99 & \dots & 77 \end{pmatrix} \quad B = \begin{pmatrix} 222 & 222 & 222 & \dots & 215 \\ 222 & 222 & 222 & \dots & 230 \\ 222 & 222 & 222 & \dots & 223 \\ \vdots & & \ddots & & \vdots \\ 84 & 84 & 85 & \dots & 64 \end{pmatrix}$$

# RGB to HSI-2

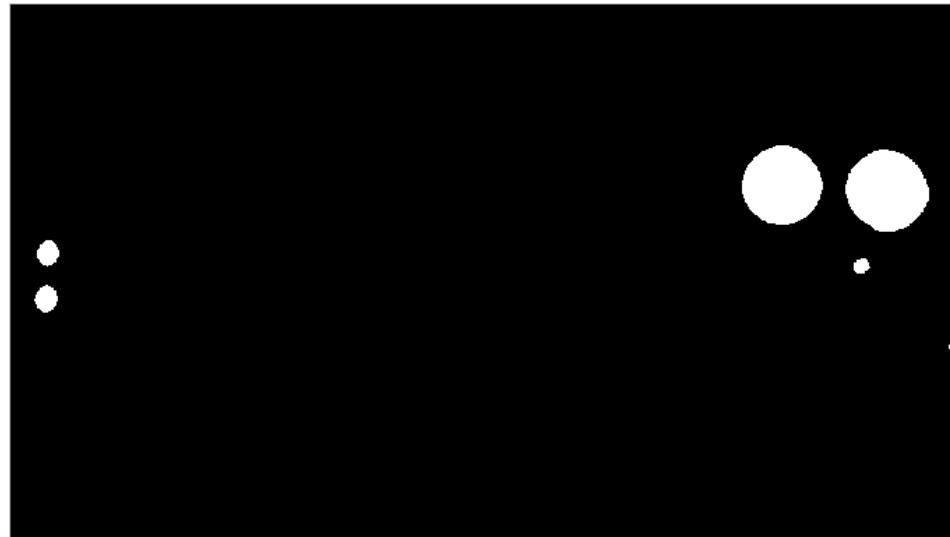
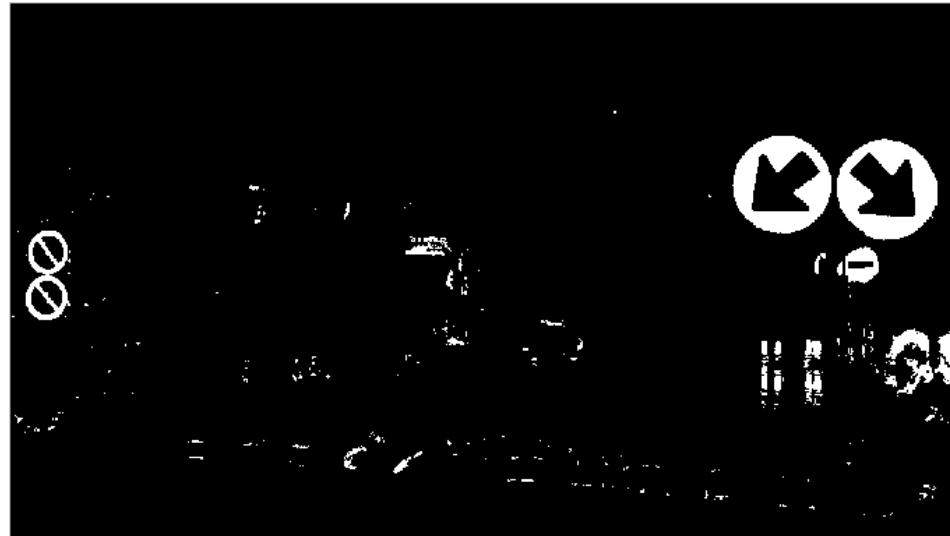


$$H = \begin{pmatrix} 42.5 & 42.5 & 42.5 & \cdots & 30.2 \\ 42.5 & 42.5 & 42.5 & \cdots & 30.2 \\ 42.5 & 42.5 & 42.5 & \cdots & 30.2 \\ \vdots & & \ddots & & \vdots \\ 30.5 & 30.5 & 30.5 & \cdots & 18.8 \end{pmatrix} \quad S = \begin{pmatrix} 1.5 & 1.5 & 1.5 & \cdots & 8.4 \\ 1.5 & 1.5 & 1.5 & \cdots & 7.8 \\ 1.5 & 1.5 & 1.5 & \cdots & 8.1 \\ \vdots & & \ddots & & \vdots \\ 30.3 & 30.3 & 30 & \cdots & 45.8 \end{pmatrix} \quad I = \begin{pmatrix} 222.3 & 222.3 & 222.3 & \cdots & 222.3 \\ 222.3 & 222.3 & 222.3 & \cdots & 237.3 \\ 222.3 & 222.3 & 222.3 & \cdots & 230.3 \\ \vdots & & \ddots & & \vdots \\ 95.3 & 95.3 & 96.3 & \cdots & 78 \end{pmatrix}$$

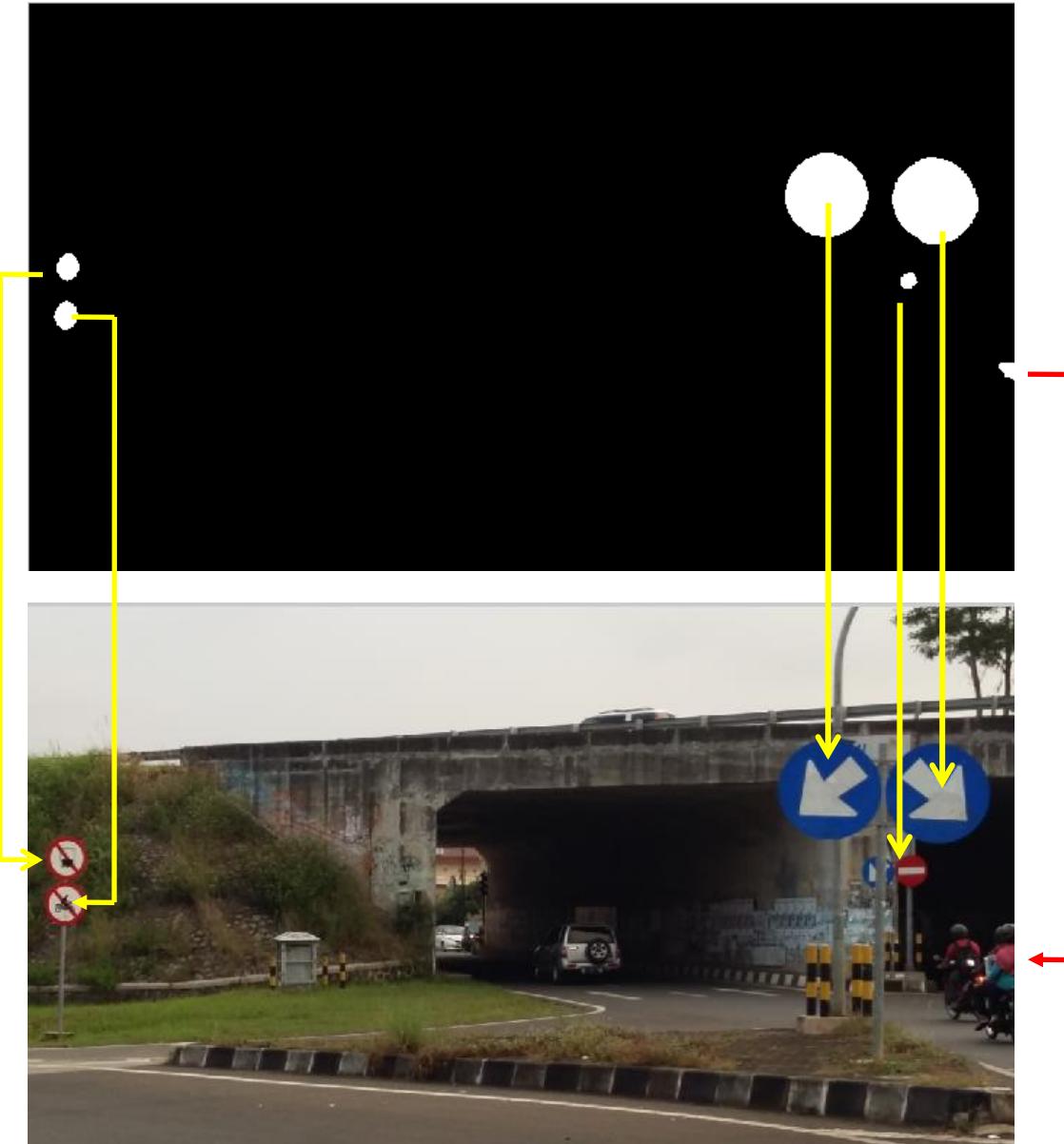
# Segmented



# Morphology-Opening



# ROI - Matching



# Object Detection-1



# Object Detection-2

deteksi\_rambu\_rambu\_ver3

**DETEKSI RAMBU-RAMBU LALU LINTAS**  
GAMBAR JALAN



2018/06/11 12:18:34

**EKSTRAKSI CIRI**

PHOG  
 Haar-PHOG  
 KNN

**DETEKSI**

SVM  
 Data Latih:  
 Jogja-Semarang,Solo-Jogja,Solo-Semarang  
 Jogja-Semarang,Solo-Jogja,Tol Smrg-Sltg  
 Jogja-Semarang,Solo-Semarang,Tol Smrg-Sltg  
 Solo-Jogja,Solo-Semarang,Tol Smrg-Sltg

Data Uji:  
Solo - Jogja

FILE Ke 799 Dari 5458

NAMA FILE solo-jogja\_17\_2836.jpg

16

FOLDER D:\Disertasi\DataSet\SOLO-JOGJA\DataUji

**MORFOLOGI**



**ROI KANDIDAT**



**RAMBU TERDETEKSI**



**FOLDER**  
**<< PREVIOUS**  
**NEXT >>**  
**PROCESS**  
**EXIT**

# Object Detection-3

deteksi\_rambu\_rambu\_ver3

## DETEKSI RAMBU-RAMBU LALU LINTAS

GAMBAR JALAN

2018/06/11 12:45:13

MORFOLOGI

ROI KANDIDAT

RAMBU TERDETEKSI

FOLDER

<< PREVIOUS

NEXT >>

PROCESS

EXIT

EKSTRAKSI CIRI

PHOG  
 Haar-PHOG

DETEKSI

SVM  
 KNN

Data Latih

Jogja-Semarang,Solo-Jogja,Solo-Semarang  
 Jogja-Semarang,Solo-Jogja,Tol Smrg-Sltg  
 Jogja-Semarang,Solo-Semarang,Tol Smrg-Sltg  
 Solo-Jogja,Solo-Semarang,Tol Smrg-Sltg

Data Uji

Solo - Jogja

FILE Ke 2092 Dari 5458

NAMA FILE solo-jogja\_26\_2481.jpg

1025

FOLDER D:\Disertasi\DataSet\SOLO-JOGJA\DataUji

**Thank You !**

# PENGOLAHAN CITRA DIGITAL

IMAGES COMPRESSION

GKV - IF 2020

# IMAGES COMPRESSION

- Pendahuluan
- Kriteria, Jenis dan Klasifikasi Metode Kompresi
- Huffman Code
- RLE
- JPEG

## Pemampatan citra (*image compression*).

- Pada proses ini, citra dalam representasi tidak mampu dikodekan dengan representasi yang meminimumkan kebutuhan memori.
- Citra dengan format *bitmap* pada umumnya *tidak dalam bentuk mampat*.
- *Citra yang sudah dimampatkan* disimpan ke dalam arsip dengan format tertentu. Kita mengenal format JPG, GIF sebagai format citra yang sudah dimampatkan.

## Penirmampatkan citra (*image decompression*).

- Pada proses ini, citra yang sudah dimampatkan harus dapat dikembalikan lagi (*decoding*) menjadi *representasi yang tidak mampat*.
- *Proses ini* diperlukan jika citra tersebut ditampilkan ke layar atau disimpan ke dalam arsip dengan format tidak mampat.
- Dengan kata lain, penirmampatan citra mengembalikan citra yang termampatkan menjadi data *bitmap*.

# Aplikasi Pemampatan Citra

Pengiriman data (*data transmission*) pada saluran komunikasi data

- Aplikasi pengiriman gambar lewat *fax*, *video conference*, *pengiriman data medis*, pengiriman gambar dari satelit luar angkasa, pengiriman gambar via telefon genggam, *download gambar dari internet*, dan sebagainya.

Penyimpanan data (*data storing*) di dalam media sekunder (*storage*)

- Aplikasi nya antara lain aplikasi basisdata gambar, *office automation*, *video storage* (seperti *VCD*, *DVD*)

# Kriteria Pemampatan Citra

Waktu pemampatan dan penirmampatan (*decompression*).

- Waktu pemampatan citra dan penirmampatannya sebaiknya cepat

Kebutuhan memori.

- Memori yang dibutuhkan untuk merepresentasikan citra seharusnya berkurang secara signifikan.

Kualitas pemampatan (fidelity)

- Informasi yang hilang akibat pemampatan seharusnya seminimal mungkin sehingga kualitas hasil pemampatan tetap dipertahankan. Alat ukur PSNR.

Format keluaran

- Format citra hasil pemampatan sebaiknya cocok untuk pengiriman dan penyimpanan data.

# Jenis Pemampatan Citra

Pendekatan statistik.

- Pemampatan citra didasarkan pada frekuensi kemunculan derajat keabuan *pixel di dalam seluruh bagian gambar*.
- Contoh metode: *Huffman Coding*.

Pendekatan ruang

- Pemampatan citra didasarkan pada hubungan spasial antara *pixel-pixel* di dalam suatu kelompok yang memiliki derajat keabuan yang sama di dalam suatu daerah di dalam gambar.
- Contoh metode: *Run-Length Encoding*.

Pendekatan kuantisasi

- Pemampatan citra dilakukan dengan mengurangi jumlah derajat keabuan yang tersedia.
- Contoh metode: metode pemampatan kuantisasi.

Pendekatan fraktal

- Pemampatan citra didasarkan pada kenyataan bahwa kemiripan bagianbagian di dalam citra dapat dieksplorasi dengan suatu matriks transformasi.
- Contoh metode: *Fractal Image Compression*.

# Klasifikasi Metode Pemampatan

## Metode *lossless*

- Metode *lossless* selalu menghasilkan citra hasil penirmampatan yang tepat sama dengan citra semula, pixel per pixel. Tidak ada informasi yang hilang akibat pemampatan.
- Nisbah (*ratio*) pemampatan citra metode *lossless* sangat rendah.
- Contoh metode *lossless* adalah metode *Huffman*.

## Metode *lossy*

- Metode *lossy* menghasilkan citra hasil pemampatan yang hampir sama dengan citra semula.
- Ada informasi yang hilang akibat pemampatan, tetapi dapat ditolerir oleh persepsi mata.
- Mata tidak dapat membedakan perubahan kecil pada gambar.
- Metode pemampatan *lossy* menghasilkan nisbah pemampatan yang tinggi daripada metode *lossless*.
- Contoh metode *lossy* adalah metode *JPEG* dan metode *fraktal*.

# Metode Pemampatan Huffman

Metode pemampatan Huffman menggunakan prinsip bahwa nilai (atau derajat) keabuan yang sering muncul di dalam citra akan dikodekan dengan jumlah bit yang lebih sedikit sedangkan nilai keabuan yang frekuensi kemunculannya sedikit dikodekan dengan jumlah bit yang lebih panjang.

# Algoritma metode Huffman

1. Urutkan secara menaik (*ascending order*) *nilai-nilai keabuan berdasarkan frekuensi kemunculannya*. Setiap nilai keabuan dinyatakan sebagai pohon bersimpul tunggal. Setiap simpul di-*assign dengan frekuensi kemunculan nilai keabuan tersebut*.
2. Gabung dua buah pohon yang mempunyai frekuensi kemunculan paling kecil pada sebuah akar. Akar mempunyai frekuensi yang merupakan jumlah dari frekuensi dua buah pohon penyusunnya.
3. Ulangi langkah 2 sampai tersisa hanya satu buah pohon biner.
4. Beri label setiap sisi pada pohon biner. Sisi kiri dilabeli dengan 0 dan sisi kanan dilabeli dengan 1.
5. Telusuri pohon biner dari akar ke daun. Barisan label-label sisi dari akar ke daun menyatakan kode Huffman untuk derajat keabuan yang bersesuaian.

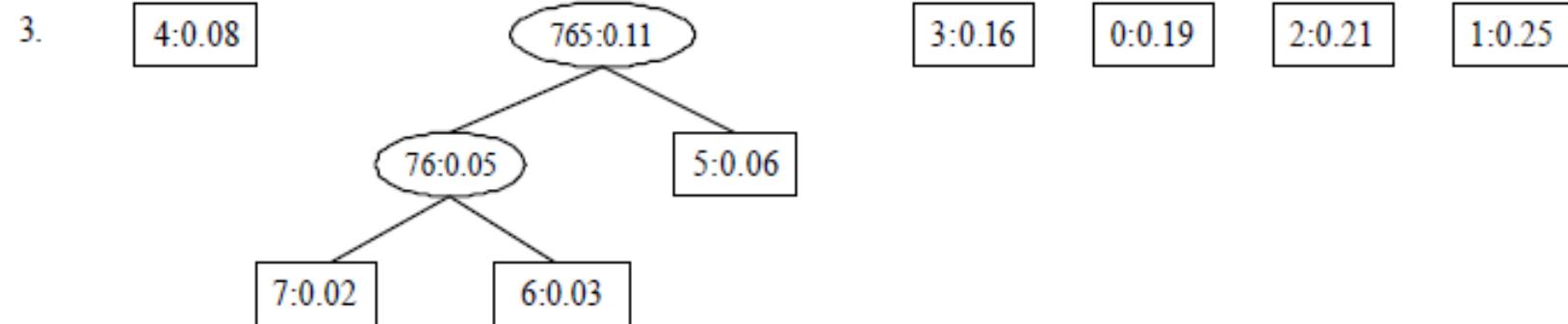
# Contoh Huffman Code

Misalkan terdapat citra yang berukuran  $64 \times 64$  dengan 8 derajat keabuan ( $k$ ) dan jumlah seluruh pixel ( $n$ ) =  $64 \times 64 = 4096$

$k$	$n_k$	$p(k) = n_k/n$
0	790	0.19
1	1023	0.25
2	850	0.21
3	656	0.16
4	329	0.08
5	245	0.06
6	122	0.03
7	81	0.02

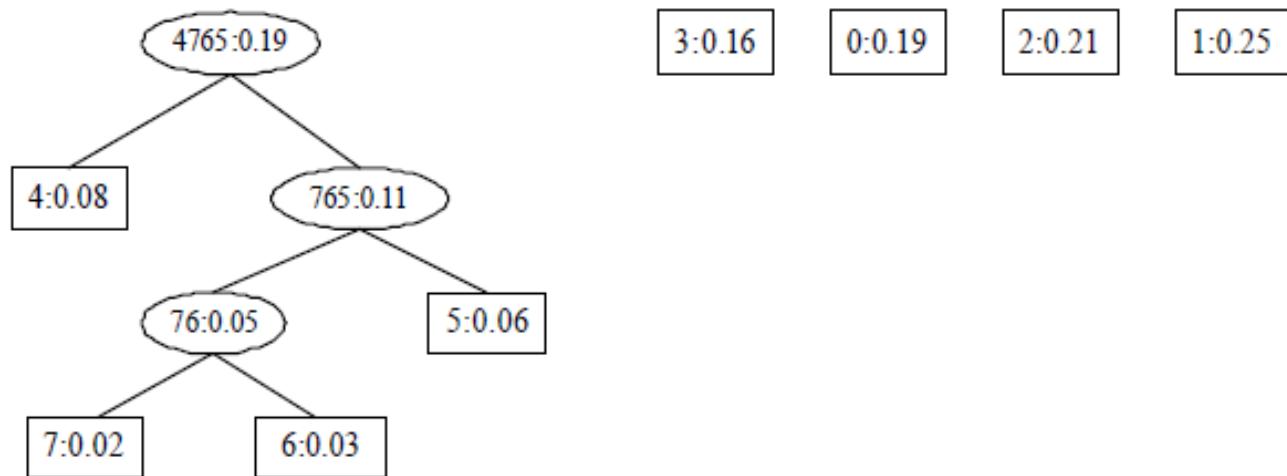
# Contoh Huffman Code

1. 7:0.02 6:0.03 5:0.06 4:0.08 3:0.16 0:0.19 2:0.21 1:0.25

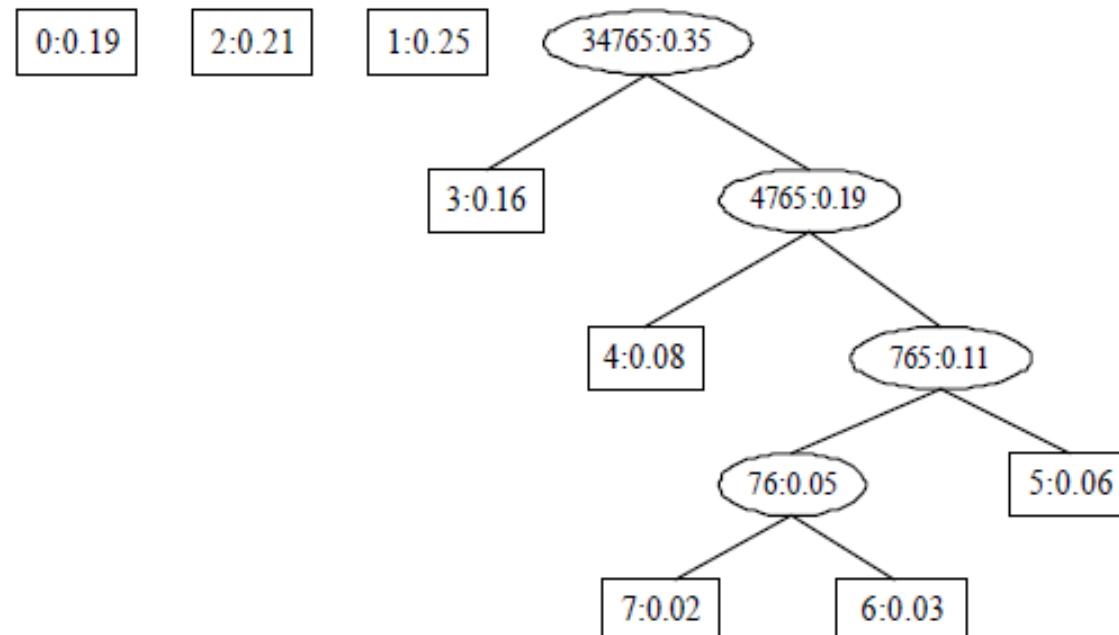


# Contoh Huffman Code

4.

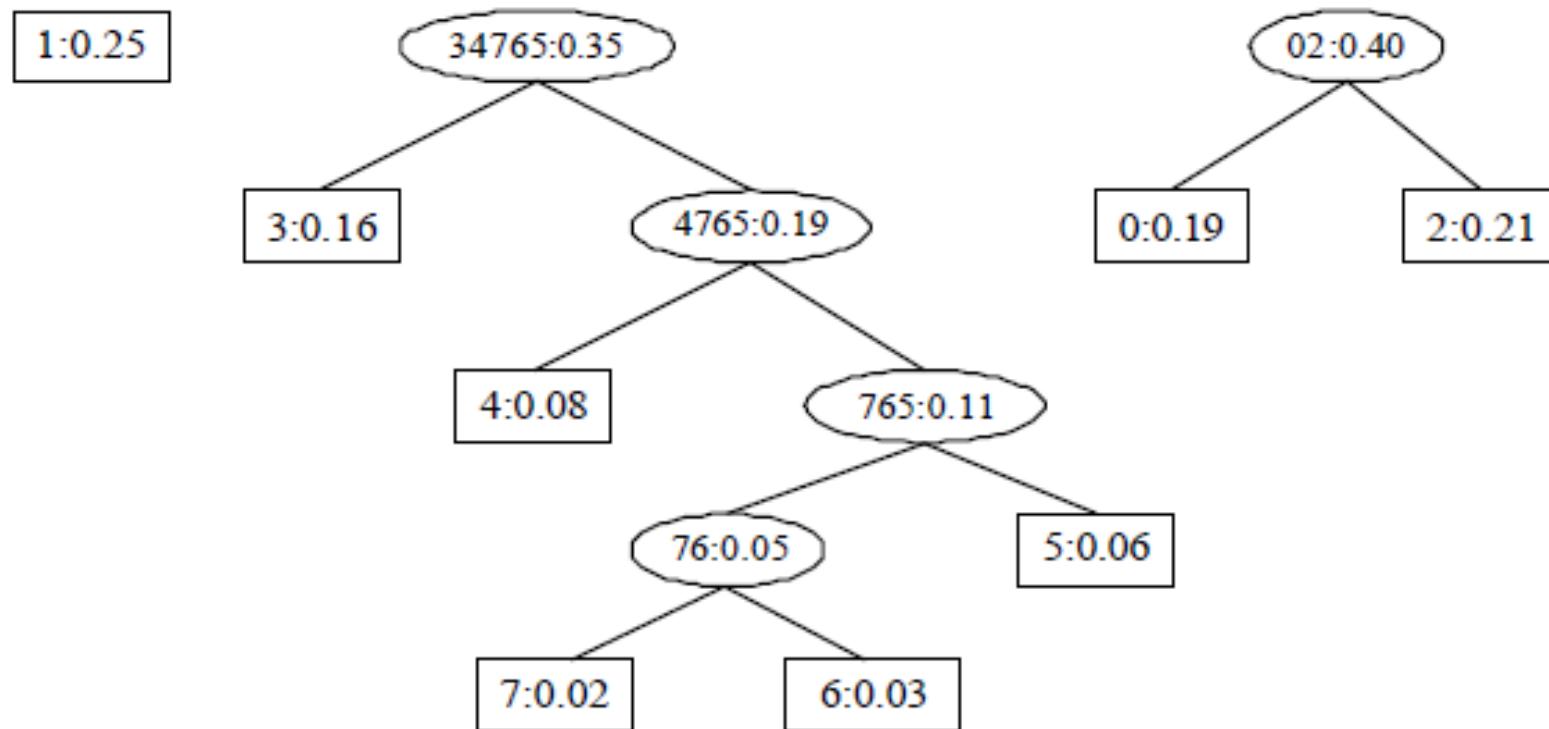


5.



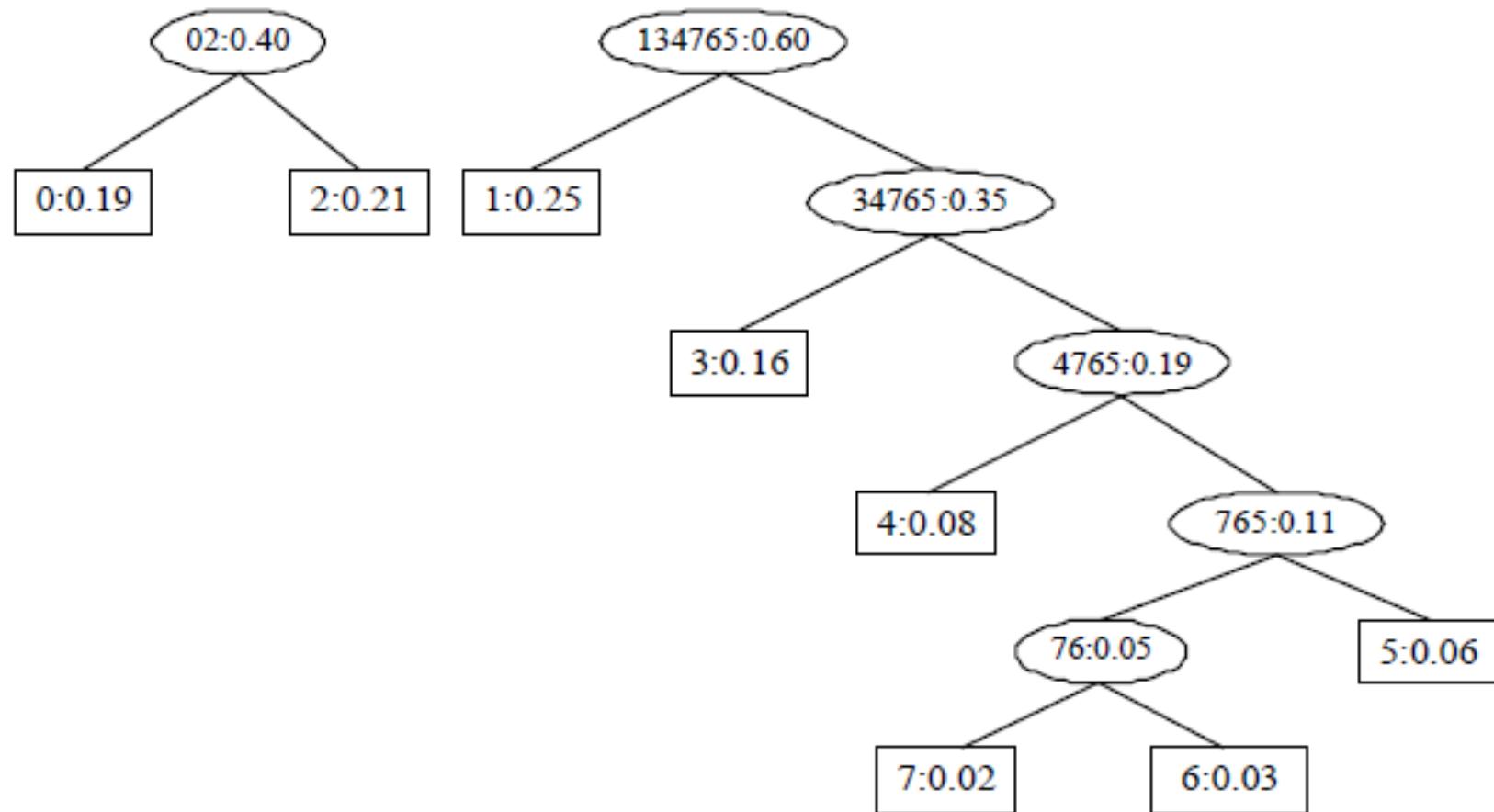
# Contoh Huffman Code

6.



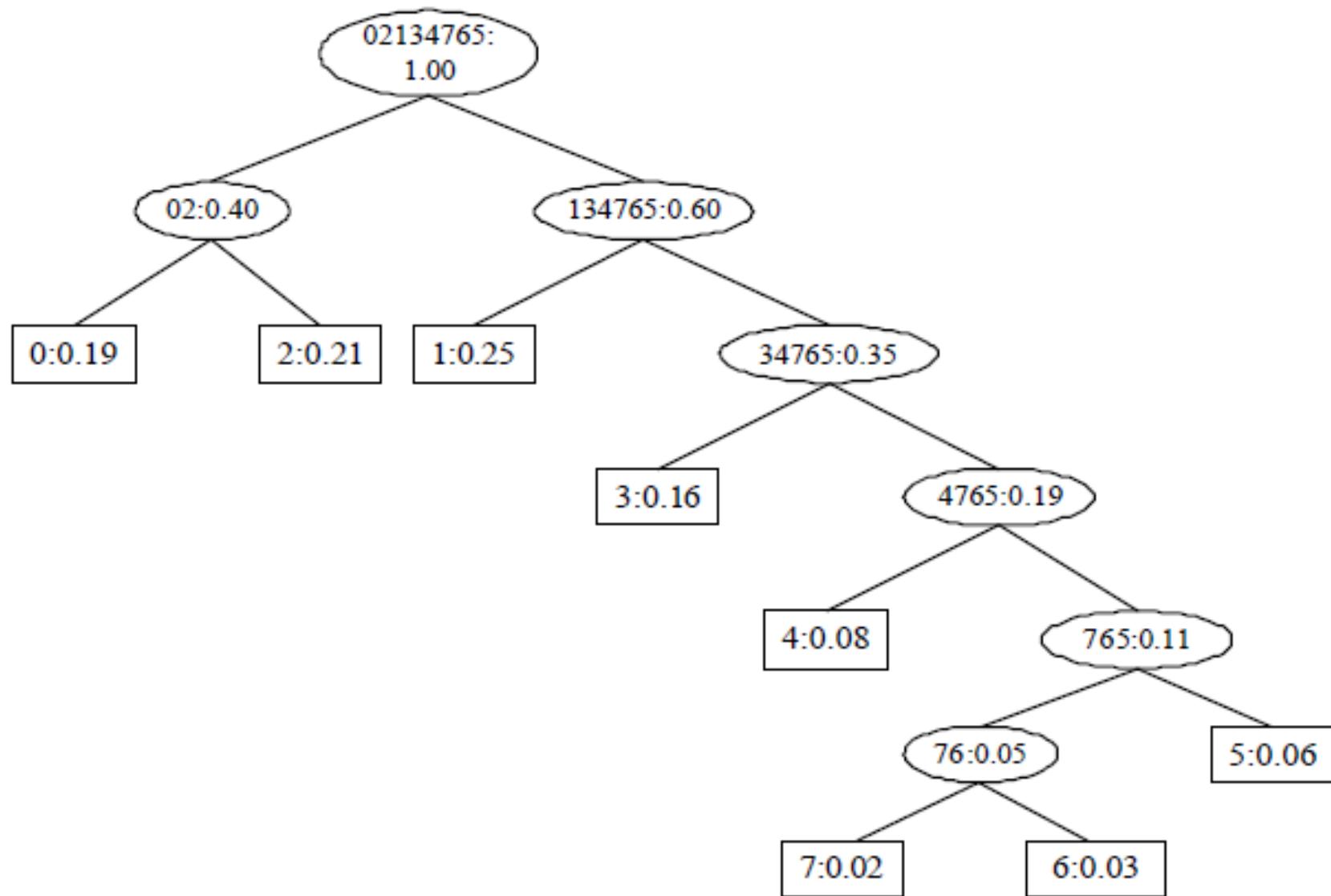
# Contoh Huffman Code

7.



# Contoh Huffman Code

8.



# Contoh Huffman Code

Kode Huffman yang dihasilkan untuk derajat keabuan 0-7

$$\begin{array}{l} 0 = 00 \\ 1 = 10 \end{array}$$

$$\begin{array}{l} 2 = 01 \\ 3 = 110 \end{array}$$

$$\begin{array}{l} 4 = 1110 \\ 5 = 11111 \end{array}$$

$$\begin{array}{l} 6 = 111101 \\ 7 = 111100 \end{array}$$

Ukuran citra asli =  $4096 \times 3$  bit = 12288 bit

Setelah dikompresi =  $(790 \times 2$  bit $) + (1023 \times 2$  bit $) + (850 \times 2$  bit $) +$   
 $(656 \times 3$  bit $) + (329 \times 4$  bit $) + (245 \times 5$  bit $) +$   
 $(122 \times 6$  bit $) + (81 \times 6$  bit $) = 11053$  bit

Nisbah pemampatan =  $(100\% - \frac{11053}{12288} \times 100\%) = 10\%$  , yang artinya 10% dari citra semula telah dimampatkan. ■

# Tentukan Huffman Code

<b>Derajat Keabuan</b>	<b>probabilitas</b>
0	0,3
1	0,1
2	0,2
3	0,06
4	0,09
5	0,07
6	0,03
7	0,15

# Metode Pemampatan *Run-Length Encoding (RLE)*

- Metode *RLE* cocok digunakan untuk memampatkan citra yang memiliki kelompok-kelompok pixel berderajat keabuan sama.
- Pemampatan citra dengan metode *RLE* dilakukan dengan membuat rangkaian pasangan nilai  $(p, q)$  untuk setiap baris pixel, nilai pertama ( $p$ ) menyatakan derajat keabuan, sedangkan nilai kedua ( $q$ ) menyatakan jumlah pixel berurutan yang memiliki derajat keabuan tersebut (dinamakan *run length*).

# *Run-Length Encoding (RLE)*

Tinjau citra ~~10 x 10 pixel~~ dengan 8 derajat keabuan yang dinyatakan sebagai matriks derajat keabuan sebagai berikut

0	0	0	0	0	2	2	2	2	2
0	0	0	1	1	1	1	2	2	2
1	1	1	1	1	1	1	1	1	1
4	4	4	4	3	3	3	3	2	2
3	3	3	5	5	7	7	7	7	6
2	2	6	0	0	0	0	1	1	0
3	3	4	4	3	2	2	2	1	1
0	0	0	0	0	0	0	0	1	1
1	1	1	1	0	0	0	2	2	2
3	3	3	2	2	2	1	1	1	1

# **Run-Length Encoding (RLE)**

Pasangan nilai untuk setiap baris *run* yang dihasilkan dengan metode pemampatan RLE:

(0, 5), (2, 5)

(0, 3), (1, 4), (2, 3)

(1, 10)

(4, 4), (3, 4), (2, 2)

(3, 3), (5, 2), (7, 4), (6, 1)

(2, 2), (6, 1), (0, 4), (1, 2), (0, 1)

(3, 2), (4, 2), (3, 1), (2, 2), (1, 2)

(0, 8), (1, 2)

(1, 4), (0, 3), (2, 3)

(3, 3), (2, 3), (1, 4)

semuanya ada 31 pasangan nilai atau  $31 \times 2 = 62$  nilai.

→ Ada D sebanyak 5x

# **Run-Length Encoding (RLE)**

$10 \times 10$

- Ukuran citra sebelum pemampatan (1 derajat keabuan = 3 bit) adalah  $100 \times 3$  bit = 300 bit
- Sedangkan ukuran citra setelah pemampatan (derajat keabuan = 3 bit, *run length* = 4 bit) :

$$(31 \times 3) + (31 \times 4) \text{ bit} = 217 \text{ bit}$$

- Nisbah pemampatan =  $(100\% - (217/300) \times 100\%)$   
= 27.67%

(27.67% dari citra semula telah dimampatkan)

# Kompresi JPEG

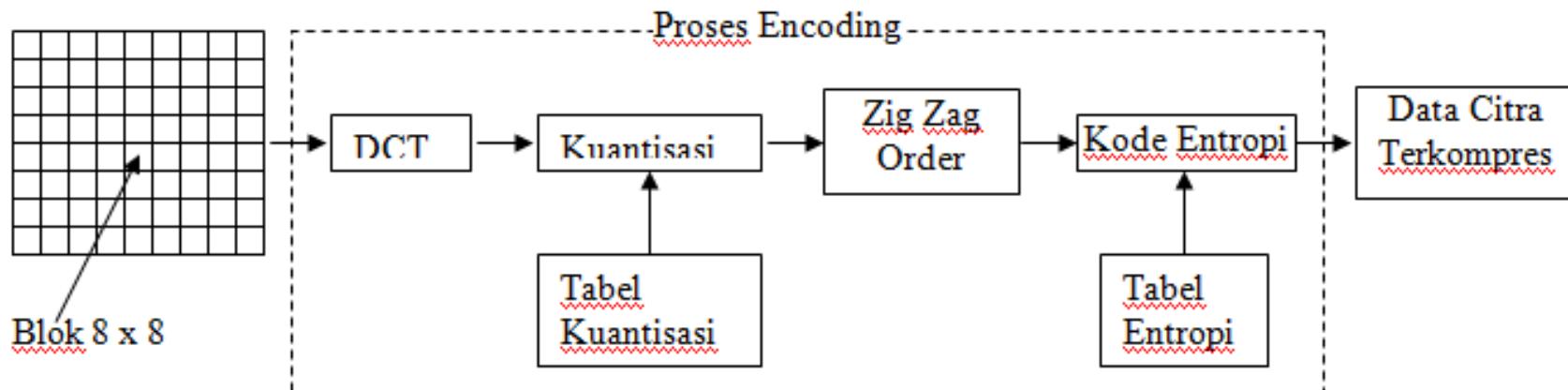
- JPEG (***Joint Photographic Expert Group***) merupakan kompresi citra pertama yang memiliki standar internasional.
- Teknik kompresi ini merupakan hasil kerja sama antara ITU (International Telecommunication Union), ISO (International Organization for Standardization) dan IEC (International Electrotechnical Commission)
- JPEG memanfaatkan keterbatasan mata manusia dalam melihat warna.
- Mata manusia tidak sensitif terhadap perubahan warna yang kecil, dibandingkan dengan perubahan kecerahan (*brightness*) yang kecil.
- Kompresi ini sangat cocok sekali diaplikasikan pada foto-foto digital (*Digital Images*).

# Algoritma – JPEG (Encoding)

- Sebagai inputan dalam *encoding*, *pixel-pixel* dalam matriks citra dikelompokkan menjadi blok matriks 8x8
- Untuk setiap blok ini, tiap-tiap pixelnya dikurangi 128, kemudian ditransformasi menggunakan DCT, sehingga menghasilkan 64 koefisien DCT
- Proses kuantisasi, dengan membagi tiap elemen dalam matriks yang telah dikonversi dengan tabel kuantisasi dan bulatkan ke integer terdekat.
- Membentuk vektor dengan men-scan *AC Coefficient* matriks hasil kuantisasi secara zig-zag
- RLE atau Huffman Code untuk kompresi

# JPEG - Encoding

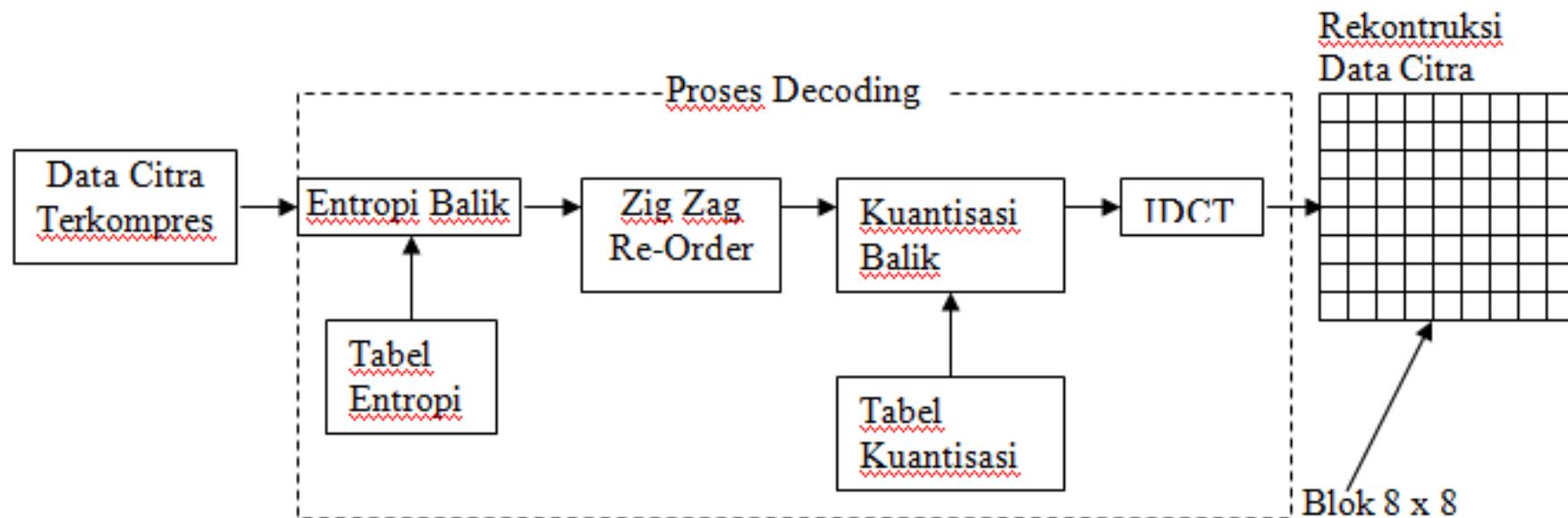
Data Citra Input



# Algoritma – JPEG (Decoding)

- Lakukan *Huffman Decoding* yaitu proses membalik dari kode bit-bit proses pengkodean Huffman menjadi nilai semula
- *Inverse zig-zag*
- Dekuantisasi dengan mengkalikan matriks dengan tabel kuantisasi *Inverse DCT* yaitu melakukan proses transformasi invers DCT untuk mendapatkan nilai – nilai piksel kembali.
- Tambahkan masing-masing kelompok  $8 \times 8$  piksel dengan 128
- Gabung kembali kelompok-kelompok piksel menjadi sebuah satu kesatuan matriks awal.

# JPEG - Decoding



# JPEG-1



110	110	118	118	121	126	131	131
108	111	125	122	120	125	134	135
106	119	129	127	125	127	138	144
110	126	130	133	133	131	141	148
115	116	119	120	122	125	137	139
115	106	99	110	107	116	130	127
110	91	82	101	99	104	120	118
103	76	70	95	92	91	107	106

-18	-18	-10	-10	-7	-2	3	3
-20	-17	-3	-6	-8	-3	6	7
-22	-9	1	-1	-3	-1	10	16
-18	-2	2	5	5	3	13	20
-13	-12	-9	-8	-6	-3	9	11
-13	-22	-29	-18	-21	-12	2	-1
-18	-37	-46	-27	29	-24	-8	-10
-25	-52	-58	-33	-36	-37	-21	-22

Matriks 8x8 (kiri) dan hasilnya setelah dikurang 128 (kanan)

# JPEG-2

-18	-18	-10	-10	-7	-2	3	3
-20	-17	-3	-6	-8	-3	6	7
-22	-9	1	-1	-3	-1	10	16
-18	-2	2	5	5	3	13	20
-13	-12	-9	-8	-6	-3	9	11
-13	-22	-29	-18	-21	-12	2	-1
-18	-37	-46	-27	29	-24	-8	-10
-25	-52	-58	-33	-36	-37	-21	-22

DCT



-89.00	-63.47	18.21	-6.85	7.50	13.45	-7.00	0.13
74.14	-2.90	-19.93	-21.04	-17.88	-10.81	8.29	5.26
-63.65	3.10	5.08	14.82	10.12	9.33	1.31	-0.62
3.73	2.85	6.67	8.99	-3.38	1.54	1.04	-0.62
2.50	0.57	-4.46	0.52	3.00	-2.89	-0.32	1.33
7.52	-1.80	-0.63	-0.10	0.41	-3.21	-2.74	-2.07
-3.40	0.43	0.81	0.28	-0.40	-0.19	-0.58	-1.09
-2.26	-0.88	1.73	0.23	-0.21	-0.12	1.23	1.61

# JPEG-3

- *DCT:*  $C(p, q) = \alpha_p \alpha_q \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} I(m, n) \cos \frac{\pi(2m+1)p}{2N} \cos \frac{\pi(2n+1)q}{2N}$
- *IDCT:*  $I(m, n) = \underbrace{\sum_{p=0}^{N-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q C(p, q)}_{\text{Pixel}} \cos \frac{\pi(2m+1)p}{2N} \cos \frac{\pi(2n+1)q}{2N}$
- Keterangan: Citra berukuran  $M \times N$

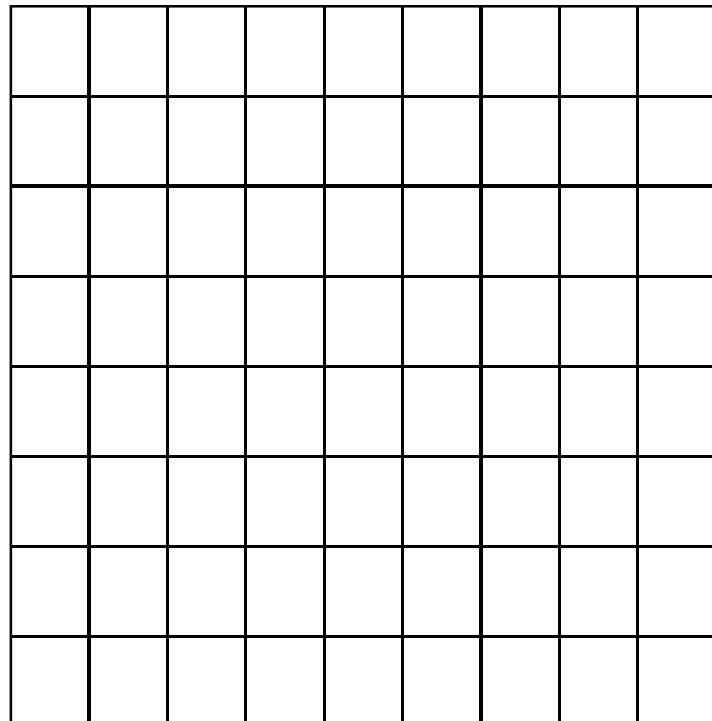
$$0 \leq p \leq M - 1 \quad 0 \leq q \leq N - 1$$

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}} & , p = 0 \\ \sqrt{\frac{2}{M}} & , 1 \leq p \leq M - 1 \end{cases}$$

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}} & , q = 0 \\ \sqrt{\frac{2}{N}} & , 1 \leq q \leq N - 1 \end{cases}$$

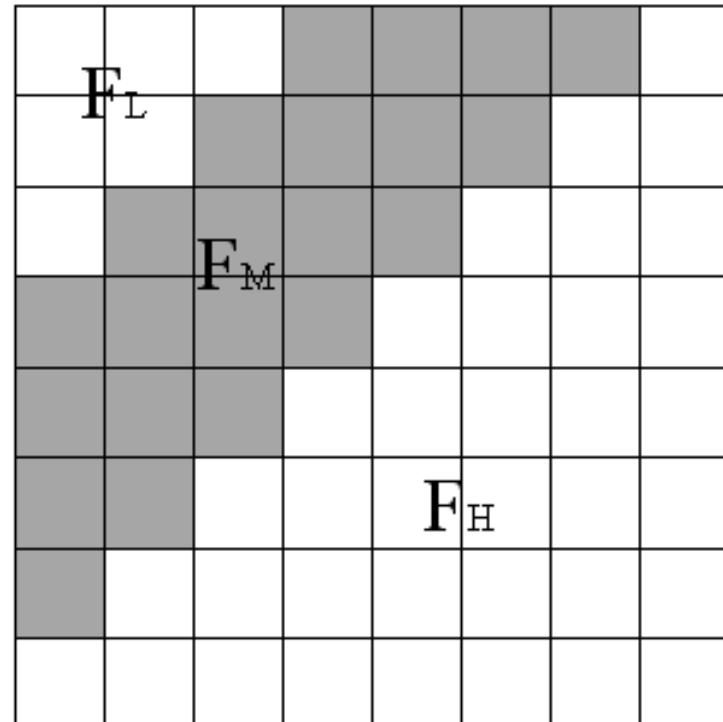
# JPEG-4

Piksel 8x8



DCT  
→

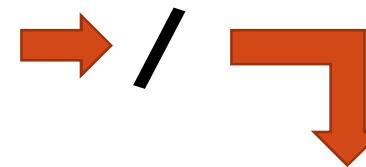
Koefisien DCT



# JPEG-5

-89.00	-63.47	18.21	-6.85	7.50	13.45	-7.00	0.13
74.14	-2.90	-19.93	-21.04	-17.88	-10.81	8.29	5.26
-63.65	3.10	5.08	14.82	10.12	9.33	1.31	-0.62
3.73	2.85	6.67	8.99	-3.38	1.54	1.04	-0.62
2.50	0.57	-4.46	0.52	3.00	-2.89	-0.32	1.33
7.52	-1.80	-0.63	-0.10	0.41	-3.21	-2.74	-2.07
-3.40	0.43	0.81	0.28	-0.40	-0.19	-0.58	-1.09
-2.26	-0.88	1.73	0.23	-0.21	-0.12	1.23	1.61

-6	-6	2	0	0	0	0	0
6	0	-1	-1	-1	0	0	0
-5	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Matriks Kuantisasi JPEG

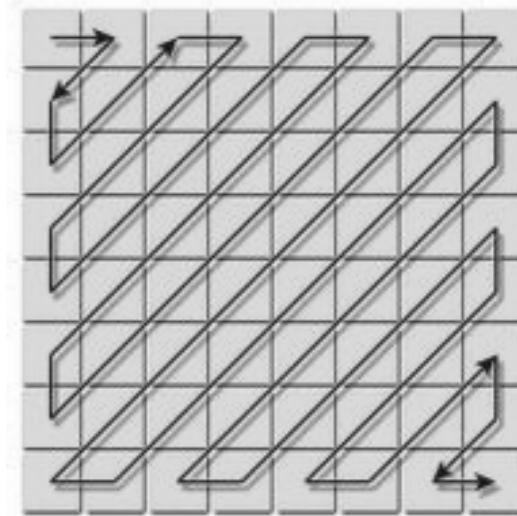


Bulatkan ke integer terdekat

$$\frac{1.6}{9.5} \approx 0$$

# JPEG-6

-6	-6	2	0	0	0	0	0
6	0	-1	-1	-1	0	0	0
-5	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

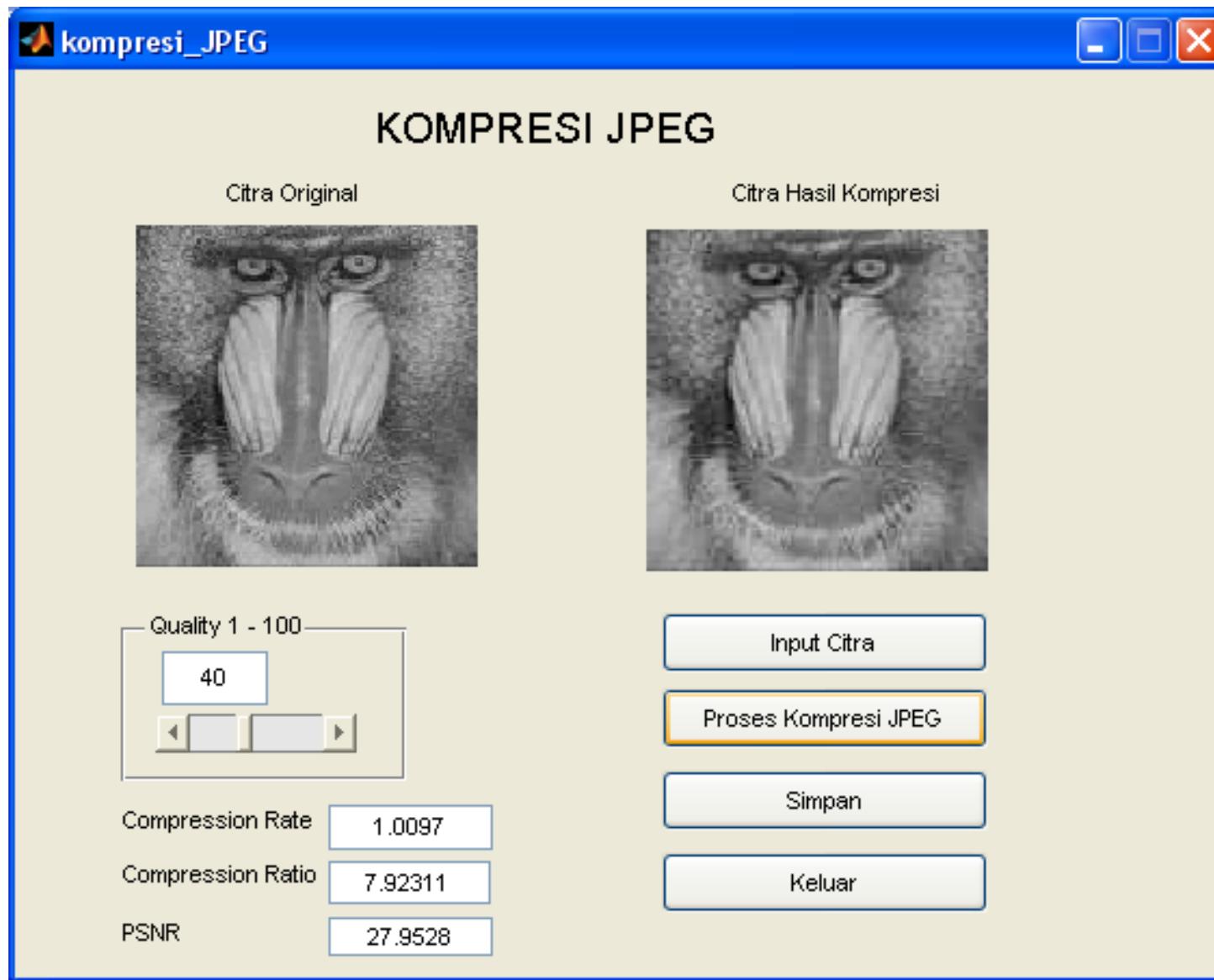


Sehingga dari vektor hasil dari *zig-zag scan* untuk *AC coefficient* yaitu :

-6 -6 6 -5 0 2 0 -1 0 0 0 0 0 -1 0 0 -1 1 0 0 0 0 0 0 0 0 0 0 0 0  
0 0

RLE atau Huffman Code dari vektor di atas

# JPEG-7



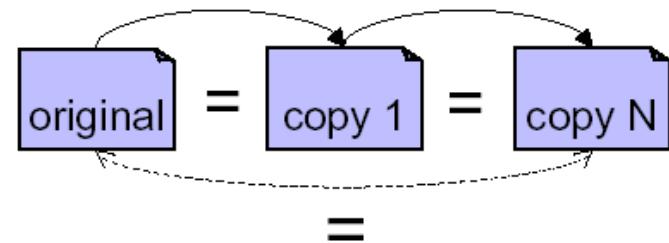
# PENGOLAHAN CITRA DIGITAL

STEGANOGRAPHY

GKV - IF 2020

# Pendahuluan

- Dokumen digital
  - citra (*JPEG/GIF/BMP/TIFF Images*)
  - audio (*MP3/WAV audio*)
  - video (*MPEG video*)
  - teks (*MsWord document*)
- Tepat sama kalau digandakan
- Mudah didistribusikan (misal: via internet)
- Mudah di-edit (diubah)
- Tidak ada perlindungan terhadap kepemilikan, *copyright, editing*, dll.
- Solusi: ***digital watermarking***.



- *Digital watermarking*: penyisipan informasi (disebut *watermark*) ke dalam dokumen digital untuk tujuan:
  - perlindungan *copyright*/kepemilikan
  - *fingerprinting*
  - otentikasi (*integritas content*)
  - dll
- *Watermark* dapat berupa teks, logo, suara, dsb.
- *Watermarking* merupakan aplikasi steganografi.



+ shanty =



Citra semula

Watermark

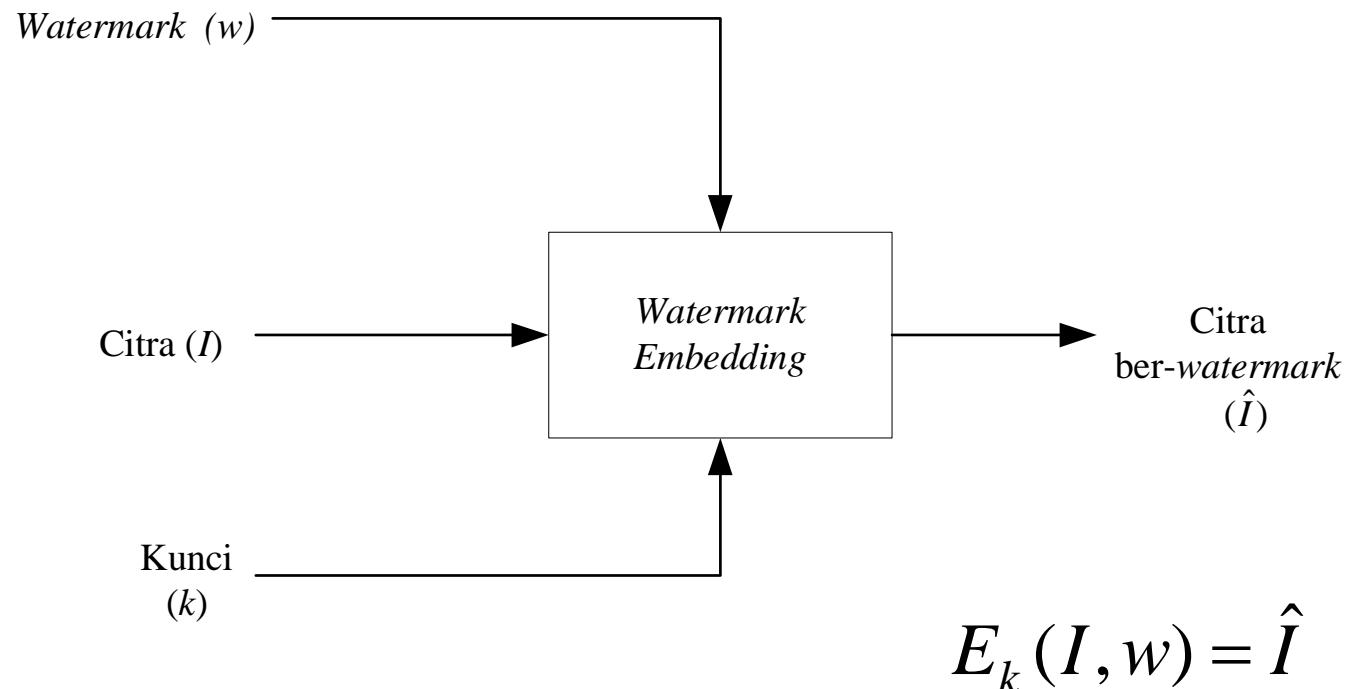
Citra ber-watermark

# Jenis-jenis Watermarking

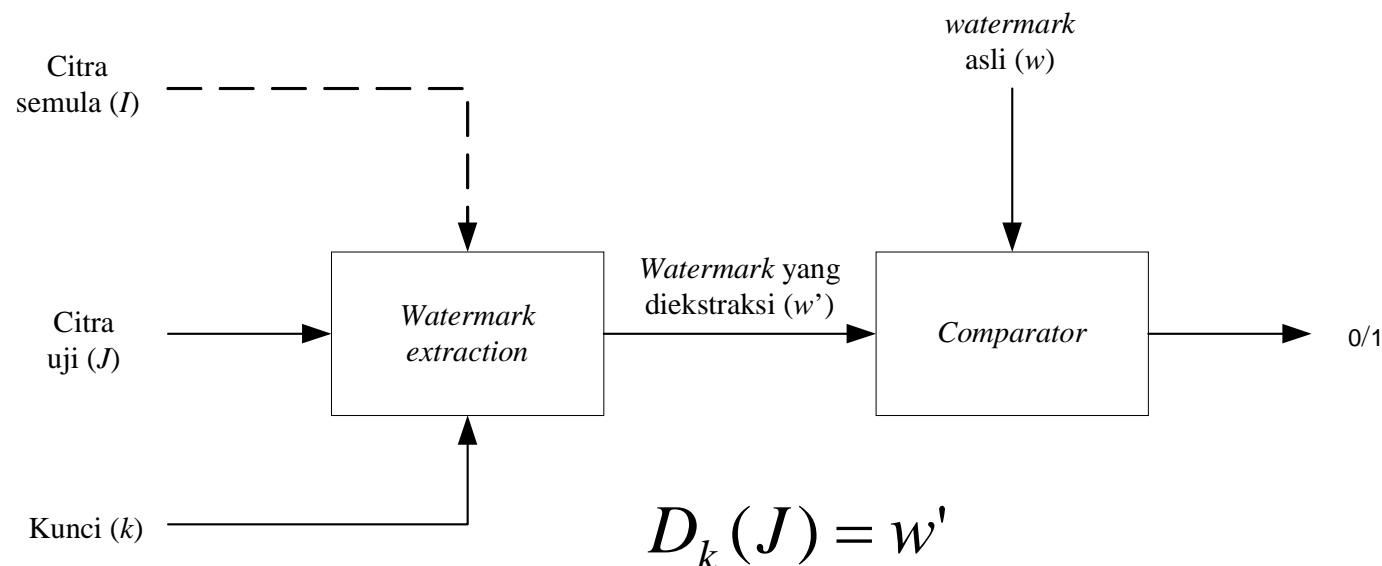
- Bergantung media yang di-watermark, *watermarking* ada beberapa jenis:
  - *Image Watermarking*
  - *Video Watermarking*
  - *Audio Watermarking*
  - *Text Watermarking*

# Digital Image Watermarking

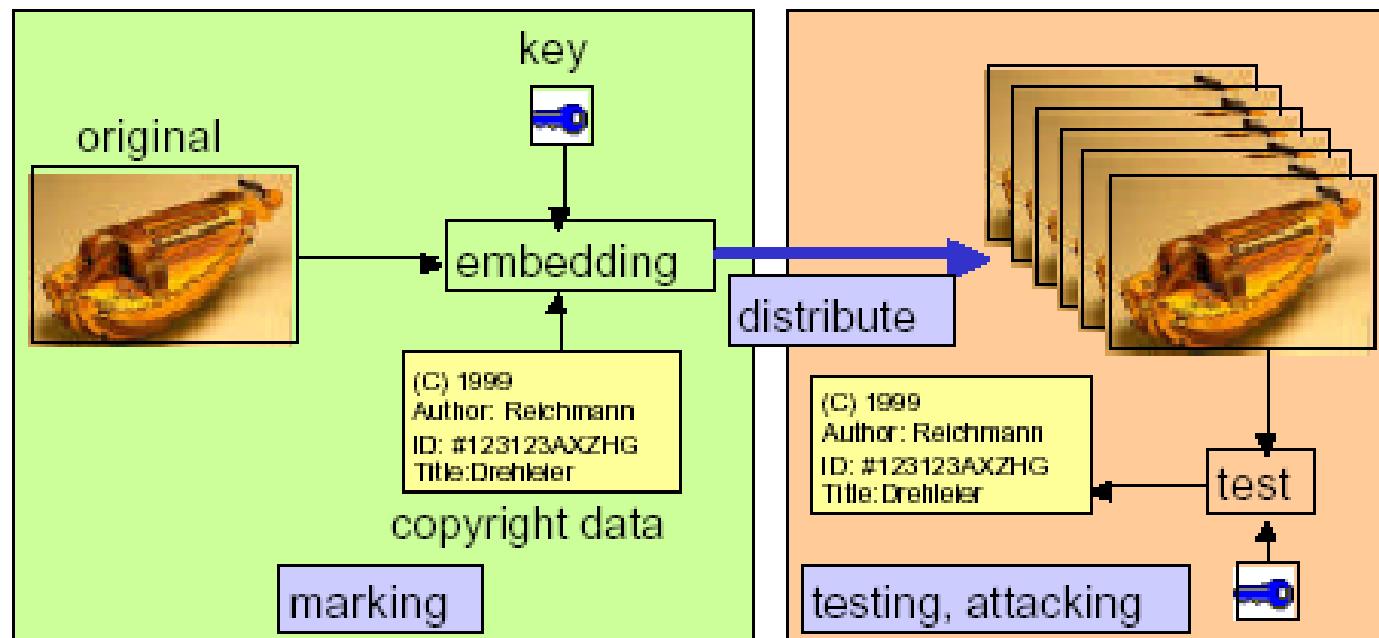
- Penyisipan watermark



- Ekstraksi/deteksi *watermark*



$$C_t(w, w') = \begin{cases} 1, & c \leq t \\ 0, & c > t \end{cases}$$





(a)



(b)



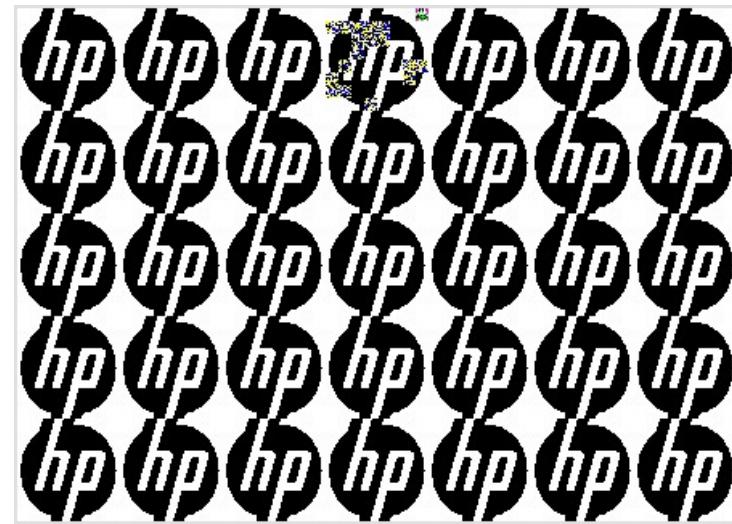
(c)



(d)



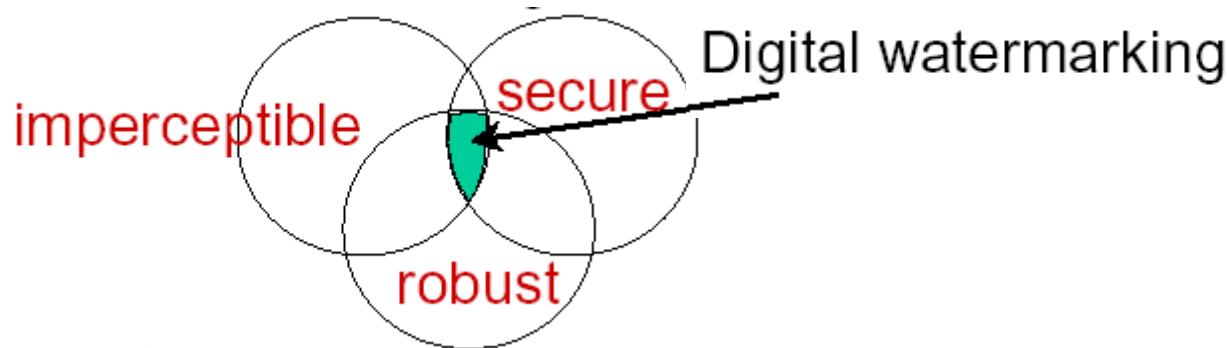
(e)



(f)

- *Watermark* dapat dianggap sebagai **sidi<sup>k</sup> digital** (*digital signature*) atau stempel digital (*finger print*) dari pemilik yang sah atas produk multimedia tersebut.
- Pemberian *signature* dengan teknik *watermarking* ini dilakukan sedemikian sehingga informasi yang disisipkan tidak merusak data digital yang dilindungi.

- Persyaratan umum *watermarking*:
  - *imperceptible*: *watermark* tidak dapat dipersepsi secara visual/auditori karena *watermark* tidak boleh merusak kualitas media *host*.
  - *robustness*: kokoh terhadap manipulasi yang ditujukan untuk merusak atau menghapus *watermark*.
  - *secure*: hanya pihak yang punya otoritas dapat mengakses *watermark*.



# Perbedaan Steganografi dan Watermarking

## Steganografi

- Tujuan: mengirim pesan rahasia apapun tanpa menimbulkan kecurigaan
- Persyaratan: aman, sulit dideteksi, sebanyak mungkin menampung pesan (*large capacity*)
- Komunikasi: *point-to-point*
- Media penampung tidak punya arti apa-apa (*meaningless*)

## *Watermarking:*

- Tujuan: perlindungan *copyright*, pembuktian kepemilikan (*ownership*), *fingerprinting*
- Persyaratan: *robustness*, sulit dihapus (*remove*)
- Komunikasi: *one-to-many*
- Komentar lain: media penampung justru yang diberi proteksi, *watermark* tidak rahasia, tidak mementingkan kapasitas *watermark*

# Jenis-jenis Watermarking

- *Fragile watermarking*

Tujuan: untuk menjaga integritas/orisinalitas media digital.

- *Robust watermarking*

Tujuan: untuk menyisipkan informasi kepemilikan media digital.

## Watermarking pada Citra

- *Visible Watermarking*
- *Invisible Watermarking*

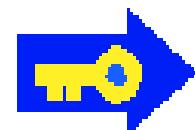
# *Visible Watermarking*



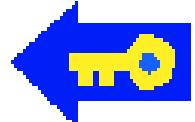
# *Visible Watermarking*



Embed



Remove



# *Invisible Watermarking*



# Aplikasi Watermark

- Memberi label kepemilikan (*ownership*) pada karya digital
- Melindungi isi karya digital (*copyright*).
- Memeriksa integritas isi karya digital (*tamper proofing*) → *Data authentication*
- *User authentication/fingerprinting*: mengotentikasi pengguna spesifik.  
Contoh: distribusi DVD
- Aplikasi medis: foto sinar-X diberi *watermark* berupa ID pasien (memudahkan identifikasi pasien).
- *Convert communication*: untuk sistem komunikasi di negara2 di mana kriptografi tidak dibolehkan.
- *Piracy protection*: mencegah penggandaan yang tidak berizin.

# Sejarah Watermarking

- Abad 13, pabrik kertas di Fabriano, Italia, membuat kertas yang diberi *watermark* dengan cara menekan bentuk cetakan gambar pada kertas yang baru setengah jadi.
- Ketika kertas dikeringkan terbentuklah suatu kertas yang ber-*watermark*. Kertas ini biasanya digunakan oleh seniman/sastrawan untuk menulis karya seni.
- Kertas yang sudah dibubuh tanda-air dijadikan identifikasi bahwa karya seni di atasnya adalah milik mereka.

- *Watermark* pada data digital umumnya audio atau gambar.
- *Watermark* berupa teks mengandung kelemahan karena kesalahan satu bit akan menghasilkan hasil teks yang berbeda pada waktu verifikasi (ekstraksi).

## Contoh *robustness*

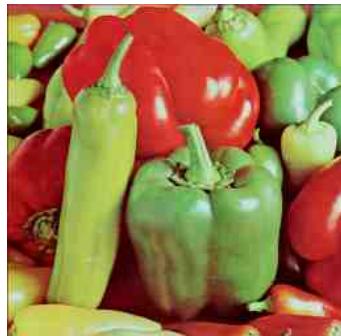
Citra asli



Citra ber-watermark



Citra ber-watermark  
dikompresi 75%



Citra ber-watermark di-crop



## *Domain Image Watermarking*

- *Spasial*

Menyisipkan watermark langsung pada nilai *byte* dari *pixel* citra.

- *Frekuensi*

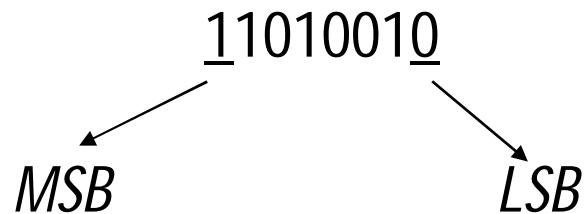
Menyisipkan watermark pada koefisien frekuensi dari citra.

- *Wavelet*

Menyisipkan watermark pada koefisien wavelet dari citra.

## Domain Spasial – *LSB coding*

- Sama seperti steganografi.
- Mengganti bit *LSB* dengan bit data.



*LSB = Least Significant Bit*

*MSB = Most Significant Bit*

- Mengubah bit *LSB* hanya mengubah nilai *byte* satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya

- Misalkan sebagian pixel adalah citra

00110011

10100010

11100010 01101111

(sekelompok *pixel* berwarna merah)

- Misalkan *watermark*: 0111
- *Encoding*:

00110010

10100011 11100011 01101110

(*pixel* berwarna “merah berubah sedikit”)

- Kelemahan:
  1. tidak kokoh terhadap perubahan
  2. mudah dihapus dengan mengganti semua bit *LSB* dari media ber-watermark.

# Metode *Spread Spectrum*

- Diusulkan pertama kali oleh Cox dalam makalah "*Secure Spread Spectrum Watermarking for Multimedia*" (1997)
- *Watermark* disebar (*spread*) di dalam citra.
- *Spread spectrum* dapat dilakukan dalam 2 ranah:
  1. Domain spasial  
Menyisipkan *watermark* langsung pada nilai *byte* dari *pixel* citra.
  2. Domain *transform* (*frekuensi* dan *wavelet*)  
Menyisipkan *watermark* pada koefisien transformasi dari citra.

# DCT

- Penyisipan dalam ranah frekuensi lebih *robust* dibandingkan dalam ranah spasial.
- Pada metode Cox, komponen frekuensi yang disisipi adalah komponen yang signifikan secara persepsi.
- Ada *trade-off* antara *robustness* dan *visibility* ( $\alpha$ )
- Citra ditransformasi ke dalam ranah frekuensi dengan *DCT (Discrete Cosine Transform)*
- Setelah penyisipan, ranah frekuensi dikembalikan ke ranah spasial dengan *IDCT (Inverse Discrete Cosine Transform)*

- *DCT*:  $C(p, q) = \alpha_p \alpha_q \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} I(m, n) \cos \frac{\pi(2m+1)p}{2N} \cos \frac{\pi(2n+1)q}{2N}$
- *IDCT*:  $I(m, n) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \alpha_p \alpha_q C(p, q) \cos \frac{\pi(2m+1)p}{2N} \cos \frac{\pi(2n+1)q}{2N}$
- Keterangan: Citra berukuran  $M \times N$

$$0 \leq p \leq M - 1 \quad 0 \leq q \leq N - 1$$

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}} & , p = 0 \\ \sqrt{\frac{2}{M}} & , 1 \leq p \leq M - 1 \end{cases} \quad \alpha_q = \begin{cases} \frac{1}{\sqrt{N}} & , q = 0 \\ \sqrt{\frac{2}{N}} & , 1 \leq q \leq N - 1 \end{cases}$$

# Koefisien DCT

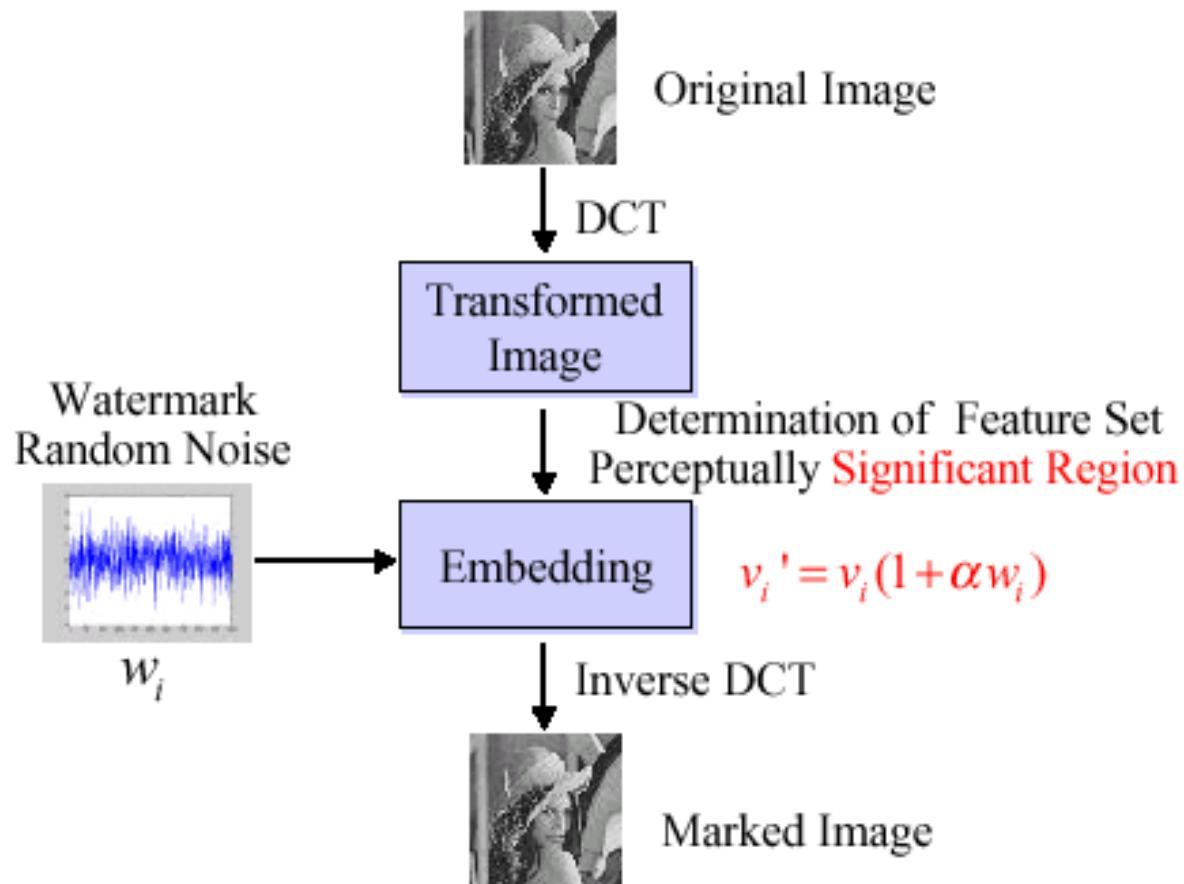
The diagram consists of a 6x6 grid of squares. The squares are shaded in two patterns: a diagonal band of 6 squares from the top-left to the bottom-right is shaded gray; all other squares are white. The label  $F_L$  is positioned at the top-left corner (white square). The label  $F_M$  is positioned in the center column of the second row (gray square). The label  $F_H$  is positioned in the center column of the fifth row (white square).

- $\text{Watermark } W = w_1, w_2, \dots, w_n$
- $\text{Watermark}$ : bilangan riil acak (*pseudo-noise*) yang mempunyai distribusi Normal:

$$p(w) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{w^2}{2\sigma^2}\right)$$

- Cox memilih  $\text{watermark}$  mempunyai distribusi  $N(0, 1)$ , yaitu *mean* = 0, variansi = 1.
- Menurut Cox,  $\text{watermark}$  tsb mempunyai kinerja lebih baik daripada data yang terdistribusi *uniform*.

- Penyisipan *watermark*:



- Pendeksiyan *watermark*:

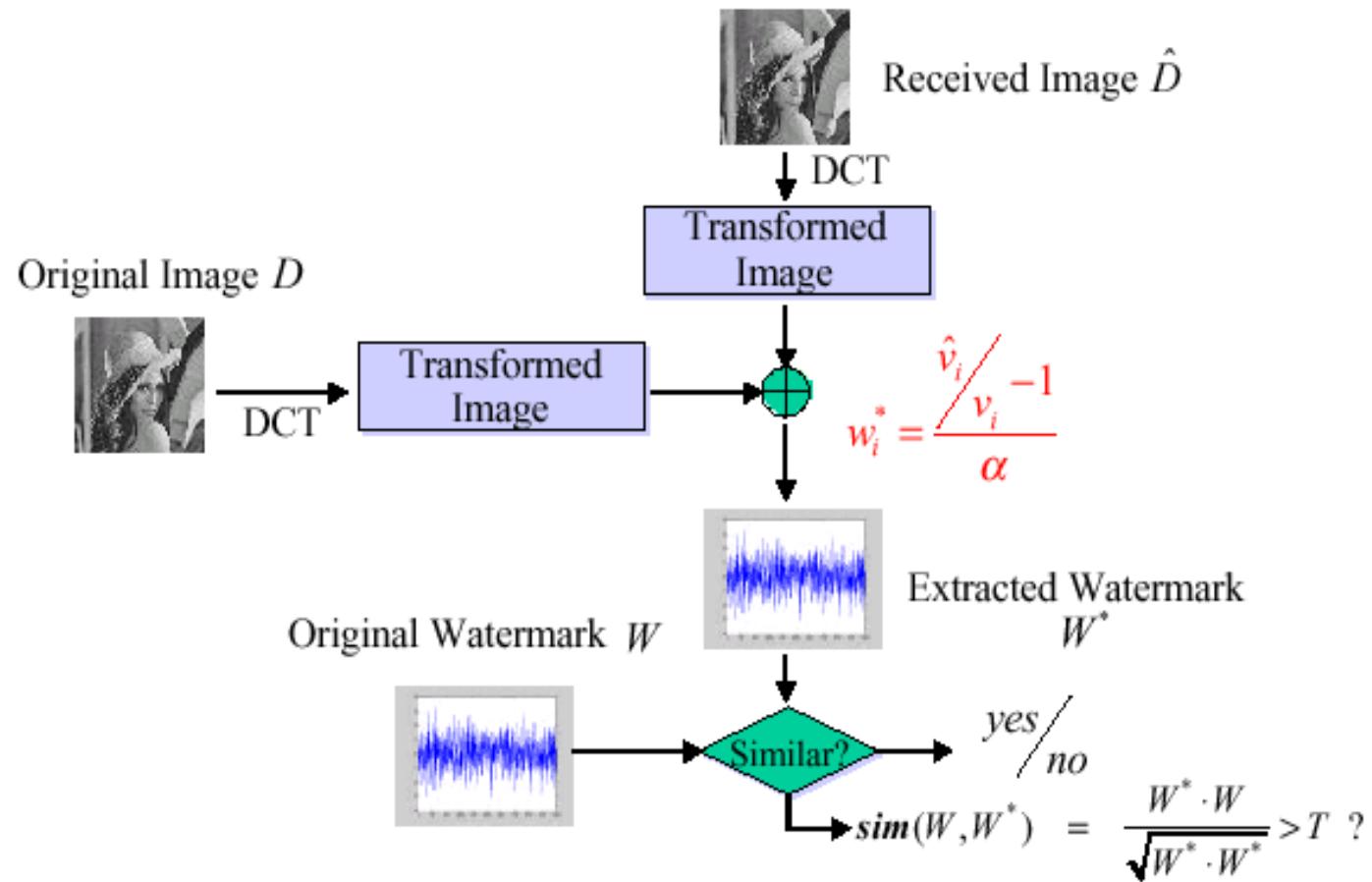




Fig. 4. Bavarian couple image courtesy of Corel Stock Photo Library.

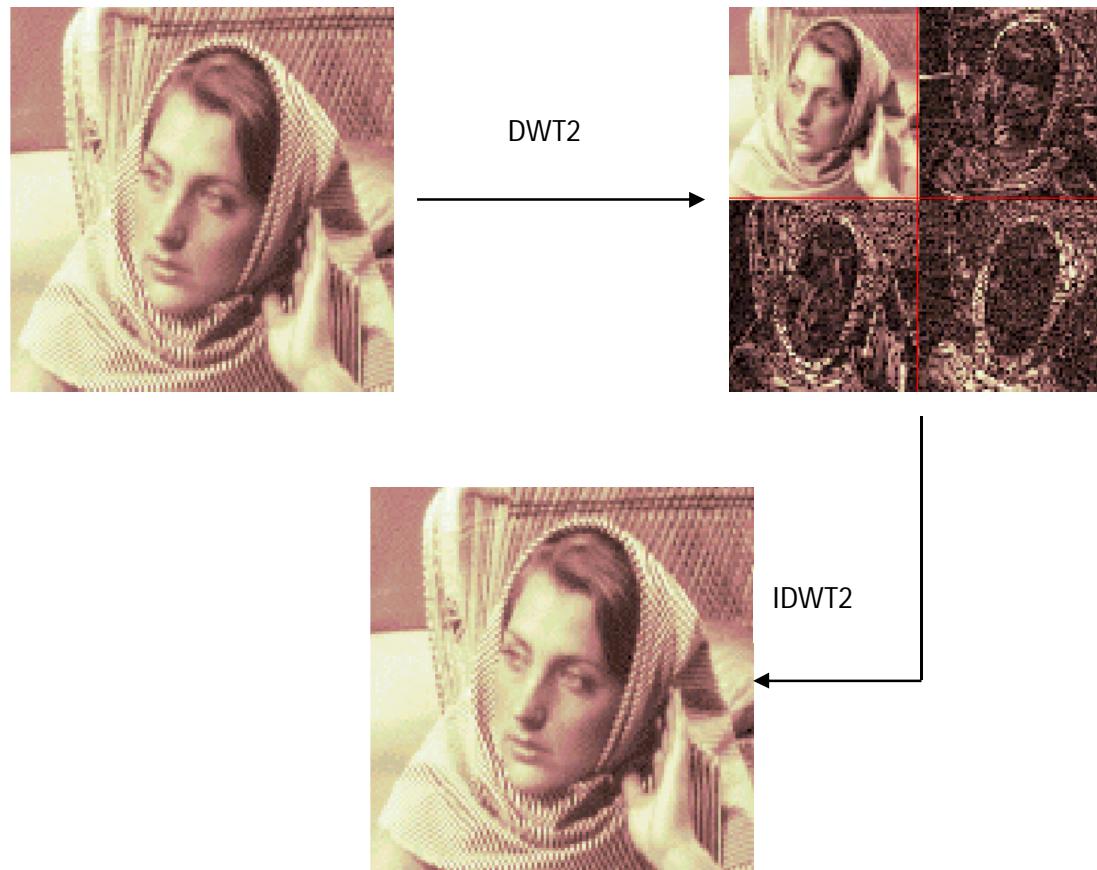


Fig. 5. Watermarked version of Bavarian couple.

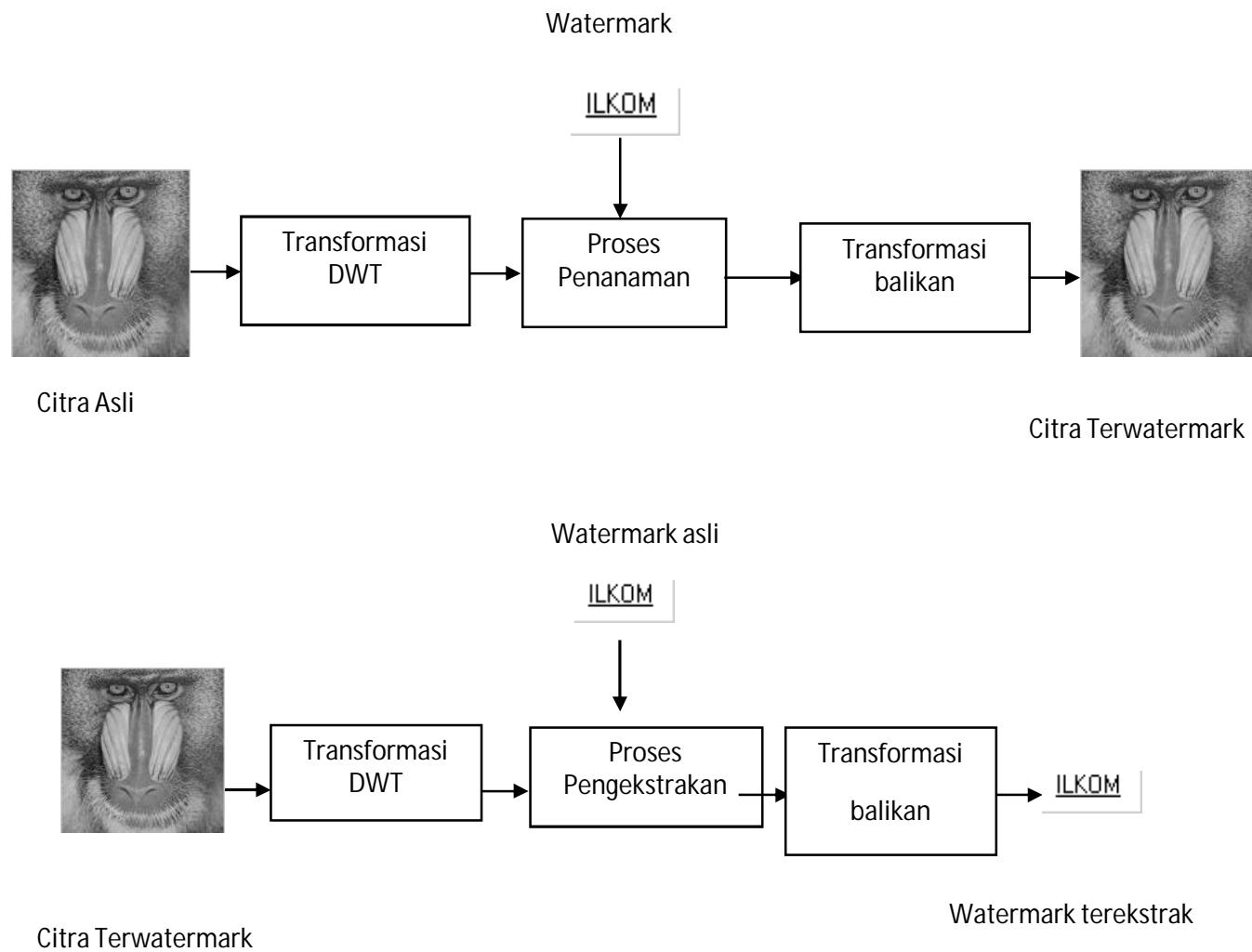
- Panjang *watermark* =  $n = 1000$
- Cox menggunakan 1000 koefisien terbesar. Inilah yang dinamakan *frequency spreading*.
- Cox memilih  $\alpha = 0.1$  dan  $T = 6$
- Kelemahan: perlu citra asli untuk deteksi *watermark* (*non-blind watermarking*).
- Kelebihan: kokoh terhadap
  - konversi analog-ke-digital
  - Konversi digital-ke-analog
  - *Cropping*
  - Kompresi, rotasi, translasi, dan penskalaan

# Domain Wavelet

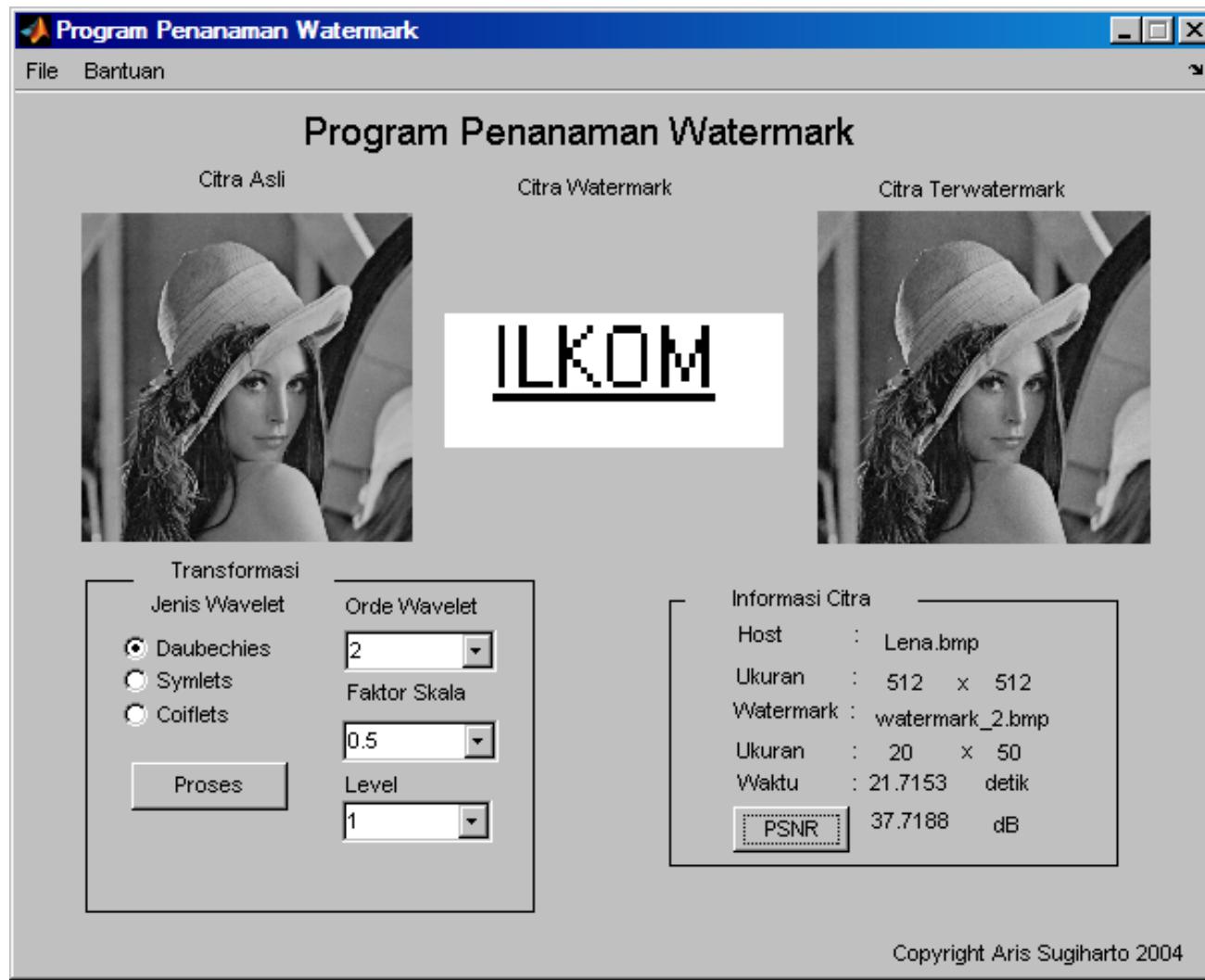
- Merubah Citra menjadi koefisien wavelet



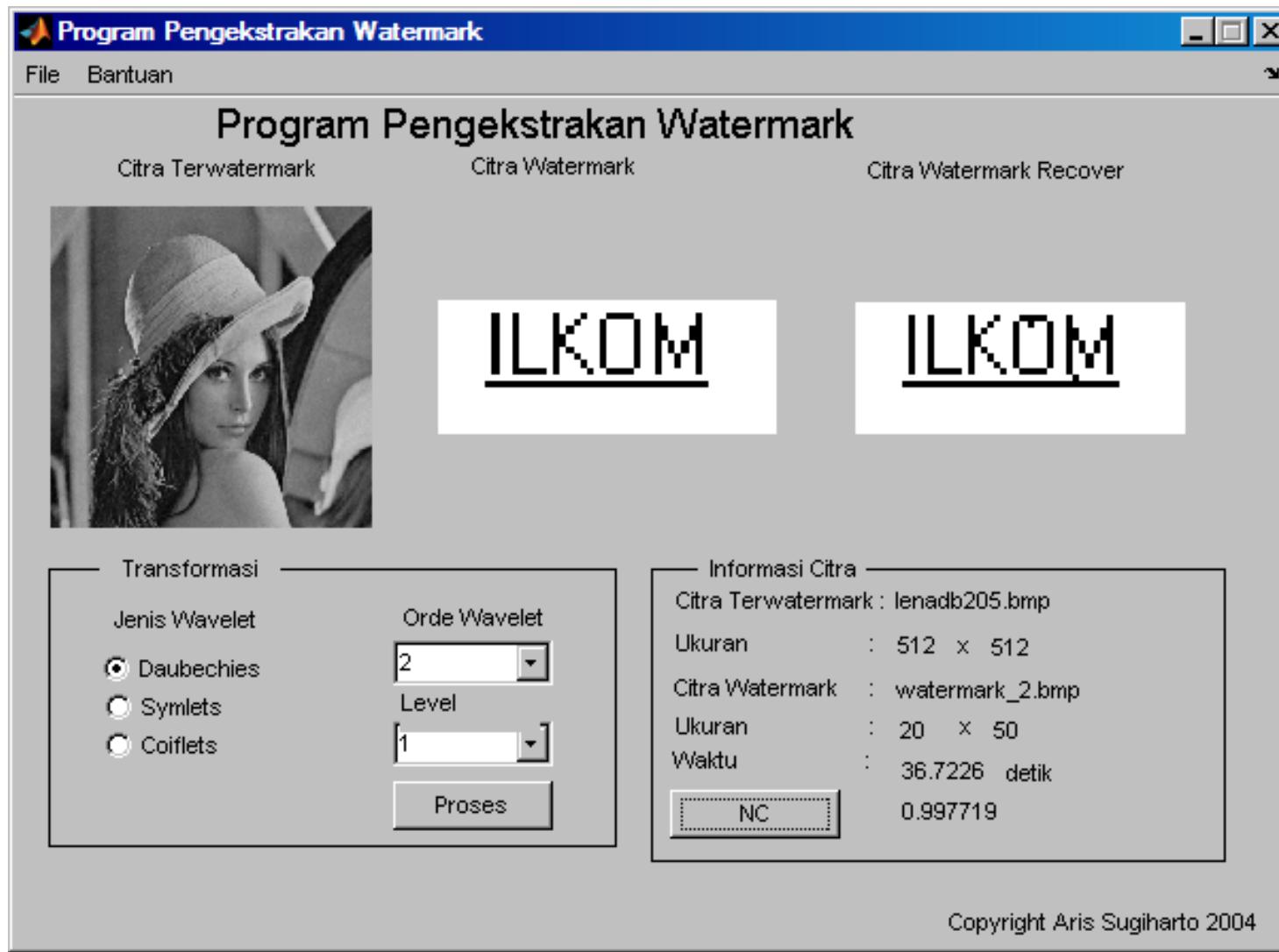
# Operasi Penanaman dan Pengekstrakan



# Interface Penanaman watermark



# Interface Pengekstrakan watermark



# Serangan (attack)

- Serangan Pasif
- Serangan Aktif