



Perbaikan Citra (Image Restoration)

IMAGE RESTORATION

- || Seperti halnya Image Enhancement, **tujuan utama teknik restorasi** adalah untuk meningkatkan kualitas suatu citra
- || Restorasi berupaya untuk **merekonstruksi (reconstruct)** atau mendapatkan kembali (recover) suatu citra yang telah mengalami penurunan kualitas (degraded) dengan menggunakan pengetahuan mengenai fenomena degradasi
- || **Teknik restorasi :**
 - || memodelkan degradasi dan menerapkan proses inverse yang bertujuan untuk memulihkan citra asli

IMAGE RESTORATION

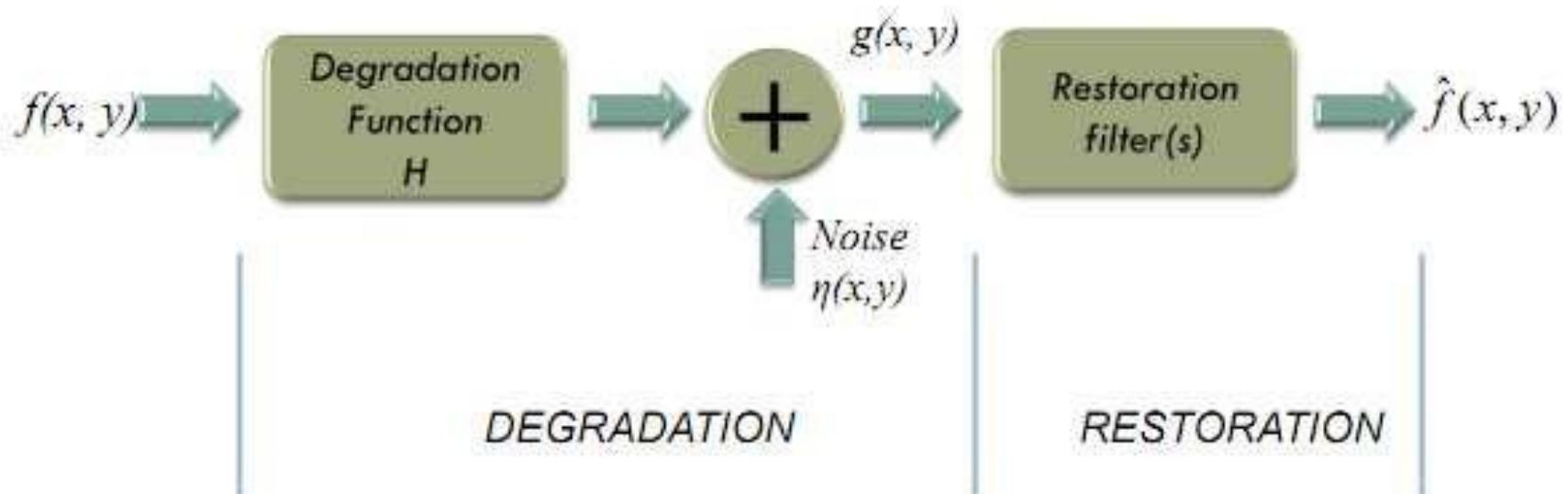
I **Image Enhancement :**

- memperbaiki kualitas citra untuk tujuan tertentu atau bahkan memberi efek berlebih pada citra

I **Image Restoration :**

- memperbaiki suatu citra yang terkena noise (model noise sudah diketahui atau diduga sebelumnya)

MODEL DEGRADASI CITRA/PROSES RESTORASI



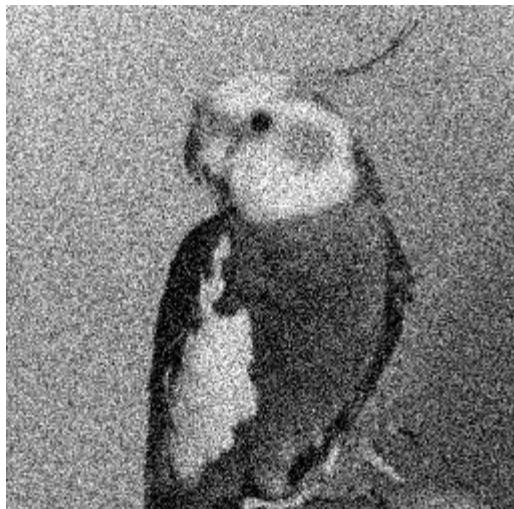
SUMBER NOISE

- Setiap gangguan pada citra dinamakan dengan **noise**
- Noise bisa terjadi :
 - Pada saat proses capture (pengambilan gambar), ada beberapa gangguan yang mungkin terjadi, seperti :
 - Kamera tidak fokus
 - Munculnya bintik-bintik yang bisa jadi disebabkan oleh proses capture yg tdk sempurna
 - Adanya kotoran-kotoran yang terjadi pada citra

KARAKTERISTIK NOISE

- Berdasarkan bentuk dan karakteristiknya, noise pada citra dibedakan menjadi beberapa macam, yakni sebagai berikut :
 - **Gaussian**
 - Merupakan model noise yg mengikuti distribusi normal standard dengan rata-rata nol dan standard deviasi 1
 - **Efek dari noise** ini adalah munculnya titik-titik berwarna yg jumlahnya sama dengan prosentase noise.
 - **Speckle**
 - Merupakan model noise yg memberikan warna hitam pada titik yg terkena noise
 - **Noise salt & pepper**
 - Memberikan noise seperti halnya taburan garam, akan memberikan warna putih pada titik yang terkena noise.

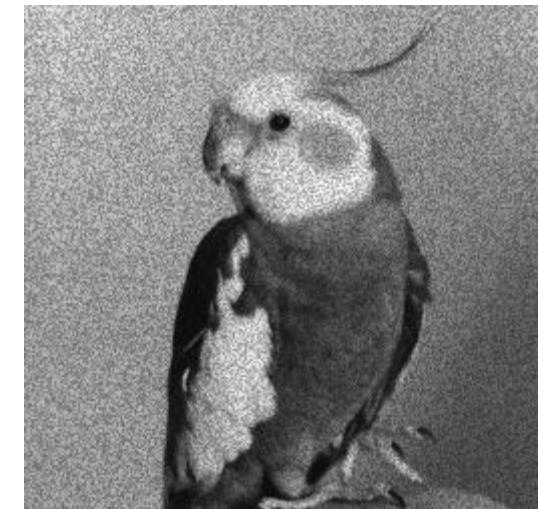
GAMBAR NOISE



A



B



C

Macam-macam Noise (A). Gaussian (B). Salt & Pepper (C) Speckle

NOISE GAUSSIAN

- || Dibuat dengan cara membangkitkan bilangan random [0,1] dengan **distribusi Gaussian**
- || Untuk piksel yang terkena noise, nilai fungsi citra ditambahkan dengan noise yang ada, atau dirumuskan dengan :

$$y(i, j) = x(i, j) + p.a$$

- || Dimana :
 - || a : Bilangan acak berdistribusi Gaussian
 - || p : Prosentase noise
 - || $y(i,j)$: nilai citra yang terkena noise
 - || $x(i,j)$: nilai citra sebelum kena noise

NOISE UNIFORM

- || Noise Uniform seperti halnya noise gaussian dapat dibangkitkan dengan cara membangkitkan bilangan acak [0,1] dengan distribusi uniform.
- || Kemudian untuk titik-titik yang terkena noise, nilai fungsi citra ditambahkan dengan nilai noise yang ada, atau dirumuskan dengan:

$$y(i, j) = x(i, j) + p.a$$

- || Dimana :
 - || a : Bilangan acak berdistribusi Uniform dari noise
 - || p : Persentase noise
 - || $y(i,j)$: nilai citra yang terkena noise
 - || $x(i,j)$: nilai citra sebelum kena noise

- | Untuk membangkitkan bilangan acak berdistribusi Gaussian, tidak dapat langsung menggunakan fungsi rnd, tetapi diperlukan suatu metode yang digunakan untuk mengubah distribusi bilangan acak ke dalam fungsi f tertentu

$g = imnoise(f, 'gaussian', m, var)$

*Default untuk m = 0 dan var =
0.01*

NOISE SPECKLE

- Noise ini dapat dibangkitkan dengan cara membangkitkan bilangan 0 (warna hitam) pada titiktitik yang secara probabilitas lebih kecil dari nilai probabilitas noise, dan dirumuskan dengan

$$f(x, y) = 0 \text{ jika } p(x, y) < \text{ProbNoise}$$

- **Dimana :**
 - $f(x,y)$ adalah nilai gray-scale pada titik (x,y)
 - $p(x,y)$ adalah probabilitas acak
- Jika menggunakan Matlab bisa menggunakan perintah :
 $g = imnoise(f, 'speckle', var)$
Default nilai var = 0.04

NOISE SALT & PEPPER

|| Noise ini dapat dibangkitkan dengan cara membangkitkan bilangan 255 (warna putih) pada titik-titik yang secara probabilitas lebih kecil dari nilai probabilitas noise, dan dirumuskan dengan:

$$f(x, y) = 255 \text{ jika } p(x, y) < \text{ProbNoise}$$

|| Dimana :

- || $f(x,y)$ adalah nilai gray-scale pada titik (x,y)
- || $p(x,y)$ adalah probabilitas acak

|| Bila menggunakan Matlab bisa menggunakan perintah :

$g = imnoise(f, 'salt & pepper', d)$

Default nilai $d = 0.05$

MEAN FILTER

- **Arithmetic Mean Filter**

$$\hat{f}(x, y) = \frac{1}{mn} \sum_{(s,t) \in S_{x,y}} g(s, t)$$

- Dapat diimplementasikan dengan menggunakan mask konvolusi yang semua koefisiennya bernilai **1/mn**
- Noise berkurang sebagai akibat dari blurring

$$H = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} \text{ atau ditulis } H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} / 9$$

$$H = \begin{bmatrix} \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \end{bmatrix}$$

MEAN FILTER

- ***Geometric Mean Filter***
- Tiap pixel yang telah dipulihkan (*restored pixel*) diperoleh dari hasil perkalian pixel-pixel pada subimage yang kemudian dipangkatkan dengan $1/mn$
- Lebih mengarah ke smoothing, namun cenderung kehilangan detail citra dalam prosesnya

REDUKSI NOISE MENGGUNAKAN FILTER GAUSSIAN

- Filter gaussian ini sebenarnya hampir sama dengan filter rata-rata hanya ada nilai bobot yang tidak rata seperti pada filter rata-rata, tetapi mengikuti fungsi gaussian sebagai berikut:

$$G(x, y) = \frac{1}{s\sqrt{\pi}} e^{-((x-m_x)^2 + (y-m_y)^2)/2s^2}$$

- dimana:
 - s adalah sebaran dari fungsi gaussian
 - (m_x, m_y) adalah titik tengah dari fungsi gaussian

KERNEL GAUSSIAN

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix} / 13 \quad \text{atau} \quad H = \begin{bmatrix} 0.077 & 0.077 & 0.077 \\ 0.077 & 0.308 & 0.077 \\ 0.077 & 0.077 & 0.077 \end{bmatrix}$$

Harmonic Mean Filter

- I Harmonic mean filter

$$\hat{f}(x, y) = \frac{mn}{\sum_{(s,t) \in S_{x,y}} \frac{1}{g(s,t)}}$$

- I **Baik** digunakan untuk **salt noise**, namun **buruk** digunakan untuk **pepper noise**
- I Selain itu baik juga digunakan untuk model noise yang lain, seperti Gaussian noise

Contraharmonic Mean Filter

I *Contraharmonic mean filter*

$$\hat{f}(x, y) = \frac{\sum_{(s,t) \in S_{x,y}} g(s,t)^{Q+1}}{\sum_{(s,t) \in S_{x,y}} g(s,t)^Q}$$

Contraharmonic Mean Filter

- | Q adalah order dari filter Filter ini sesuai digunakan untuk mengurangi efek salt-and-pepper noise
- | Q positif, filter mengurangi pepper noise
- | Q negatif, filter mengurangi salt noise
- | Q = 0, reduksi noise dengan arithmetic mean filter
- | Q = -1, reduksi noise dengan harmonic mean filter

- | **Arithmetic dan geometric mean filters** → random noise seperti Gaussian atau uniform noise
- | **Contraharmonic filter** → impulse noise, seperti salt- and-pepper noise. Namun kekurangannya, harus diketahui apakah noise gelap atau terang agar dapat menentukan nilai Q yang sesuai

MSE (Mean Square Error)

- I **MSE merupakan** salah satu image quality metrics yang digunakan untuk mengevaluasi algoritma noise reduction
- I **Semakin kecil nilai MSE**, maka **semakin baik** algoritma noise reduction yang digunakan untuk merestorasi citra

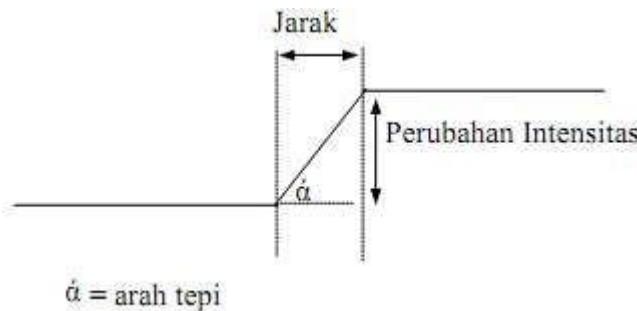
$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [f(x, y) - \hat{f}(x, y)]^2$$



Pendeteksian Tepi (Edge Detection)

DEFINISI TEPI

- **Tepi (Edge) adalah** perubahan nilai intensitas derajat keabuan yang cepat/tiba-tiba (besar) dalam jarak yang singkat



- **Tepi (Edge) dari suatu citra** bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya

DEFINISI TEPI

- **Tepi (Edge) berguna:**
 - Untuk menentukan letak objek didalam suatu citra
 - Menetukan bentuk dan ukuran objek didalam suatu citra
 - Tekstur objek didalam suatu citra

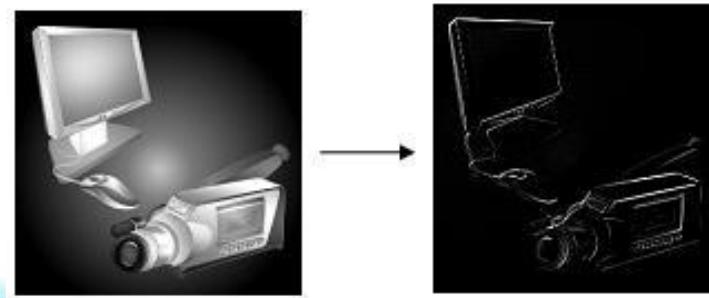
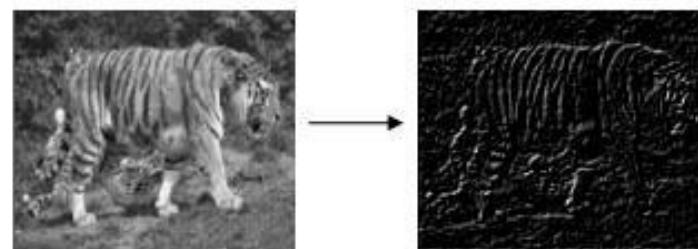
Tujuan Pendektsian Tepi

- **Tujuan pendektsian tepi adalah** untuk meningkatkan penampakan garis batas suatu daerah atau objek didalam citra.



Tujuan Pendektsian Tepi

- **Tujuan pendektsian tepi adalah :**
 - untuk menandai bagian yang menjadi detail citra
 - Untuk memperbaiki detail dari citra yang kabur, yang terjadi karena error atau adanya efek dari proses akuisisi citra



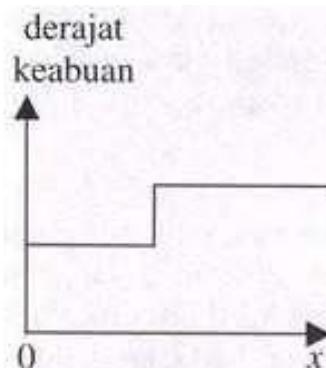
Pendeteksian Tepi

- **Analisis Citra :** ekstraksi ciri – segmentasi – klasifikasi
 - **Ekstraksiciri :** melakukan proses pendeteksian tepi
 - **Segmentasi :** mereduksi citra menjadi objek atau region
 - **Klasifikasi :** memetakan segmen-semen dalam kelas dan objek yang berbeda

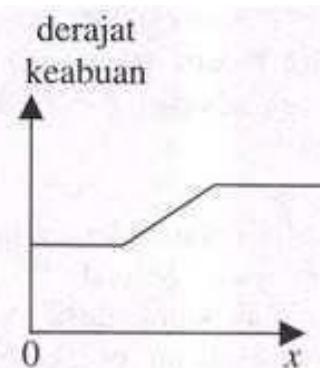
JENIS TEPI

- **Tepi Curam:**
 - Perubahan intensitas tajam, berkisar 90°
- **Tepi Landai:**
 - Tepi lebar, sudut arah kecil. Terdiri dari sejumlah tepi-tepi lokal yang lokasinya berdekatan
- **Tepi mengandung noise :**
 - Perlu dilakukan image enhancement

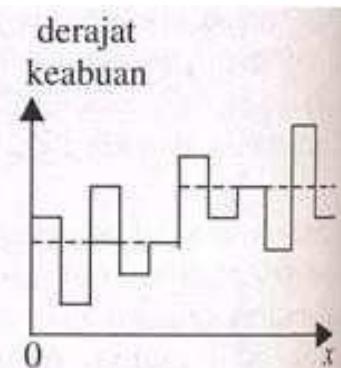
JENIS TEPI



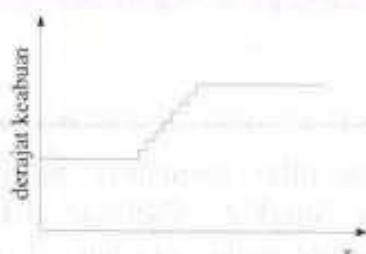
(a) Tepi curam



(b) tepi landai



(c) tepi curam dengan derau



(d) *break down* tepi landai

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

(e) citra dengan tepi

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

(f) citra dengan tepi landai



TEKNIK MENDETEKSI TEPI

□ **TANPA KONVOLUSI**

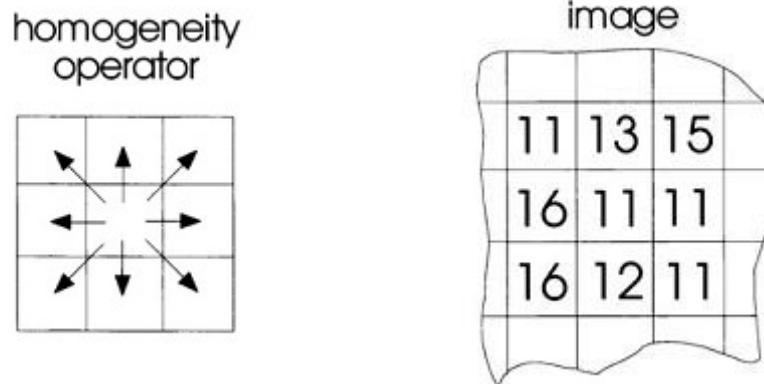
- Homogeneity Operator (Operator Homogenitas)
- Pendeteksian Tepi Selisih (difference)
- Threshold Citra

TEKNIK MENDETEKSI TEPI

- **DENGAN KONVOLUSI**
- **Operator Gradien Pertama :**
 - Differential Gradient
 - Center Difference
 - Sobel
 - Prewit
 - Roberts
- **Operator Turunan Kedua :**
 - Laplacian
 - Laplacian of Gaussian (LoG)
- **Operator Kompas**

TEKNIK MENDETEKSI TEPI (Tanpa Konvolusi)

- **Homogeneity Operator (Operator homogenitas)**
 - Menghitung selisih titik yang dicari dengan titik disekitarnya. Dari selisih tersebut dicari nilai absolut paling besar dan hasilnya digunakan untuk mengganti nilai titik tengah yang dihitung.

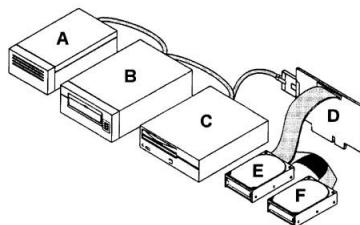


- New Pixel = Maksima; $\{ |11-11| \quad |11-13| \quad |11-15| \quad |11-16| \quad |11-11| \quad |11-16| \quad |11-12| \quad |11-11| \} = 5$

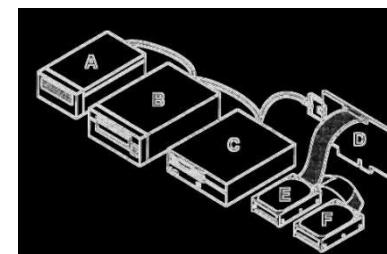
TEKNIK MENDETEKSI TEPI (Tanpa Konvolusi)

□ **Operator homogenitas**

- Output dari operator homogenitas :
 - Nilai terbesar dari harga mutlak 8 selisih



HASIL

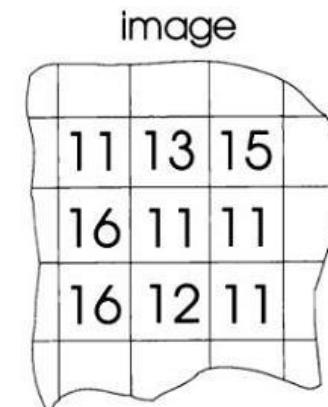
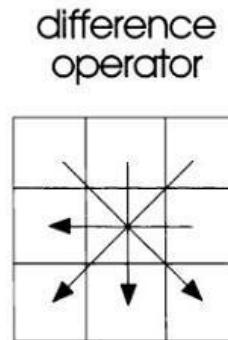


HASIL



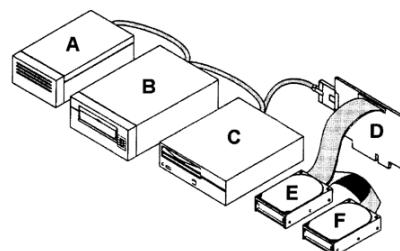
TEKNIK MENDETEKSI TEPI (Tanpa Konvolusi)

- **Pendeteksian tepi selisih (*difference operator*)**
 - Menghitung selisih antara pixel-pixel yang mengelilingi titik tengah dari daerah 3x3 yang dideteksi. Hasil selisih diabsolutkan dan dicari nilai paling besar.
- **New Pixel =**
maximum { |11-11| |13-12| |15-16| |11-16| } = 5
- **OUTPUT**=nilai terbesar dari harga mutlak 4 selisih

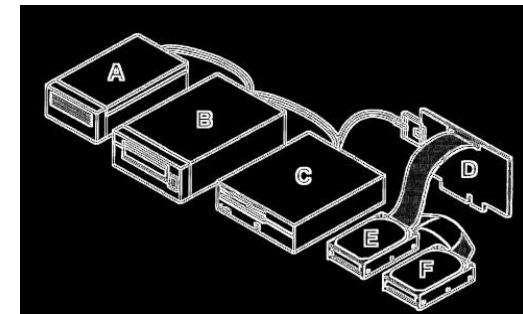


TEKNIK MENDETEKSI TEPI (Tanpa Konvolusi)

- Pendekalian tepi selisih (*difference operator*)



HASIL



HASIL



TEKNIK MENDETEKSI TEPI (Tanpa Konvolusi)

□ **Threshold Citra**

- **Threshold Citra** dapat mempertegas tepi
- **Terdapat 2 jenis threshold citra :**
 - Threshold dengan 1 batas
 - Threshold dengan 2 batas
- **Threshold dengan 1batas :**
 - Pixel diatas batas → nilainya diubah menjadi 255
 - Pixel dibawah batas → nilainya diubah menjadi 0
- **Threshold dengan 2batas (batas 1 <batas 2):**
 - Pixel diatas batas 2 → nilainya diubah menjadi 255
 - Pixel diantara batas-batas → nilainya tidak diubah
 - Pixel dibawah batas 1 → nilainya diubah menjadi 0

TEKNIK MENDETEKSI TEPI (Dengan Konvolusi)

□ Operator Turunan Pertama

- Perubahan intensitas yang besar dalam jarak yang singkat dipandang sebagai fungsi yang memiliki kemiringan yang besar.
- Kemiringan dilakukan dengan menghitung turunan pertama (gradient)

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

□ Dengan G_x dan G_y

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

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

Mask Konvolusi

$$G_1(x) = [-1 \quad 1]$$

$$G_1(y) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

TEKNIK MENDETEKSI TEPI (Dengan Konvolusi)

□ Operator Turunan Pertama

- Kekuatan tepi merupakan magnitudo dari gradien, dapat dihitung dengan :

$$(i) G[f(x,y)] = \sqrt{|G_x|^2 + |G_y|^2}, \text{ atau}$$

$$(ii) G[f(x,y)] = |G_x| + |G_y|, \text{ atau}$$

$$(iii) G[f(x,y)] = \max\{|G_x|^2, |G_y|^2\}, \text{ atau}$$

$$(iv) G[f(x,y)] = \max\{|G_x|, |G_y|\}.$$

- Hasil pendekalian tepi adalah **citra tepi** $g(x,y)$ yang nilai setiap pixelnya menyatakan kekuatan tepi =

$$g(x,y) = G[f(x,y)]$$

TEKNIK MENDETEKSI TEPI (Dengan Konvolusi)

□ **Operator Turunan Pertama**

- Keputusan apakah suatu pixel merupakan tepi atau bukan tepi dinyatakan dengan operasi pengambangan sebagai berikut :

$$g(x, y) = \begin{cases} 1, & \text{jika } G[f(x, y)] \geq T \\ 0, & \text{lainnya} \end{cases}$$

OPERATOR TURUNAN PERTAMA

□ Selisih terpusat(*center-difference*)

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

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

□ Mask Konvolusi

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

OPERATOR TURUNAN PERTAMA

□ Sobel (dengan $\sigma=2$)

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

- Dengan magnitude

$$M = \sqrt{s_x^2 + s_y^2}$$

- Mask Konektivitas

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

OPERATOR TURUNAN PERTAMA

□ Sobel

CONTOH

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

(i) citra semula

$$\begin{bmatrix} * & * & * & * & * \\ * & 18 \end{bmatrix}$$

(ii) hasil konvolusi

Nilai 18 pada citra hasil konvolusi diperoleh dengan perhitungan berikut

$$S_x = (3)(-1) + (2)(-2) + (3)(-1) + (2)(1) + (6)(2) + (7)(1) = 11$$

$$S_y = (3)(1) + (4)(2) + (2)(1) + (3)(-1) + (5)(-2) + (7)(-1) = -7$$

$$M = \sqrt{s_x^2 + s_y^2} = \sqrt{11^2 + (-7)^2} \cong |S_x| + |S_y| = |11| + |-7| = 18$$

OPERATOR TURUNAN PERTAMA

□ **Prewit**

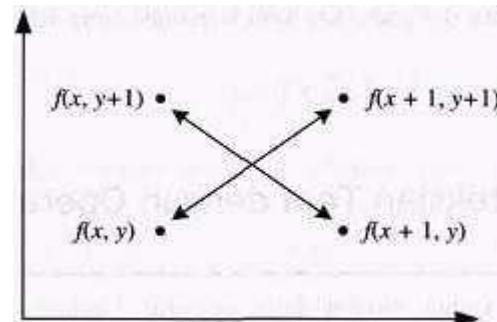
- Sama dengan sobel hanya konstanta yang digunakan adalah c=1

□ **Mask Konvolusi :**

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

OPERATOR TURUNAN PERTAMA

□ Robert (operator silang)



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

dimana R_+ turunan berarah 45° , dan R_- berarah 135°

- Mask Konvolusi :

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

OPERATOR TURUNAN PERTAMA

- **Robert (operator silang)**

- **Mask Konvolusi :**

-1	0
0	1

0	-1
1	0

p_1	p_2	p_3
p_4	p_5	p_6
p_7	p_8	p_9

$$p'_5 = |p_9 - p_5| + |p_8 - p_6|$$

Citra semula

OPERATOR TURUNAN PERTAMA

- **Robert (operator silang)**
- **CONTOH :**

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

(i) citra semula



OPERATOR TURUNAN KEDUA

□ LAPLACIAN

- Disebut pula Operator Turunan Kedua
- Termasuk dalam High Pass Filter
- Lebih akurat khususnya pada tepi-tepi curam
- Turunan keduanya mempunyai persilangan nol (zero crossing), yang merupakan lokasi tepi yang akurat
- Rumus :

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

OPERATOR TURUNAN KEDUA

□ LAPLACIAN

□ Dengan : $G_3(x) = \frac{\partial f(x, y)}{\partial x} = \frac{f(x, y) - f(x - \Delta x, y)}{\Delta x}$

$$G_3(y) = \frac{\partial f(x, y)}{\partial y} = \frac{f(x, y) - f(x, y - \Delta y)}{\Delta y}$$

□ Maka : $\nabla^2 f(x, y) = f(x+1, y) - 2f(x, y) + f(x-1, y) + f(x, y+1) - 2f(x, y) + f(x, y-1)$
 $= f(x, y-1) + f(x-1, y) - 4f(x, y) + f(x+1, y) + f(x, y+1)$

□ Mask Konvolusi :

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \text{ dan } \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

OPERATOR TURUNAN KEDUA

□ LAPLACIAN

- Filter/kernel dari rumus Laplacian

0	1	0
1	-4	1
0	1	0

- Filter/kernel dari rumus laplacian yang diperluas

0	-1	0
-1	4	-1
0	-1	0

- Filter/kernel lain

1	1	1
1	8	1
1	1	1

- Filter Laplacian untuk bobot lebih pada pixel tengah diantara pixel tetangga

1	4	1
4	-20	4
1	4	1

OPERATOR TURUNAN KEDUA

□ LAPLACIAN

□ Contoh deteksi tepi vertikal

$$\begin{array}{ccc|cccc} 4 & 4 & 4 & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & 8 & 8 & 8 & 8 \end{array}$$

(i) Citra semula

$$\begin{bmatrix} * & * & * & * & * & * & * \\ * & 0 & +4 & -4 & 0 & 0 & * \\ * & 0 & +4 & -4 & 0 & 0 & * \\ * & 0 & +4 & -4 & 0 & 0 & * \\ * & * & * & * & * & * & * \end{bmatrix}$$

(ii) Hasil konvolusi

□ Contoh deteksi tepi diagonal

$$\begin{array}{ccccc|cccc} 4 & 4 & 8 & 8 & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & 8 & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & 4 & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & 4 & 4 & 8 & 8 & 8 \\ 4 & 4 & 4 & 4 & 4 & 4 & 8 & 8 \end{array}$$

(i) Citra semula

$$\begin{bmatrix} * & * & * & * & * & * & * & * \\ * & 0 & +8 & -4 & 0 & 0 & 0 & * \\ * & 0 & 0 & +8 & -4 & 0 & 0 & * \\ * & 0 & 0 & 0 & +8 & -4 & 0 & * \\ * & * & * & * & * & * & * & * \end{bmatrix}$$

(ii) Hasil konvolusi

OPERATOR TURUNAN KEDUA

- **LAPLACIAN**
 - **Contoh deteksi tepi landai**

$$\begin{bmatrix} 2 & 2 & 2 & 5 & 8 & 8 & 8 & 8 \\ 2 & 2 & 2 & 5 & 8 & 8 & 8 & 8 \\ 2 & 2 & 2 & 5 & 8 & 8 & 8 & 8 \\ 2 & 2 & 2 & 5 & 8 & 8 & 8 & 8 \\ 2 & 2 & 2 & 5 & 8 & 8 & 8 & 8 \end{bmatrix}$$

(i) Citra semula

$$\begin{bmatrix} * & * & * & * & * & * & * & * \\ * & 0 & +3 & 0 & -3 & 0 & 0 & * \\ * & 0 & +3 & 0 & -3 & 0 & 0 & * \\ * & 0 & +3 & 0 & -3 & 0 & 0 & * \\ * & * & * & * & * & * & * & * \end{bmatrix}$$

(ii) Hasil konvolusi

OPERATOR TURUNAN KEDUA

- **LAPLACIAN of Gaussian Filtering (LoG)**
 - Untuk mengurangi deteksi tepi yang palsu difilter dulu dengan Fungsi Gaussian
 - **Laplacian Gaussian filtering bertujuan** untuk menghilangkan noise dan meningkatkan kualitas detail.
 - **Laplacian bertujuan** untuk meningkatkan kualitas detail (detail enhancement)
 - **Laplacian Operator (HPF) =**

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

OPERATOR TURUNAN KEDUA

□ LAPLACIAN of Gaussian Filtering (LoG)

- Untuk mendeteksi tepi citra yang mengalami gangguan (noise/derau) dapat dilakukan salah satu dari operasi berikut:
 - Konvolusi citra dengan fungsi gauss $G(x,y)$ kemudian dilakukan operasi laplacian terhadap hasilnya atau
 - Konvolusi Citra dengan LoG
- **Mask Konvolusi :**

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

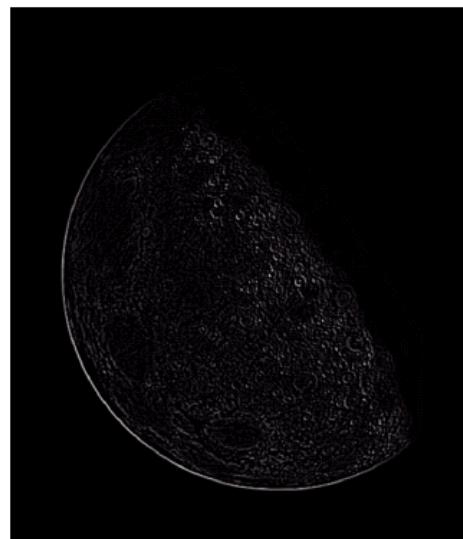
OPERATOR TURUNAN KEDUA

□ LAPLACIAN

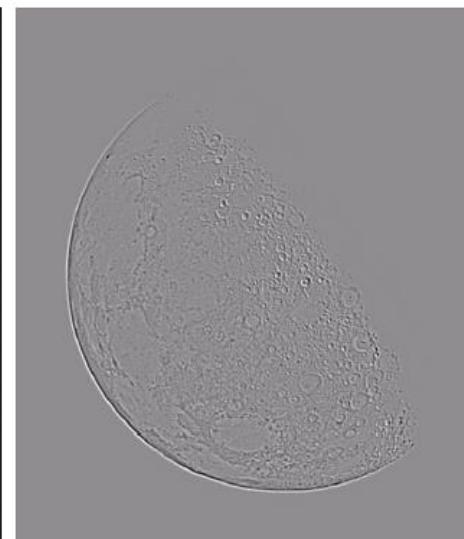
- Applying the Laplacian to an image we get a new image that highlights edges and other discontinuities



Citra asli



Citra terfilter
Laplacian



Citra terfilter laplacian
Diskalakan utk tampilan

OPERATOR TURUNAN KEDUA

□ LAPLACIAN

- Hasil dari filter Laplacian bukan citra dengan kenampakan yang baik
- Untuk itu diperlukan proses lanjutan, yaitu :
 - Mengurangkan citra hasil filter laplacian dari citra aslinya untuk mebentuk hasil citra yang baru

$$g(x, y) = f(x, y) - \nabla^2 f$$

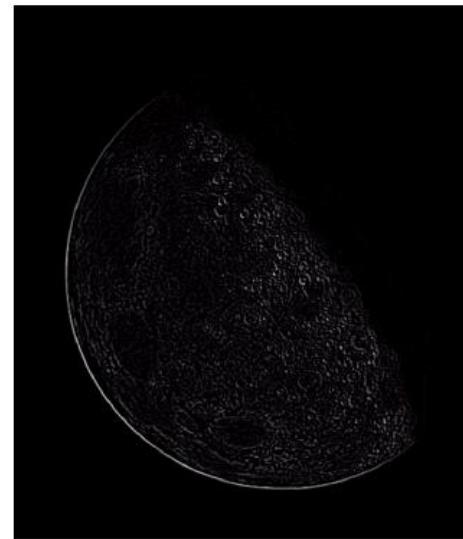
OPERATOR TURUNAN KEDUA

□ LAPLACIAN



Citra
asli

-



Citra terfilter
Laplacian

=



Citra akhir yg
Lbh tajam
kenampakannya

- Pada citra hasil akhir, batas tepi dan kedekilan unsur tampak lebih jelas

OPERATOR TURUNAN KEDUA

□ Operator KOMPAS

- Digunakan untuk mendeteksi semua tepi dari berbagai arah dalam citra
- Menampilkan dari 8 macam arah mata angin

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

- Dilakukan dengan mengkonvolusi citra dengan berbagai mask kompas lalu dicari nilai magnitude (kekuatan tepi) yang terbesar dan arahnya

OPERATOR TURUNAN KEDUA

- **Operator KOMPAS**
- **Operator**

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

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



Spatial Filtering

FILTERING

- Digunakan untuk proses pengolahan citra :
 - Perbaikan kualitas citra (image enhancement)
 - Penghilangan derau (noise)
 - Mengurangi erotan
 - Penghalusan/pelembutan citra g / p
 - Deteksi tepi, penajaman tepi
 - Dll.

TEORI KONVOLUSI(1)

- Untuk mengaplikasikan filtering pada citra, digunakan metode **konvolusi**.
- Konvolusi 2 fungsi $f(x)$ dan $g(x)$

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha)g(x - \alpha)d\alpha$$

α = peubah bantu

- Fungsi Diskrit:

$$f(x) * g(x) = \sum f(\alpha)g(x - \alpha)$$

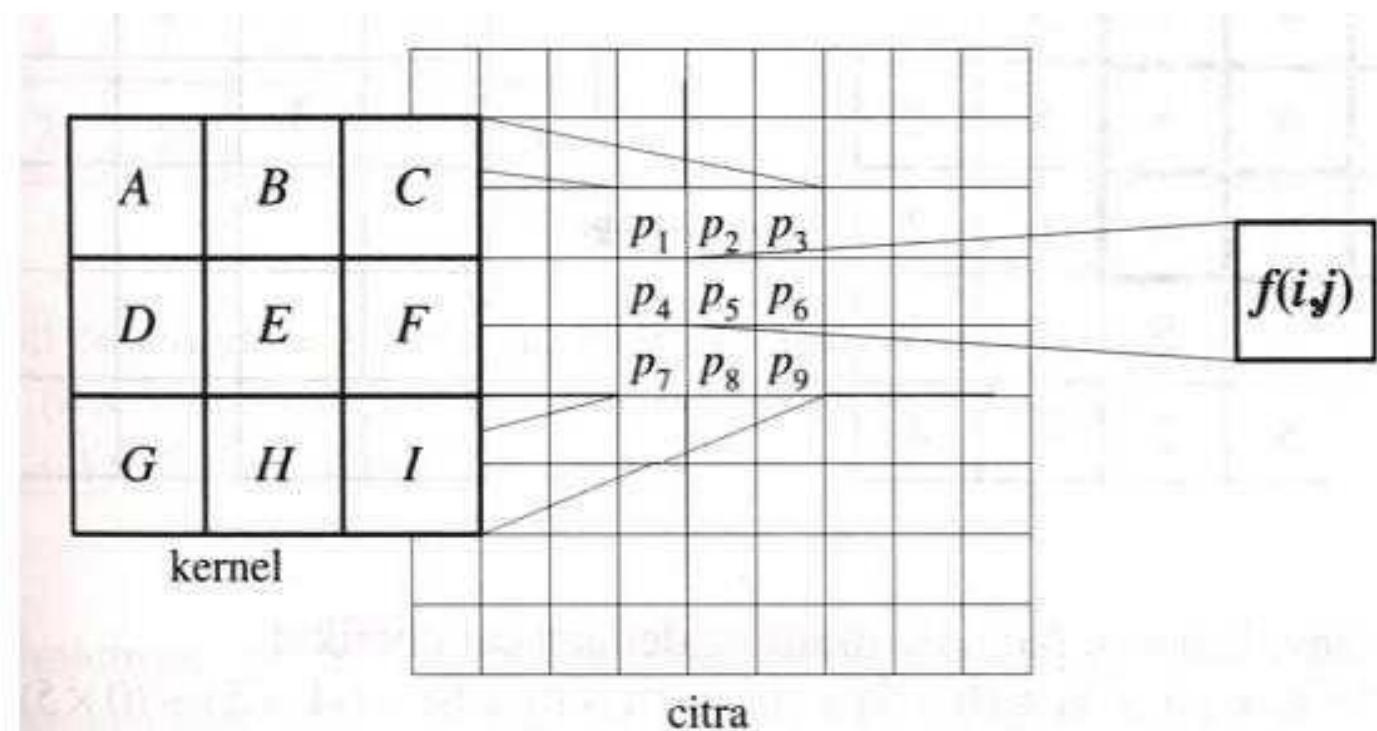
$$f(x, y) * g(x, y) = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f(a, b)g(x-a, y-b)$$

TEORI KONVOLUSI(2)

- $g(x)$ → convolution mask / filter / kernel atau template.
- **Notasi:** $f(x, y) * g(x, y) = f(x, y) \otimes g(x, y)$

- Konvolusi bisa dinyatakan dalam matriks.
- Tiap elemen matriks filter : **koefisien konvolusi**.
- **Operasi konvolusi** → menggeser kernel pixel per pixel - hasil disimpan dalam matriks baru.

ILUSTRASI KONVOLUSI



$$f(i, j) = Ap_1 + Bp_2 + Cp_3 + Dp_4 + Ep_5 + Fp_6 + Gp_7 + Hp_8 + Ip_9$$

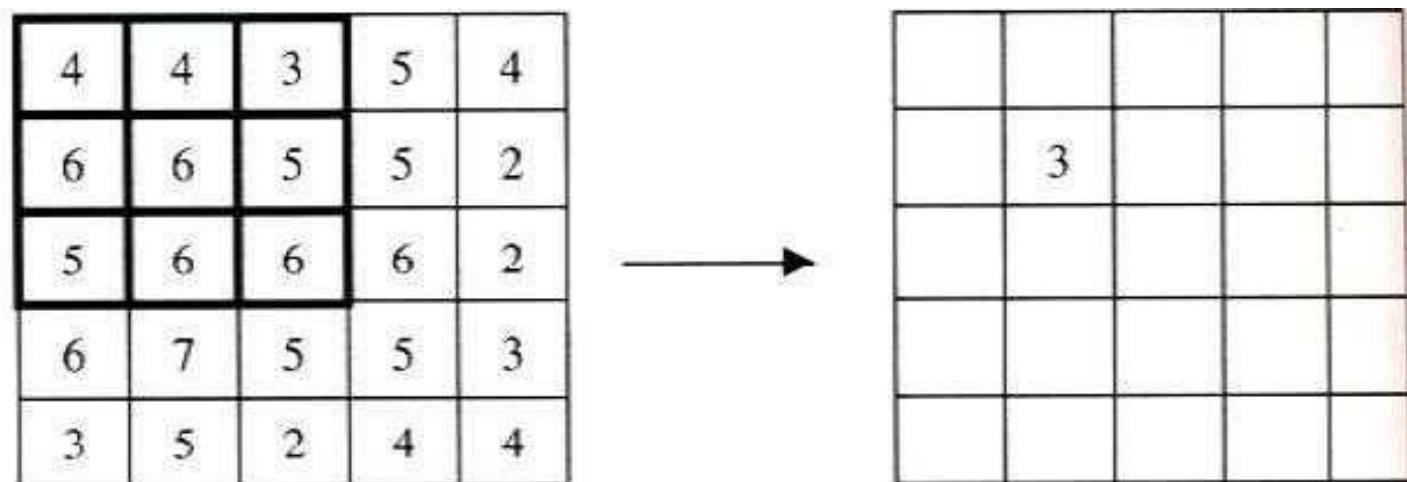
CONTOH KONVOLUSI

- y Citra $f(x,y)$ berukuran 5×5 dan sebuah kernel berukuran 3×3

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

- Tanda \bullet → posisi $(0,0)$ dari kernel

CONTOH KONVOLUSI



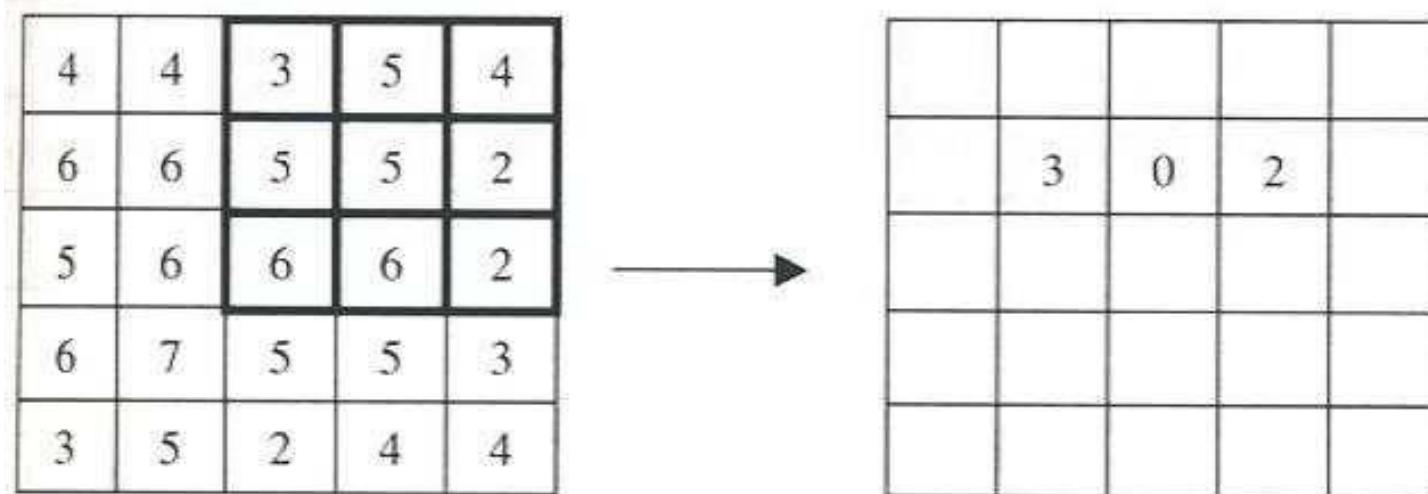
CONTOH KONVOLUSI

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	

CONTOH KONVOLUSI



CONTOH KONVOLUSI

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				

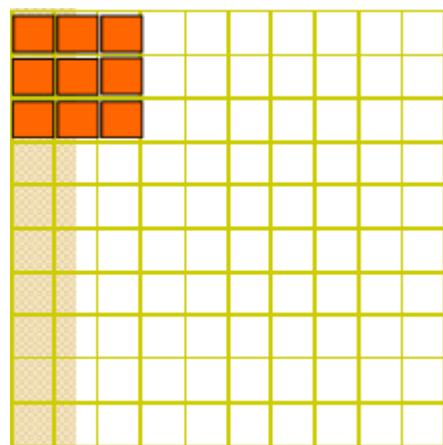
HASIL CONTOH KONVOLUSI

- **Bila hasil konvolusi negatif**, maka nilai dijadikan 0. (clipping)
- **Bila hasil konvolusi > derajat keabuan maksimum**, maka nilai diubah kederajat keabuan maksimum (clipping)

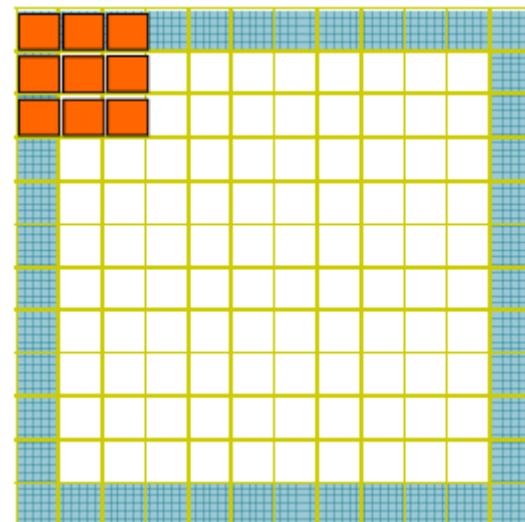
YANG PERLU DIINGAT

- Gunakan citra baru untuk menampung hasil perhitungan
- Selalu gunakan nilai dari citra asli untuk input (bukan nilai pixel hasil perhitungan sebelumnya)
- Bila filter sampai pada pinggir citra, terdapat beberapa pilihan:
 - *Biarkan pixel di pinggir tanpa diproses*
 - *Perlebar citra, pixel di pinggir diisi perulangan nilai pixel pada pinggir tersebut*
 - *Perlebar citra, pixel di pinggir diisi konstanta tertentu*
 - *Perlebar citra dengan melakukan image warping*

MEMBERI ELEMEN TAMBAHAN



padding



diberi kolom dan baris tambahan, dan diisi dengan



nilai 0 (nol), atau



konstanta

Hasil Konvolusi Pinggir Diabaikan

- Solusi ketiga elemen pinggir tadi mengasumsikan bahwa pixel pinggir berukuran amat kecil → mata tidak bisa melihat.

4	4	3	5	4
6	4	0	8	2
5	0	2	6	2
6	6	0	2	3
3	5	2	4	4

Contoh Aplikasi Konvolusi



(a) Citra Lena semula

$$* \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} =$$

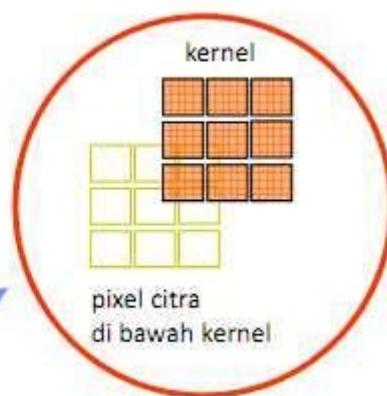
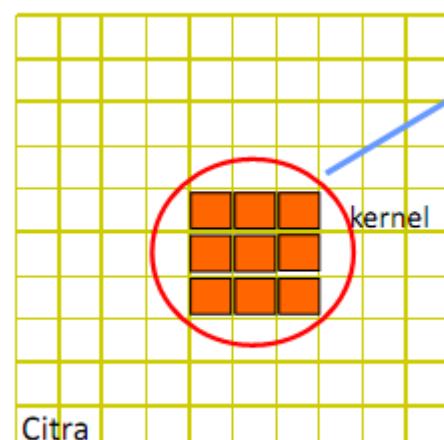


(b) Citra Lena sesudah konvolusi

Filtering

- Penapisan (filtering) termasuk pengolahan lokal, yaitu dalam transformasinya melibatkan:
 - nilai-nilai pixel tetangganya
 - nilai-nilai suatu sub-citra yang memiliki dimensi yang sama.
 - Sub-citra ini dikenal sebagai filter, mask, kernel, template, atau window. atau window.
 - Nilai dalam sub-citra tidak disebut sebagai nilai intensitas pixel, tetapi sebagai koefisien
- Filtering yang dibicarakan saat ini adalah penapisan spasial (spatial filtering)

Konsep Filtering



Ukuran kernel $m \times n$
dengan:

$$m = 2a + 1$$

$$n = 2b + 1$$

a dan b adalah integer non-negatif

Secara umum dikatakan ukuran kernel selalu ganjil/gasal

Contoh: 3x3, 5x5, 7x7, 3x5, 3x7, dst.

Pada umumnya $m = n$

Konsep Filtering

- Beberapa penapis yang sering dipakai :
 - Lowpass spatial filtering
 - Median
 - Highpass spatial filtering
 - Laplacian
 - Directional
 - Roberts
 - Sobel
 - Gaussian

Low Pass Spatial Filter

- Untuk menghaluskan citra
- Didasarkan pada perata-rataan nilai pixel dengan tetangga
- Filter lolos-bawah (*low-pass filter*) juga disebut penapis perataan (*averaging filter*)
- Filter ini akan menghasilkan citra yang lebih lembut (*smooth*) sehingga terkesan kabur (blur); dan mengurangi kisaran aras abu-abu
- Bobot filter selalu positif yang totalnya bernilai 1

Low Pass Spatial Filter

- Contoh beberapa filter

$$\frac{1}{5} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\frac{1}{32} \begin{bmatrix} 1 & 3 & 1 \\ 3 & 16 & 3 \\ 1 & 3 & 1 \end{bmatrix}$$

$$\frac{1}{8} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\frac{1}{N^2} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}_{N \times N}$$

Low Pass Spatial Filter

- Contoh beberapa filter

$$\frac{1}{9} \times \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

Kernel penapis perata 3×3

Penapis dengan semua koefisien sama
disebut penapis kotak (*box filter*)

$$\frac{1}{16} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$

Kernel penapis perata berbobot
(*weighted averaging*) 3×3

Matriks Low Pass Spatial Filter

136	127	142	135	143	152	172	173	189	184
143	138	125	145	138	145	137	171	175	185
157	145	139	133	146	139	136	147	157	164
165	154	157	145	143	147	147	133	146	158
174	158	167	162	158	147	146	140	142	149
185	169	167	159	180	160	146	143	137	145
190	180	161	162	172	178	149	141	141	139
192	166	177	158	172	156	161	144	137	141
192	186	170	172	146	159	147	137	140	140
205	184	184	152	160	142	157	142	140	141

135	135	135	138	143	150	160	172	180	187
142	139	136	138	141	145	152	161	171	182
151	147	142	141	142	142	144	149	159	172
161	157	151	150	146	145	142	143	148	161
169	166	159	159	155	152	145	142	143	151
178	172	165	165	164	159	150	142	141	146
183	176	166	167	166	163	153	144	140	142
186	179	170	165	163	160	152	144	140	142
190	184	172	165	157	155	149	145	140	139
194	187	176	165	157	152	148	144	141	139

Matriks citra semula

Sampel 10x10 kiri atas

$$\begin{aligned}142 &= (138 + 125 + 145 \\&\quad + 145 + 139 + 133 \\&\quad + 154 + 157 + 145) / 9 \\&= 1281 / 9 = 142.33\end{aligned}$$

Matriks citra hasil penapisan

Sampel 10x10 kiri atas

Contoh Hasil Low Pass Filtering



$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Median Filter

- Digunakan untuk menghilangkan noise
- Menggunakan nilai tengah dari pixel-pixel yang tertutup filter



Matriks hasil median

136	127	142	135	143	152	172	173	189	184
143	138	125	145	138	145	137	171	175	185
157	145	139	133	146	139	136	147	157	164
165	154	157	145	143	147	147	133	146	158
174	158	167	162	158	147	146	140	142	149
185	169	167	159	180	160	146	143	137	145
190	180	161	162	172	178	149	141	141	139
192	166	177	158	172	156	161	144	137	141
192	186	170	172	146	159	147	137	140	140
205	184	184	152	160	142	157	142	140	141

Matriks citra semula

Sampel 10x10 kiri atas

125 133 138 139 145 145 145 154 157

median

136	127	142	135	143	152	172	173	189	184
143	139	138	139	143	143	147	171	173	185
157	145	145	143	145	143	145	147	158	175
165	157	154	146	146	146	146	147	158	
174	167	159	159	158	147	146	143	143	146
185	169	162	162	162	158	146	142	141	145
190	177	166	167	162	161	149	143	141	141
192	180	170	170	162	159	149	141	140	141
192	184	172	170	158	157	147	142	140	140
205	186	175	170	158	154	145	145	141	140

Matriks citra hasil penapisan

Sampel 10x10 kiri atas

Highpass Spatial Filter

- Disebut sebagai ***sharpening mask***, karena mempercepat pergantian batas gelap-terang
- Filter memiliki nilai positif di tengah , negatif dipinggir dan **total bobot harus 0**
 - $\sum \text{koefisien} = 0 \rightarrow \text{komponen freq. rendah turun}$
 - $\sum \text{koefisien} = 1 \rightarrow \text{komponen freq. rendah tetap}$

-1	-1	-1
-1	8	-1
-1	-1	-1

$$\sum = 0$$

0	-1	0
-1	5	-1
0	-1	0

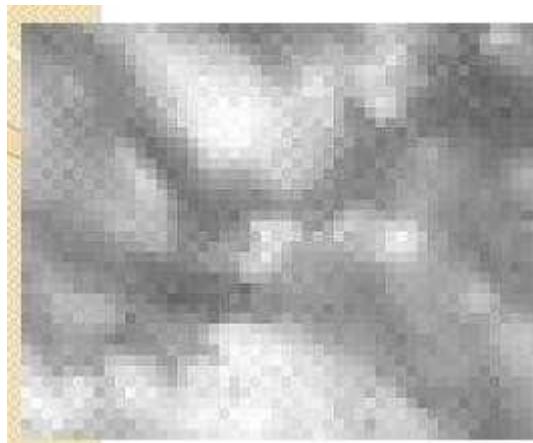
$$\sum = 1$$

Highpass Spatial Filter

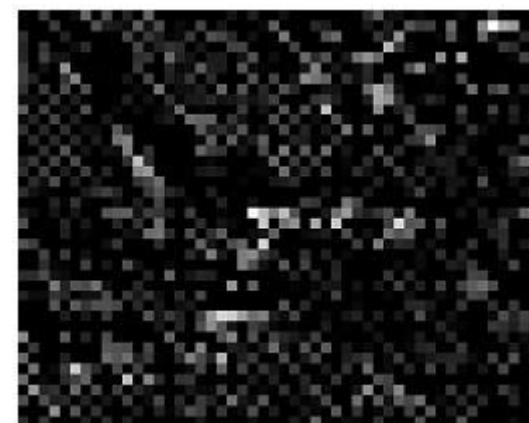
- **Hasil highpass filtering** adalah selisih antara citra asli dengan citra yang lain

$$g(m,n) = f(m,n) - \text{lowpass}(f(m,n))$$

- **Contoh hasil highpass filtering:**



-1	-1	-1
-1	8	-1
-1	-1	-1



Matriks hasil highpass

136	127	142	135	143	152	172	173	189	184
143	138	125	145	138	145	137	171	175	185
157	145	139	133	146	139	136	147	157	164
165	154	157	145	143	147	147	133	146	158
174	158	167	162	158	147	146	140	142	149
185	169	167	159	180	160	146	143	137	145
190	180	161	162	172	178	149	141	141	139
192	166	177	158	172	156	161	144	137	141
192	186	170	172	146	159	147	137	140	140
205	184	184	152	160	142	157	142	140	141

2	0	62	0	0	14	101	6	78	0
5	0	0	59	0	0	0	82	30	26
46	0	0	0	33	0	0	0	0	0
36	0	53	0	0	14	41	0	0	0
37	0	65	20	21	0	5	0	0	0
60	0	18	0	142	4	0	2	0	0
61	33	0	0	51	128	0	0	1	0
48	0	61	0	73	0	77	0	0	0
14	18	0	57	0	31	0	0	0	3
94	0	66	0	22	0	81	0	0	11

Matriks citra semula

Sampel 10x10 kiri atas

$$\begin{aligned}0 &= (-138 - 125 - 145 \\&\quad -145 + 8 \times 139 - 133 \\&\quad -154 - 157 - 145) \\&= -1142 + 1112 = -30\end{aligned}$$

Matriks citra hasil penapisan

Sampel 10x10 kiri atas



Pemampatan Citra (Image Compression)

KOMPRESI CITRA

I Mengapa perlu kompresi dan reduksi data

- || Data citra umumnya berukuran besar
- || Tidak praktis dalam penyimpanan, proses dan transmisi
- || Perlu reduksi atau pemampatan data dengan mengurangi *redundancy* atau duplikasi data

I DATA REDUNDANCY

- || adalah bagian data yang tidak mengandung informasi terkait atau merupakan pengulangan dari informasi yang sudah dinyatakan sebelumnya atau sudah diketahui

CONTOH APLIKASI

- **Aplikasi yang membutuhkan image compression:**
(dimana perkembangannya ditentukan oleh efisiensi pada manipulasi data, penyimpanan, dan transmisi citra biner / monokrom / berwarna)
 - Televideo-conferencing
 - Remote sensing
 - Telemedical / Medical imaging
 - Facsimile transmission

KOMPRESI CITRA

I Kompresi Citra adalah

■ Aplikasi kompresi data yang dilakukan terhadap citra digital dengan tujuan untuk **mengurangi redundansi** dari data-data yang terdapat dalam citra sehingga dapat **disimpan** atau **ditransmisikan secara efisien**.

I Tujuan Kompresi Citra adalah :

■ **meminimalkan kebutuhan memori** untuk merepresentasikan citra digital dengan mengurangi duplikasi data di dalam citra **sehingga memori yang dibutuhkan menjadi lebih sedikit daripada representasi citra semula.**

KOMPRESI CITRA

■ **Manfaat Kompresi Citra adalah :**

- Waktu pengiriman data pada saluran komunikasi data lebih singkat
 - Contoh : pengiriman gambar dari fax, video conferencing, handphone, download dari internet, pengiriman data medis, pengiriman internet, pengiriman dari satelit, dsb
- Membutuhkan ruang memori dalam storage lebih sedikit dibandingkan dengan citra yang tidak dimampatkan

KOMPRESI CITRA

- **Proses kompresi** merupakan **proses mereduksi** ukuran suatu data untuk menghasilkan **representasi digital yang padat** atau memampatkan namun tetap dapat mewakili kuantitas informasi yang terkandung pada data tersebut.
- **Pada citra, video atau audio**, kompresi mengarah pada **minimisasi jumlah bit rate** untuk representasi digital.
 - Bit Rate disebut juga dengan nama data rate
 - Bit rate menentukan jumlah data yang ditampilkan saat video dimainkan, yang dinyatakan dalam satuan bps (bit per second).
 - Data rate berkaitan erat dengan pemakaian dan pemilihan codec (metode kompresi video).

KOMPRESI CITRA

- || **Semakin besar ukuran citra, semakin besar memori** yang dibutuhkan. Namun kebanyakan citra mengandung **duplicasi data**, yaitu :
 - || suatu pixel memiliki intensitas yang sama dengan dengan pixel tetangganya, sehingga penyimpanan setiap pixel memboroskan tempat
 - || citra banyak mengandung bagian (region) yang sama, sehingga bagian yang sama ini tidak perlu dikodekan berulangkali karena mubazir atau redundant

PEMAMPATAN VS PENGKODEAN

I PEMAMPATAN

- Citra dikodekan
- Representasi memori menjadi lebih kecil
- Menerapkan proses Compress dan Decompress
- Aplikasi : Pengiriman dan penyimpanan data

I PENGKODEAN

- Citra dikodekan Representasi memori belum tentu lebih kecil
- Menerapkan proses encode dan Decode

KRITERIA PEMAMPATAN (KOMPRESI)

- | Waktu pemampatan
- | Kebutuhan memory
- | Kualitas pemampatan (fidelity)

$$PSNR = 20 \times \log_{10} \left(\frac{b}{rms} \right)$$

$$rms = \sqrt{\frac{1}{\text{Lebar} \times \text{Tinggi}} \sum_{i=1}^N \sum_{j=1}^M (f_{ij} - f'_{ij})^2}$$

- | Format Keluaran

JENIS KOMPRESI

|| Pendekatan Statistik

- || Melihat frekuensi kemunculan derajat keabuan pixel

|| Pendekatan Ruang

- || Melihat hubungan antar pixel yang mempunyai derajat keabuan yang sama pada wilayah dalam citra

|| Pendekatan Kuantisasi

- || Mengurangi jumlah derajat keabuan yang tersedia

|| Pendekatan Fraktal

- || Kemiripan bagian citra dieksplorasi dengan matriks transformasi

KLASIFIKASI METODE KOMPRESI

I Metode Lossless

- menghasilkan citra yang sama dengan citra semula
- Tidak ada informasi yang hilang
- Biasa digunakan pada citra medis
- Nisbah/ratio kompresi sangat rendah

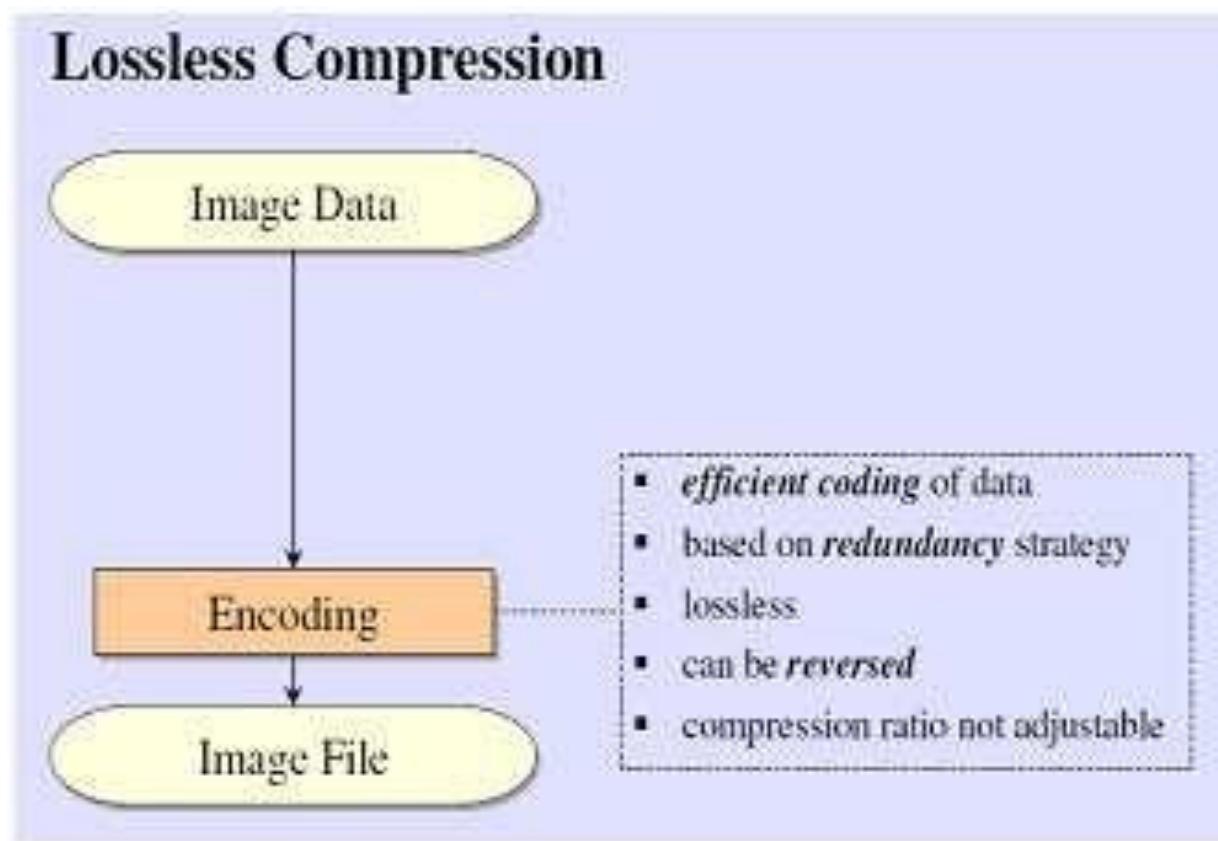
$$\text{Nisbah} = 100\% - \left(\frac{\text{ukuran citra hasil pempatatan}}{\text{ukuran citra semula}} \times 100\% \right)$$

II Metode Lossles :

- Run Length Encoding, Entropy Encoding (Huffman, Aritmatik), dan Adaptive Dictionary Based (LZW)

KLASIFIKASI METODE KOMPRESI

I Metode Lossless



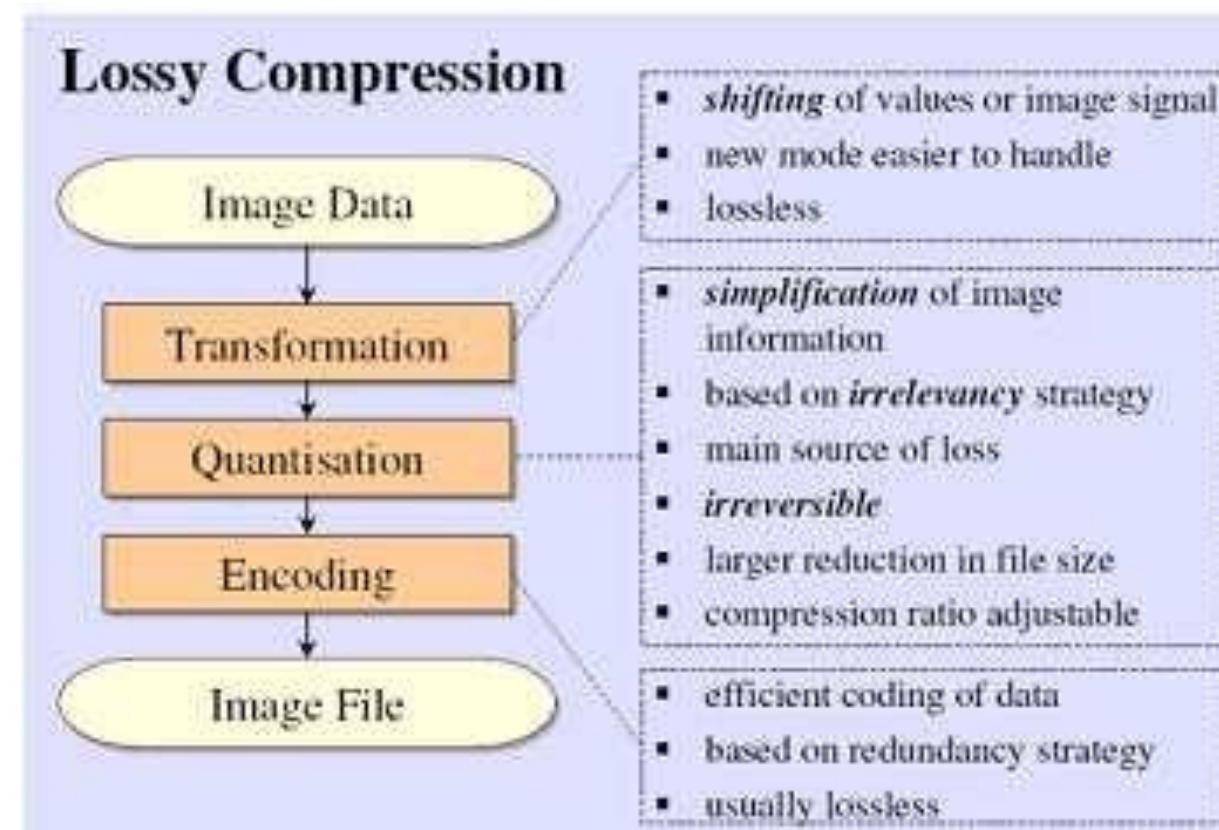
KLASIFIKASI METODE KOMPRESI

■ Metode lossy

- Menghasilkan citra yang hampir sama dengan citra semula
- Ada informasi yang hilang akibat kompresi tapi masih bisa ditolerir oleh persepsi mata
- Teknik ini mengubah detail dan warna pada file citra menjadi lebih sederhana tanpa terlihat perbedaan yang mencolok dalam pandangan manusia, sehingga ukurannya menjadi lebih kecil.
- Biasanya digunakan pada citra foto atau image lain yang tidak terlalu memerlukan detail citra, dimana kehilangan bit rate foto tidak berpengaruh pada citra.
- Nisbah/ratio pemampatan tinggi
- Contoh, JPEG dan Fraktal

KLASIFIKASI METODE KOMPRESI

I Metode lossy



METODE KOMPRESI SHANNON-FANO

1. Buatlah daftar peluang atau frekuensi kehadiran setiap simbol dari data (pesan) yang akan dikodekan.
2. Urutkanlah daftar tersebut menurut frekuensi kehadiran simbol secara menurut (Descending)
3. Bagilah daftar tersebut menjadi dua bagian dengan pembagian didasari pada jumlah total frekuensi suatu bagian (bagian atas) sedekat mungkin dengan jumlah total frekuensi dengan bagian yang lain (bagian bawah).
4. Daftar bagian atas diberi nilai 0 dan 1 untuk bagian bawah.
5. Lakukan proses secara rekursif (berulang) untuk langkah 3 dan 4.

METODE KOMPRESI SHANNON-FANO

I Soal

I Suatu data sebagai berikut:

BCEEDDBBAAAABEEEEDDDCCCAAACCDAAAAAABB
BAAA

I Jawab

<u>Simbol</u>	<u>Frekuensi</u>
A	15
B	7
C	6
D	6
E	5

METODE KOMPRESI SHANNON-FANO

<u>Simbol</u>	<u>Frekuensi</u>	
A	15	0
B	7	0
C	6	1
D	6	1
E	5	1

METODE KOMPRESI SHANNON-FANO

Simbol Frekuensi

A	15	0	0
B	7	0	1
C	6	1	0
D	6	1	1
E	5	1	1

METODE KOMPRESI SHANNON-FANO

<u>Simbol</u>	<u>Frekuensi</u>				
A	15	0	0		
B	7	0	1		
C	6	1	0		
D	6	1	1	0	
E	5	1	1	1	

METODE KOMPRESI SHANNON-FANO

<u>Simbol</u>	<u>Frekuensi</u>	<u>Kode</u>	<u>Bit</u>
A	15	00
B	7	01
C	6	10
D	6	110
E	5	111

METODE KOMPRESI SHANNON-FANO

<u>Simbol</u>	<u>Frekuensi</u>	<u>Kode</u>	<u>Bit</u>	<u>Tot Bit</u>
A	15	00	2	...
B	7	01	2	...
C	6	10	2	...
D	6	110	3	...
E	5	111	3	...

METODE KOMPRESI SHANNON-FANO

<u>Simbol</u>	<u>Frekuensi</u>	<u>Kode</u>	<u>Bit</u>	<u>Tot Bit</u>
A	15	00	2	30
B	7	01	2	14
C	6	10	2	12
D	6	110	3	18
E	5	111	3	15

..... byte

..... bit

METODE KOMPRESI SHANNON-FANO

<u>Simbol</u>	<u>Frekuensi</u>	<u>Kode</u>	<u>Bit</u>	<u>Tot Bit</u>
A	15	00	2	30
B	7	01	2	14
C	6	10	2	12
D	6	110	3	18
E	5	111	3	15

39 byte

89 bit

$$89/8=11 \text{ byte}$$

METODE KOMPRESI HUFFMAN

1. Urutkan nilai keabuan berdasarkan frekuensi kemunculannya
2. Gabung dua pohon yang frekuensi kemunculannya paling kecil
3. Ulangi 2 langkah diatas sampai tersisa satu pohon biner
4. Beri label 0 untuk pohon sisi kiri dan 1 untuk pohon sisi kanan
5. Telusuri barisan label sisi dari akar ke daun yang menyatakan kode Huffman

METODE KOMPRESI HUFFMAN (Contoh)

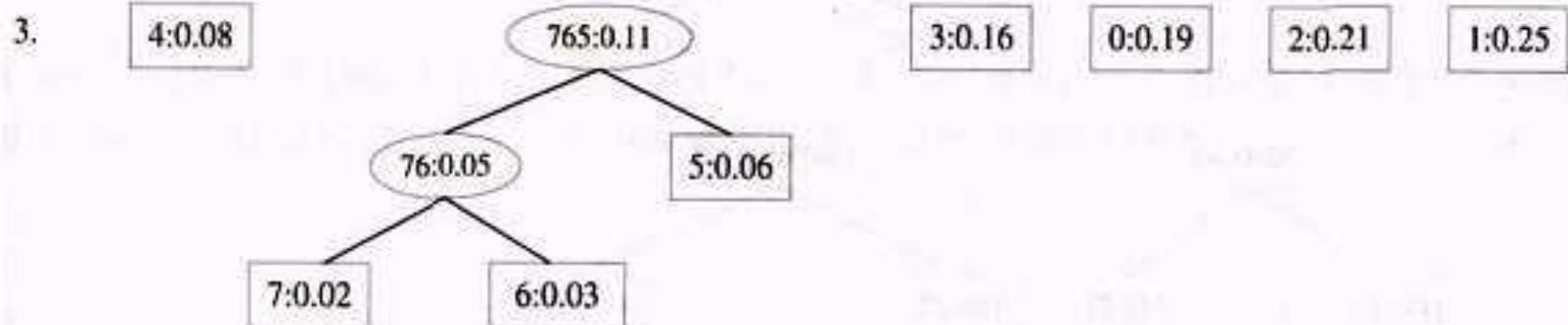
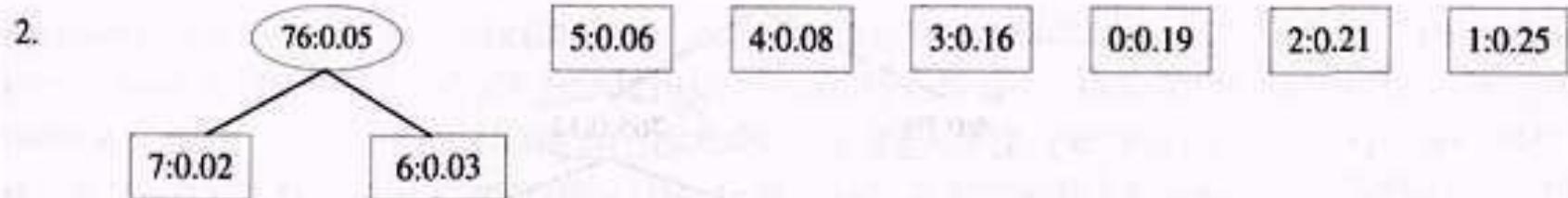
I Contoh, citra 64x64 dengan 8 derajat keabuan (k)

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

METODE KOMPRESI HUFFMAN (Contoh)

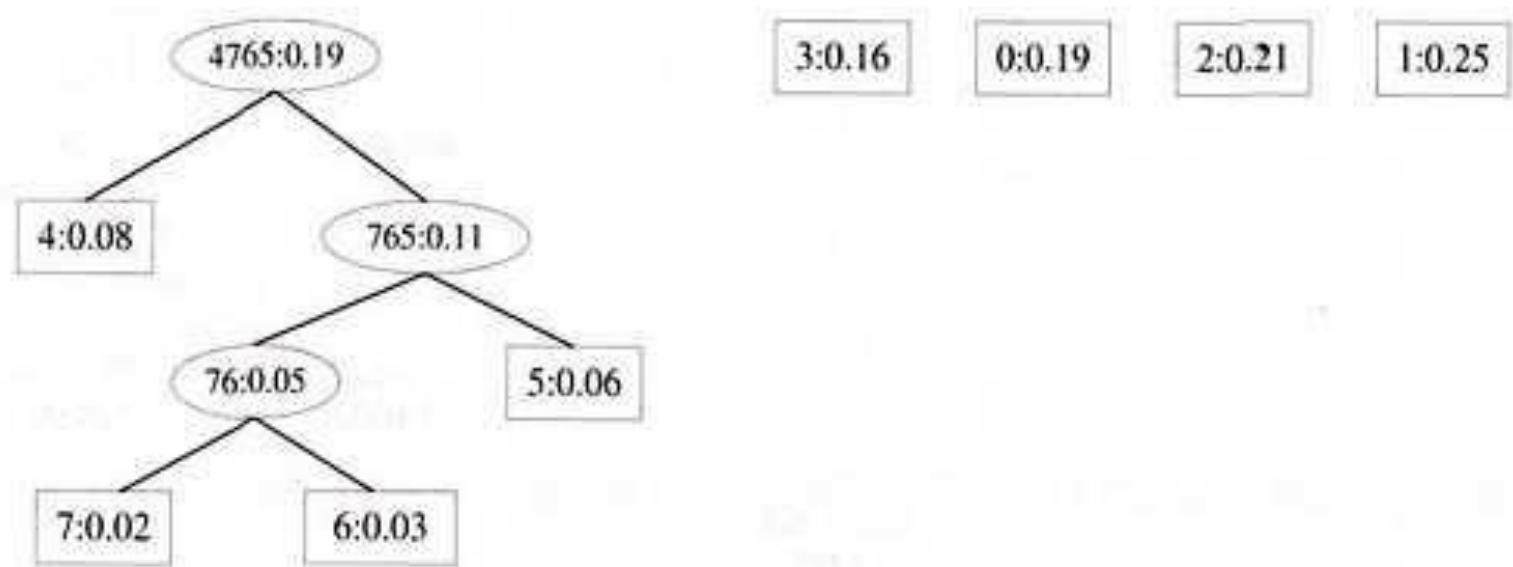
1.

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



METODE KOMPRESI HUFFMAN (Contoh)

4.



METODE KOMPRESI HUFFMAN (Contoh)

5.

0:0.19

2:0.21

1:0.25

34765:0.35

3:0.16

4765:0.19

4:0.08

765:0.11

76:0.05

5:0.06

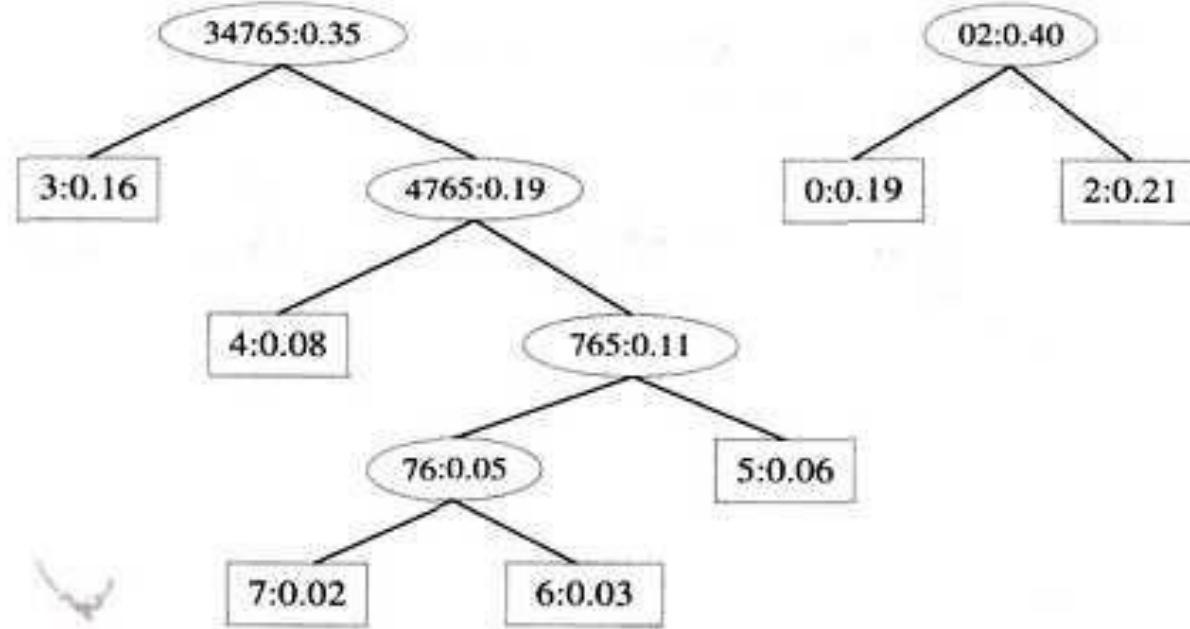
7:0.02

6:0.03

METODE KOMPRESI HUFFMAN (Contoh)

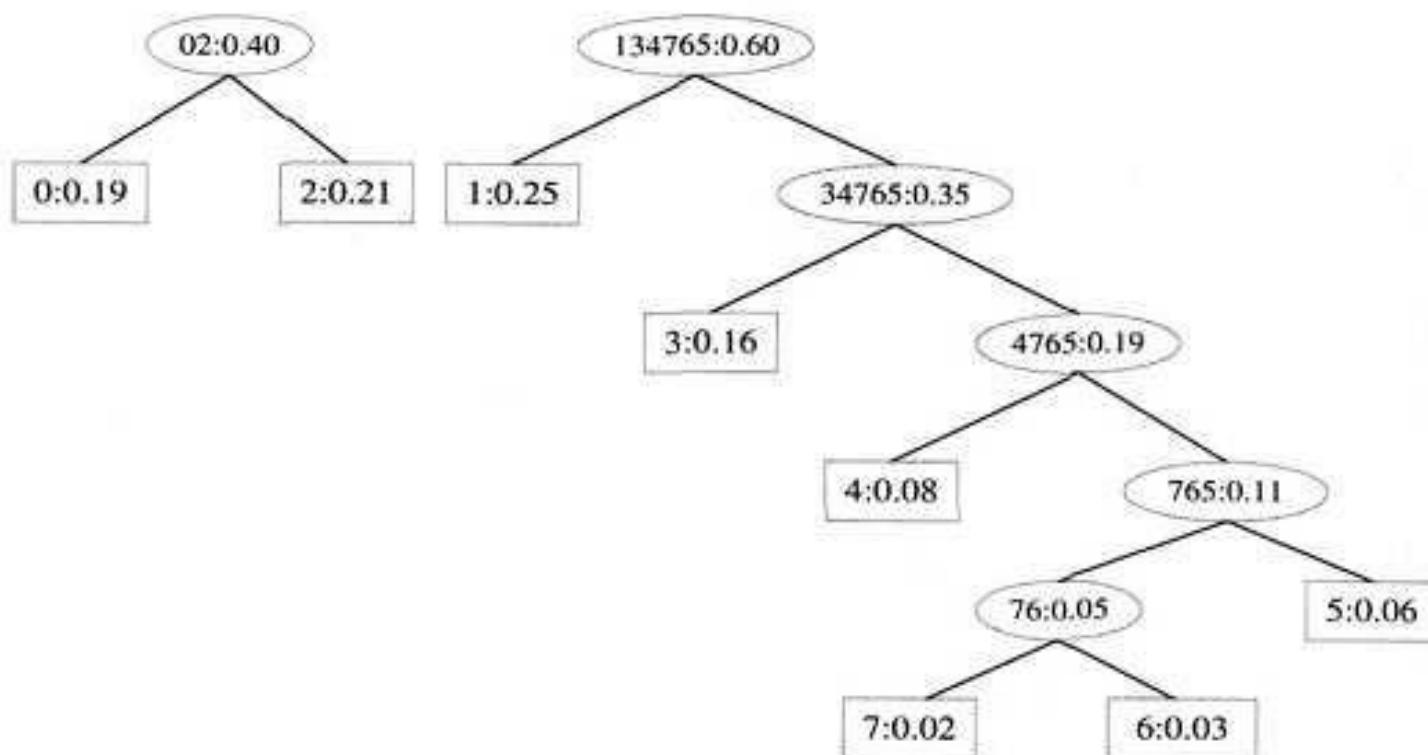
6.

1:0.25

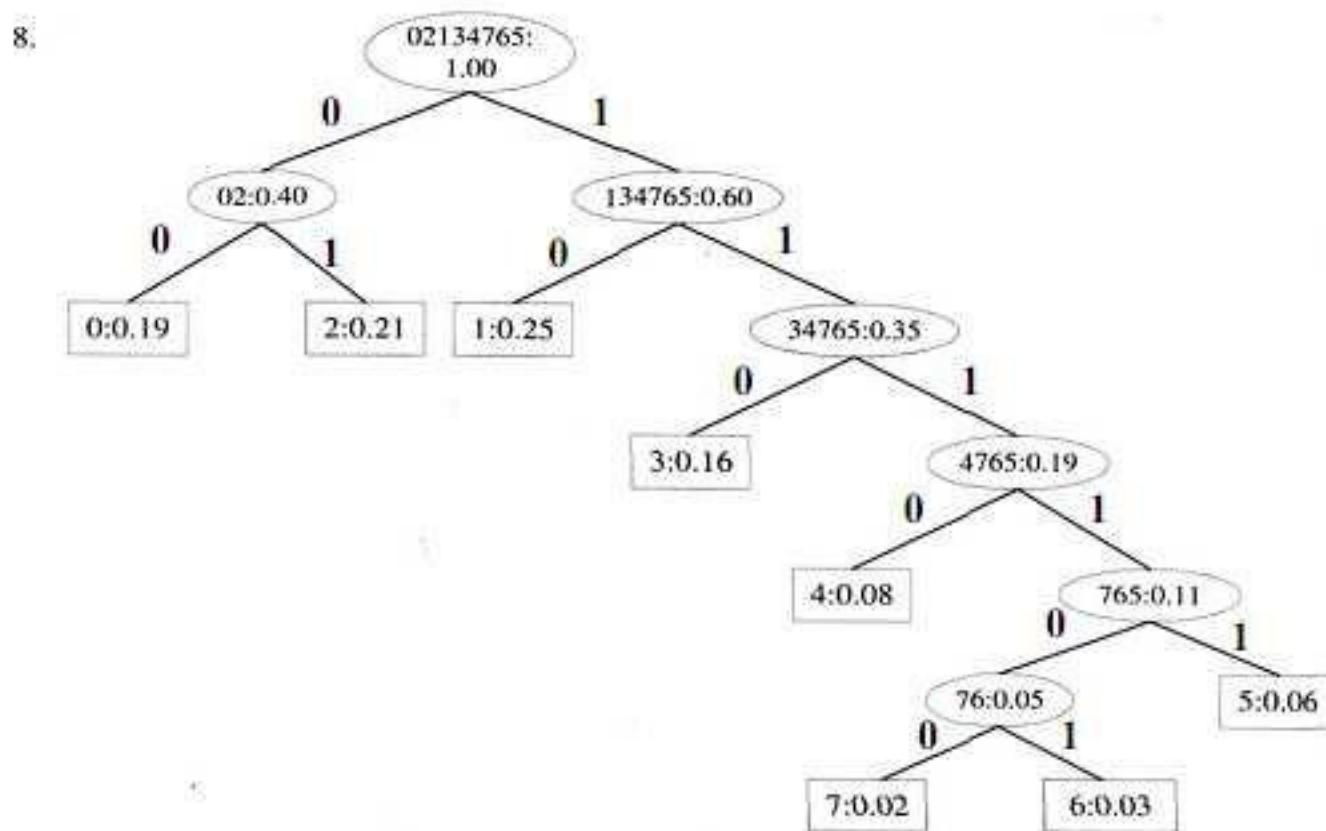


METODE KOMPRESI HUFFMAN (Contoh)

7.



METODE KOMPRESI HUFFMAN (Contoh)



METODE KOMPRESI HUFFMAN (Contoh)

- | Contoh, citra 64x64 dengan 8 derajat keabuan (k)
- | Kode untuk setiap derajat keabuan

0 = 00

2 = 01

4 = 1110

6 = 111101

1 = 10

3 = 110

5 = 11111

7 = 111100

- | Ukuran citra sebelum dimampatkan (8 derajat keabuan = 3 bit) adalah 4096×3 bit = 12288 bit
- | Ukuran citra setelah pemampatan

$$\begin{aligned}(790 \times 2 \text{ bit}) + (1023 \times 2 \text{ bit}) + (850 \times 2 \text{ bit}) + \\(656 \times 3 \text{ bit}) + (329 \times 4 \text{ bit}) + (245 \times 5 \text{ bit}) + \\(122 \times 6 \text{ bit}) + (81 \times 6 \text{ bit}) = 11053 \text{ bit}\end{aligned}$$

$$\text{Nisbah pemampatan} = (100\% - \frac{11053}{12288} \times 100\%) = 10\%$$

METODE KOMPRESI RLE

■ **Run Length Encoding**

- RLE merupakan metode kompresi yang banyak didukung oleh format file gambar seperti: TIFF, BMP, PCX.
- RLE bekerja dengan mengurangi ukuran fisik dengan adanya pengulangan string dari deretan karakter / byte data.
- Cocok untuk pemampatan citra yang memiliki kelompok pixel berderajat keabuan yang sama

METODE KOMPRESI RLE

■ Run Length Encoding

- Contoh citra 10x10 dengan 8 derajat keabuan

0	0	0	0	0	2	2	2	2	(0, 5), (2, 5)
0	0	0	1	1	1	2	2	2	(0, 3), (1, 4), (2, 3)
1	1	1	1	1	1	1	1	1	(1, 10)
4	4	4	4	3	3	3	2	2	(4, 4), (3, 4), (2, 2)
3	3	3	5	5	7	7	7	6	(3, 3), (5, 2), (7, 4), (6, 1)
2	2	6	0	0	0	0	1	1	(2, 2), (6, 1), (0, 4), (1, 2), (0, 1)
3	3	4	4	3	2	2	1	1	(3, 2), (4, 2), (3, 1), (2, 2), (1, 2)
0	0	0	0	0	0	0	1	1	(0, 8), (1, 2)
1	1	1	1	0	0	0	2	2	(1, 4), (0, 3), (2, 3)
3	3	3	2	2	2	1	1	1	(3, 3), (2, 3), (1, 4)

Pasangan derajat keabuan (p)
dan jumlah pixel (q)

METODE KOMPRESI RLE

- | Ukuran citra sebelum dimampatkan (1 derajat keabuan = 3 bit) adalah $100 \times 3 \text{ bit} = 300 \text{ bit}$
- | Ukuran citra setelah pemampatan (run length = 4) adalah
$$(31 \times 3) + (31 \times 4) \text{ bit} = 217 \text{ bit}$$

Nisbah pemampatan = $(100\% - \frac{217}{300} \times 100\%) = 27.67\%$, yang artinya 27.67%

METODE KOMPRESI KUANTISASI

- | Buat histogram citra yang akan dikompres P jumlah pixel
- | Identifikasi n buah kelompok di histogram sedemikian sehingga setiap kelompok mempunyai kira-kira P/n pixel
- | Nyatakan setiap kelompok dengan derajat keabuan 0 sampai $n-1$.
- | Setiap kelompok dikodekan kembali dengan nilai derajat keabuan yang baru

METODE KOMPRESI KUANTISASI

- Contoh, Citra 5×13

2	9	6	4	8	2	6	3	8	5	9	3	7
3	8	5	4	7	6	3	8	2	8	4	7	3
3	8	4	7	4	9	2	3	8	2	7	4	9
3	9	4	7	2	7	6	2	1	6	5	3	0
2	0	4	3	8	9	5	4	7	1	2	8	3

- Akan dimampatkan dengan 4 derajat keabuan (0 - 3) atau dengan 2 bit

Histogram

0	**
1	**
2	*****
3	*****
4	*****
5	***
6	***
7	***
8	***
9	***

Kelompoknya

0	**	
13	**	0

20	*****	1

17	***	2

15	***	3

METODE KOMPRESI KUANTISASI

- | Setelah dimampatkan

0	3	2	1	3	0	2	1	3	2	3	1	2
1	3	2	1	2	2	1	3	0	3	1	2	1
1	3	1	2	1	3	0	1	3	0	2	1	3
1	3	1	2	0	2	2	0	0	2	2	1	0
0	0	1	1	3	3	2	1	2	0	0	3	0

- | Ukuran sebelum pemampatan (1 derajat keabuan = 4 bit) adalah 65×4 bit = 260 bit
- | Ukuran citra setelah pemampatan (1 derajat keabuan = 2 bit) adalah 65×2 bit = 130 bit

$$\text{Nisbah pemampatan} = \left(100\% - \frac{130}{260} \times 100\%\right) = 50\%$$

TEKNIK KOMPRESI GIF

- || **GIF (Graphic Interchange Format)** dibuat oleh Compuserve pada tahun 1987 untuk menyimpan berbagai file bitmap menjadi file lain yang mudah diubah dan ditransmisikan pada jaringan diubah dan ditransmisikan pada jaringan komputer .
- || **GIF merupakan :**
 - || format citra web yang tertua yang mendukung kedalaman warna sampai 8 bit (256 warna),
 - || menggunakan 4 langkah interlacing,
 - || mendukung transparency, dan
 - || mampu menyimpan banyak image dalam 1 file.

TEKNIK KOMPRESI PNG

- || **PNG (Portable Network Graphics)** digunakan di Internet dan merupakan format terbaru setelah GIF, bahkan menggantikan GIF untuk Internet image karena GIF terkena patent LZW yang dilakukan oleh Unisys.
- || Menggunakan teknik **loseless** dan mendukung:
 - || Kedalaman warna 48 bit,
 - || Tingkat ketelitian sampling: 1,2,4,8, dan 16 bit,
 - || Teknik pencocokan warna yang lebih canggih dan akurat



TEKNIK KOMPRESI JPG

|| **JPEG (*Joint Photographic Experts Group*)**

menggunakan teknik kompresi lossy sehingga sulit untuk proses pengeditan.

|| **JPEG :**

- || cocok untuk citra pemandangan (natural Generated image),
- || tidak cocok untuk citra yang mengandung banyak garis, ketajaman warna, dan computer generated image

TEKNIK KOMPRESI JPG

I JPEG 2000

- I Adalah pengembangan kompresi JPEG.
- I Didesain untuk internet, scanning, foto digital, remote sensing, medical imaging, perpustakaan digital dan ecommerce.
- I Dapat digunakan pada bit-rate rendah sehingga dapat digunakan untuk network image dan remote sensing.
- I Menggunakan **Lossy** dan **lossless** tergantung kebutuhan bandwidth.
- I **Lossless** digunakan untuk medical image.
- I Transmisi progresif dan akurasi & resolusi pixel tinggi.

TEKNIK KOMPRESI TIFF

I **TIFF (Tagged Image File Format)**

- Dikembangkan oleh Aldus Corporation, tahun 80-an
- Dalam perkembangannya didukung oleh Microsoft
- Mendukung adanya pengalokasian untuk informasi tambahan (tag) || fleksibel
- Dapat menyimpan berbagai tipe gambar : 1 bit, grayscale, 8 bit, 24 bit RGB, dll

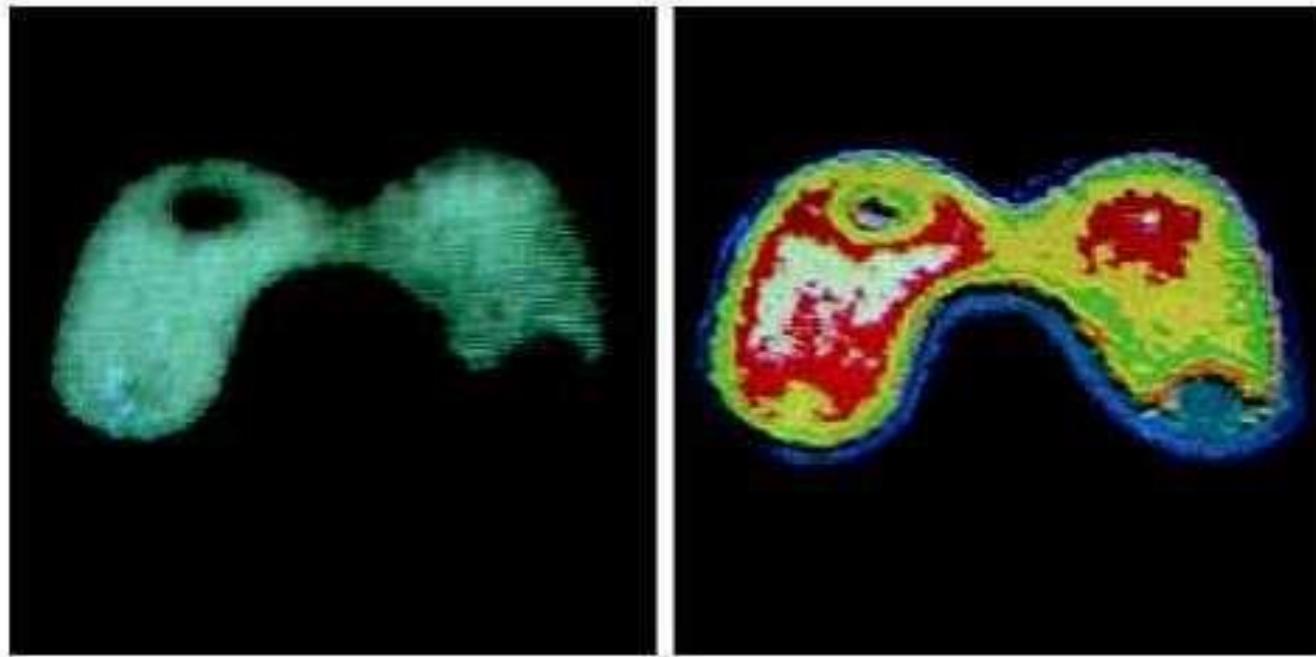


Citra Warna

PENGOLAHAN CITRA BERWARNA

- || Mengapa kita menggunakan citra berwarna :
 - || Dalam analisa citra otomatis, warna merupakan deskriptor yang sangat berguna → menyederhanakan proses identifikasi dan ekstraksi objek pada citra
 - || Mata manusia dapat membedakan ribuan warna dan intensitas
- || **Bagian dari pengolahan citra berwarna:**
 - || **Pengolahan full-color** → citra diperoleh dengan sensor full-color (kamera TV berwarna atau scanner berwarna, dll)
 - || **Pengolahan pseudo-color** → diperoleh dengan cara mengassign warna pada kisaran keabuan.

PENGOLAHAN CITRA BERWARNA



a b

FIGURE 6.20 (a) Monochrome image of the Pickle Thyroid Phantom. (b) Result of density slicing into eight colors. (Courtesy of Dr. J. L. Blankenship, Instrumentation and Controls Division, Oak Ridge National Laboratory.)

KONSEP WARNA

- Secara teknik, **warna** adalah **spektrum** tertentu yang terdapat di dalam suatu **cahaya** sempurna (berwarna putih).
- Identitas suatu warna ditentukan **panjang gelombang** cahaya tersebut.
- Panjang gelombang warna yang masih bisa ditangkap **mata** manusia berkisar antara 380-780 nanometer. Sebagai contoh warna **biru** memiliki panjang gelombang 460 nanometer.

SPECTRUM WARNA

- || Cahaya matahari yang dilewatkan pada prisma menghasilkan spektrum warna.
- || 'warna' objek yang diterima oleh penglihatan manusia ditentukan oleh cahaya dipantulkan oleh objek tersebut.

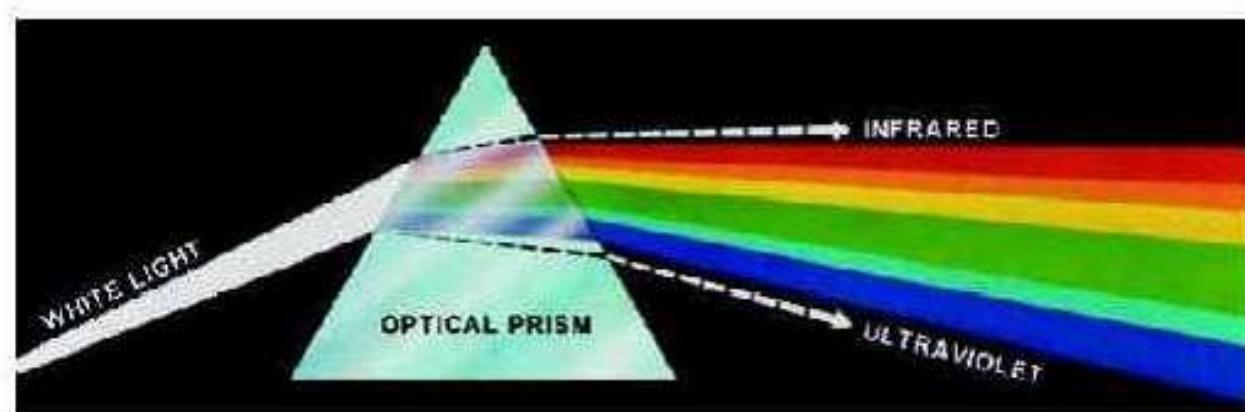
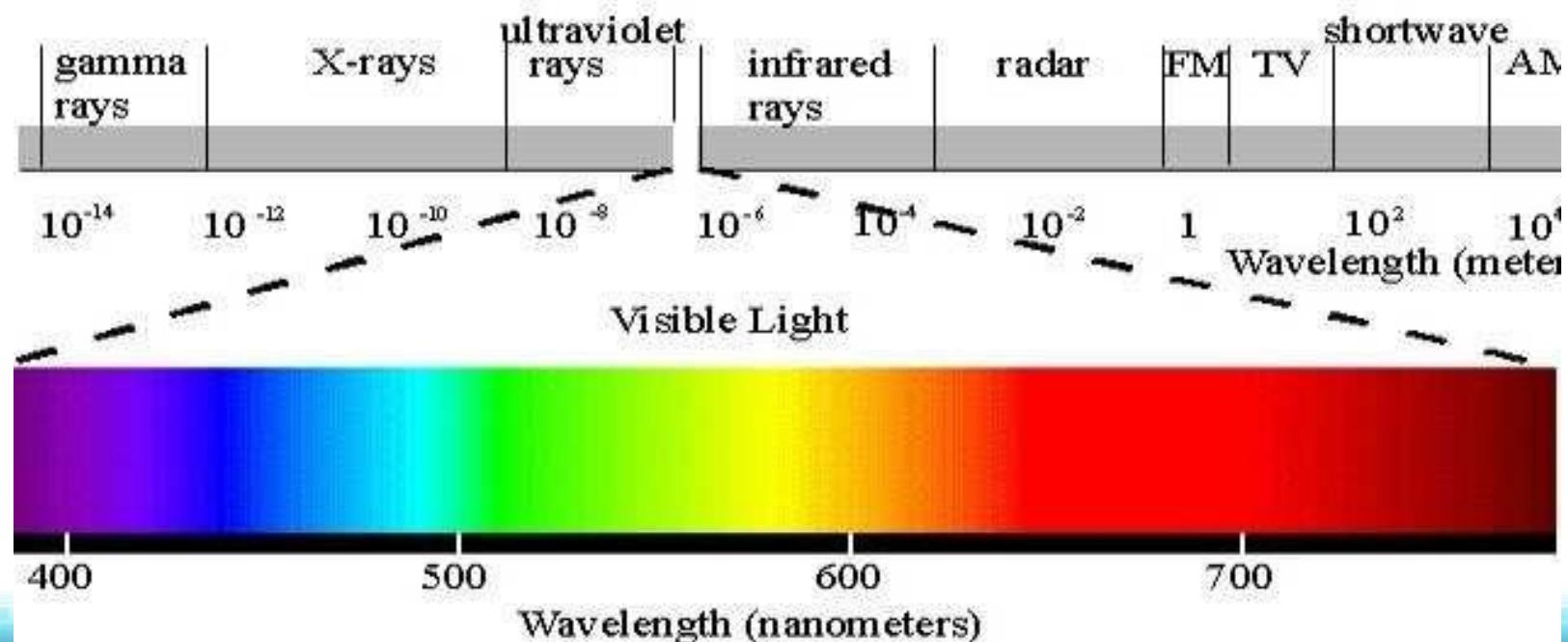


FIGURE 6.1 Color spectrum seen by passing white light through a prism. (Courtesy of the General Electric Co., Lamp Business Division.)

SPECTRUM WARNA

- Spektrum warna berada di antara infra-merah dan ultra-violet.
- Warna merah mempunyai range yang sangat lebar dan panjang gelombang yang paling tinggi (frekwensi paling rendah)



KONSEP WARNA

- Cahaya yang kita lihat adalah bukan cahaya dengan satu panjang gelombang melainkan kumpulan panjang gelombang tertentu.
- Warna terbentuk dari kumpulan gelombang dengan panjang gelombang yang berbeda-beda.
- Ini bisa diartikan bahwa **warna yang kita lihat** adalah **kombinasi** dari beberapa **elemen dasar warna**.
- Cara penyajian campuran elemen dasar untuk menghasilkan warna dinamakan dengan **Color Space** atau Ruang Warna.

KOMBINASI ELEMEN DASAR WARNA

- Sebuah warna bisa dihasilkan dengan kombinasi elemen dasar warna.
- Kombinasi elemen dasar warna bukan hitungan matematis, tetapi lebih pada pengalaman manusai dalam mengenali warna.
- Kadang-kadang untuk bisa menyajikan secara matematis, harus menggunakan lebih dari 3 komponen warna.

WARNA PRIMER VS WARNA SEKUNDER (CAHAYA)

I Warna primer:

- red (R), green (G), blue (B)
- perhatikan bahwa komponen RGB saja tidak bisa menghasilkan semua spektrum warna, kecuali jika panjang gelombangnya juga dapat bervariasi

I Warna sekunder:

- Magenta (R+B), cyan (G+B), yellow(R+G)

I Campuran 3 warna primer: putih

WARNA PRIMER VS WARNA SEKUNDER (PIGMEN)

I Warna primer:

- magenta, cyan, yellow
- Definisi: menyerap warna primer cahaya dan merefleksikan/mentransmisikan dua warna lainnya

I Warna sekunder:

- R,G,B

I Campuran 3 warna primer: hitam

BRIGHTNESS, HUE, SATURATION

- | Tiga karakteristik yang digunakan untuk membedakan satu warna dengan lainnya
- | **Brightness:** intensitas kromatik
- | **Hue:** panjang gelombang dominan dalam campuran gelombang cahaya (warna dominan yang diterima oleh observer). Kita menyebut suatu benda 'merah' atau 'biru' -> berarti kita menyebutkan hue-nya
- | **Saturasi:** kemurnian relatif (pada spektrum warna murni: merah, oranye, kuning, hijau, biru, dan violet tersaturasi penuh, sedangkan pink saturasinya lebih rendah)
- | **Hue + saturasi + kromatisitas**

FORMAT WARNA

- || Gambar (Digital) adalah sekumpulan titik yang disusun dalam bentuk matriks, dan nilainya menyatakan suatu derajat kecerahan (derajat keabuan/gray-scale).
 - || MISAL : Derajat keabuan 8 bit menyatakan 256 derajat kecerahan.
- || Pada gambar berwarna nilai setiap titiknya adalah nilai derajat keabuan pada setiap komponen warna RGB.
 - || Misal : Bila masing-masing komponen R, G dan B mempunyai 8 bit, maka satu titik dinyatakan dengan $(8+8+8)=24$ bit atau 2^{24} derajat keabuan



Macam-Macam Color Space

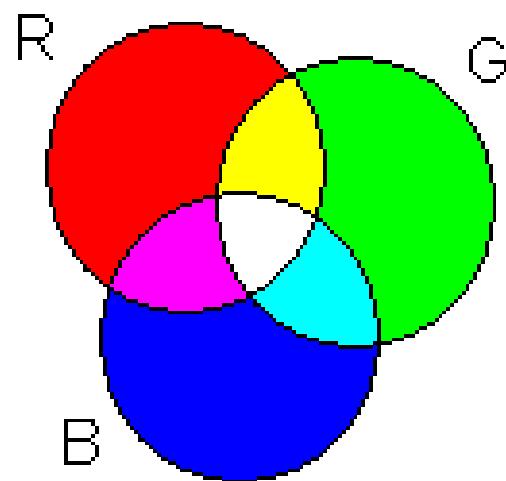
- RGB
- CMY (K)
- HSV
- CIE XYZ
- Lab
- Luv
- YCrCb

FORMAT RGB

- | Format RGB (Red, Green & Blue) adalah format dasar yang digunakan oleh banyak peralatan elektronik seperti monitor, LCD atau TV untuk menampilkan sebuah gambar.
- | Pada format RGB, suatu warna didefinisikan sebagai kombinasi (campuran) dari komponen warna R, G dan B.

FORMAT RGB

- I Pada format warna RGB 24 bit, maka nilai R, G dan B masing-masing 0-255



Warna	R	G	B
Hitam	0	0	0
Merah	255	0	0
Hijau	0	255	0
Biru	0	0	255
Kuning	255	255	0
Magenta	255	0	255
Cyan	0	255	255
Putih	255	255	255
Abu-Abu	127	127	127
Orange	255	110	0
Ungu	128	0	255
Coklat	128	25	0
Pink	255	190	220
Navy	0	0	120

FORMAT CMY(K)

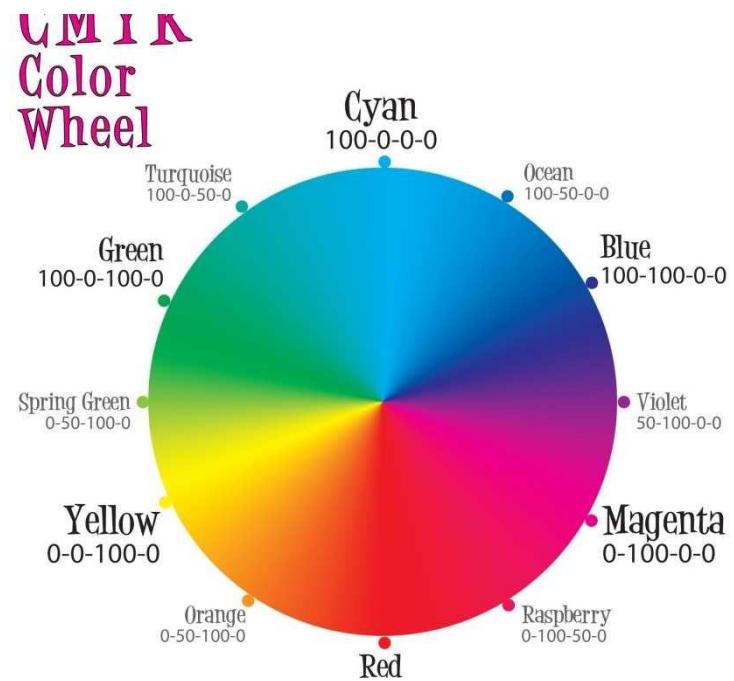
- CMY(K) menggunakan elemen dasar **Cyan**, **Magenta** dan **Yellow**. Untuk lebih lengkapnya ditambahkan elemen K (Chroma).
- CMY(K) adalah kombinasi warna yang digunakan dalam pencetakan (*printing*).
- Dikenal sebagai subtractive color.

FORMAT CMY(K)

- CMY(K) menggunakan elemen dasar **Cyan**, **Magenta** dan **Yellow**. Untuk lebih lengkapnya ditambahkan elemen K (Chroma).
- CMY(K) adalah kombinasi warna yang digunakan dalam pencetakan (*printing*).
- Dikenal sebagai subtractive color.

FORMAT CMY(K)

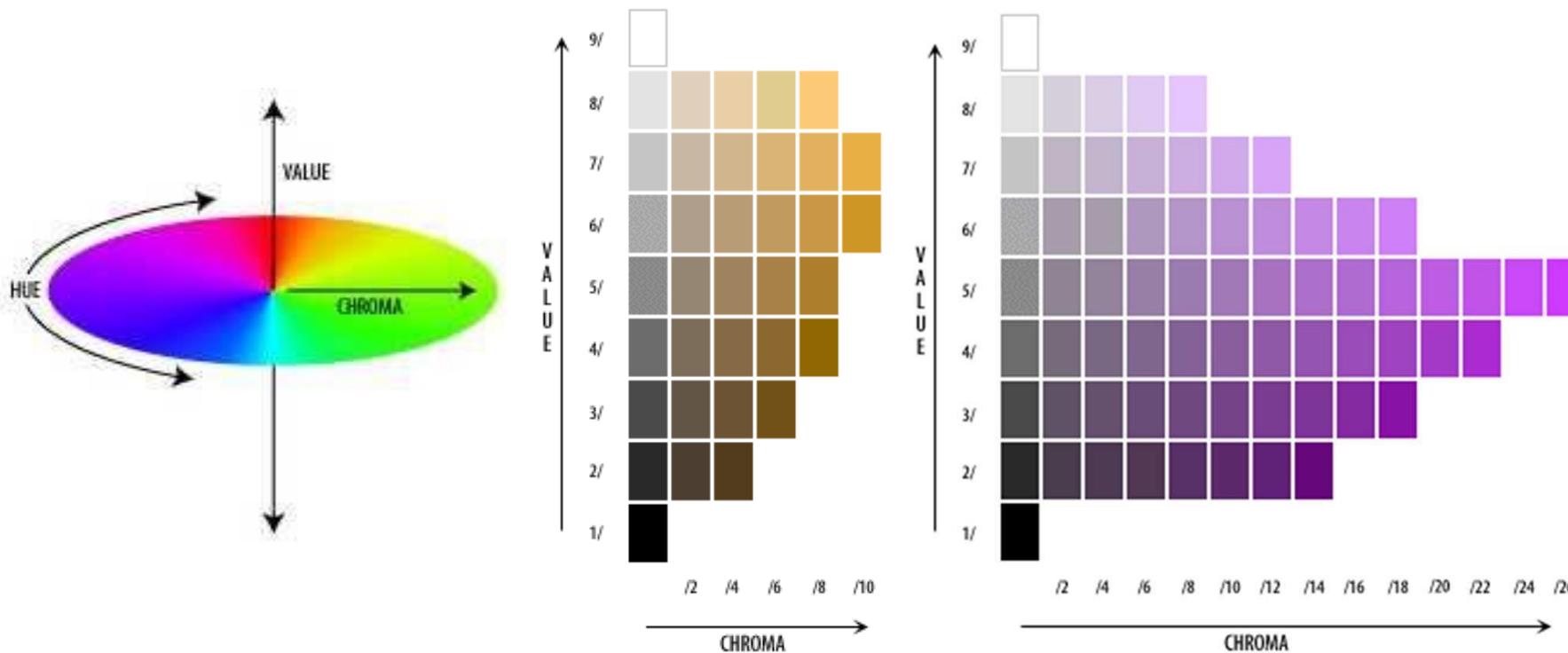
- Kombinasi setiap elemen menggunakan skala persentase.
- Warna Biru dan Merah mempunyai area yang lebih luas.
- Banyak digunakan dalam pencetakan.



FORMAT HSV

- HSV mempunyai elemen dasar **Hue, Saturation** dan **Value**:
 - **Hue** menyatakan keluarga warna (dalam satuan derajat)
 - **Saturation** menyatakan sensasi/intensitas warna
 - **Value** menyatakan derajat keabuan atau terang/gelap gambar.
- HSV dikembangkan menggunakan sistem koordinat polar.
- HSV banyak digunakan untuk fitur warna pada gambar.

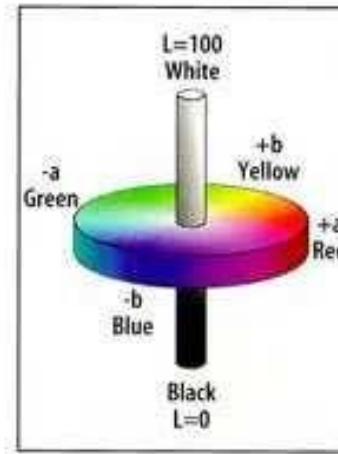
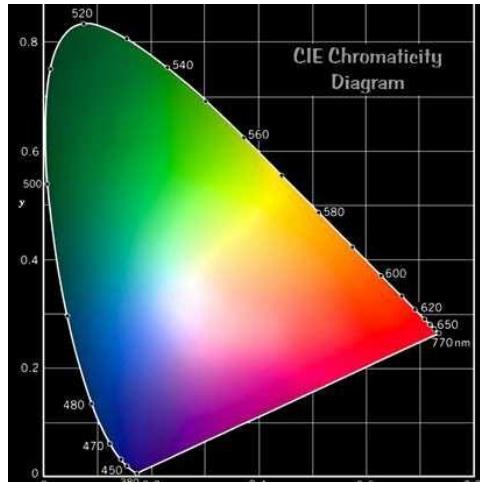
FORMAT HSV



- Nilai H → 0 s/d 360
- Nilai S → 0 s/d 1
- Nilai V → 0 s/d 255

FORMAT CIE

- CIE: International Commission on Illumination (Comission Internationale de l'Eclairage).
- Standar disusun berdasarkan persepsi manusia dan baik untuk percobaan perbandingan warna (1931).
- Standard observer: gabungan dari grup dengan anggota 15-20 orang.



Format Yuv & YCrCb

- Y (luminance) adalah $Y = 0.299R + 0.587G + 0.114B$
- **Chrominance** adalah perbedaan warna dan putih. Ini dapat dinyatakan dalam nilai U dan V.

$$U = B - Y; V = R - Y$$

- **YCrCb** adalah versi skala dari YUV dan digunakan dalam JPEG dan MPEG (semua komponen bernilai positif).

$$Cb = (B - Y) / 1.772 + 0.5;$$

$$Cr = (R - Y) / 1.402 + 0.5$$

Perbedaan Color Space



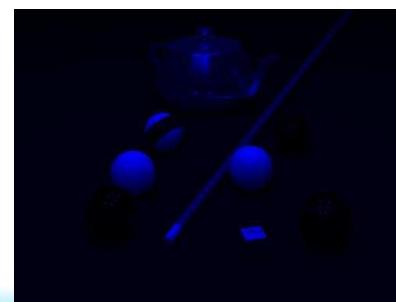
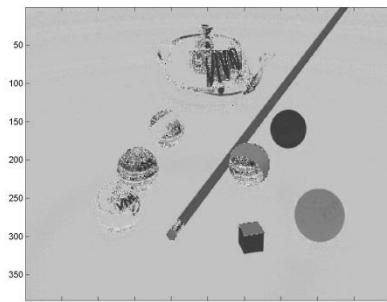
RGB



HSV

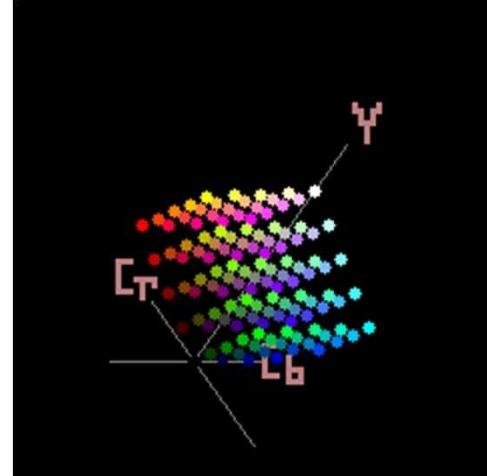


Luv



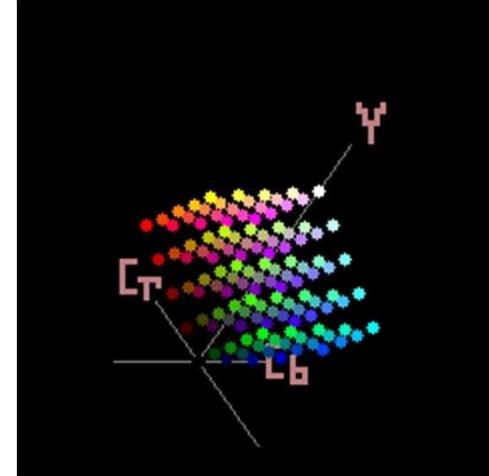
Color Space Conversion

- Konversi RGB ke CMYK
- Konversi RGB ke HSV
- Konversi RGB ke YCrCb



Konversi RGB → CMYK

- $R' = R/255$
- $G' = G/255$
- $B' = B/255$
- $K = 1 - \max(R', G', B')$
- $C = (1-R'-K)/(1-K)$
- $M = (1-G'-K)/(1-K)$
- $Y = (1-B'-K)/(1-K)$



Konversi RGB → HSV

- $R' = R/255$
- $G' = G/255$
- $B' = B/255$
- $Cmax = \max(R', G', B')$
- $Cmin = \min(R', G', B')$
- $D = Cmax - Cmin$

$$H = \begin{cases} 60 \cdot \left(\frac{G' - B'}{D} \bmod 6 \right), & \text{if } cmax = R' \\ 60 \cdot \left(\frac{B' - R'}{D} + 2 \right), & \text{if } cmax = G' \\ 60 \cdot \left(\frac{R' - G'}{D} + 4 \right), & \text{if } cmax = B' \end{cases}$$

$$S = \begin{cases} 0, & \text{jika } D = 0 \\ \frac{D}{Cmax}, & \text{jika } D \neq 0 \end{cases}$$

$$V = cmax$$

Konversi RGB → YCrCb

- $Y = 0.299R + 0.587G + 0.114B$
- $U = B - Y;$
- $V = R - Y$
- $Cb = (B - Y) / 1.772 + 0.5$
- $Cr = (R - Y) / 1.402 + 0.5$

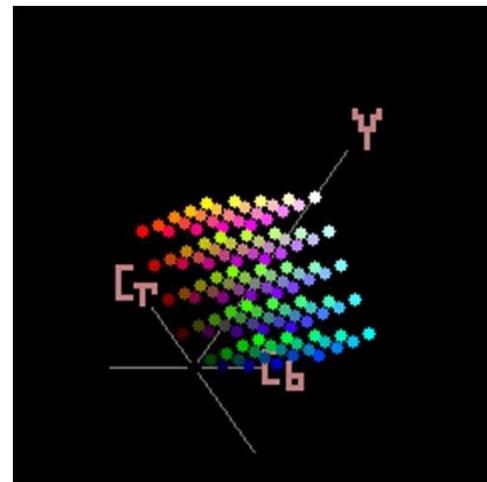


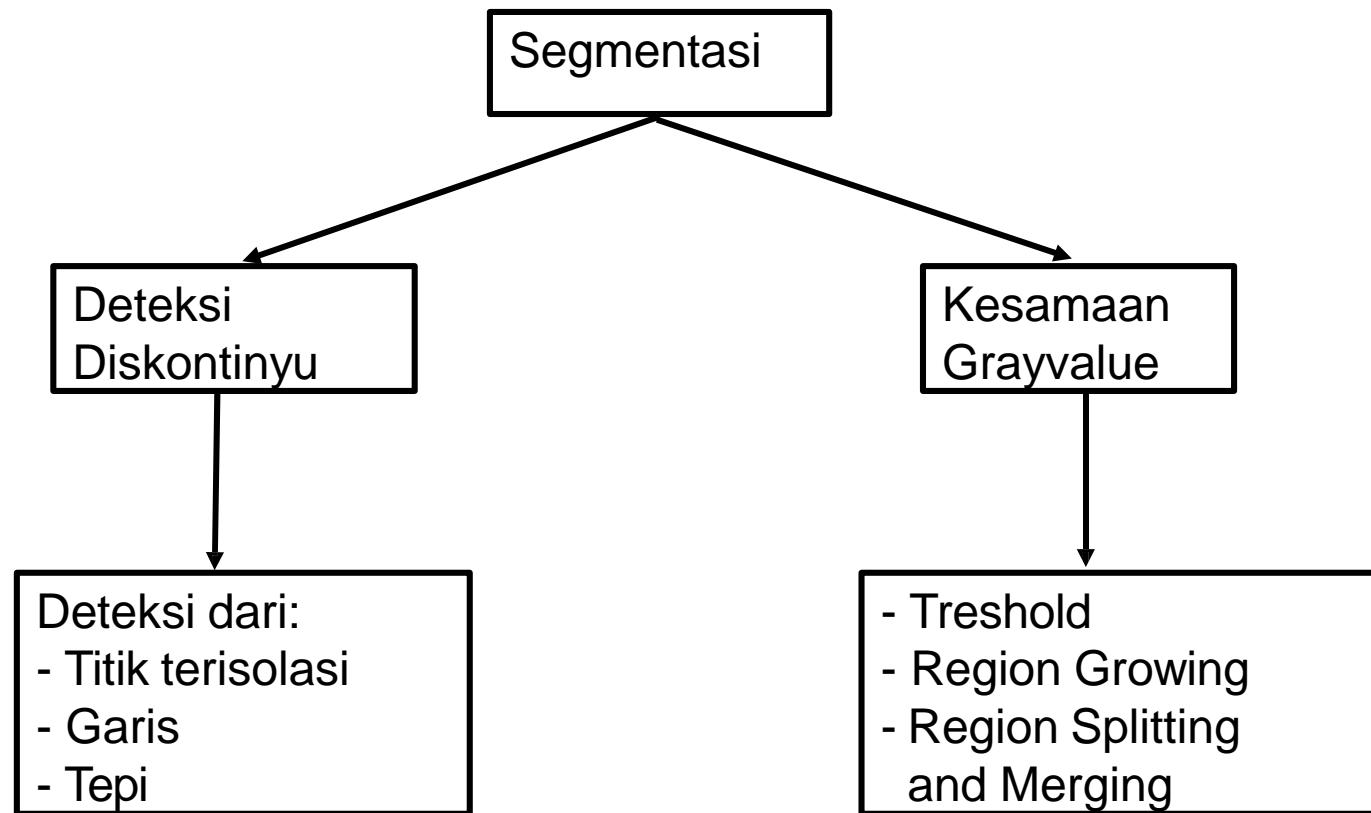


Image Segmentation

SEGMENTASI CITRA

- | **Segmentasi merupakan** proses mempartisi citra menjadi beberapa daerah atau objek
- | **Segmentasi Citra adalah** proses analisa citra yang akan membagi citra menjadi beberapa daerah terpisah untuk analisis lebih lanjut.
- | Daerah yang terpisah biasanya merupakan object-object yang berbeda.
- | Proses segmentasi ini merupakan bagian yang sangat penting dan biasanya merupakan proses yang rumit dalam pemrosesan Citra Digital.

SEGMENTASI CITRA



SEGMENTASI CITRA

- || Segmentasi citra pada umumnya berdasar pada sifat ***discontinuity*** atau ***similarity*** dari intensitas piksel
- || **Pendekatan *discontinuity*:**
 - || mempartisi citra bila terdapat perubahan intensitas secara tiba-tiba (edge based)
- || **Pendekatan *similarity*:**
 - || mempartisi citra menjadi daerah-daerah yang memiliki kesamaan sifat tertentu (region based)
 - || contoh: thresholding, region growing, region splitting and merging

JENIS ALGORITMA SEGMENTASI CITRA

I Diskontinuitas

- Pembagian citra berdasarkan perbedaan dalam intensitasnya
- Dapat menggunakan mask/kernel,
- Untuk setiap jenis deteksi memiliki mask/kernel yang berbeda
- Contoh: deteksi titik, deteksi garis, deteksi tepi/sisi

I Similaritas

- Pembagian citra berdasarkan kesamaan kriteria yang dimiliki
- Contoh: thresholding, mean clustering, region growing, region splitting, region merging

DISKONTINUITAS

I DETEKSI TITIK

- Mengisolasi suatu titik yang secara signifikan berbeda dengan titik-titik di sekitarnya.
- Graylevel dari titik tersebut jelas berbeda dengan tetangganya

Persamaan:

$$|R| \geq T \rightarrow R = \sum_{i=1}^9 w_i Z_i$$

Mask Umum

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

- $T \rightarrow$ tresshold positif; $R \rightarrow$ nilai persamaan

- Kemel yang dipergunakan:

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

Mask Laplacian

- Hasil output umumnya berupa **Threshold**

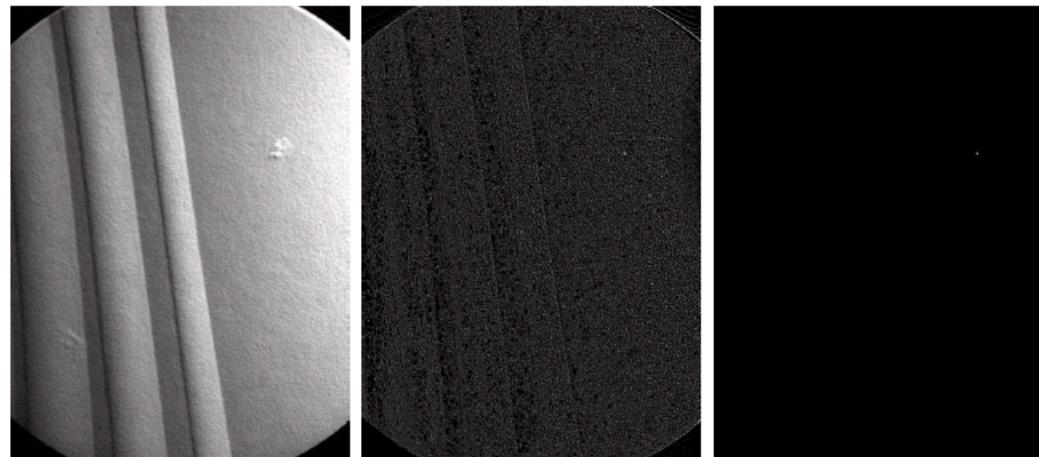
DISKONTINUITAS

I Contoh Deteksi Titik



DISKONTINUITAS

I Contoh Deteksi Titik



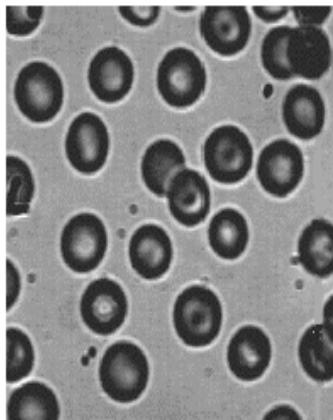
a
b c d

FIGURE 10.2

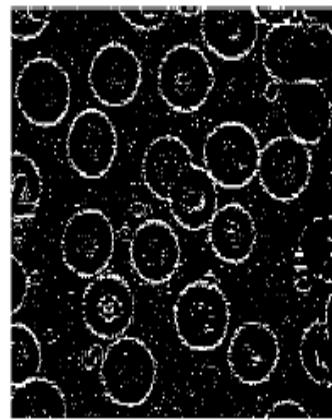
- (a) Point detection mask.
- (b) X-ray image of a turbine blade with a porosity.
- (c) Result of point detection.
- (d) Result of using Eq. (10.1-2). (Original image courtesy of X-TEK Systems Ltd.)

DISKONTINUITAS

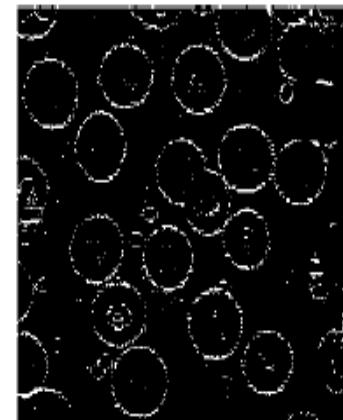
I Contoh Deteksi Titik



Orig. Image



$T = 0.25$



$T = 0.35$



$T = 0.5$

DISKONTINUITAS

|| Deteksi Garis

- || Mencocokkan dengan kernel dan menunjukkan bagian tertentu yang berbeda secara garis lurus vertikal, horizontal, diagonal kanan maupun diagonal kiri.
- || Digunakan untuk mendeteksi garis pada citra
- || Persamaan :
 $| R_i | > | R_j |$ dimana $i \neq j$

DISKONTINUITAS

I DETEKSI GARIS

- Dapat diselesaikan minimal dengan 4 mask:

$$\mathcal{D}_{0^\circ} = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$

Detects horizontal lines

$$\mathcal{D}_{45^\circ} = \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$$

Detects 45° lines

$$\mathcal{D}_{90^\circ} = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

Detects vertical lines

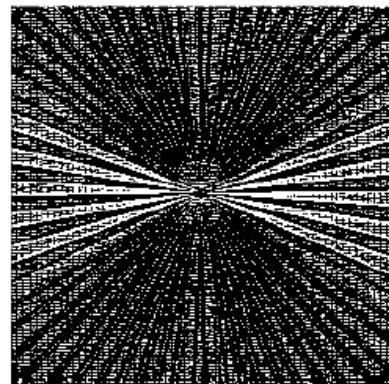
$$\mathcal{D}_{135^\circ} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Detects 135° lines

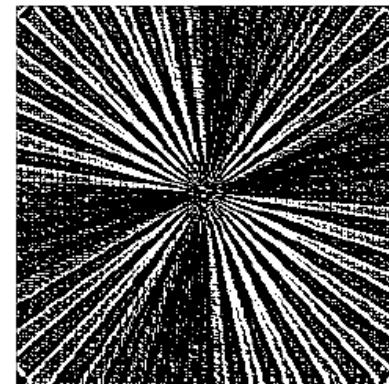
- Lakukan deteksi dengan 4 matrix secara berturut-turut, kemudian temukan nilai yang terbesar, misalnya ditemukan nilai terbesar setelah dikerjakan dengan matrix \mathcal{D}_{135° , maka dapat diartikan bahwa titik tersebut dilewati garis 135°

DISKONTINUITAS

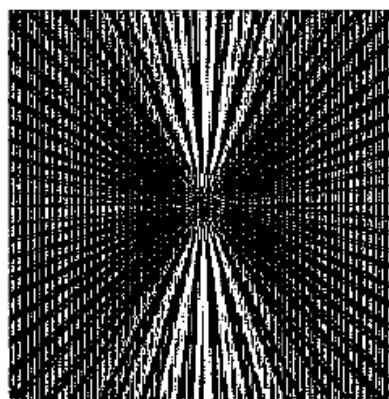
I Contoh Deteksi Garis



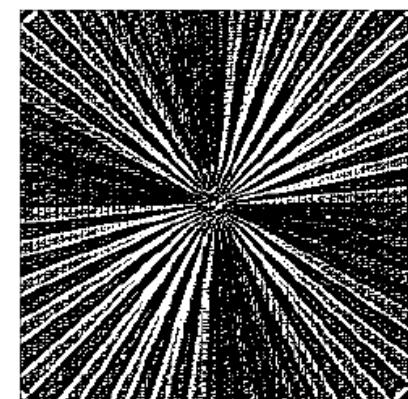
$$R_{0^\circ} = \max\{R_{0^\circ}, R_{45^\circ}, R_{90^\circ}, R_{135^\circ}\}$$



$$R_{45^\circ} = \max\{R_{0^\circ}, R_{45^\circ}, R_{90^\circ}, R_{135^\circ}\}$$



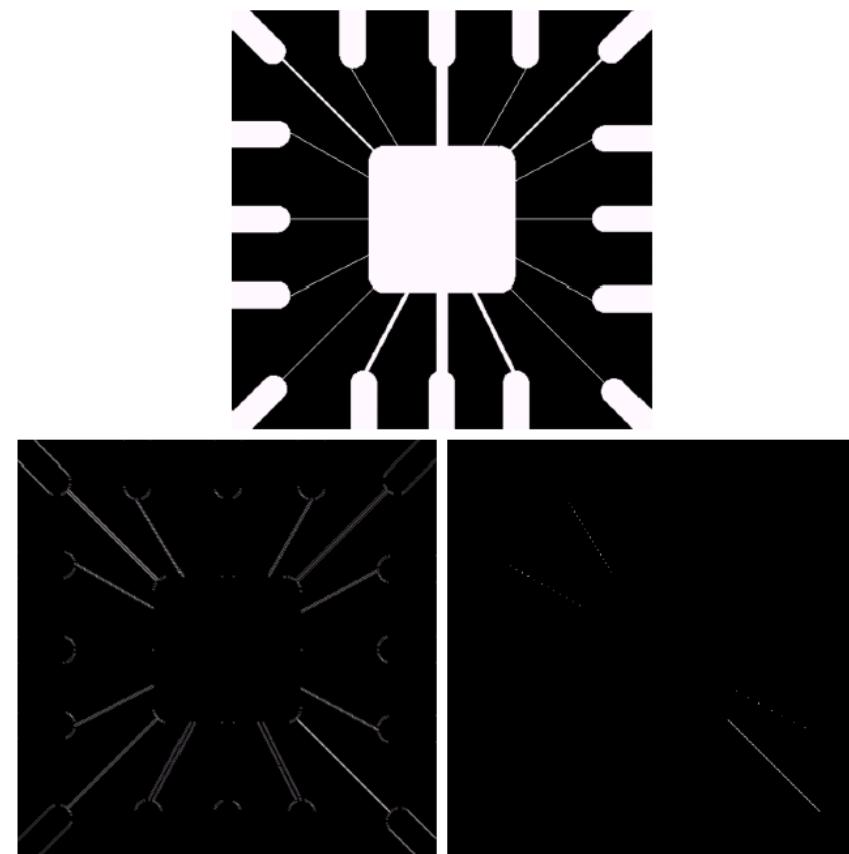
$$R_{90^\circ} = \max\{R_{0^\circ}, R_{45^\circ}, R_{90^\circ}, R_{135^\circ}\}$$



$$R_{135^\circ} = \max\{R_{0^\circ}, R_{45^\circ}, R_{90^\circ}, R_{135^\circ}\}$$

DISKONTINUITAS

I Contoh Deteksi Garis



a
b c

FIGURE 10.4
Illustration of line detection.
(a) Binary wire-bond mask.
(b) Absolute value of result after processing with -45° line detector.
(c) Result of thresholding image (b).

DISKONTINUITAS

■ DETEKSI TEPI

■ TANPA KONVOLUSI

- Homogeneity Operator (Operator Homogenitas)
- Pendekripsi Tepi Selisih (difference)
- Threshold Citra

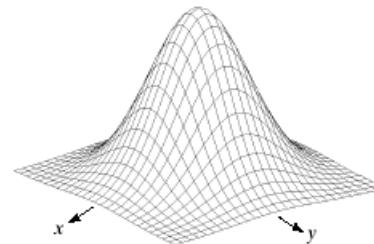
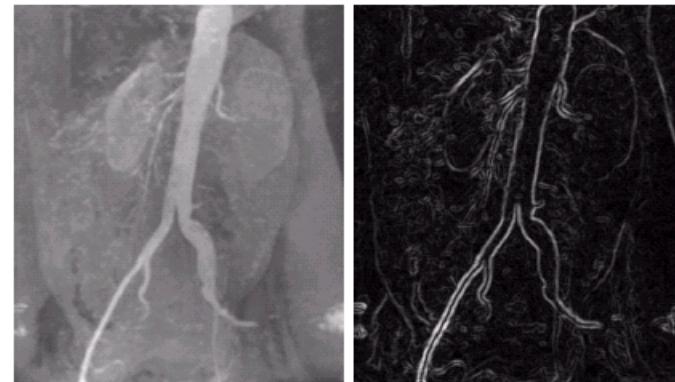
■ DENGAN KONVOLUSI

DISKONTINUITAS

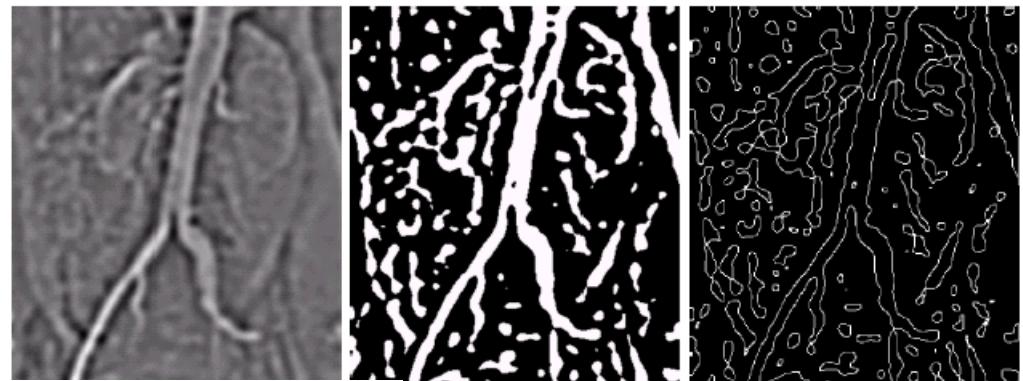
- | **DETEKSI TEPI**
 - | **DENGAN KONVOLUSI**
 - | **Operator Gradien Pertama :**
 - | Differential Gradient
 - | Center Difference
 - | Sobel
 - | Prewitt
 - | Roberts
 - | **Operator Turunan Kedua :**
 - | Laplacian
 - | Laplacian of Gaussian (LoG)
 - | **Operator Kompas**

DISKONTINUITAS

I Contoh Deteksi Tepi



-1	-1	-1
-1	8	-1
-1	-1	-1



a b
c d
e f g

FIGURE 10.15 (a) Original image. (b) Sobel gradient (shown for comparison). (c) Spatial Gaussian smoothing function. (d) Laplacian mask. (e) LoG. (f) Thresholded LoG. (g) Zero crossings. (Original image courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)



Citra grayscale



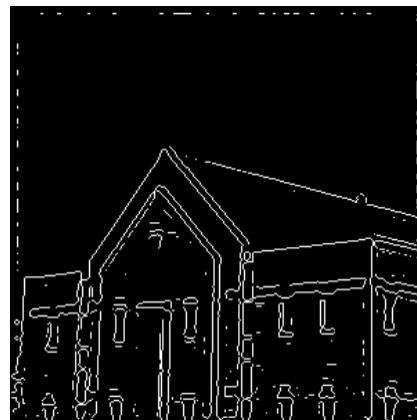
Hasil deteksi tepi Sobel
dengan threshold otomatis



Hasil deteksi tepi Prewitt
dengan threshold otomatis



Hasil deteksi tepi Robert
dengan threshold otomatis



Hasil deteksi tepi LoG
dengan threshold otomatis



Hasil deteksi tepi Canny
dengan threshold otomatis

```
>> [g_sobel_default, ts] = edge(f, 'sobel');      >> g_sobel_best = edge(i, 'sobel', 0.05);
>> [g_log_default, tlog] = edge(f, 'log');        >> g_log_best = edge(i, 'log', 0.003, 2.25);
>> [g_canny_default, tcan] = edge(f, 'canny');    >> g_canny_best = edge(i, 'canny', [0.04,
0.10], 1.5);
```

DISKONTINUITAS

- || Hasil dari deteksi sisi/tepi seringkali **tidak menghasilkan sisi/tepi yang lengkap**, karena adanya **noise, patahnya sisi karena iluminasi, dan lain-lain.**
- || Oleh karena itu proses deteksi sisi biasanya dilanjutkan dengan proses edge linking.
- || Dari sekian banyak cara, yang akan dibahas:
 - || Local Processing
 - || Global Processing dengan teknik Graph-Theoretic

DISKONTINUITAS

|| Local Processing

- || Merupakan cara paling sederhana, dengan menggunakan analisa ketetanggaan 3x3 atau 5x5.
- || Sifat utama yang digunakan untuk menentukan kesamaan piksel sisi adalah
 - || Nilai gradien: $\nabla f(x,y) = |G_x| + |G_y|$
 - || Arah gradien: $\alpha(x,y) = \tan^{-1}(G_x/G_y)$
- || Jika piksel pada posisi (x_0, y_0) adalah piksel sisi, maka piksel pada posisi (x, y) dalam jendela ketetanggaan yang sama bisa dikategorikan sisi pula jika
 - || $|\nabla f(x,y) - \nabla f(x_0,y_0)| \leq E$
 - || $|\alpha(x,y) - \alpha(x_0,y_0)| < A$
 - || E dan A adalah nilai ambang non negatif

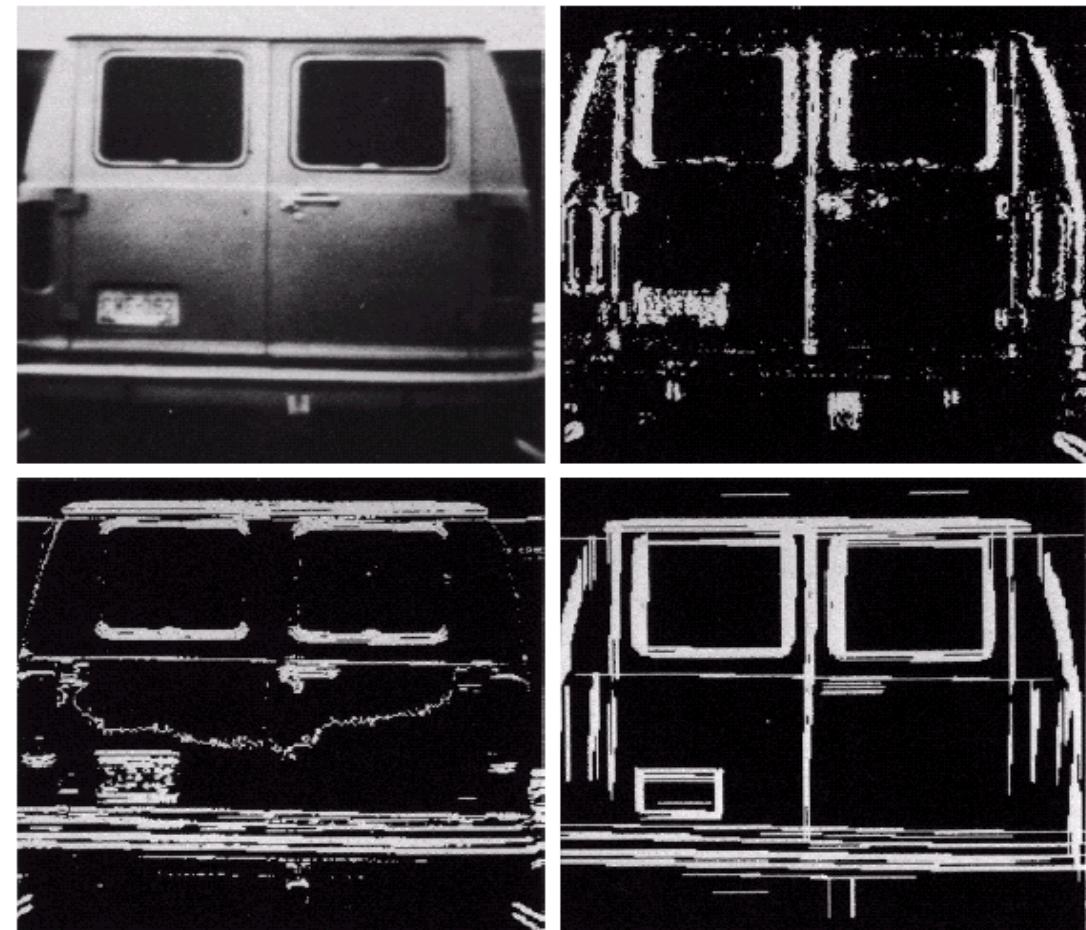
DISKONTINUITAS

Contoh Local Processing

a b
c d

FIGURE 10.16

- (a) Input image.
- (b) G_y component of the gradient.
- (c) G_x component of the gradient.
- (d) Result of edge linking. (Courtesy of Perceptics Corporation.)



DISKONTINUITAS

I Global Processing with graph theoretic

- Dapat mengatasi noise
- Komputasi lebih sulit
- Global : langsung mencakup seluruh citra, bukan menggunakan jendela ketetanggan
- Menggunakan representasi graph
 - Setiap piksel dianggap sebagai node
 - Setiap piksel hanya bisa dihubungkan dengan piksel lain jika mereka bertetangga (4-connected)
 - Edge yang menghubungkan piksel (node) p dan q pada graf memiliki nilai (weighted graph):
 - $C(p,q) = H - [f(p) - f(q)]$
 - H: nilai intensitas keabuan tertinggi pada citra
 - f(p) dan f(q): nilai keabuan piksel p dan q
 - Kemudian dicari lintasan sisi dengan cost terendah

DISKONTINUITAS

|| Contoh Global Processing

|| Diberikan sebuah citra dengan nilai graylevel sbb:

	0	1	2
0	(7)	(2)	(2)

1	(5)	(7)	(2)
---	-----	-----	-----

2	(5)	(1)	(0)
---	-----	-----	-----

|| Cari pixel2 mana saja yang harus dikoneksikan dan mana yang tidak.

DISKONTINUITAS

|| Contoh Global Processing

|| Rumus :

$$C(p,q) = H - [f(p) - f(q)]$$

H = nilai intensity tertinggi pada sebuah citra

f(p) = nilai intensitas dari p

f(q) = nilai intensitas dari q

|| Dan:

H=7 (karena intensitas tertinggi =7 / pada titik 0,0 dan 1,1)

DISKONTINUITAS

I Contoh Global Processing

$$C[0,0 - 0,1] = 7 - (7-2) = 2$$

$$C[0,1 - 0,2] = 7 - (2-2) = 7$$

$$C[1,0 - 1,1] = 7 - (7-2) = 9$$

$$C[1,1 - 1,2] = 7 - (2-2) = 2$$

$$C[2,0 - 2,1] = 7 - (7-2) = 3$$

$$C[2,1 - 2,2] = 7 - (2-2) = 6$$

$$C[0,0 - 1,0] = 7 - (7-2) = 9$$

$$C[1,0 - 2,0] = 7 - (2-2) = 7$$

$$C[0,1 - 1,1] = 7 - (7-2) = 12$$

$$C[1,1 - 2,1] = 7 - (2-2) = 1$$

$$C[0,2 - 1,2] = 7 - (7-2) = 7$$

$$C[1,2 - 2,2] = 7 - (2-2) = 5$$

$$C[0,1 - 0,0] = 7 - (7-2) = 12$$

$$C[0,2 - 0,1] = 7 - (2-2) = 7$$

$$C[1,1 - 1,0] = 7 - (7-2) = 5$$

$$C[1,2 - 1,1] = 7 - (2-2) = 12$$

$$C[2,1 - 2,0] = 7 - (7-2) = 11$$

$$C[2,2 - 2,1] = 7 - (2-2) = 8$$

$$C[1,0 - 0,0] = 7 - (7-2) = 5$$

$$C[2,0 - 1,0] = 7 - (2-2) = 7$$

$$C[1,1 - 0,1] = 7 - (7-2) = 2$$

$$C[2,1 - 1,1] = 7 - (2-2) = 13$$

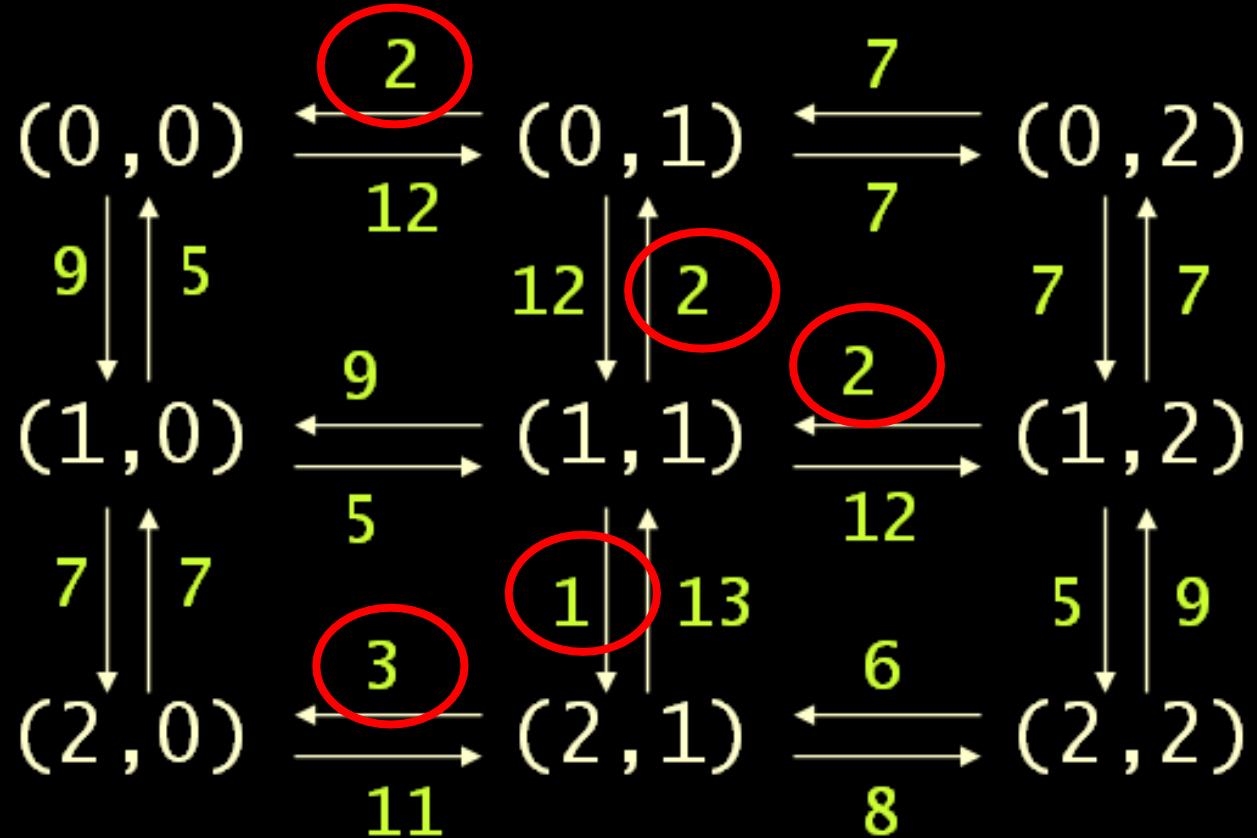
$$C[1,2 - 0,2] = 7 - (7-2) = 7$$

$$C[2,2 - 1,2] = 7 - (2-2) = 9$$

DISKONTINUITAS

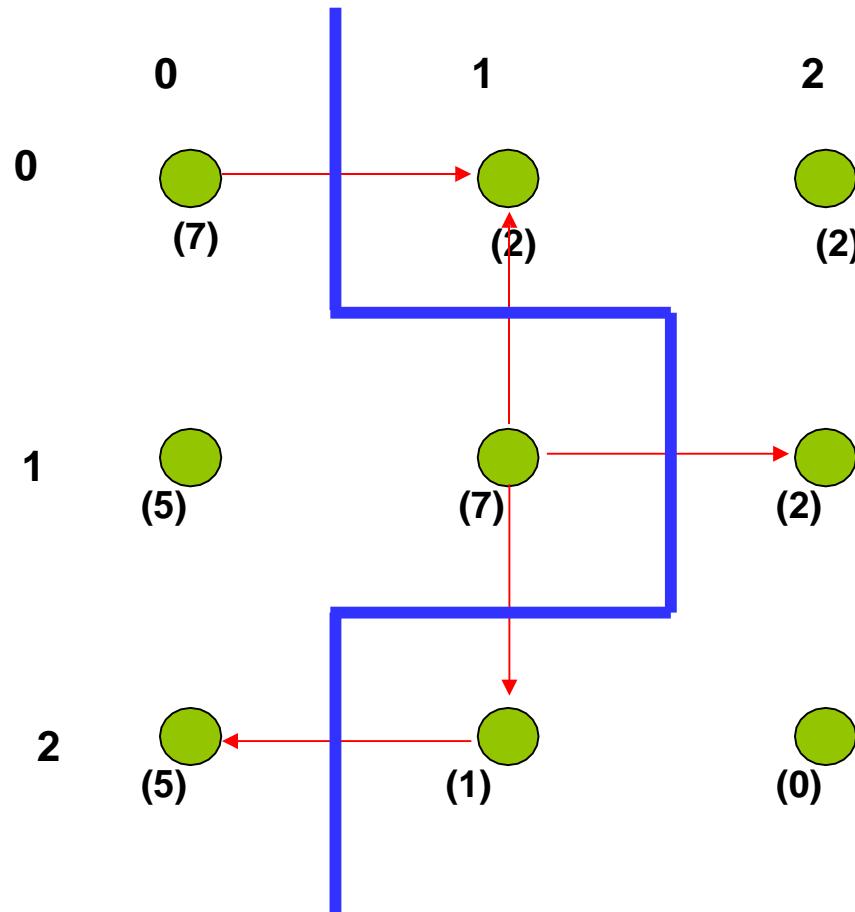
Contoh Global Processing

$$c(p,q) = H - [f(p) - f(q)]$$



DISKONTINUITAS

I Contoh Global Processing



DISKONTINUITAS

I Contoh Global Processing (2)

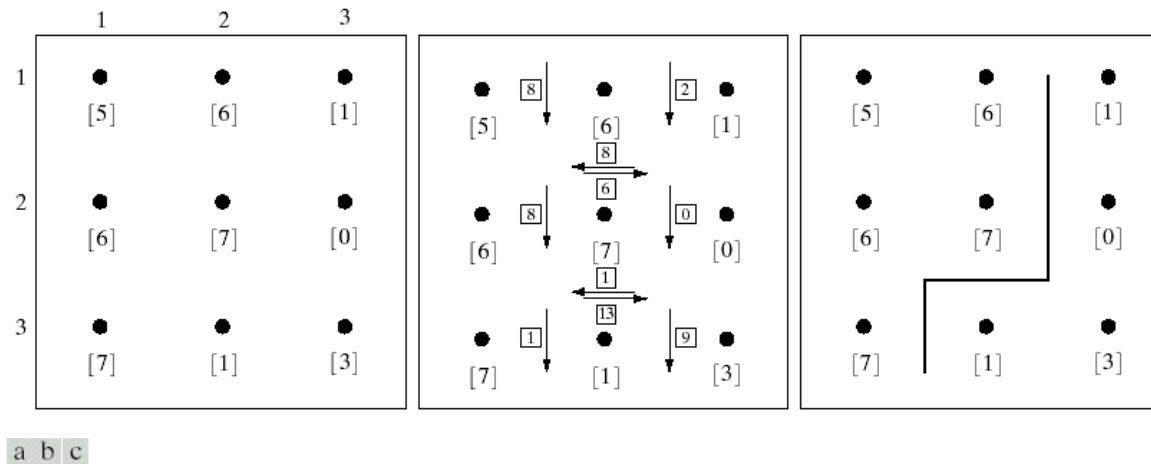


FIGURE 10.23 (a) A 3×3 image region. (b) Edge segments and their costs. (c) Edge corresponding to the lowest-cost path in the graph shown in Fig. 10.24.

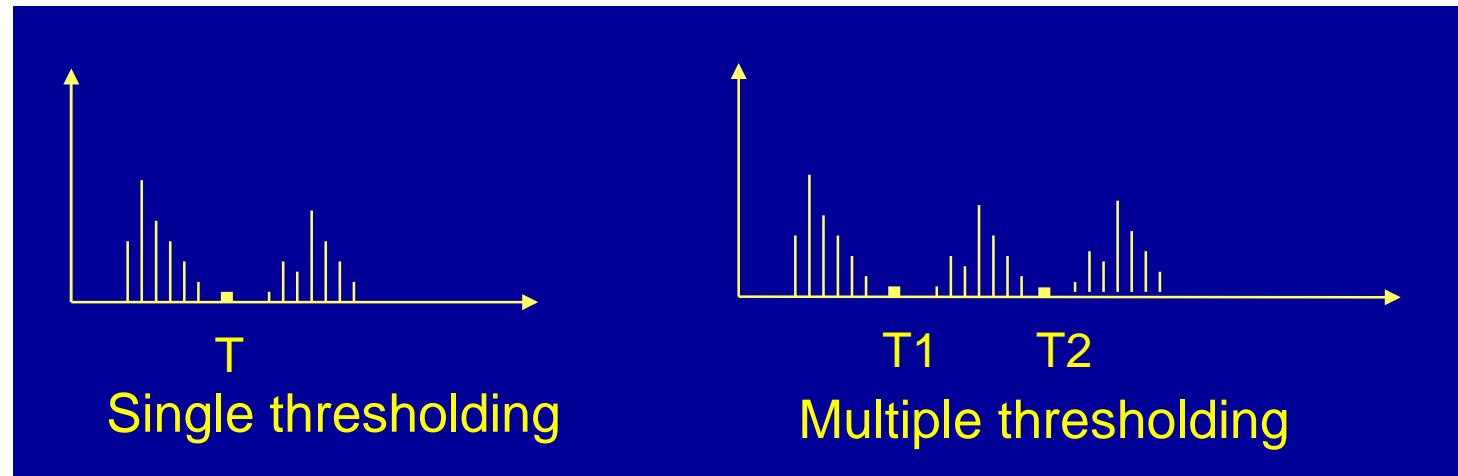
SIMILARITY (region based)

- | Pengelompokan berdasar distribusi properti pixel (warna), contoh : thresholding
- | Mencari region secara langsung berdasar 'persamaan' karakteristik suatu area, contoh: region growing, split & merge

SIMILARITY (region based)

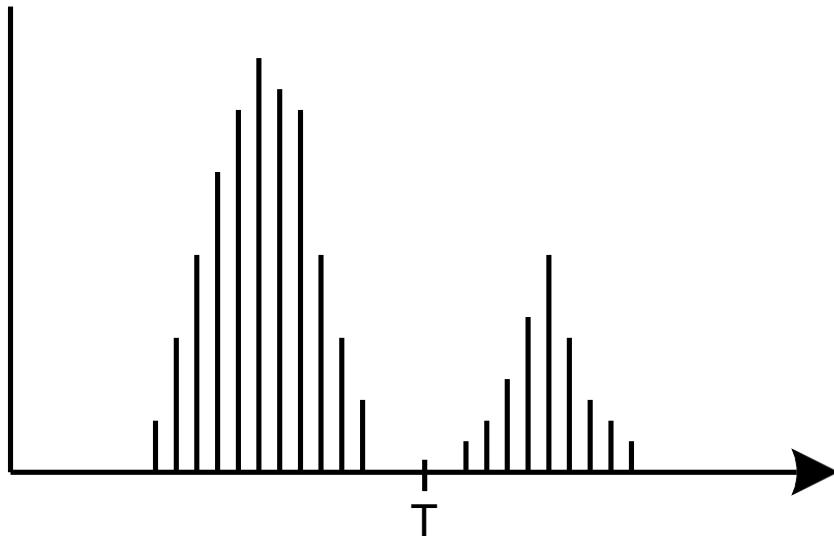
I Thresholding

- Sering digunakan untuk segmentasi karena mudah dan intuitif.
- Diasumsikan setiap objek cenderung memiliki warna yang homogen dan terletak pada kisaran keabuan tertentu

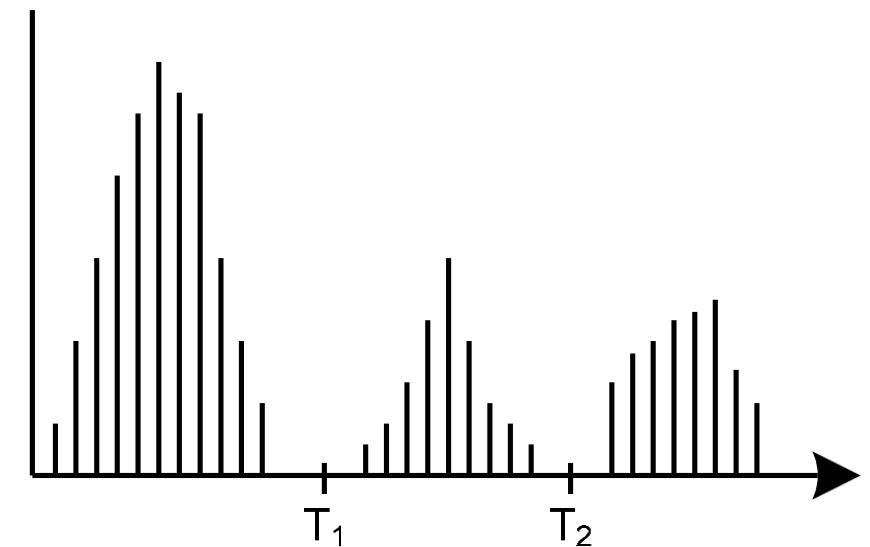


SIMILARITY (region based)

I Thresholding



$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases}$$



$$g(x, y) = \begin{cases} 0.0 & f(x, y) \leq T_1 \\ 0.5 & T_1 < f(x, y) \leq T_2 \\ 1.0 & T_2 < f(x, y) \end{cases}$$

SIMILARITY (region based)

|| Thresholding

|| Kelemahan :

- || Penentuan nilai threshold yang tepat
- || Bermasalah jika kemunculan tiap warna dalam citra cenderung sama → tidak bisa diprediksi batas antar objek

SIMILARITY (REGION BASED)

- || *Kekurangannya: belum tentu menghasilkan wilayah-wilayah yang bersambungan*
- || **Prosedur:**
 - || *Memerlukan criteria of uniformity (**kriteria**)*
 - || *Memerlukan penyebaran seeds atau dapat juga dengan pendekatan scan line*
 - || *Dilakukan proses region growing*

SIMILARITY (REGION BASED)

I Konsep Dasar Region Based

- || Anggap himpunan R adalah seluruh daerah citra. Kita akan mempartisi R menjadi daerah-daerah R_1, R_2, \dots, R_n , sedemikian hingga:
 - || $R_1 \cup R_2 \cup R_3 \cup \dots \cup R_n = R$
 - || R_i adalah daerah yang terhubung (untuk $i = 1, 2, \dots, n$)
 - || $R_i \cap R_j = \emptyset$ untuk semua i, j , $i \neq j$
 - || $P(R_i) = \text{TRUE}$ untuk $i = 1, 2, \dots, n$
 - || $P(R_i \cup R_j) = \text{FALSE}$ untuk $i \neq j$
- || P adalah predikat/kriteria tertentu yang harus dimiliki suatu daerah.

SIMILARITY (REGION BASED)

I Region Growing

- || Prosedur yang mengelompokkan pixel atau sub-region menjadi region yang lebih besar
- || Pendekatan paling sederhana: *pixel aggregation*
 - || Mulai dengan sekumpulan titik 'benih' (seed)
 - || Dari titik-titik tsb region diperluas dengan menambahkan titik-titik tetangganya yang memiliki properti yang sama (misal: gray level, tekstur, warna)
 - || Jika tidak ada lagi titik tetangga yang dapat ditambahkan lagi, maka proses untuk region tersebut dihentikan

SIMILARITY (REGION BASED)

I Region Growing (ILUSTRASI)

Seed

0	0	5	6	7
1	1	5	6	7
0	1	6	7	7
2	0	7	6	6
0	1	5	6	5

Citra asli

a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b

Hasil segmentasi;
perbedaan warna
absolut dg seed < 3

a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a

Hasil segmentasi;
perbedaan warna
absolut dg seed < 8

SIMILARITY (REGION BASED)

■ Region Growing

■ Masalah dengan region growing :

- Penentuan lokasi seeds yang tepat
 - Tergantung aplikasi
 - Misal: warna yang sering muncul, warna terang dll
- Penentuan properti yang tepat untuk mengelompokkan titik menjadi region
 - Tergantung masalah dan data citra yang tersedia
 - Misal: intensitas, tekstur, data multispektral dll
- Kondisi penghenti
 - Dasar: jika tidak ada lagi titik tetangga yang memenuhi syarat
 - Tambahan: ukuran region, bentuk dll

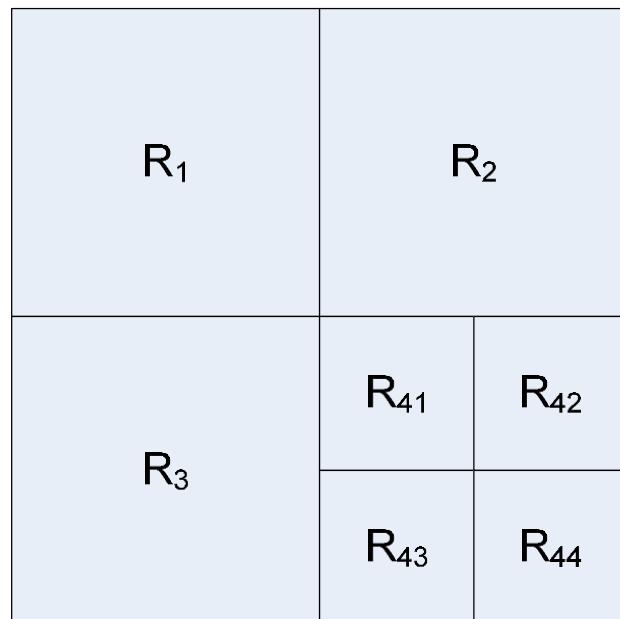
SIMILARITY (REGION BASED)

I Split & Merge

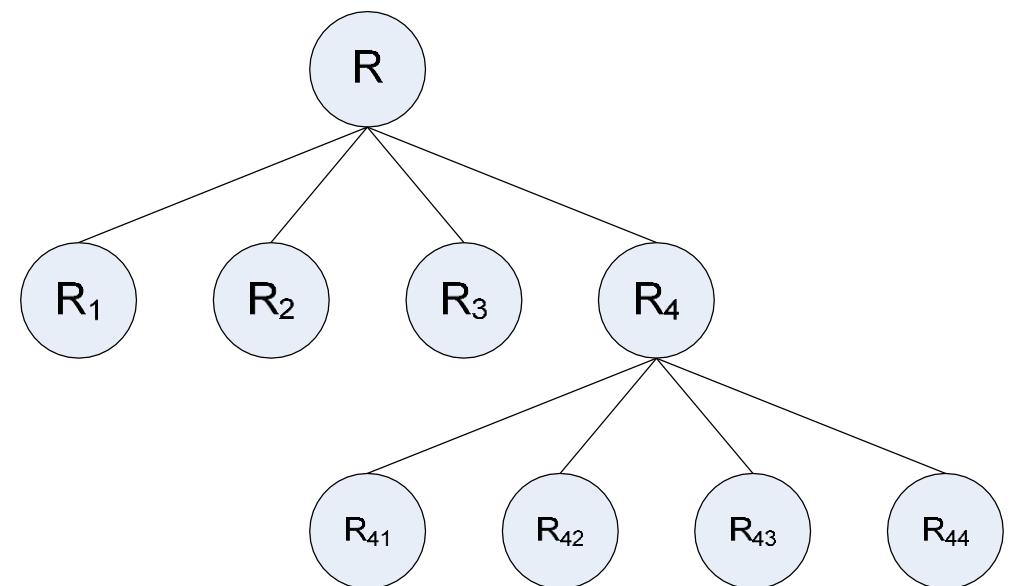
- || **Splitting:** membagi citra menjadi beberapa daerah berdasarkan kriteria tertentu (teknik quadtree)
- || **Merging:** gabungkan daerah-daerah berdekatan yang memiliki kriteria yang sama.
- || **Kriteria:** bisa varian keabuan dll
- || **Prosedur umum:**
 - || Split R menjadi 4 kuadran disjoint jika $P(R) = \text{FALSE}$
 - || Merge sembarang daerah berdekatan R_i, R_j jika $P(R_i \cup R_j) = \text{TRUE}$
 - || Berhenti jika tidak ada proses split n merge yang bisa dilakukan

SIMILARITY (REGION BASED)

I Split & Merge

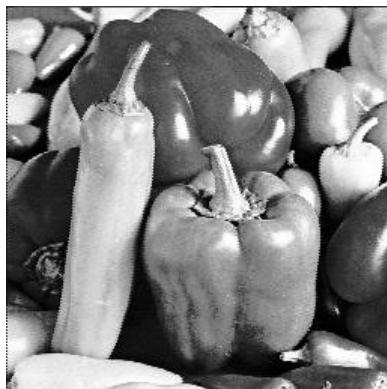


Citra terpartisi

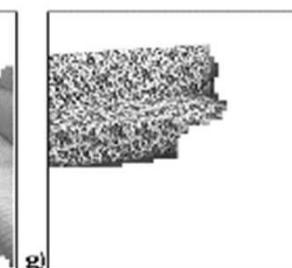
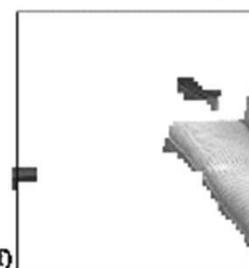
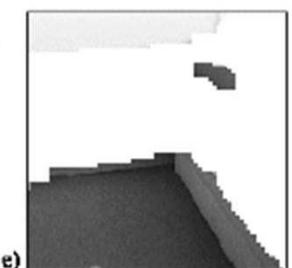
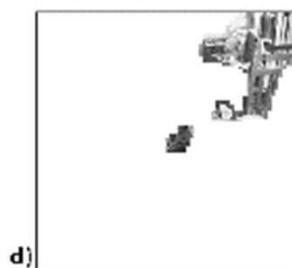
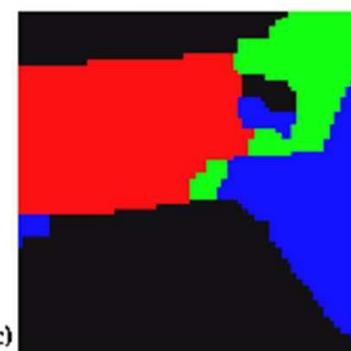
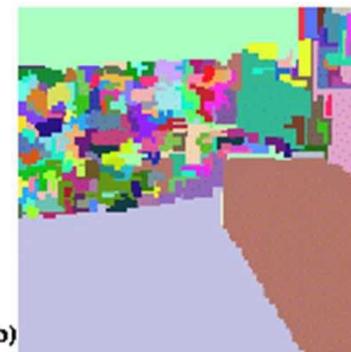


Representasi Quadtree

CONTOH SEGMENTASI



CONTOH SEGMENTASI





Morfologi

PEMROSESAN CITRA SECARA MORFOLOGIS

- || **Perbedaan antara pemrosesan citra secara morfologis dengan pemrosesan biasa (yang telah dipelajari)**
 - || Dulu sebuah citra dipandang sebagai suatu fungsi intensitas terhadap posisi (x,y)
 - || Dengan pendekatan morfologi, suatu citra dipandang sebagai himpunan

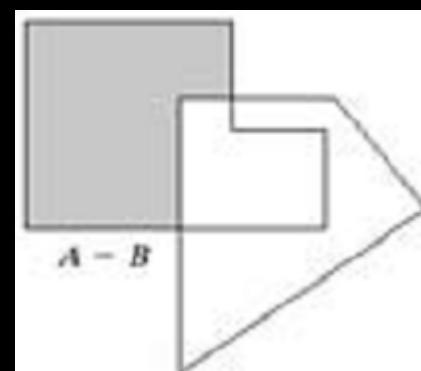
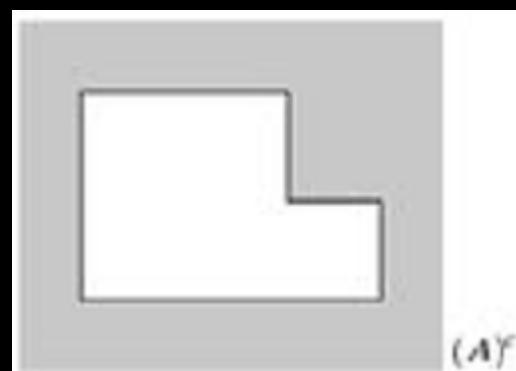
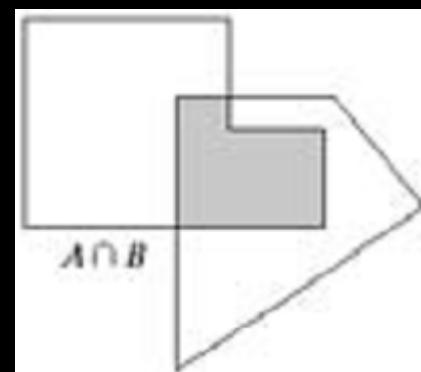
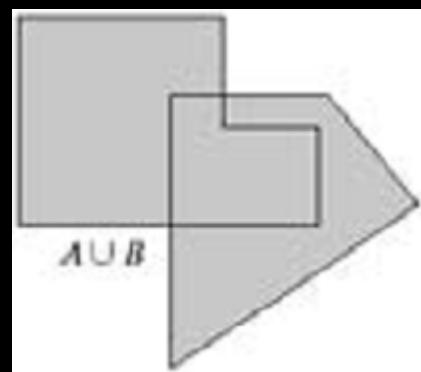
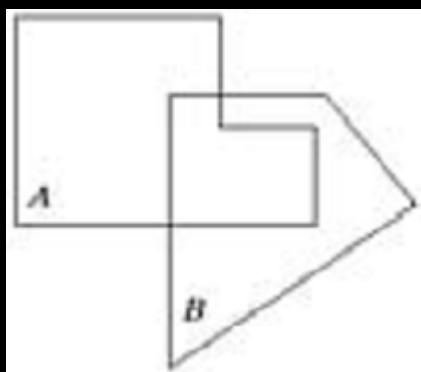
PEMROSESAN CITRA SECARA MORFOLOGIS

Pemrosesan citra secara morfologi biasanya dilakukan terhadap citra biner

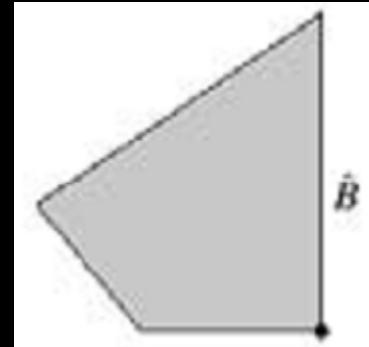
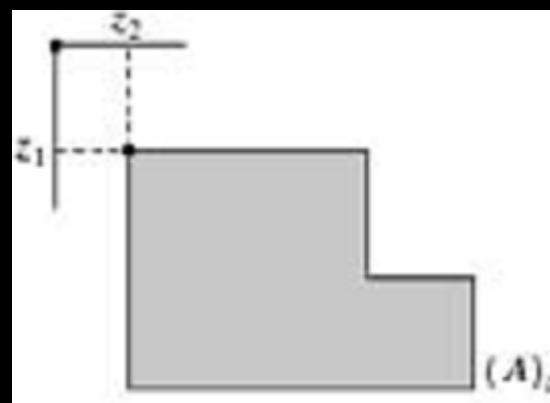
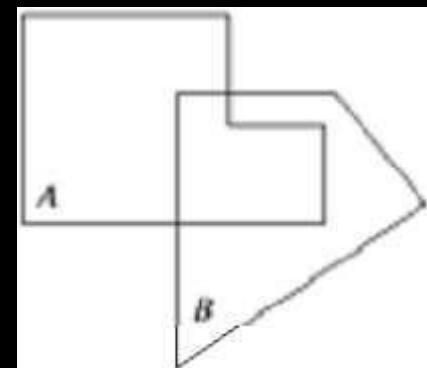
Tidak menutup kemungkinan dilakukan terhadap citra dengan skala keabuan 0-255

Yang akan dipelajari adalah pemrosesan morfologi terhadap citra biner

SET CITRA DAN OPERASINYA



TRANSLASI DAN REFLEKSI



TIPE OPERASI LOGIKA DASAR

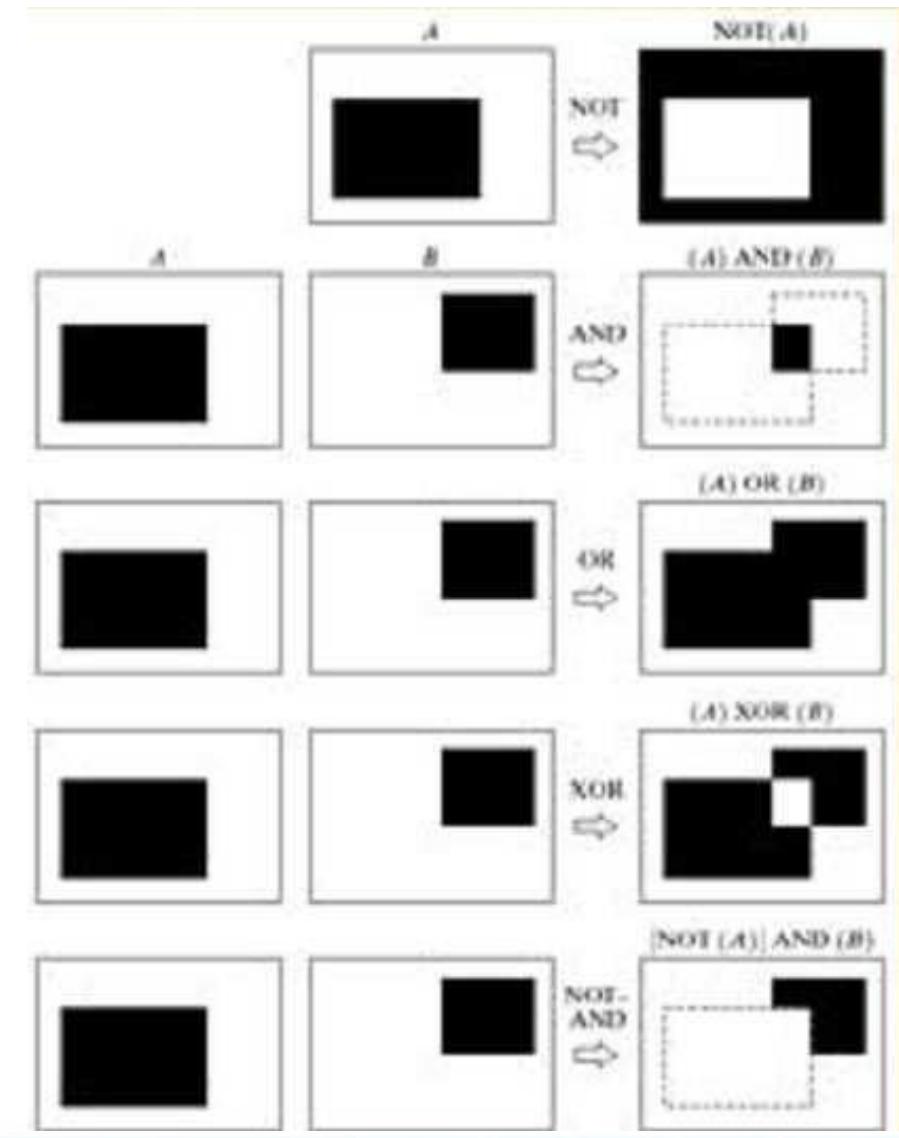
p	q	p AND q	p OR q	NOT p
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

BEBERAPA CONTOH

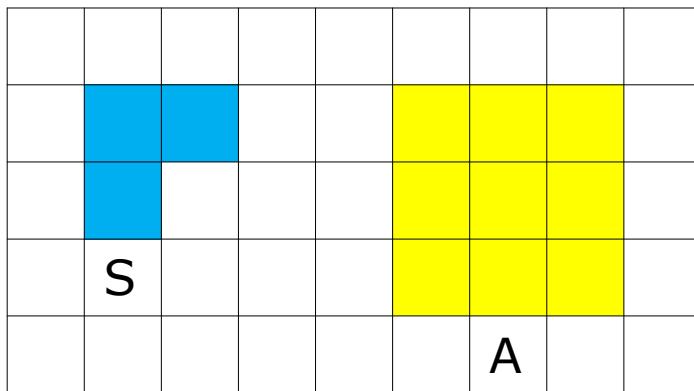
Catatan :

1 → Hitam

0 → Putih



Contoh Citra Masukan



$$S = \{(0,0), (0,1), (1,0)\}$$

$$A = \{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2)\}$$

Objek S dan A dapat direpresentasikan dalam bentuk himpunan dari posisi -posisi (x,y) yang bernilai 1 (1 = berwama, 0 = putih)

OPERASI MORFOLOGI

Secara umum, pemrosesan citra secara morfologi dilakukan dengan cara mem-passing sebuah *structuring element* terhadap sebuah citra dengan cara yang hampir sama dengan konvolusi

***Structuring element* dapat diibaratkan dengan mask pada pemrosesan citra biasa (bukan secara morfologi)**

STRUCTURING ELEMENT



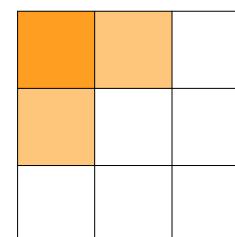
Structuring element dapat berukuran sembarang



Structuring element juga memiliki titik poros (disebut juga titik origin/titik asal/titik acuan)



Contoh *structuring element* seperti objek S dengan titik poros di (0,0) yang berwarna merah



S



OPERASI MORFOLOGI

- | **DILASI & EROSI**
- | **OPENING & CLOSING**
- | **THINNING & TICKENING**

DILASI

$$D(A, S) = A \oplus S$$

Dilasi merupakan proses penggabungan titik-titik latar (0) menjadi bagian dari objek (1), berdasarkan structuring element S yang digunakan

Cara: untuk setiap titik pada A

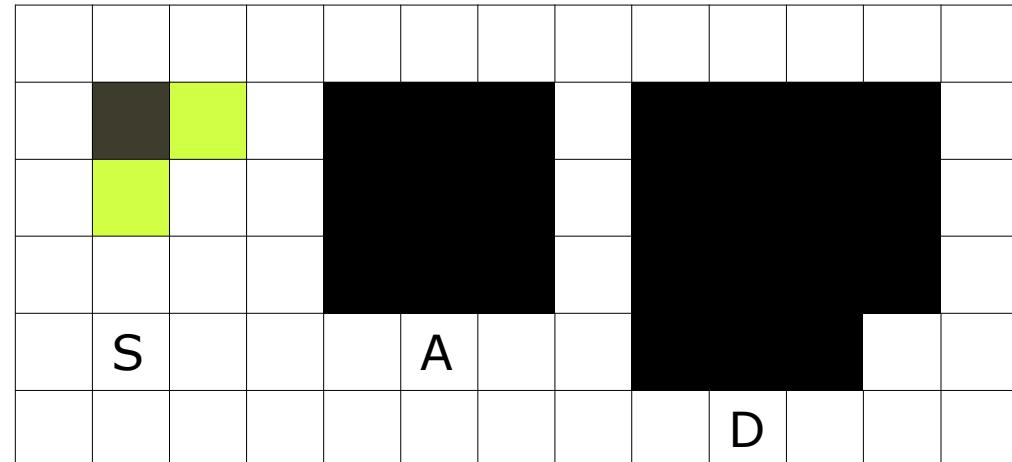
Letakkan titik poros S pada titik A tersebut

Beri angka 1 untuk semua titik (x,y) yang terkena / tertimpa oleh struktur S pada posisi tersebut

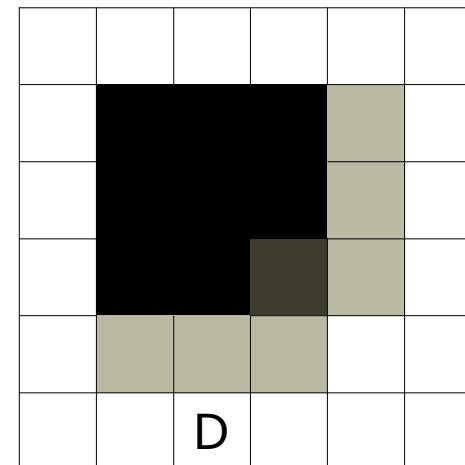
Contoh DILASI

$$S = \{(0,0), (0,1), (1,0)\}$$

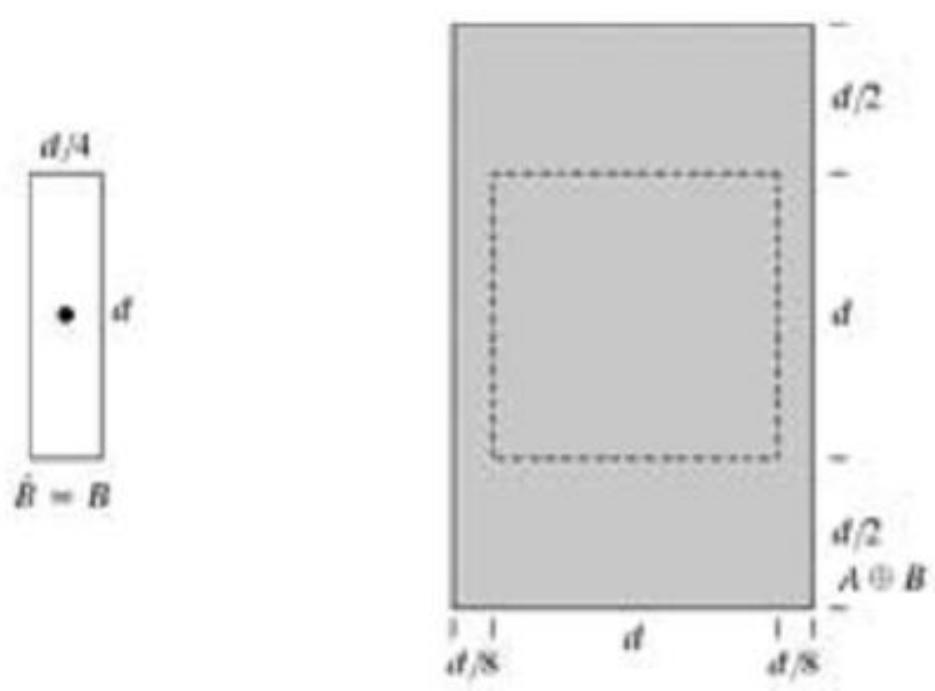
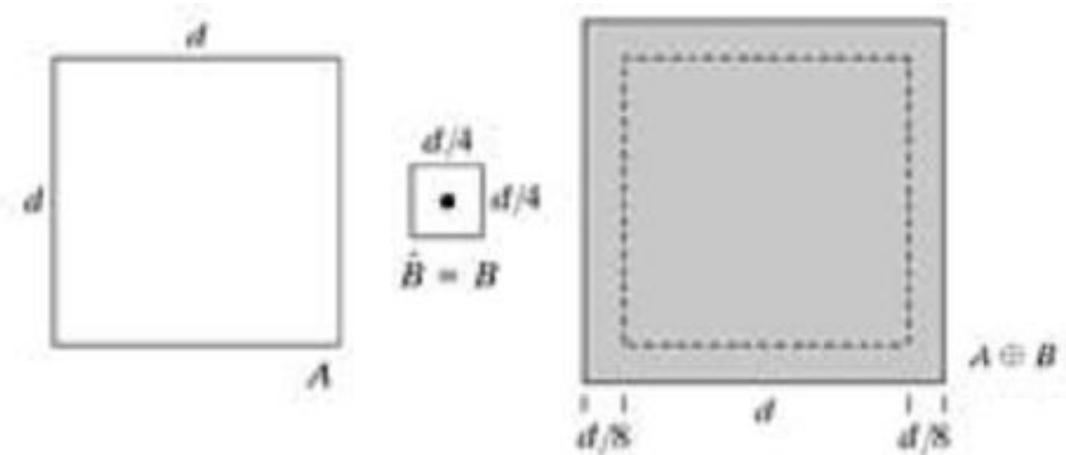
$$A = \{(0,0), (0,1), (0,2),\\ (1,0), (1,1), (1,2),\\ (2,0), (2,1), (2,2)\}$$



Posisi poros $((x,y) \in A)$	S_{xy}
(0,0)	$\{(0,0), (0,1), (1,0)\}$
(0,1)	$\{(0,1), (0,2), (1,1)\}$
(0,2)	$\{(0,2), (0,3), (1,2)\}$
.....
(2,2)	$\{(2,2), (2,3), (3,2)\}$



Contoh DILASI



EROSI

$$E(A, S) = A \ominus S$$

Erosi merupakan proses penghapusan titik-titik objek (1) menjadi bagian dari latar belakang (0), berdasarkan structuring element S yang digunakan

Cara: untuk setiap titik pada A

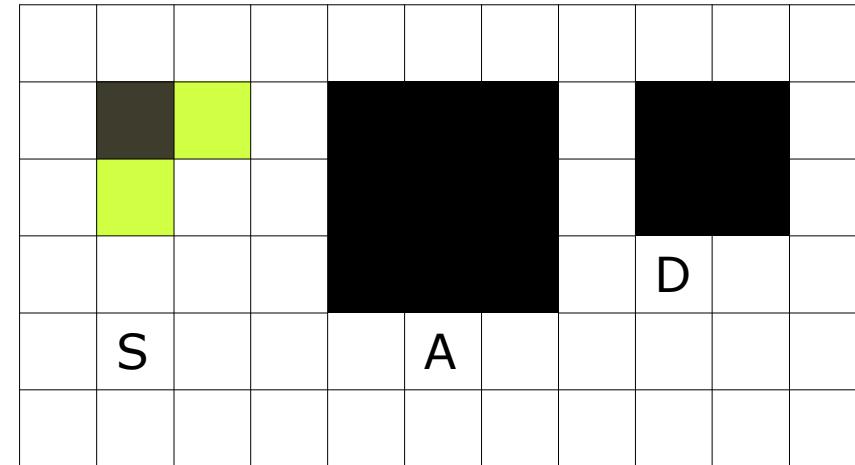
Letakan titik poros Spada titik A tersebut

Jika ada bagian dari S yang berada di luar A , maka titik poros dihapus/dijadikan latar

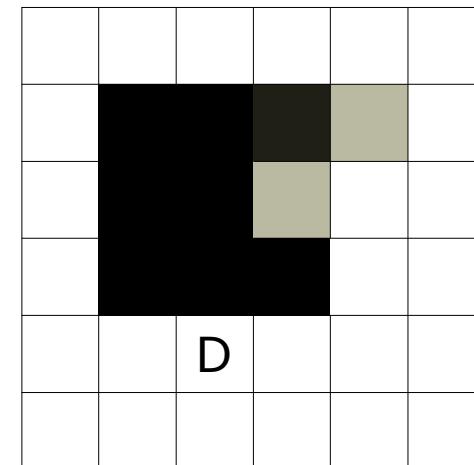
Contoh EROSI

$$S = \{(0,0), (0,1), (1,0)\}$$

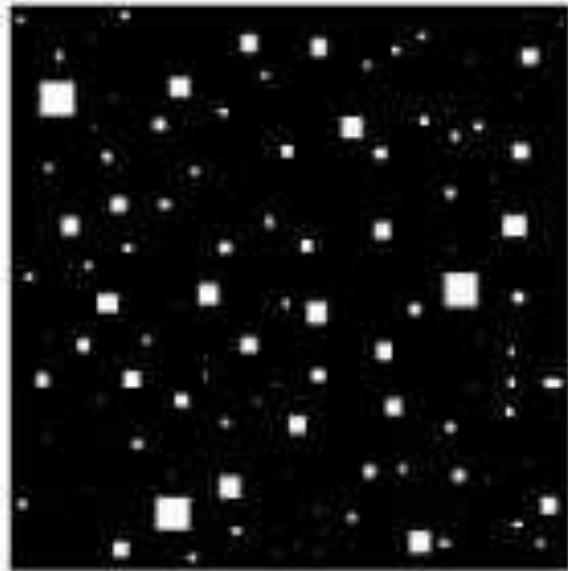
$$A = \{(0,0), (0,1), (0,2), \\ (1,0), (1,1), (1,2), \\ (2,0), (2,1), (2,2)\}$$



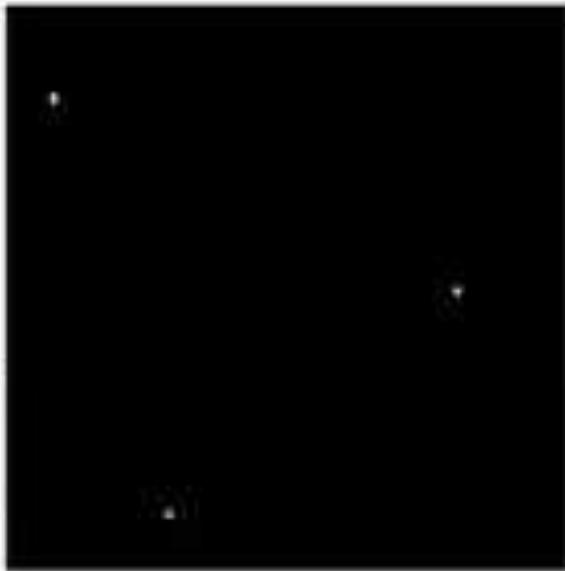
Posisi poros $((x,y) \in A)$	S_{xy}
(0,0)	$\{(0,0), (0,1), (1,0)\}$
(0,1)	$\{(0,1), (0,2), (1,1)\}$
(0,2)	$\{(0,2), (0,3), (1,2)\}$
.....
(2,2)	$\{(2,2), (2,3), (3,2)\}$



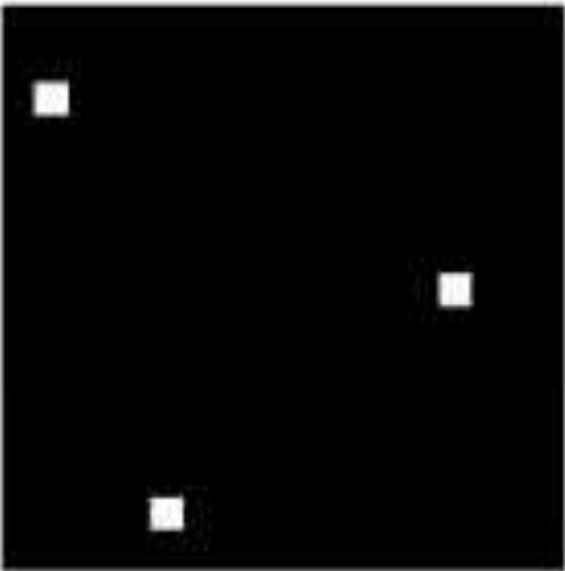
CONTOH APLIKASI



(a) Citra dengan kotak putih yang berukuran 1, 3, 5, 7, 9, dan 15 piksel



(b) Erosi citra (a) dengan ukuran matriks structuring element 13 piksel dan semua elemennya bernilai 1



(c) Dilasi citra (b) dengan structuring element yang sama

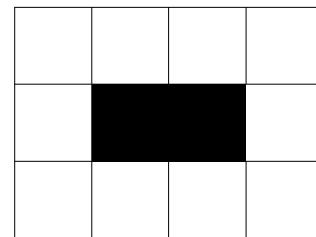
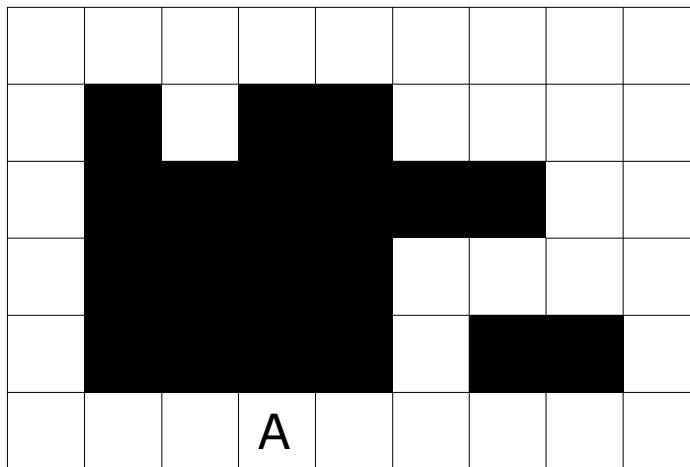
OPENING

$$A \circ S = (A \Theta S) \oplus S$$

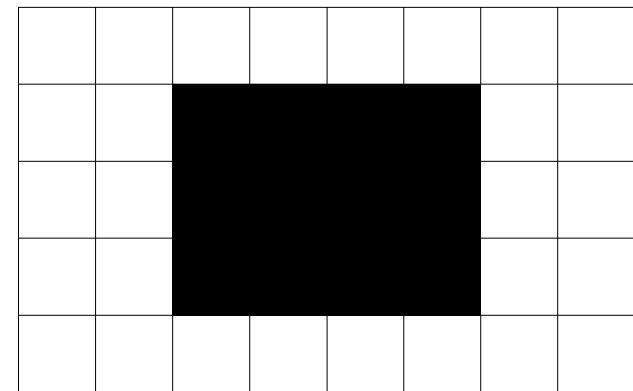
Opening adalah proses erosi yang diikuti dengan dilasi

Efek yang dihasilkan adalah menghilangnya objek-objek kecil dan kurus, dan memecah objek pada titik-titik yang kurus,

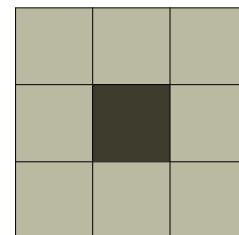
CONTOH OPENING



$A \Theta S$

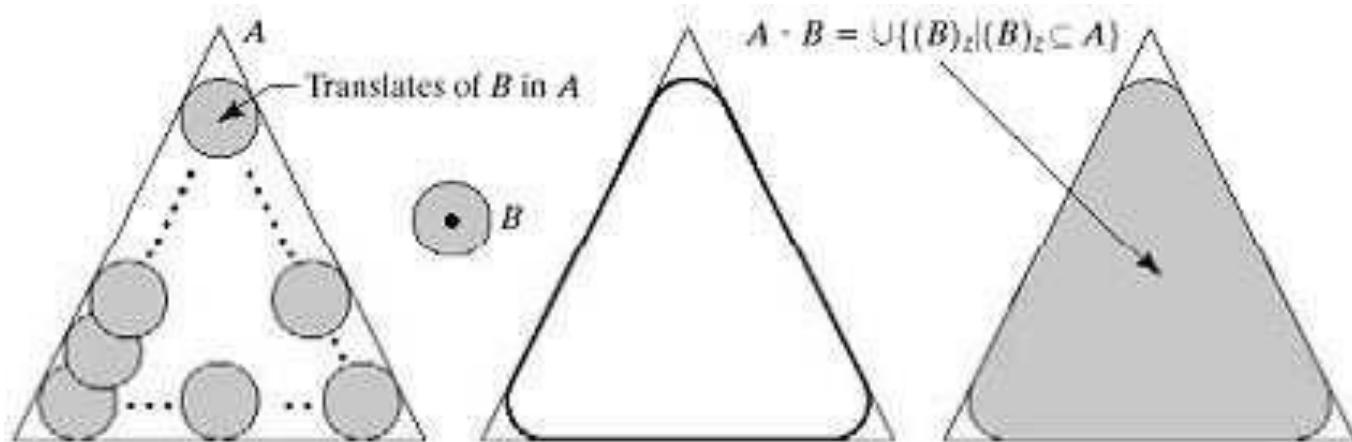


$(A \Theta S) \oplus S$



S

CONTOH LAIN



CLOSING

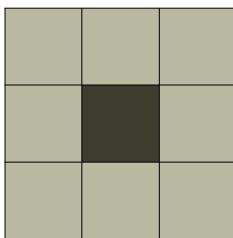
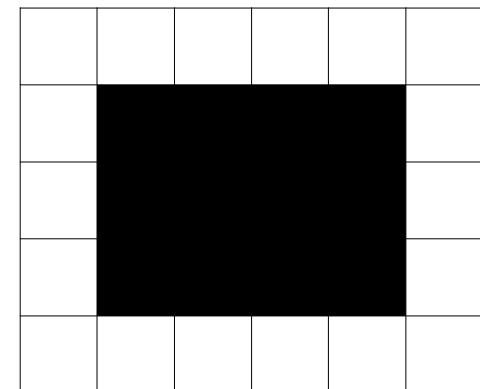
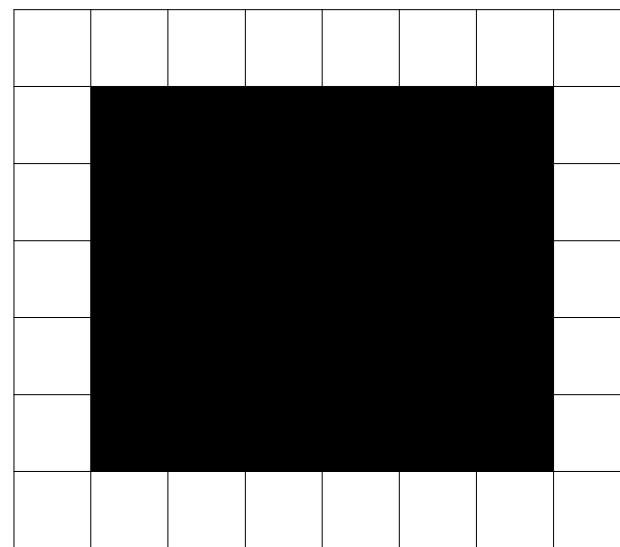
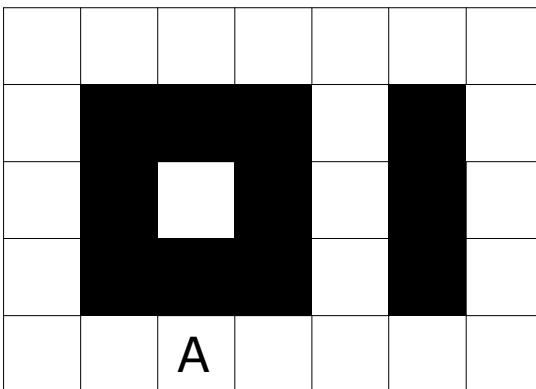
$$A \bullet S = (A \oplus S) \Theta S$$

Closing adalah proses dilasi yang diikuti dengan erosi

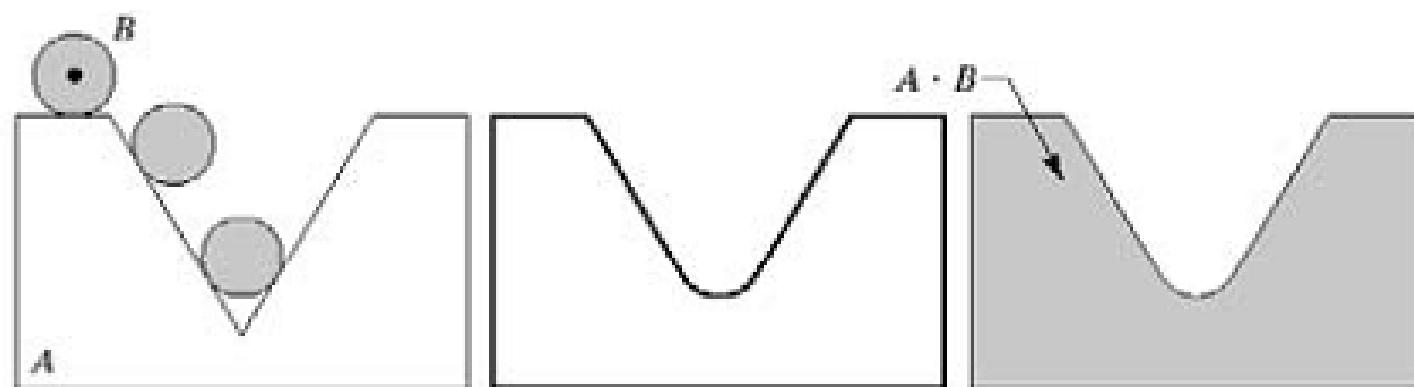
Efek yang dihasilkan adalah mengisi lubang-lubang kecil pada objek, dan menggabungkan objek-objek yang berdekatan



CONTOH CLOSING

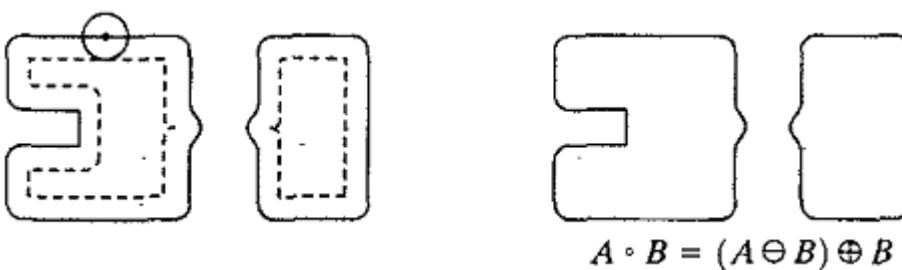
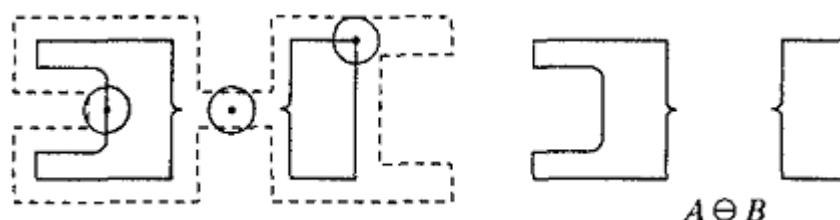
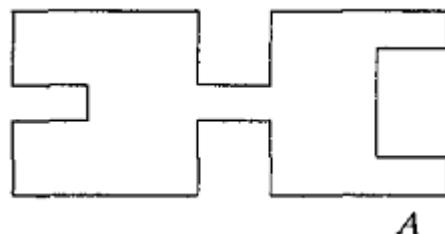


CONTOH LAIN



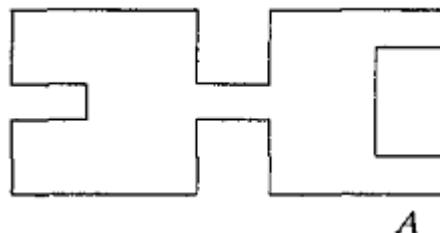
CONTOH OPENING & CLOSING

Opening

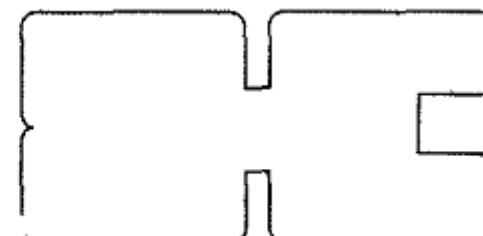
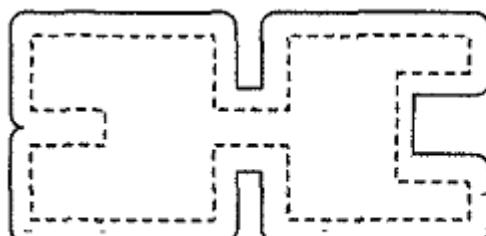


CONTOH OPENING & CLOSING

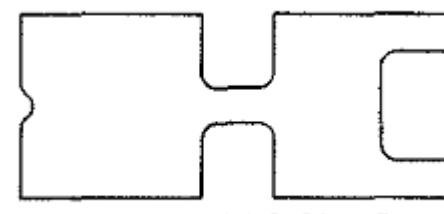
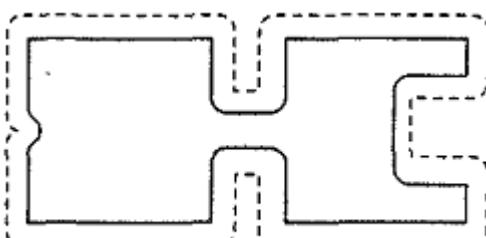
Closing



A

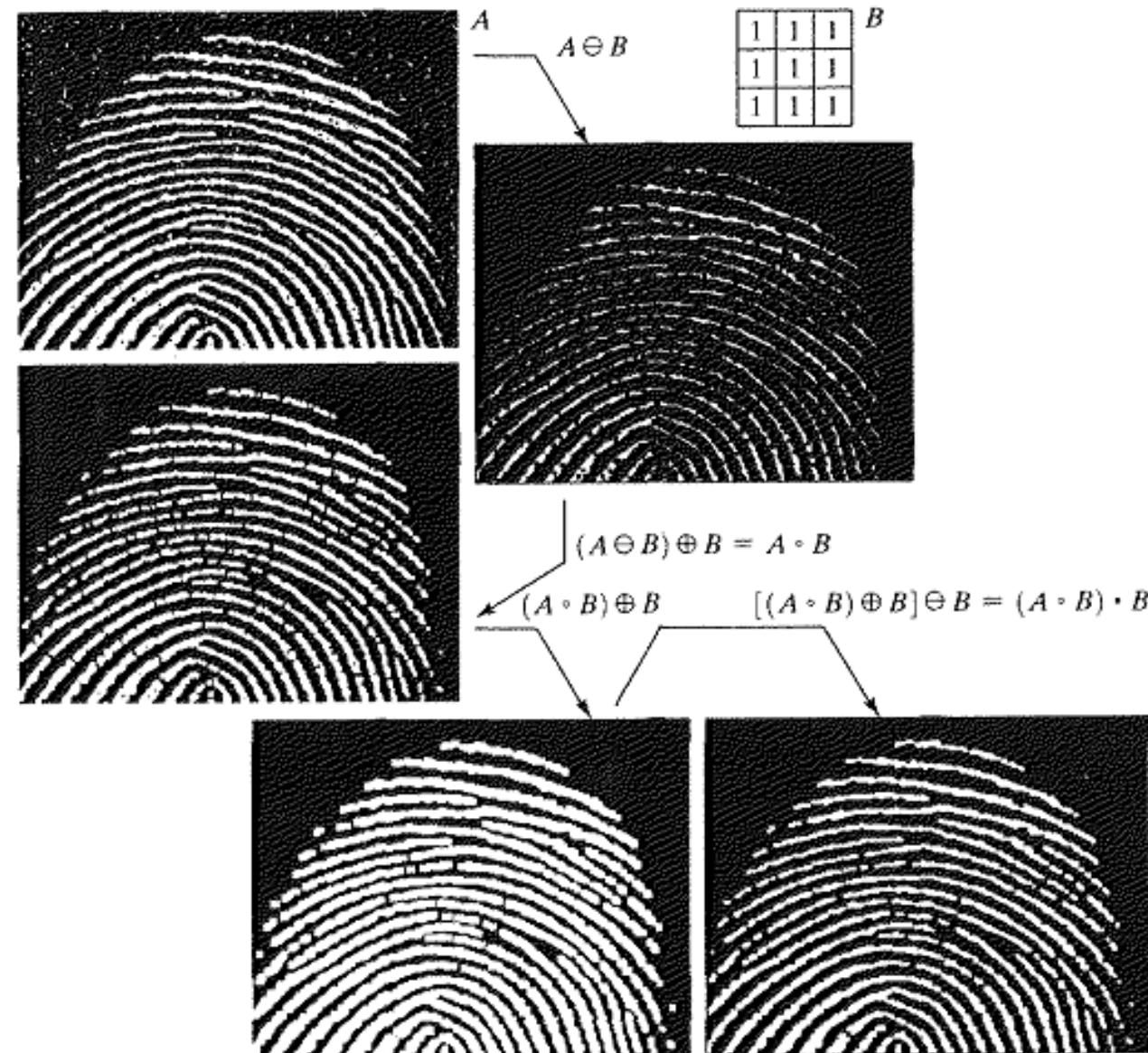


A ⊕ B



$$A \cdot B = (A \oplus B) \ominus B$$

CONTOH APLIKASI



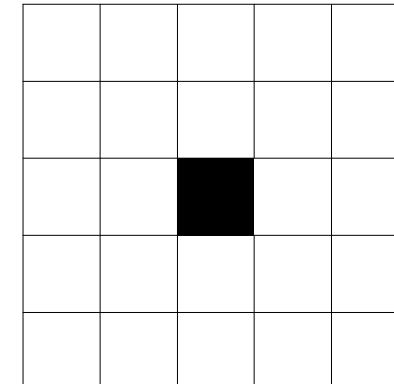
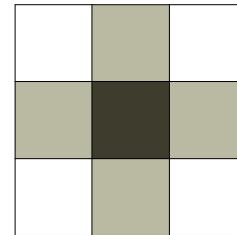
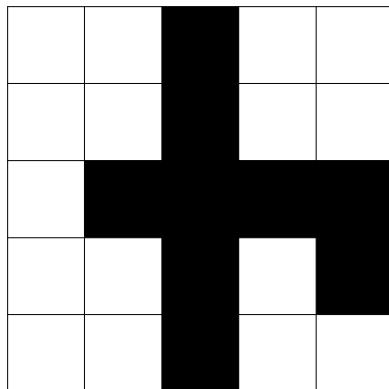
HIT -OR- MIS TRANSFORM

$$A^* S = (A \Theta S_1) \cap (A^c \Theta S_2)$$

Suatu structuring element S dapat direpresentasikan dalam bentuk (S₁,S₂) dimana S₁ adalah kumpulan titik-titik objek (hitam) dan S₂ adalah kumpulan titik-titik latar(putih)

Hit-and-miss transform A*S adalah kumpulan titik-titik dimana S₁ menemukan match di A dan pada saat yang bersamaan S₂ juga menemukan match di luar A

Contoh HIT -OR- MIS TRANSFORM



- Yang match dipertahankan
- Yang tidak match dihapus

ALGORITMA MORFOLOGI

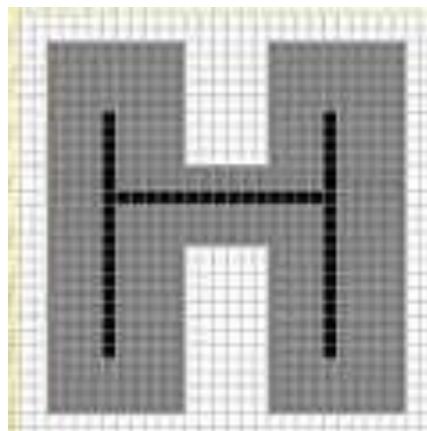
■ **Thinning**

- Menguruskan objek dalam citra

■ **Thickening**

- Menebalkan objek pada citra

THINNING



Salah satu kegunaan thinning adalah pada proses pengenalan karakter/huruf

Ada banyak cara mengimplementasikan thinning, salah satu diantaranya adalah dengan hit-or-miss transform

$$A \otimes B = A - (A * B) = A \cap (A * B)^c$$

THINNING

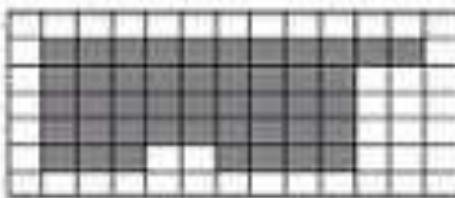
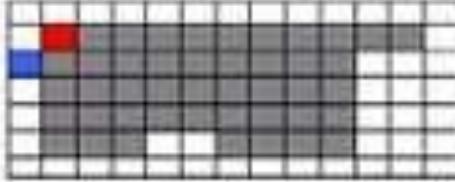
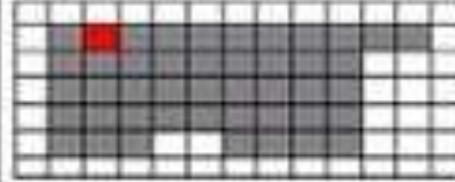
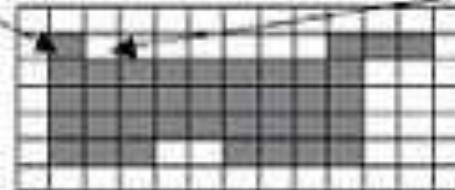
I Dapat didefinisikan sebagai :

- I Thinning $(A, \{B\}) = A - (A * \{B\}) = A - ((\dots(A * B_1) *$
 $B_2)...B_n)$
- I $B_1, B_2, B_3, \dots, B_n$ adalah structuring element

I NOTE :

- I $A - (A * B)$ berarti kebalikan dari $A * B$
- I Yang match dihapus
- I Yang tidak match dipertahankan

Contoh THINNING

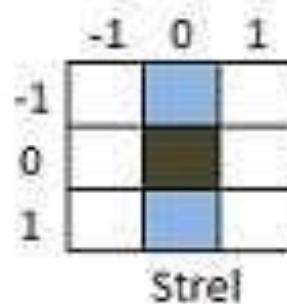
 <p>Reference: pojok kiri atas adalah posisi (0,0)</p>	 <p>B^1 origin: merah Objek: biru Latar: kuning Don't care: arus</p>
 <p>Ketika origin B^1 ada di posisi (0,0) Ada bagian biru yang tidak match. Tidak semua match \rightarrow (0,0) dipertahankan</p>	 <p>Ketika origin B^1 ada di posisi (0,1) Bagian biru match semua di objek. Bagian kuning match semua di latar Semua match \rightarrow (0,1) dihapus</p>
	

SOAL 1

- I Tentukan citra output, jika dilakukan Operasi :
1. Opening
 2. Thinning untuk citra dibawah ini :

	1	2	3	4	5
1	0	0	0	0	0
2	0	1	1	1	0
3	0	1	1	1	0
4	0	0	1	0	0
5	0	0	0	0	0

Citra Asli





Pemrograman Pengolahan Citra

Pengenalan Phyton

Phyton

- Python merupakan bahasa pemrograman tingkat tinggi yang dirancang oleh Guido Van Rossum
- Untuk mempelajari pemrograman python diperlukan:
 - Python: Interpreter yang menerjemahkan bahasa python ke bahasa mesin, sehingga program bisa dijalankan.
 - Teks Editor/IDE: Program yang digunakan untuk menulis kode



Anaconda

- Anaconda adalah paket distribusi Python dari Continuum Analytics yang berisi paket Python ditambah beberapa paket tambahan untuk pemrograman data science, matematika hingga teknik dalam satu distribusi platform
- Penggunaan Anaconda cukup User Friendly jika dibandingkan dengan aplikasi bawaan dari Phyton yang belum terinstall package yang dibutuhkan.
- Anaconda menyediakan berbagai package bawaan, seperti Jupyter, Numpy, Pandas, Spyder dll



Pustaka Phyton

- Python dapat digunakan untuk pengolahan citra.
- Terdapat sepuluh pustaka (*library*) Python yang paling umum digunakan untuk tugas manipulasi citra.

Jupyter

- Jupyter Notebook lebih dikenal dengan sebutan Jupyter.
- Jupyter pengembangan dari IPython, IPython disini bertindak sebagai kernel dan Jupyter menggunakan antarmuka Notebook Interface.
- Jupyter dapat dikatakan sebagai editor dalam bentuk web aplikasi di localhost komputer
- Jupyter dapat digunakan untuk menulis kode Python, equations, visualisasi dan bisa juga sebagai Markdown editor.



scikit-image

- scikit-image adalah Python open source yang bekerja dengan array NumPy yang mengimplementasikan algoritma dan utilitas untuk digunakan dalam penelitian, pendidikan, dan aplikasi industri.

NumPy

- NumPy adalah salah satu pustaka inti dalam pemrograman Python dan menyediakan dukungan untuk array.
- Citra pada dasarnya adalah array NumPy standar yang mengandung piksel titik data. Oleh karena itu, dengan menggunakan operasi NumPy dasar, seperti *slicing*, *masking*, dan *fancy indexing*, nilai piksel suatu citra dapat diubah.
- Citra dapat dimuat menggunakan skimage dan ditampilkan menggunakan Matplotlib.

SciPy

- SciPy adalah modul inti ilmiah Python lainnya (seperti NumPy) dan dapat digunakan untuk manipulasi citra dasar dan pengilahan citra.
- Secara khusus, submodule *scipy.ndimage* (dalam SciPy v1.1.0) menyediakan fungsi yang beroperasi pada array NumPy n-dimensi.
- Paket saat ini mencakup fungsi untuk filtering linier dan non-linear, morfologi biner, interpolasi B-spline, dan pengukuran objek.

PIL/Pillow

- PIL (Python Imaging Library) adalah pustaka gratis untuk bahasa pemrograman Python yang menambahkan dukungan untuk membuka, memanipulasi, dan menyimpan banyak format file gambar yang berbeda.
- Pengembangannya dihentikan, dengan rilis terakhir pada tahun 2009.
- Sebagai penggantinya ada Pillow yang lebih mudah untuk diinstal, dapat berjalan pada semua sistem operasi utama, dan mendukung Python 3.
- Pustaka ini berisi citra fungsi dasar pengolahan citra, termasuk operasi titik, *filtering* dengan set kernel konvolusi bawaan, dan konversi ruang-warna.

OpenCV-Python

- OpenCV (*Open Source Computer Vision Library*) adalah salah satu pustaka yang paling banyak digunakan untuk aplikasi computer vision.
- OpenCV-Python adalah API Python untuk OpenCV.
- OpenCV-Python tidak hanya cepat, karena latar belakang terdiri dari kode yang ditulis dalam C / C ++ dan Python di latar depan, tetapi juga mudah untuk kode dan digunakan

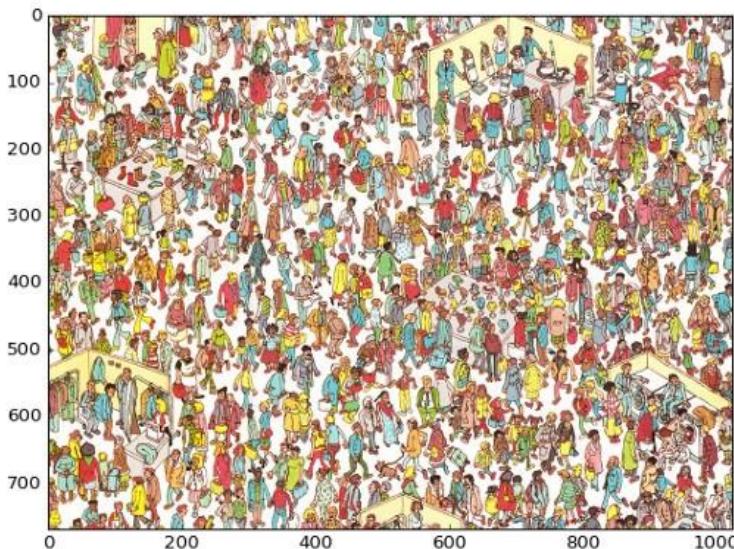
SimpleCV

- SimpleCV adalah *framework* untuk membangun aplikasi *computer vision* yang bersifat *open source*.
- SimpleCV menawarkan akses ke beberapa pustaka *computer vision* seperti OpenCV, tanpa harus tahu tentang kedalaman bit, format file, ruang warna, dll.
- Kurva pembelajarannya secara substansial lebih kecil daripada OpenCV

Mahotas

- Mahotas adalah pustaka computer vision dan pengolahan citra lainnya untuk Python.
- Berisi fungsi pengolahan citra seperti operasi *filtering* dan morfologi, serta fungsi *computer vision* yang lebih modern untuk komputasi fitur, termasuk deteksi titik dan deskriptor lokal.
- Antarmuka menggunakan Python dan algoritma diimplementasikan dalam C++ .
- Pustaka Mahotas bekerja cepat dengan kode minimalis dan ketergantungan minimum.

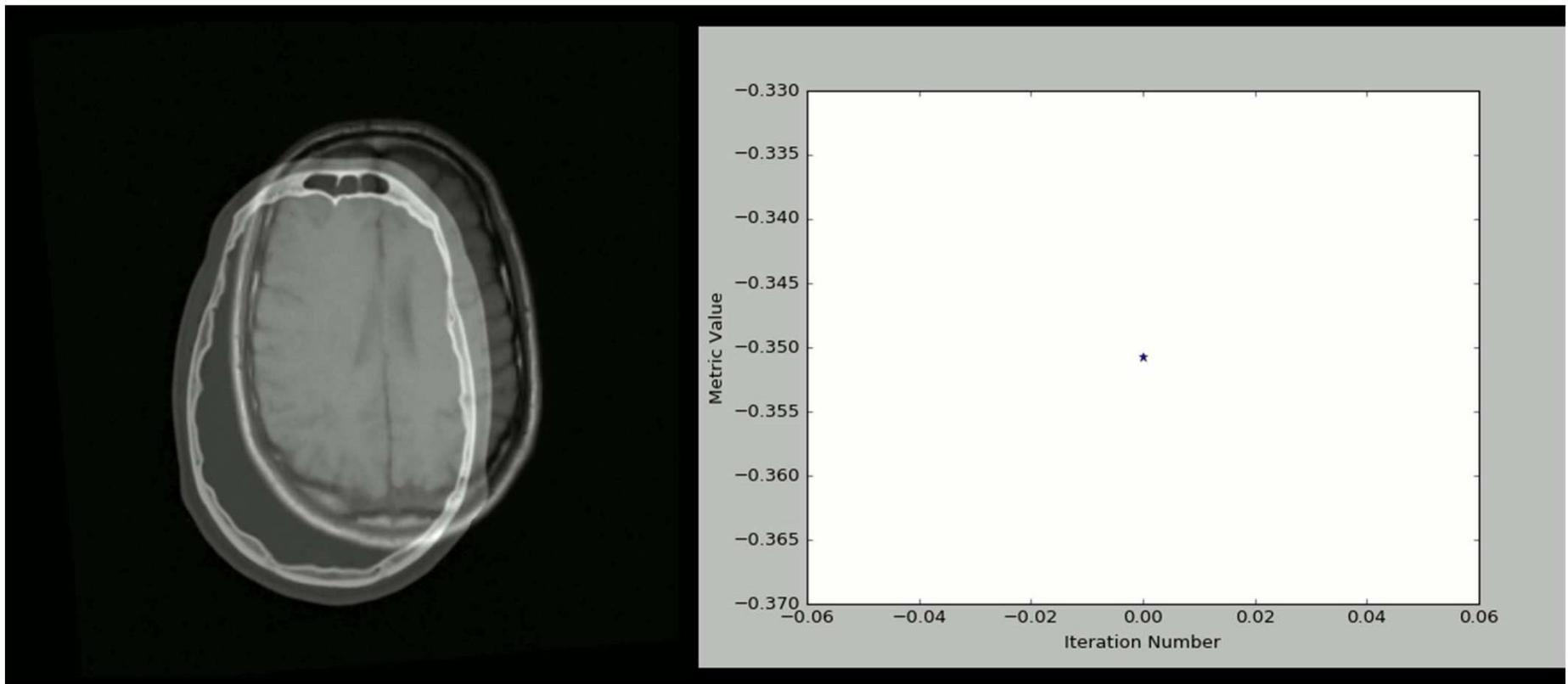
- Pustaka Mahotas memiliki kode yang sederhana untuk menyelesaikan masalah.
- Misalnya, pada masalah *Finding Wally*



SimpleITK

- ITK (*Insight Segmentation and Registration Toolkit*) sistem lintas platform yang menyediakan pengembang dengan seperangkat alat perangkat lunak yang luas untuk analisis citra.
- SimpleITK adalah lapisan sederhana yang dibangun di atas ITK, dimaksudkan untuk memfasilitasi penggunaannya prototyping dengan cepat, pendidikan, *interpreted language*.
- Juga merupakan alat analisis gambar dengan banyak komponen yang mendukung operasi filtering, segmentasi citra, dan registrasi.
- SimpleITK ditulis dalam C++, tetapi bisa juga untuk bahasa pemrograman lainnya termasuk Python.

Contoh hasil aplikasi yang menggunakan SimpleITK



pgmagick

- pgmagick adalah pembungkus berbasis-Python untuk pustaka GraphicsMagick.
- Sistem pengolahan citra GraphicsMagick disebut juga *Swiss Army Knife* untuk pengolahan citra.
- Koleksi alat dan pustaka yang handal dan efisien mendukung baca, tulis, dan memanipulasi gambar lebih dari 88 format termasuk DPX, GIF, JPEG, JPEG-2000, PNG, PDF, PNM, dan TIFF.
- pgmagick dapat digunakan untuk *Image scaling* dan ekstraksi tepi

Contoh hasil aplikasi yang menggunakan pgmagick



Image Scalling



Edge extraction

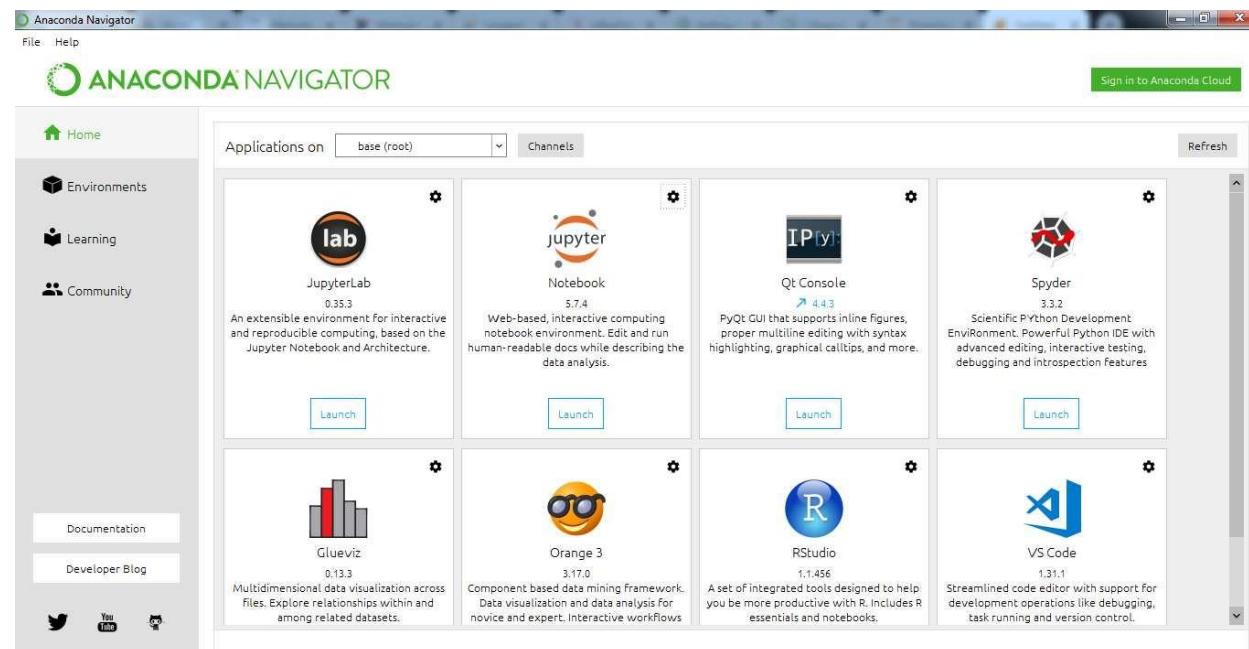


Pycairo

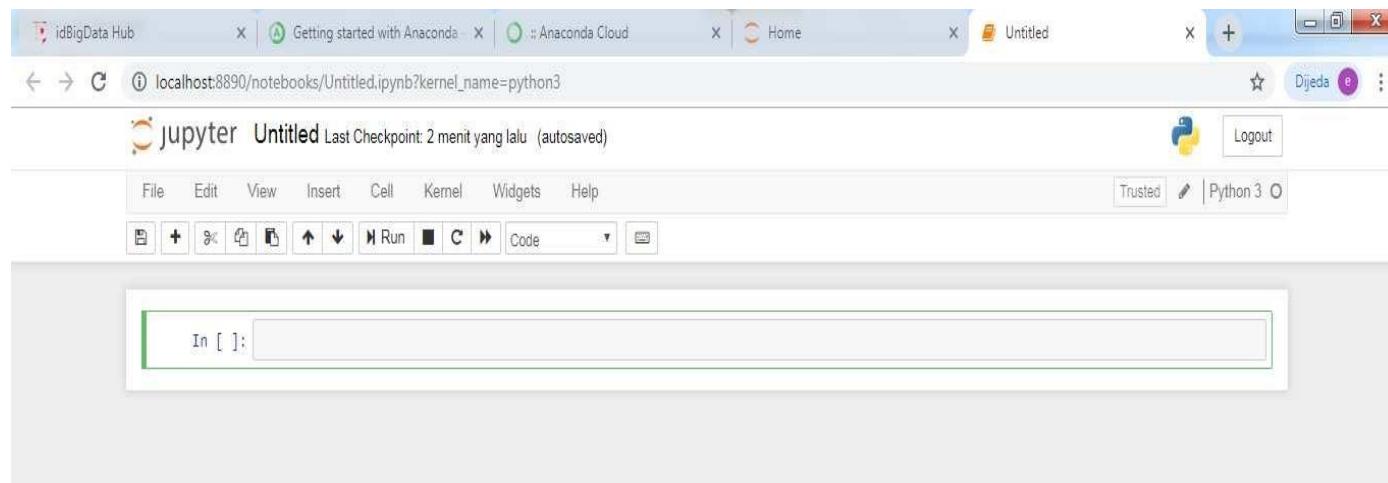
- Pycairo adalah satu set binding Python untuk pustaka grafis Cairo.
- Cairo adalah pustakaan grafis 2D untuk menggambar grafik vektor.
- Pycairo dapat memanggil perintah Cairo dari Python.

Praktikum

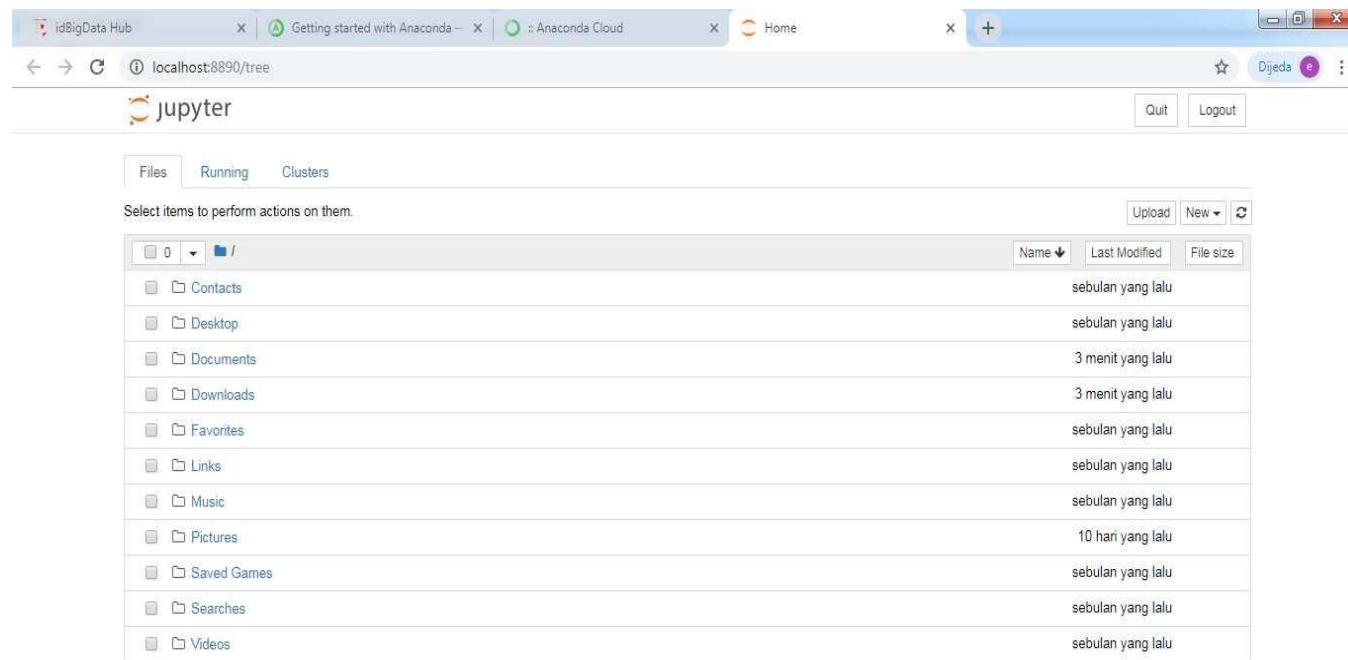
- Install Phyton dan Anaconda sesuai spesifikasi komputer
- Jika sudah selesai menginstal, Dari tombol Start pilih All Programs → Anaconda Navigator lalu akan tampil seperti gambar berikut, pilih jupyter dengan klik Launch



- Kemudian tampil notebook (area untuk mengetik program):



- Maka akan tampil gambar berikut di browser:



- Memulai program pilih New → Python 3

- Ketik program berikut

```
In [1]: from PIL import Image

def main():
    try:
        #Relative Path
        img = Image.open("picture.jpg")

        #Angle given
        img = img.rotate(180)

        #Saved in the same relative location
        img.save("rotated_picture.jpg")
    except IOError:
        pass

if __name__ == "__main__":
    main()
```

In []:

- Untuk menjalankan klik Run atau shortcut Ctrl+Enter
- Tiap baris program yang dijalankan di sebelah kiri ada keterangan In [], jika dijalankan maka muncul nomor baris, misal In [1].
- Hasil akan tampil dengan keterangan Out[] pada bagian kiri.
- Jika muncul tanda In[*] artinya kernel sedang sibuk mengeksekusi kode lain

- Program dapat diedit, lalu dijalankan kembali, dan nomor baris akan berubah yang menunjukkan running ke berapa yang telah dilakukan terhadap baris-baris kode tersebut.
- Batas program pada python ditandai dengan posisi penulisan kode (tab). Biasanya posisi otomatis, dan akan berpengaruh pada hasil, tapi dapat diatur
- Komentar ada dua macam:
 - Komentar 1 baris menggunakan tanda #
 - Komentar 1 baris menggunakan tanda "" di awal dan di akhir baris komentar
- Menghapus baris kosong Esc+D+D (tombol escape dan huruf D dua kali)
- Untuk menu-menu shortcut bisa dilihat di tombol help → Keyboard Shortcuts



Latihan: cropped dengan Pillow

```
from PIL import Image

def main():
    try:
        #Relative Path
        img = Image.open("D:Koala.jpg")
        width, height = img.size

        area = (0, 0, width/2, height/2)
        img = img.crop(area)

        #Saved in the same relative location
        img.save("D:cropped_Koala.jpg")

    except IOError:
        pass

if __name__ == "__main__":
    main()
```



Latihan: *scikit-image* untuk *image filtering*

```
from skimage import  
data, filters image =  
data.coins()  
#atau NumPy array lainnya!  
edges = filters.sobel(image)  
plt.imshow(edges,  
cmap='gray')
```



Latihan: interpolasi dengan NumPy

```
import matplotlib.pyplot as plt
import numpy as np

methods = [None, 'none', 'nearest', 'bilinear', 'bicubic', 'spline16',
           'spline36', 'hanning', 'hamming', 'hermite', 'kaiser', 'quadric',
           'catrom', 'gaussian', 'bessel', 'mitchell', 'sinc', 'lanczos']

# Fixing random state for reproducibility
np.random.seed(19680801)

grid = np.random.rand(4, 4)

fig, axs = plt.subplots(nrows=3, ncols=6, figsize=(9, 6),
                       subplot_kw={'xticks': [], 'yticks': []})

for ax, interp_method in zip(axs.flat, methods):
    ax.imshow(grid, interpolation=interp_method, cmap='viridis')
    ax.set_title(str(interp_method))

plt.tight_layout()
plt.show()
```

Sumber:

- <https://opensource.com/article/19/3/python-image-manipulation-tools>
- <http://www.jendelastatistik.com/2017/05/belajar-dasar-python-dengan-tools.html>
- <https://medium.com/@renopp/perjalanan-belajar-deep-learning-introduction-dan-setup-environment-part-1-afc6d731960f>



Pengenalan Pengolahan Citra

Pengertian Citra

- **Citra (Image)** adalah istilah lain untuk gambar komponen multimedia yang memegang peranan sangat penting sebagai salah satu bentuk informasi visual
- Secara harfiah, **citra (image)** adalah gambar pada bidang dwimatra (dua dimensi).

Pembentukan Citra

- **Citra ada 2 macam :**

- **Citra Kontinu**

Dihasilkan dari sistem optik yang menerima sinyal analog. Contoh : Mata manusia, kamera analog

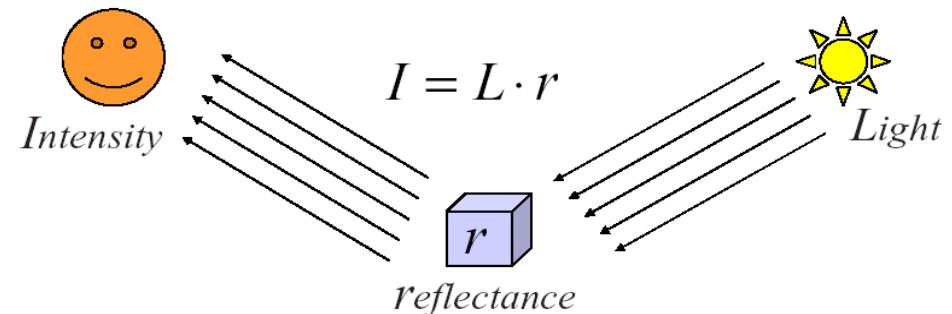
- **Citra Diskrit**

Dihasilkan melalui proses digitalisasi terhadap citra continue. Contoh : Kamera digital, scanner

Pengertian Citra

- Citra adalah gambar dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinus menjadi gambar diskrit melalui proses sampling (Wikepedia, 2006).
- Gambar analog dibagi menjadi N baris dan M kolom sehingga menjadi gambar diskrit.

- Citra merupakan fungsi malar (kontinyu) dari intensitas cahaya. Secara matematis, fungsi intensitas cahaya pada bidang 2D disimbolkan dengan $f(x,y)$, dimana:
 - **(x,y)**: koordinat pada bidang dwi warna
 - **F(x,y)**: intensitas cahaya pada titik (x,y)
- Nilai $f(x,y)$ adalah hasil kali dari :
 - **i(x,y)** = jumlah cahaya yang berasal dari sumber, nilainya antara 0 sampai tak terhingga.
 - **r(x,y)** = derajat kemampuan objek memantulkan cahaya , nilainya antara 0 dan 1.
 - Jadi **$f(x,y) = i(x,y) \cdot r(x,y)$**

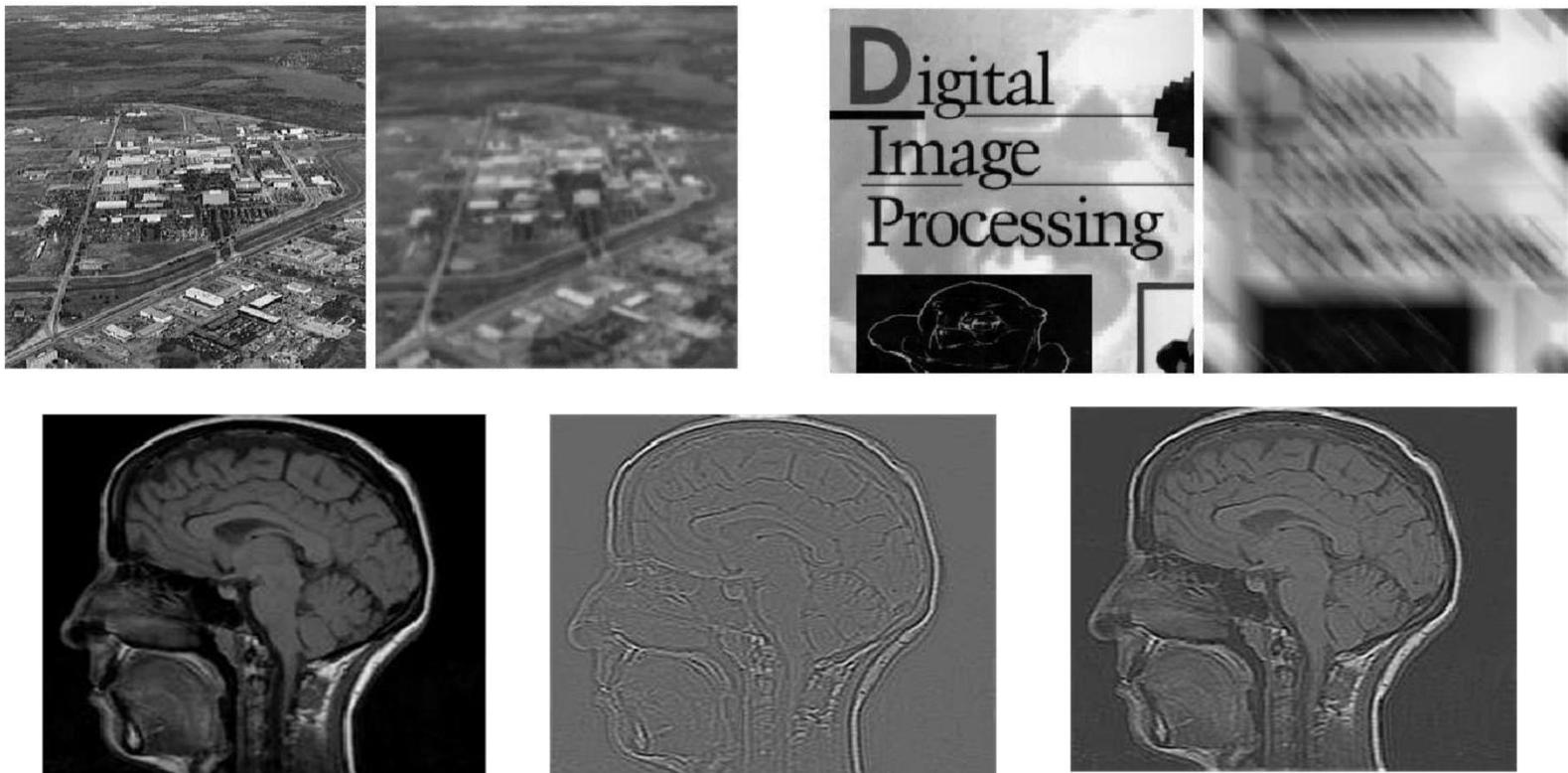


Tujuan Pengolahan Citra

- Memperbaiki kualitas gambar dilihat dari :
 - **Aspek Radiometrik :**
 - Peningkatan kontras, transformasi warna, restorasi citra
 - **Aspek Geometrik**
 - Rotasi, translasi, skala, transformasi geometrik
- Melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terdapat pada citra
- Melakukan kompresi atau reduksi data untuk tujuan penyimpanan data, transmisi data dan waktu proses data

Aplikasi

- Ada 2 area aplikasi dari digital image processing:
 - Perbaikan kualitas image untuk interpretasi manusia
 - Pemrosesan image untuk persepsi mesin secara otomatis



Digital vs Analog

- Data digital direpresentasikan dalam komputer berbentuk kode seperti biner, decimal. Contoh data digital : WAV, MP3, RMI, BMP, JPG, GIF, TIF
- Data analog tidak direpresentasikan dalam komputer, semua merupakan fakta, contoh : gelombang suara, gambar. Data analog tersimpan dalam pita kaset.

Citra Digital

- Citra digital merupakan suatu array 2 dimensi yang elemennya menyatakan tingkat keabuan keabuan dari elemen gambar.
- Citra yang dihasilkan direkam datanya bersifat kontinue harus dirubah dahulu menjadi citra digital dengan konversi agar dikenali komputer.
- Proses tersebut disebut digitalisasi, yaitu membuat kisi-kisi arah horizontal dan vertical sehingga terbentuk array 2 dimensi.

Model Citra

□ **Citra Hitam-Putih**

- Citra monokrom (*monochrome image*) atau citra satu fungsi intensitas

□ **Citra Berwarna**

- Citra spektral, karena warna pada citra disusun oleh tiga komponen warna RGB (*Red-Green-Blue*)
- Intensitas suatu titik pada citra berwarna merupakan kombinasi dari intensitas merah ,hijau dan biru *terletak antara hitam dan putih.*

Derajat Keabuan

- Derajat keabuan adalah intensitas citra hitam-putih pada titik (x,y)
- Derajat keabuan bergerak dari hitam ke putih
- Skala keabuan : $[0,1]$, dimana intensitas 0 menyatakan hitam dan 1 menyatakan putih
- Contoh :Citra Hitam Putih dengan 256 level, artinya :
 - Derajat keabuan : $0 - 255$ atau $[0,255]$
 - 0 menyatakan hitam, dan 255 menyatakan putih
 - Nilai antara 0-255: menyatakan warna keabuan yang terletak antara hitam dan putih.

Jenis Citra

- **Citra Diam (*Stil Images*) adalah**
 - citra tunggal yang tidak bergerak
- **Citra bergerak(*Moving Images*), adalah**
 - rangkaian citra diam yang ditampilkan secara **beruntun (*sequential*)**, sehingga memberikesan pada mata kita sebagai **gambar bergerak**.
 - Setiap citra dalam rangkaian disebut *frame*.
 - Gambar-gambar pada film atau TV terdiri dari ratusan sampai ribuan *frame*.

Elemen Dasar Citra Digital

□ **Kecerahan (*Brightness*)**

- Intensitas cahaya rata-rata dari suatu area yang melingkupinya

□ **Kontras (*contrast*)**

- Sebaran terang (*lightness*) dan gelap (*darkness*) di dalam sebuah citra
- Citra dengan **kontras rendah**, komposisi citranya sebagian besar terang atau sebagian besar gelap
- Citra dengan **kontras baik**, komposisi gelap dan terangnya tersebar merata

Elemen Dasar Citra Digital

□ **Kontur (*Contour*)**

- Keadaan yang ditimbulkan oleh perubahan intensitas pada pixel-pixel tetangga, sehingga kita dapat mendeteksi tepi objek didalam citra

□ **Warna (*colour*)**

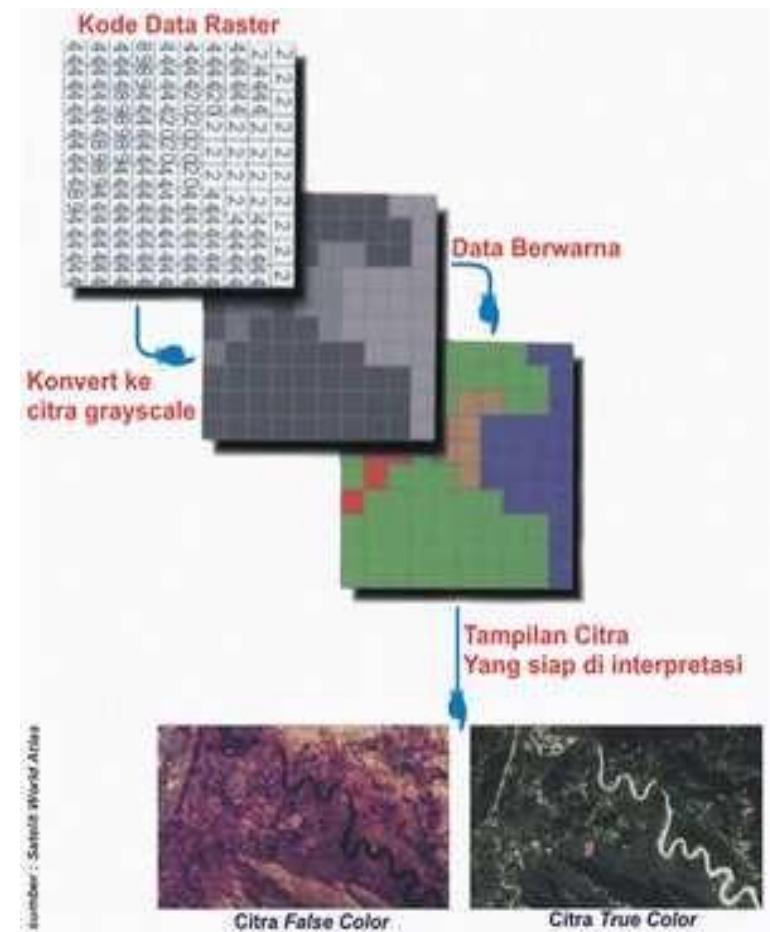
- Persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek
- Warna merupakan kombinasi cahaya dengan panjang berbeda, yaitu Red, Green, Blue

□ **Bentuk (*Shape*)**

- Umumnya citra yang dibentuk oleh manusia → **2D**
- Objek yang dilihat → **3D**

Digitalisasi Citra

- Supaya bisa diolah dengan komputer, citra harus direpresentasikan secara numerik dengan nilai diskrit .
- Citra digital dinyatakan dengan suatu **matrik ukuran $N \times M$** . Masing-masing elemen disebut ***pixel (picture element)***
- Umumnya citra dibentuk dari kotak-kotak persegi empat yang teratur sehingga jarak horizontal dan vertikal antara piksel adalah sama pada seluruh bagian citra



Digitalisasi Citra

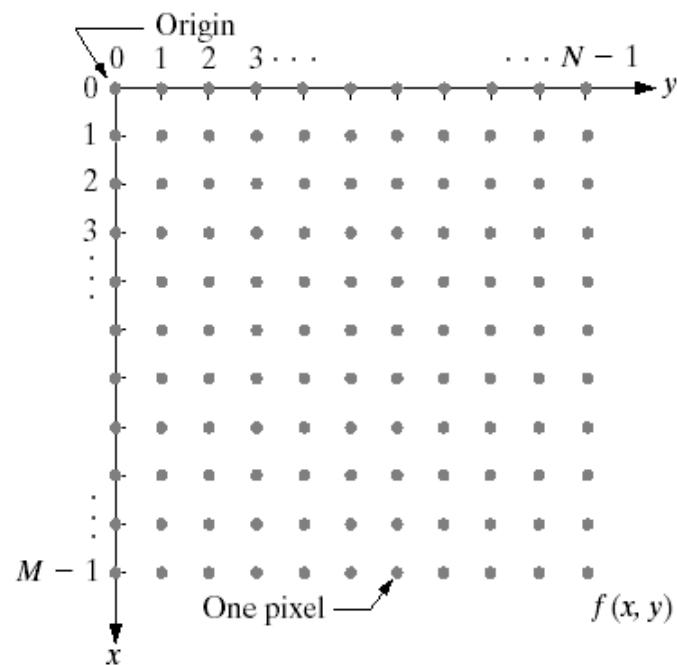
□ Contoh

- Suatu Citra berukuran 256×256 pixel dengan intensitas beragam pada tiap pixelnya, direpresentasikan secara numerik dengan matrik terdiri dari 256 baris dan 256 kolom

0	134	145	231
0	167	201	197
220	187	189	120
:	:	:	:	:	:
:	:	:	:	:	:
221	219	210	156

Digitalisasi Citra

□ Aturan Koordinat



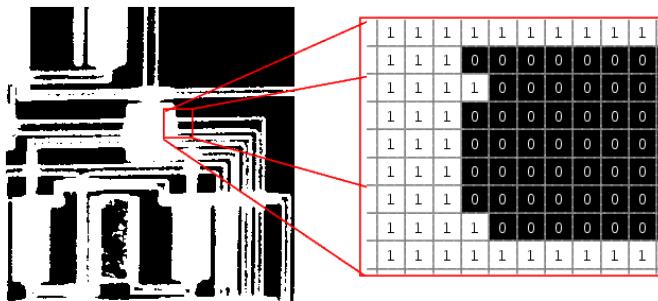
Aturan koordinat representasi citra digital
(sumber : Gonzalez dan Woods, 2008)

Digitalisasi Citra

$$f(x,y) = \begin{pmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \dots & \dots & & \dots \\ \dots & \dots & & \dots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{pmatrix}$$

- Indeks **baris (*i*)** dan indeks **kolom (*j*)** menyatakan **koordinat titik pada citra**, sedang ***f(i,j)*** merupakan **intensitas (derajat keabuan) pada titik (*i,j*)**

Digitalisasi Citra



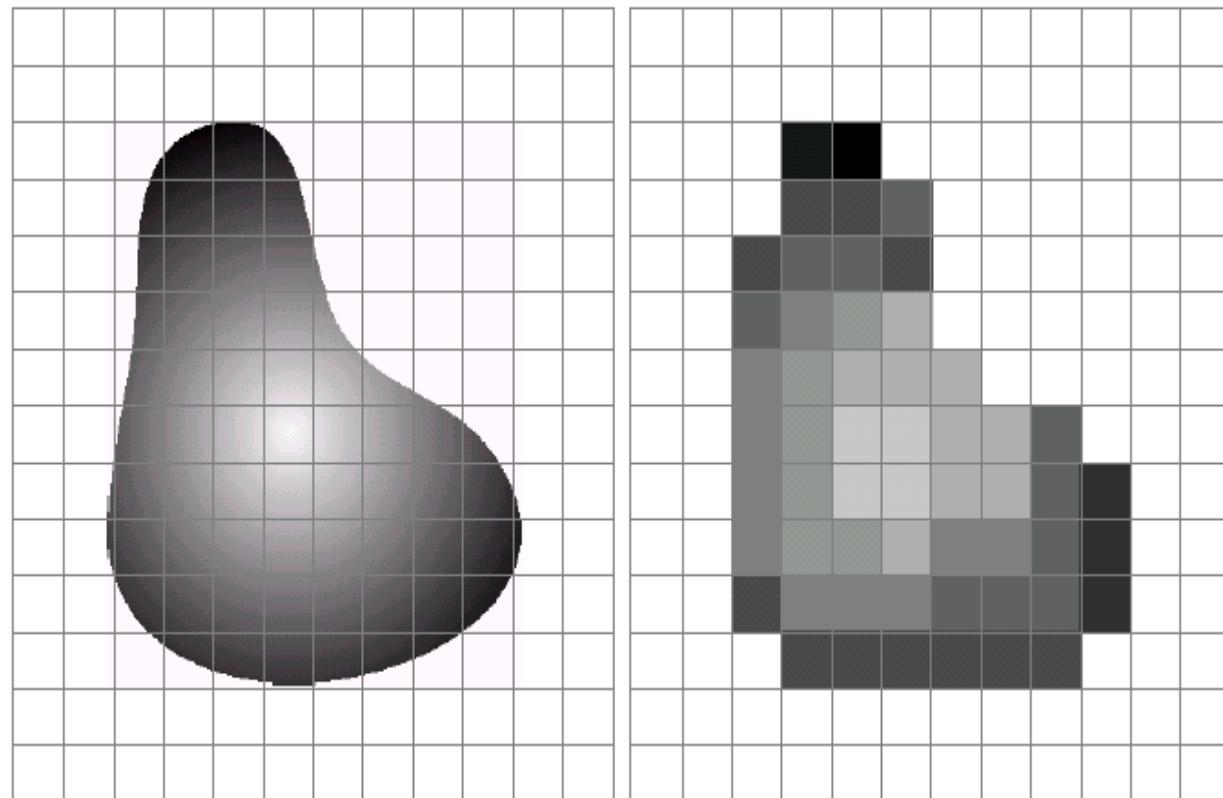
	0.2235	0.1294	Blue	0.4196
0.5804	0.2902	0.0627	0.2902	0.2902
0.5804	0.0627	0.0627	0.0627	0.2235
0.5176	0.1922	0.0627	Green	0.1922
0.5176	0.1294	0.1608	0.1294	0.1294
0.490	0.2588	0.2902	0.2588	0.2588
0.5176	0.1608	0.0627	0.1608	0.1922
0.5490	0.2235	0.5490	Red	0.7412
0.5490	0.3882	0.5176	0.5804	0.7765
0.490	0.2588	0.2902	0.2588	0.7765
0.5176	0.2235	0.1608	0.2588	0.2235
0.5490	0.1608	0.2588	0.1608	0.2588
0.5804	0.2588	0.2588	0.2588	0.2588



0.2251	0.2563	0.2826	0.2826	0.4
0.5342	0.2051	0.2157	0.2826	0.3822
0.5342	0.1789	0.1307	0.1789	0.2051
0.4308	0.2483	0.2624	0.3344	0.3344
0.3344	0.2624	0.3344	0.3344	0.33



Digitalisasi Citra



a b

FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Digitalisasi Citra

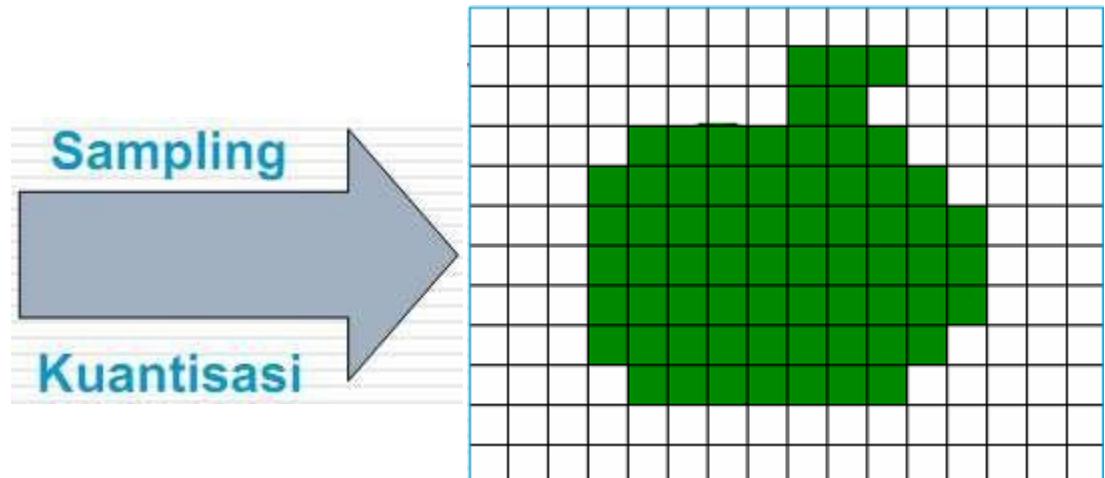
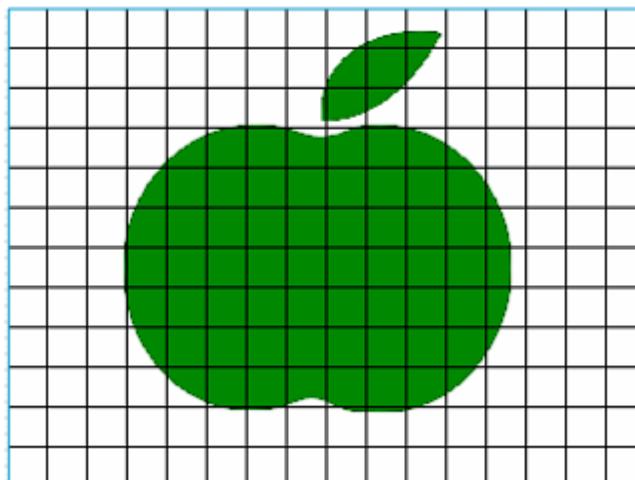
□ **Skala/Derajat Keabuan (G)**

- $G = 2^n$
- G = derajat keabuan
- n = bilangan bulat positif

Skala Keabuan	Nilai Keabuan	Pixel Depth (jumlah bit perpixel)
2^1 (2 nilai)	0 dan 1	1 bit
2^2 (4 nilai)	0 sampai 3	2 bit
2^4 (16 nilai)	0 sampai 15	4 bit
2^8 (256 nilai)	0 sampai 255	8 bit

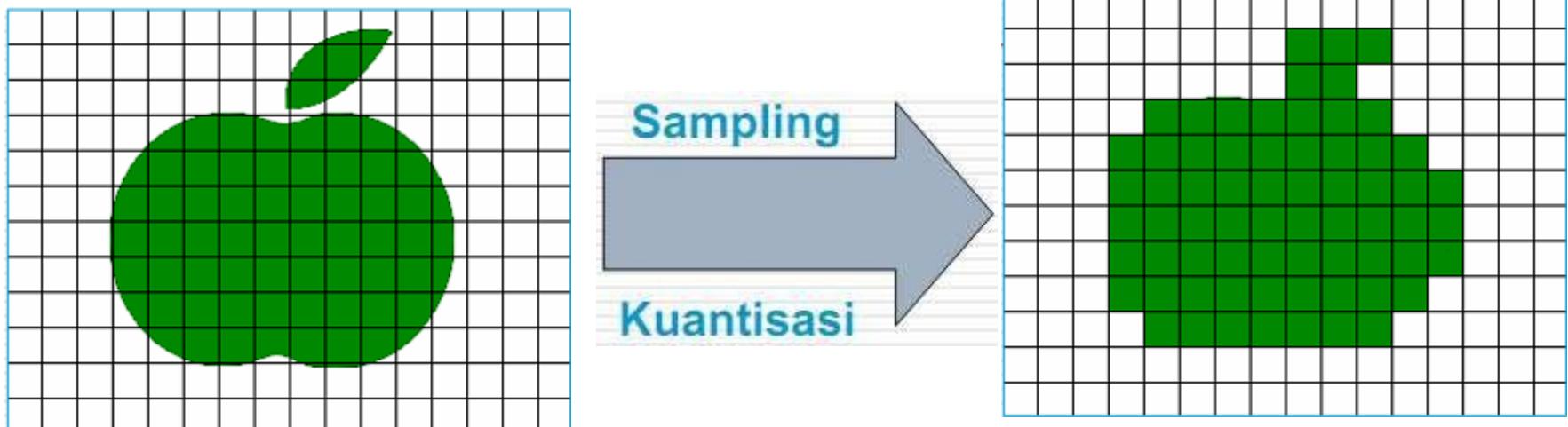
- Penyimpanan citra digital **NxM pixel** dan dikuantisasi menjadi **G = 2^n** memerlukan memori sebanyak : **NxM x n = $512 \times 512 \times 8 = 2.048.000$ bit**

SAMPLING DAN KUANTISASI



- Citra harus mengalami sampling dan kuantisasi agar dapat diproses dengan komputer yang bersifat diskrit
- ***Sampling***
 - adalah proses mapping fungsi kontinyu ke diskrit
 - Menunjukkan banyaknya pixel (blok) untuk mendefinisikan suatu gambar

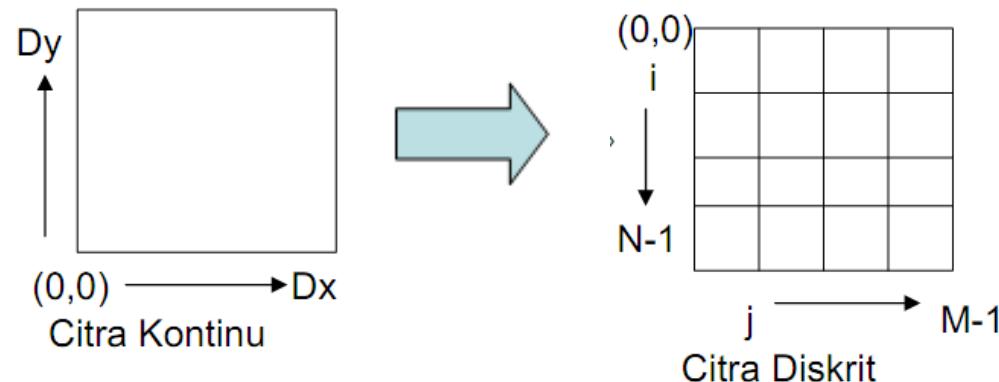
SAMPLING DAN KUANTISASI



- **Kuantisasi**
 - adalah proses mapping variabel kontinyu ke diskrit
 - Menunjukkan banyaknya derajat nilai pada setiap pixel (menunjukkan jumlah bit pada gambar digital → b/w dengan 2 bit, grayscale dengan 8 bit, true color dengan 24 bit)

Sampling (1/2)

- **Sampling** : digitalisasi spasial (x,y)
- Citra kontinu disampling pada grid-grid yang berbentuk bujursangkar (kisi-kisi arah horizontal dan vertikal)



Sampling (2/2)

- Pembagian gambar menjadi ukuran tertentu menentukan **RESOLUSI** (derajat rincian yang dapat dilihat) spasial yang diperoleh
- Semakin tinggi resolusinya semakin kecil ukuran pixel atau semakin halus gambar yang diperoleh karena informasi yang hilang semakin kecil

Kuantisasi (1/2)

- Kuantisasi : pembagian skala keabuan (0,L) menjadi G Level;
- $G = 2^m$;
- **Hitam dinyatakan** dengan nilai derajat keabuan terendah
- **Putih dinyatakan** dengan nilai derajat keabuan tertinggi, misal 15 untuk 16 level

Kuantisasi (2/2)

- **Besarnya derajat keabuan digunakan :** untuk menentukan resolusi kecerahan dari citra yang diperoleh
- **Semakin banyak jumlah derajat keabuan** (jumlah bit kuantisasinya makin banyak), semakin bagus gambar yang diperoleh karena kuantitasi derajat keabuan akan semakin tinggi sehingga mendekati citra aslinya

Hasil Sampling dan Kuantisasi

$$f(x,y) = \begin{pmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \dots & \dots & & \dots \\ \dots & \dots & & \dots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{pmatrix}$$

Ukuran spatial (=resolusi) adalah hasil sampling
Color depth (=max warna) adalah hasil quantization

Contoh Perbedaan Spatial Resolution



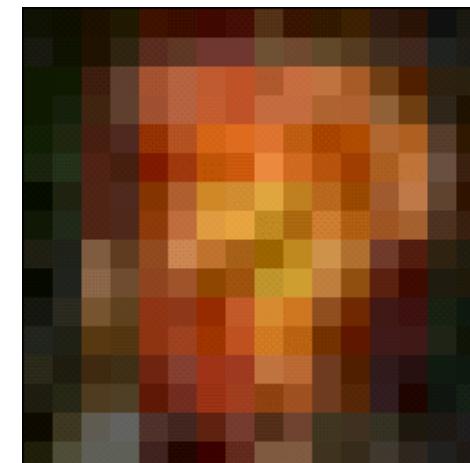
256 x 256



128 x 128



64x64



16x16

Contoh Perbedaan Color Depth



24 bits



256 warna



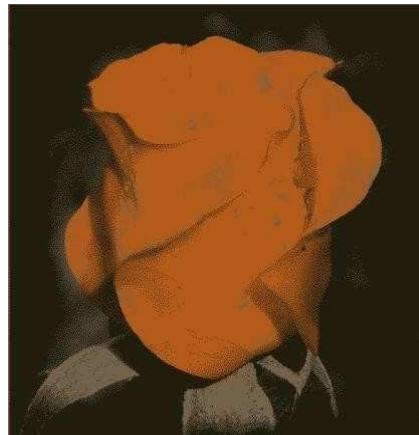
64 warna



grayscale



16 warna

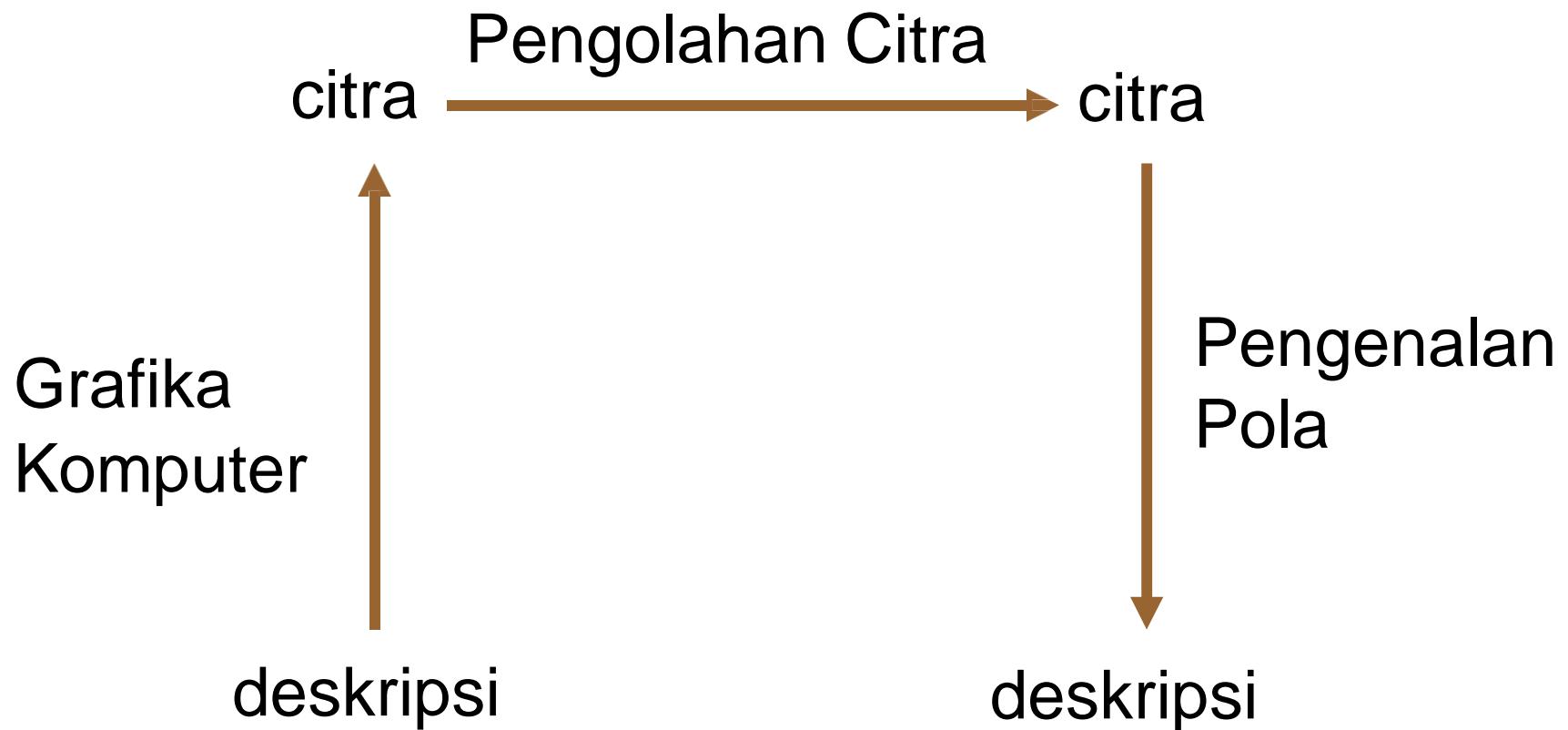


4 warna



bitmap

Bidang studi yang terkait



Bidang studi terkait

- **Grafika Komputer (*computer graphics*),**
 - menghasilkan citra dengan primitif-primitif geometri spt: garis, lingkaran, elips dll. Hal ini penting dalam visualisasi
- **Pengolahan Citra (*image processing*),**
 - memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau komputer
- **Pengenalan Pola (*pattern recognition*),**
 - mengelompokkan data numerik dan simbolik untuk mengenali suatu objek di dalam citra

Bidang tambahan

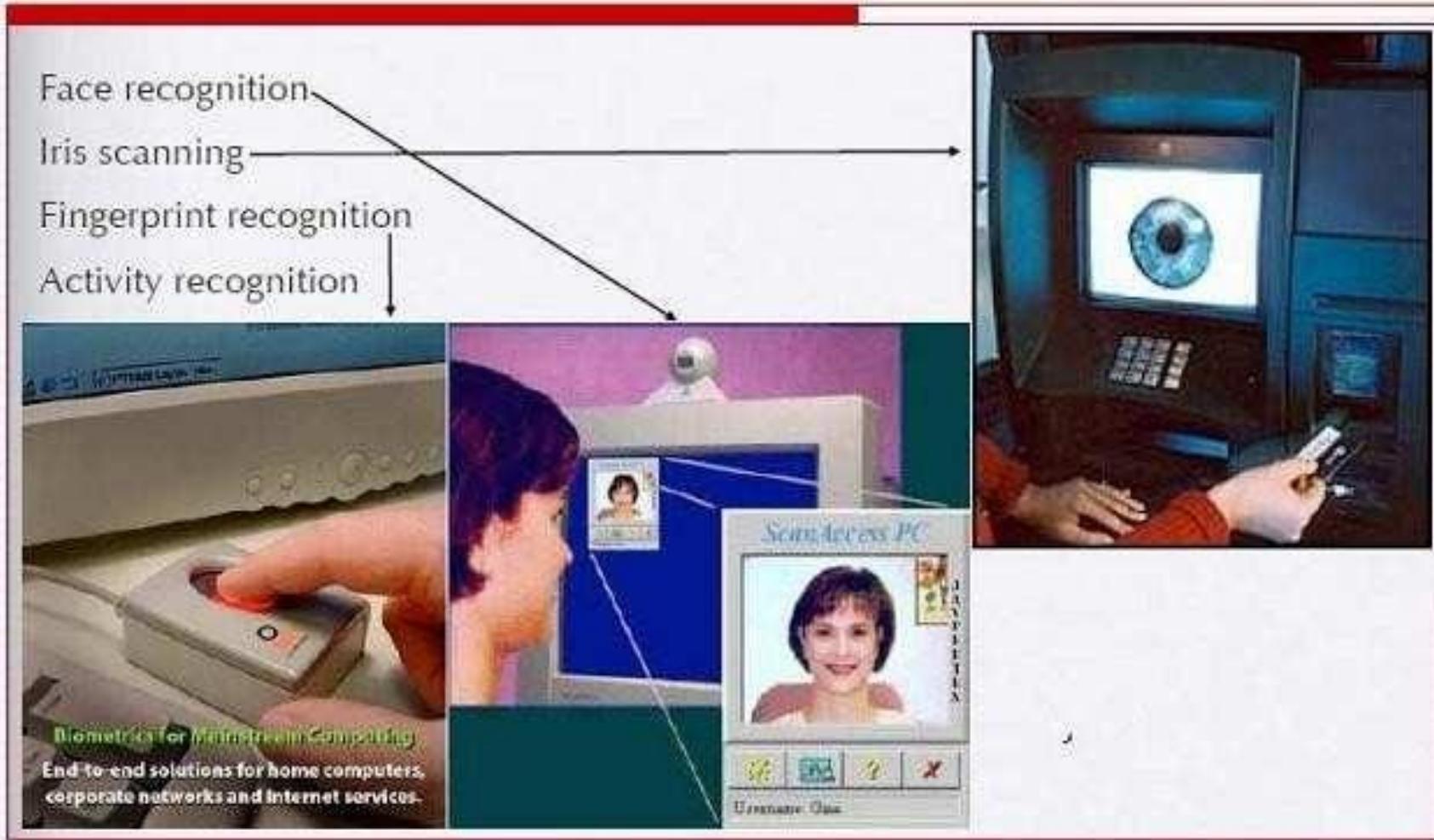
- **Artificial intelligence (Kecerdasan Buatan),**
 - **menganalisis pemandangan dalam citra**
dengan perhitungan simbol-simbol yang mewakili isi pemandangan tsb setelah citra diolah untuk memperoleh ciri khas
- **Artificial neural network**
 - mengolah berbagai data yang dihasilkan oleh sistem visual dalam upaya pengambilan keputusan yang tepat berdasarkan data-data
- **Psychophysics**
 - sistem visual manusia dalam bidang kedokteran dan fisika



Aplikasi Image Processing

- Biometric
- Medical Image
- Image Databases
- Robot Vision
- Motion Capture
- Document Analysis

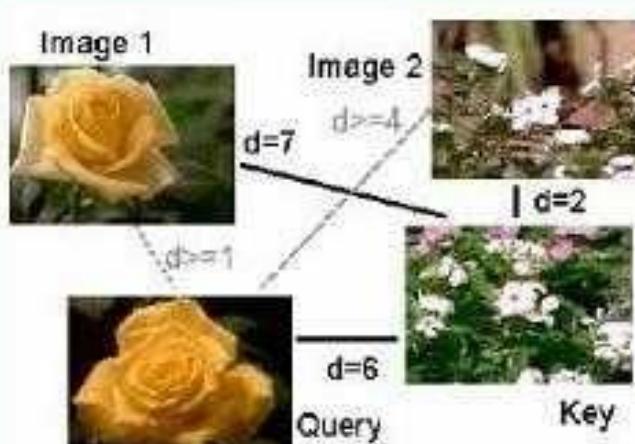
BIOMETRIC



Medical Image



Image Database



The diagram illustrates the image retrieval process. A "Query" image of a yellow rose is compared against two database images. The distance between the Query and Image 1 is labeled $d=7$. The distance between the Query and Image 2 is labeled $d \geq 4$, and the distance between Image 1 and Image 2 is labeled $d=2$.

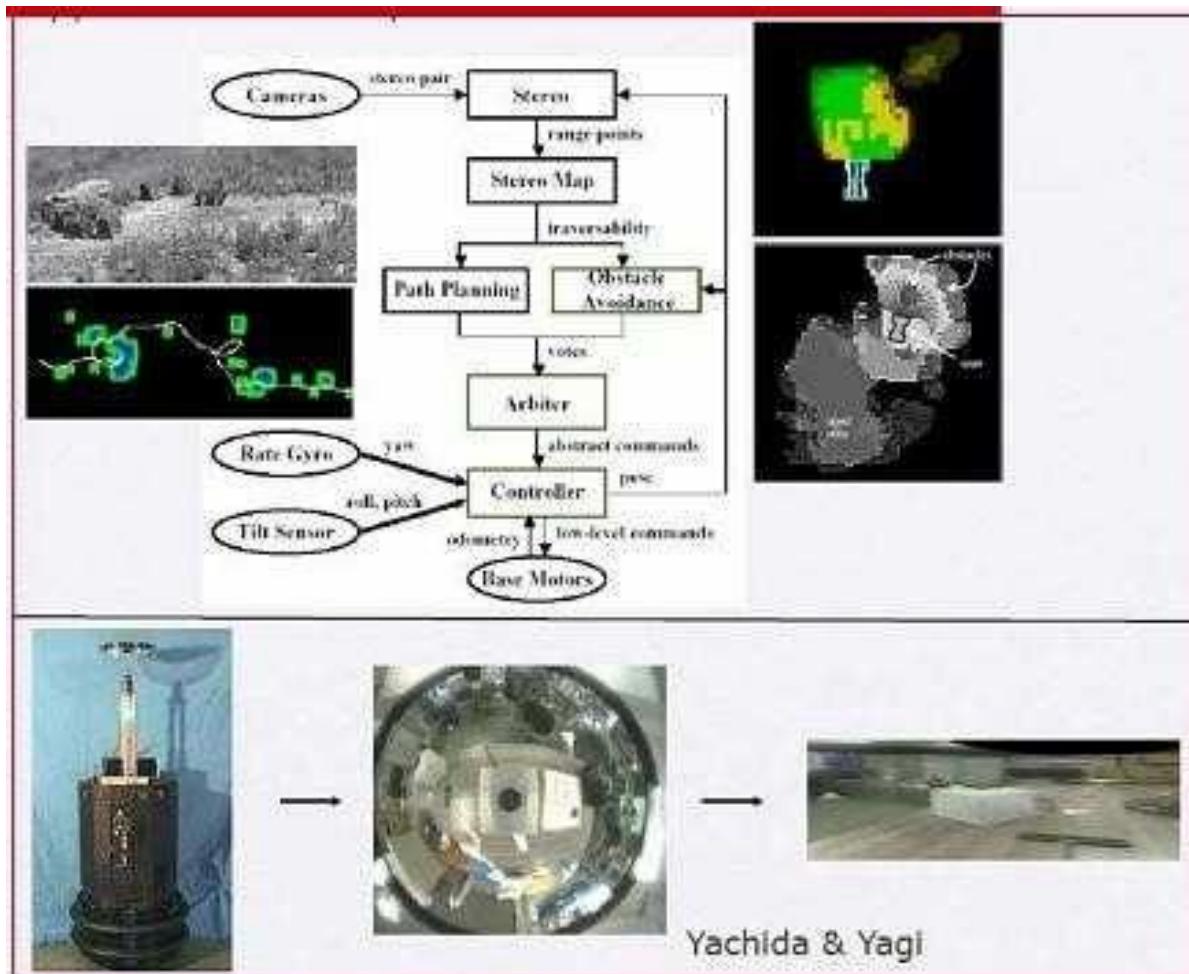
Image retrieval

From a search for horse pix in 100 horse images and 1086 non-horse images.



Forsyth & Ponce

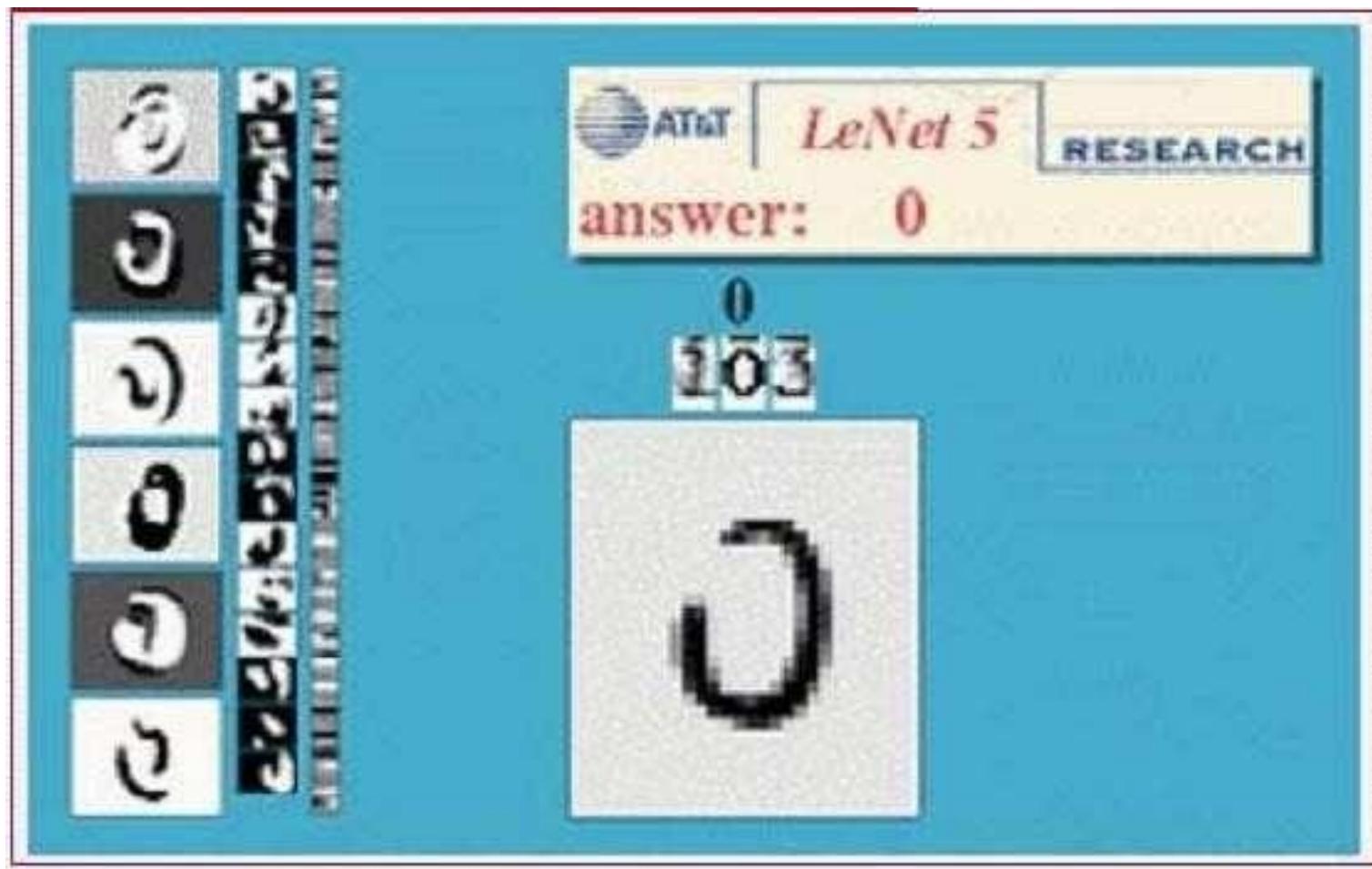
Robot Vision



Motion Capture



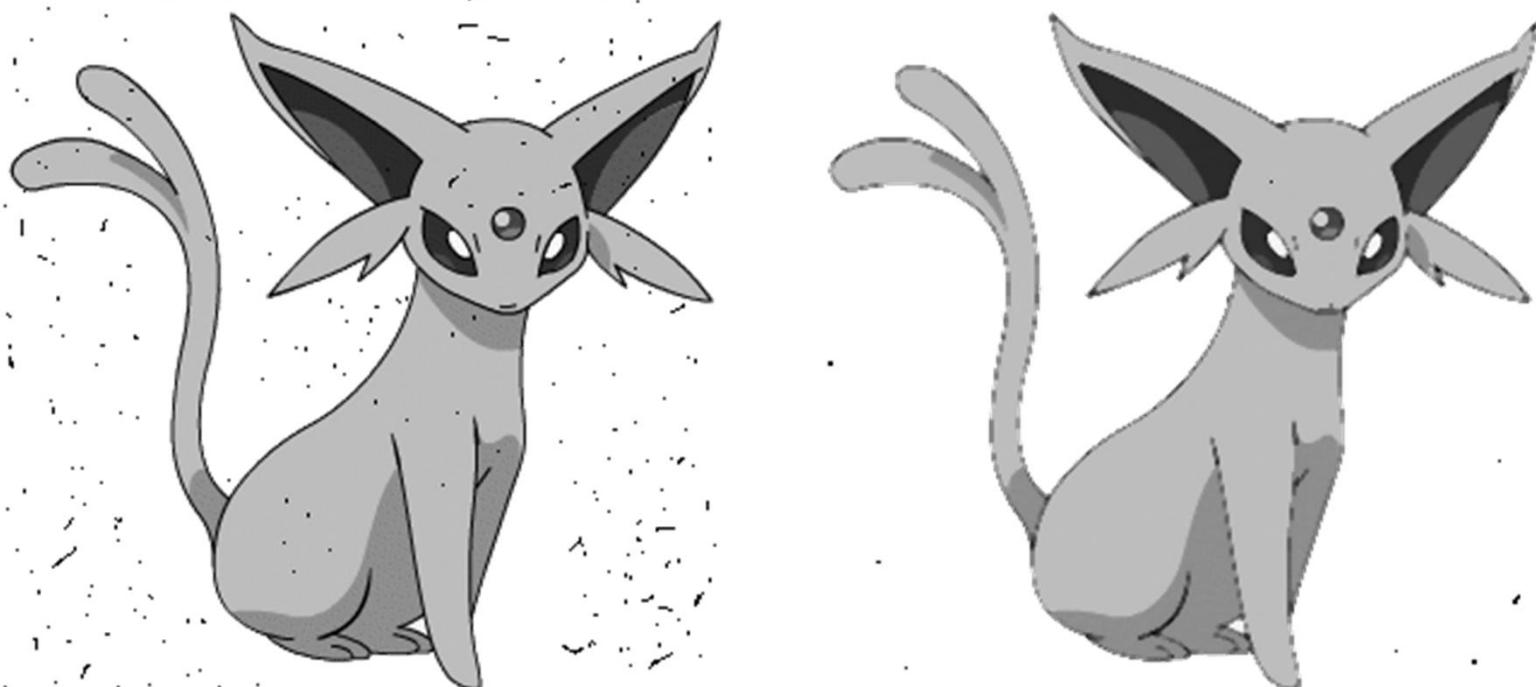
Document Analysis

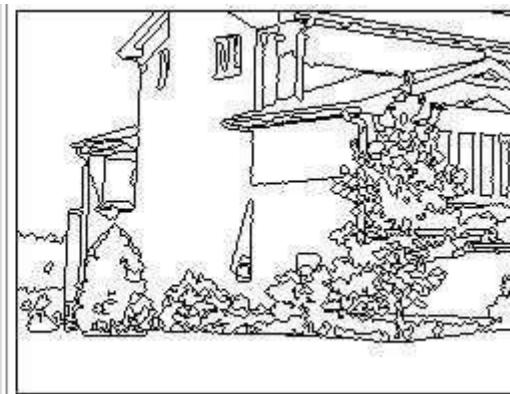




Operasi Pengolahan Citra

- Citra kaya informasi, seringkali mengalami penurunan mutu (degradasi):
 - a. Mengandung cacat atau derau (*noise*)
 - b. Warna terlalu kontras,
 - c. Kurangtajam,
 - d. Kabur (*bluring*)
 - e. Dsb.







Operasi Pengolahan Citra

- **Perbaikan Kualitas Citra (*Image Enhancement*)**
 - Tujuan : memperbaiki kualitas citra dengan memanipulasi parameter-parameter citra
- **Pemugaran Citra (*Image Restoration*)**
 - Tujuan : menghilangkan cacat pada citra
- **Pemampatan Citra (*Image Compression*)**
 - Tujuan : citra direpresentasikan dalam bentuk lebih kecil, sehingga keperluan memori lebih sedikit namun tetap mempertahankan kualitas gambar (mis. Dari .BMP ke .JPG)
 - Analisis dari pengolahan citra

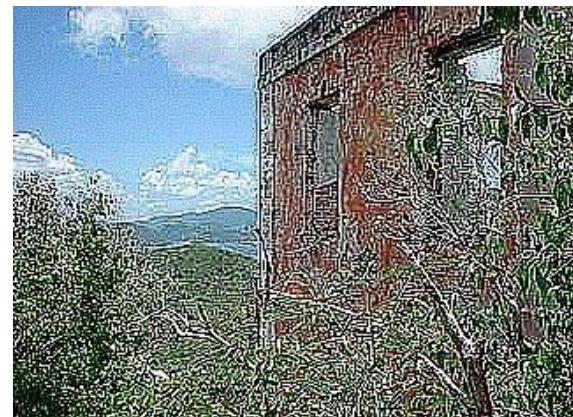
- **Segmentasi Citra (*Image Segmentation*)**
 - Tujuan : memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu
 - Berkaitan erat dengan pengenalan pola
- **Pengorakan Citra (*Image Analysis*)**
 - Tujuan : menghitung besaran kuantitatif dari citra untuk menghasilkan deskripsinya.
 - Diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya
- **Rekonstruksi Citra (*Image Reconstruction*)**
 - Tujuan : membentuk ulang objek dari beberapa citra hasil proyeksi

- **Perbaikan KualitasCitra (*ImageEnhancement*)**
 - **Tujuan:** memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra
 - **Contoh:**
 - PerbaikanKontras/Gelap
 - Perbaikan tepian objek (*Edge Enhancement*)
 - Penajaman (*Sharpening*)
 - Pemberian warna semu (*pseudocoloring*)
 - Penapisan Derau (*Noise Filtering*)

- **Perbaikan Kualitas Citra (*Image Enhancement*)**
Contoh Operasi Penajaman

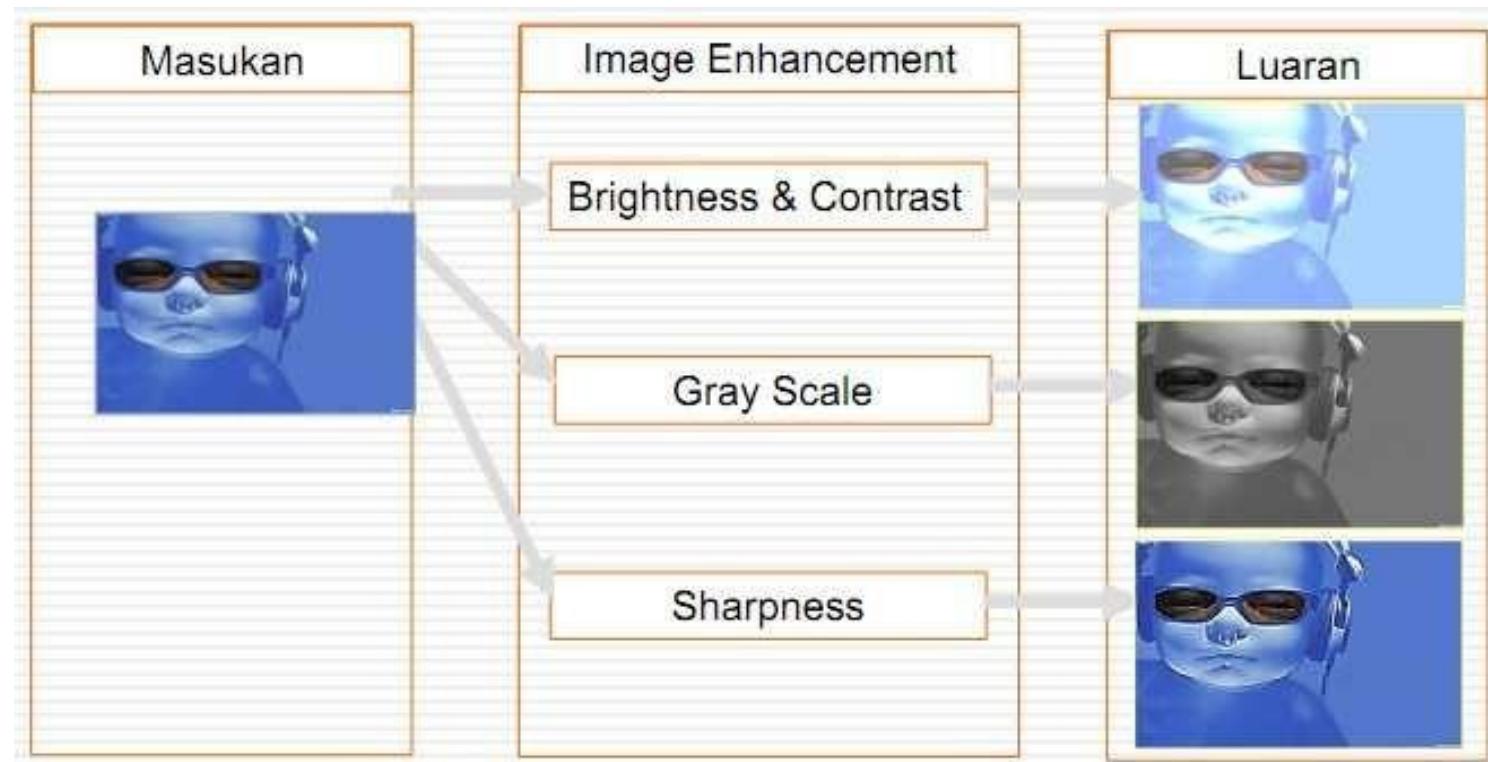


Citra asli



Citra setelah penajaman

- **Perbaikan Kualitas Citra (*Image Enhancement*)**
- Contoh:**



- **Pemugaran Citra (*Image Restoration*)**
 - **Tujuan:** menghilangkan/memminimumkan cacat pada citra
 - Pada operasi ini penyebab degradasi dapat diketahui
 - **Contoh:**
 - Penghilangan Kesamaran (*Debluring*)
 - Penghilangan Derau (*Noise*)

- **PemampatanCitra(*ImageCompre sion*)**
 - **Tujuan:** agar citra dapat direpresentasikan dalam bentuk yang lebih kompak, sehingga memori yang diperlukan sedikit
 - **Contoh:**
 - Metode Lossless (tepat sama): Huffman
 - Metode Lossy (hampir sama) : Run Length, Quantizing Compression, Pemampatan Fraktal



Citra apel mandarin.bmp (41KB)



Citra apel mandarin.jpg (3KB)

Operasi Pengolahan Citra

□ Segmentasi Citra (*Image Segmentation*)

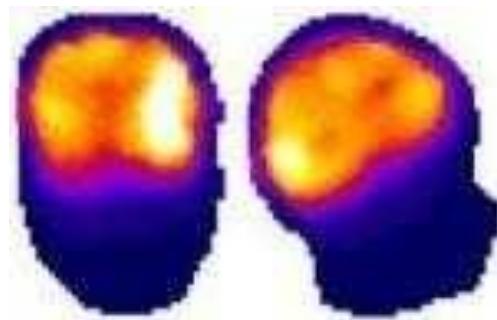
□ Tujuan:

- memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu
- Mengelompokkan gambar sesuai dengan objek gambarnya
- Jenis operasi ini berkaitan erat dengan **pengenalan pola**

□ Contoh : Aplikasi Pengenalan Pola

□ Segmentasi Citra

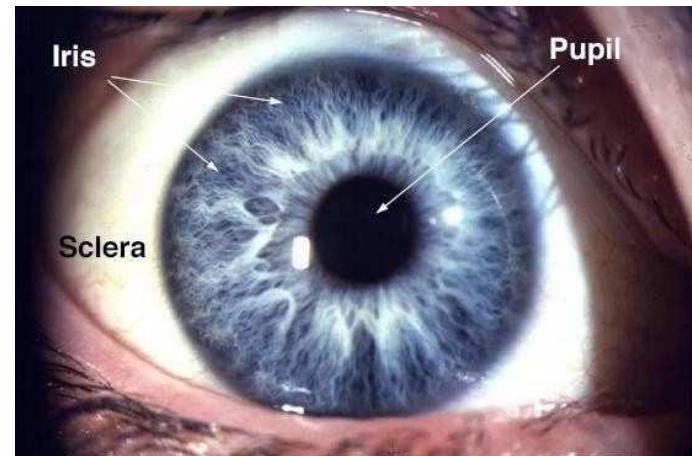
Contoh: Aplikasi Pengenalan Pola pada Bidang Kedokteran



Citra otak bayi direkam menggunakan NIRS.

□ Segmentasi Citra

Aplikasi Pengenalan Pola pada Bidang Biometrik
untuk diabetes

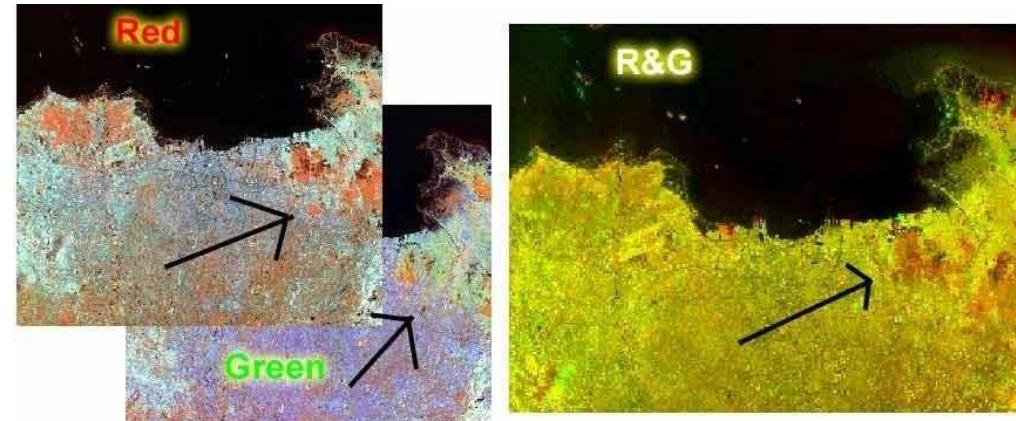


Pengenalan pola bentuk iris sebagai akses ID

□ Segmentasi Citra

Contoh:

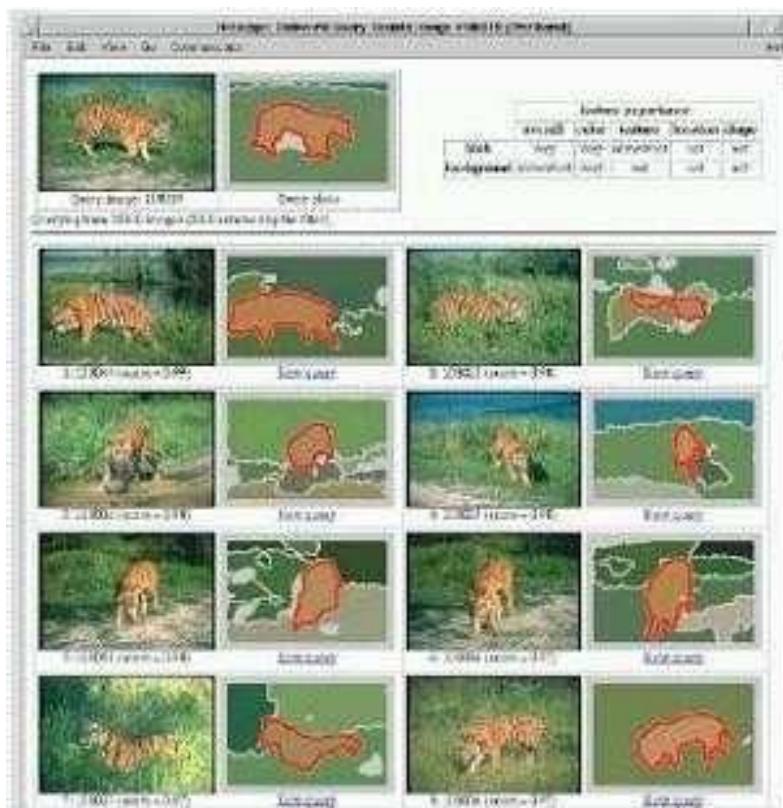
Aplikasi Pengenalan Pola pada Bidang Geologi



Aplikasi RG untuk melihat areal yang berubah (tanda panah)

□ Segmentasi Citra

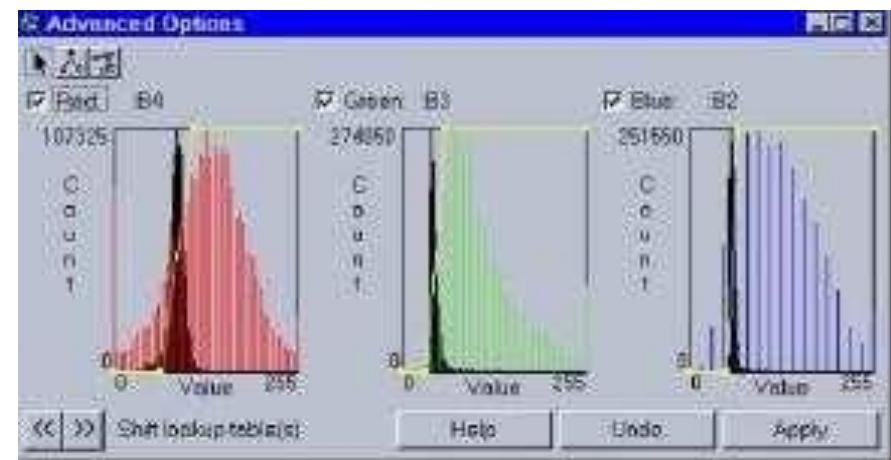
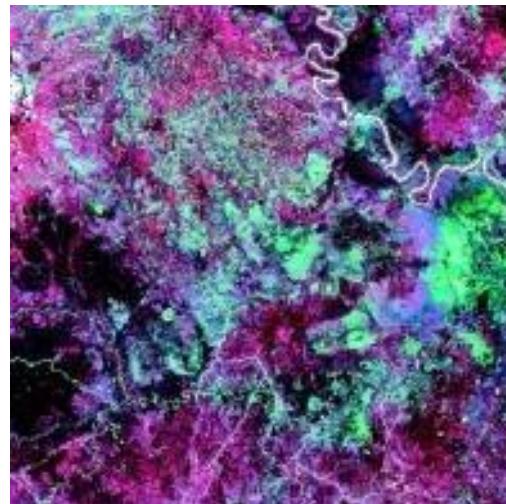
Contoh:



□ **Segmentasi Citra**

Contoh :

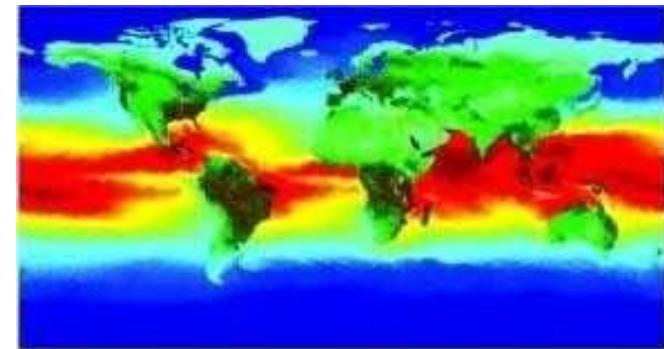
Aplikasi Pengenalan Pola pada Bidang Geologi



Citra Lansat dan histogramnya.

Pengorakan Citra (*Image Analysis*)

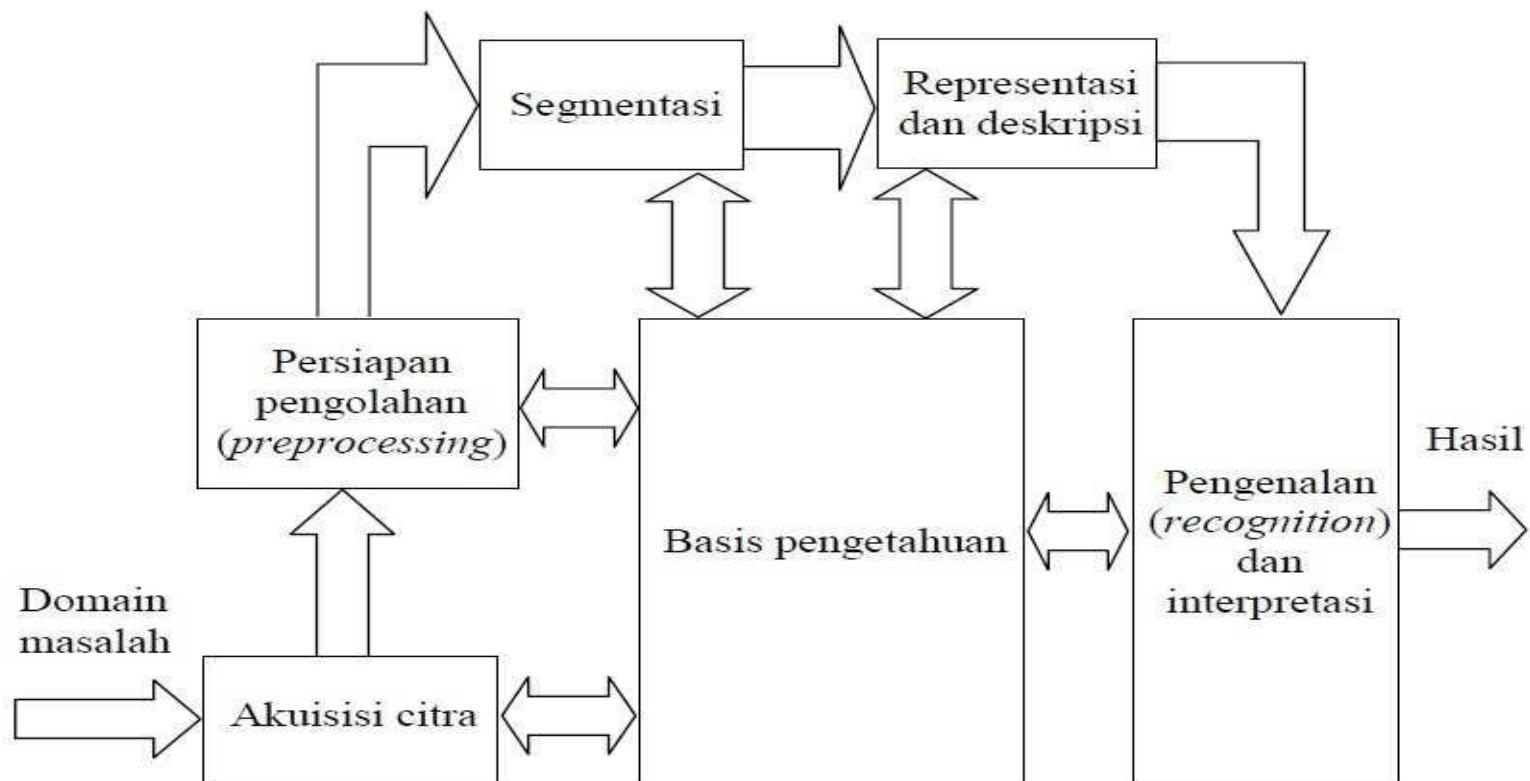
- **Tujuan:** menghitung besaran kuantitatif dari citra untuk menghasilkan deskripsinya
- **Contoh:**
 - Pendekslsian Tepi Objek (*edge detection*)
 - Ekstraksibatas (*boundary*)
 - Representasi daerah (*region*)



Citra Landsat berikut menunjukkan suhu permukaan laut global (dengan thermal IR). Dengan analisis representasi daerah dapat diketahui temperatur permukaan laut global di daerah tertentu.

- **Rekonstruksi Citra (*Image Reconstruction*)**
 - **Tujuan:** membentuk ulang objek dari beberapa citra hasil proyeksi
 - Jenis operasi ini banyak digunakan dalam bidang medis
 - **Contoh :Pengenalan Pola**

Langkah-langkah Pengolahan Citra secara umum





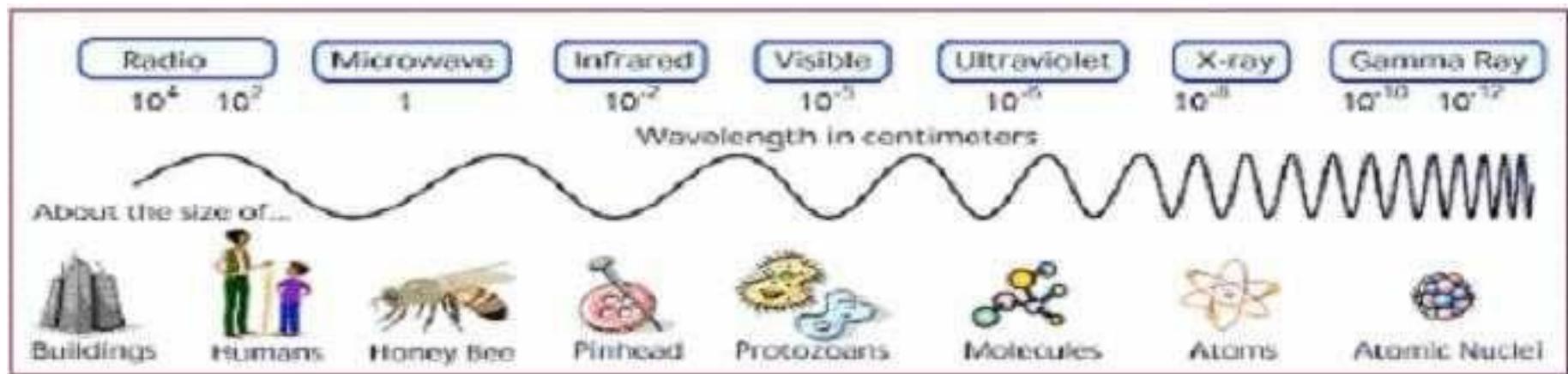
Persoalan di dalam pengolahan citra

- Capture
- Modelling
- Feature Extraction
- Image Segmentation

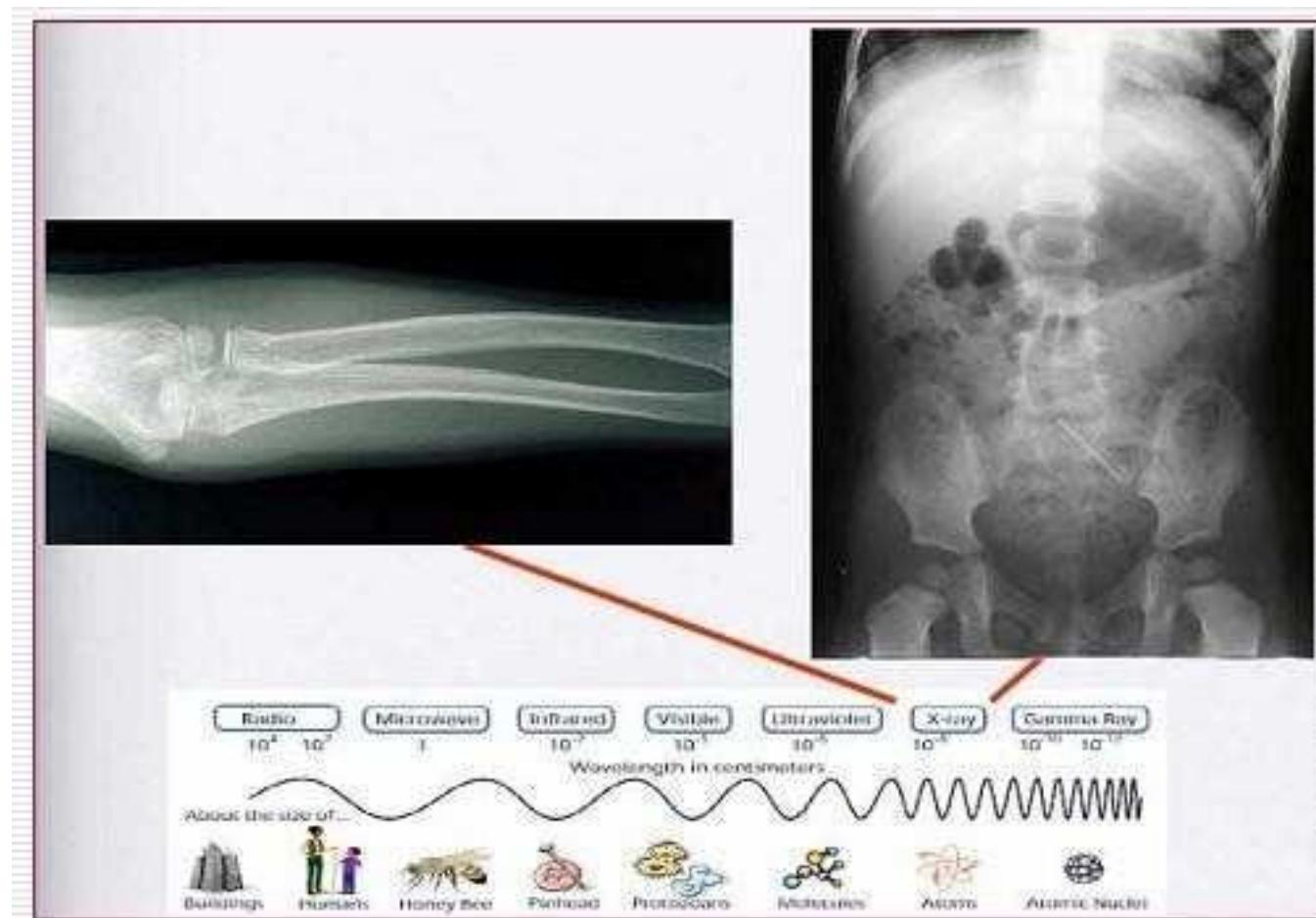
Permasalahan Capture

- Capture (Menangkap Gambar) merupakan proses awal dari image processing untuk mendapatkan gambar
- Proses capture membutuhkan alat-alat capture yang baik seperti kamera, scanner, light-pen dan lainnya, agar diperoleh gambar yang baik.
- Gambar yang baik akan banyak membantu dalam proses selanjutnya.

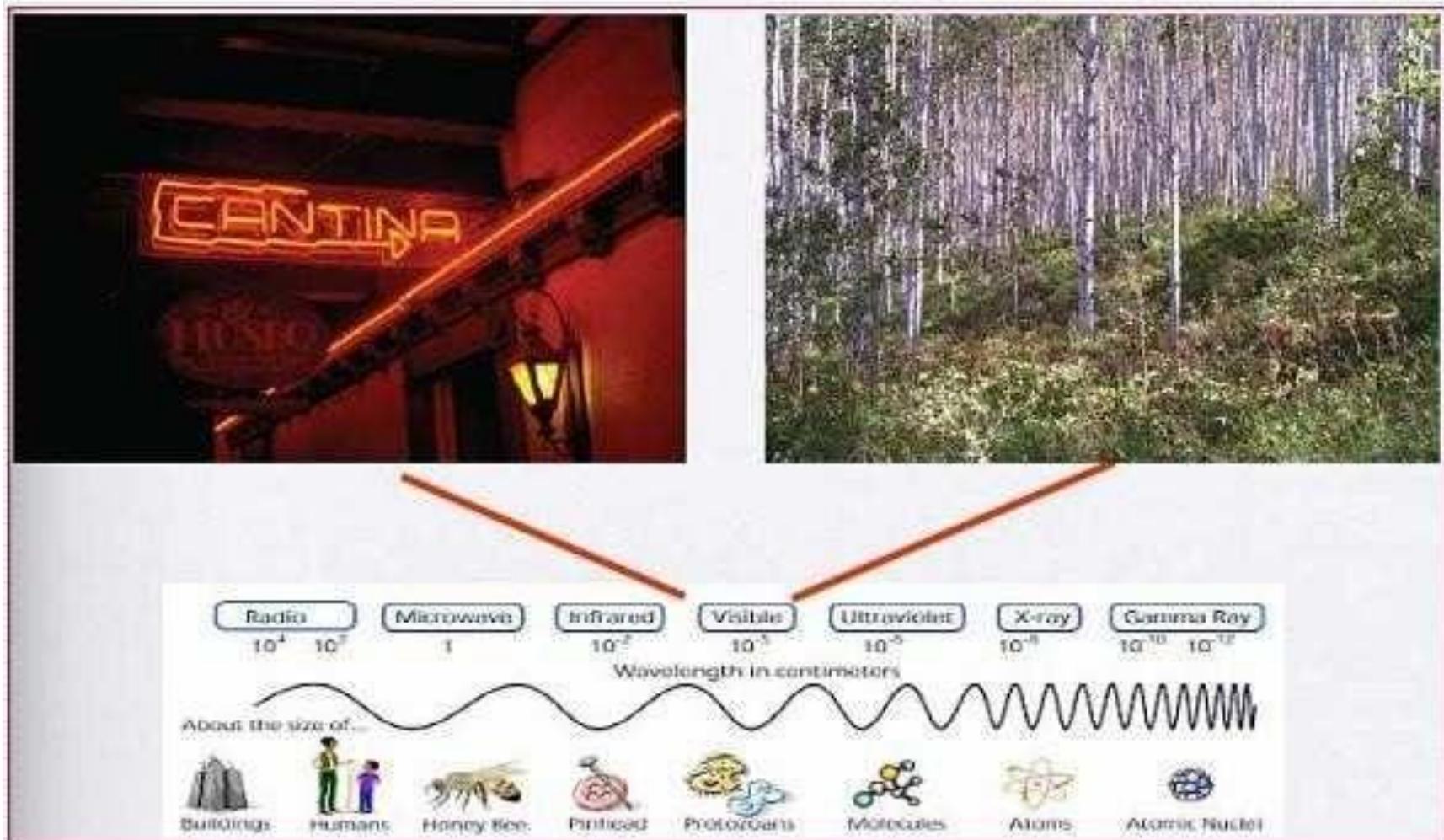
Alat-alat capture sesuai frekwensinya



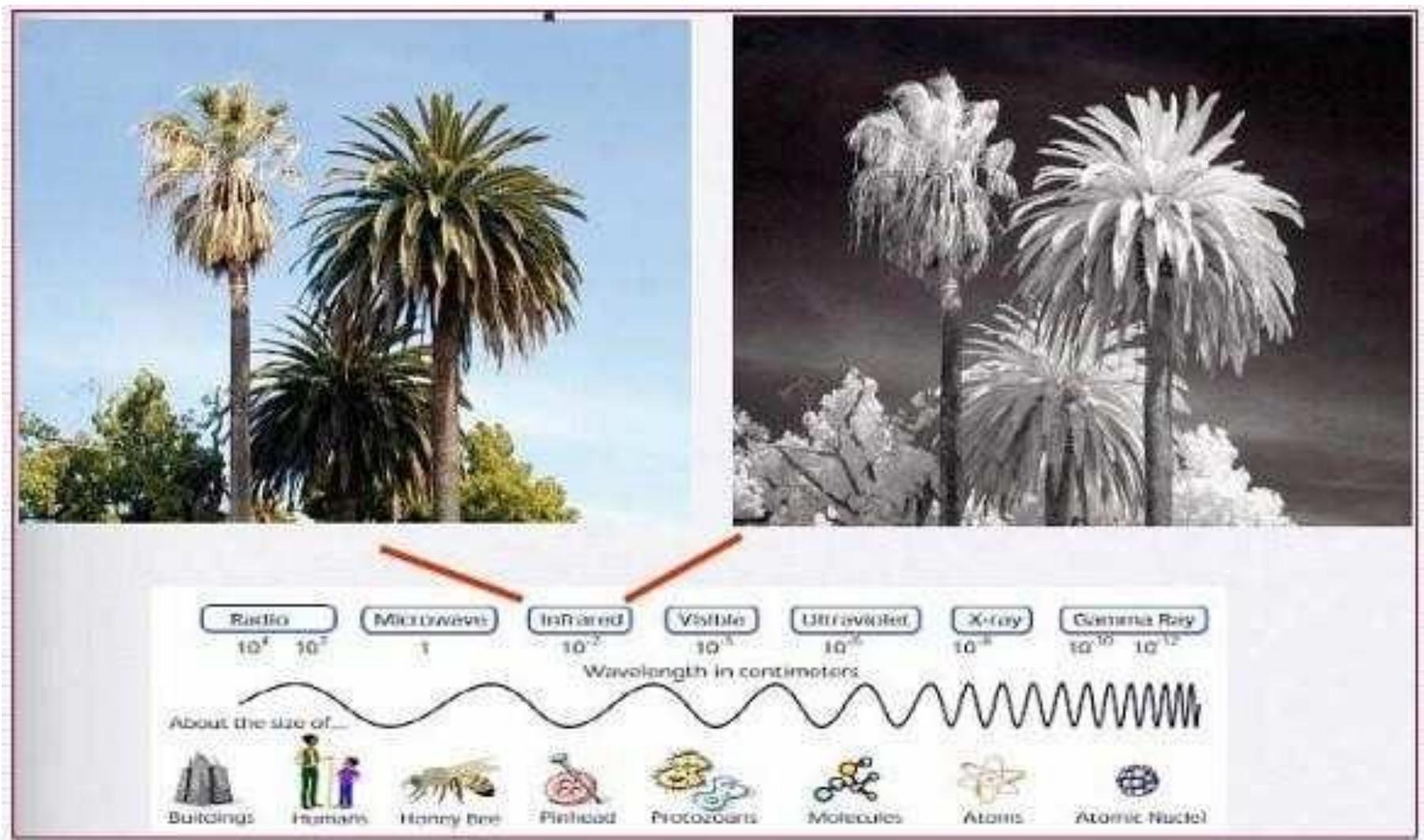
Hasil Capture



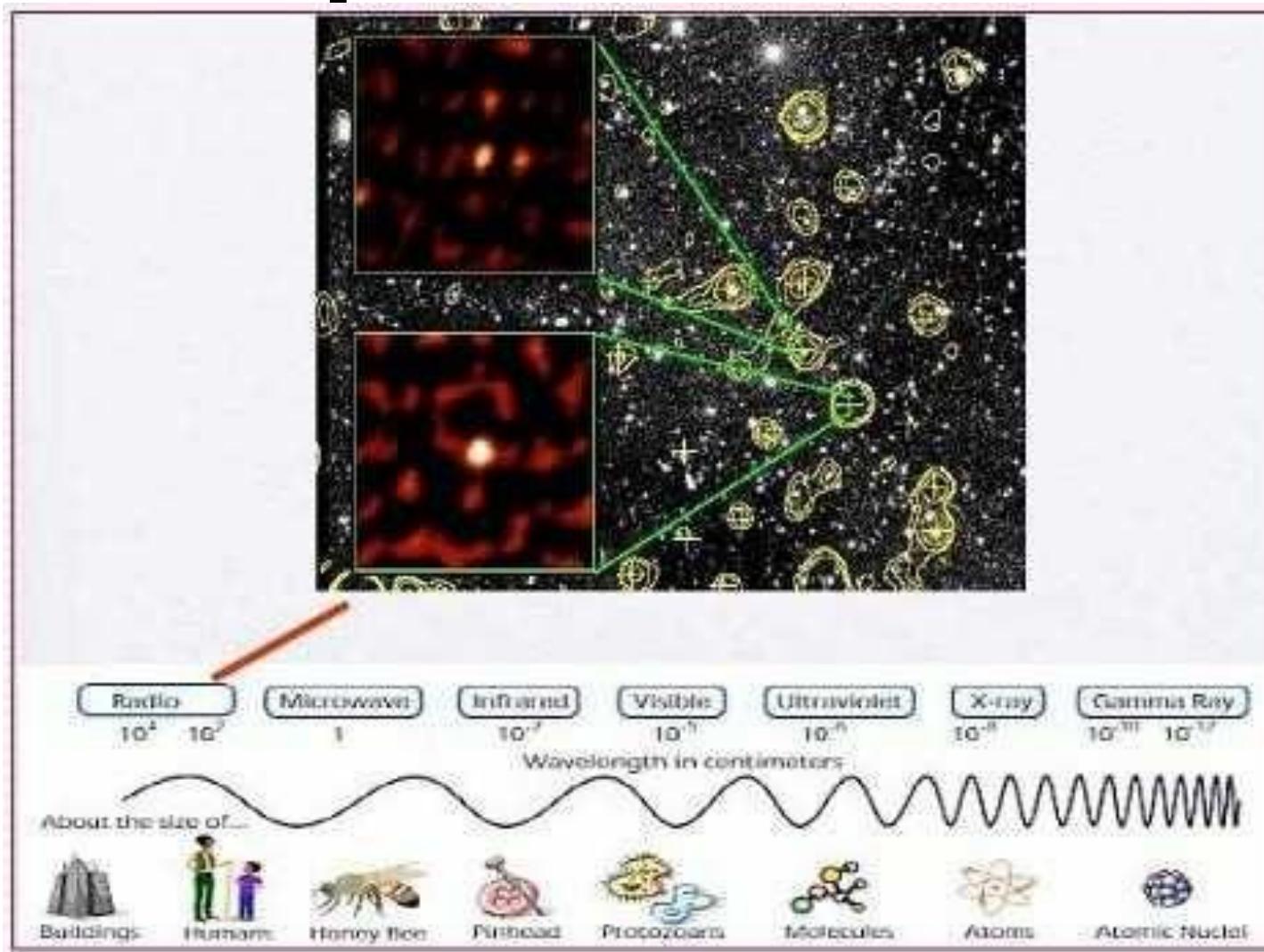
Hasil Capture



Hasil Capture



Hasil Capture



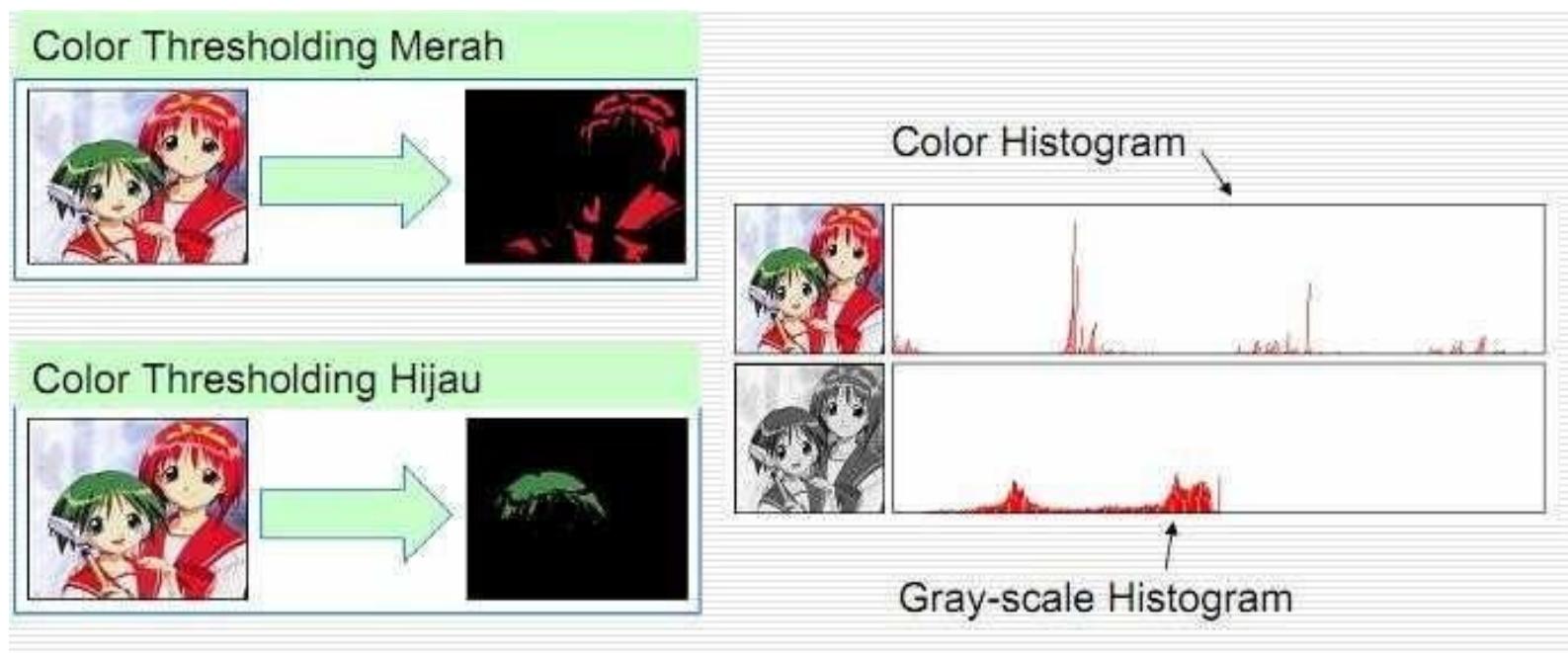
Permasalahan Feature Extraction

- Setiap gambar mempunyai karakteristik tersendiri, sehingga fitur tidak dapat bersifat general tetapi sangat tergantung pada model dan obyek gambar yang digunakan.
- Fitur dasar yang bisa diambil adalah warna, bentuk dan tekstur. Fitur yang lebih kompleks menggunakan segmentasi, clustering dan motion estimation.
- Pemakaian statistik dan probabilitas, pengolahan sinyal sampai pada machine learning diperlukan di sini.

Permasalahan Feature Extraction

□ Fitur Warna

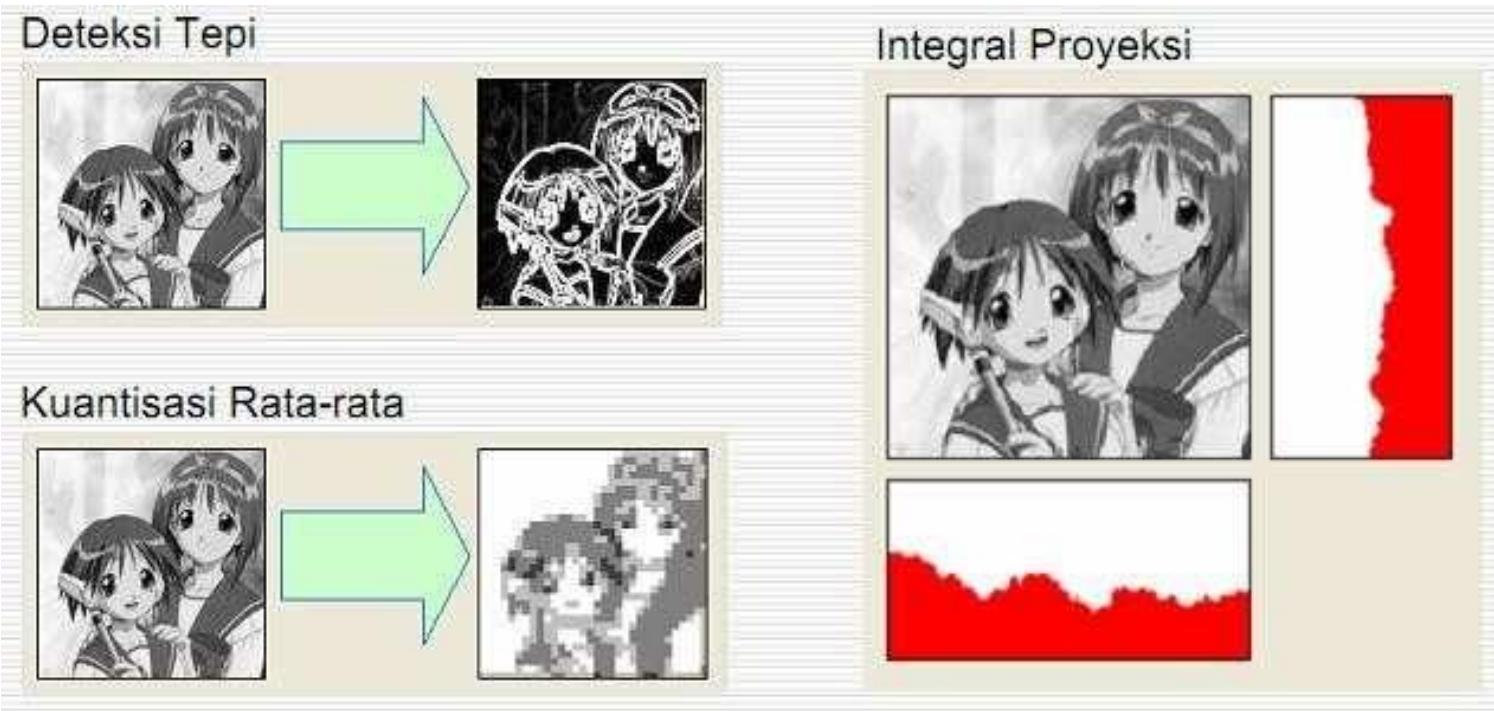
Fitur ini digunakan bila setiap objek gambar mempunyai warna yang spesifik



Permasalahan Feature Extraction

□ Fitur Bentuk

Fitur ini digunakan bila gambar setiap objek mempunyai bentuk yang spesifik

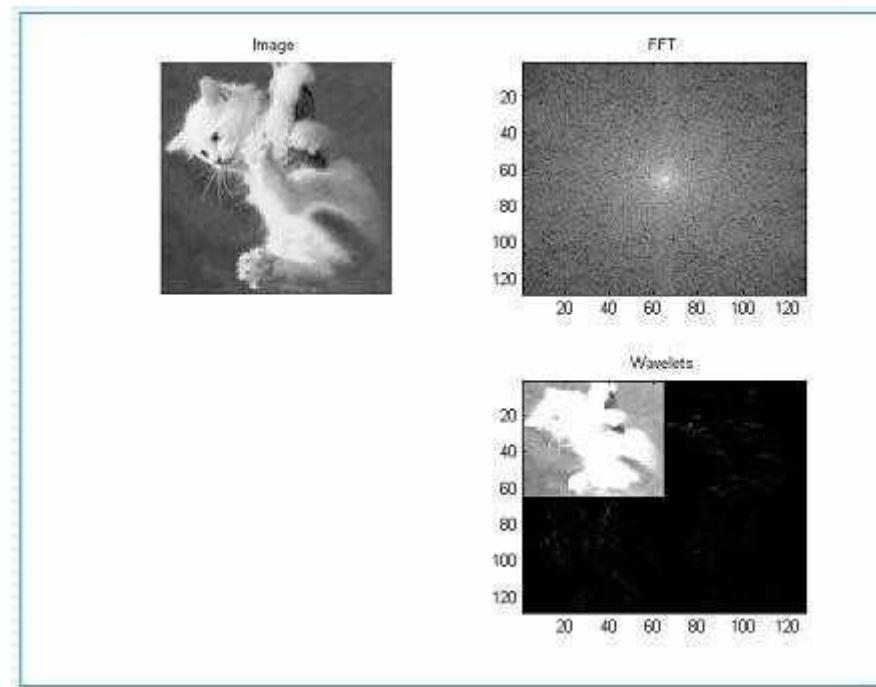


Permasalahan Feature Extraction

Fitur Tekstur

Beberapa Algoritma untuk mendapatkan fitur tekstur:

- (1) F T; (2) Wavelet; (3) Image Filter; (4) Filter Gabor



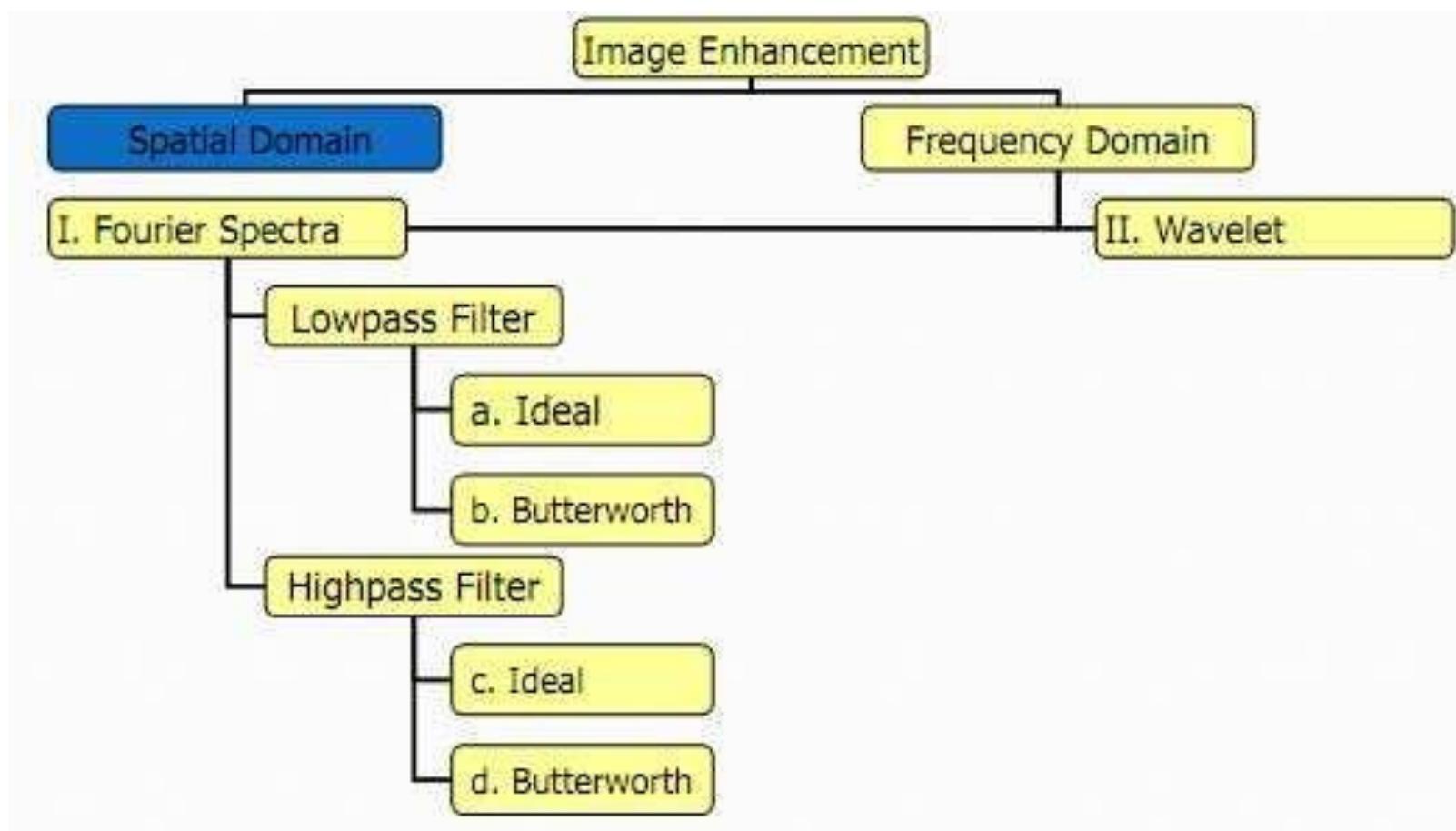


Permasalahan Image Segmentation

- Bagaimana memisahkan obyek gambar dengan backgroundnya
- Bagaimana memisahkan setiap obyek gambar.
- Teknik clustering apa yang sesuai dengan model dan obyek gambar yang digunakan



Transformasi Fourier



Domain Spasial vs Domain Frekuensi

DOMAIN SPASIAL	DOMAIN FREKUENSI
Konsep koordinat baris dan kolom	Konsep frekuensi, perubahan intensitas piksel ke piksel (frekuensi rendah dan tinggi)
Pemrosesan pixel-by-pixel	Pemrosesan berdasarkan pemilihan frekuensi yang akan difilter atau tidak
Komputasi lama (terutama citra dengan ukuran spasial tinggi)	Komputasi relatif cepat (terutama citra dengan ukuran spasial tinggi)

Konsep Frekuensi dalam Citra

- || Sembarang sinyal spasial mempunyai representasi frekuensi
- || **Makna frekuensi dalam citra:**
 - || Komponen frekuensi tinggi dikaitkan dengan perubahan piksel ke piksel secara cepat sepanjang citra. Misal: teks, tekstur, dsb.
 - || Komponen frekuensi tinggi dikaitkan dengan fitur berskala besar pada citra. Misal: daerah dengan intensitas konstan, atau piksel yang jumlahnya mendominasi dalam seluruh daerah citra.

TRANSFORMASI FOURIER

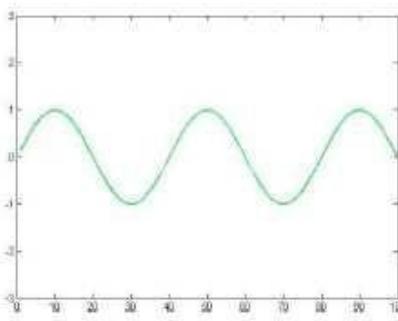
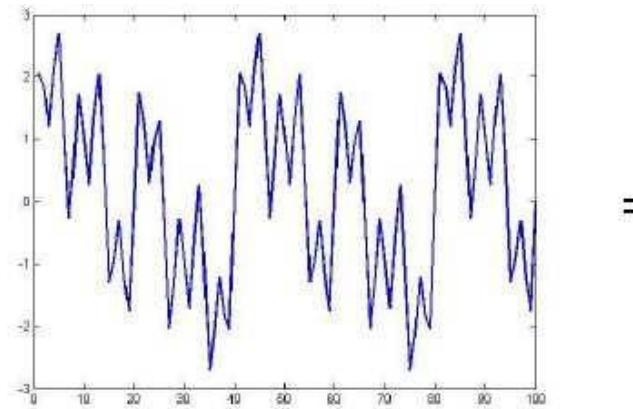
- || Konvolusi per pixel → LAMA, terdapat operasi perkalian dan penjumlahan untuk setiap pixel
- || **Untuk mempercepat komputasi :**
 - || Mengubah citra dari domain spatial ke domain frekuensi, dengan **transformasi fourier**
- || Keuntungan penggunaan domain frekuensi adalah:
 - || Proses konvolusi dapat diterapkan dalam bentuk perkalian langsung

DASAR-DASAR TRANSFORMASI FOURIER

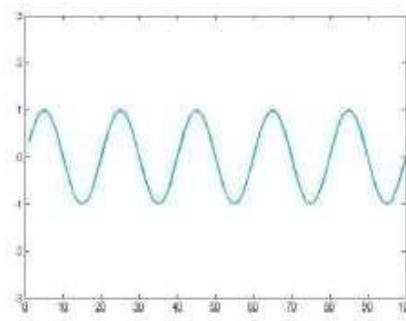
- | Transformasi fourier adalah suatu model transformasi yang memindahkan domain spasial atau domain waktu menjadi domain frekuensi.
- | Di dalam pengolahan citra digital transformasi fourier digunakan untuk mengubah domain spasial pada citra menjadi domain frekuensi.
- | Analisis dalam domain frekuensi banyak digunakan seperti filtering.
- | Dengan menggunakan transformasi fourier, sinyal atau citra dapat dilihat sebagai suatu objek dalam domain frekuensi

TRANSFORMASI FOURIER

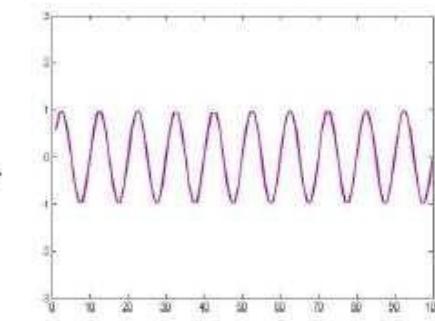
- I Fungsi periodik dapat dinyatakan sebagai jumlah sinus dan/atau cosinus dari perbedaan frekuensi setiap perkaliannya dengan koefisien yang berbeda



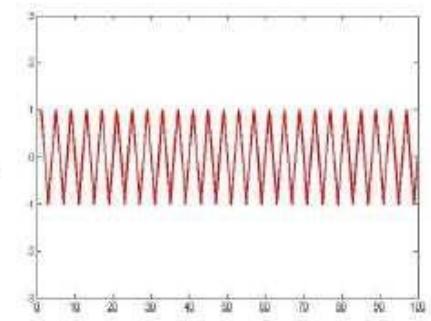
+



+



+



TRANSFORMASI FOURIER

- | Fungsi yang tidak periodik tetapi dengan daerah kurva yang terbatas dapat dinyatakan sebagai integral sinus dan/atau cosinus dikalikan dengan fungsi bobot.
- | Transformasi Fourier 1 dimensi:

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx$$

- | Invers :

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du$$

TRANSFORMASI FOURIER

I Transformasi Fourier 2 dimensi:

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux + vy)} dx dy$$

Invers :

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u) e^{j2\pi(ux + vy)} du dv$$

TRANSFORMASI FOURIER DISKRIT

- I Karena citra adalah gelombang diskrit, maka fungsi $f(x)$, $x=0,1,\dots,M-1$, untuk satu dimensi kita mendapatkan:

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M}$$

Invers :

$$f(x) = \sum_{u=0}^{M-1} F(u) e^{-j2\pi ux/M}$$

TRANSFORMASI FOURIER DISKRIT 1-D

■ **Formula Euler :**

$$e^{j\theta} = \cos\theta + j \sin\theta$$

■ **Sehingga didapatkan :**

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)(\cos(2\pi ux / N) - j \sin(2\pi ux / N))$$

- Untuk $u = 0, \dots, M-1$, $f(x)$ adalah nilai intensitas setiap piksel
- Nilai u adalah komponen dalam domain frekuensi
- Setiap $F(u)$ adalah nilai frekuensi dalam transformasi

- Hasil transformasi Fourier mengandung bilangan real dan imajiner yang berturut-turut dapat dinyatakan sebagai **(R(u)) dan (I(u))**
- Cara lain untuk menampilkan hasil transformasi untuk menghindari bilangan imajiner tersebut adalah menggunakan **spektrum (*magnitude*) dan sudut (*phase*) Fourier**

Spektrum Fourier :

$$| F(u) | = \left[R(u)^2 + I(u)^2 \right]^{1/2}$$

Sudut Fase Transformasi :

$$\theta(u, v) = \tan^{-1} \left[\frac{I(u)}{R(u)} \right]$$

TRANSFORMASI FOURIER DISKRIT 2-D

- Untuk citra 2 dimensi, DFT yang digunakan:

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)}$$

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \left[\cos 2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right) - j \sin 2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right) \right]$$

- Invers**

$$f'(x, y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} F'(u, v) \left[\cos 2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right) + j \sin 2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right) \right]$$

■ **Spektrum Fourier**

$$| F(u, v) | = \left[R(u, v)^2 + I(u, v)^2 \right]^{1/2}$$

■ **Sudut fase transformasi**

$$\theta(u, v) = \tan^{-1} \left[\frac{I(u, v)}{R(u, v)} \right]$$

■ Untuk menampilkan pada layar monitor, citra hasil transformasi Fourier sering ditampilkan dengan rumus :

$$D(u, v) = c \log(1 + | F(u, v) |)$$

■ dengan c menyatakan suatu konstanta

- Untuk $u=0, v=0$, didapatkan:

$$F(0,0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)$$

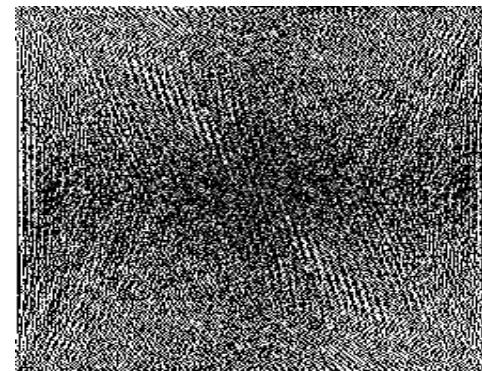
- Sama dengan rata-rata nilai intensitas.
- Lokasi ini juga adalah titik origin pada domain frekuensi.

Mendapatkan spektrum Fourier Citra

```
>> f = imread('bird.bmp');  
>> f = im2double(f);  
>> F = fft2(f);  
>> figure, imshow(F);  
>> F2 = log(1+abs(F));  
>> figure, imshow(F2,[ ]);  
>> Fs = fftshift(F2);  
>> figure, imshow(Fs,[ ]);  
>> f2 = ifft2(F);
```

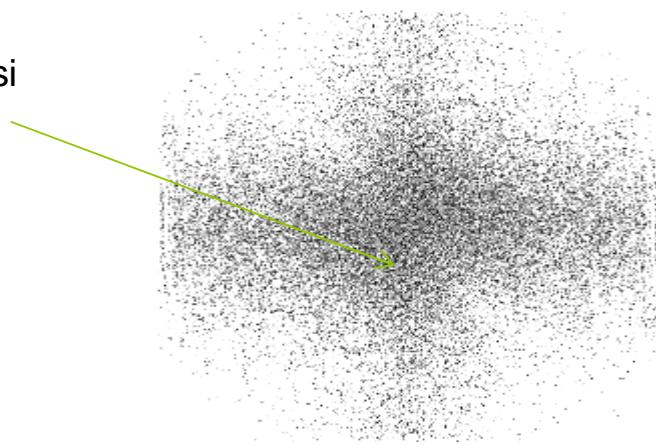


Citra asli

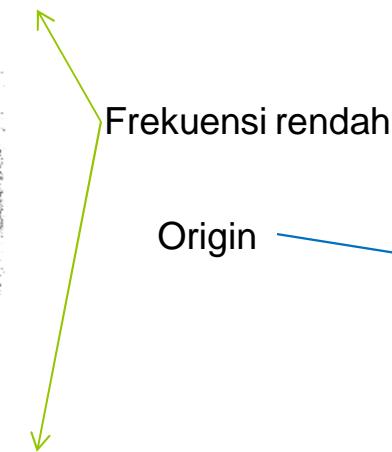


Spektrum asli

Frekuensi tinggi



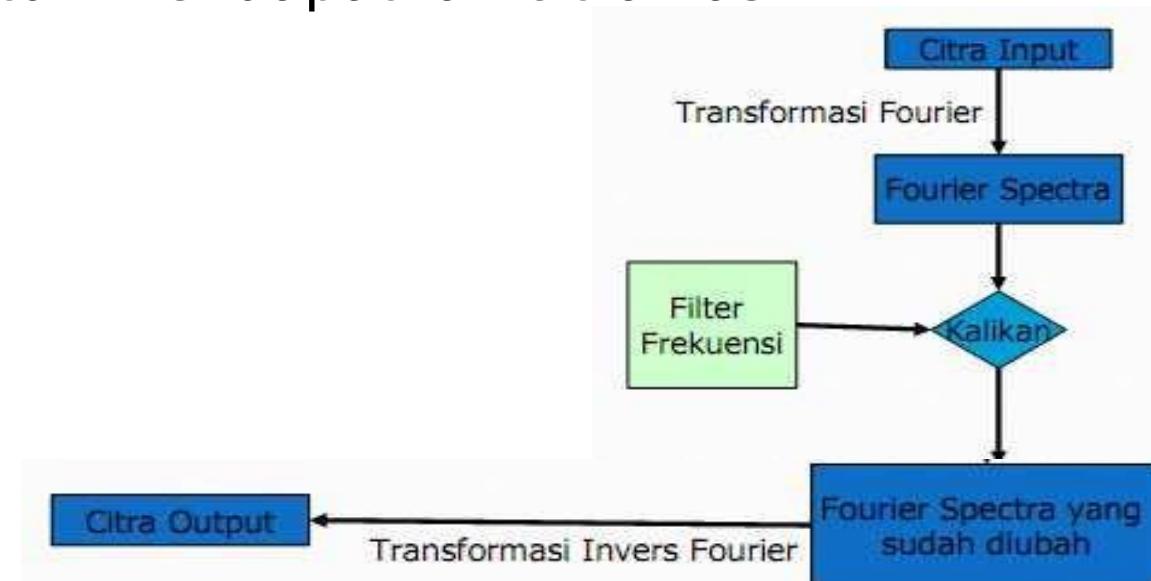
Spektrum setelah di-enhance dengan log



Setelah digeser (memusatkan origin)

FOURIER SPECTRA

- || Peningkatan mutu citra pada domain frekuensi Fourier dilakukan secara *straightforward* :
 - || Hitung transformasi fourier dari citranya → kalikan hasilnya dengan fungsi filter → lakukan transformasi invers untuk mendapatkan citra hasil



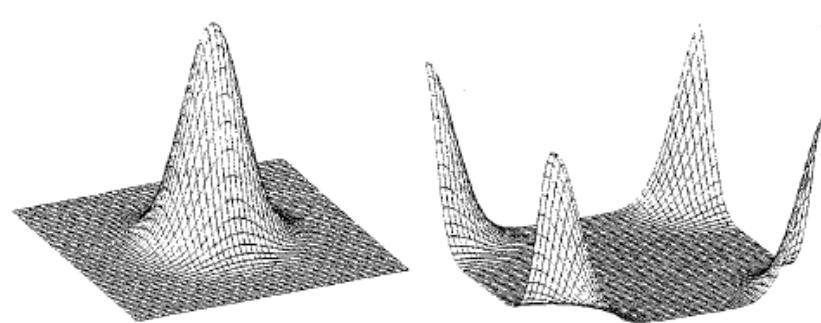
FILTER DALAM DOMAIN FREKUENSI

- I Dasar untuk filter linear dalam domain spasial dan frekuensi adalah teori konvolusi, yang dapat dituliskan dengan:

$$f(x, y) * h(x, y) \Leftrightarrow H(u, v)F(u, v)$$

- I Pemfilteran dalam domain spasial berisi konvolusi citra $f(x,y)$ mask filter $h(x,y)$.
- I Seperti halnya teori konvolusi, juga bisa mendapatkan hasil yang sama dalam domain frekuensi dengan perkalian antara $F(u,v)$ dengan $H(u,v)$, transformasi Fourier filter spasial.

FILTER DALAM DOMAIN FREKUENSI



- I Dasarnya, ide dalam pemfilteran domain frekuensi adalah untuk memilih fungsi transfer filter yang memodifikasi $F(u,v)$ dengan cara tertentu.

Transformasi Fourier untuk Analisa Citra

- | Untuk menganalisis citra pada domain frekuensi, hasil transformasi fourier dapat ditampilkan dalam bentuk citra di mana intensitasnya sebanding dengan besarnya $|F(u,v)|$ atau spektrum fourier
- | Agar citra dapat ditampilkan, maka sebelumnya dilakukan transformasi log :

$$D(u, v) = c \log(1 + |F(u, v)|)$$

$$G(u, v) = D(u, v) \cdot (-1)^{u+v}$$

TEKNIK FILTER DALAM DOMAIN FREKUENSI

Filter Penghalusan (Smoothing)

- Ideal Lowpass Filter (ILPF)
- Butterworth Lowpass Filter (BLPF)
- Gaussian Lowpass Filter (GLPF)

Filter Penajaman (Sharpening)

- Ideal Highpass Filter (IHPF)
- Butterworth Highpass Filter (BHPF)
- Gaussian Highpass Filter (GHPF)

TEKNIK FILTER DALAM DOMAIN FREKUENSI

■ ***Low frequencies :***

- Tingkat keabuan citra pada area yang halus (smooth)

■ ***High frequencies :***

- Detail citra seperti tepian (edges) dan noise

■ ***Lowpass filter :***

- Meloloskan *low frequencies*, meredam *high frequencies*

■ ***Highpass filter***

- Meloloskan *high frequencies*, meredam *low frequencies*

LANGKAH PEMFILTERAN

1. Kalikan citra input $f(x,y)$ dengan $(-1)^{u+v}$
2. Hitung $F(u,v)$, DFT dari citra pada langkah (1)
3. Kalikan $F(u,v)$ dengan fungsi filter $H(u,v)$
4. Hitung invers DFT berdasarkan hasil pada langkah (3)
5. Ambil komponen real berdasarkan hasil pada langkah (4)
6. Kalikan hasil pada langkah (5) dengan $(-1)^{x+y}$

Low Pass Filter

- || Smoothing (*blurring*) dicapai dalam domain frekuensi dengan pelemahan frekuensi tinggi; yang disebut dengan **lowpass filter**.
- || Meloloskan low frequencies, meredam high frequencies
- || **Jenis Lowpass Filter :**
 - || **Ideal Lowpass filter**
 - || **Butterworth lowpass filter**
 - || **Gaussian lowpass filter**

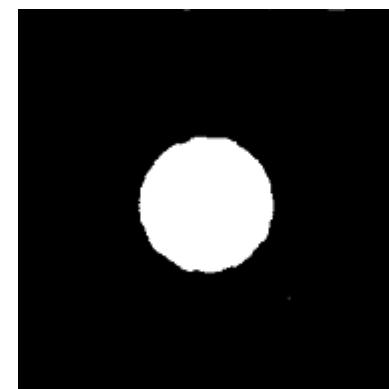
Low Pass Filter

■ Ideal Lowpass filter

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$$

- $D(u, v) \rightarrow$ jarak dari titik (u, v) ke titik pusat transformasi fourier

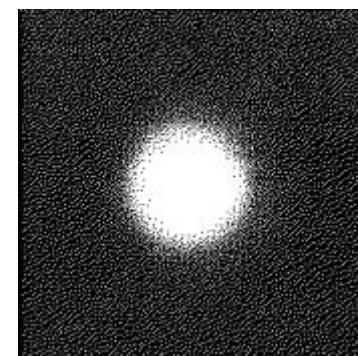
$$D(u, v) = \left[(u - M / 2)^2 + (v - N / 2)^2 \right]^{1/2}$$



Low Pass Filter

I Butterworth Lowpass filter

$$H(u, v) = \frac{1}{1 + [D(u, v) / D_0]^{2n}}$$



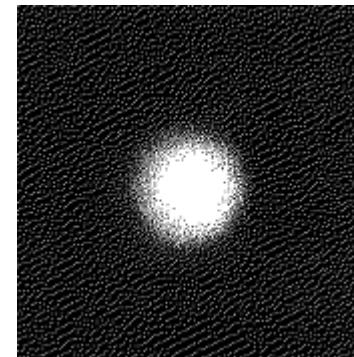
Low Pass Filter

■ Gaussian Lowpass filter

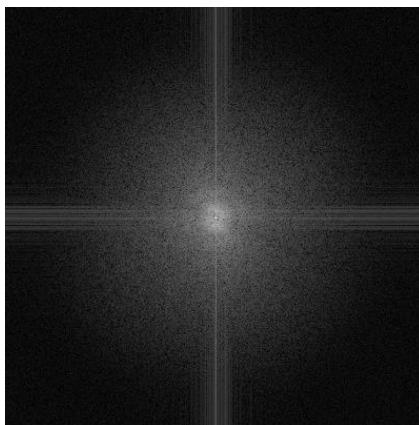
$$H(u, v) = e^{-D_2(u, v) / 2\sigma_2}$$

■ Bila $\sigma = D_0$ maka :

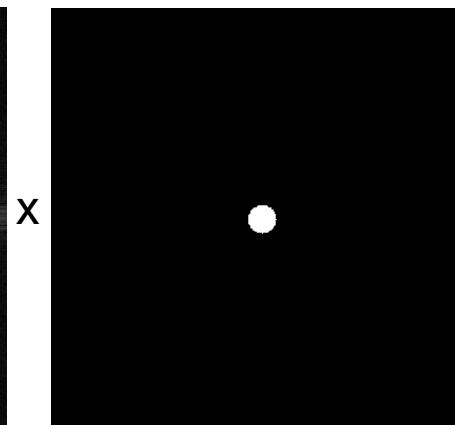
$$H(u, v) = e^{-D_2(u, v) / 2D_0}$$



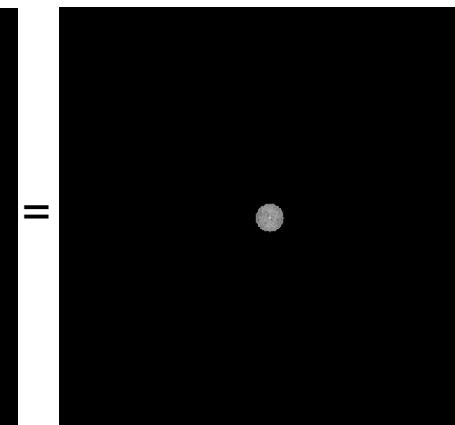
Ideal Low Pass Filter



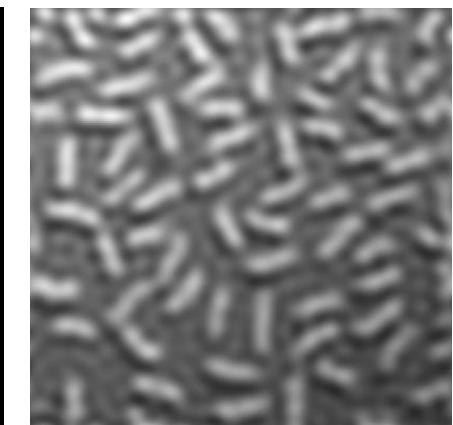
Spektrum asli



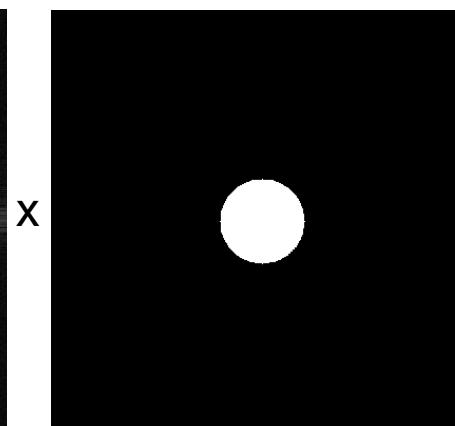
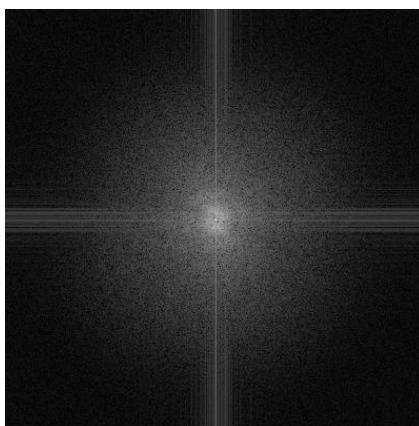
ILPF, $D_0 = 20$



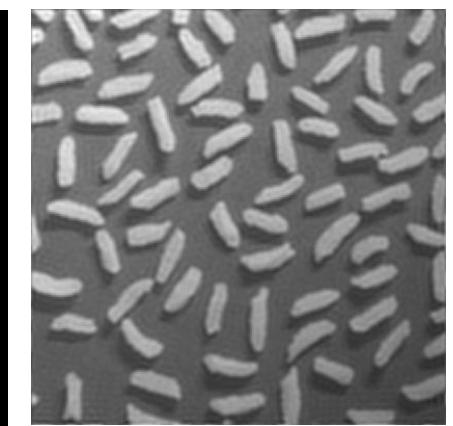
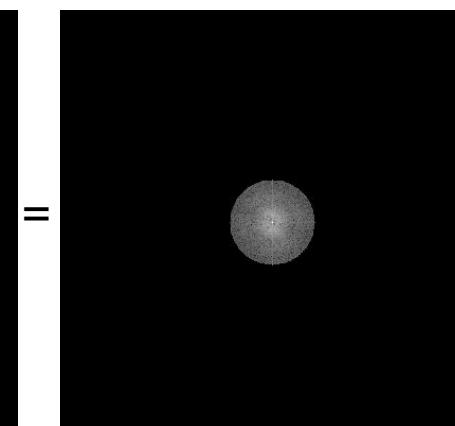
Spektrum hasil



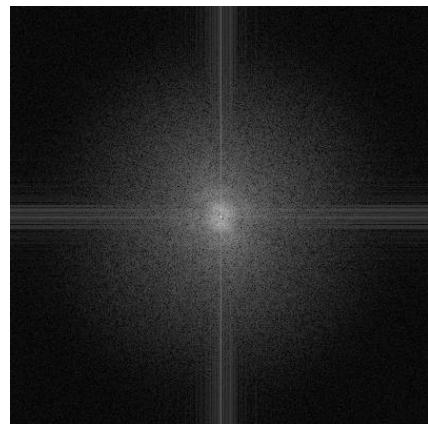
Citra hasil



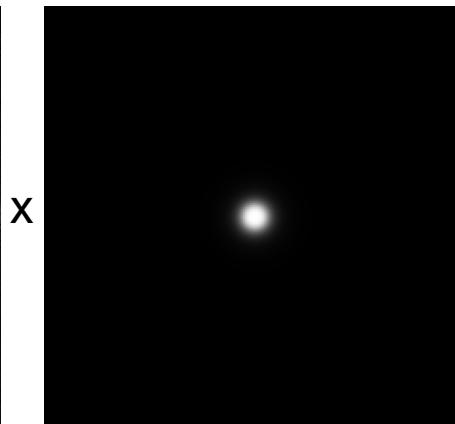
ILPF, $D_0 = 60$



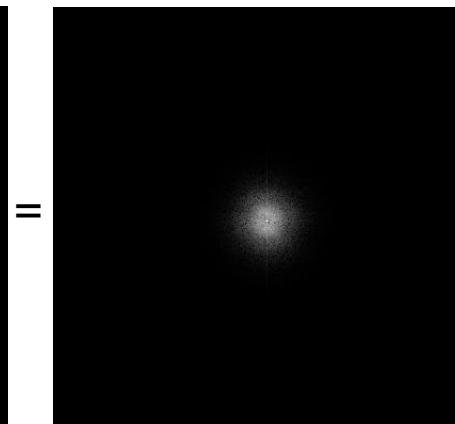
Butterworth lowpass Filter



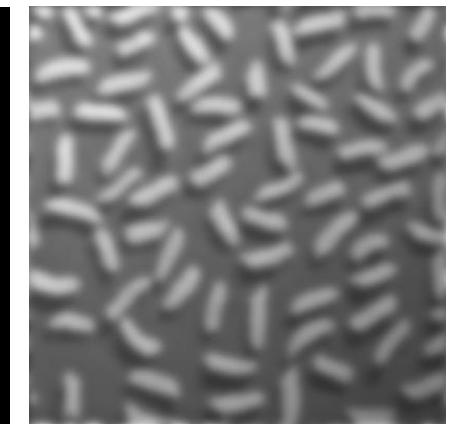
Spektrum asli



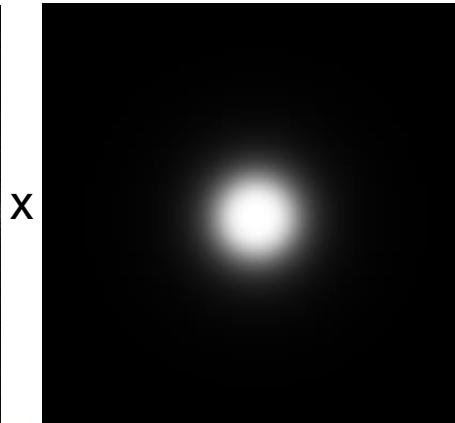
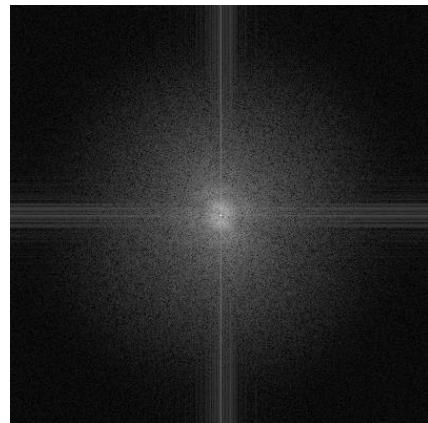
BLPF, $D_0 = 20$, sig = 2



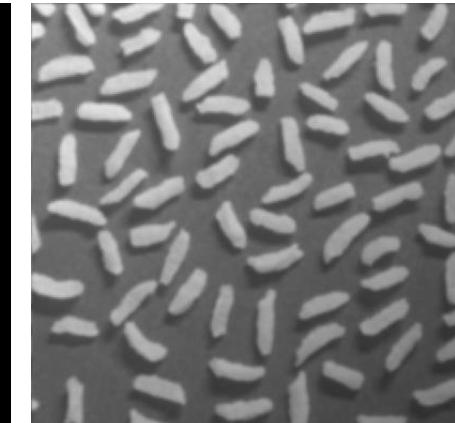
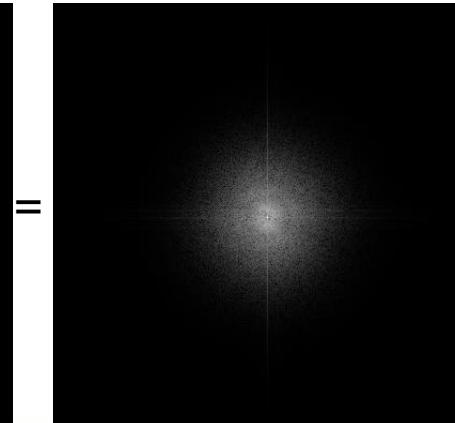
Spektrum hasil



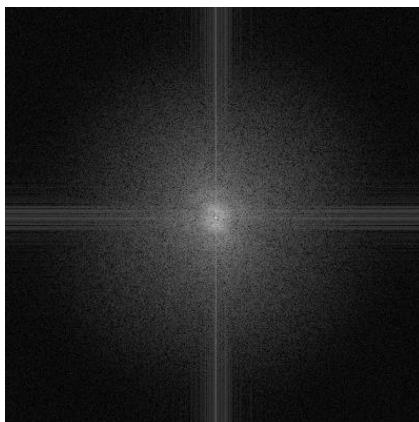
Citra hasil



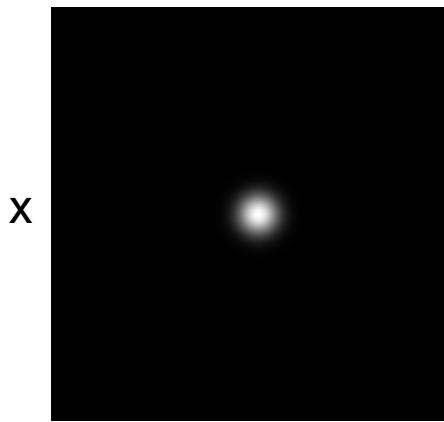
BLPF, $D_0 = 60$, sig = 2



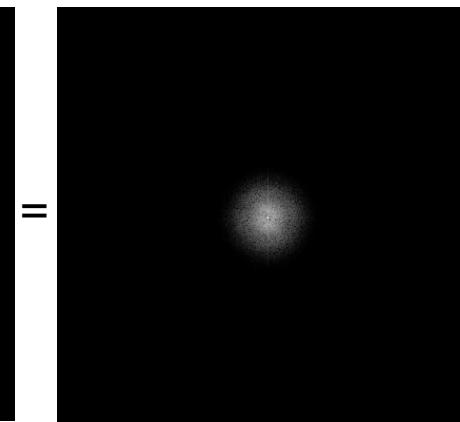
Gaussian lowpass Filter



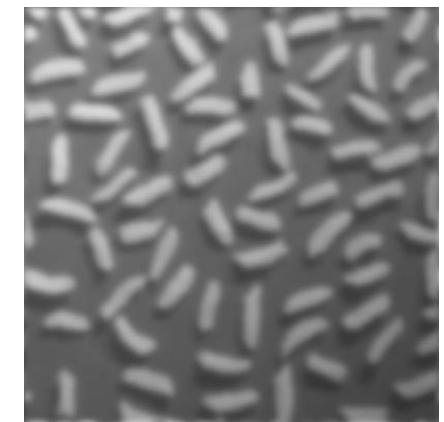
Spektrum asli



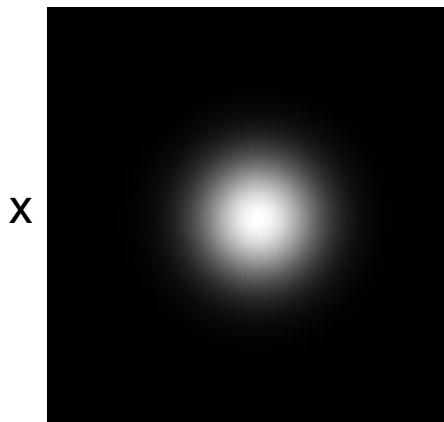
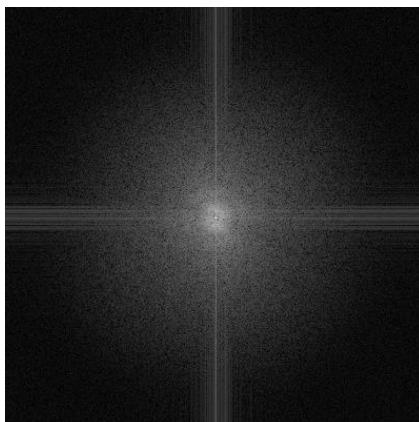
GLPF, $D_0 = 20$



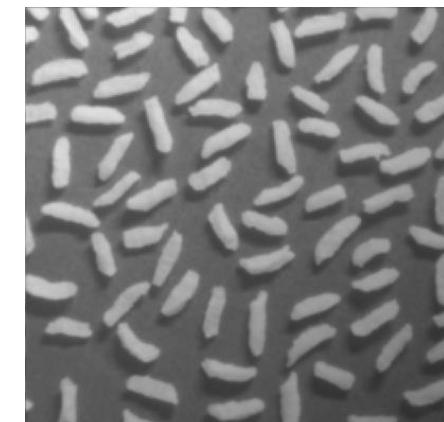
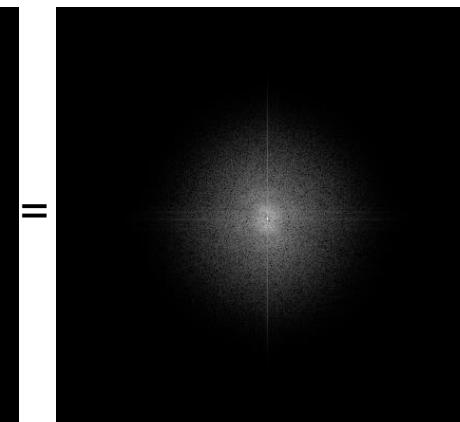
Spektrum hasil



Citra hasil



GLPF, $D_0 = 60$



Highpass Filter

- || Meloloskan *high frequencies*, meredam *low frequencies*
- || Kebalikan dari lowpass filtering

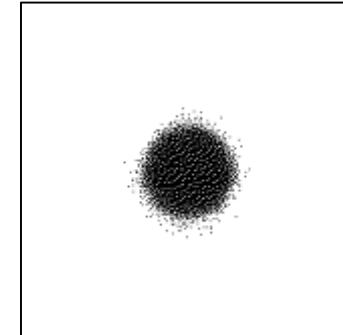
$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

- || **Jenis Highpass filter :**
 - || **Ideal Highpass filter**
 - || **Butterworth highpass filter**
 - || **Gaussian highpass filter**

Highpass Filter

| **Butterworth Highpass filter :**

$$H(u, v) = \frac{1}{1 + [D(u, v) / D_0]^{2n}}$$



Highpass Filter

| **Gaussian Highpass filter :**

$$H(u, v) = 1 - e^{-D(u, v)^2 / 2 D_0^2}$$

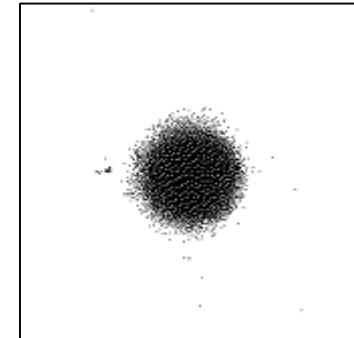
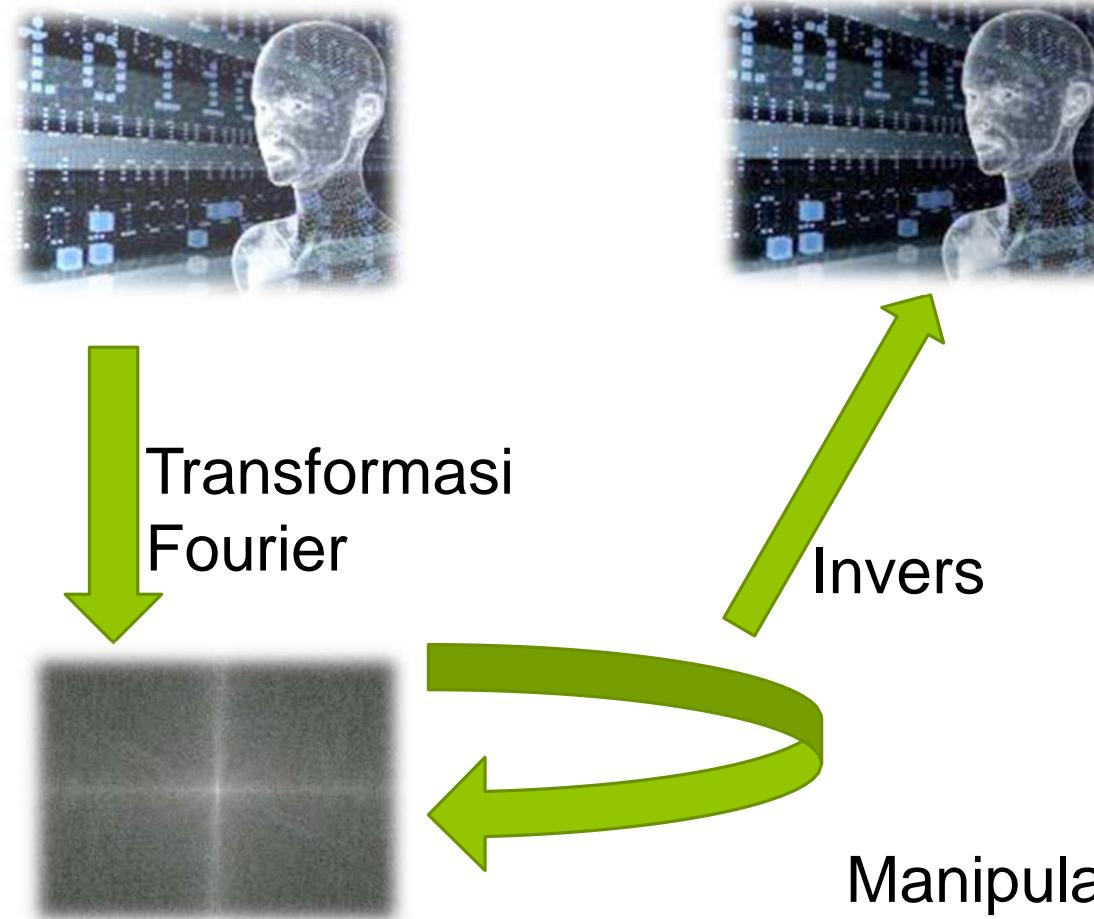


Image Restoration pada domain frekuensi



Contoh Soal

- || Hitung transformasi fourier untuk data citra berikut :
 - || $f(0,0) = 1, f(1,0) = 1,$
 - $f(0,1) = 1, f(1,1) = 1$

1	1
1	1

Dilakukan proses filtering dengan menggunakan Ideal Lowpass filtering $D_0=1$



Peningkatan Kualitas Citra (Image Enhancement)



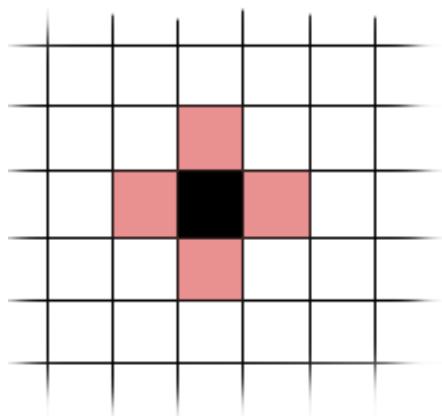
Hubungan antar pixel (1/4)

- Neighbourhood**
- Connectivity**
- Distance**

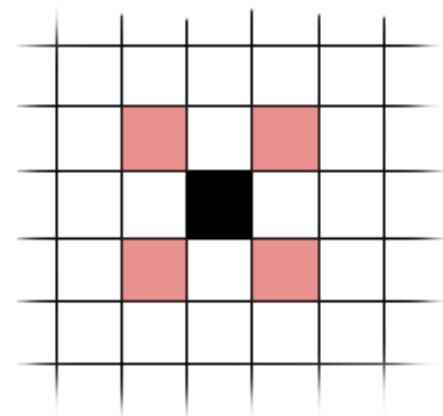
Hubungan antar pixel (2/4)

□ Neighbourhood (tetangga pixel)(1)

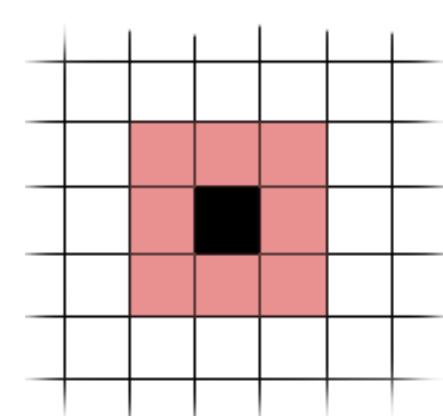
- Tetangga horisontal dan vertikal → $N_4(p)$
- Tetangga diagonal → $N_D(p)$
- 8-tetangga → $N_8(p)$



$N_4(p)$
)



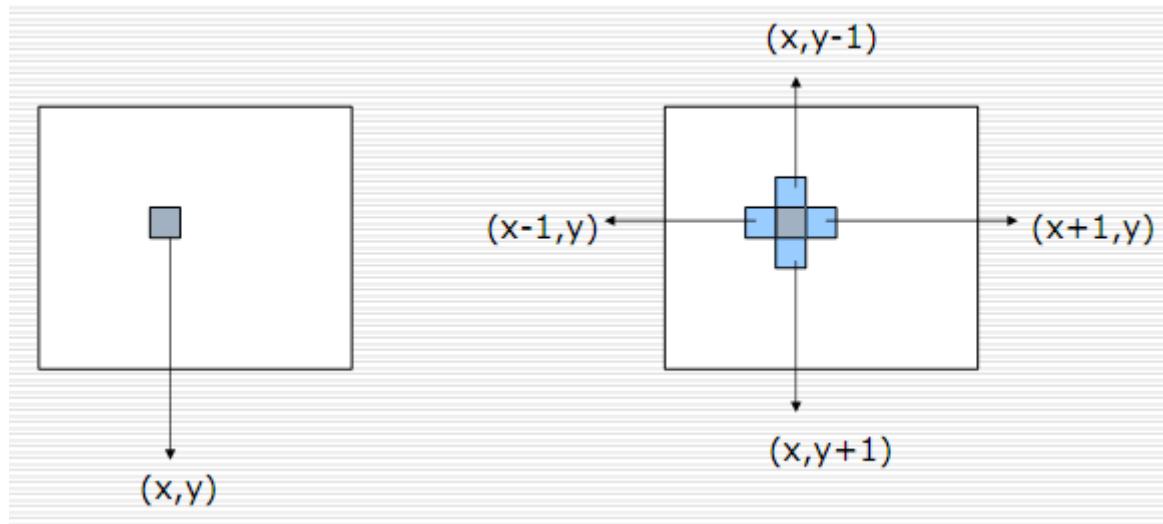
$N_D(p)$
)



$N_8(p)$
)

Hubungan antar pixel (2/4)

- **Neighbourhood (tetangga pixel)(2)**
 - **Tetangga horisontal dan vertikal → $N_4(p)$**
 - 4 titik tetangga (x,y) adalah titik-titik: $(x-1,y)$, $(x+1,y)$, $(x,y-1)$ dan $(x,y+1)$ sebagai tetangga kiri, kanan, atas dan bawah

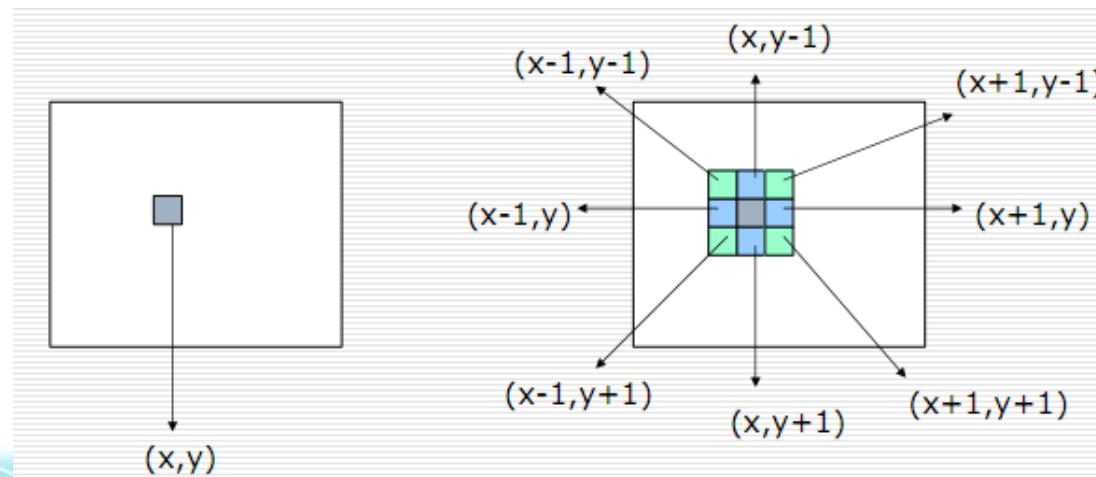


Hubungan antar pixel (2/4)

□ Neighbourhood (tetangga pixel)(3)

□ 8-tetangga $\rightarrow N_8(p)$

- Titik (x,y) dan 8 titik tetangganya merupakan suatu matrik ukuran 3×3 yang merupakan dasar dari pengolahan citra lebih lanjut.
- 8 titik tetangga (x,y) adalah titik-titik: $(x-1,y-1), (x-1,y), (x-1,y+1), (x,y-1), (x,y+1), (x+1,y-1), (x+1,y)$ dan $(x+1,y+1)$



Hubungan antar pixel (3/4)

□ **Connectivity (1)**

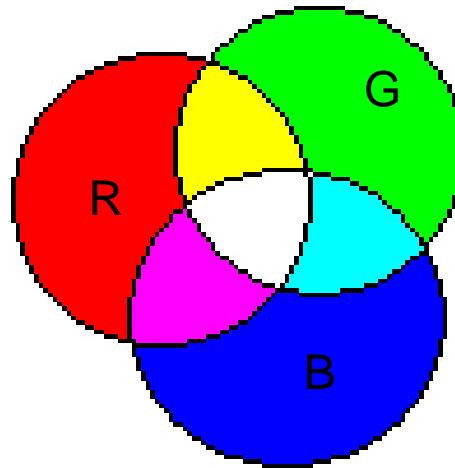
- Menentukan apakah 2 pixel saling berhubungan berdasarkan kriteria tertentu
- Merupakan konsep penting untuk menentukan batas objek
- **Syarat konektivitas adalah :**
 - 2 pixel memiliki gray level yang hampir sama
 - 2 pixel tersebut bertetangga
- **Contoh kriteria gray level :**
 - Misalkan pada suatu image **8bit (warna =256)**, konektivitas terjadi bila kedua pixel terletak pada himpunan $V=(32,33,34,\dots,62,63)$

Hubungan antar pixel (4/4)

- **Connectivity (2)**
 - **3 macam konektivitas :**
 - **4-konektivitas**
 - Dua pixel pdan q dengan gray level termasuk dalam V bila q adalah anggota himpunan $N_4(p)$
 - **8-konektivitas**
 - Dua pixel pdan q dengan gray level termasuk dalam V bila q adalah anggota himpunan $N_8(p)$
 - **M-konektivitas (konektivitas campur)**
 - Dua pixel p dan q dengan gray level termasuk dalam V bila
 - i. q adalah anggota himpunan $N_4(p)$ atau
 - ii. q adalah anggota himpunan $N_8(p) \cap N_4(q) = \emptyset$

(pengecekan i dilakukan lebih dulu)

Format Warna

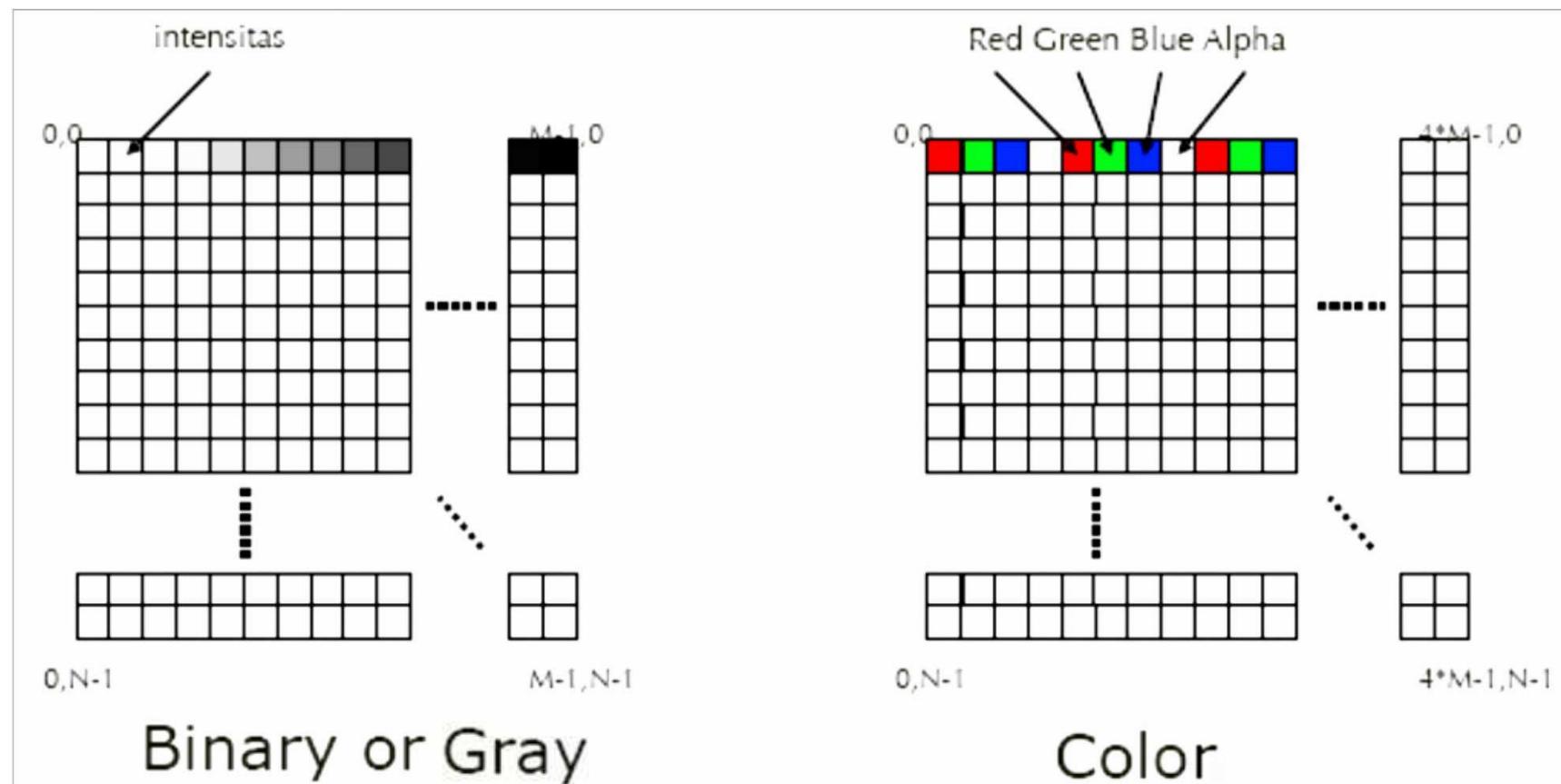


- Format Warna 24 BIT dinyatakan dengan :
11001001 01011001 00001011
R (8 bit) G (8 bit) B (8 bit)
- Masing-masing komponen warna RGB mempunyai nilai 0 s/d 255 (8bit) derajat kecerahan (derajat keabuan)

Format Warna

Warna		R	G	B
Merah		255	0	0
Hijau		0	255	0
Biru		0	0	255
Kuning		255	255	0
Magenta		255	0	255
Cyan		0	255	255
Putih		255	255	255
Hitam		0	0	0
Abu-abu		128	128	128

Representasi Citra



Konversi RGB ke Gray Scale



Setiap pixel mempunyai nilai red (r), green (g) dan blue (b) dengan nilai masing-masing 0-255

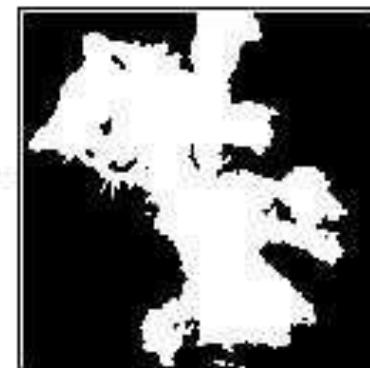
$$x = \frac{r + g + b}{3}$$

Setiap pixel mempunyai nilai derajat keabuan x dengan nilai 0-255

$$x = a_r \cdot r + a_g \cdot g + a_b \cdot b$$

dimana : $a_r + a_g + a_b = 1$

Konversi Gray Scale ke Biner



Setiap pixel mempunyai nilai derajat keabuan x dengan nilai 0-255

Setiap pixel mempunyai nilai warna x_{bw} dengan nilai 0 dan 1

$$x_{bw} = \begin{cases} 1 & \text{jika } x \geq 128 \\ 0 & \text{jika } x < 128 \end{cases}$$
$$x_{bw} = \begin{cases} 1 & \text{jika } x \geq x \\ 0 & \text{jika } x < x \end{cases}$$

Konversi Gray Scale ke m-Bit



Setiap pixel mempunyai nilai derajat keabuan x dengan nilai 0-255

Setiap pixel mempunyai nilai warna x_{th} dengan nilai 0 sampai dengan 2^{m-1}



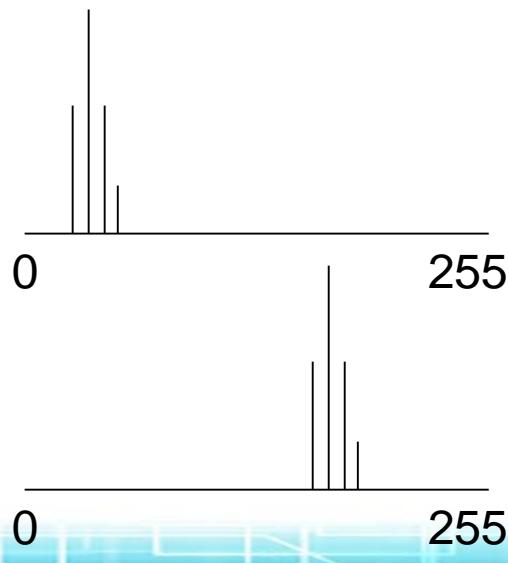
$$x_{th} = (2^m) \text{int}\left(\frac{x}{2^m}\right)$$

Contoh :
 $X=100$, gray scale 4 bit (0-64)
 $X_{\text{baru}} = 64 \times (100/64)$
 $= 64 \times 1 = 64$

Pengaturan Brightness



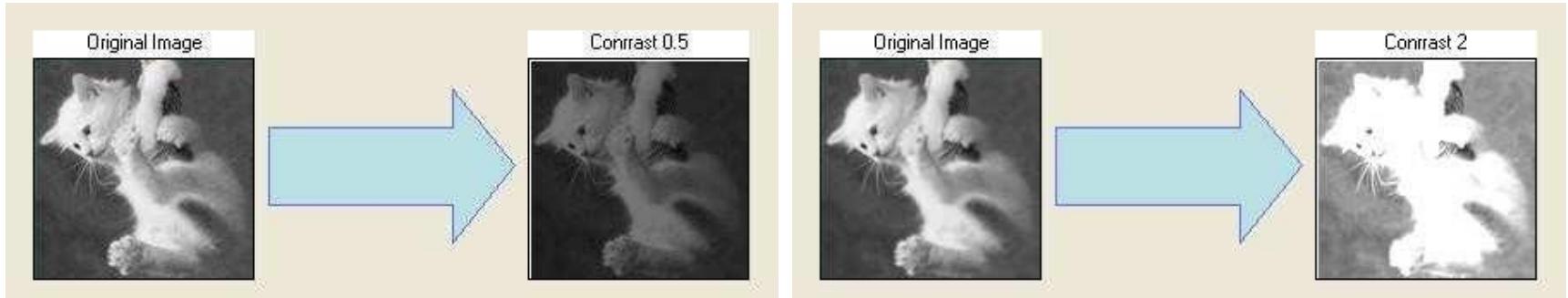
Proses pengaturan brightness adalah proses penambahan nilai derajat keabuan x dengan nilai perubahan brightness $t_{brightness}$



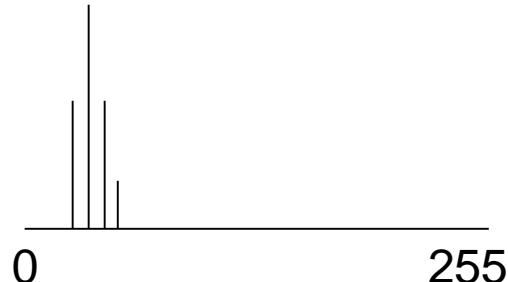
$$x_{brightness} = x + t_{brightness}$$

$t_{brightness}$ bisa positif dan negatif

Pengaturan Contrast

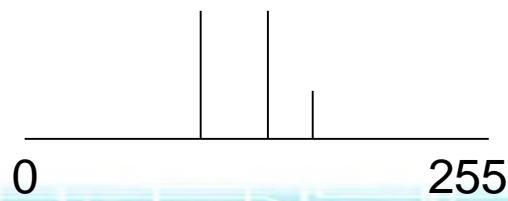


Proses pengaturan contrast adalah proses perkalian nilai derajat keabuan x dengan nilai perubahan contrast $t_{contrast}$



$$x_{contrast} = x \times t_{contrast}$$

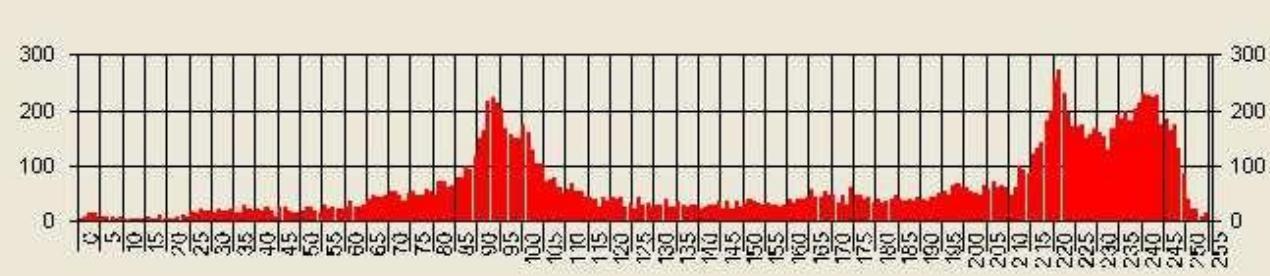
$0 < t_{kontras} < m$, dengan m positif



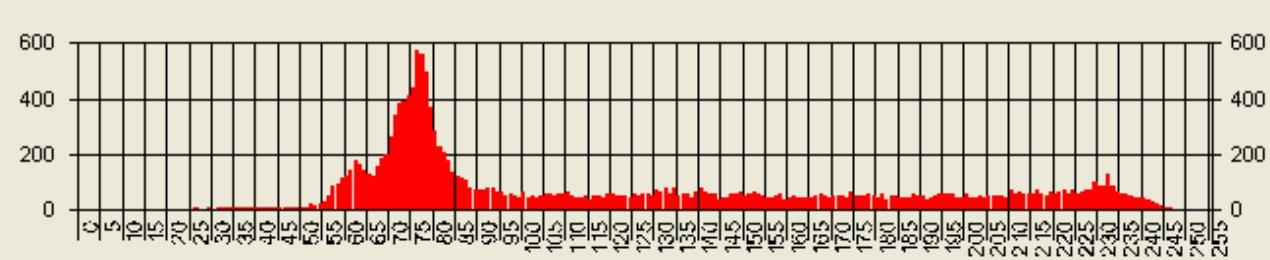
Gray Scale Histogram

- Histogram di dalam gambar gray-scale → **distribusi dari derajat keabuan (terang/gelap)** pada suatu gambar.
- Dari histogram ini dapat dilihat :
 - Gambar tersebut lebih banyak warna gelap atau
 - Lebih banyak warna terang
- **Histogram Equalization adalah** suatu teknik untuk meratakan distribusi terang/gelap sehingga gambar kelihatan lebih jelas
 - Teknik histogram ini dapat dikembangkan untuk memperbaiki kualitas gambar (image enhancement)

Gray Scale Histogram



Gambar ini didominasi warna terang, karena grafik di sebelah kanan terlihat lebih banyak.



Gambar ini didominasi warna gelap, karena grafik di sebelah kiri terlihat lebih banyak.

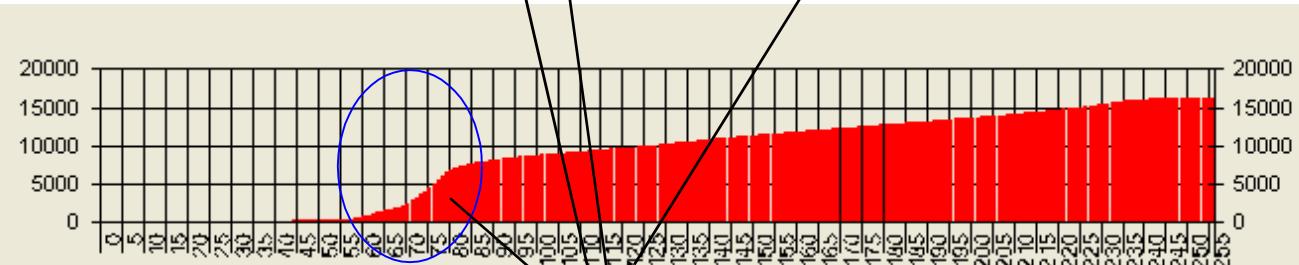
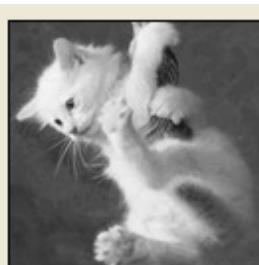
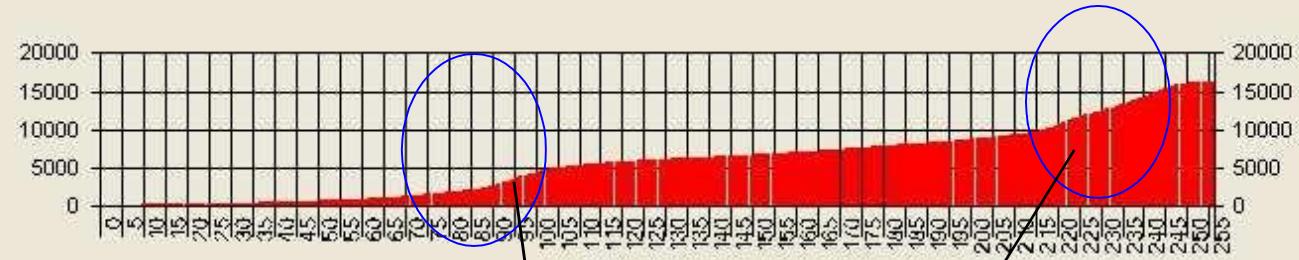
Distribusi Kumulatif

- **Distribusi kumulatif C(x)** adalah nilai total histogram dari tingkat keabuan=0 sampai dengan tingkat keabuan=x, dan didefinisikan dengan:

$$C(x) = \sum_{w=0}^x H(w)$$

- **Dapat digunakan** untuk menunjukkan perkembangan dari setiap step derajat keabuan.
- **Gambar dikatakan baik bila** mempunyai distribusi kumulatif yang pergerakannya hampir sama pada semua derajat keabuan.

Distribusi Kumulatif



Perubahan yang tajam

Distribusi Kumulatif

- Gambar-gambar hasil photo mempunyai perubahan yang tidak terlalu tajam dan biasanya tidak lebih dari satu. Hal ini menunjukkan **tingkat gradiasi yang halus pada gambar hasil photo.**
- Gambar-gambar kartun mempunya banyak perubahan yang tajam, hal ini menunjukkan **tingkat gradiasi pada gambar kartun rendah (kasar).**

Histogram Equalization

- **Histogram Equalization adalah** suatu proses untuk meratakan histogram agar derajat keabuan dari yang paling rendah (0) sampai dengan yang paling tinggi (255) mempunyai kemunculan yang rata.
- Dengan histogram equalization hasil gambar yang memiliki **histogram yang tidak merata** atau **distribusi kumulatif yang banyak loncatan gradiasinya** akan menjadi **gambar yang lebih jelas** karena derajat keabuannya tidak dominan gelap atau dominan terang.
- **Proses histogram equalization ini mengunakan distribusi kumulatif**, karena dalam proses ini dilakukan perataan gradien dari distribusi kumulatifnya.

Formulasi Histogram Equalization

- Histogram Equalization dari suatu distribusi kumulatif C adalah:

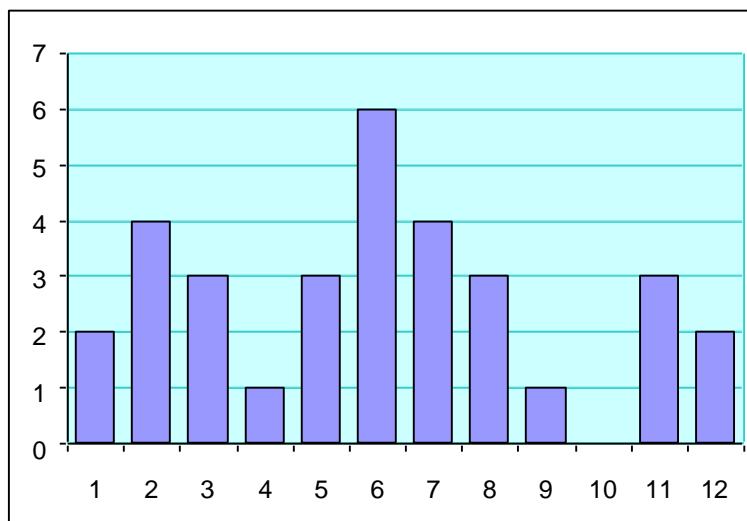
$$w = \frac{c_w \cdot t}{n_x \cdot n_y}$$

- Dengan :
 - **C_w** : nilai distribusi kumulatif pada derajat keabuan w
 - **t** : nilai threshold derajat keabuan= 2⁸ atau 256
 - **n_x.n_y** : ukuran gambar.

Perhitungan Histogram Equalization

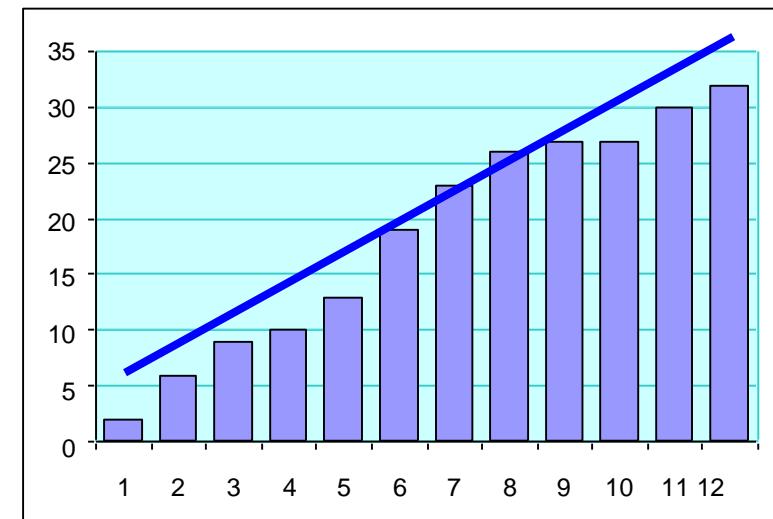
Perhatikan histogram berikut:

2 4 3 1 3 6 4 3 1 0 3 2



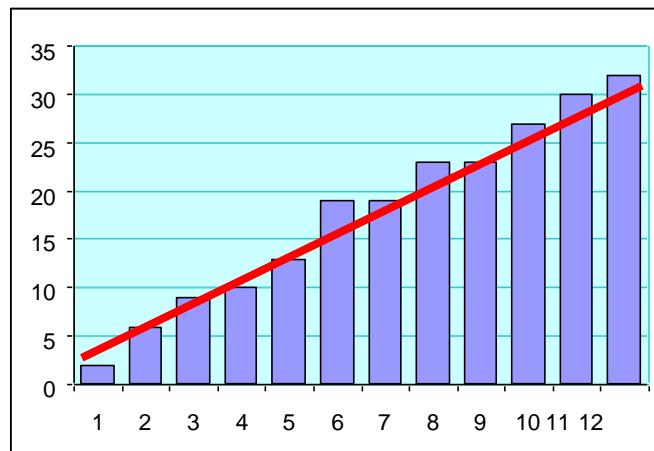
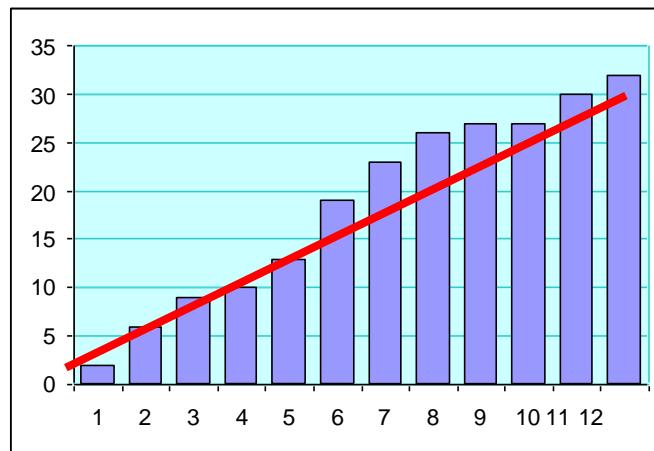
Distribusi Kumulatifnya

2 6 9 10 13 19 23 26 27 27 30 32



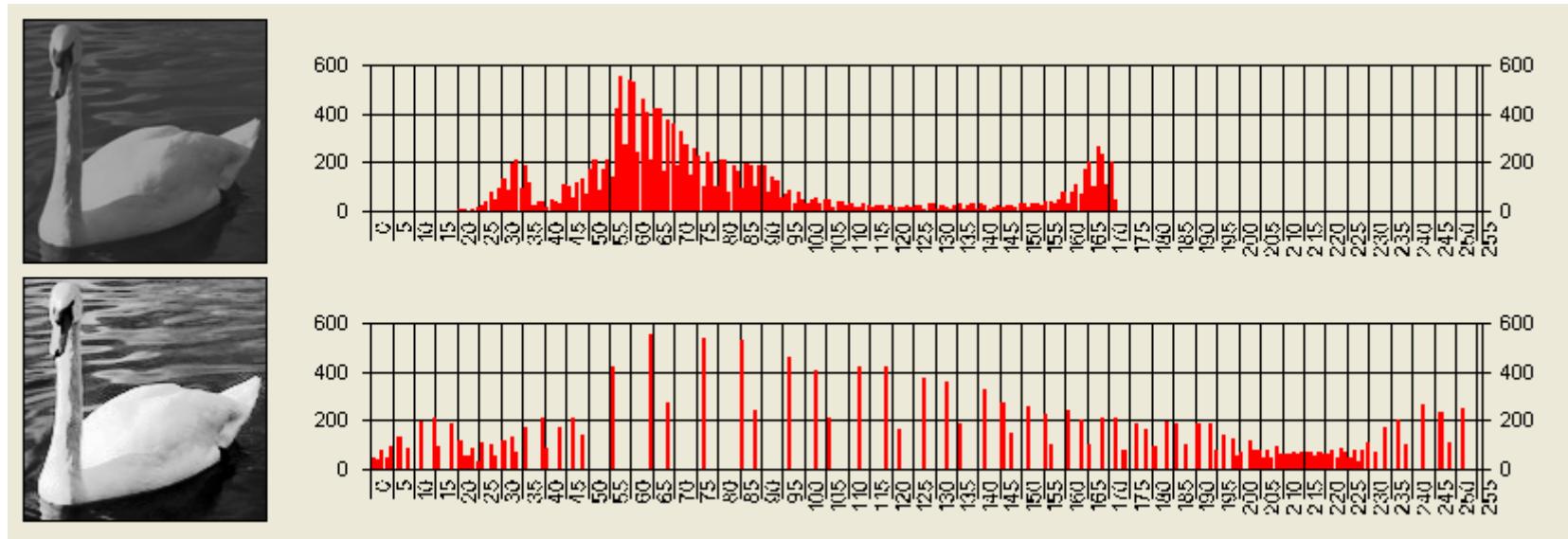
Perhitungan Histogram Equalization

Distribusi Kumulatif: 2 6 9 10 13 19 23 26 27 27 30 32

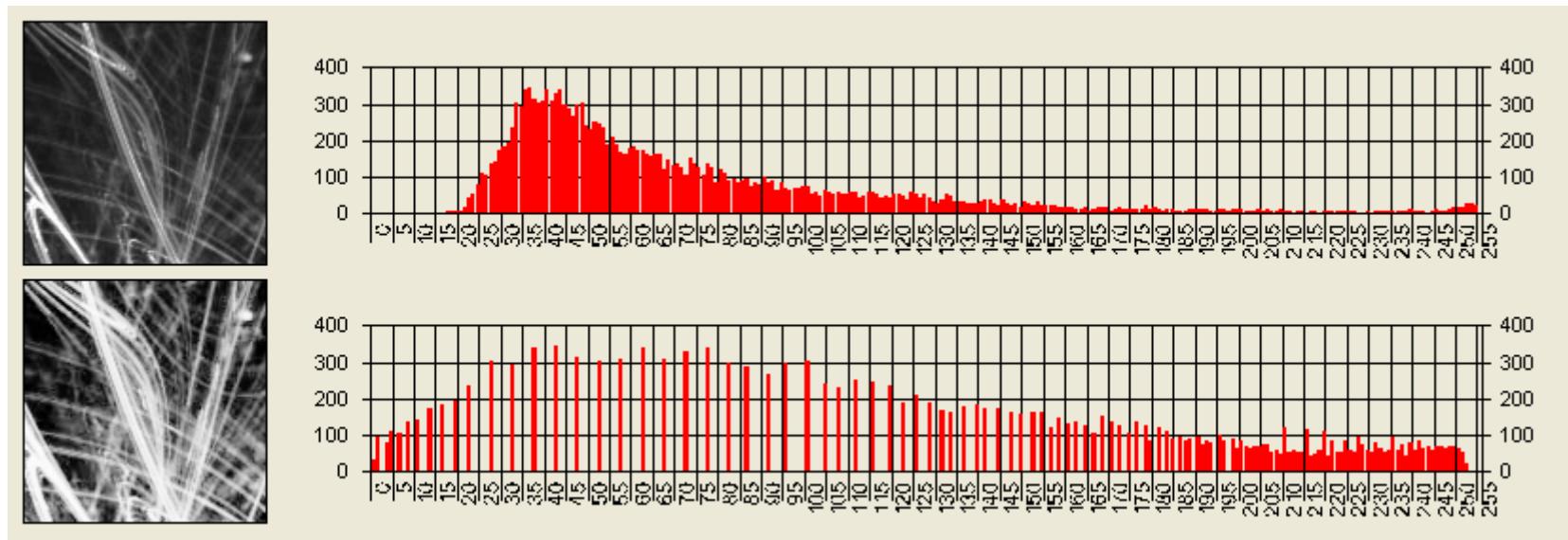


w	Cw	w-baru
1	2	(2*12)/41
2	6	1
3	9	2
4	10	3
5	13	4
6	19	5
7	23	7
8	26	9
9	27	10
10	27	10
11	30	11
12	32	12

Histogram Equalization Pada Gambar



Histogram Equalization Pada Gambar



Histogram Equalization Pada Gambar

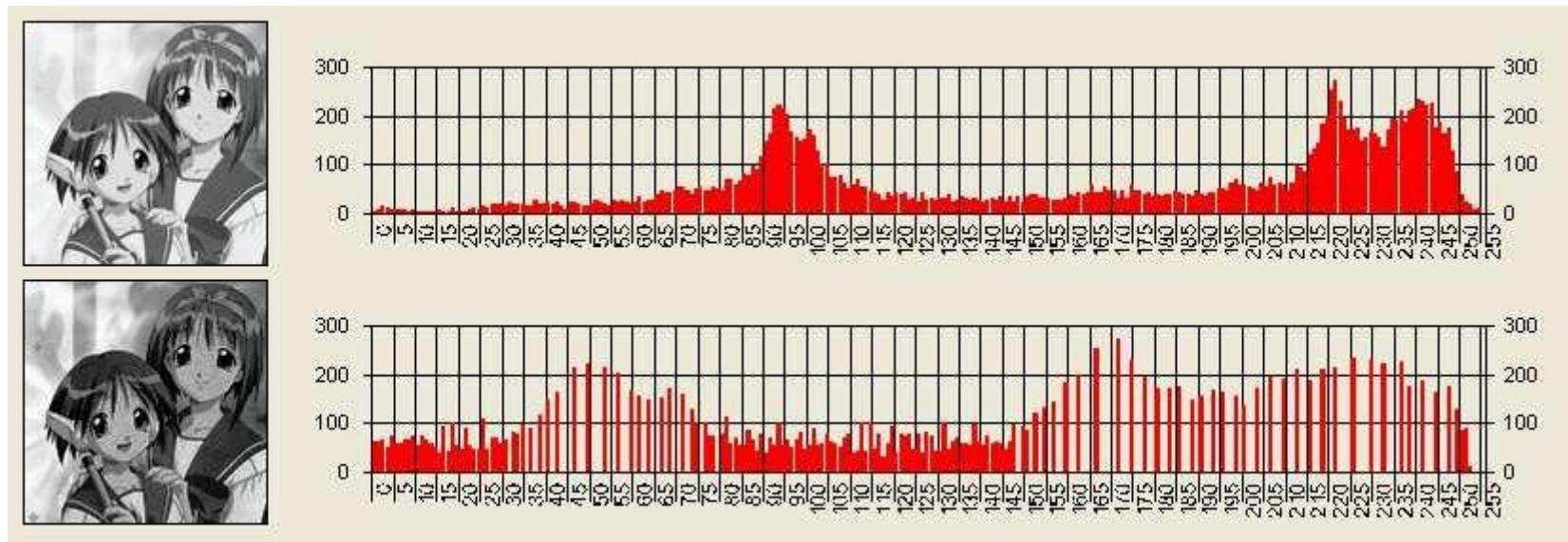
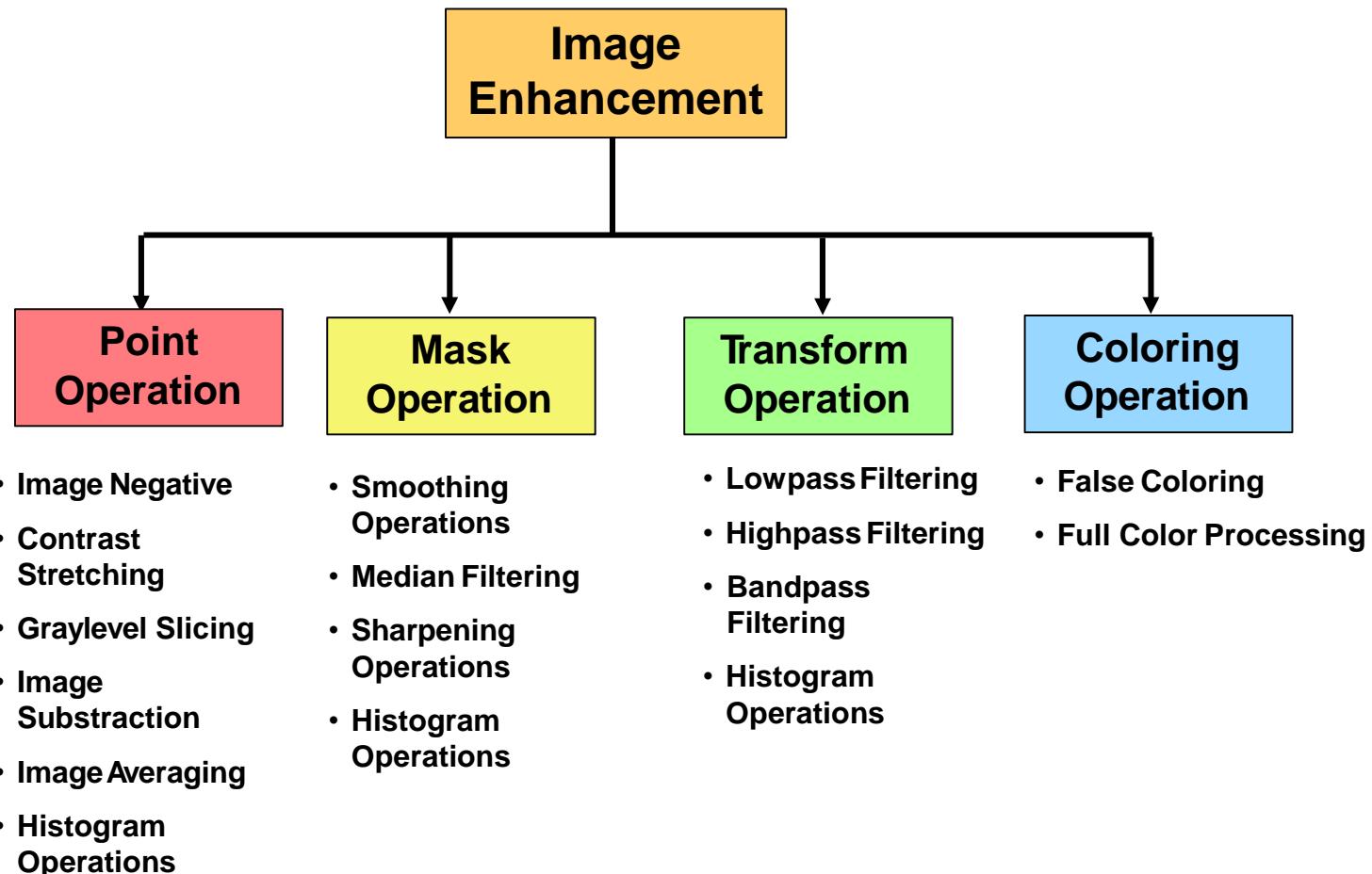


Image Enhancement

- **Image Enhancement adalah** proses agar citra menjadi lebih baik secara visual untuk aplikasi tertentu
- Proses sangat bergantung pada kebutuhan, dan pada keadaan citra input
- **Image Enhancement dapat dilakukan** dalam
 - **Spatial Domain** (dilakukan pada citra asli)
 - $g(m,n) = T[f(m,n)]$
 - **Frequency Domain** (dilakukan pada hasil DFT citra)
 - $G(u,v) = T[F(u,v)]$ dimana
 - $G(u,v) = F[g(m,n)]$ dan $F(u,v) = F[f(m,n)]$

Teknik Image Enhancement

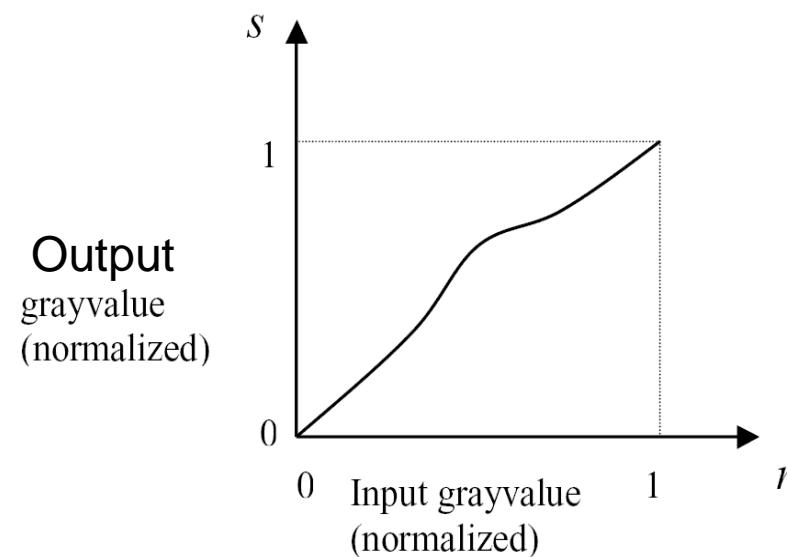


Point Operation

- Cara paling mudah untuk melakukan peningkatan mutu pada domain spasial
- Melibatkan satu piksel saja (tidak menggunakan jendela ketetanggaan)
- Pengolahan menggunakan histogram juga termasuk dalam bagian point processing

Point Operation

- Output pixel $g(m,n)$ hanya berdasar input sebuah pixel $f(m,n)$. Pixel tetangga tidak berpengaruh.
- Biasanya **dinotasikan** sebagai: $s=T(r)$
- Digambarkan seperti fungsi sbb:



Point Operation

□ **Image Negative**

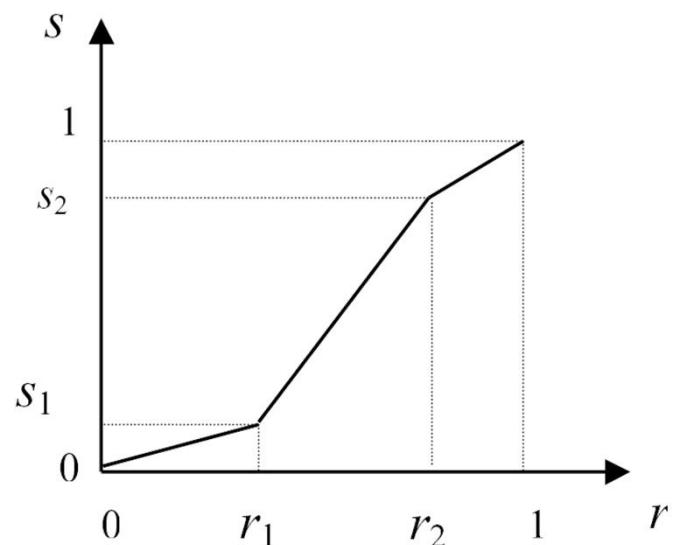
- Mengubah nilai gray-level piksel citra input dengan
$$G_{\text{baru}} = 255 - G_{\text{lama}}$$
- Hasilnya seperti klise foto



Point Operation

□ Contrast Stretching

- **Kontras adalah** tingkat penyebaran piksel-piksel ke dalam intensitas warna.
- Mengubah kontras dari suatu image dengan cara mengubah grey level piksel-piksel pada citra menurut fungsi $s = T(r)$ tertentu
 - $r_1 \leq r_2, s_1 \leq s_2$
 - $r_1 = r_2, s_1 = s_2 \rightarrow$ tidak ada perubahan
 - $r_1 = r_2, s_1 = 0, s_2 = 255 \rightarrow$ tresholding menjadi citra biner dengan ambang r_1



Point Operation

- **Contrast Stretching**
 - Salah satu fungsi kontras secara matematis adalah sbb:
$$f_o(x, y) = G \cdot (f_i(x, y) - P) + P$$
 - Dengan :
 - G : bilangan skalar, sebagai koefisien penguatan kontras
 - P : matriks konstan yang dipakai sebagai pusat pengontraskan
 - $F_o(x, y)$: matriks hasil kontras
 - $F_i(x, y)$: matriks citra asal

Point Operation

□ Contrast Stretching

- Contoh :
- Matriks ukuran 5x5 piksel akan dilakukan kontras dengan koefisien penguatan kontras G=2 dan pusat pengontrasan P=10. Perhitungan fungsi kontras dilakukan sebagai berikut :

$$2. \left[\begin{array}{ccccc} 15 & 30 & 40 & 50 & 45 \\ 25 & 25 & 25 & 40 & 30 \\ 40 & 50 & 35 & 45 & 35 \\ 50 & 40 & 15 & 25 & 40 \\ 30 & 40 & 20 & 25 & 20 \end{array} \right] - \left[\begin{array}{ccccc} 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \end{array} \right] + \left[\begin{array}{ccccc} 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \end{array} \right] = \left[\begin{array}{ccccc} 20 & 50 & 70 & 90 & 80 \\ 40 & 40 & 40 & 70 & 50 \\ 70 & 90 & 60 & 80 & 60 \\ 90 & 70 & 20 & 40 & 70 \\ 50 & 70 & 30 & 40 & 30 \end{array} \right]$$

Matriks Asal

Matriks Konstan

Matriks Konstan

Matriks Hasil

Point Operation

- **Contrast Stretching**
- Ada tiga macam kontras:
 - **Kontras Rendah**
 - Kontras ini terjadi karena kurangnya pencahayaan
 - Memiliki kurva histogram yang sempit (tepi paling kanan berdekatan dengan tepi paling kiri)
 - Artinya : titik tergelap dalam citra tidak mencapai hitam pekat dan titik paling terang tidak mencapai berwarna putih cemerlang
 - **Kontras Tinggi**
 - Memiliki kurva histogram yang terlalu lebar
 - Intensitas terang dan gelap merata ke seluruh skala intensitas
 - **Kontras Normal**
 - Lebar kurva histogram terlalu sempit dan tidak terlalu melebar

Point Operation

□ Contoh Contrast Stretching



Point Operation

Peregangan Kontras

□ Contoh Contrast Stretching

Kontras rendah



Kontras tinggi



Kontras bagus



Point Operation

□ **Thresholding (Ambang Batas)**

- **Thresholding adalah** proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih, sehingga dapat diketahui daerah mana yang termasuk objek dan background dari citra secara jelas.
- Citra hasil **thresholding** biasanya digunakan lebih lanjut untuk proses pengenalan objek serta ekstrasi fitur.
- Fungsi threshold :

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases}$$

- Dengan T : nilai threshold yang diberikan

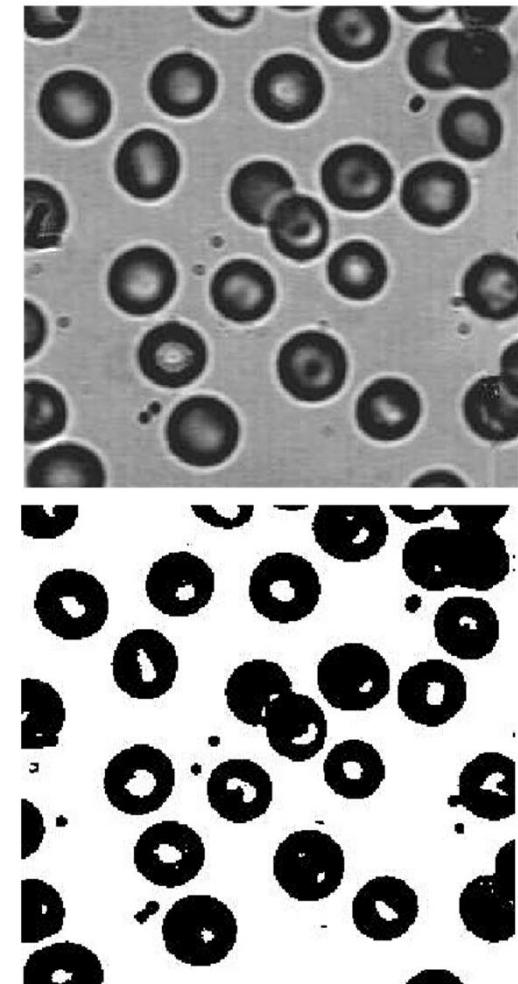
Point Operation

□ Contoh Thresholding

- Citra hitam putih yang memiliki *grayscale* 256, dipetakan menjadi citra biner (hanya mempunyai 2 warna, yaitu hitam dan putih), yang menggunakan fungsi transformasi sbb:

$$f_o(x,y) = \begin{cases} 0, f_i(x, y) < 128 \\ 255, f_i(x, y) \geq 128 \end{cases}$$

- Hasilnya adalah elemen-elemen matriks yang nilainya dibawah 128 diubah menjadi 0 (hitam), sedangkan elemen-elemen matriks yang nilainya diatas 128 diubah menjadi 255 (putih)



Point Operation

□ Contoh Perhitungan Thresholding

- Misalkan diketahui citra grayscale 256 warna dengan ukuran 5x5 piksel

40	160	69	170	123
20	250	140	80	90
70	30	128	115	85
40	234	70	211	125
20	34	80	221	30

- Maka perhitungan yang dilakukan adalah :
 - Setiap elemen-elemen matriks yang nilainya <128 diubah menjadi 0, sedangkan setiap elemen yang nilai >=128 diubah menjadi 255