

05 - Operasi-operasi Dasar Pengolahan Citra

IF4073 Interpretasi dan Pengolahan Citra

Oleh: Rinaldi Munir



Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2021

Aras operasi

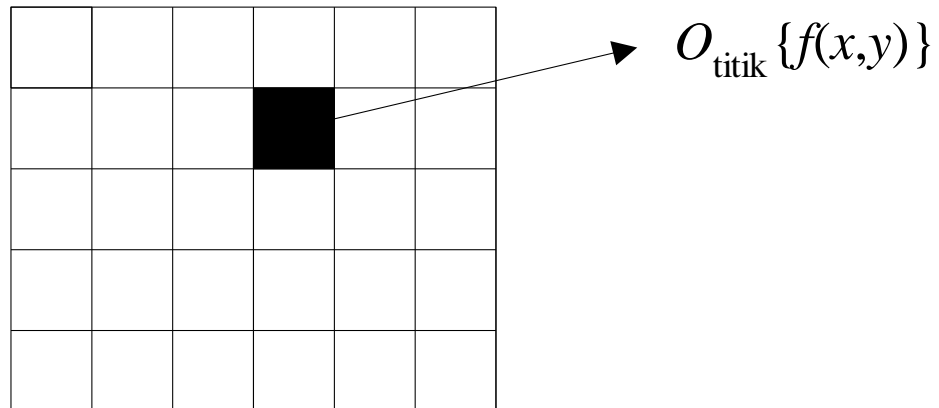
- Operasi-operasi yang dilakukan pada pengolahan citra dapat dikelompokkan ke dalam empat aras (*level*) komputasi:
 - aras titik
 - aras local
 - aras global
 - aras objek

1. Aras Titik

- Operasi pada aras titik hanya dilakukan pada *pixel* tunggal di dalam citra.

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

- Operasi ini diulangi untuk keseluruhan *pixel* di dalam citra.



Contoh-contoh operasi titik:

1. Pengambangan (*thresholding*)

$$f(x, y)' = \begin{cases} a_1, & f(x, y) < T \\ a_2, & f(x, y) \geq T \end{cases}$$

Jika $a_1 = 0$ dan $a_2 = 1$, \rightarrow transformasi citra hitam-putih ke **citra biner**

Contoh $T = 128$:

$$f(x, y)' = \begin{cases} 0, & f(x, y) < 128 \\ 1, & f(x, y) \geq 128 \end{cases}$$



```

void biner(citra A, citra_biner B, int T, int M, int N)
/* Membuat citra biner dari citra A berdasarkan nilai ambang
(threshold) T yang dispesifikasikan. Ukuran citra adalah M x N.
citra_biner adalah tipe data untuk citra biner).
*/
{ int i, j;
  citra_biner B;

  for (i=0; i<=M-1; i++)
    for (j=0; j<=N-1; j++)
    {
      if (A[i][j] < T)
        B[i][j] = 0;
      else
        B[i][j] = 1;
    }
}

```

Algoritma mengubah citra A menjadi citra biner.

2. Membuat citra negatif

- Seperti film negatif pada fotografi.
- Caranya: kurangi nilai intensitas *pixel* dari nilai keabuan maksimum.

Contoh pada citra *grayscale* 8-bit:

$$f(x, y)' = 255 - f(x, y)$$





```
void negatif(citra A, citra B, int M, int N)
/* Membuat citra negatif dari citra A. Hasilnya disimpan di
dalam citra B. Ukuran citra adalah M x N.
*/
{ int i, j;

  for (i=0; i<=M-1; i++)
    for (j=0; j<=N-1; j++)
    {
      B[i][j] = 255 - A[i][j];
    }
}
```


3. Pencerahan Citra (*image brightening*)

- Kecerahan citra dapat diperbaiki dengan menambahkan/mengurangkan sebuah konstanta kepada (atau dari) setiap *pixel*.

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

- Jika b positif, kecerahan citra bertambah,
Jika b negatif kecerahan citra berkurang
- Perlu operasi *clipping* jika nilai $f(x, y)'$ berada di bawah nilai intensitas minimum atau di atas nilai intensitas maksimum:

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



Gambar Kiri: citra Zelda (agak gelap); **kanan:** citra Zelda setelah operasi pencerahan citra, $b = 100$

```
void image_brightening(citra A, int b, citra B, int M, int N)
/* Pencerahan citra dengan menjumlahkan setiap pixel di dalam citra A dengan
sebuah skalar b. Hasil disimpan di dalam citra B. Citra berukuran M x N. */
{ int i, j, temp;

  for (i=0; i<=M-1; i++)
    for (j=0; j<=N-1; j++)
    {
      temp = A[i][j] + b;

      /* clipping */
      if (temp < 0)
        B[i][j] = 0;
      else
        if (temp > 255)
          B[i][j]=255;
        else
          B[i][j]=temp;
    }
}
```

4. Konversi citra berwarna ke citra *grayscale*

- Konversi citra dengan kanal RGB menjadi satu kanal Y (*luminance*)
- Cara paling sederhana: $Y = (R + G + B)/3$
- Hasil yang lebih baik: $Y = 0.299R + 0.587G + 0.144B$



$$Y = (R + G + B)/3$$



Kurang bagus!



$$Y = 0.299R + 0.587G + 0.144B$$



Lebih bagus!

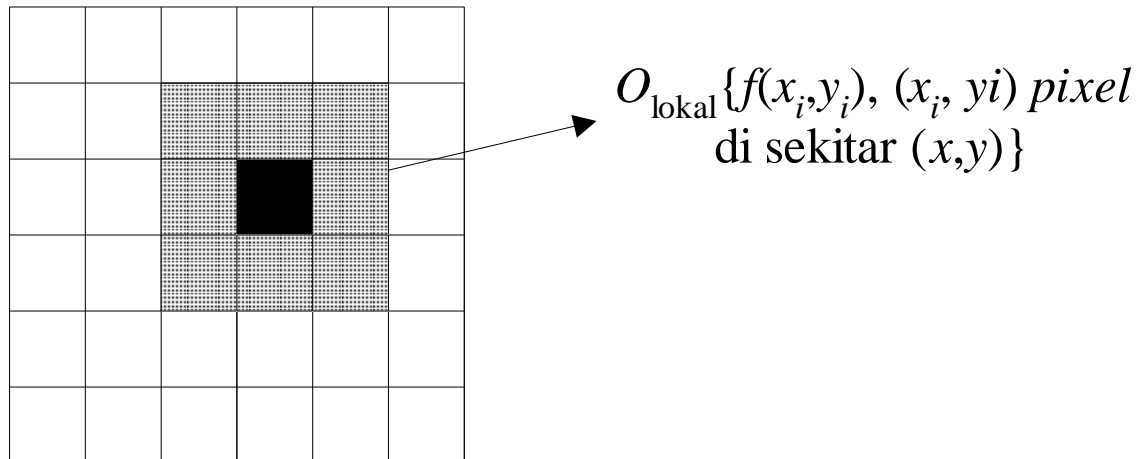


2. Aras Lokal

- Operasi pada aras lokal menghasilkan citra luaran yang intensitas *pixel-nya* bergantung pada intensitas *pixel-pixel* tetangganya

$$f_B(x, y)' = O_{\text{lokal}} \{ f_A(x_i, y_j); (x_i, y_j) \in N(x, y) \}$$

(keterangan: $N = \text{neighborhood}$, yaitu *pixel-pixel* yang berada di sekitar (x, y))



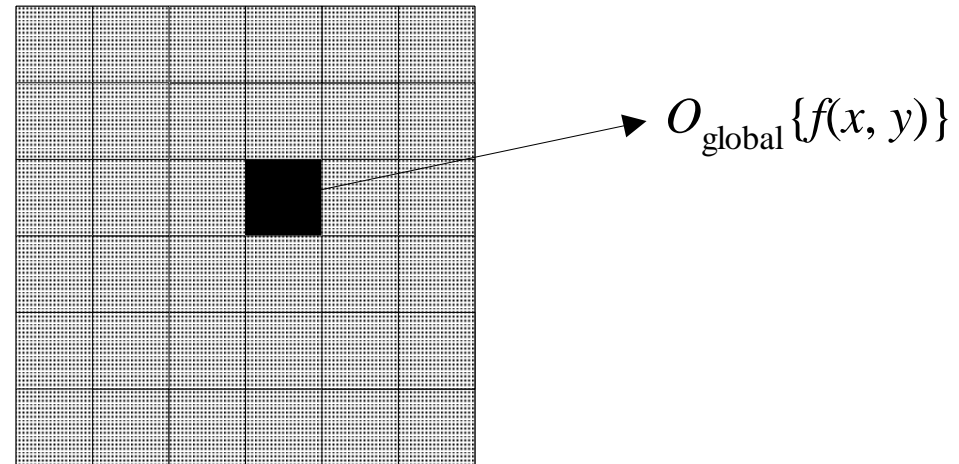
- Contoh operasi beraras lokal adalah operasi konvolusi untuk mendeteksi tepi (*edge detection*) dan pelembutan citra (*image smoothing*). Akan dijelaskan pada kuliah selanjutnya.



3. Aras Global

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

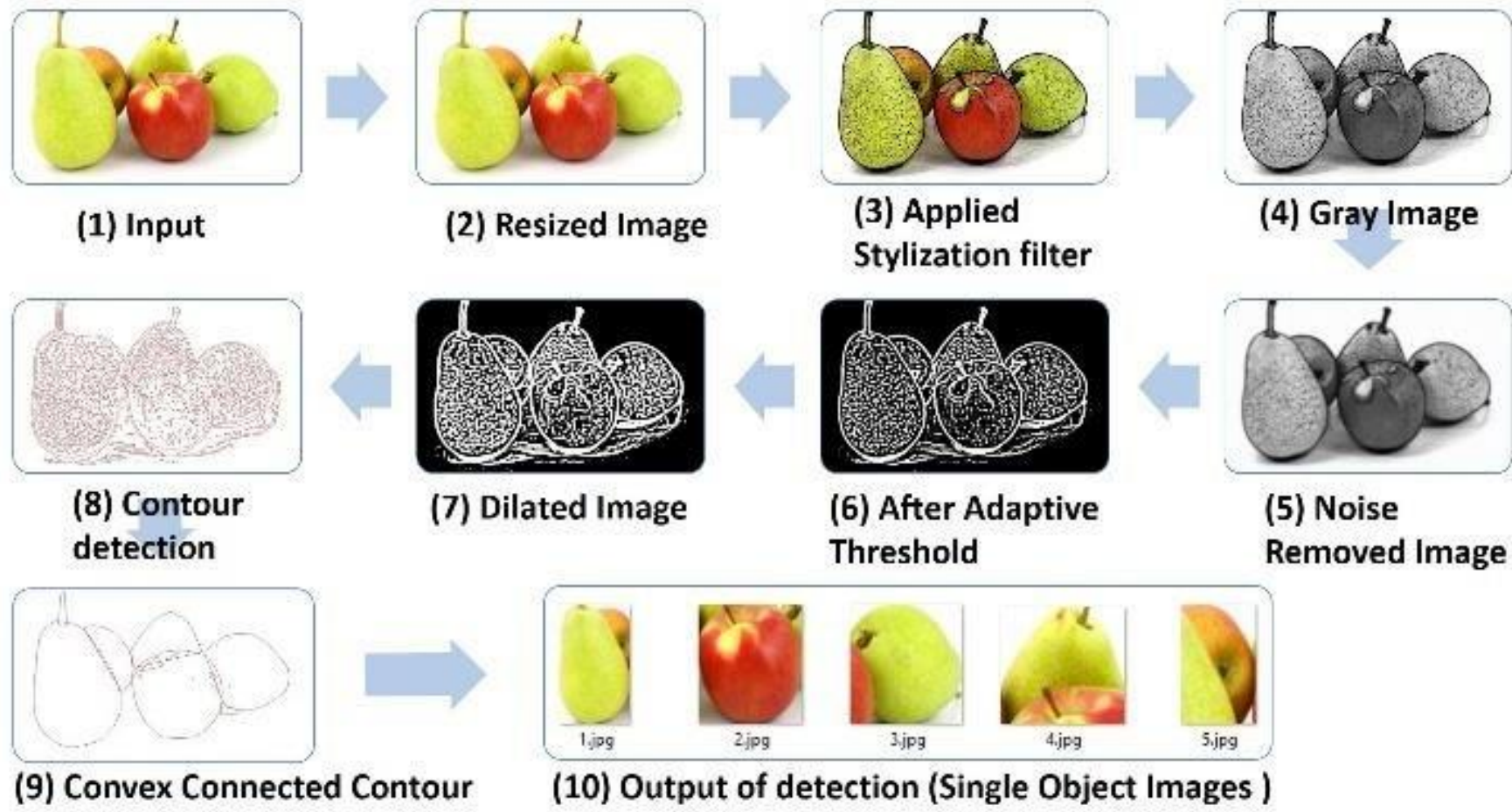
$$f_B(x, y)' = O_{\text{global}} \{f_A(x, y)\}$$



- Contoh operasi beraras global adalah operasi penyetaraan histogram untuk meningkatkan kualitas citra → akan dibahas pada kuliah selanjutnya.

4. Aras Objek

- Operasi jenis ini hanya dilakukan pada objek tertentu di dalam citra.
- Tujuan dari operasi pada aras objek adalah untuk mendeteksi objek dan mengenali objek tersebut, misalnya dengan menghitung ukuran, bentuk, dan karakteristik lain dari objek.
- Operasi aras objek adalah operasi yang kompleks, karena sebelumnya kita harus dapat menjawab: apakah objek itu, bagaimana menemukannya?



Sumber: Rafflesia Khan & Rameswar Debnath, *Multi class fruit classification using efficient object detection and recognition techniques*

Operasi Aritmetika

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

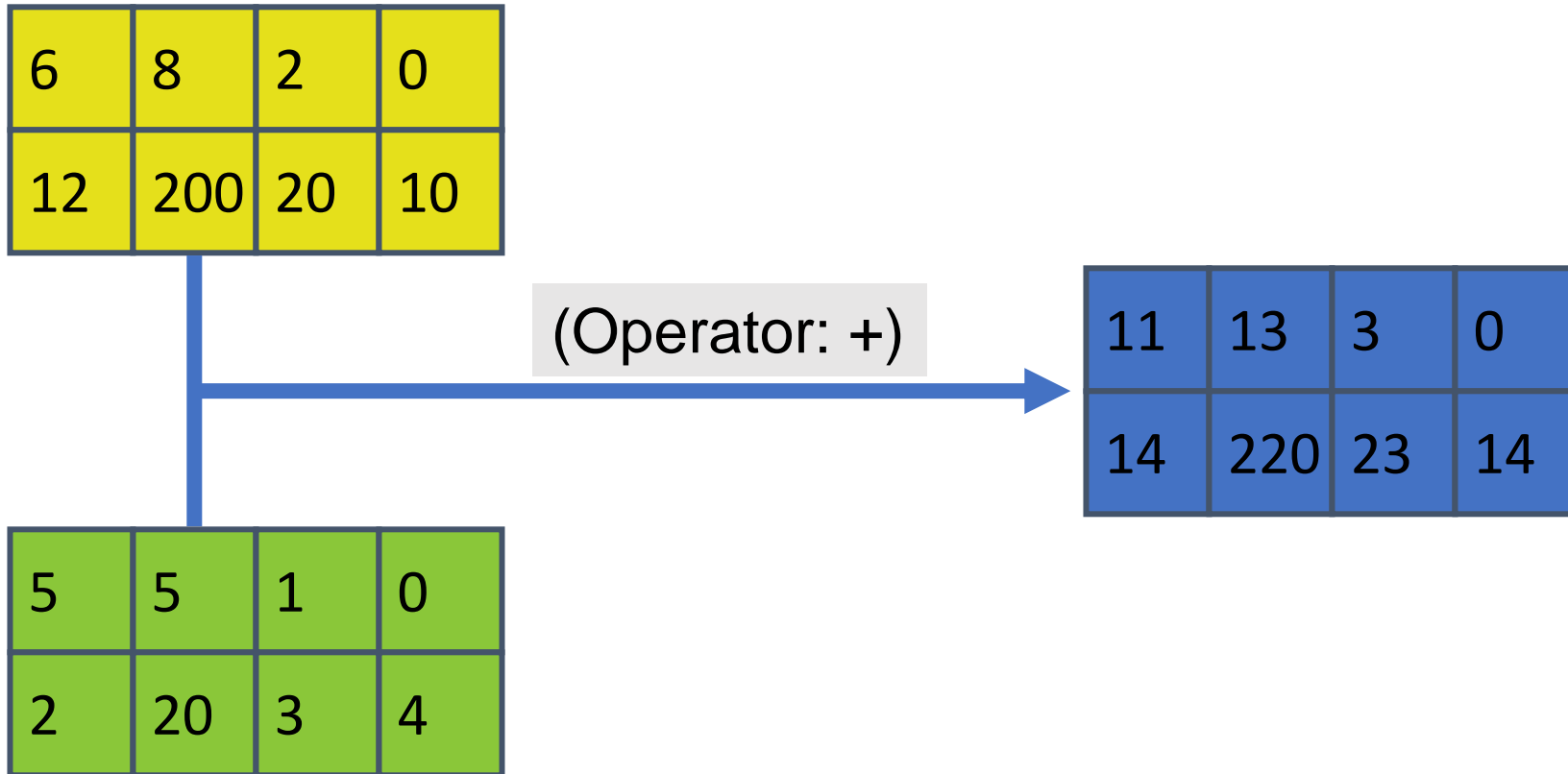
- Penjumlahan/pengurangan citra: $C(x, y) = A(x, y) \pm B(x, y)$
- Perkalian citra: $C(x, y) = A(x, y) .* B(x, y)$
- Penjumlahan/pengurangan citra A dengan skalar c: $B(x, y) = A(x, y) \pm c$
- Perkalian/pembagian citra A dengan sebuah skalar c: $B(x, y) = c \cdot A(x, y)$

Penjumlahan citra

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

```
void addition(citra A, citra B, citra C, int M, int N)
/* Menjumlahkan dua buah citra A dan B menjadi citra baru, C.
   Citra A, B, dan C masing-masing berukuran M x N.
*/
{ int i, j, temp;

  for (i=0; i<=M-1; i++)
    for (j=0; j<=N-1; j++)
      {
        temp=A[i][j] + B[i][j];
        if (temp > 255) C[i][j]=255; else C[i][j]=temp;
      }
}
```

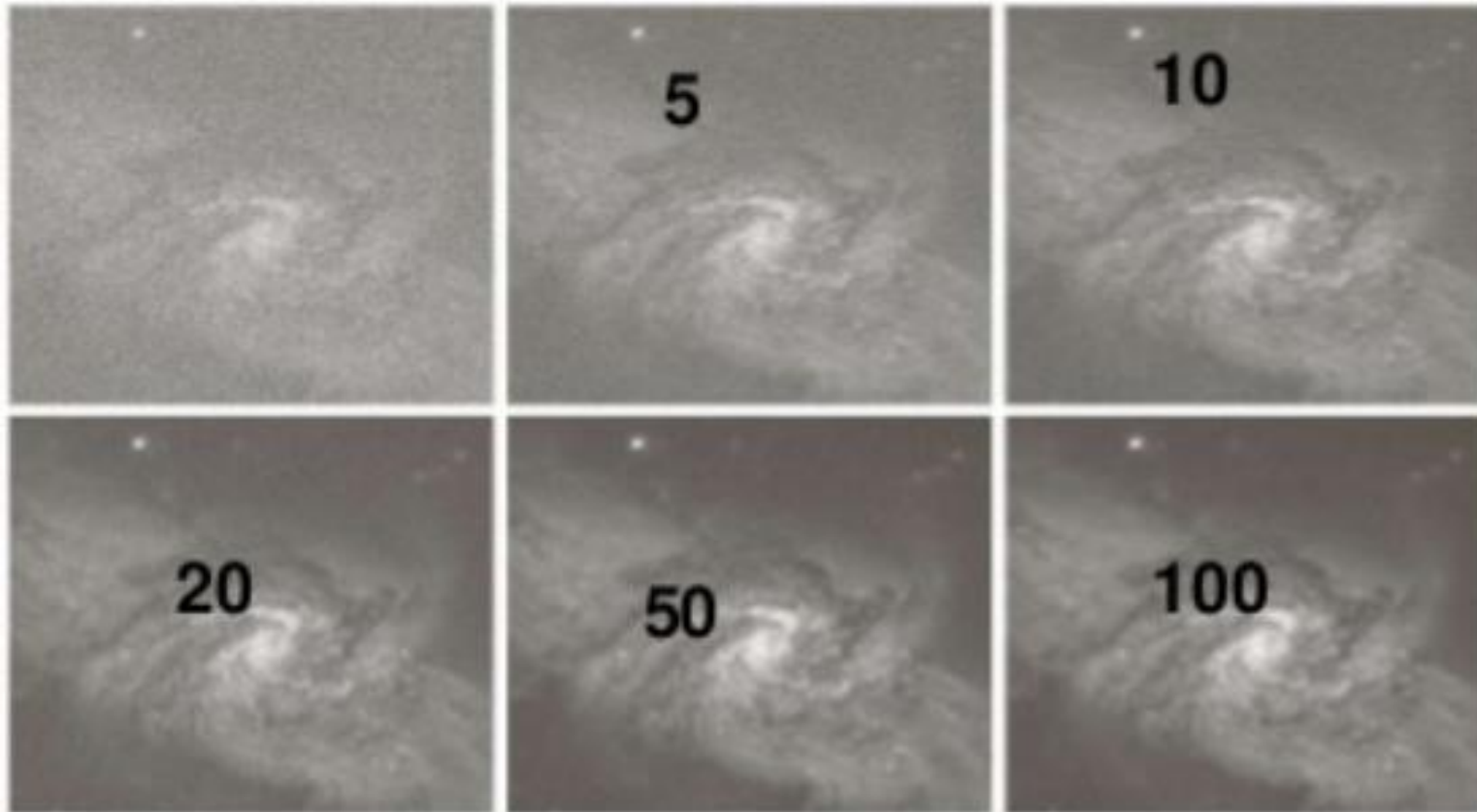


Sumber gambar: Dr. Rolf Lakaemper, CIS 601 Image ENHANCEMENT in the SPATIAL DOMAIN

- Contoh penjumlahan dua buah citra: mengurangi pengaruh derau (*noise*) di dalam data, dengan cara merata-ratakan derajat keabuan setiap *pixel* dari citra yang sama yang diambil berkali-kali.
- Misalnya untuk citra yang sama direkam dua kali, f_1 dan f_2 , lalu dihitung intensitas rata-rata untuk setiap *pixel*

$$f'(x,y) = \frac{1}{2} \{ f_1(x,y) + f_2(x,y) \}$$

- Pada aplikasi astronomi, kualitas citra dapat ditingkatkan dengan merata-ratakan sejumlah citra



Sumber: *Digital Image Processing*, Amity Scholl of Engineering and Technology

- Penjumlahan citra dapat digunakan untuk menambahkan komponen dari sebuah citra ke citra lain.



+



Dalam kode Matlab:

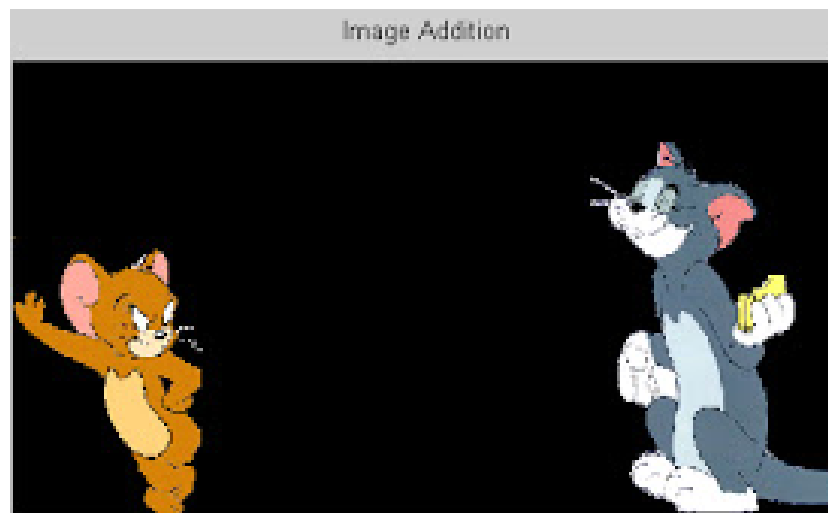
```
A=imread('tommy1.bmp');
```

```
B=imread('jerry1.bmp');
```

```
%Image addition
```

```
%Both A and B are of same size
```

```
object=A+B;
```



Sumber: <https://www.imageprocessing.com/2011/05/image-arithmetic.html>

$$O(r, c) = \alpha I_1(r, c) + (1 - \alpha)I_2(r, c)$$

$$0 \leq \alpha \leq 1$$



Sumber: *Digital Image Processing*, Amity Scholl of Engineering and Technology

Pengurangan citra

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

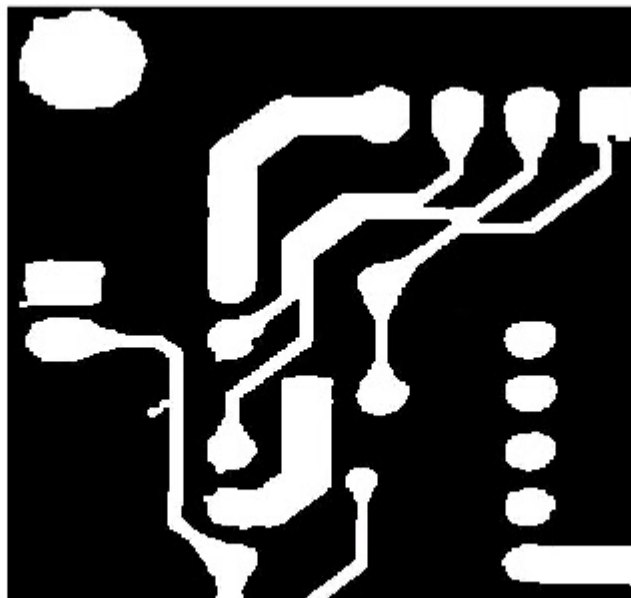
```
void subtraction (citra A, citra B, citra C, int M, int N)
/* Mengurangkan dua buah citra A dan B menjadi citra baru, C.
   Citra A, B, dan C berukuran M x N.
*/
{ int i, j;

  for (i=0; i<=M-1; i++)
    for (j=0; j<=N-1; j++)
      {
        C[i][j]=A[i][j] - B[i][j];
        if (C[i][j] != 0) C[i][j]=255; /* nyatakan objek berwarna putih */
      }
}
```

- Pengurangan citra banyak digunakan untuk deteksi perubahan. Misalnya untuk mendeteksi komponen yang hilang dalam perakitan produk, contohnya untuk mendeteksi cacat pada PCB.



Subtracted Image



Original Image

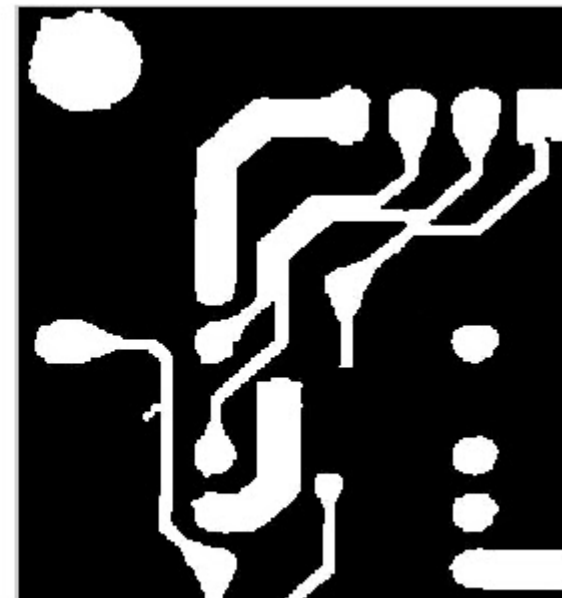
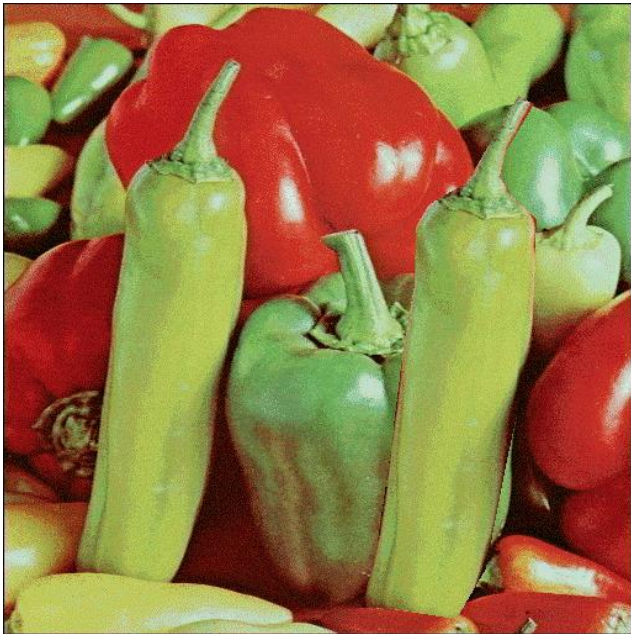


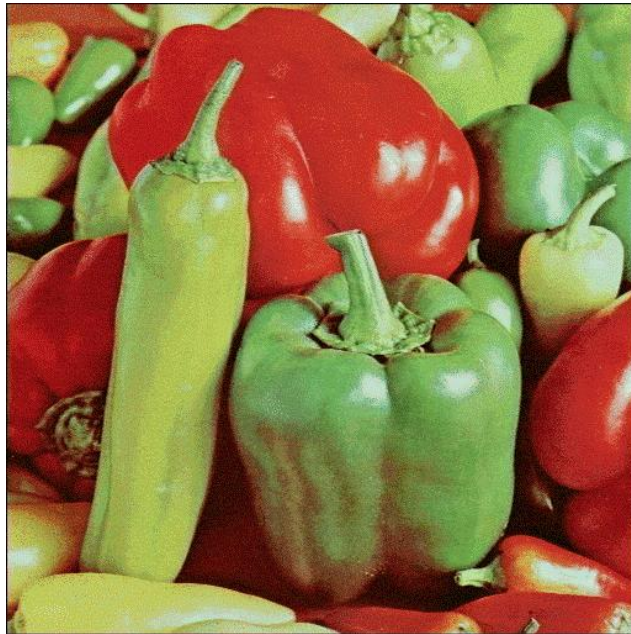
Image with defects

Sumber: <https://www.imageprocessing.com/2011/05/image-arithmetic.html>

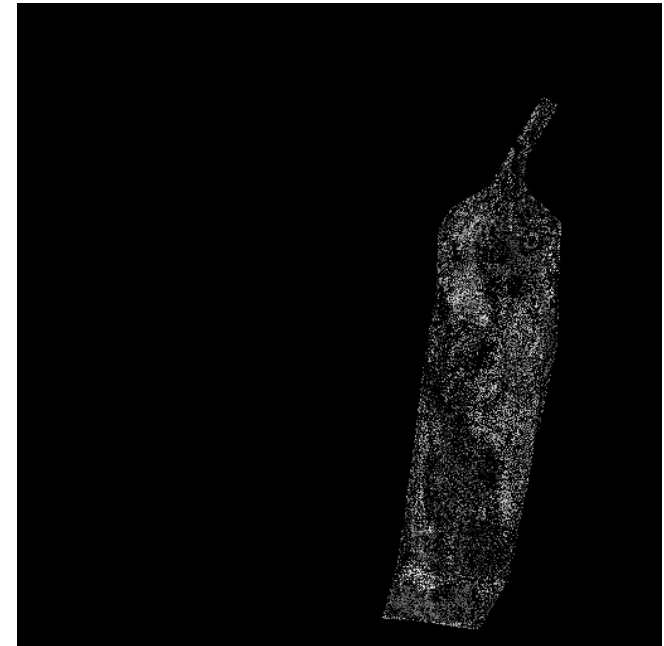
Pengurangan citra untuk mendeteksi objek tambahan di dalam gambar:



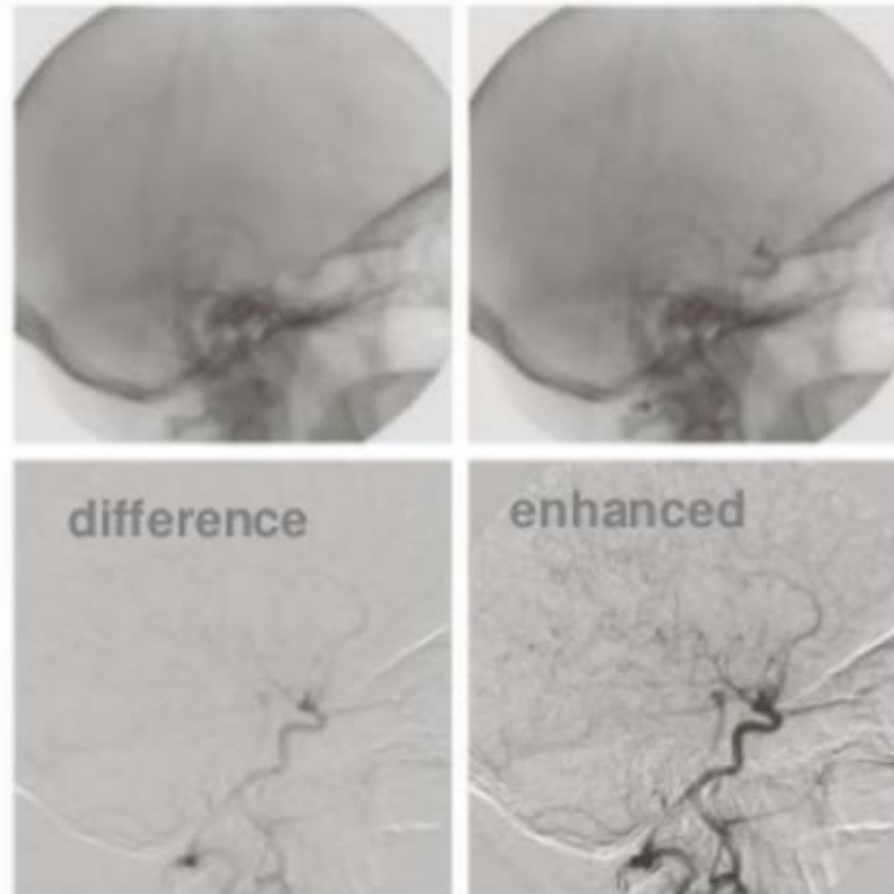
-



=



- Pada aplikasi medis, pengurangan citra berguna untuk mendeteksi kelainan pada sebuah citra medis.



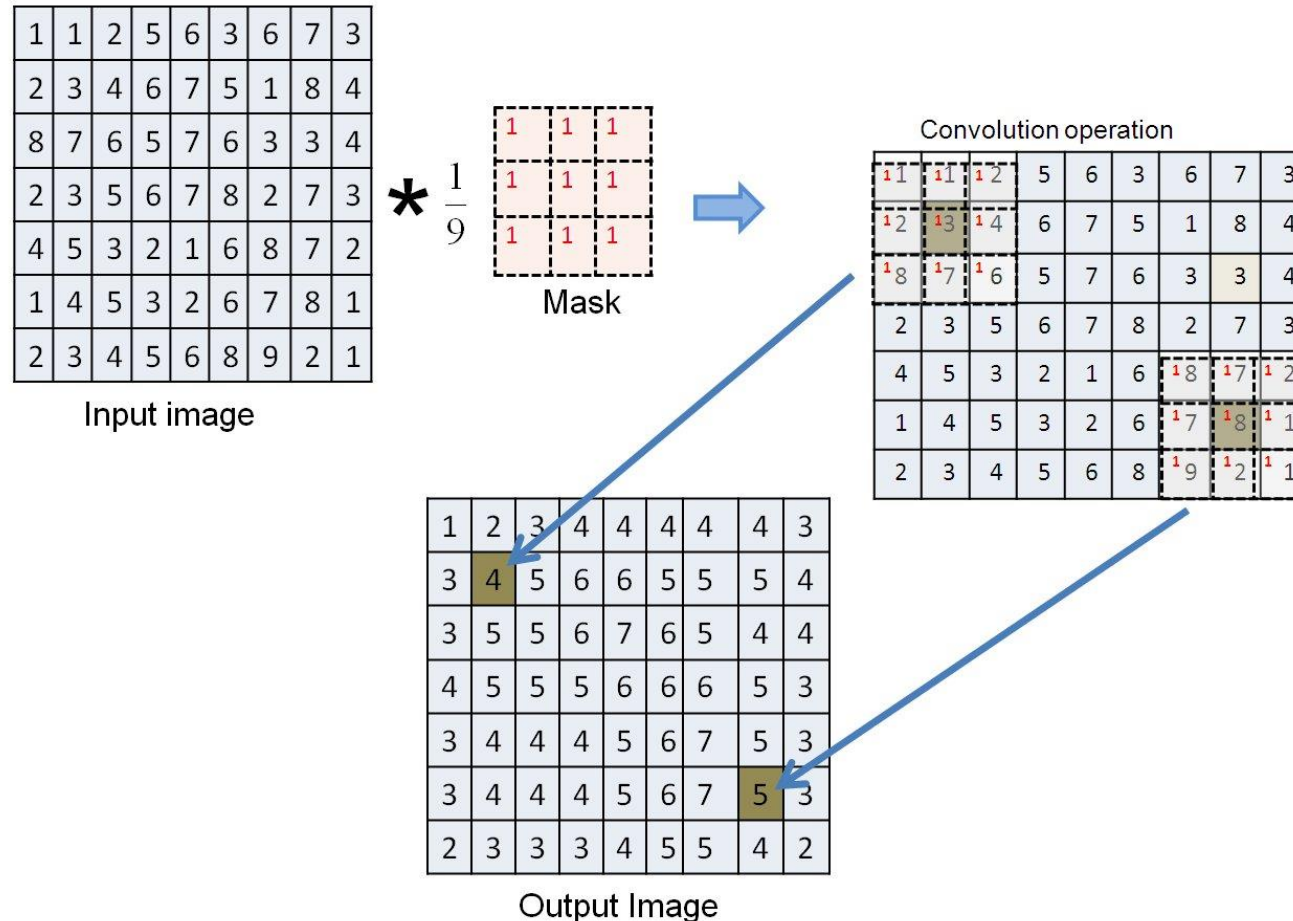
Perkalian citra

$C(x, y) = A(x, y) \cdot B(x, y)$ → mengalikan *pixel-pixel* pada posisi yang bersesuaian

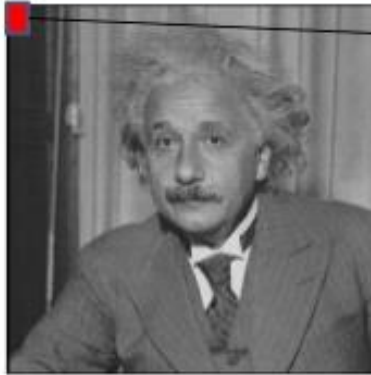
```
void multiplication(citra A, citra B, citra C, int M, int N)
/* Mengalikan citra A dengan citra B menjadi citra C.
   Citra A, matriks B, dan hasil perkalian C berukuran M x N.
*/
{ int i, j;

  for (i=0; i<=M-1; i++)
    for (j=0; j<=N-1; j++)
      for (k=0; k<=N-1; k++)
        { C[i][j] = A[i][j]*B[i][j];
          /* clipping */
          if (C[i][j] < 0)
            C[i][j] = 0;
          else
            if (C[i][j] > 255)
              C[i][j]=255;
        }
}
```

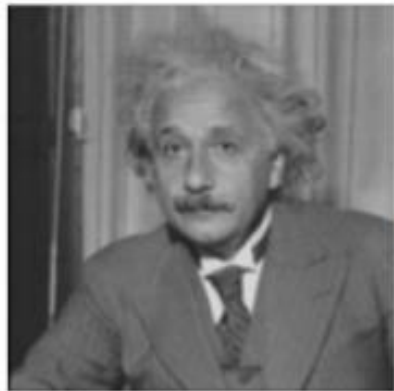
- Perkalian citra umumnya digunakan di dalam penapisan citra (*image masking* atau *image filtering*) → mengalikan citra dengan sebuah *mask* yang berukuran lebih kecil dari ukuran citra



Mask Operation



0	3	0	0
0	6	1	16
0	0	2	46
0	0	2	43



X

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

DIGITAL IMAGE PROCESSING

Penjumlahan/pengurangan citra dengan skalar

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

Contoh: *Image brightening*



$$B(x, y) = A(x, y) + 100$$

Nilai sebagian *pixel* sebelum pencerahan:

108	108	108	107	107	107	107	106	106	106
107	107	107	107	107	107	107	106	106	106
107	107	107	107	106	106	106	106	106	106
107	107	107	107	106	106	105	106	107	106
107	108	108	107	107	107	107	106	106	107
108	108	108	108	106	106	106	107	107	107
108	108	108	108	106	106	106	107	107	107
108	108	108	108	106	106	106	107	107	106
108	108	108	107	106	106	107	107	106	106
108	108	108	108	107	107	106	106	107	107

Nilai sebagian *pixel* setelah pencerahan (+ 100):

208	208	208	207	207	207	207	206	206	206
207	207	207	207	207	207	207	206	206	206
207	207	207	207	206	206	206	206	206	206
207	207	207	207	206	206	205	206	207	206
207	208	208	207	207	207	207	206	206	207
208	208	208	208	206	206	206	207	207	207
208	208	208	208	206	206	206	207	207	207
208	208	208	208	206	206	206	207	207	206
208	208	208	207	206	206	207	207	206	206
208	208	208	208	207	207	206	206	207	207

Perkalian/pembagian Citra dengan Skalar

$$B(x, y) = c \cdot A(x, y), \text{ dan } B(x, y) = A(x, y) / c$$

- Operasi perkalian citra dengan skalar dipakai untuk kalibrasi kecerahan (*calibration of brightness*).
- Operasi pembagian citra dengan skalar dipakai untuk normalisasi kecerahan (*normalization of brightness*).

Contoh perkalian citra dengan skalar:



Citra semula

$$B(x,y) = 3 \cdot A(x,y)$$



Citra setelah setiap pixel dikali dengan 3

Sumber: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/pixmult.htm>

Operasi Boolean

- Operasi AND: $C(x, y) = A(x, y) \text{ and } B(x, y)$
- Operasi OR: $C(x, y) = A(x, y) \text{ or } B(x, y)$
- Operasi NOT: $C(x, y) = \text{not } A(x, y)$
- Operasi XOR: $C(x, y) = A(x, y) \text{ xor } B(x, y)$

Logical Operations for Binary Images

Binary Image - Image A



Binary Image - Image B



Complement of Image A



Image A XOR Image B



Image A OR Image B



Image A AND Image B



Activate Windows
Go to Settings to activate Windows.

- Operasi NOT dapat digunakan untuk menghasilkan komplemen dari sebuah citra
- Contoh operasi NOT:



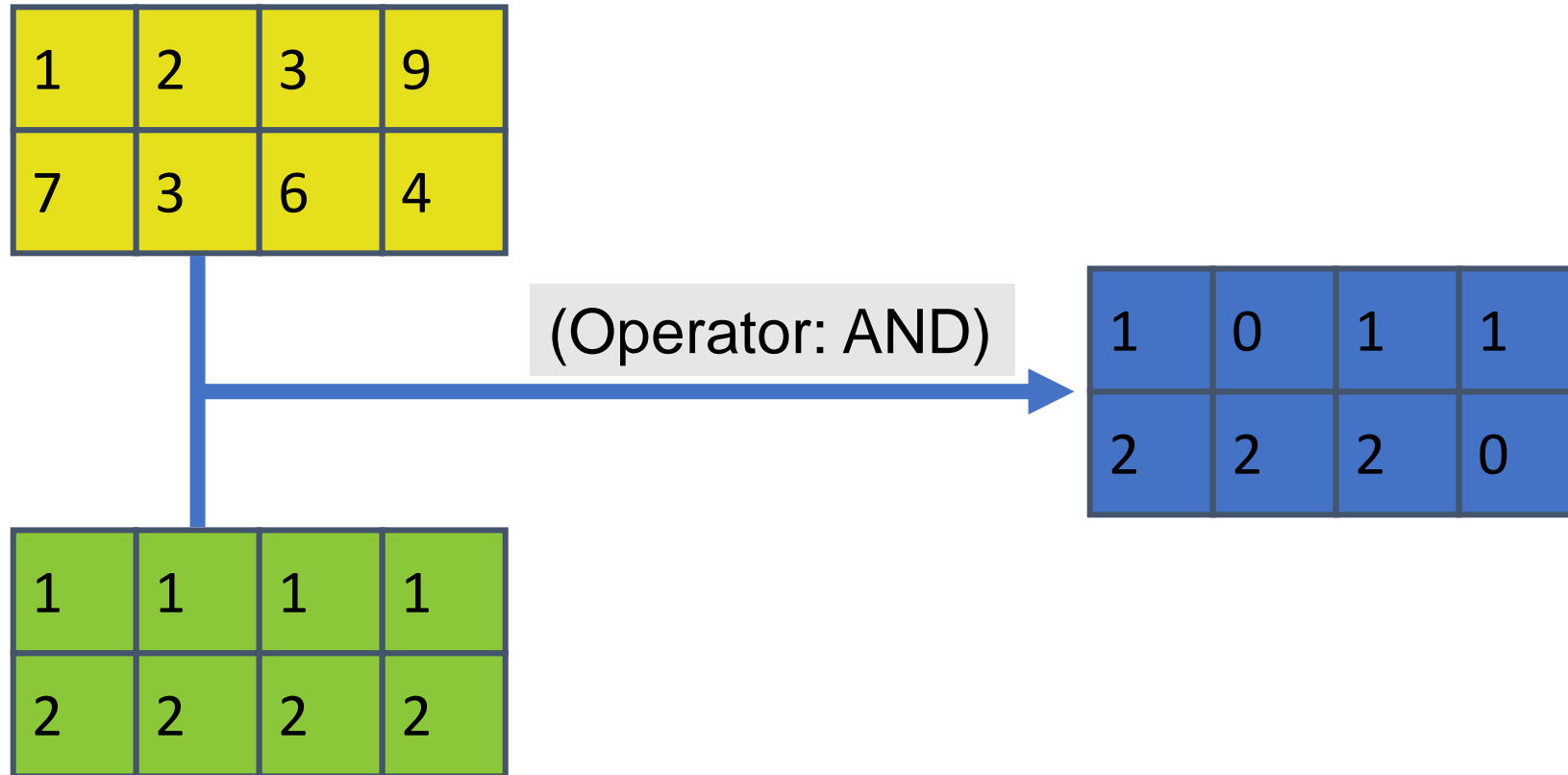
Ganesha



NOT Ganesha

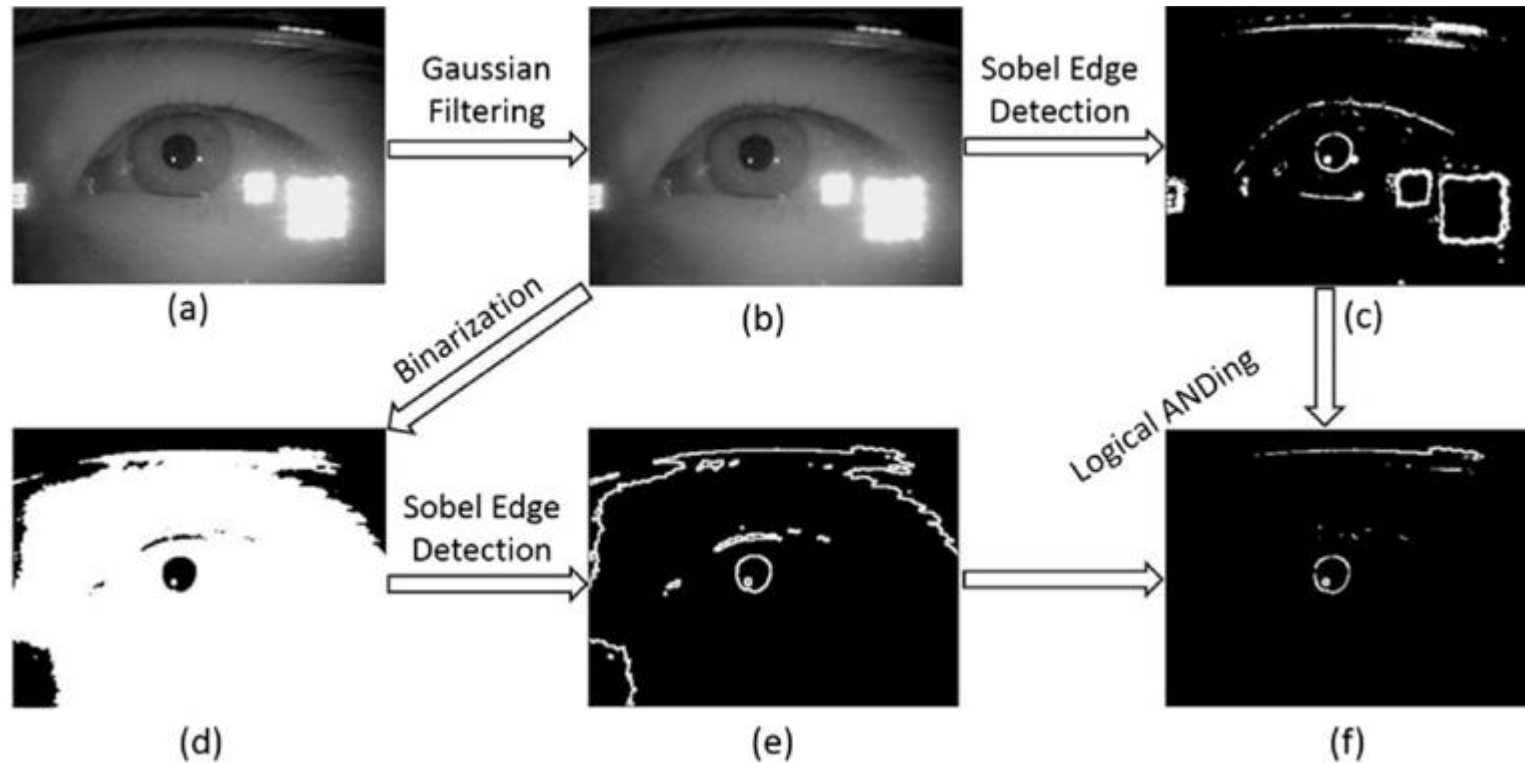
```
void not(citra_biner A, citra_biner B, int M, int N)
/* Membuat citra komplemen dari citra biner A.
Komplemennya disimpan di dalam B. Ukuran citra A
adalah M x N.
*/
{ int i, j;

  for (i=0; i<=M-1; i++)
    for (j=0; j<=N-1; j++)
    {
      B[i][j] = !A[i][j];
    }
}
```



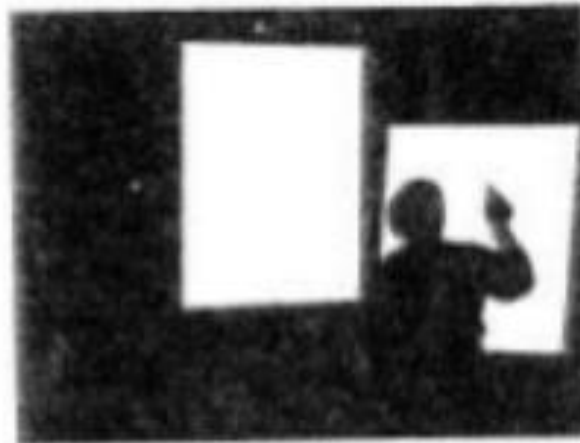
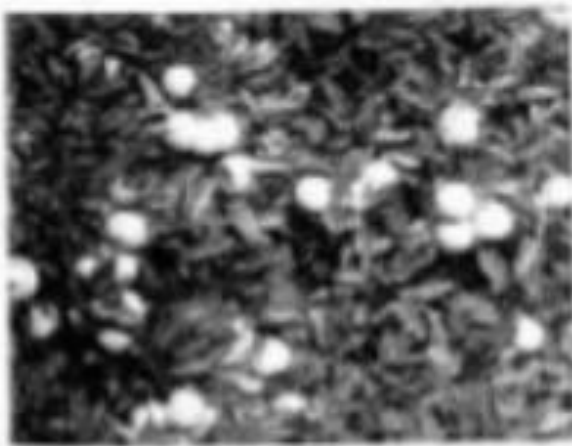
Sumber gambar: Dr. Rolf Lakaemper, CIS 601 Image ENHANCEMENT in the SPATIAL DOMAIN

- Operasi AND dan OR dapat digunakan untuk memilih *subimage* dari sebuah *image*



Sumber: Neeti Taneja, Er. Mohammad Shabaz, Vinayak Khajuria, *Iris Detection Using Segmentation Techniques*

- Operasi OR dapat digunakan untuk menambahkan bagian citra lain ke sebuah citra tujuan.

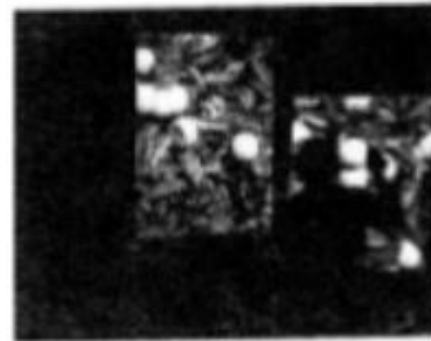


Sumber: *Digital Image Processing*, Amity Scholl of Engineering and Technology

Result of AND



Result of OR



OR



Sumber: *Digital Image Processing*, Amity Scholl of Engineering and Technology

Operasi Geometri

- Operasi geometri: translasi, rotasi, penskalaan citra, pencerminan citra (*flipping*).
- Pengubahan geometri dari citra $f(x, y)$ menjadi citra baru $f'(x, y)$ dapat ditulis sbb:

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

yang dalam hal ini, $g_1(x)$ dan $g_2(y)$ adalah fungsi transformasi geometrik. Dengan kata lain,

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

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

1. Translasi

Rumus translasi sejauh m dalam arah x dan n dalam arah y :

$$x' = x + m$$

$$y' = y + n$$

Translasi citra A menjadi citra B sejauh m dalam arah x dan n dalam arah y :

$$B[x][y] = A[x + m][y + n]$$

```
void translation(citra A, citra B, int M, int N, int m, int n)
/* Mentranslasi citra A sejauh m, n menjadi citra B. Ukuran citra M x N. */
{ int i, j;

  for (i=0; i<=M-1; i++) `
    for (j=0; j<=N-1; j++)
    {
      B[i][j]=A[i+m][j+n];
    }
}
```



Translasi pada citra *camera*: (a) citra semula, (b) citra hasil translasi dengan $m = 30$ dan $n = 25$.

2. Rotasi

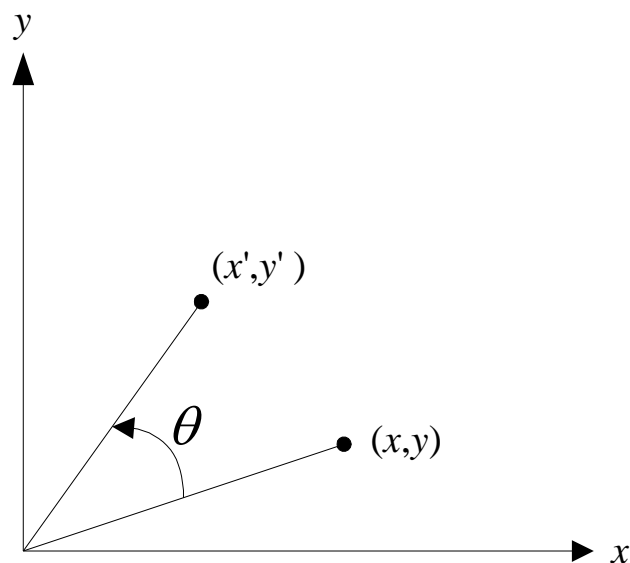
Rumus rotasi citra sejauh θ radian berlawanan arah jarum jam :

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

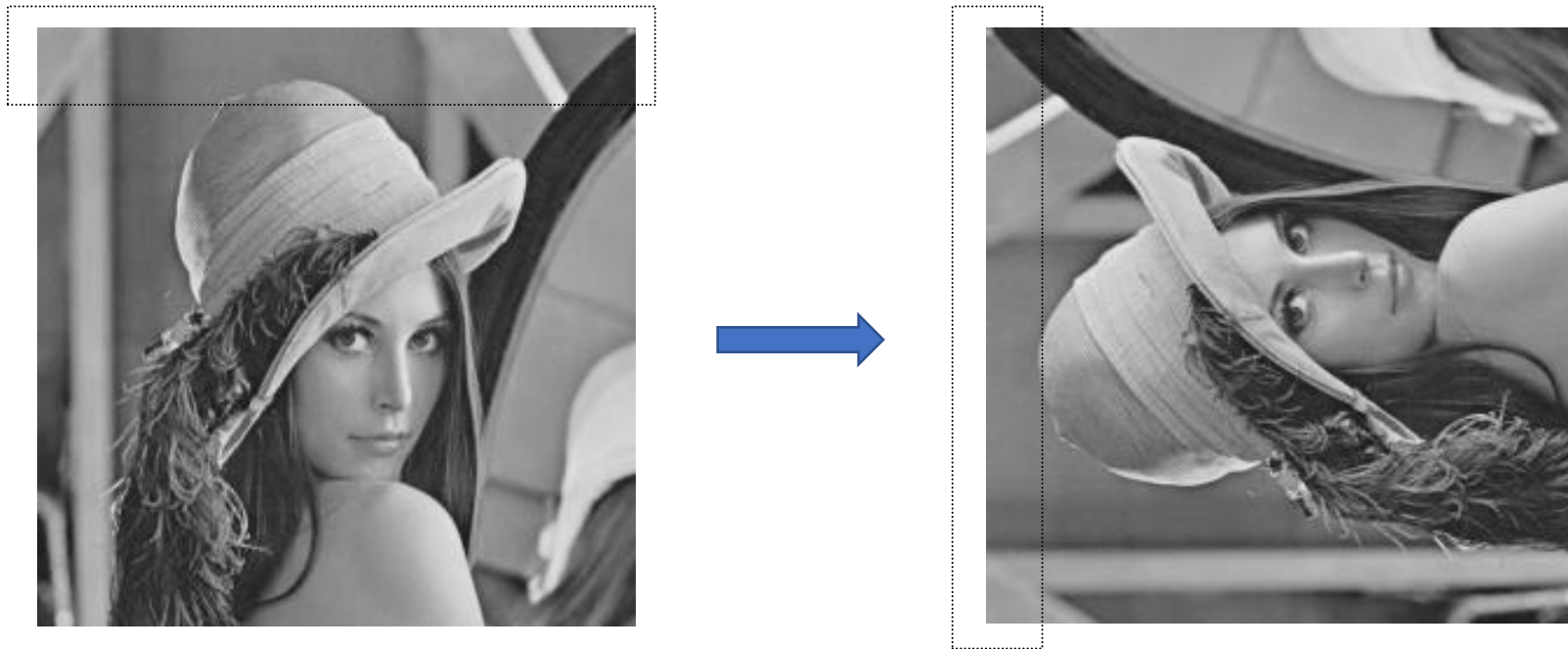
$$y' = x \sin(\theta) + y \cos(\theta)$$

Rotasi citra A menjadi citra B sejauh θ radian berlawanan arah jarum jam :

$$B[x'] [y'] = B[x \cos(\theta) - y \sin(\theta)] [x \sin(\theta) + y \cos(\theta)] = A[x] [y]$$



- Jika sudut rotasinya 90° , maka implementasinya lebih mudah dilakukan dengan cara menyalin *pixel-pixel* baris ke *pixel-pixel* kolom pada arah rotasi.
- Rotasi 180° diimplementasikan dengan melakukan rotasi 90° dua kali.



Gambar rotasi citra Lena sejauh 90° berlawanan arah jarum jam


```
void rotation90CCW(citra A, citra B, int M, int N)
/* Rotasi citra A sejauh 90° berlawanan arah jarum jam (CCW = Clock Counter-
wise). Ukuran citra adalah M x N. Hasil rotasi disimpan di dalam citra B.
*/
{ int i, j, k;

  for (i=0; i<=M-1; i++)
  {
    k=N-1;
    for (j=0; j<=N-1; j++)
    {
      B[k][i]=A[i][j];
      k--;
    }
  }
}
```

```
void rotation90CW(citra A, citra B, int M, int N)
/* Rotasi citra A sejauh 90° searah jarum jam (CW = Clock-wise).
   Ukuran citra adalah M x N. Hasil rotasi disimpan di dalam cira B.
*/
{ int i, j, k;

  k=N-1;
  for (i=0; i<=M-1; i++)
  {
    for (j=0; j<=N-1; j++)
    {
      B[j][k]=A[i][j];
    }
    k--;
  }
}
```

3. Flipping

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



(a) citra



(b) *flip* horizontal



(c) *flip* vertikal

- *Flipping* horizontal: pencerminan pada sumbu- Y (*cartesian*) dari citra A menjadi citra B :

$$B[x][y] = A[N - x][y]$$

- *Flipping* vertical: pencerminan pada sumbu- X (*cartesian*) dari citra A menjadi citra B :

$$B[x][y] = A[x][M - y]$$

- Pencerminan pada titik asal (*cartesian*) dari citra A menjadi citra B :

$$B[x][y] = A[N - x][M - y]$$

- Pencerminan pada garis $x = y$ dari citra A menjadi citra B :

$$B[x][y] = A[y][x]$$

```
void vertical_flip(citra A, citra B, int M, int N)
/* Flipping vertikal (pencerminan terhadap sumbu-X) terhadap citra A. */
   Ukuran citra adalah  $M \times N$ . Hasil flipping disimpan di dalam citra B.
*/
{ int i, j, k;

  k=N-1;
  for (i=0; i<=M-1; i++)
  {
    for (j=0; j<=N-1; j++)
    {
      B[k][j]=A[i][j];
    }
    k--;
  }

}
```

4. Penskalaan citra (*image zooming*)

- Pengubahan ukuran citra (membesar/*zoom out* atau mengecil/*zoom in*).
- Rumus penskalaan citra:

$$x' = s_x \cdot x$$

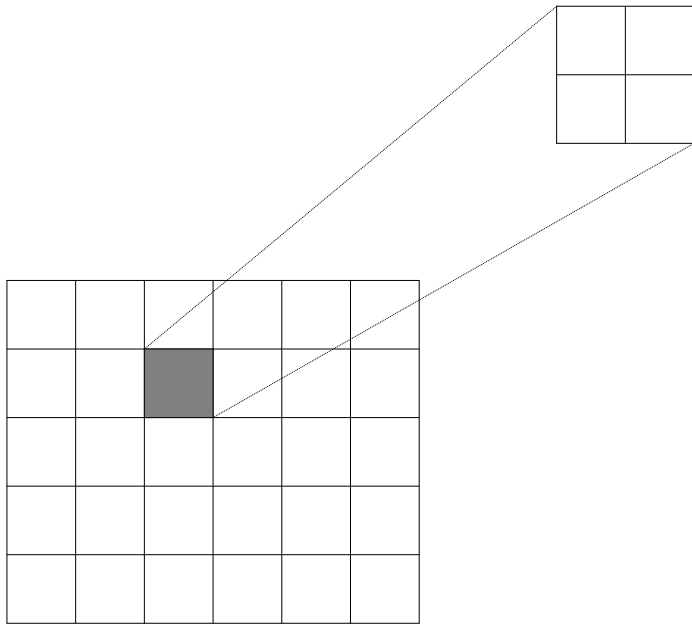
$$y' = s_y \cdot y$$

s_x dan s_y adalah faktor skala masing-masing dalam arah x dan arah y .

- Penskalaan citra A menjadi citra B :

$$B[x'][y'] = A[s_x \cdot x][s_y \cdot y]$$

- Operasi *zoom out* dengan faktor 2 (yaitu, $s_x = s_y = 2$) diimplementasikan dengan menyalin setiap *pixel* sebanyak 4 kali.



Gambar Kiri: citra kota San Fransisco (ukuran normal),
Kanan: citra kota San Fransisco setelah diperbesar 2 kali ($s_x = s_y = 2$):

```
void zoom_out(citra A, citra B, int M, int N)
/* perbesaran citra A dengan faktor skala 2
   Ukuran citra adalah  $M \times N$ . Hasil perbesaran disimpan dalam citra B.
*/
{ int i, j, k, m, n;

  m=0; n=0;
  for (i=0; i<=M-1; i++)
  {
    for (j=0; j<=N-1; j++)
    {
      B[m][n]= A[i][j];
      B[m][n+1]= A[i][j];
      B[m+1][n]= A[i][j];
      B[m+1][n+1]= A[i][j];
      n=n+2;
    }
    m=m+2;
    n=0;
  }
}
```


- Operasi *zoom in* (pengecilan) dengan faktor skala = $\frac{1}{2}$ dilakukan dengan mengambil rata-rata dari 4 *pixel* yang bertetangga menjadi 1 *pixel*

