

# Image Enhancement - Filtering / Masking

# Tujuan Perbaikan Citra

- Tujuan dari teknik peningkatan mutu citra adalah untuk melakukan pemrosesan terhadap citra agar hasilnya mempunyai kualitas relatif lebih baik dari citra awal untuk aplikasi tertentu.
- Kata baik disini tergantung pada jenis aplikasi dan problem yang dihadapi

# Teknik peningkatan mutu citra

- Teknik peningkatan mutu citra salah satunya dengan meningkatkan mutu citra domain spasial
  - Point Processing
  - Mask Processing
- Teknik peningkatan mutu citra salah satunya dengan meningkatkan mutu citra domain Frekuensi

# I. Point Processing

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

- Prosedur yang secara langsung memanipulasi pixel.

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

Dimana

$f(x,y)$  adalah image input

$g(x,y)$  adalah image yang diproses

$T$  adalah sebuah operator pada  $f$  yang didefinisikan berdasar nilai neighborhood dari  $(x,y)$



- Operator  $T$  dapat berupa :
  - Kumpulan pixels  $(x,y)$  dari image
  - Kumpulan dari 'neighborhoods'  $N(x,y)$  dari setiap pixel
  - Kumpulan dari images  $f_1, f_2, f_3, \dots$

## Contoh Operator : div by 2

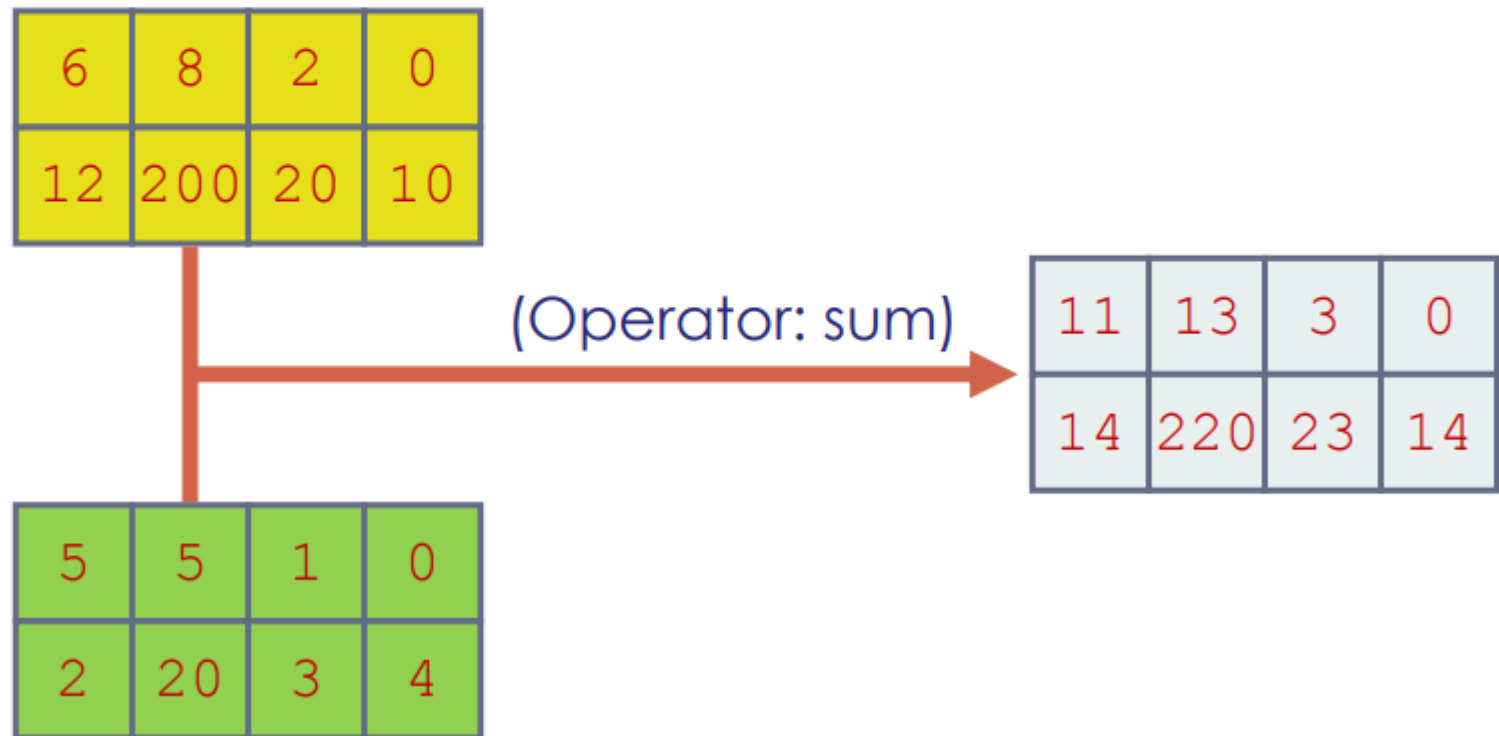
- Operasi terhadap himpunan pixel dari image



(Operator: Div. by 2)

## Contoh Operator : sum

- Operasi terhadap kumpulan image  $f_1, f_2, \dots$





# Point Processing

- Point processing, tetangga 1x1 piksel
  - Output pixel pada titik tertentu hanya bergantung pada input pixel pada titik tersebut dan tidak bergantung pada nilai pixel tetangganya
- $g$  hanya bergantung pada nilai  $f$  pada posisi  $(x,y)$
- $T$  = fungsi transformasi gray level (atau intensitas mapping)

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

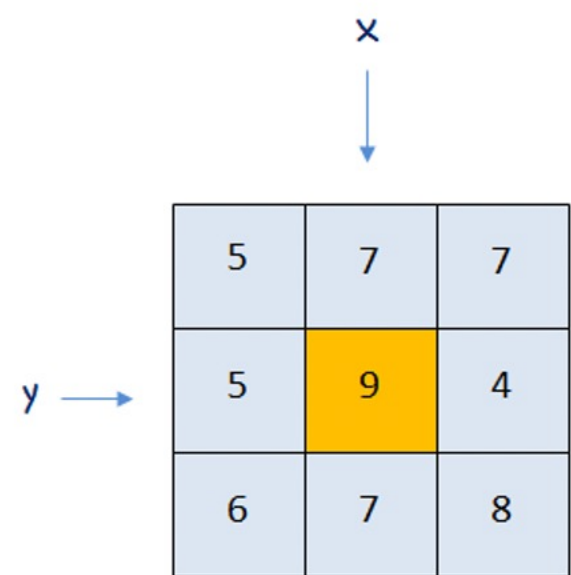
## II. Mask Processing

- Jika pada point processing kita hanya melakukan operasi terhadap masing-masing piksel, maka pada mask Processing kita melakukan operasi terhadap suatu jendela ketetanggaan pada citra.
- Kemudian kita menerapkan suatu *mask* terhadap jendela tersebut. *Mask* sering juga disebut *filter*.

## Aplikasi Ketetanggaan Piksel pada Filter

- Ada tiga jenis filter yang menggunakan operasi ketetanggaan piksel : filter batas, filter pererataan, dan filter median.
- Sebagai filter, operasi ketetanggaan piksel berfungsi untuk menyaring atau paling tidak mengurangi gangguan atau penyimpangan pada citra.
- Idenya adalah mencegah piksel yang intensitasnya di luar intensitas piksel-piksel tetangga.

- $\min = \text{minimum}(5, 7, 7, 5, 4, 6, 7, 8) = 4$ ;
- $\text{maks} = \text{maksimum}(5, 7, 7, 5, 4, 6, 7, 8) = 8$ ;
- mengingat  $f(y, x)$  bernilai 9 dan lebih besar daripada 8 ( $\text{maksInt}$ ) maka  $g(y, x)$  bernilai 8;
- seandainya  $f(y, x)$  pada keadaan di atas bernilai 2 (bukan 9),  $g(y, x)$  akan bernilai 4.



	5	7	7
y →	5	9	4
	6	7	8

Perlu diketahui, pemrosesan hanya dilakukan selain baris pertama, baris terakhir, kolom pertama, dan kolom terakhir. Keempat area tersebut tidak diproses karena tidak mempunyai tetangga yang lengkap (sebanyak 8).

# Efek hasil filter batas



(a) Citra mobil yang telah diberi bintik-bintik putih



(b) Hasil pemfilteran gambar (a)



(c) Citra bird dengan derau



(d) Hasil pemfilteran gambar (c)

# Filter Pererataan

- *Filter pererataan menggunakan Rumus*

$$g(y, x) = \frac{1}{9} \sum_{p=-1}^1 \sum_{q=-1}^1 f(y + p, x + q)$$

Sebagai contoh, piksel pada  $f(y, x)$  dan kedelapan tetangganya memiliki nilai-nilai kecerahan seperti berikut.

65	50	55
78	68	60
60	60	62

# Contoh

65	50	55
78	68	60
60	60	62

Pada contoh di atas, yang diarsir (yaitu yang bernilai 68) merupakan nilai pada  $f(y, x)$ .

Nilai rerata pengganti untuk  $g(y, x)$  dihitung dengan cara seperti berikut:

$$g(y, x) = 1/9 \times (65+50+55+78+68+60+60+60+62) = 61,7778$$

$$\cong 62$$

Jadi, nilai 68 pada  $f(y, x)$  diubah menjadi 62 pada  $g(y, x)$ .

# Hasil dari filter pererataan



(a) Citra mobil dengan bintik-bintik putih



(b) Hasil pemrosesan mobil



(c) Citra boneka berbintik dengan derau



(d) Hasil pemrosesan boneka

efek pemrosesan dengan filter pererataan. Dibandingkan dengan filter batas, hasil pemrosesan filter pererataan tidak menghilangkan bintik-bintik putih pada citra mobil, tetapi hanya agak menyamarkan. Pada citra boneka derau lebih dihaluskan



# Filter Median

- Filter median sangat populer dalam pengolahan citra.
- Filter ini dapat dipakai untuk menghilangkan derau bintik-bintik.
- Nilai yang lebih baik digunakan untuk suatu piksel ditentukan oleh nilai median dari setiap piksel dan kedelapan piksel tetangga pada 8-ketetanggaan.

$$g(y, x) = \text{median}(\begin{aligned} &f(y - 1, x - 1), f(y - 1, x), f(y - 1, x + 1), \\ &f(y, x - 1), f(y, x), f(y, x + 1), \\ &f(y + 1, x - 1), f(y + 1, x), f(y + 1, x + 1)) \end{aligned})$$

10	13	10
10	10	12
12	12	12



10 10 10 10 12 12 12 12 13 ← Diurutkan

Nilai di tengah  
(median)

untuk mendapatkan median,  
diperlukan pengurutan (*sorting*)  
terlebih dulu.

# Hasil dari Filter Median



(a) Citra mobil dengan bintik-bintik putih



(b) Hasil pemrosesan terhadap gambar (a)



(c) Citra boneka dengan derau



(d) Hasil pemrosesan terhadap gambar (c)

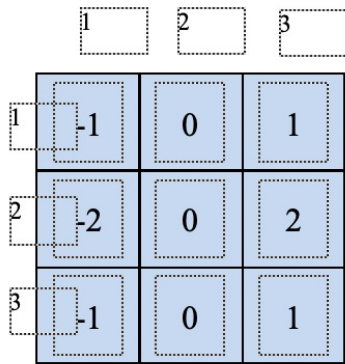
Hasilnya terlihat bahwa derau dapat dihilangkan, detail pada citra tetap dipertahankan.

Namun, hal ini tentu saja didapat dengan tambahan beban komputasi karena adanya sorting

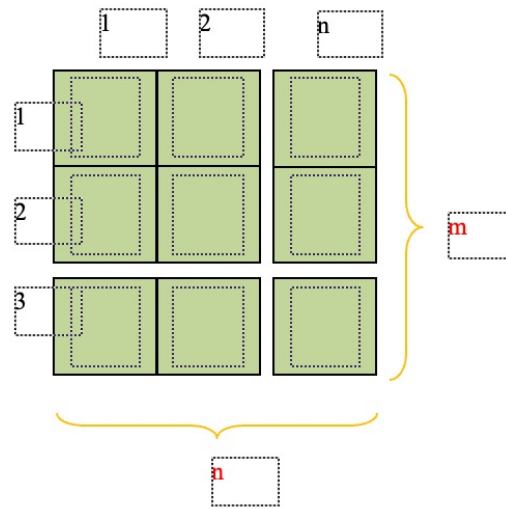
# Teori Konvolusi

- Konvolusi terdapat pada operasi pengolahan citra yang mengalikan sebuah citra dengan sebuah mask atau kernel
- Operasi yang mendasar dalam pengolahan citra adalah operasi konvolusi

- Konvolusi seringkali dilibatkan dalam operasi ketetanggaan piksel.
- Konvolusi pada citra sering disebut sebagai konvolusi dua-dimensi (konvolusi 2D).
- Konvolusi 2D didefinisikan sebagai proses untuk memperoleh suatu piksel didasarkan pada nilai piksel itu sendiri dan tetangganya, dengan melibatkan suatu matriks yang disebut kernel yang merepresentasikan pembobotan.
- Wujud kernel umumnya bujur sangkar, tetapi dapat pula berbentuk persegi panjang

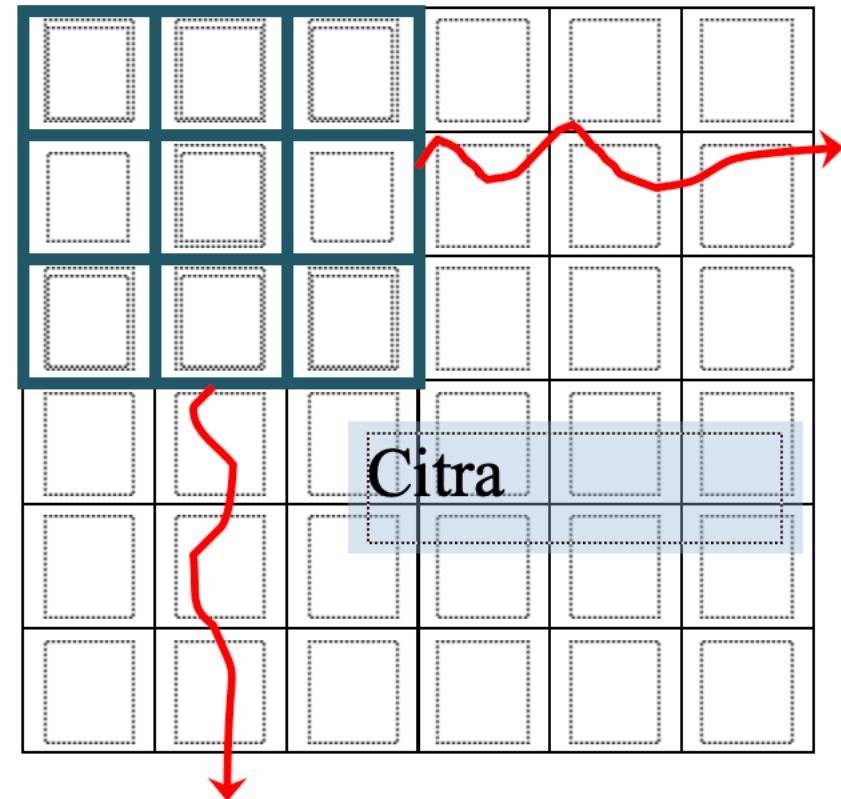


(b) Kernel 3x3

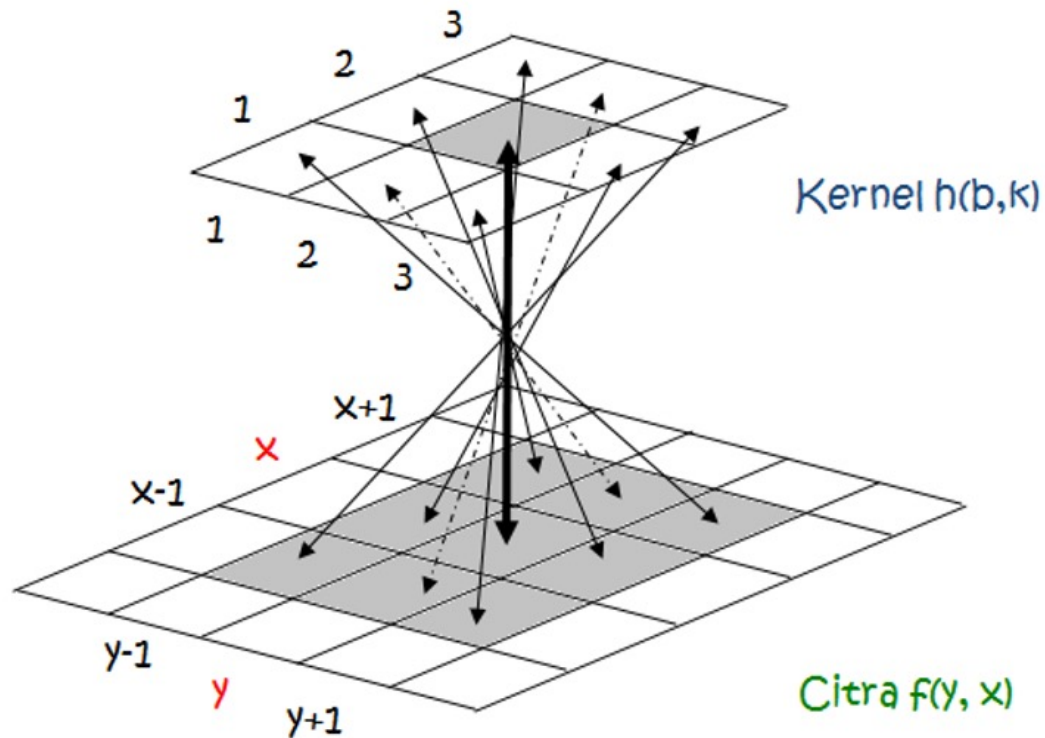


(a) Kernel  $m \times n$

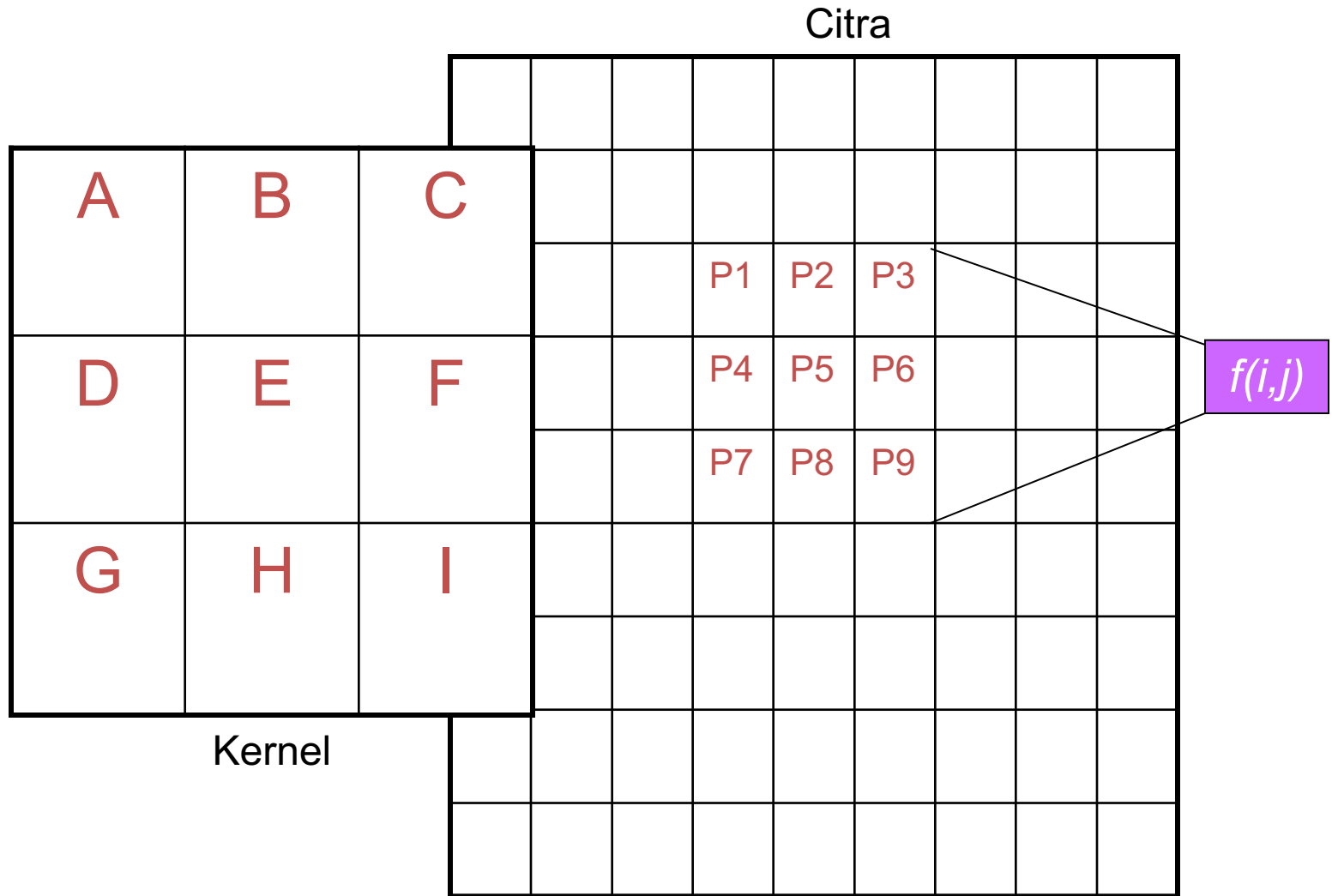
Kernel digerakkan di sepanjang baris dan kolom



# Ilustrasi konvolusi



# Ilustrasi Konvolusi



$$f(i,j) = AP1 + BP2 + CP3 + DP4 + EP5 + FP6 + GP7 + HP8 + IP9$$



# Contoh Operasi Konvolusi

- Operasi konvolusi dilakukan dengan mengeser kernel konvolusi pixel per pixel
- Hasil konvolusi disimpan dalam matrik yang baru

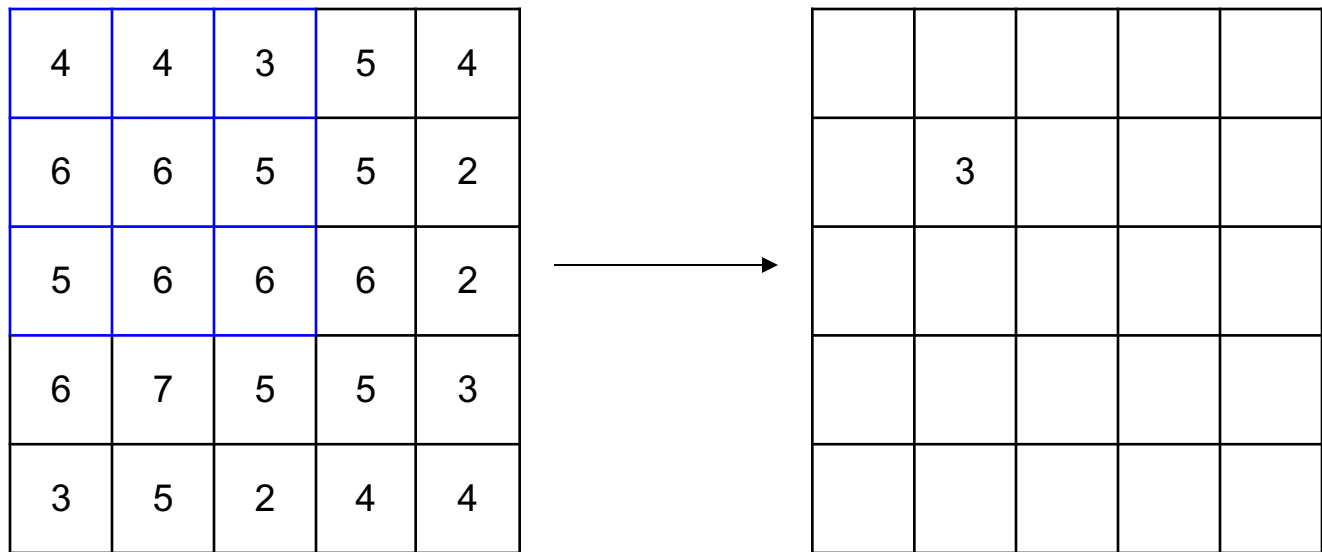
$$f(x,y) = \begin{pmatrix} 4 & 4 & 3 & 5 & 4 \\ 6 & 6 & 5 & 5 & 2 \\ 5 & 6 & 6 & 6 & 2 \\ 6 & 7 & 5 & 5 & 3 \\ 3 & 5 & 2 & 4 & 4 \end{pmatrix} \quad g(x,y) = \begin{pmatrix} 0 & -1 & 0 \\ -1 & \bullet & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

**Citra 5 x 5** **Kernel 3 x 3**

Tanda  menyatakan posisi (0,0) dari kernel

# Contoh Operasi Konvolusi [1]

- Tempatkan kernel pada sudut kiri atas, kemudian hitung nilai pixel pada posisi (0,0) dari kernel

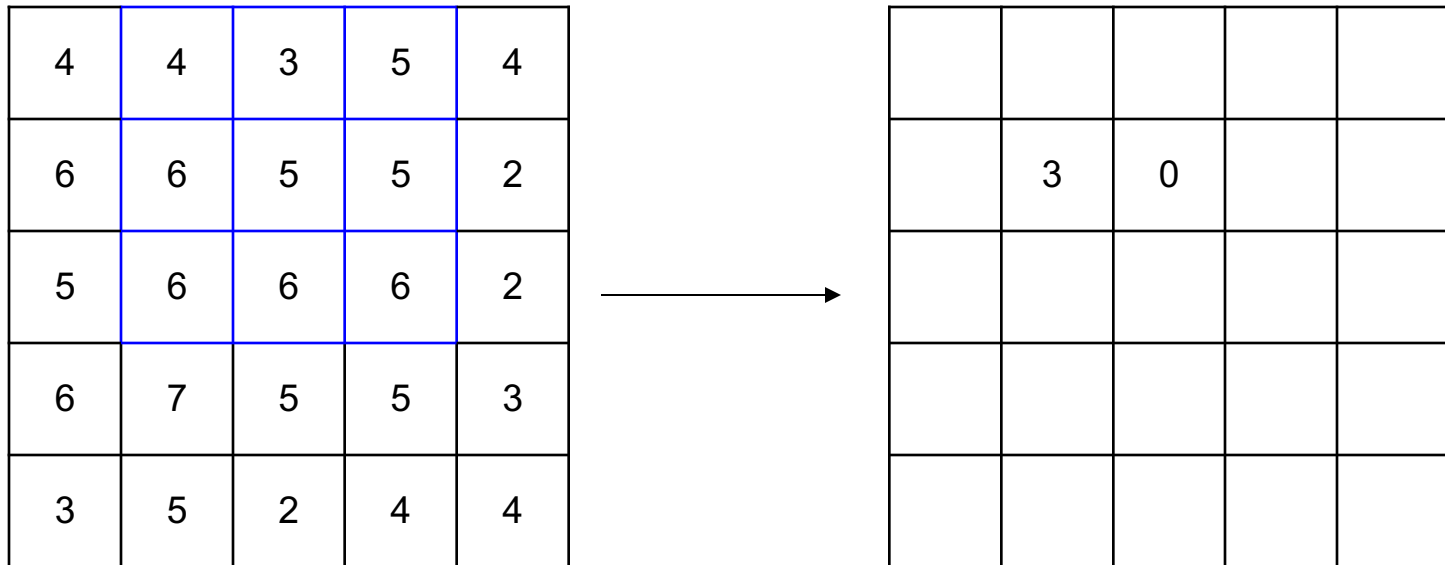


**Hasil konvolusi =3. Nilai ini dihitung dengan cara berikut :**

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

## Contoh Operasi Konvolusi [2]

- Geser kernel satu pixel ke kanan, kemudian hitung nilai pixel pada posisi **(0,0)** dari kernel

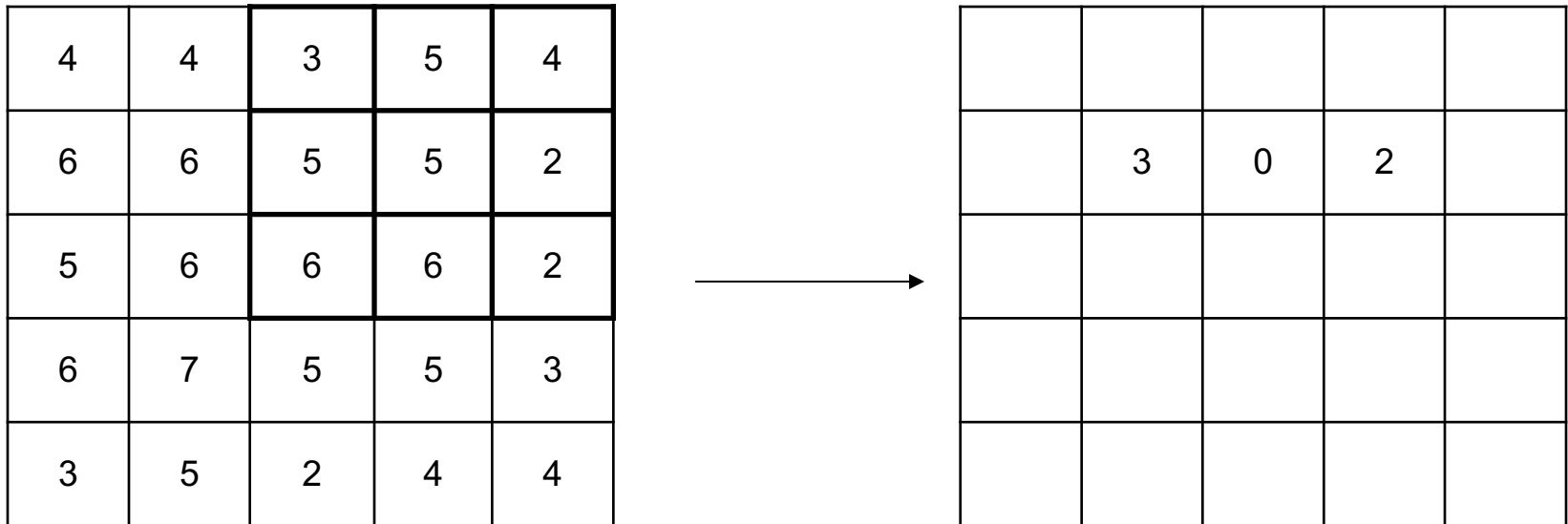


**Hasil konvolusi =0. Nilai ini dihitung dengan cara berikut :**

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

# Contoh Operasi Konvolusi [3]

- Geser kernel satu pixel ke kanan, kemudian hitung nilai pixel pada posisi (0,0) dari kernel

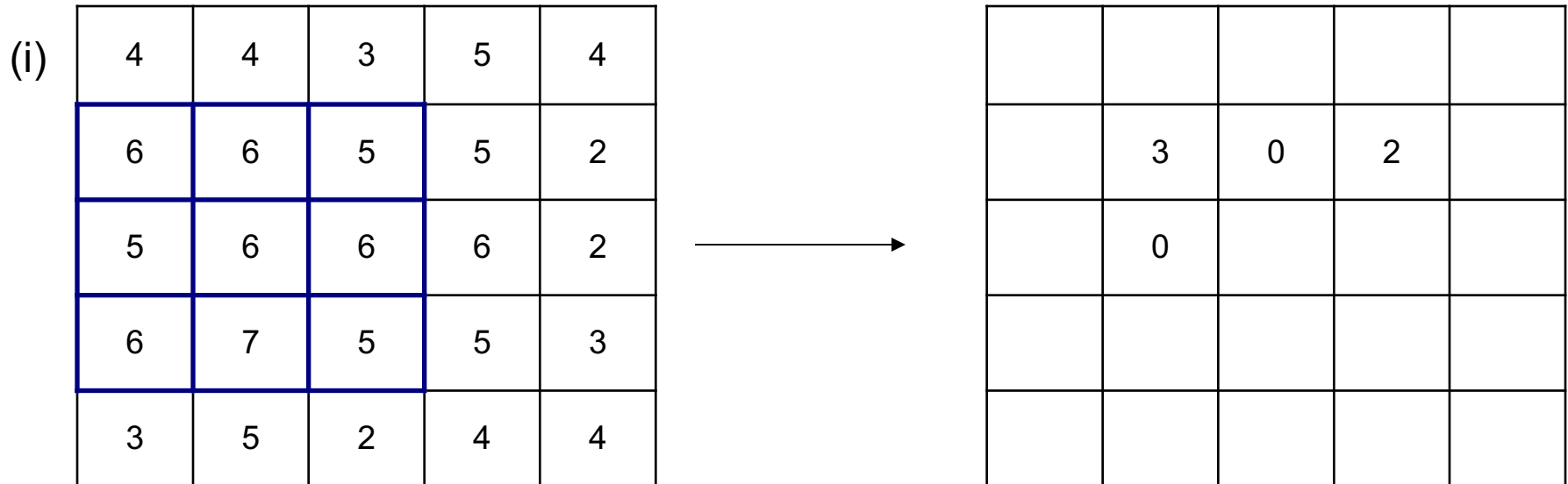


**Hasil konvolusi =2. Nilai ini dihitung dengan cara berikut :**

$$(0 \times 3) + (-1 \times 5) + (0 \times 4) + (-1 \times 5) + (4 \times 5) + (-1 \times 2) + (0 \times 6) + (-1 \times 6) + (0 \times 2) = 2$$

## Contoh Operasi Konvolusi [4]

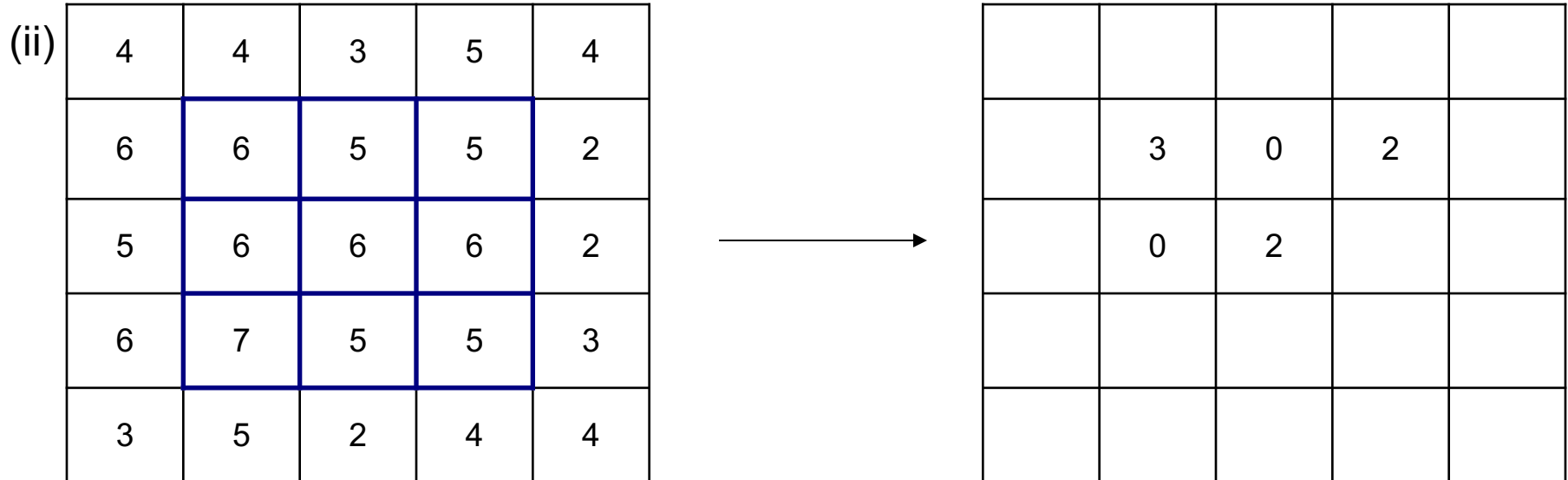
- Selanjutnya, Geser *Kernel* Satu *Pixel* ke bawah, lalu mulai lagi melakukan Konvolusi dari sisi kiri citra. Setiap kali Konvolusi, Geser *Kernel* Satu *Pixel* Ke Kanan:



**Hasil konvolusi =0. Nilai ini dihitung dengan cara berikut :**

$$(0 \times 6) + (-1 \times 6) + (0 \times 5) + (-1 \times 5) + (4 \times 6) + (-1 \times 6) + (0 \times 6) + (-1 \times 7) + (0 \times 5) = 0$$

## Contoh Operasi Konvolusi [4]



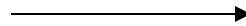
**Hasil konvolusi =2. Nilai ini dihitung dengan cara berikut :**

$$(0 \times 6) + (-1 \times 5) + (0 \times 5) + (-1 \times 6) + (4 \times 6) + (-1 \times 6) + (0 \times 7) + (-1 \times 5) + (0 \times 5) = 2$$

# Contoh Operasi Konvolusi [4]

(iii)

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	2	6	

**Hasil konvolusi =6. Nilai ini dihitung dengan cara berikut :**

$$(0 \times 5) + (-1 \times 5) + (0 \times 2) + (-1 \times 6) + (4 \times 6) + (-1 \times 2) + (0 \times 5) + (-1 \times 5) + (0 \times 3) = 6$$

## Contoh Operasi Konvolusi [4]

- Dengan cara yang sama seperti tadi , maka *pixel – pixel* pada baris ke tiga dikonvolusi sehingga menghasilkan :

	3	0	2	
	0	2	6	
	6	0	2	

Sebagai catatan, Jika hasil Konvolusi menghasilkan nilai Pixel negatif, maka nilai tersebut di jadikan 0, sebaliknya jika hasil Konvolusi menghasilkan nilai *pixel* lebih besar dari nilai keabuan maksimum, maka nilai tersebut dijadikan nilai keabuan maksimum



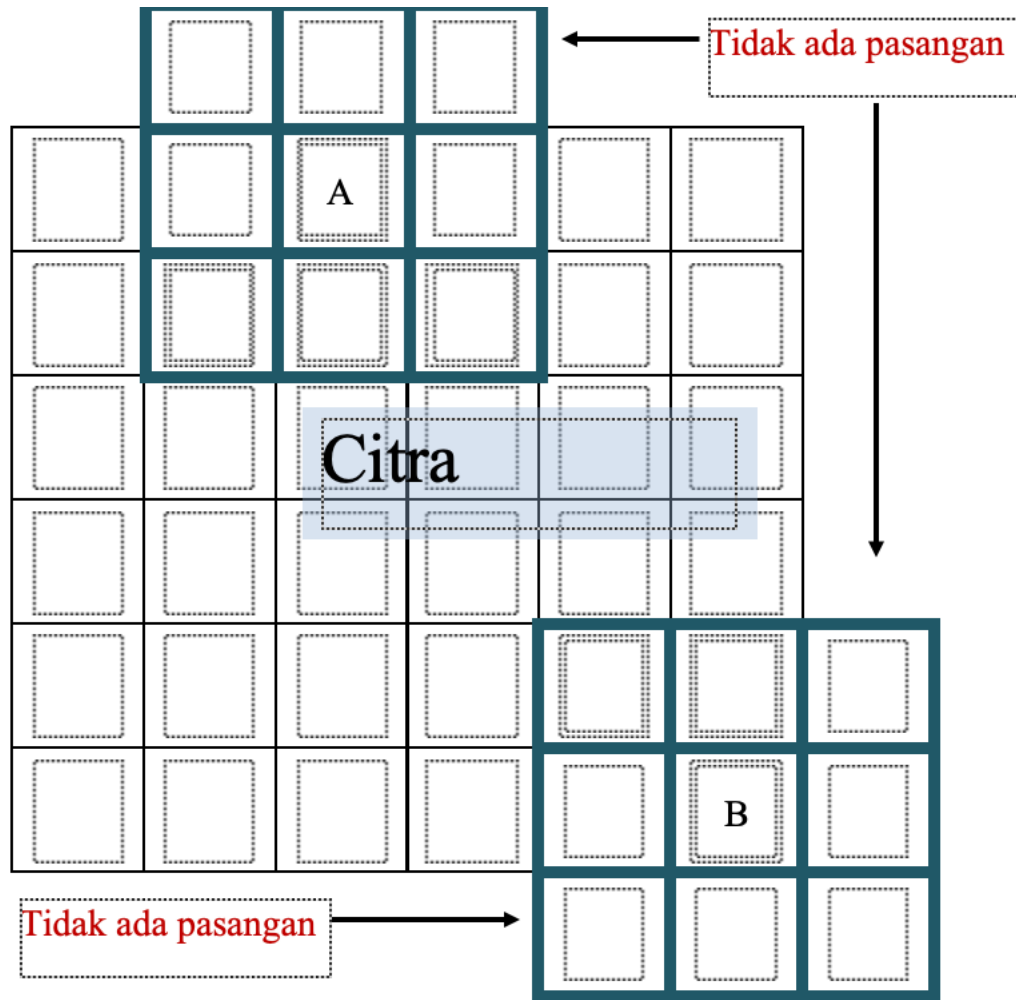
# Contoh Operasi Konvolusi [4]

Masalah timbul bila *Pixel* yang di konvolusi adalah *Pixel* pinggir ( border), karena beberapa Koefisien Konvolusi tidak dapat di Posisikan pada *Pixel* – *pixel* Citra ( “Efek Menggantung” ), seperti contoh di bawah ini:

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	

Masalah “Menggantung” Seperti ini Selalu Terjadi pada Pixel – pixel pinggir kiri, kanan, atas, dan bawah.

# Ilustrasi



# Contoh Operasi Konvolusi

1. **Piksel pinggir di abaikan**, tidak di – Konvolusi. Solusi ini banyak di pakai di dalam pustaka fungsi – fungsi pengolahan citra. Dengan cara seperti ini, maka pixel – pixel pinggir nilainya sama seperti citra asal.
2. **Dupliaksi elemen citra**, misalnya elemen kolom pertama disalin ke kolom  $M+1$ , begitu juga sebaliknya, lalu konvolusi *piksel* pinggir tersebut.
3. **Buat baris tambahan pada bagian tepi.** → Baris dan kolom ditambahkan pada bagian tepi sehingga proses konvolusi dapat dilaksanakan. Dalam hal ini, baris dan kolom baru diisi dengan nilai 0.
4. **Ambil bagian yang tidak punya pasangan dengan bagian lain dari citra.** Indeks melingkar dilaksanakan dengan mengambil data pada posisi di seberang citra, sedangkan indeks tercermin diambilkan dari baris/kolom yang ada di dekatnya.

# Hasil Konvolusi



(a) Citra asli



(b) Bagian tepi diberi nilai nol



(c) Pengisian dari baris atau kolom terpinggir



(d) Pengisian dengan pencerminan

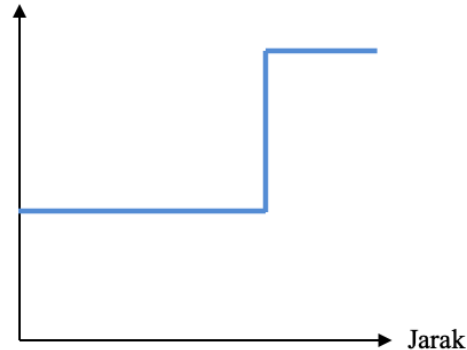


(e) Pengulangan dari tepi yang berseberangan

# Domain Frekuensi

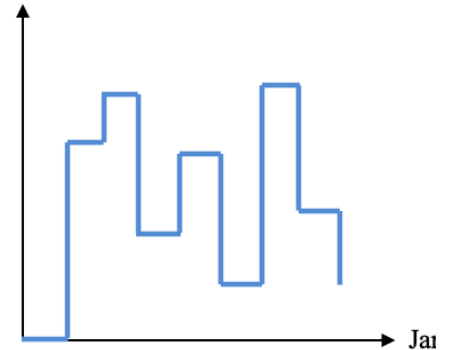
Frekuensi domain pada citra menunjukkan seberapa sering suatu perubahan aras keabuan terjadi dari suatu posisi ke posisi berikutnya. Gambar disamping menunjukkan secara visual perbedaan antara frekuensi rendah dan frekuensi tinggi. Pada citra berfrekuensi tinggi, perubahan aras sering terjadi seiring dengan pergeseran jarak.

Aras keabuan



(a) Frekuensi rendah

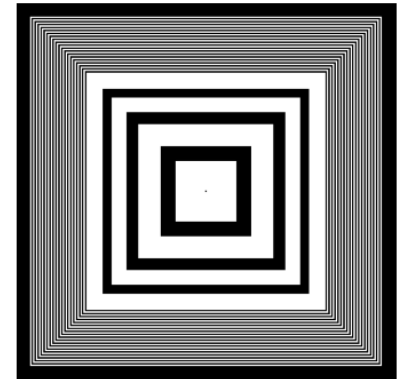
Aras keabuan



(b) Frekuensi tinggi



(c) Citra dengan frekuensi rendah



(d) Citra berfrekuensi tinggi

# Jenis – jenis filter frekuensi

- Smoothing filters:
  - Lowpass filter
  - Highpass filter
  - Highboost Filter

# Lowpass filter

- Filter lolos-bawah (*low-pass filter*) adalah filter yang mempunyai sifat dapat meloloskan yang berfrekuensi rendah dan menghilangkan yang berfrekuensi tinggi.
- Efek filter ini membuat perubahan aras keabuan menjadi lebih lembut
- Adapun yang melibatkan konvolusi menggunakan kernel antara lain berupa seperti yang terlihat pada gambar berikut

1/6	0	1	0
	1	2	1
	0	1	0
#1			

1/9	1	1	1
	1	1	1
	1	1	1
#2			

1/10	1	1	1
	1	2	1
	1	1	1
#3			

1/16	1	2	1
	2	4	2
	1	2	1
#4			

# Contoh

40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128

(a) Citra dengan frekuensi rendah

128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40

(b) Citra dengan frekuensi tinggi

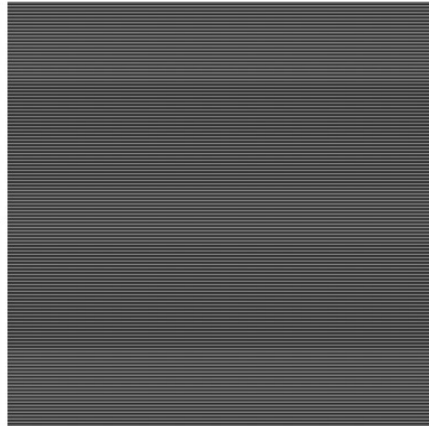
Dengan menggunakan kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1



(c) Citra dengan frekuensi rendah



(d) Citra dengan frekuensi tinggi



# Hasil lowpass Filter

40	40	40
40	40	40
40	40	40
40	40	40
40	40	40
40	40	40
40	40	40
69	69	69
99	99	99
128	128	128
128	128	128
128	128	128
128	128	128
128	128	128
128	128	128
128	128	128
128	128	128
128	128	128

(a) Citra dengan frekuensi rendah

99	99	99
69	69	69
99	99	99
69	69	69
99	99	99
69	69	69
99	99	99
69	69	69
99	99	99
69	69	69
99	99	99
69	69	69
99	99	99
69	69	69
99	99	99
69	69	69

(b) Citra dengan frekuensi tinggi

Gambar (a). filter tidak membuat perubahan yang sangat berarti pada citra kecuali perubahan pada baris yang berisi 69 dan 99.

Gambar (b), frekuensi memang tidak berubah, tetapi terjadi penghalusan perubahan aras (128 menjadi 99 dan 40 menjadi 69).

Melalui filter lowpass, hal-hal yang menyatakan frekuensi tinggi akan diredupkan, sedangkan bagian berfrekuensi rendah hampir tidak berubah.



(a) Citra boneka yang dilengkapi derau



(b) Hasil penapisan citra boneka

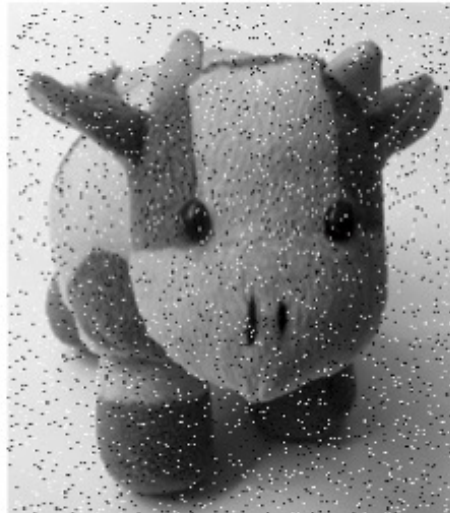


(c) Citra goldhill

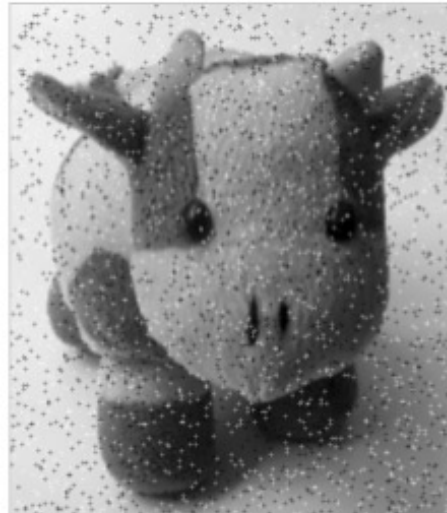


(b) Hasil penapisan citra goldhill

filter lowpass mampu menghaluskan perubahan-perubahan yang drastis. Perhatikan ketajaman genting pada *goldhill* menjadi diperhalus setelah melalui penapisan. Begitu pula derau pada boneka.



(a) Citra boneka yang dilengkapi derau



(b) Hasil dengan kernel #1



(c) Hasil dengan kernel #2



(b) Hasil dengan kernel #4

Hasil Low pass  
filter dengan  
perbedaan kernel

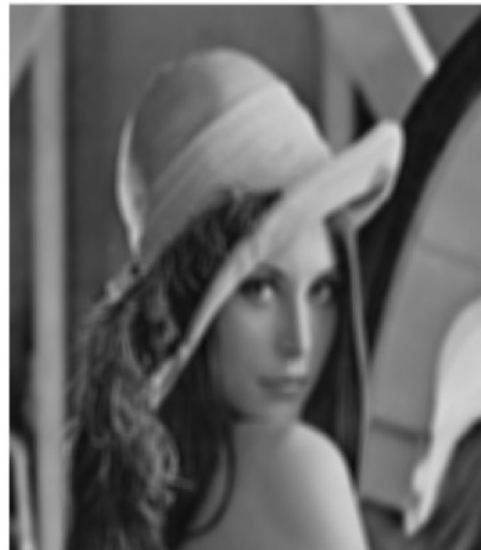
**Efek pemakaian  
filter lolos-rendah  
dengan berbagai  
ukuran kernel.**



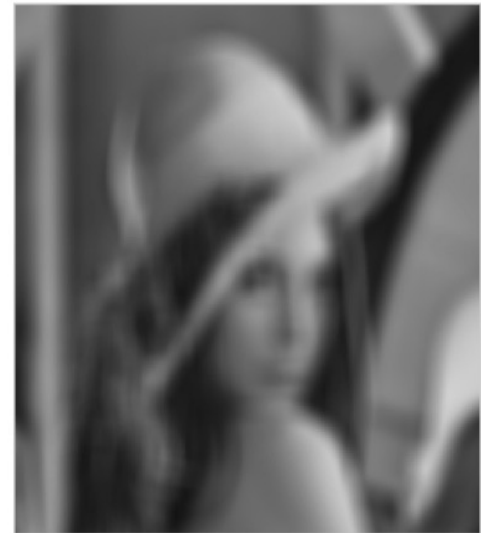
(a) Citra lena256



(b) Hasil dengan kernel 3x3



(c) Hasil dengan kernel 5x5



(b) Hasil dengan kernel 13x13

# Highpass filter

- Filter highpass adalah filter yang ditujukan untuk melewatkan frekuensi tinggi dan menghalangi yang berfrekuensi rendah. Hal ini biasa dipakai untuk mendapatkan tepi objek dalam citra atau menajamkan citra
- Filter lolos-tinggi mempunyai sifat yaitu jumlah seluruh koefisien adalah nol

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

-1	-1	-1
-1	8	-1
-1	-1	-1
#2		

1	-2	1
-2	4	-2
1	-2	1
#3		

# Contoh

40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
40	40	40	40	40
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128
128	128	128	128	128

(a) Citra dengan frekuensi rendah

128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40
128	128	128	128	128
40	40	40	40	40

(b) Citra dengan frekuensi tinggi

Jika kernel seperti berikut

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

# Hasil

0	0	0	0	0	0
0	0	0	176	176	176
0	0	0	0	0	0
0	0	0	176	176	176
0	0	0	0	0	0
0	0	0	176	176	176
0	0	0	0	0	0
0	0	0	176	176	176
88	88	88	0	0	0
0	0	0	176	176	176
0	0	0	0	0	0
0	0	0	176	176	176
0	0	0	0	0	0
0	0	0	176	176	176
0	0	0	0	0	0
0	0	0	176	176	176
0	0	0	0	0	0
0	0	0	176	176	176
0	0	0	0	0	0
0	0	0	176	176	176

| (a) Citra dengan frekuensi rendah |  |  | (b) Citra dengan frekuensi tinggi |  |  |

Gambar (a) menunjukkan bahwa hanya pada perbatasan antara perubahan aras keabuan yang ditonjolkan (baris berisi 88) dan nilai yang lain bernilai rendah (nol). Dengan demikian, akan muncul garis putih.

Gambar

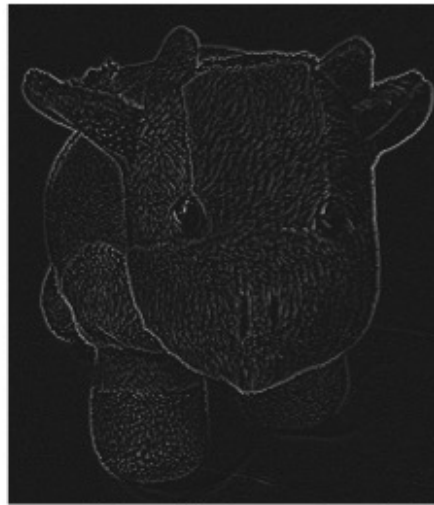
(b) menunjukkan bahwa citra yang berfrekuensi tinggi hampir tidak mengalami perubahan, kecuali nilainya saja yang berefek pada penajaman perbedaan aras keabuan (nilai 150 menjadi 176 dan nilai 40 menjadi 0).



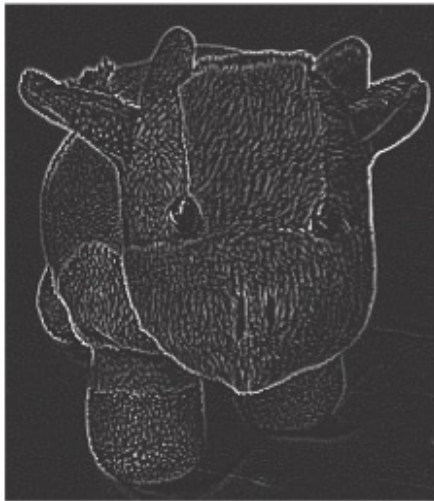
# Hasil high pass filter



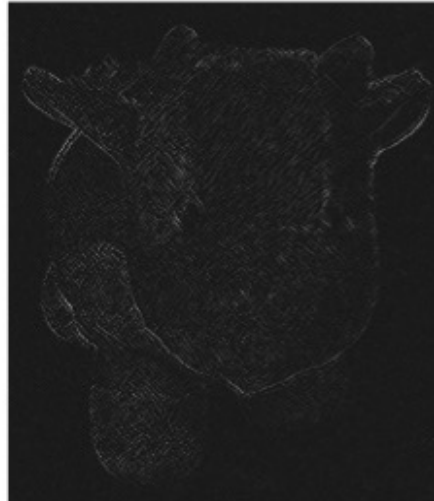
(a) Citra boneka.png



(b) Hasil dengan kernel #1



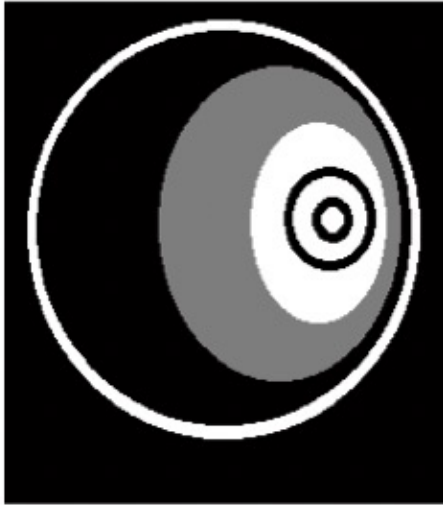
(c) Hasil dengan kernel #2



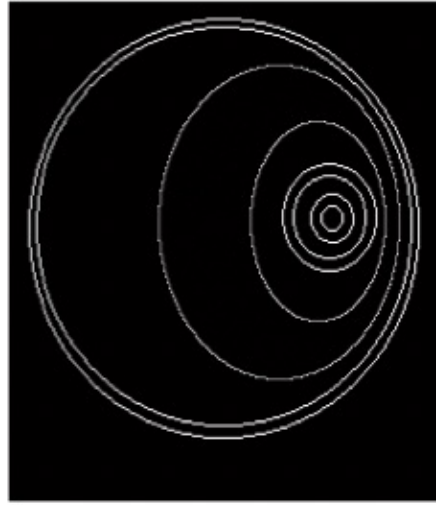
(b) Hasil dengan kernel #3



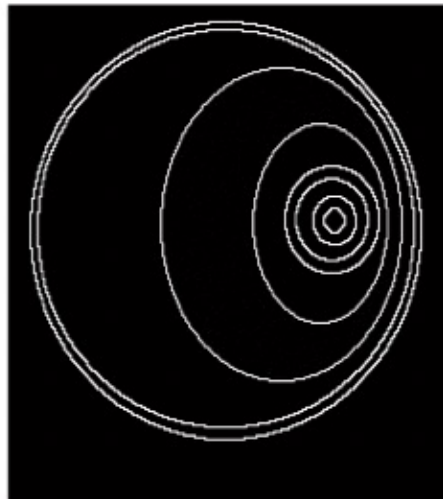
# Hasil high pass filter



(a) Citra bulat.png



(b) Hasil dengan kernel #1



(c) Hasil dengan kernel #2



(b) Hasil dengan kernel #3

# Filter Highboost

- Filter “high boost” dapat digunakan untuk menajamkan citra melalui konvolusi.
- Kernel yang dapat dipakai adalah kernel filter lolos-tinggi dengan nilai di pusat diisi dengan nilai yang lebih besar daripada nilai pada posisi tersebut untuk filter lolos-tinggi.

-1	-1	-1
-1	c	-1
-1	-1	-1

$c > 8$ ; misalnya 9

efek saat  $c$  diisi  
dengan 9, 10, dan  
11

Tampak bahwa dengan menggunakan filter *high boost* bernilai tengah tertentu (pada contoh di atas berupa 9), diperoleh hasil berupa penajaman citra.



(a) Citra boneka



(b) Hasil untuk  $c=9$



(c) Hasil untuk  $c=10$



(d) Hasil untuk  $c=11$

# LATIHAN

Hitunglah nilai citra dibawah ini dengan

- teknik konvolusi
- Filter median
- Filter lowpass

$$f(x,y) = \begin{pmatrix} 4 & 4 & 3 & 5 & 4 \\ 6 & 6 & 5 & 5 & 2 \\ 5 & 6 & 6 & 6 & 2 \\ 6 & 7 & 5 & 5 & 3 \\ 3 & 5 & 2 & 4 & 4 \end{pmatrix}$$

**Citra 5 x 5**

$$g(x,y) = \begin{pmatrix} 0 & -1 & 0 \\ -2 & \bullet & 3 & -2 \\ 0 & -1 & 0 \end{pmatrix}$$

**Kernel 3 x 3**

Tanda  menyatakan posisi (0,0) dari kernel