

Operasi-operasi Dasar Pengolahan Citra

Aras operasi

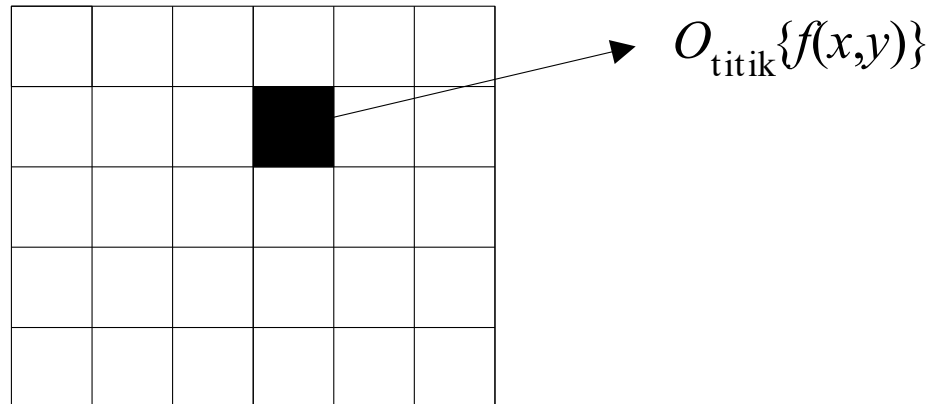
- Operasi-operasi yang dilakukan pada pengolahan citra dapat dikelompokkan ke dalam empat aras (*level*) komputasi:
 - aras titik
 - aras local
 - aras global

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. Pengembangan (*thresholding*)

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

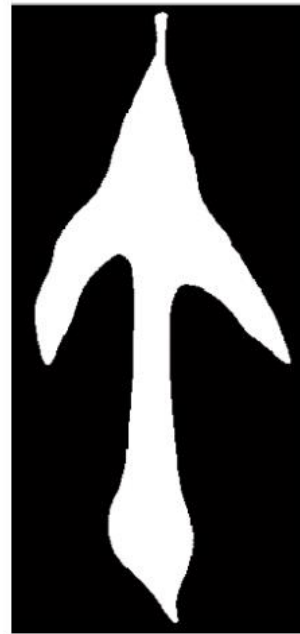
Jika $a_1 = 0$ dan $a_2 = 1$, $\rightarrow$ transformasi citra hitam-putih ke **citra biner**

Contoh $T = 128$:

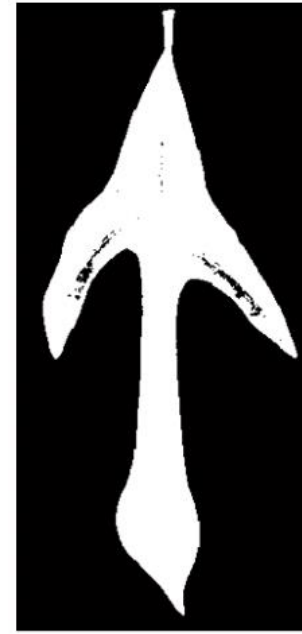
$$f(x, y)' = \begin{cases} 0, & f(x, y) < 128 \\ 1, & f(x, y) \geq 128 \end{cases}$$



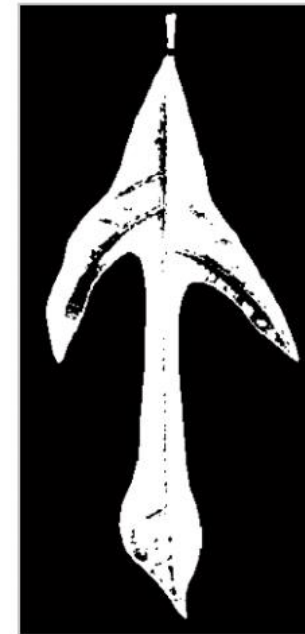
Perbedaan nilai threshold



Ambang = 210



Ambang = 160



Ambang = 130



Ambang = 110



Ambang = 100



Ambang = 70

```

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

- Caranya: kurangi nilai intensitas *pixel* da

Contoh pada citra *grayscale* :

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



```
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  $M \times N$ .
*/
{ 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



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  $M \times N$ . */
{ 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*

- 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!

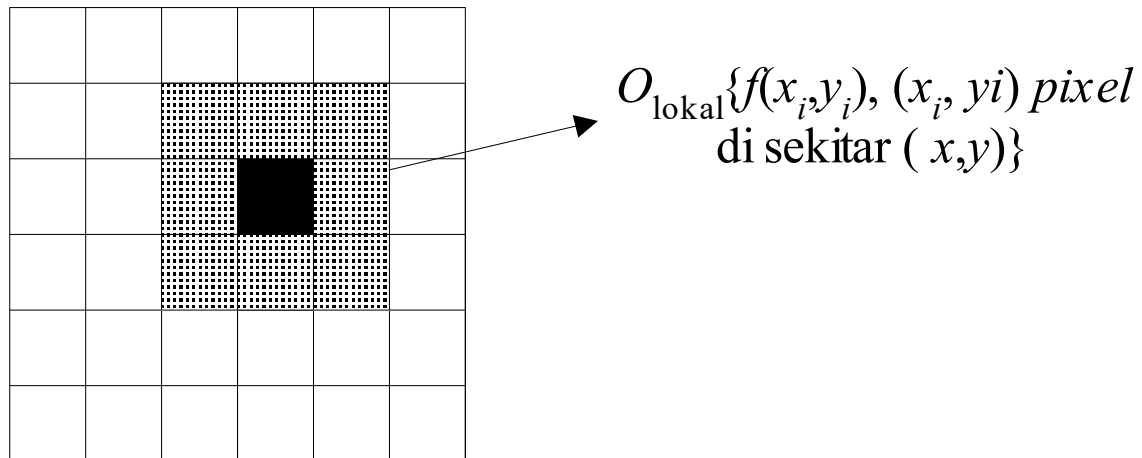


2. Aras Lokal

- 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))

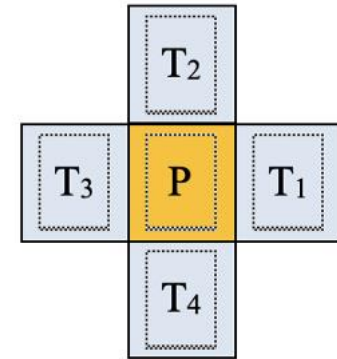


Tetangga Piksel

- Suatu piksel p pada koordinat (x,y) memiliki empat tetangga horisontal dan vertikal dengan koordinat sebagai berikut:

$$(x+1,y), (x-1,y), (x,y+1), (x,y-1)$$

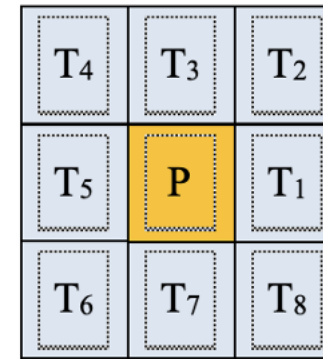
Himpunan piksel tetangga disebut tetangga-4 dari p dan dinyatakan dengan $N_4(p)$.



(a) 4-ketetanggaan

Tetangga Piksel

- Empat tetangga diagonal dari p memiliki koordinat sebagai berikut :
 $(x+1,y+1), (x+1,y-1), (x-1,y+1), (x-1,y-1)$
Dan dinyatakan dengan $N_D(p)$.
- $N_D(p)$ bersama-sama dengan $N_4(p)$ disebut tetangga-8 dari p , dan dinyatakan dengan $N_8(p)$.



(b) 8-ketetanggaan

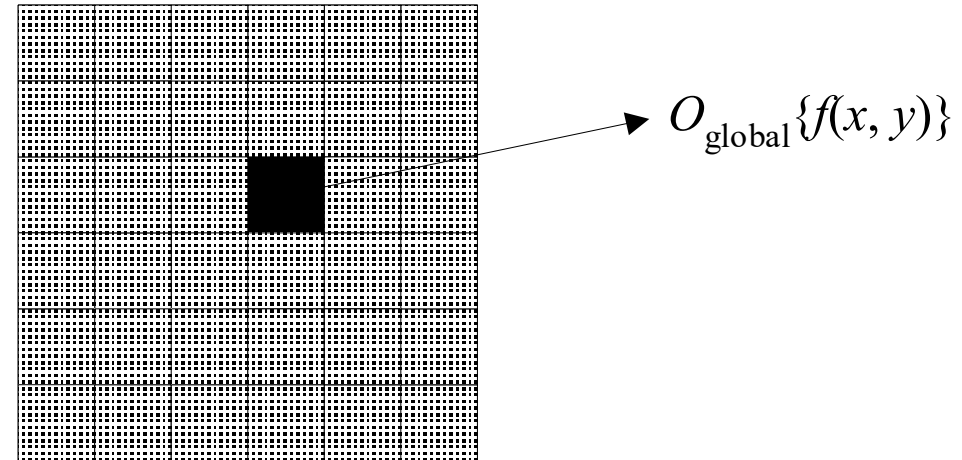
- Contoh operasi beraras lokal adalah operasi konvolusi untuk mendeteksi tepi (*edge detection*) dan pelembutan citra (*image smoothing*).



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

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;
    }
}
```

6	8	2	0
12	200	20	10

(Operator: +)

11	13	3	0
14	220	23	14

5	5	1	0
2	20	3	4

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*

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

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



+



Dalam kode Matlab:

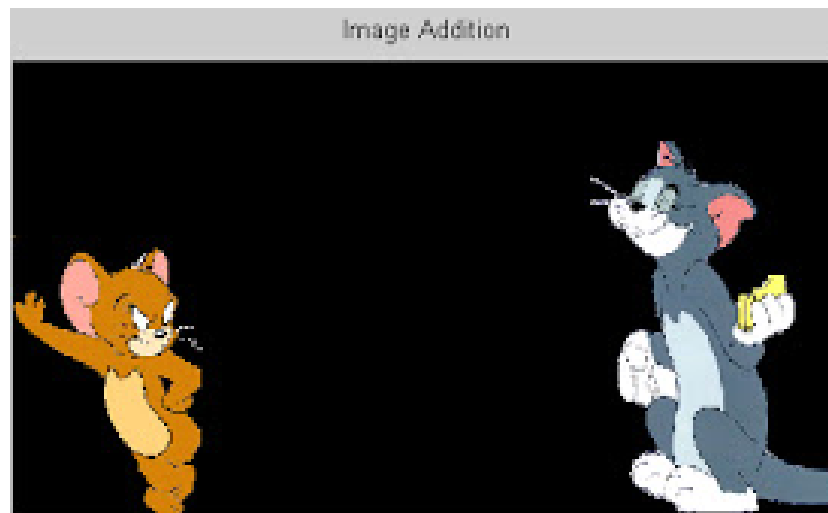
```
A=imread('tommy1.bmp');
```

```
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>



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

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

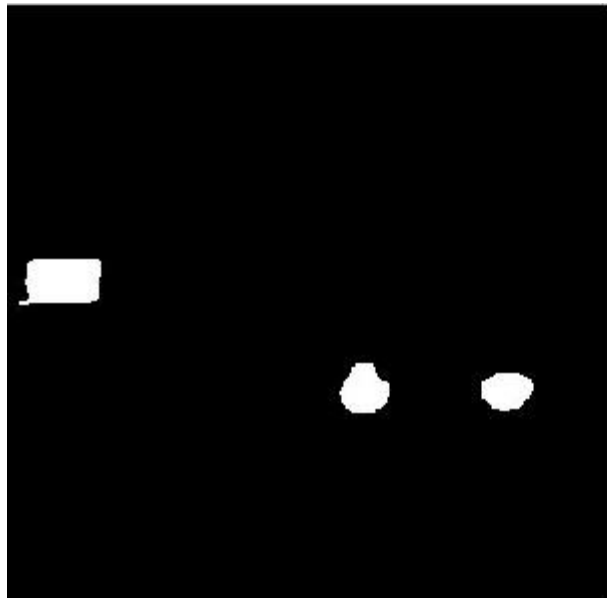
Pengurangan citra

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

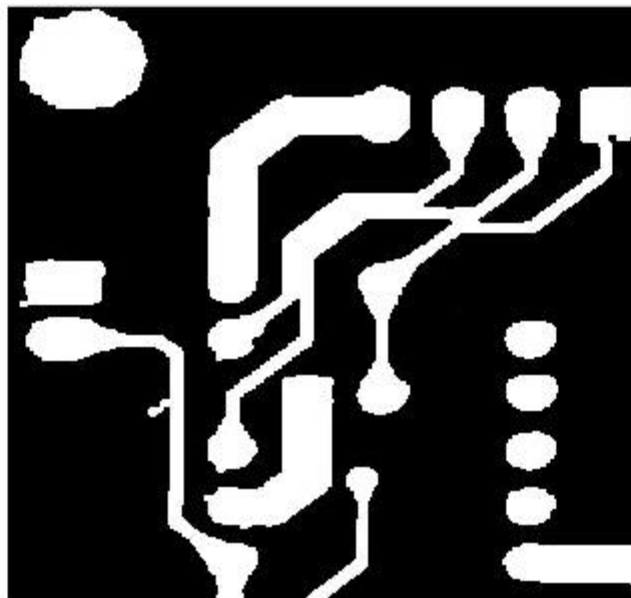
```
void subtraction (citra A, citra B, citra C, int M, int N)
/* Mengurangkan dua buah citra A dan B menjadi citra baru, C.
   Citra A, B, dan C berukuran M x N.
*/
{ int i, j;

  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

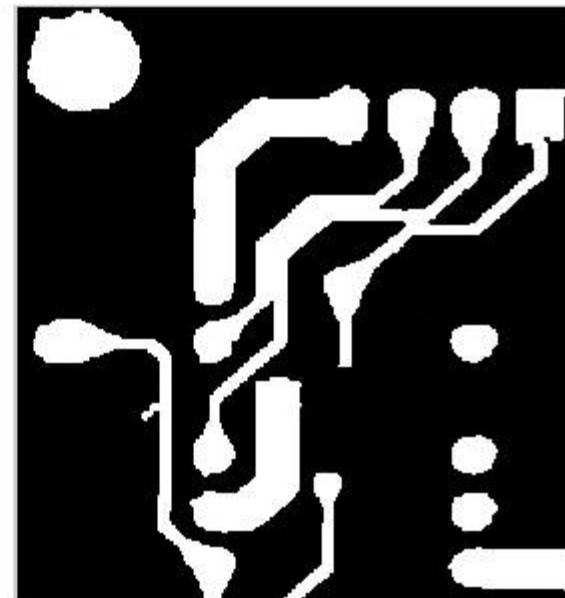
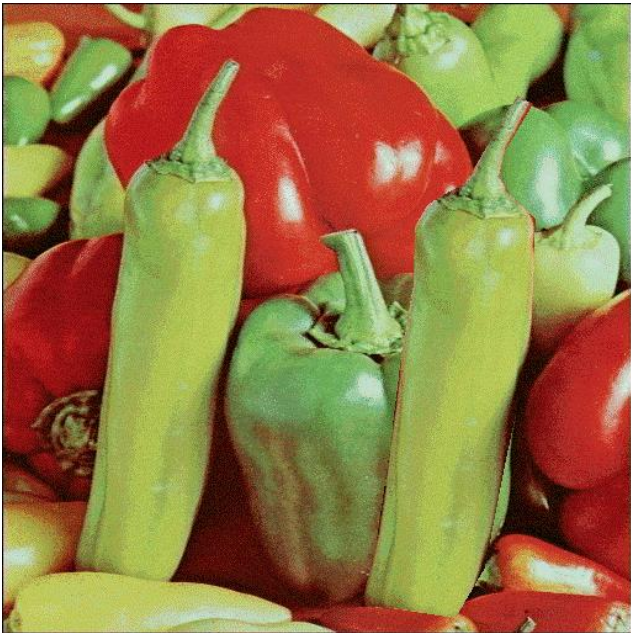


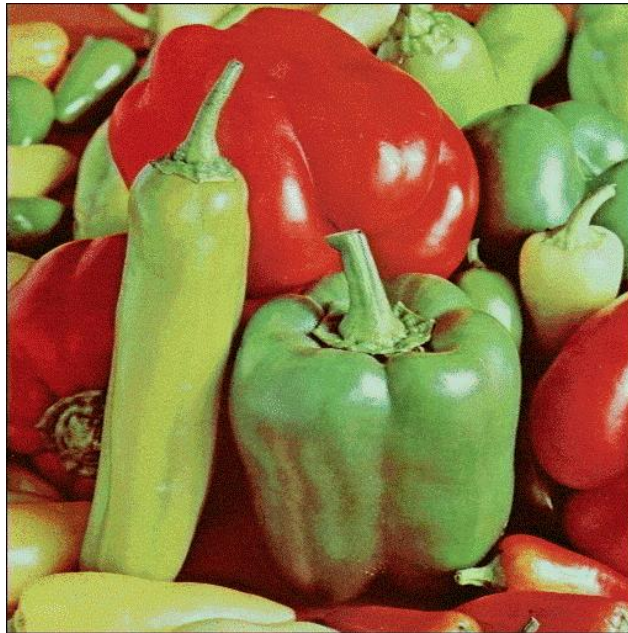
Image with defects

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

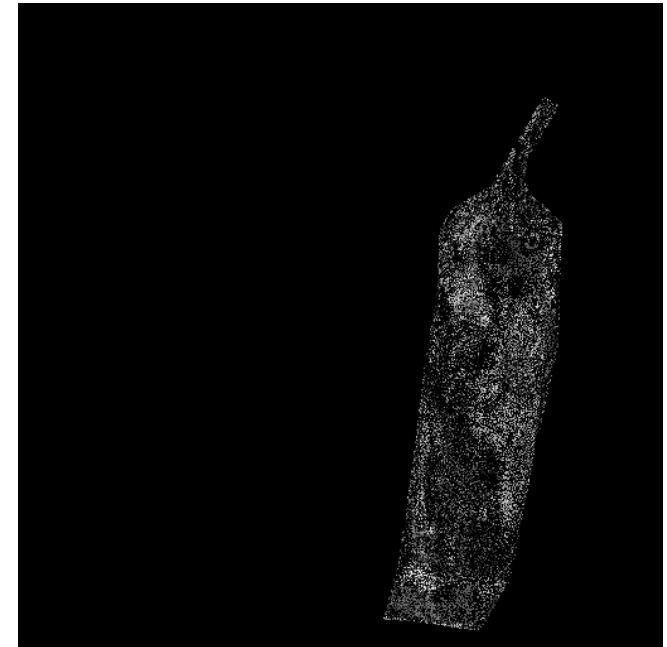
Pengurangan citra untuk mendeteksi objek tambahan di dalam gambar:



-



=



Perkalian citra

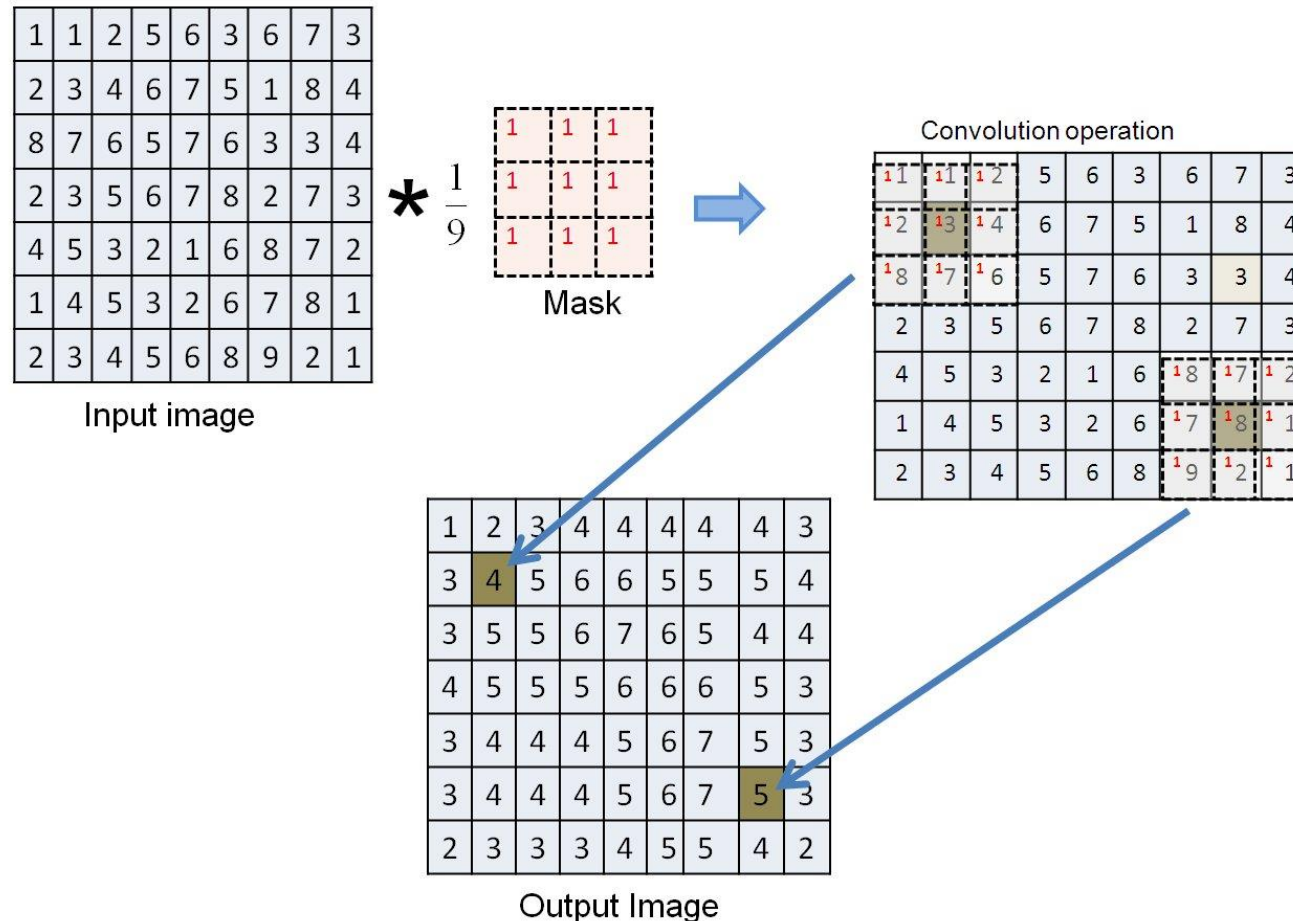
$C(x, y) = A(x, y) \cdot B(x, y) \rightarrow$ mengalikan *pixel-pixel* pada posisi yang bersesuaian

```
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



Perkalian/pembagian Citra dengan Skalar

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

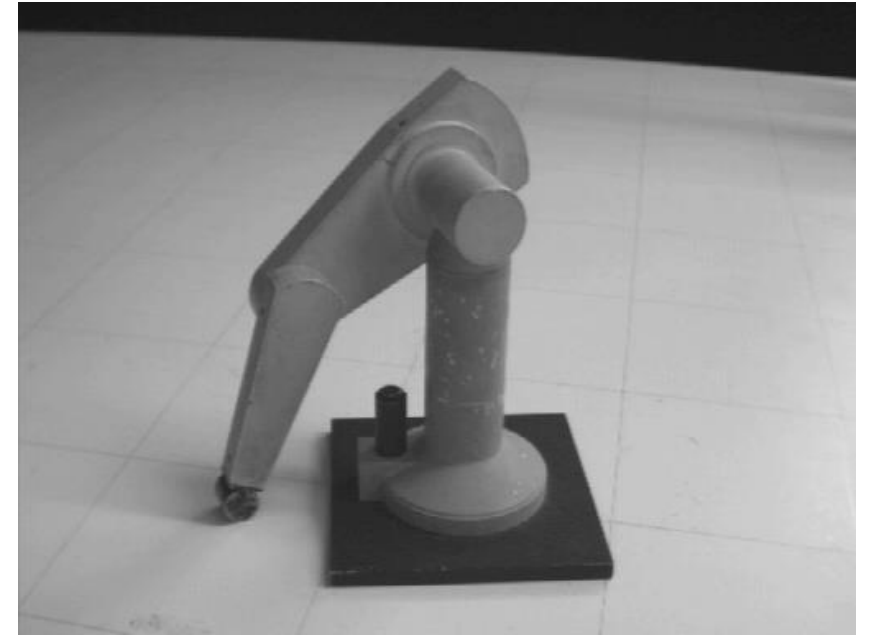
- Operasi perkalian citra dengan skalar dipakai untuk kalibrasi kecerahan (*calibration of brightness*).
- Operasi pembagian citra dengan skalar dipakai untuk normalisasi kecerahan (*normalization of brightness*).

Contoh perkalian citra dengan skalar:



Citra semula

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



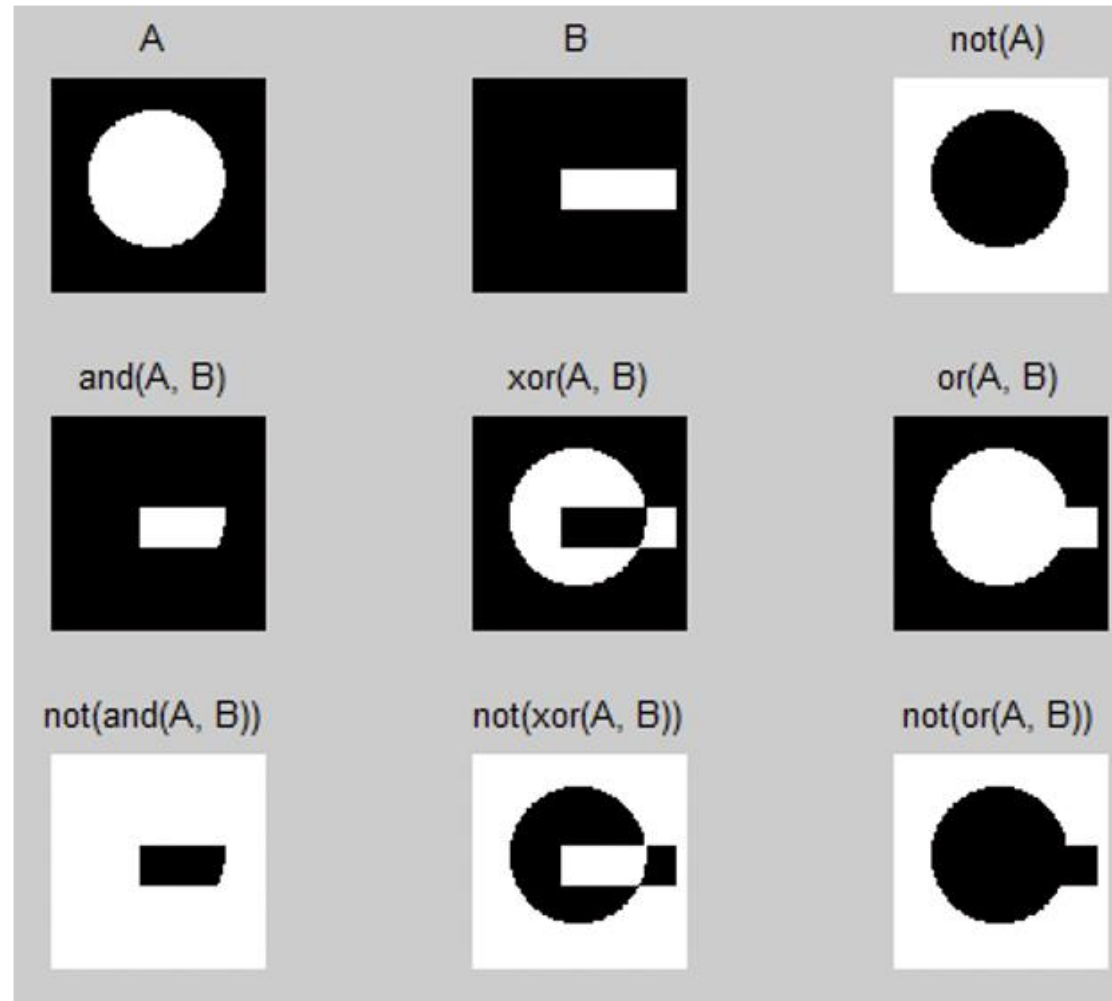
Citra setelah setiap pixel dikali dengan 3

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

Operasi Boolean

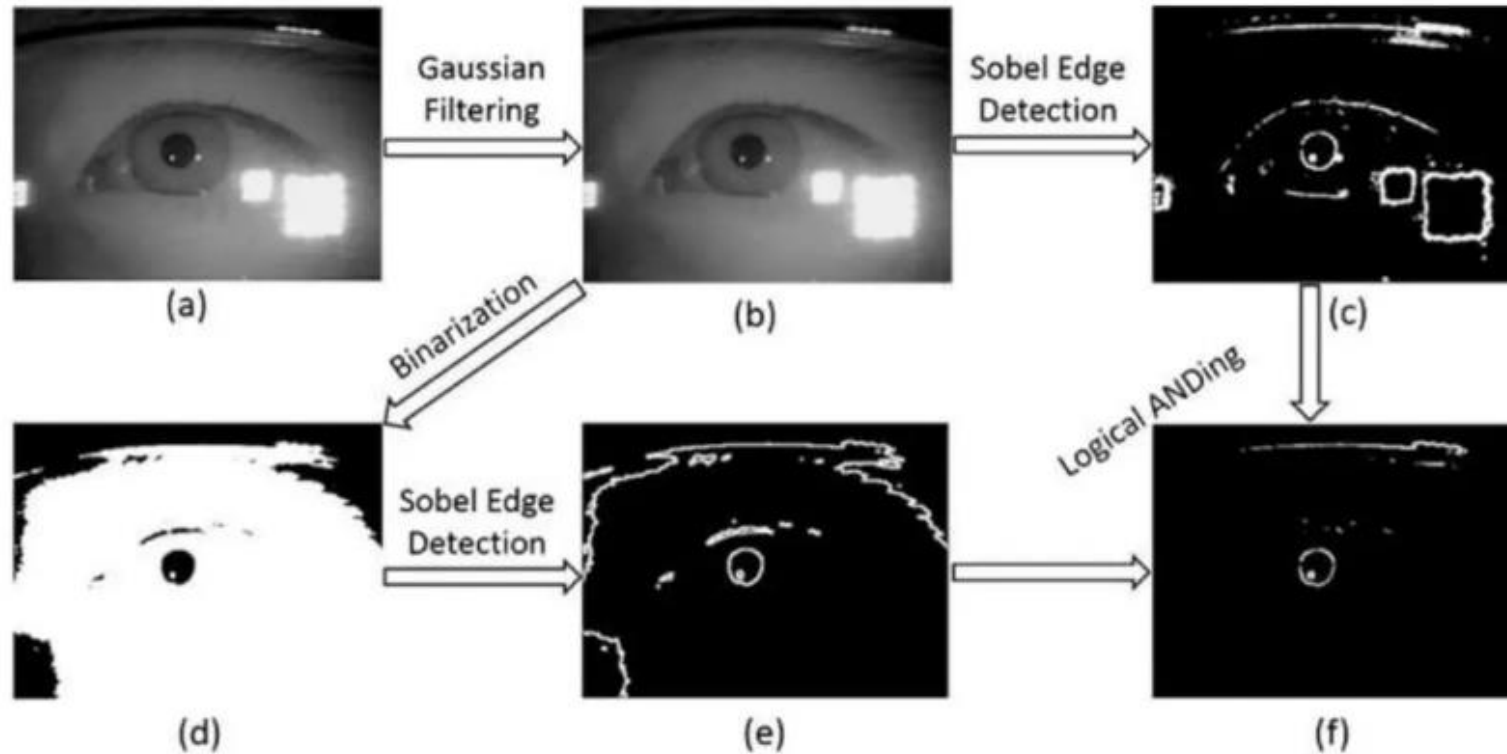
- Operasi AND: $C(x, y) = A(x, y) \text{ and } B(x, y)$
- Operasi OR: $C(x, y) = A(x, y) \text{ or } B(x, y)$
- Operasi NOT: $C(x, y) = \text{not } A(x, y)$
- Operasi XOR: $C(x, y) = A(x, y) \text{ xor } B(x, y)$

Efek operasi AND, OR, NOT, XOR, dan NAND



Hasil-hasil operasi nalar atas dua buah citra A dan B

- 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 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. Pergeseran / Translasi Citra

Penggeseran citra ke arah mendatar atau vertikal dapat dilakukan dengan

$$x_{baru} = x_{lama} + S_x$$

$$y_{baru} = y_{lama} + S_y$$

Pergeseran 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];
    }
}
```

Contoh Hasil Pergeseran / Translasi Citra

1 Rumus Pergeseran / Translasi

Pergeseran citra ke arah mendatar atau vertikal dapat dilakukan dengan:

$$x_{baru} = x_{lama} + S_x$$

$$y_{baru} = y_{lama} + S_y$$

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

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

2 Contoh Kasus

Misalkan citra Lena (512 × 512) akan digeser sejauh $m = 50$ piksel ke kanan (arah x) dan $n = 30$ piksel ke bawah (arah y).

Parameter:

$m = 50$ (arah x, kanan)

$n = 30$ (arah y, bawah)

Rumus yang digunakan:

$$B[x][y] = A[x + 50][y + 30]$$

3 Hasil Pergeseran Citra

Citra Asli (A)



Ukuran: 512 × 512 piksel

$m = 50$
(kanan)



$n = 30$
(bawah)

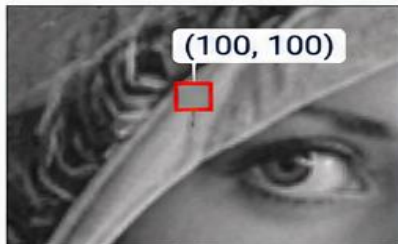
Citra Hasil (B)



Ukuran: 512 × 512 piksel

4 Ilustrasi Perhitungan Nilai Piksel

Ambil salah satu piksel pada citra A



Nilai piksel $A[100][100] = 138$

Setelah translasi ($m = 50, n = 30$)



Nilai piksel $B[150][130] = A[100][100] = 138$

Langkah perhitungan:

1 $x_{baru} = 100 + 50 = 150$

2 $y_{baru} = 100 + 30 = 130$

3 $B[150][130] = A[100][100]$
 $= 138$

5 Ringkasan

- Citra digeser 50 piksel ke kanan dan 30 piksel ke bawah.
- Setiap piksel pada citra hasil B mengambil nilai dari piksel pada citra A sesuai rumus:

$$B[x][y] = A[x + 50][y + 30]$$

- Jika koordinat baru berada di luar batas citra, nilai piksel dapat diisi dengan 0 (hitam) atau metode padding lainnya.



Catatan: Pada implementasi nyata, perlu diperhatikan penanganan nilai piksel yang keluar dari batas citra (misalnya diisi 0/ hitam, diulang, atau menggunakan metode padding lainnya).

2. Rotasi / Pemutaran Citra

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 :

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



Rumus Rotasi Citra

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:

$$B[x'][y'] = A \begin{bmatrix} x \cos(\theta) - y \sin(\theta) \\ x \sin(\theta) + y \cos(\theta) \end{bmatrix}$$

Contoh: Rotasi 90° Berlawanan Arah Jarum Jam

Citra Lena (Asli)



Ukuran: 512 × 512 piksel



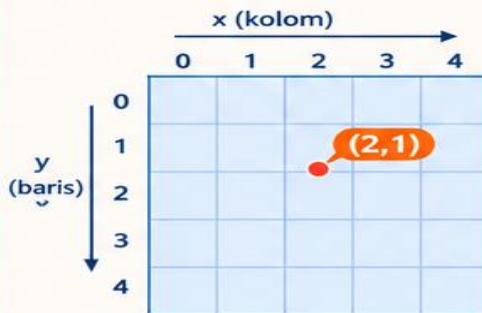
Hasil Rotasi 90° CCW



Ukuran: 512 × 512 piksel

Contoh Perhitungan Titik Piksel

Koordinat piksel pada citra A



Dengan $\theta = 90^\circ$ (CCW)

- 1 Titik $(2,1) \rightarrow (x', y')$
- 2 Hitung x' :

$$x' = 2 \cos 90^\circ - 1 \sin 90^\circ$$

$$= 2(0) - 1(1) = -1$$
- 3 Hitung y' :

$$y' = 2 \sin 90^\circ + 1 \cos 90^\circ$$

$$= 2(1) + 1(0) = 2$$
- 4 Pemetaan nilai piksel :

$$B[-1][2] = A[2][1]$$

Hasil Pemetaan Nilai Piksel

Pada citra A (asli)



Nilai piksel $A[2][1]$
= 165

Pada citra B (hasil rotasi)



Nilai piksel $B[-1][2]$
= 165



Catatan Penting

- Pada implementasi citra digital (array), koordinat (x, y) biasanya perlu disesuaikan ke sistem indeks baris-kolom (row, column).
- Selain itu, perhatikan pusat rotasi (umumnya di tengah citra), terutama jika ukuran citra genap.

```
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 :

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

Ilustrasi Contoh Perhitungan Flipping Citra

1

Flipping Horizontal pada Sumbu-Y (pencerminan terhadap sumbu-Y)

$$B[1][y] = A[4][y]$$

$$B[2][y] = A[3][y]$$

$$B[3][y] = A[2][y]$$

$$B[4][y] = A[1][y]$$

A (4 kolom × 3 baris)

	1	2	3	4
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12

Cermin
pada sumbu-Y
(horizontal)

B (4 kolom × 3 baris)

	1	2	3	4
1	4	3	2	1
2	8	7	6	5
3	12	11	10	9

Langkah hitung (contoh):

- $B[1][1] = A[4][1] = 4$
- $B[2][1] = A[3][1] = 8$
- $B[3][1] = A[2][1] = 12$
- $B[4][1] = A[1][1] = 1$

2

Flipping Vertikal pada Sumbu-X (pencerminan terhadap sumbu-X)

$$B[x][1] = A[x][3]$$

$$B[x][2] = A[x][2]$$

(M = 3)

$$B[x][3] = A[x][1]$$

A (4 kolom × 3 baris)

	1	2	3	4
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12

Cermin
pada sumbu-X
(vertikal)

B (4 kolom × 3 baris)

	1	2	3	4
1	9	10	11	12
2	5	6	7	8
3	1	2	3	4

Langkah hitung (contoh):

- $B[1][1] = A[1][3] = 9$
- $B[2][1] = A[2][3] = 5$
- $B[3][1] = A[3][3] = 1$
- $B[1][2] = A[1][2] = 10$

3

Pencerminan terhadap Titik Asal (pusat koordinat)

$$B[x][y] = A[4 - x][3 - y] \quad (\text{contoh indeks 1-based})$$

A (4 kolom × 3 baris)

	1	2	3	4
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12

Cermin
pada titik asal
(180°)

B (4 kolom × 3 baris)

	1	2	3	4
1	12	11	10	9
2	8	7	6	5
3	4	3	2	1

Pasangan posisi (contoh):

- $(1,1) \leftrightarrow (4,3) : 1 \leftrightarrow 12$
- $(1,2) \leftrightarrow (4,2) : 5 \leftrightarrow 8$
- $(2,1) \leftrightarrow (3,3) : 2 \leftrightarrow 11$
- $(2,2) \leftrightarrow (3,2) : 6 \leftrightarrow 7$
- $(3,1) \leftrightarrow (2,3) : 3 \leftrightarrow 10$
- $(3,2) \leftrightarrow (2,2) : 7 \leftrightarrow 6$
- $(4,1) \leftrightarrow (1,3) : 4 \leftrightarrow 9$
- $(4,2) \leftrightarrow (1,2) : 8 \leftrightarrow 5$

4

Pencerminan terhadap Garis $x = y$ (diagonal utama)

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

A (4 kolom × 3 baris)

	1	2	3	4
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12

Cermin
pada garis $x = y$
(diagonal utama)

B (4 kolom × 3 baris)

	1	2	3	4
1	1	5	9	
2	2	6	10	
3	3	7	11	12

Langkah hitung (contoh):

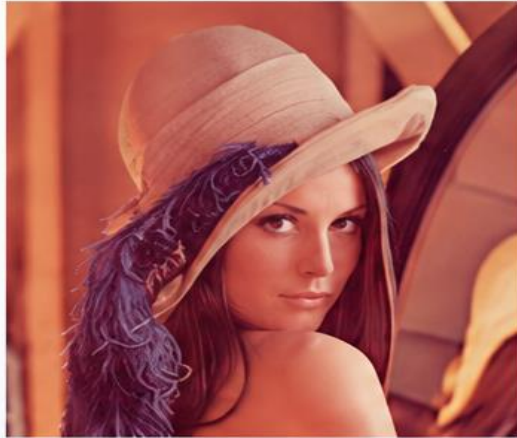
- $B[1][1] = A[1][1] = 1$
- $B[2][1] = A[1][2] = 2$
- $B[3][1] = A[1][3] = 3$
- $B[4][1] = A[1][4] = 4$



Catatan: Contoh pada ilustrasi ini menggunakan indeks **1-based** (mulai dari 1).

Contoh Hasil Transformasi Citra Lena

Citra Lena (Asli)

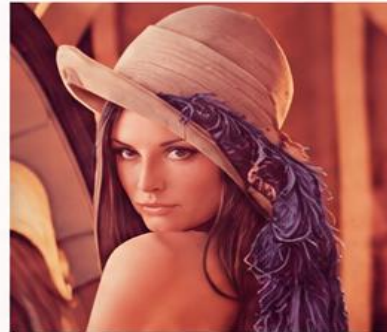


Ukuran: 512 × 512 piksel

1. Flipping Horizontal

Pencerminan pada sumbu-Y (membalik kiri–kanan)

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

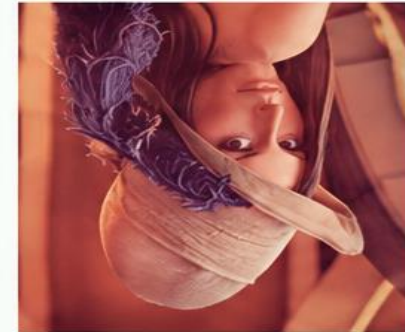


Hasil: citra terbalik kiri–kanan

2. Flipping Vertikal

Pencerminan pada sumbu-X (membalik atas–bawah)

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



Hasil: citra terbalik atas–bawah

3. Pencerminan pada garis $x = y$ (transpose)

Menukar koordinat baris dan kolom

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



Citra asli



Hasil transpose

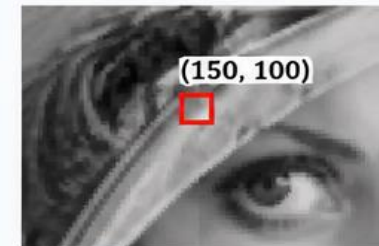
4. Contoh perhitungan nilai piksel

Misal, piksel pada koordinat $(x, y) = (100, 150)$

$$B[x, y] = A[y][x] = A[150][100]$$



Nilai piksel $A[100][150] = 138$
(citra asli)



Nilai piksel $B[100][150] = A[150][100] = 138$
(hasil transformasi)

```
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:

$$x' = s_x \cdot x$$

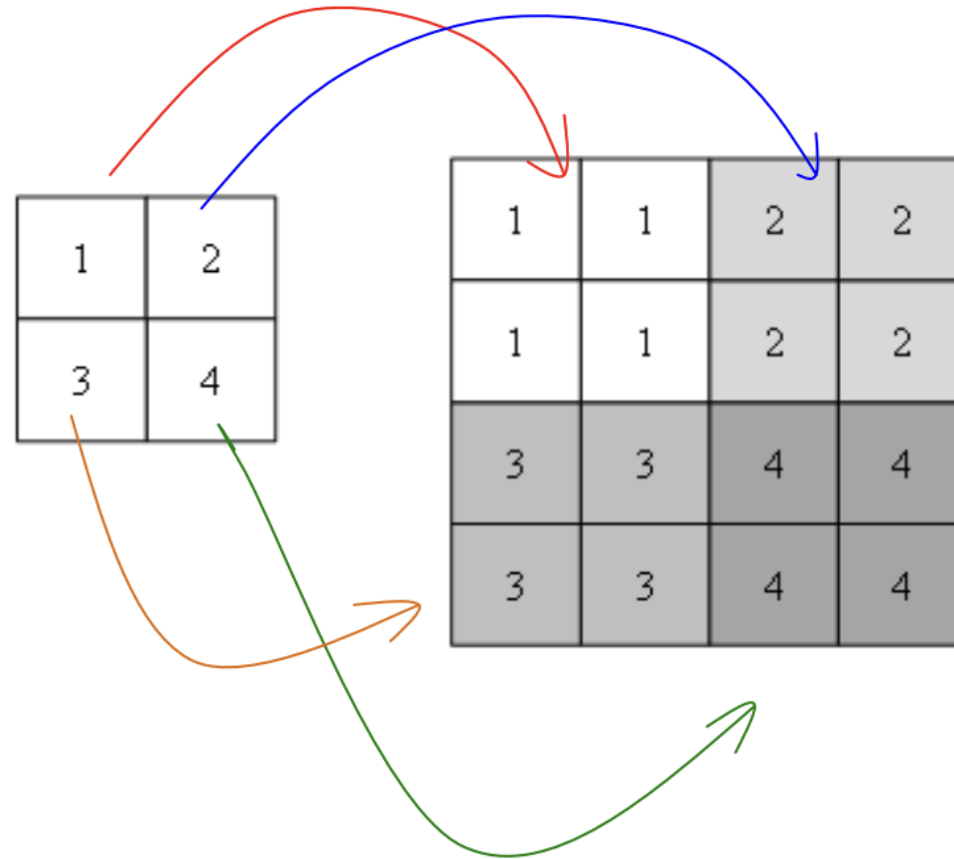
$$y' = s_y \cdot y$$

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

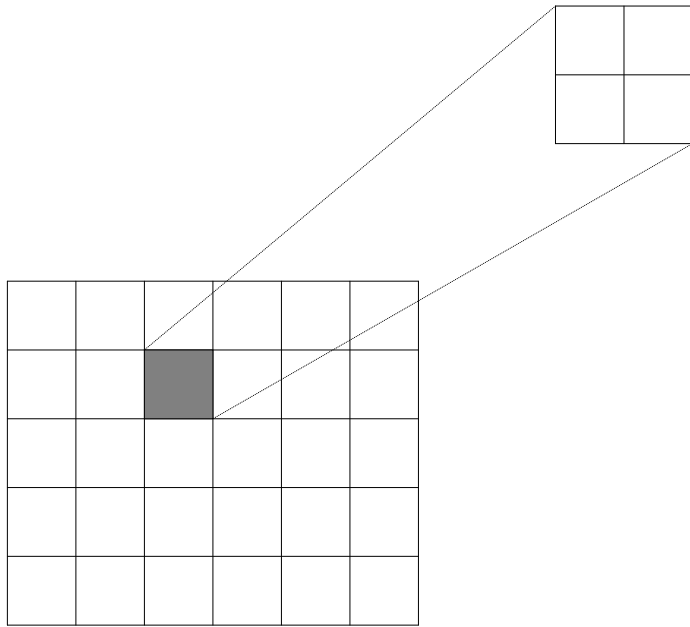
- Penskalaan citra A menjadi citra B :

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

pembesaran pada arah vertikal dan horizontal sebesar 2 kali
(Nearest Neighbor Interpolation)

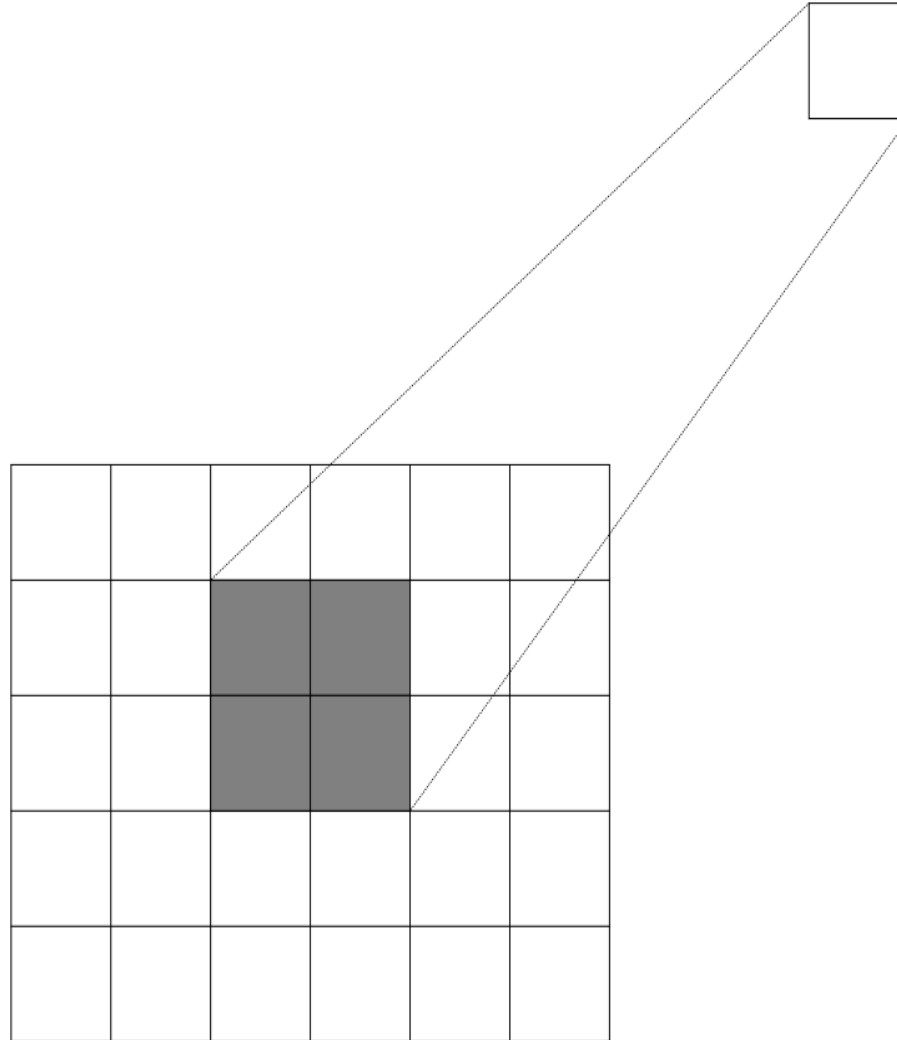


- Operasi *zoom out* (memperbesar) dengan faktor 2 (yaitu, $s_x = s_y = 2$) diimplementasikan dengan menyalin setiap *pixel* sebanyak 4 kali.



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

Operasi *zoom in* (pengecilan) dengan faktor skala $1/2$ dilakukan dengan mengambil rata-rata 4 pixel yang bertetangga menjadi 1 pixel.



Contoh Metode Area-Based / Averaging

Contoh Pengecilan Citra 4×4 → 2×2

Misalkan citra grayscale berukuran **4 × 4 piksel** berikut:

$$I = \begin{bmatrix} 10 & 20 & 30 & 40 \\ 50 & 60 & 70 & 80 \\ 90 & 100 & 110 & 120 \\ 130 & 140 & 150 & 160 \end{bmatrix}$$

Citra akan diperkecil menjadi **2 × 2 piksel**.

Misalnya piksel pertama dihitung dari:

$$\begin{bmatrix} 10 & 20 \\ 50 & 60 \end{bmatrix}$$

Nilai rata-ratanya:

$$410+20+50+60=4140=35$$

Piksel kedua:

$$\begin{bmatrix} 30 & 40 \\ 70 & 80 \end{bmatrix}$$

$$430+40+70+80=4220=55$$

Piksel ketiga:

$$\begin{bmatrix} 90 & 100 \\ 130 & 140 \end{bmatrix}$$

$$490+100+130+140=4460=115$$

Piksel keempat:

$$\frac{\begin{bmatrix} 110 & 120 \\ 150 & 160 \end{bmatrix} \quad 110 + 120 + 150 + 160}{4} = \frac{540}{4} = 135$$

Maka hasil sederhananya:

$$I' = \begin{bmatrix} 35 & 55 \\ 115 & 135 \end{bmatrix}$$

Downsampling vs Decimation

- Kedua istilah ini menjelaskan **dua cara berbeda** yang menyebabkan resolusi citra hasil penerokan (sampling) menjadi rendah — meskipun hasil akhirnya sama-sama "piksel berkurang",

1. DOWNSAMPLING (pengambilan terokan)

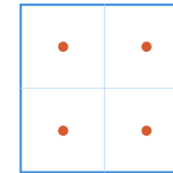


diterok langsung dgn interval jarang

terok jarang

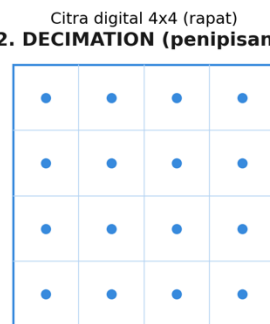


Citra digital 2x2



hasil langsung penerokan jarang

2. DECIMATION (penipisan)

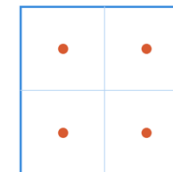


sudah diterok rapat sepenuhnya

buang sampel



Citra digital 2x2



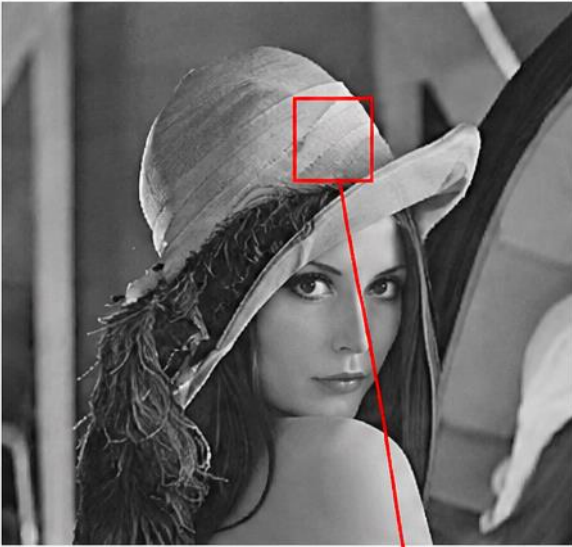
hasil dari citra rapat, ditipiskan

Hasil akhir bisa sama-sama 2x2, tapi prosesnya berbeda:
Downsampling = terok jarang sejak awal | Decimation = terok rapat dulu, baru dikurangi

Contoh Downsampling Menggunakan Area Based

Pada metode area based, setiap piksel pada citra hasil downsampling diperoleh dari rata-rata nilai piksel pada area (blok) tertentu di citra asli.

Citra Asli (512 × 512)



Zoom area (citra asli)

Hasil Downsampling Area Based (256 × 256)
(skala 2×)



Zoom area (hasil downsampling)

Ilustrasi Proses Area Based

120	130
110	140

Area 2 × 2 piksel
pada citra asli

Rata-rata
→

125

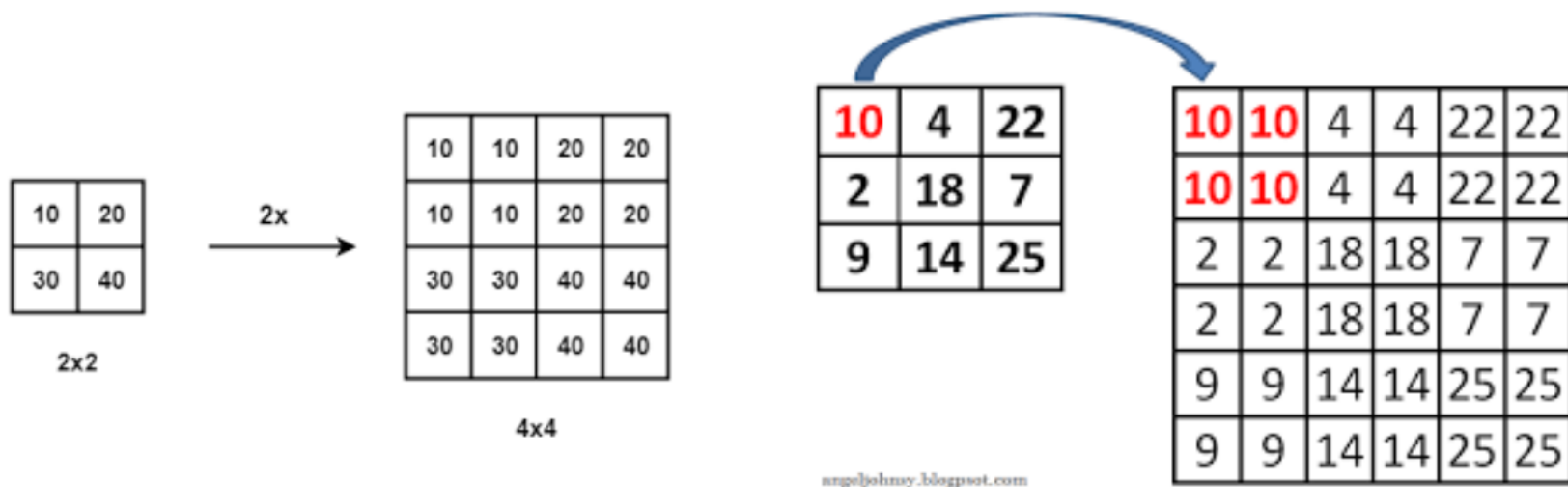
Nilai piksel pada
citra hasil downsampling

Keterangan:

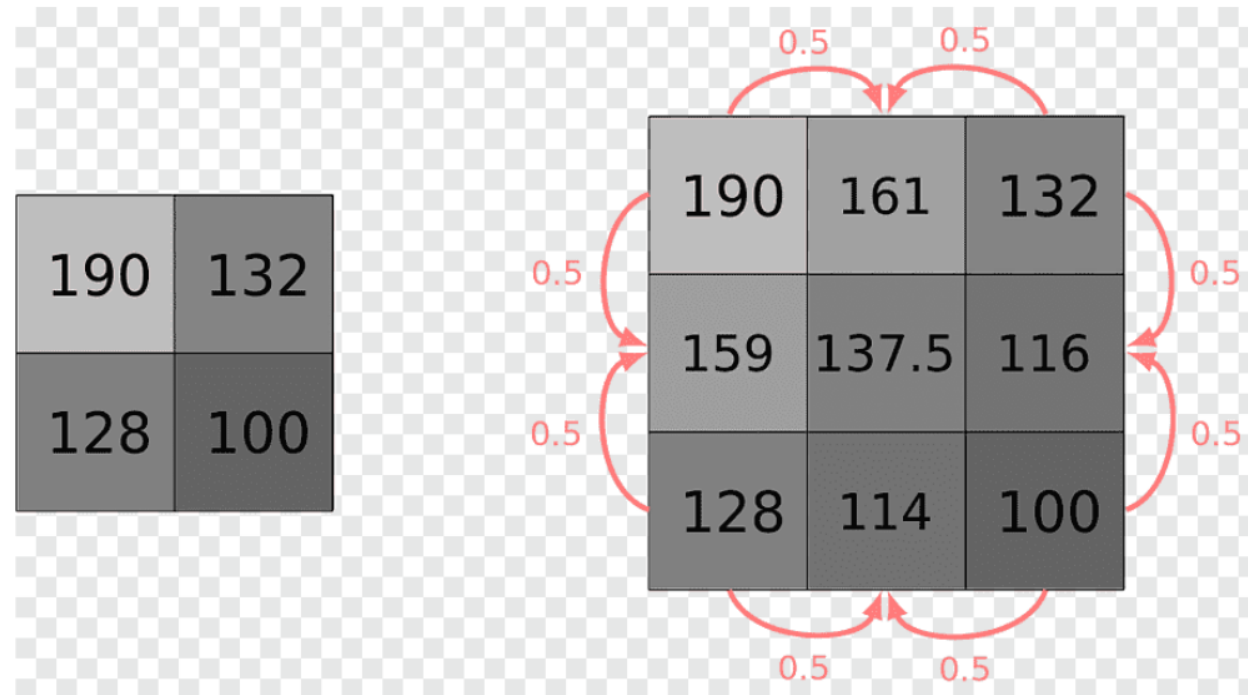
Pada *area based downsampling*, setiap piksel baru merupakan hasil rata-rata dari nilai piksel pada area (blok) tertentu di citra asli. Metode ini menghasilkan citra yang lebih halus (*smooth*) dibandingkan metode lain seperti *decimation* atau *nearest neighbor*, karena informasi dari seluruh area dipertimbangkan.

- **Nearest neighbor:** hanya pakai 1 piksel terdekat
- **Bilinear:** pakai 4 piksel tetangga (matriks 2×2)
- **Bicubic:** pakai 16 piksel tetangga (matriks 4×4) → hasil lebih halus, detail lebih terjaga, tapi komputasi lebih berat

Nearest Neighbor Interpolation



Bilinear Interpolation



Menggunakan empat pixel bertetangga

Misal mencari nilai di titik (0.5, 0.5)

1. Interpolasi di arah x (horizontal):

Untuk baris **y = 0**:

$$R1 = (1 - 0.5) \cdot 190 + 0.5 \cdot 132 = 0.5 \cdot 190 + 0.5 \cdot 132 = 161.0$$

Untuk baris **y = 1**:

$$R2 = (1 - 0.5) \cdot 128 + 0.5 \cdot 100 = 0.5 \cdot 128 + 0.5 \cdot 100 = 114.0$$

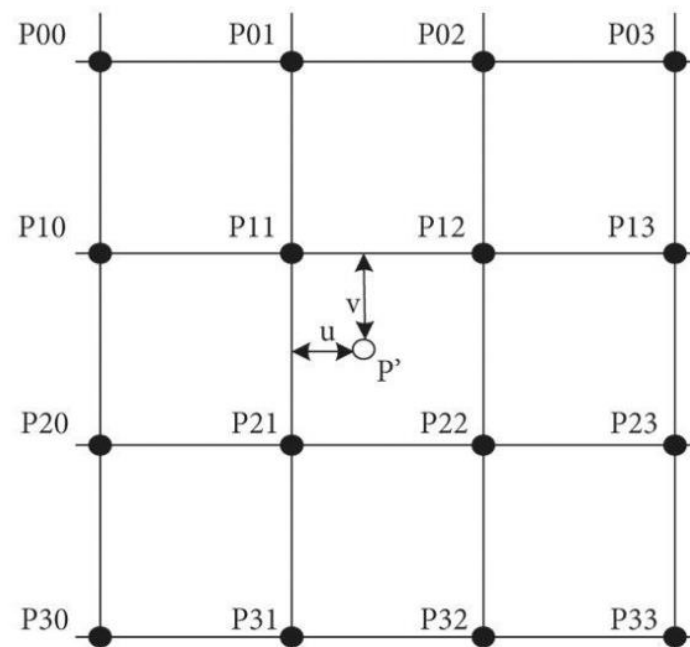
2. Interpolasi di arah y (vertikal):

interpolasi antara **R1 dan R2** untuk **y = 0.5**:

$$P = (1 - 0.5) \cdot 161.0 + 0.5 \cdot 114.0 = 0.5 \cdot 161 + 0.5 \cdot 114 = 137.5$$

- **bilinear interpolation** (pakai **4 piksel tetangga**)
- **Bicubic interpolation** adalah metode interpolasi dua dimensi yang menggunakan **nilai 16 piksel tetangga** (dari matriks 4x4) untuk menghitung nilai intensitas di suatu titik **(x, y)** di dalam citra

Bicubic Interpolation



Menggunakan 16 pixel bertetangga

Y\X	0	1	2	3
0	52	55	61	59
1	62	59	55	104
2	63	65	66	113
3	64	70	70	126

Rumus interpolasi kubik (Catmull-Rom Spline):

Untuk 4 titik: P_0, P_1, P_2, P_3 dan parameter $t \in [0, 1]$:

$f(t)$

Rumus ini ekuivalen dengan bentuk **bobot (weight)** berikut, yang lebih praktis untuk

hitungannya (Catmull-Rom Spline):

$$w_0 = -0.5t^3 + t^2 - 0.5t \quad w_1 = 1.5t^3 - 2.5t^2 + 1$$

$$w_2 = -1.5t^3 + 2t^2 + 0.5t \quad w_3 = 0.5t^3 - 0.5t^2$$

$$f(t) = w_0P_0 + w_1P_1 + w_2P_2 + w_3P_3$$

Khusus untuk $t = 0.5$ (titik tengah), bobotnya selalu tetap:

$$w_0 = -0.0625, w_1 = 0.5625, w_2 = 0.5625, w_3 = -0.0625$$

Titik yang akan dihitung (nilai interpolasi bicubic di posisi setengah langkah):

Titik (x,y)

Penjelasan

(0.5, 0.5)

Tengah antara (0,0),(1,1)

(1.5, 0.5)

Tengah antara (1,0),(2,1)

(2.5, 0.5)

Tengah antara (2,0),(3,1)

(0.5, 1.5)

Tengah antara (0,1),(1,2)

(1.5, 1.5)

Tengah antara (1,1),(2,2)

(2.5, 1.5)

Tengah antara (2,1),(3,2)

(0.5, 2.5)

Tengah antara (0,2),(1,3)

(1.5, 2.5)

Tengah antara (1,2),(2,3)

(2.5, 2.5)

Tengah antara (2,2),(3,3)

Y\X	0	1	2	3
0	52	55	61	59
1	62	59	55	104
2	63	65	66	113
3	64	70	70	126

Y\x	0.5	1.5	2.5
0.5	53	58	61
1.5	62	63	73
2.5	63	68	78

- Titik (1,5, 1,5) berada tepat di tengah blok $P_{11}, P_{12}, P_{21}, P_{22}$ sehingga $t = 0,5$ dipakai pada kedua arah (x dan y)
- Tahap 1 — Interpolasi Horizontal (arah-x) pada tiap baris, di $x = 1,5$
- Gunakan bobot $w_0 = -0,0625$, $w_1 = 0,5625$, $w_2 = 0,5625$, $w_3 = -0,0625$
- **Baris y = 0.** $\rightarrow P_0, P_1, P_2, P_3 = 52, 55, 61, 59$
- $f_0 = (-0,0625)(52) + (0,5625)(55) + (0,5625)(61) + (-0,0625)(59)$
- $f_0 = -3,25 + 30,9375 + 34,3125 - 3,6875 = 58,3125$
- **Baris y = 1:** 62, 59, 55, 104
- $f_1 = (-0,0625)(62) + (0,5625)(59) + (0,5625)(55) + (-0,0625)(104)$
- $f_1 = -3,875 + 33,1875 + 30,9375 - 6,5 = 53,75$
- **Baris y = 2:** 63, 65, 66, 113
- $f_2 = (-0,0625)(63) + (0,5625)(65) + (0,5625)(66) + (-0,0625)(113)$
- $f_2 = -3,9375 + 36,5625 + 37,125 - 7,0625 = 62,6875$
- **Baris y = 3:** 64, 70, 70, 126
- $f_3 = (-0,0625)(64) + (0,5625)(70) + (0,5625)(70) + (-0,0625)(126)$
- $f_3 = -4 + 39,375 + 39,375 - 7,875 = \mathbf{66,875}$

- **Tahap 2 — Interpolasi Vertikal (arah-y) dari hasil Tahap 1**
- Sekarang keempat hasil di atas ($f_0 = 58,3125$, $f_1 = 53,75$, $f_2 = 62,6875$, $f_3 = 66,875$) diperlakukan sebagai 4 titik baru, diinterpolasi lagi dengan $t = 0,5$ memakai bobot yang sama:
 - $f(1,5, 1,5) = (-0,0625)(58,3125) + (0,5625)(53,75) + (0,5625)(62,6875) + (-0,0625)(66,875)$
 - $= -3,6445 + 30,2344 + 35,2617 - 4,1797$
 - $= f(1,5, 1,5) \approx 57,67$
- **Jadi, nilai intensitas piksel hasil interpolasi bicubic di titik (1,5, 1,5) adalah $\approx 57,67$,**
yang merupakan gabungan halus (weighted blend) dari 16 piksel tetangga di sekitarnya — bukan sekadar rata-rata sederhana seperti bilinear, sehingga transisi warna/intensitas menjadi lebih halus dan detail tepi (edge) lebih terjaga.

- Citra asli (4x4 piksel)

52	55	61	59
62	59	55	104
63	65	66	113
64	70	70	126



Citra hasil akhir

52	?	55	?	61	?	59
?	53	?	58	?	61	?
62	?	59	?	55	?	104
?	58	?	58	?	73	?
63	?	65	?	66	?	113
?	63	?	68	?	78	?
64	?	70	?	70	?	126

titik merah = 9 titik target (0.5, 1.5, 2.5)

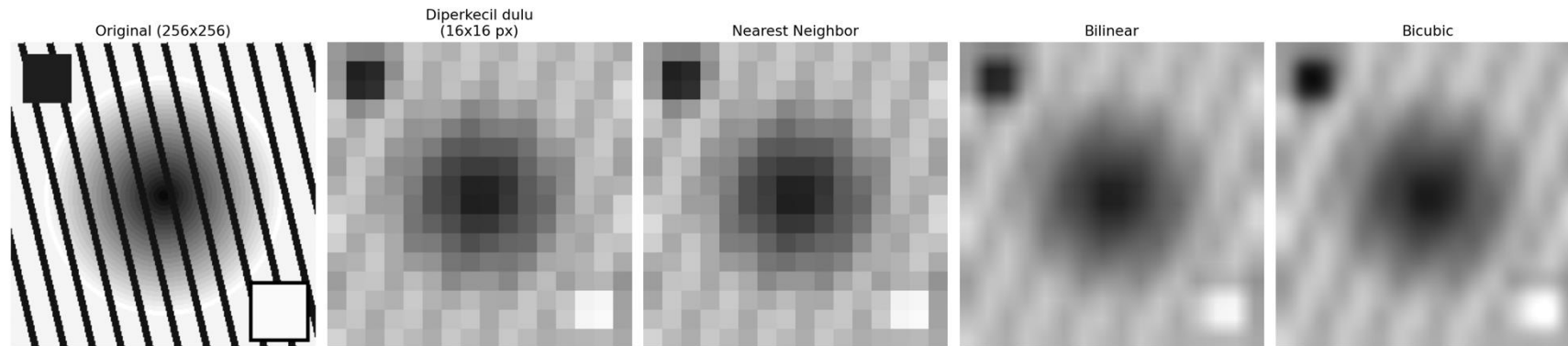
Titik biru = nilai piksel asli, tetap sama seperti data awal.

Titik oranye = piksel baru hasil perhitungan bicubic (termasuk nilai 58 di titik (1.5,1.5) yang barusan kamu hitung manual).

Tanda "?" = posisi-posisi lain yang **juga perlu dihitung** kalau ingin resize citra secara penuh (bukan cuma 9 titik contoh) — caranya persis sama, cuma beda 4x4 jendela piksel tetangga yang dipakai untuk tiap titik target.

Kalau proses ini diterapkan ke **semua** posisi (bukan cuma yang dicontohkan), hasil akhirnya adalah **citra baru berukuran lebih besar** dengan transisi warna/intensitas antar piksel yang mulus — inilah kenapa bicubic dipakai saat kamu **memperbesar (upscale/zoom) foto** agar tidak pecah kotak-kotak seperti nearest neighbor, tapi tetap tajam di bagian tepi dibanding bilinear.

Perbandingan hasil Interpolasi



Tugas

- Implementasikan menjadi program meliputi:
 1. Membuat citra negatif
 2. Mengubah citra berwarna menjadi citra *grayscale*
 3. *Image brightening*
 4. Operasi aritmetika dua buah citra
 5. Operasi geometri (translasi, rotasi, flipping, zooming)