

# 18 - Pendeteksian Tepi (Bagian 1)

IF4073 Pemrosesan Citra Digital

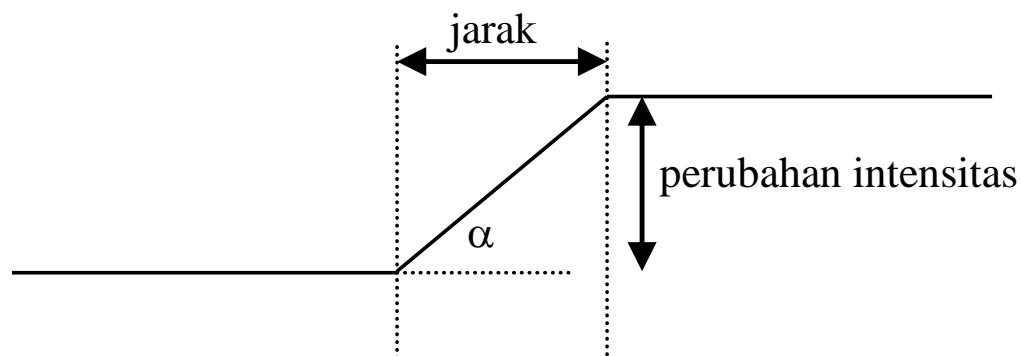
Oleh: Rinaldi Munir



Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung  
2024

# Tepi (*edge*)

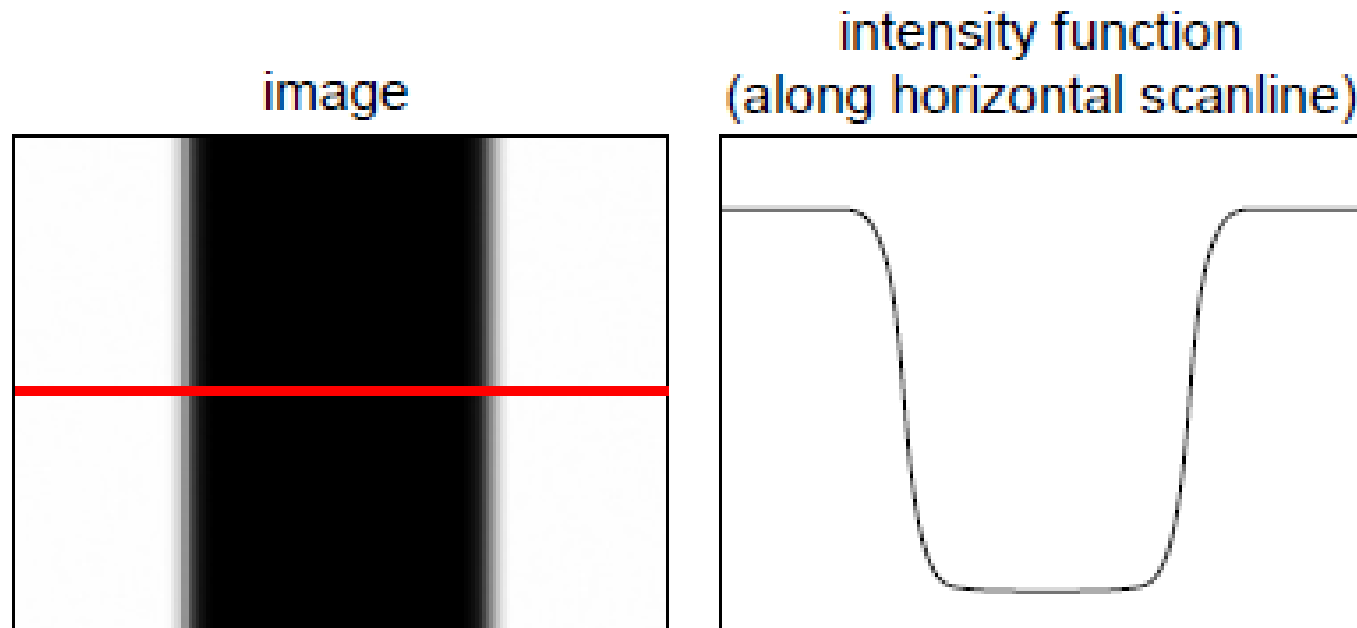
- Tepi (*edge*) adalah **perubahan** nilai intensitas nilai keabuan yang mendadak (besar) dalam jarak yang singkat.



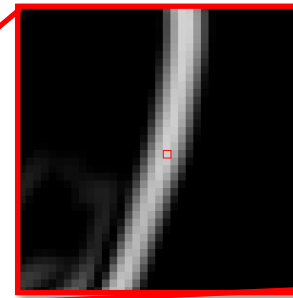
$\alpha = \text{arah tepi}$

- Tepi memiliki **arah**, dan arah ini berbeda-beda bergantung pada perubahan intensitas

- Tepi biasanya terdapat pada batas antara dua daerah yang berbeda intensitas dengan perubahan yang sangat cepat di dalam citra.

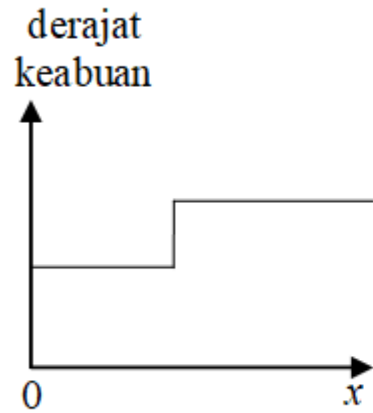


# Di mana tepi?

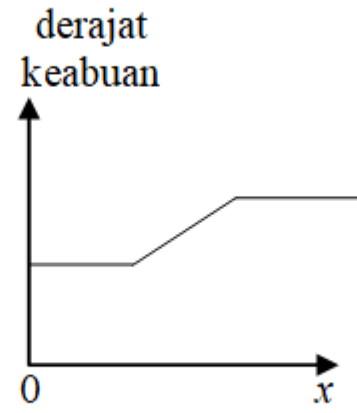


Ini tepi

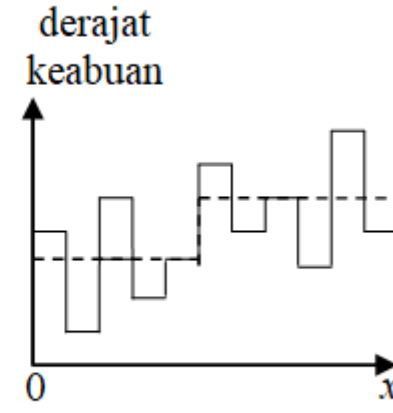
- Empat macam tepi: *tepi curam (step edge)*, *tepi landai (ramp edge)*, *tepi garis (line edge)*, dan *tepi atap (roof edge)*.



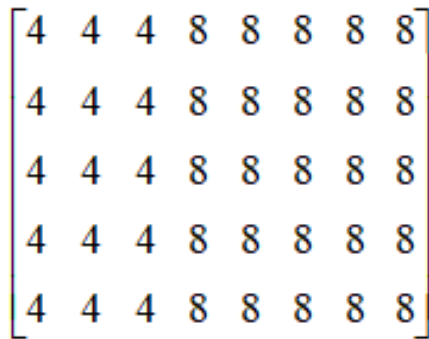
(a) Tepi curam



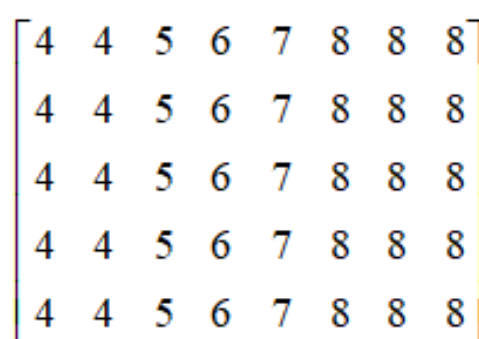
(b) tepi landai



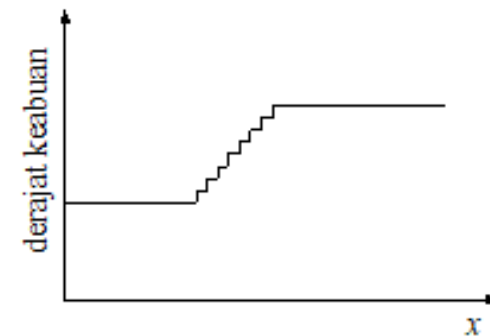
(c) tepi curam dengan derau



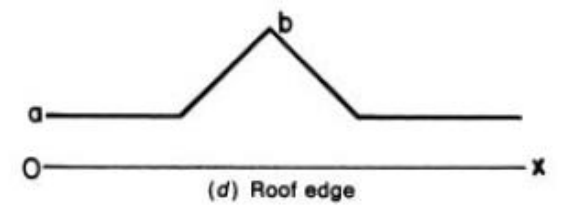
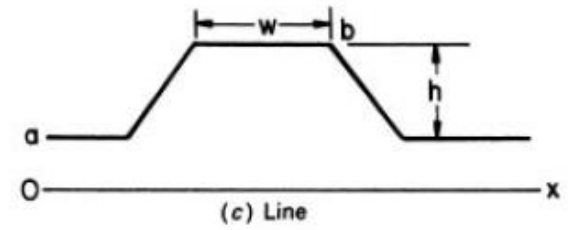
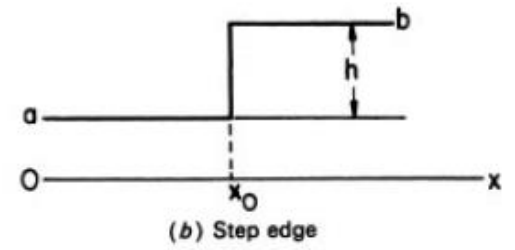
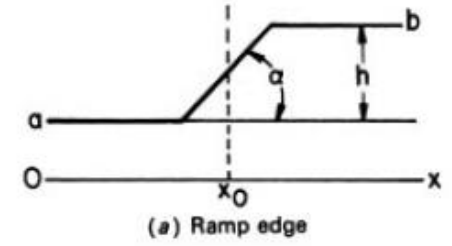
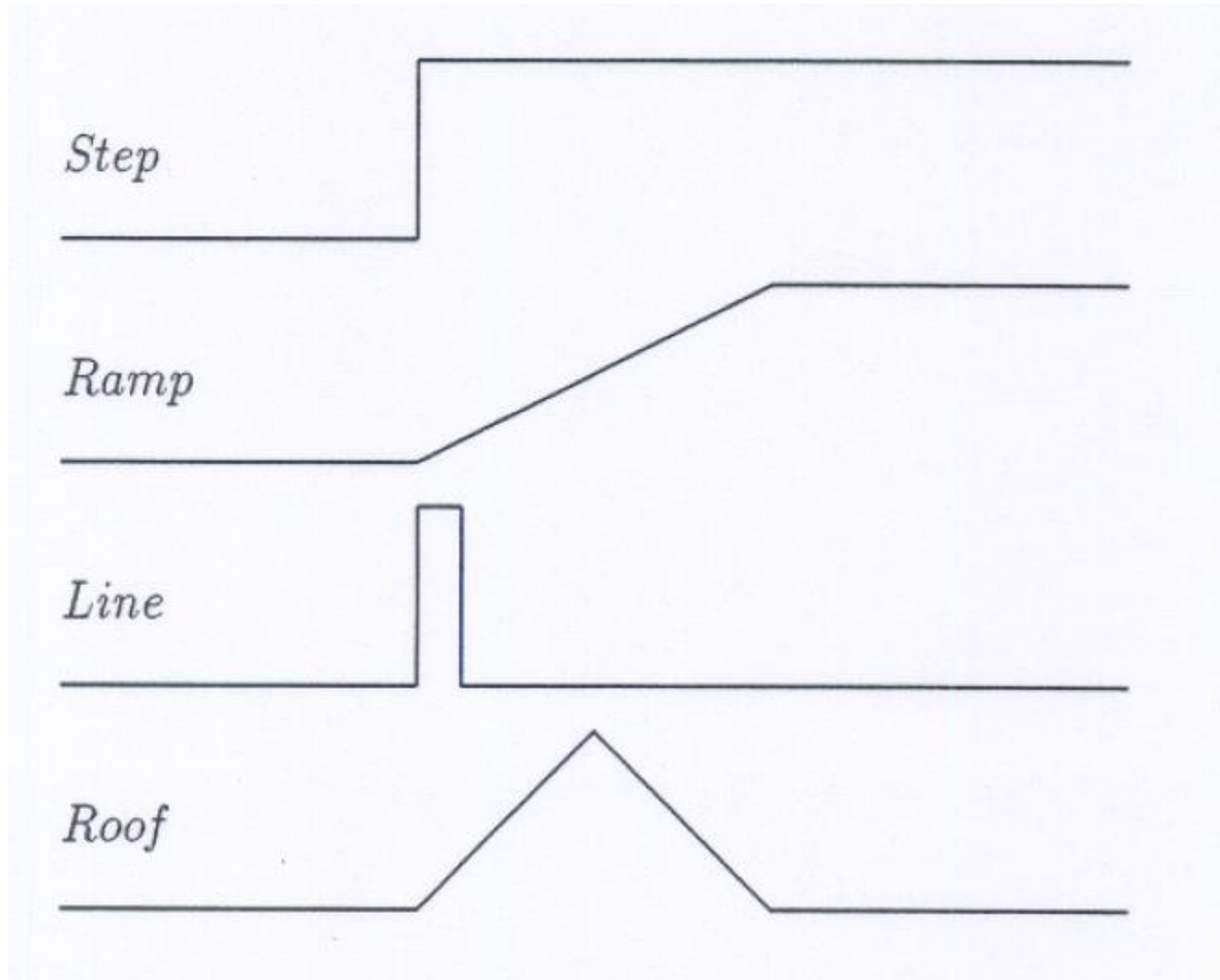
Citra dengan tepi curam



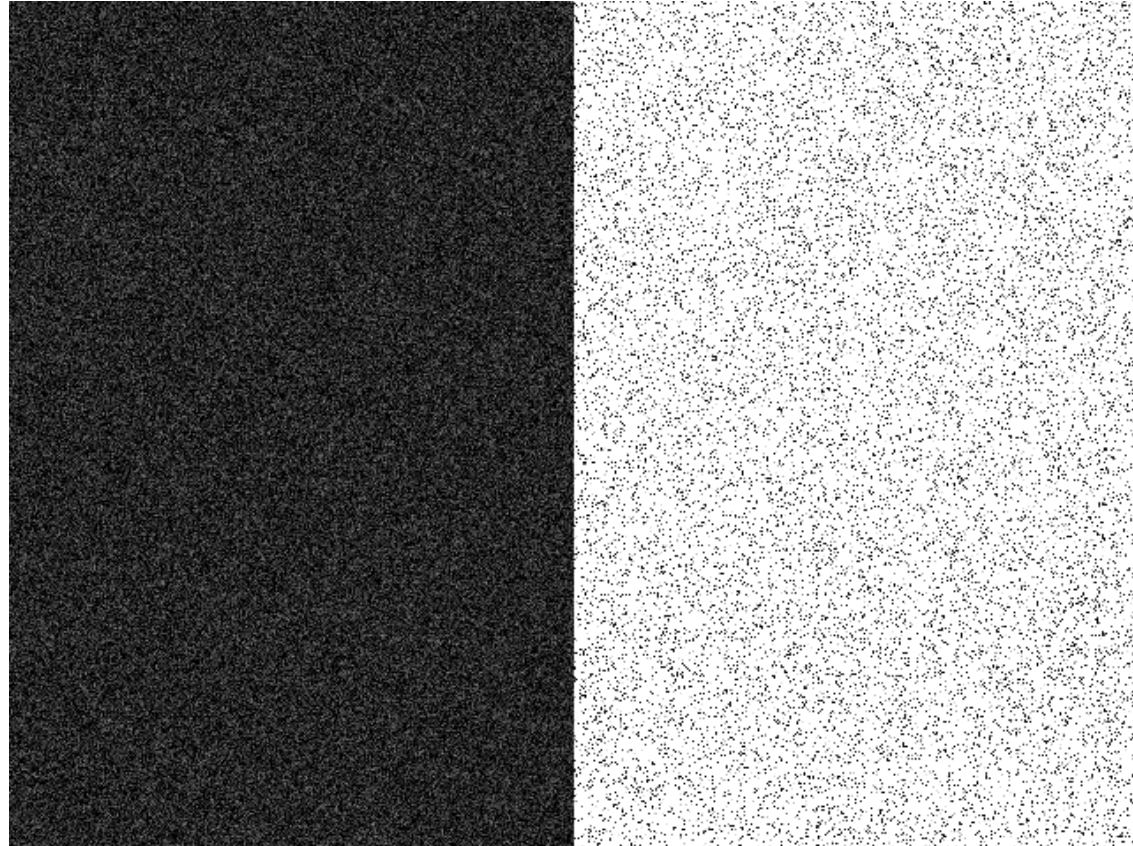
Citra dengan tepi landai



Breakdown tepi landai

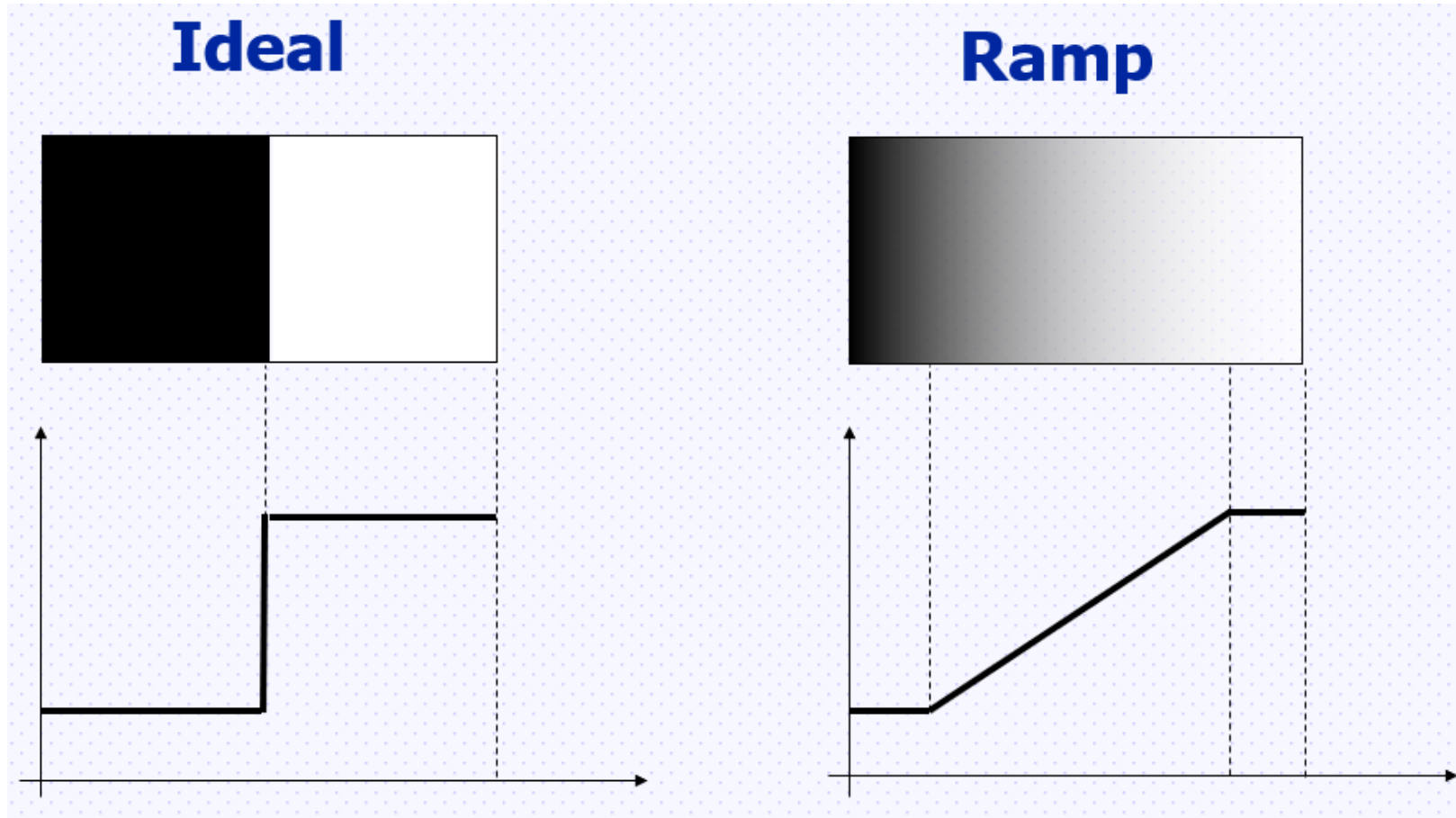


## Citra mengandung derau



Harus bisa membedakan derau dengan tepi yang sebenarnya

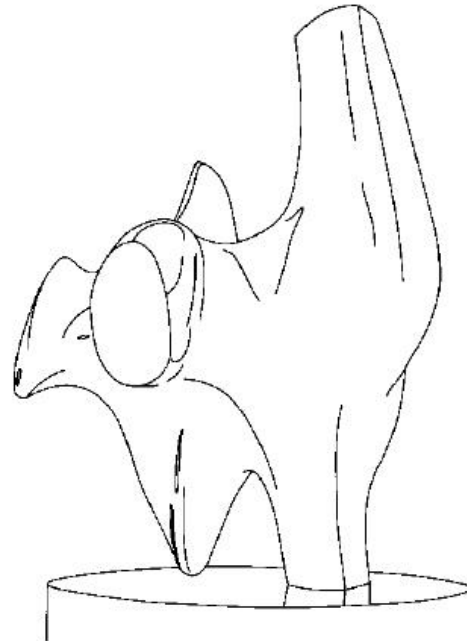
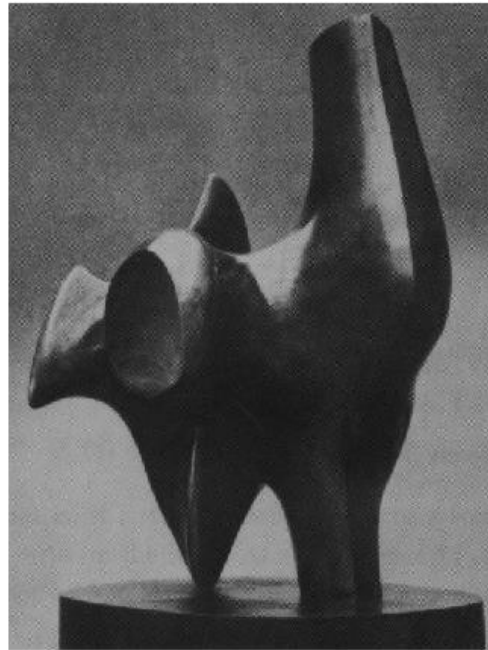
- Idealnya sebuah tepi berbentuk curam (kemiringan 90 derajat), namun kebanyakan tepi berbentuk landai (*ramp*) dan tepi yang mengandung derau





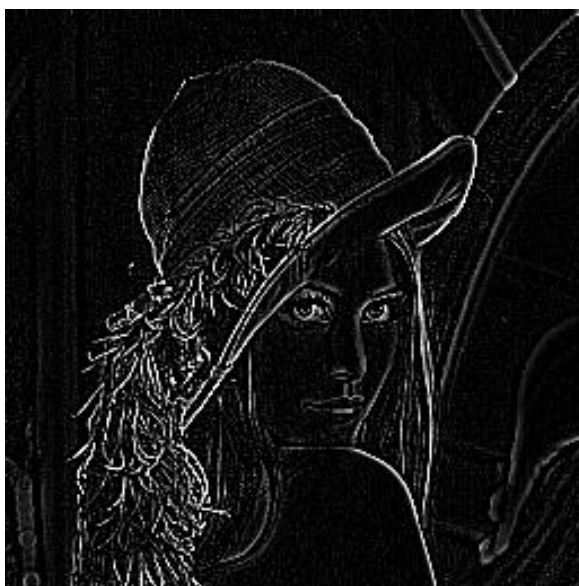
# Tujuan Pendeteksian Tepi

- Pendeteksian tepi bertujuan untuk meningkatkan penampakan garis batas atau objek di dalam citra.



- Pendeteksian tepi mengekstraksi representasi gambar garis-garis di dalam citra.





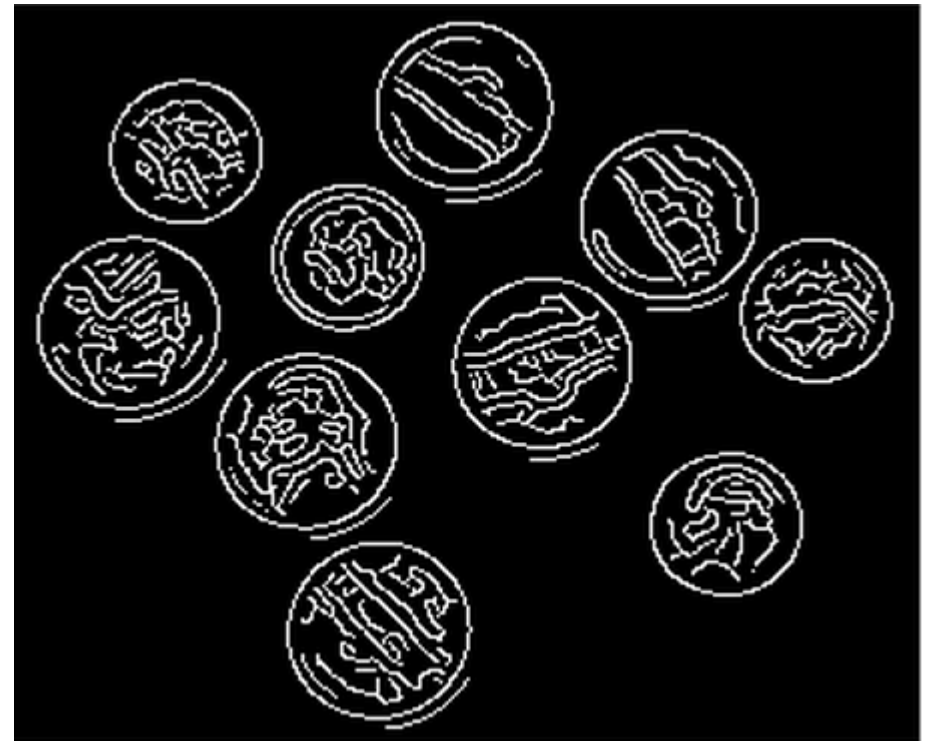
- Pendeteksian tepi berguna dalam mengenali objek di dalam citra (*image recognition*).



a) Complemented Image

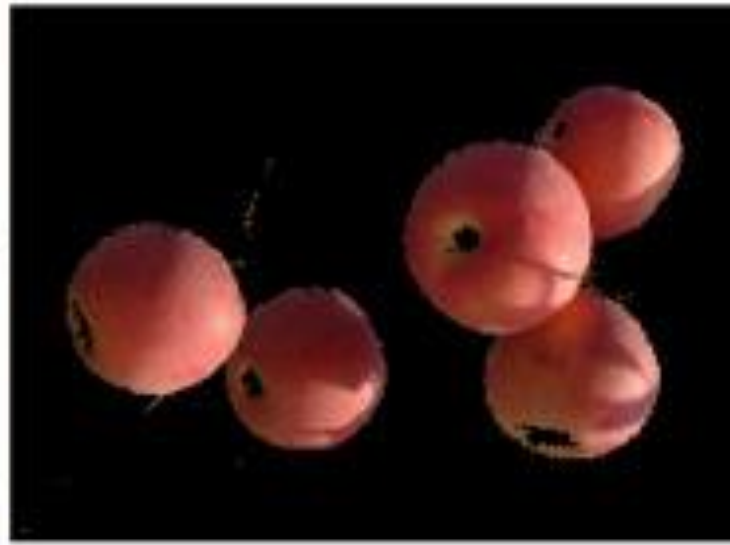


b) Edged Image

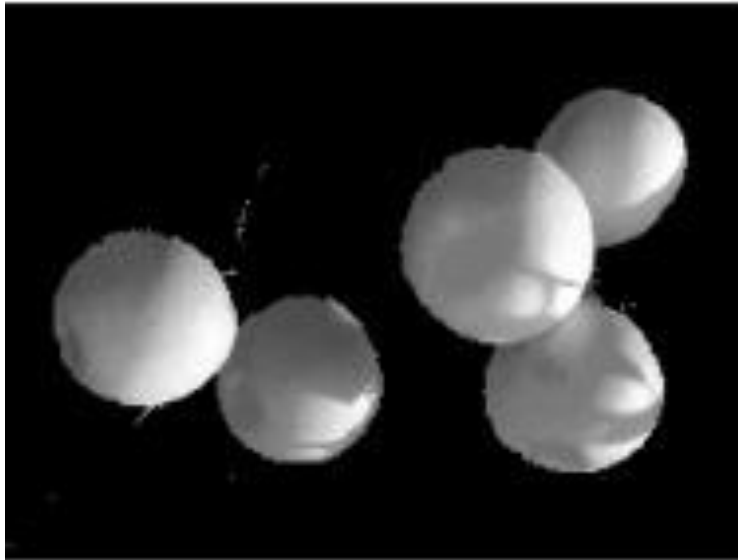




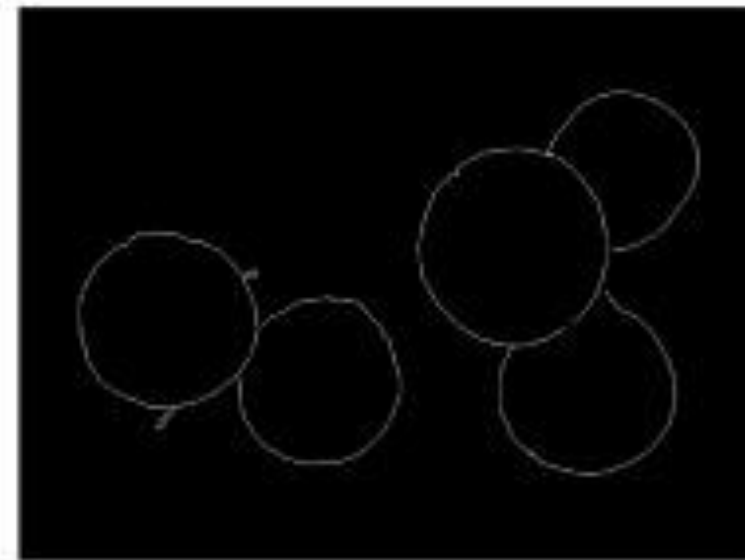
(a) Acquired image.



(b) Segmentation image.



(c) Gray image.



(d) Edge image.

- Pendeteksian tepi merupakan bagian dari **analisis citra** (*image analysis*).
- Tujuan analisis citra: mengidentifikasi parameter-parameter yang diasosiasikan dengan ciri (*feature*) dari objek di dalam citra, untuk selanjutnya parameter tersebut digunakan dalam menginterpretasi citra.
- Analisis citra pada dasarnya terdiri dari tiga tahapan:
  1. **Ekstraksi ciri** (*feature extraction*).

Faktor kunci dalam mengekstraksi ciri adalah kemampuan mendeteksi keberadaan tepi (*edge*) dari objek di dalam citra.
  2. **Segmentasi**

Setelah tepi objek diketahui, langkah selanjutnya dalam analisis citra adalah segmentasi, yaitu mereduksi citra menjadi objek atau region, misalnya memisahkan objek-objek yang berbeda dengan mengekstraksi batas-batas objek (*boundary*).
  3. **Klasifikasi**.

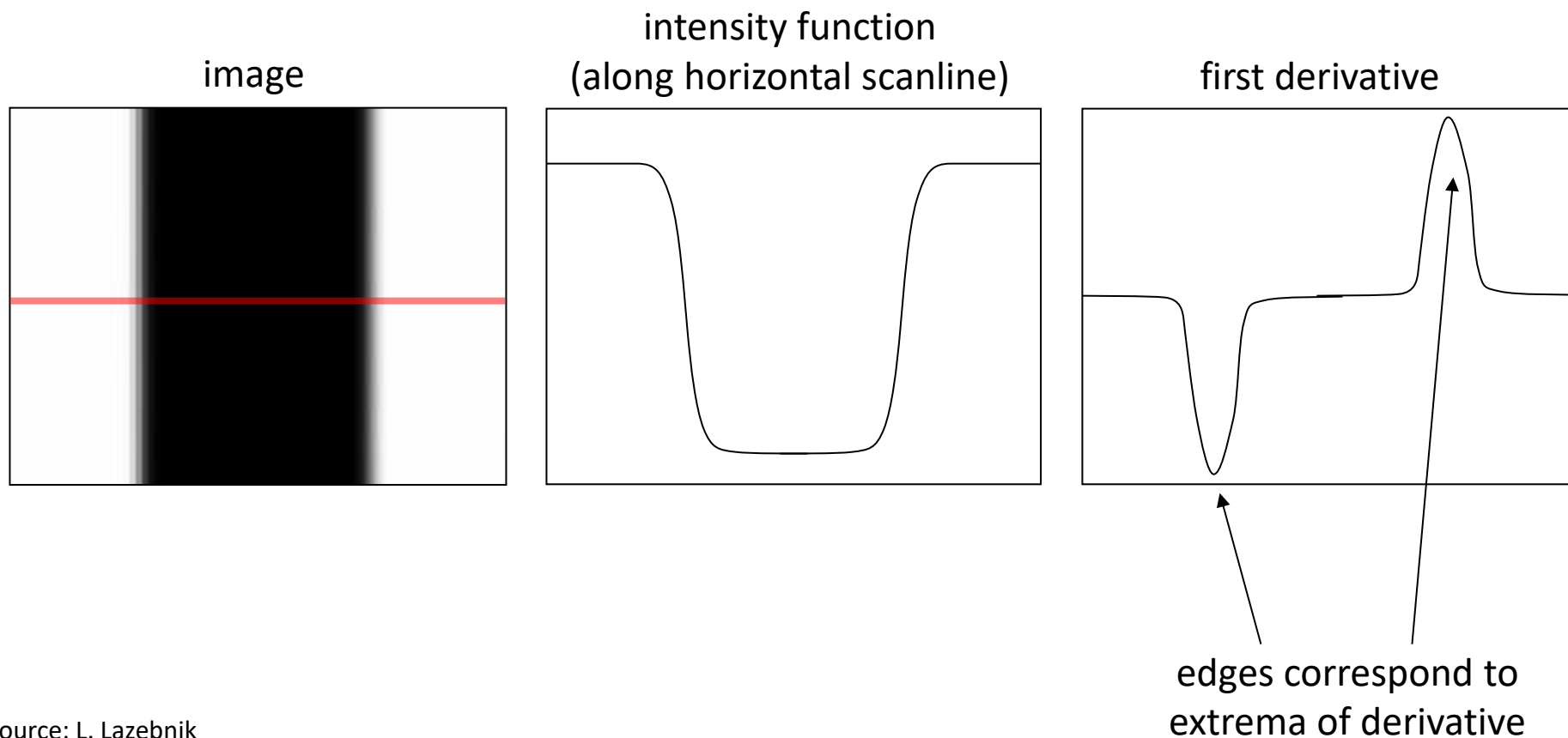
Langkah terakhir dari analisis citra adalah klasifikasi, yaitu memetakan segmen-segmen yang berbeda ke dalam kelas objek yang berbeda pula.

# Operator Gradien

- Pendeteksian tepi dapat dipahami dengan pendekatan kalkulus diferensial.
- Sebab, perubahan intensitas yang besar dalam jarak yang singkat dipandang sebagai fungsi yang memiliki kemiringan yang besar.
- Kemiringan fungsi dapat dihitung dengan turunan pertama (*gradient*)

$$\frac{\partial F}{\partial x} = \lim_{h \rightarrow 0} \frac{F(x+h, y) - F(x, y)}{h}$$





Source: L. Lazebnik

- Karena citra  $f(x,y)$  adalah fungsi dwimatra dalam bentuk diskrit, maka turunan pertamanya adalah secara parsial:

$$\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] = [G_x, G_y]$$

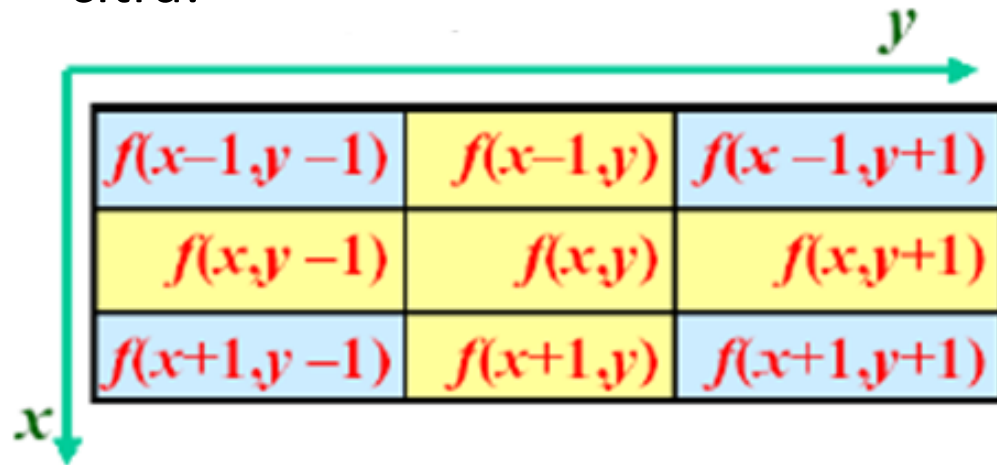
$$G_x = \frac{\partial f(x, y)}{\partial x} = \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x}$$

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

- Biasanya  $\Delta x = \Delta y = 1$  , sehingga

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

Misalkan susunan *pixel-pixel* di dalam citra:



Diferensial maju (*forward differential*):

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

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

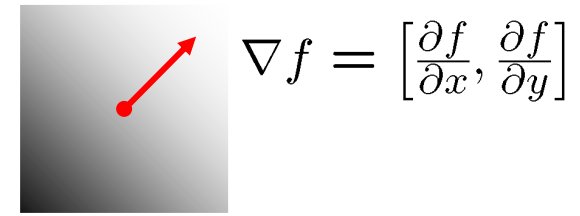
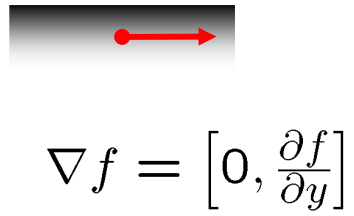
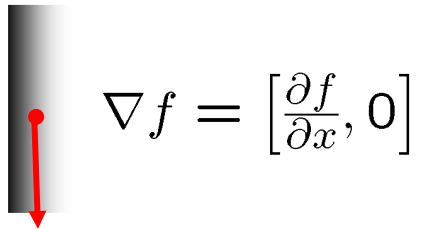
Kedua turunan parsial di atas dapat dipandang sebagai dua buah *mask* konvolusi:

$$G_x = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

- Kekuatan tepi (*edge strengthness*):  $G[f(x,y)] = \sqrt{G_x^2 + G_y^2}$
- Arah tepi:  $\alpha(x,y) = \tan^{-1} \frac{G_y}{G_x}$
- Hasil pendeteksian tepi adalah **citra tepi** (*edges image*):  $g(x, y) = G[f(x, y)]$
- Keputusan apakah suatu *pixel* merupakan tepi atau bukan tepi dinyatakan dengan operasi pengambangan:

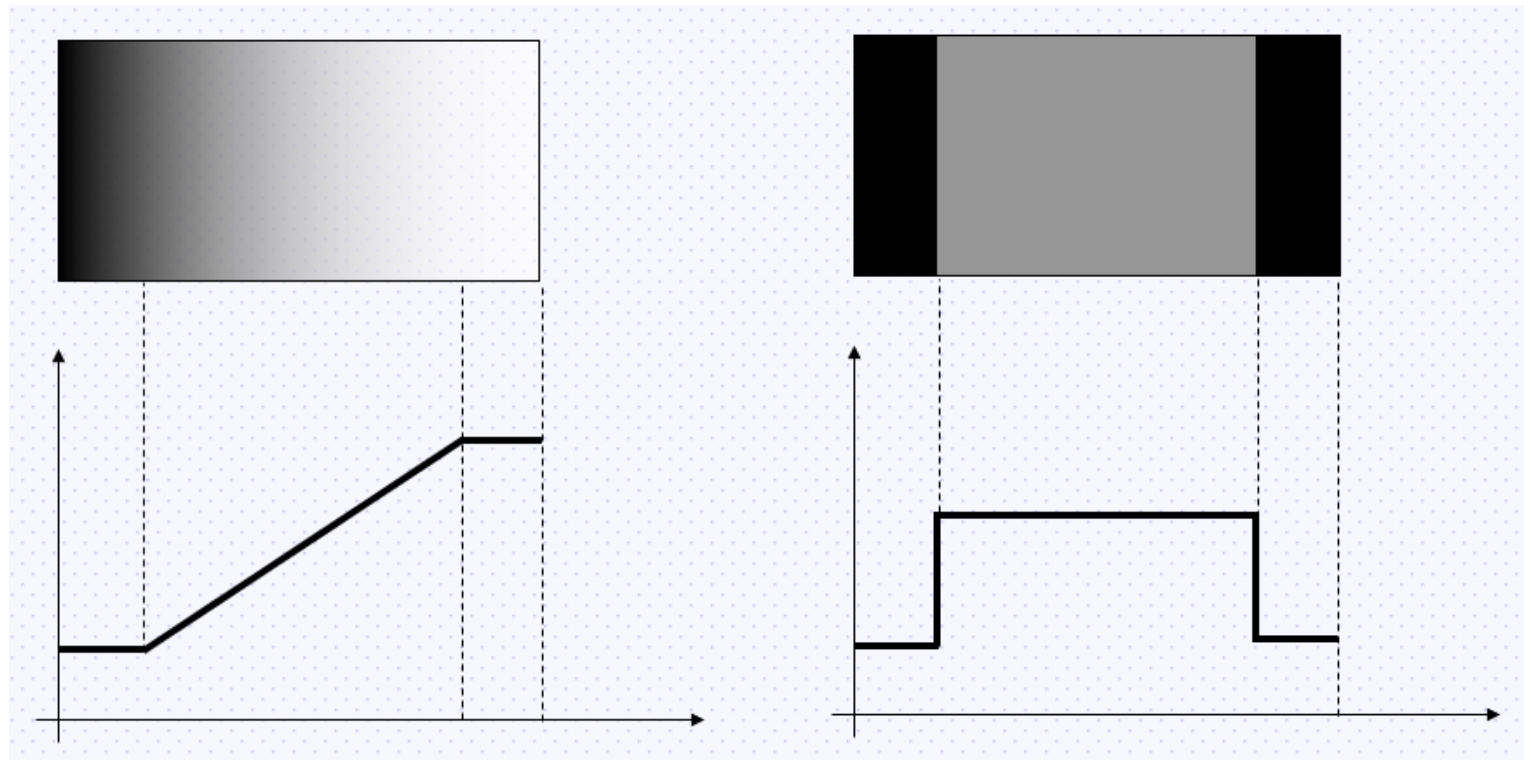
$$g(x, y) = \begin{cases} 1, & \text{jika } G[f(x, y)] \geq T \\ 0, & \text{lainnya} \end{cases}$$

- $T$  adalah nilai ambang, *pixel* tepi dinyatakan putih (1) sedangkan *pixel* bukan tepi dinyatakan hitam (0).



Citra

Gradien (turunan pertama)



**Contoh.** Misalkan terdapat sebuah  $5 \times 5$  citra dengan dua derajat keabuan sebagai berikut:

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

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

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

$$\alpha(x, y) = \tan^{-1} \frac{G_y}{G_x}$$

Hasil perhitungan gradien setiap *pixel* di dalam citra adalah sebagai berikut:

Citra	<u>Gradien-x</u>	<u>Gradien-y</u>	<u>Arah gradien</u>
$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$	0 0 0 0 0	0 0 0 0 *	* * * * *
	0 0 0 -1 -1	0 0 0 0 *	* * * 0° *
	0 0 -1 0 0	0 0 -1 0 *	* * 45° * *
	0 0 0 0 0	0 -1 0 0 *	* 90° * * *
	* * * * *	0 -1 0 0 *	* * * * *

```
%Deteksi tepi dalam arah x dengan operator Gradien
```

```
I = imread('house.jpg');
```

```
Gx = [-1; 1];
```

```
Ix = conv2(double(I), double(Gx), 'same');
```

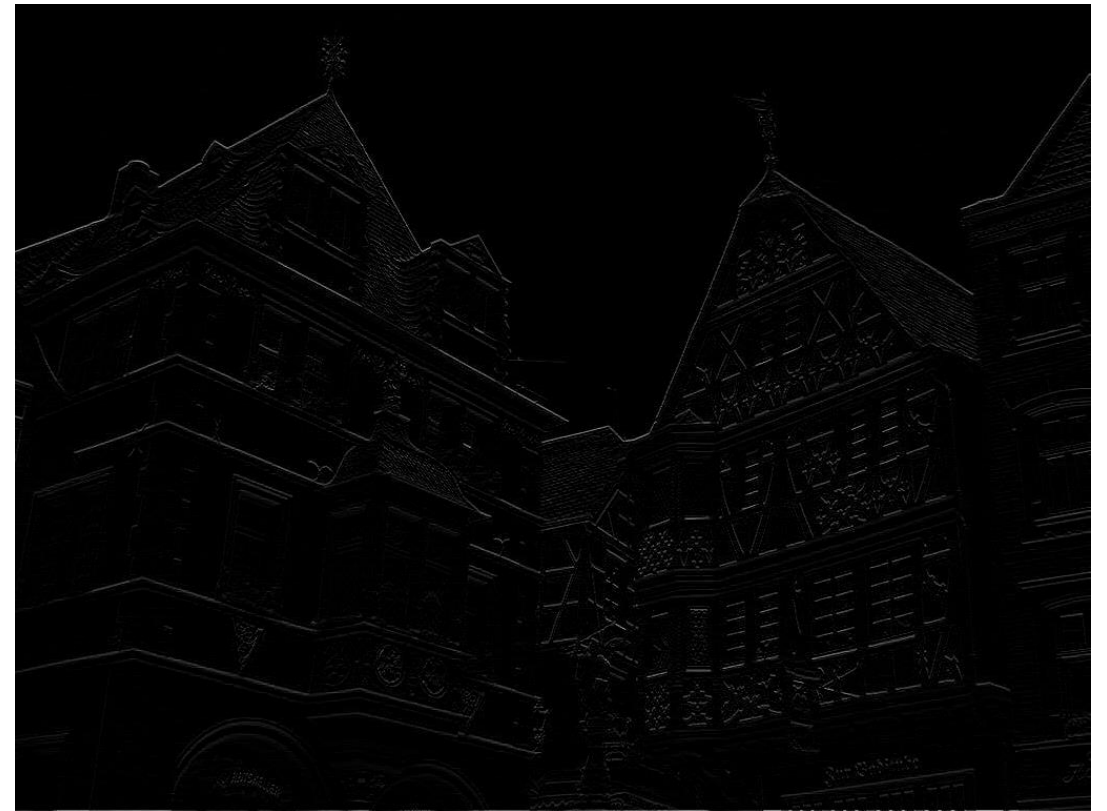
```
Jedge = Ix;
```

```
imshow(I);
```

```
figure, imshow(uint8(Jedge));
```



$f$



$\frac{\partial f}{\partial x}$

```
%Deteksi tepi dalam arah y dengan operator Gradien
```

```
I = imread('house.jpg');
```

```
Gy = [-1 1];
```

```
Iy = conv2(double(I), double(Gy), 'same');
```

```
Jedge = Iy;
```

```
imshow(I);
```

```
figure, imshow(uint8(Jedge));
```



$f$



$\frac{\partial f}{\partial y}$



%Deteksi tepi dalam arah x dan y dengan operator Gradien

```
I = imread('house.jpg');
```

```
Gx = [-1; 1];
```

```
Gy = [-1 1];
```

```
Ix = conv2(double(I), double(Gx), 'same');
```

```
Iy = conv2(double(I), double(Gy), 'same');
```

```
Jedge = sqrt(Ix.^2 + Iy.^2);
```

```
imshow(I);
```

```
figure, imshow(uint8(Jedge));
```



$f$



$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

- Kembali ke rumus kekuatan tepi:  $G[f(x,y)] = \sqrt{G_x^2 + G_y^2}$
- Karena menghitung akar lebih kompleks dan menghasilkan nilai riil, maka dalam praktek kekuatan tepi disederhanakan perhitungannya dengan menggunakan salah satu dari alternatif:

(i)  $G[f(x,y)] = |G_x^2| + |G_y^2|$ , atau

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

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

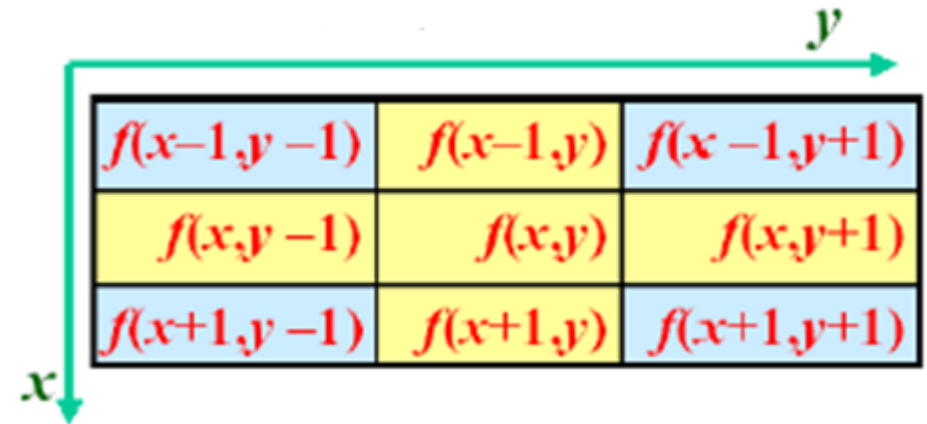
(iv)  $G[f(x,y)] = \max\{|G_x|, |G_y|\}$ .

Persamaan (ii) dan (iv) biasanya lebih disukai karena lebih mudah perhitungannya.

- Operator gradien lainnya adalah operator gradien selisih-terpusat (*center-difference*):

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

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

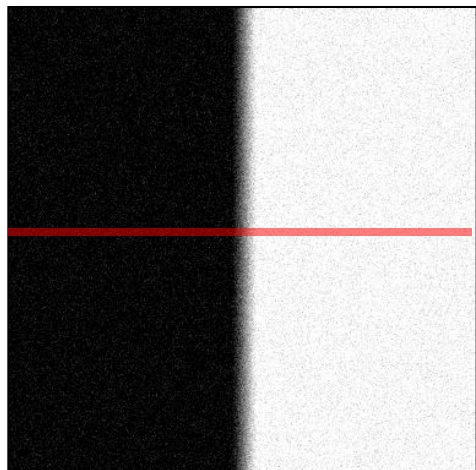


- Ekuivalen dengan *mask*:

$$D_x = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

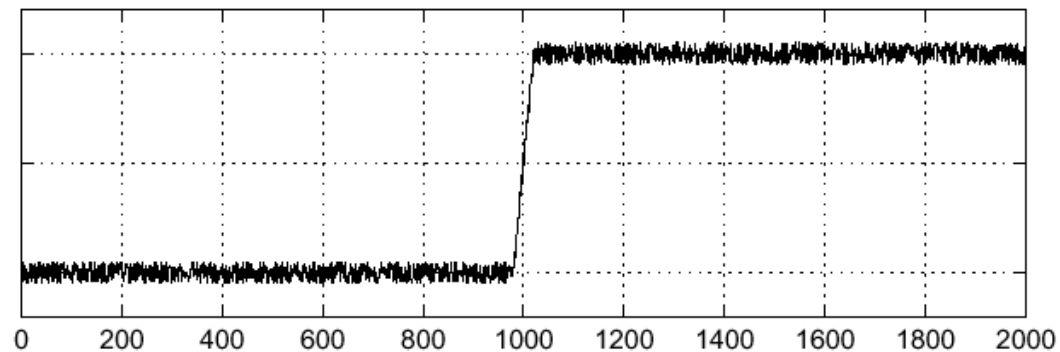
$$D_y = [-1 \quad 0 \quad 1]$$

# Efek derau di dalam citra

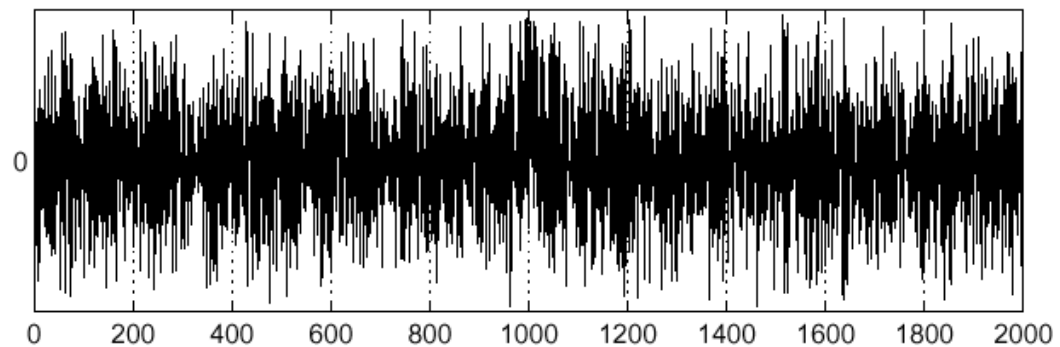


Noisy input image

$$f(x)$$

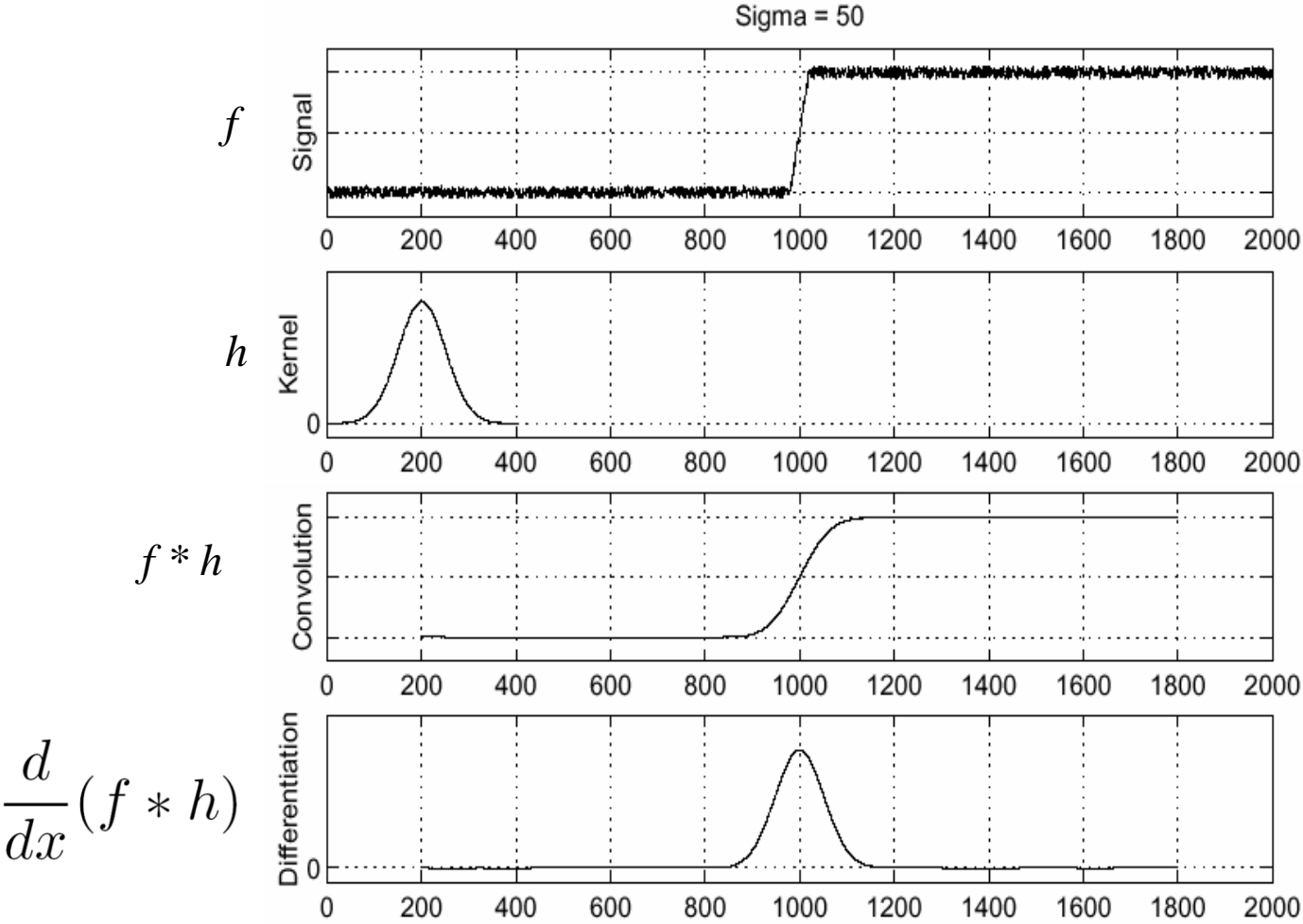


$$\frac{d}{dx} f(x)$$

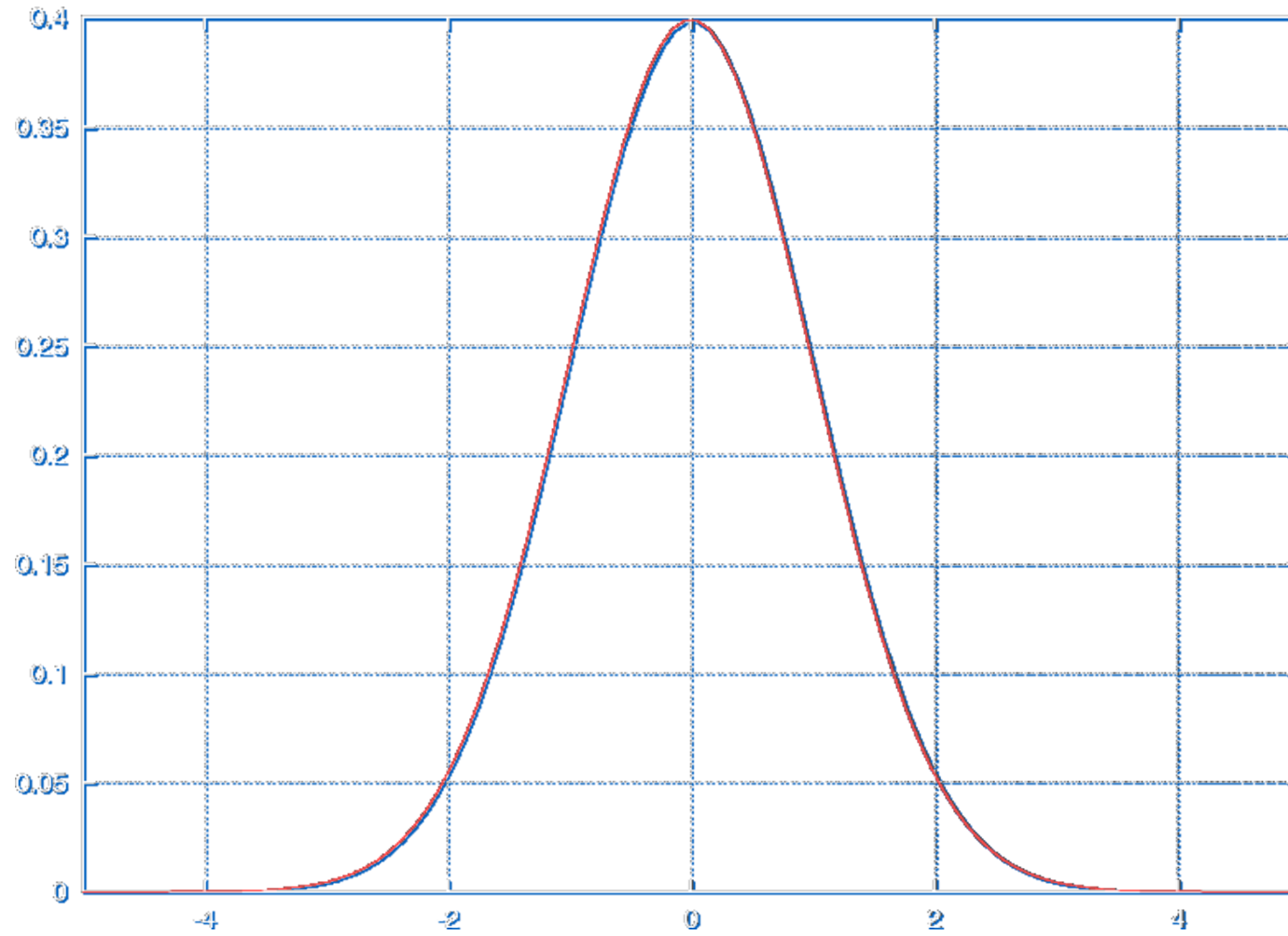


Dimanakah tepi?

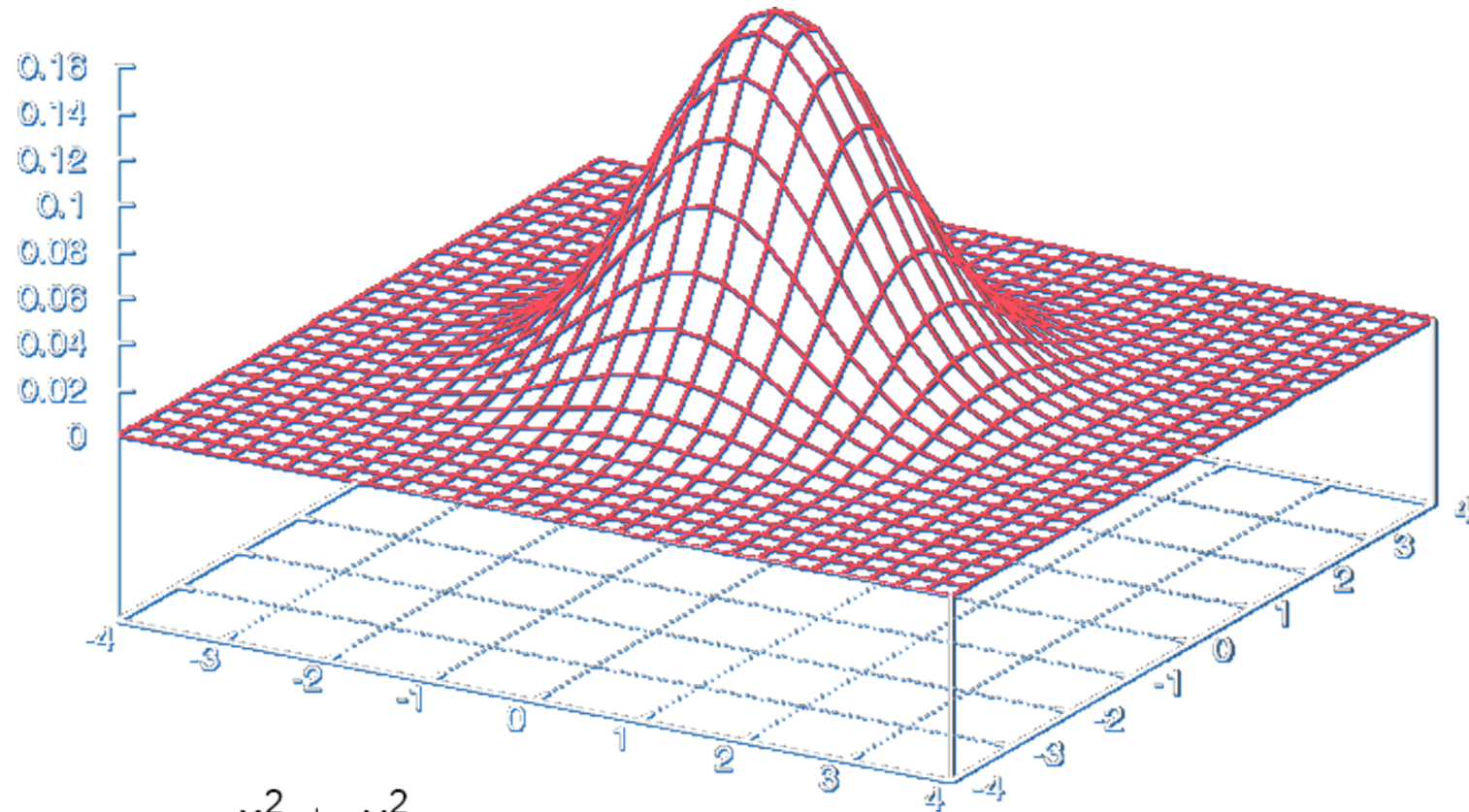
Solusi: lakukan pelembutan (*image smoothing*) terlebih dahulu, misalnya dengan penapis Gaussian



# Fungsi Gaussian (1-D)



## Fungsi Gaussian (2-D)



$$G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$



3 x 3 Gaussian mask

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

7 x 7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

15 x 15 Gaussian mask

2	2	3	4	5	5	6	6	6	5	5	4	3	2	2
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2



```

I = imread('house.jpg');
I_noise = imnoise(I, 'salt & pepper', 0.05);
imshow(I), title ('Original image');
figure, imshow(I_noise), title('Noisy image');

%Deteksi tepi dengan operator Gradien pada citra yang mengandung derau
Gx = [-1; 1];
Gy = [-1 1];
Ix = conv2(double(I_noise), double(Gx), 'same');
Iy = conv2(double(I_noise), double(Gy), 'same');
Jedge = sqrt(Ix.^2 + Iy.^2);
figure, imshow(uint8(Jedge)),
title('Hasil deteksi tepi (before smoothing)');

%Lakukan image smoothing dengan penapis Gaussian 3 x 3
h = [1/16, 1/8, 1/16; 1/8, 1/4, 1/8; 1/16, 1/8, 1/16];
Ifiltered = uint8(convn(double(I_noise), double(h)));
figure, imshow(Ifiltered), title ('Filtered image');

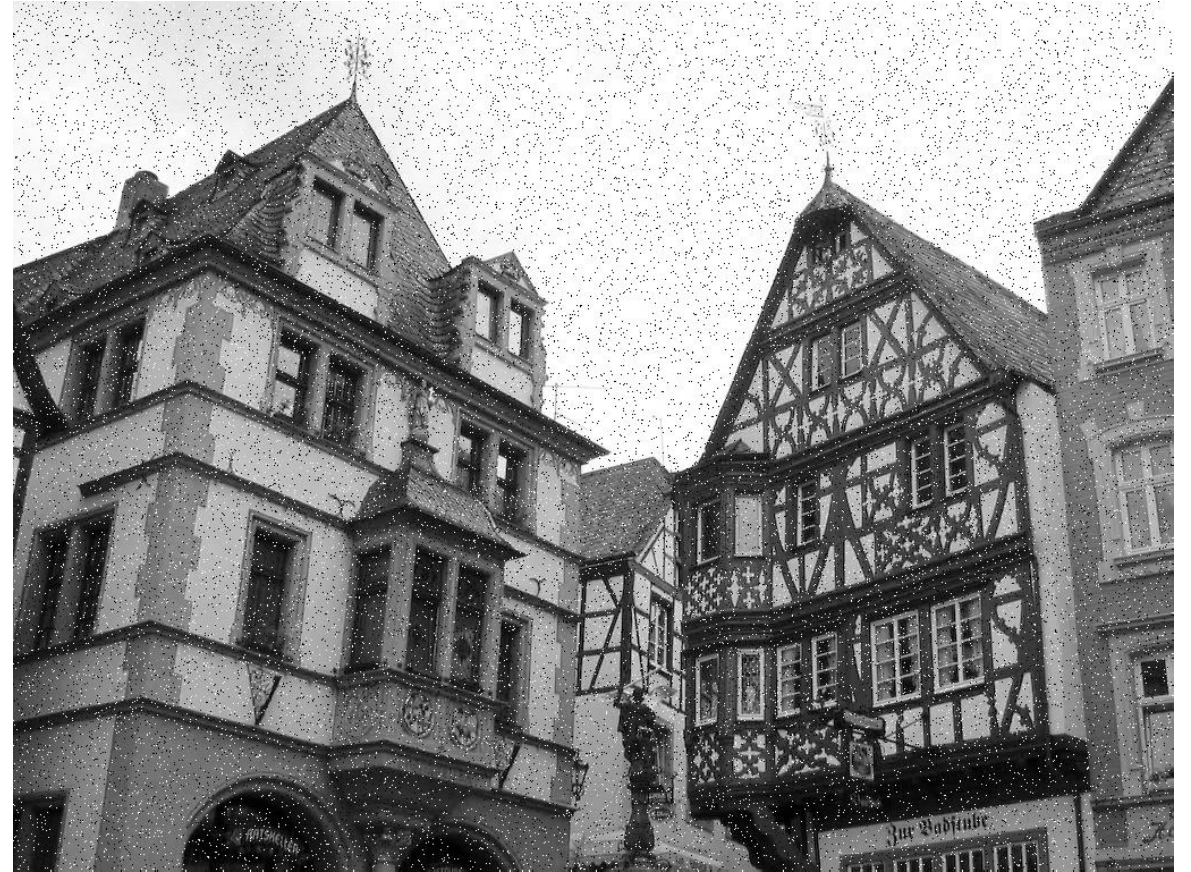
%Deteksi tepi dengan operator Gradien pada citra yang sudah ditapis
Gx = [-1; 1];
Gy = [-1 1];
Ix = conv2(double(Ifiltered), double(Gx), 'same');
Iy = conv2(double(Ifiltered), double(Gy), 'same');
Jedge = sqrt(Ix.^2 + Iy.^2);
figure, imshow(uint8(Jedge)), title('Hasil deteksi tepi (after smoothing)');

```

Original image



Noisy image



Hasil deteksi tepi (before smoothing)



Filtered image

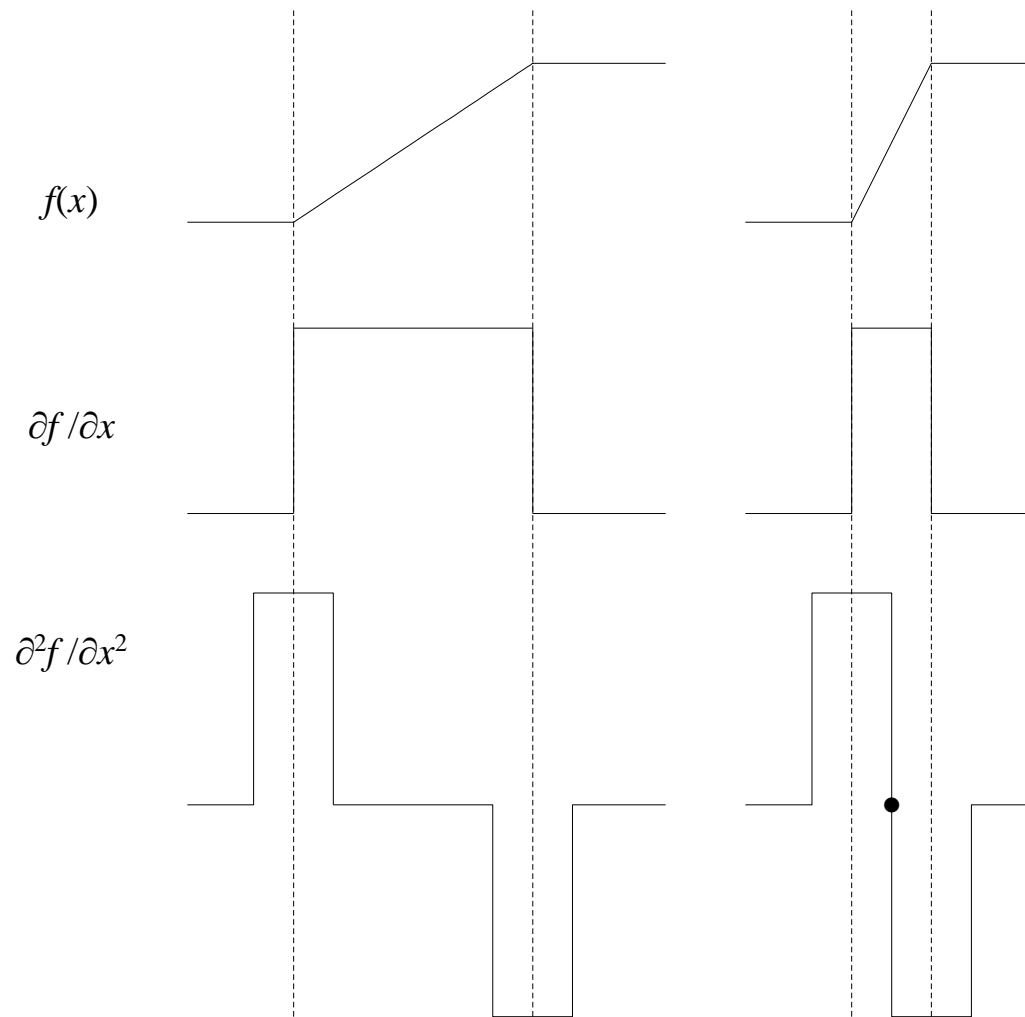


Hasil deteksi tepi (after smoothing)



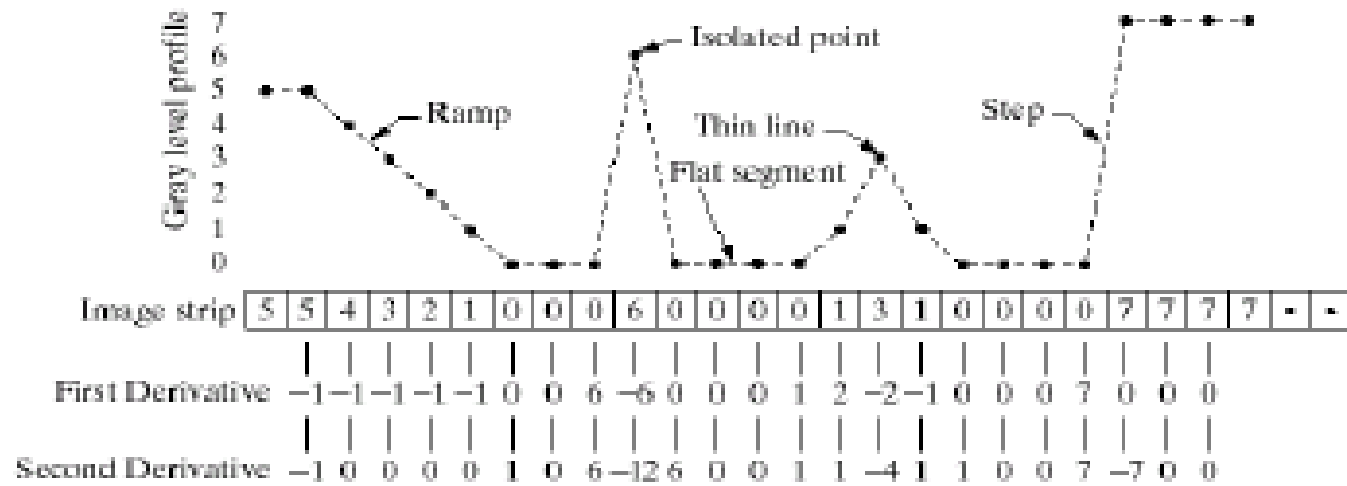
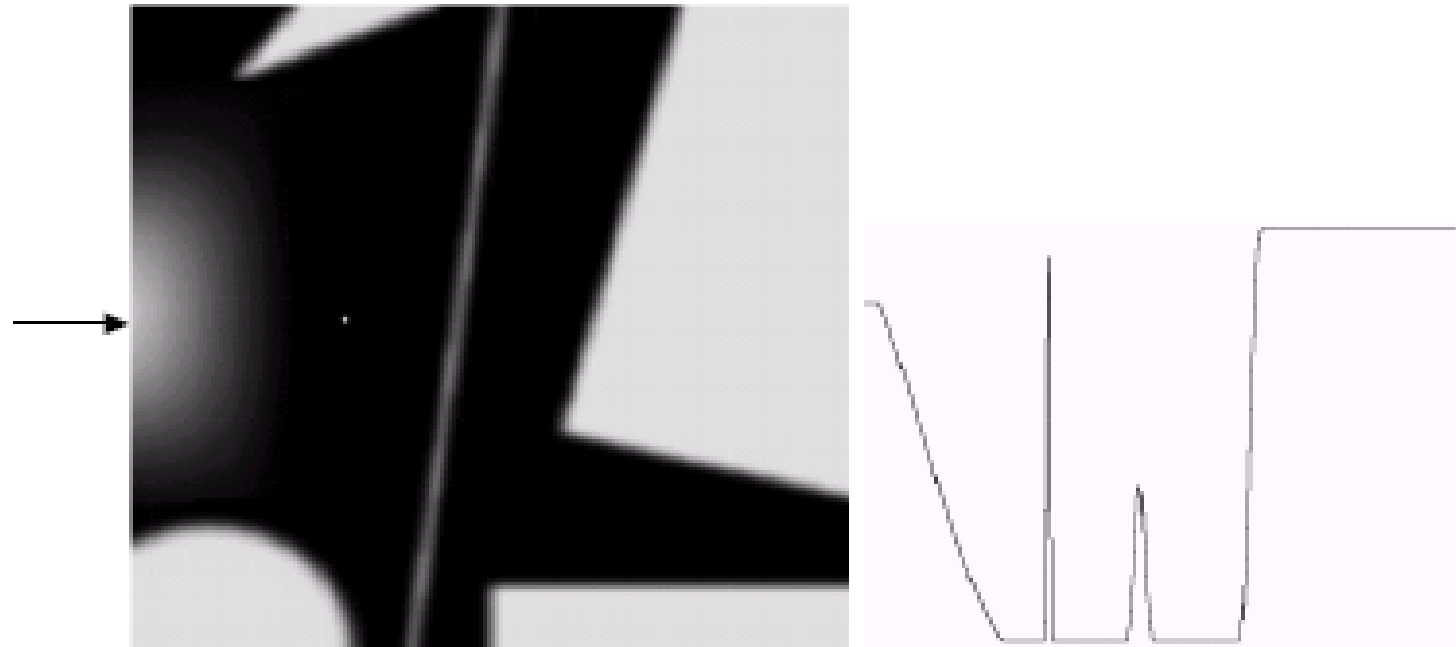
# Operator Turunan Kedua (Laplace)

- Turunan kedua:  $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
- Operator turunan kedua disebut juga **operator Laplace**.
- Operator Laplace mendeteksi lokasi tepi lebih akurat khususnya pada tepi yang curam.
- Pada tepi yang curam, turunan keduanya mempunyai persilangan nol (*zero-crossing*), yaitu titik di mana terdapat pergantian tanda nilai turunan kedua dari positif ke negative atau sebaliknya.



(a) Tepi landai

(b) Tepi curam





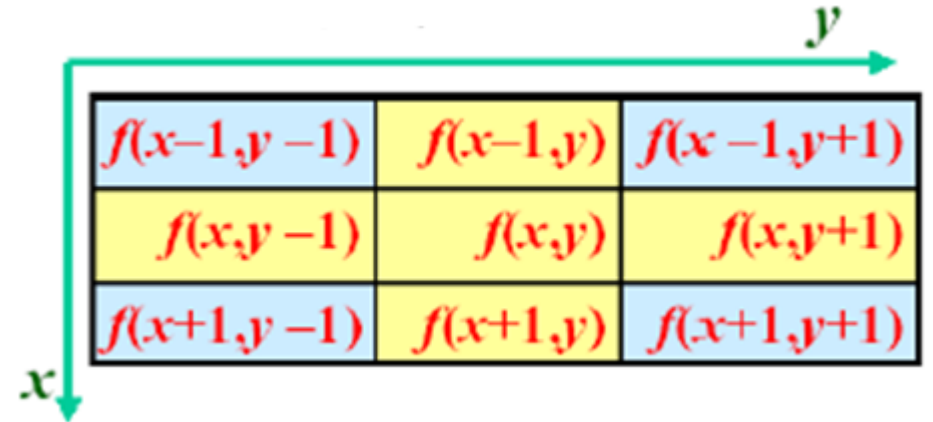
## Penurunan rumus

- Hampiri turunan pertama dengan diferensial mundur (*backward differential*):

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

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

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



- Maka, turunan kedua = turunan pertama diturunkan lagi:

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

$$= G_1(G_3(x)) + G_1(G_3(y))$$



$$\begin{aligned}
&= \frac{1}{\Delta x} G_1(f(x, y)) - G_1(f(x - \Delta x, y)) + \frac{1}{\Delta y} G_1(f(x, y)) - G_1(f(x, y - \Delta y)) \\
&= \frac{1}{\Delta x} \left\{ \frac{f(x + \Delta x, y) - f(x, y) - f(x, y) + f(x - \Delta x, y)}{\Delta x} \right\} \\
&\quad + \frac{1}{\Delta y} \left\{ \frac{f(x, y + \Delta y) - f(x, y) - f(x, y) + f(x, y - \Delta y)}{\Delta y} \right\} \\
&= \frac{f(x + \Delta x, y) - 2f(x, y) + f(x - \Delta x, y)}{(\Delta x)^2} + \frac{f(x, y + \Delta y) - 2f(x, y) + f(x, y - \Delta y)}{(\Delta y)^2}
\end{aligned}$$

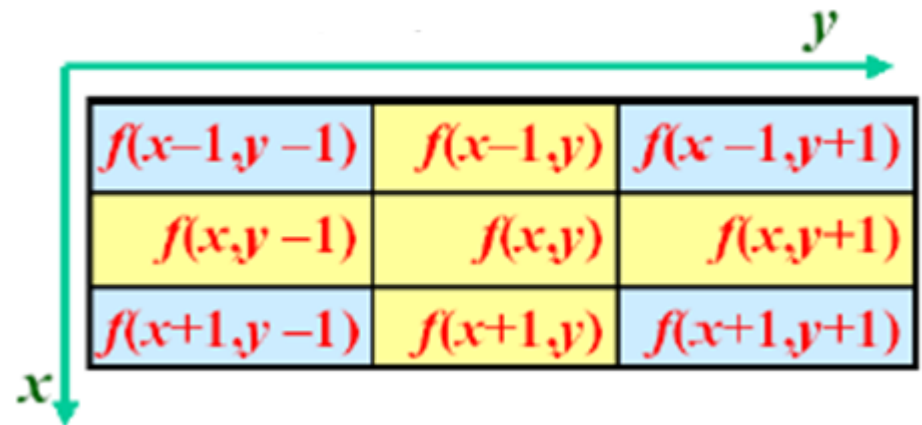
Dengan mengasumsikan  $\Delta x = \Delta y = 1$ , maka diperoleh:

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

$$\begin{aligned}
\nabla^2 f(x, y) &= f(x+1, y) - 2f(x, y) + f(x-1, y) + f(x, y+1) - 2f(x, y) + f(x, y-1) \\
&= f(x, y-1) + f(x-1, y) - 4f(x, y) + f(x+1, y) + f(x, y+1) \\
&= \begin{matrix} 0 & + & f(x-1, y) & + & 0 \\ f(x, y-1) & - & 4f(x, y) & + & f(x, y+1) \\ 0 & + & f(x+1, y) & + & 0 \end{matrix}
\end{aligned}$$

Atau dalam bentuk *mask* konvolusi:

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



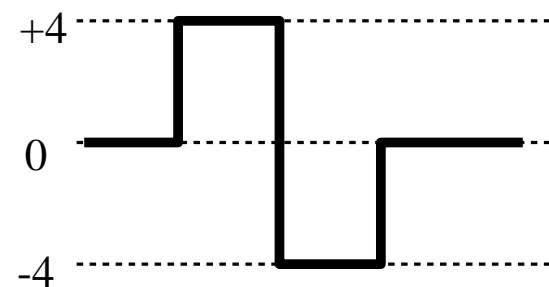
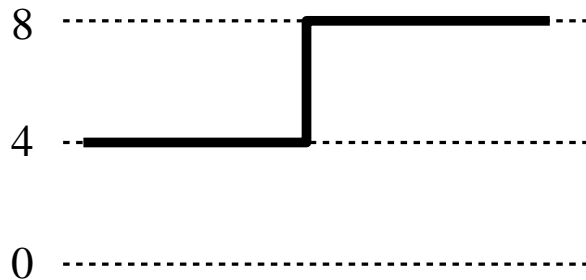
- Contoh berikut memperlihatkan pendeteksian tepi vertikal dengan operator Laplace:

$$\begin{bmatrix} 4 & 4 & 4 & | & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & | & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & | & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & | & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & | & 8 & 8 & 8 & 8 \end{bmatrix}$$

(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



Satu baris dari hasil pendeteksian tepi:  $0 \quad +4 \quad | \quad -4 \quad 0 \quad 0$

**persilangan nol**

- Contoh pendeteksian tepi diagonal (miring) dengan operator Laplace:

$$\begin{bmatrix} 4 & 4 & 8 & 8 & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & 8 & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & 4 & 8 & 8 & 8 & 8 \\ 4 & 4 & 4 & 4 & 4 & 8 & 8 & 8 \\ 4 & 4 & 4 & 4 & 4 & 4 & 8 & 8 \end{bmatrix}$$

(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

- Contoh pendeteksian tepi landai dengan operator Laplace:

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

(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

Satu baris dari hasil pendeteksian tepi: 0 +3 0 -3 0

Pada contoh di atas tidak terdapat persilangan nol; lokasi tepi yang sesungguhnya ditentukan secara interpolasi.

## Operator Laplace lainnya:

0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1
0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

a b  
c d

**FIGURE 3.39**

(a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4).  
(b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

---

- Kadang-kadang diinginkan memberi bobot yang lebih pada *pixel* tengah di antara *pixel* tetangganya:

$$\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

- Operator Laplace termasuk ke dalam penapis lolos-tinggi sebab jumlah seluruh koefisiennya nol dan koefisiennya mengandung nilai negatif maupun positif.

%Deteksi tepi dengan operator Laplace (operator turunan kedua)

```
I = imread('house.jpg');
```

```
figure, imshow(I);
```

```
H = [0 1 0; 1 -4 1; 0 1 0];
```

```
J = uint8(convn(double(I), double(H)));
```

```
figure, imshow(J)
```

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



*f*







Gambar lebih jelas

%Deteksi tepi dengan varian operator Laplace (operator turunan kedua)

```
I = imread('house.jpg');
```

```
figure, imshow(I);
```

```
H = [1 1 1; 1 -8 1; 1 1 1];
```

```
J = uint8(convn(double(I), double(H)));
```

```
figure, imshow(J)
```

1	1	1
1	-8	1
1	1	1



*f*





Gambar lebih jelas



```
I = imread('lada-gray.bmp');  
figure, imshow(I);  
H = [1 1 1; 1 -8 1; 1 1 1];  
J = uint8(convn(double(I), double(H)));  
figure, imshow(J)
```

1	1	1
1	-8	1
1	1	1



```
I = imread('lena.bmp');  
figure, imshow(I);  
H = [0 1 0; 1 -4 1; 0 1 0];  
J = uint8(convn(double(I), double(H)));  
figure, imshow(J)
```

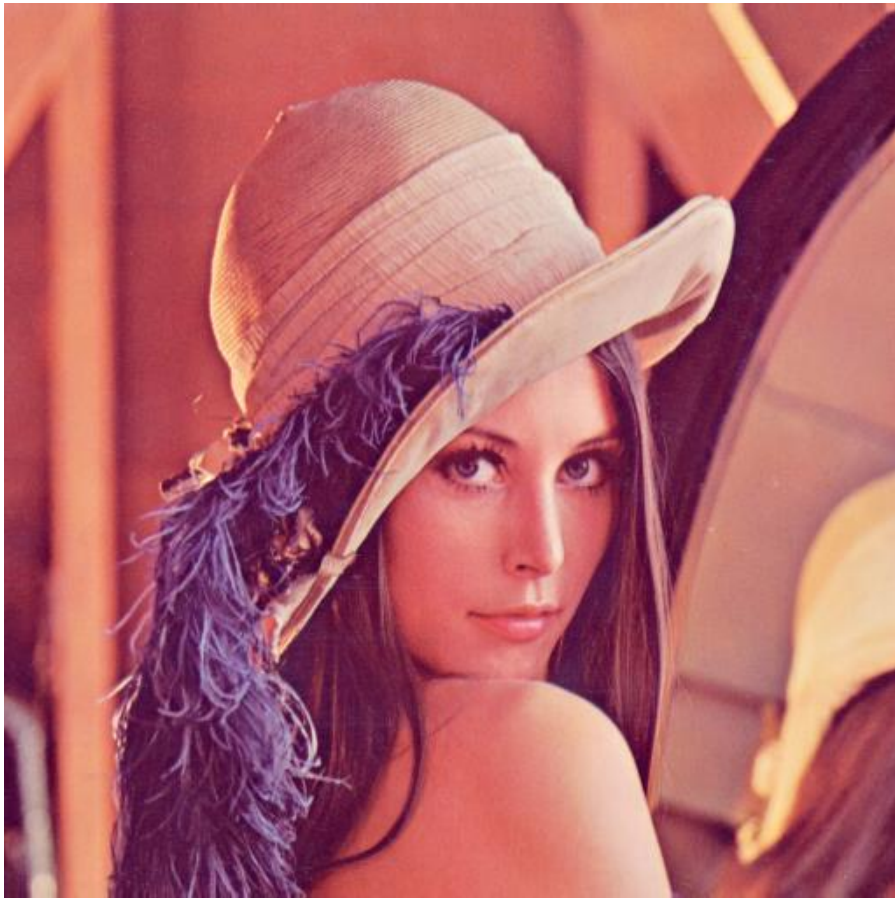
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$





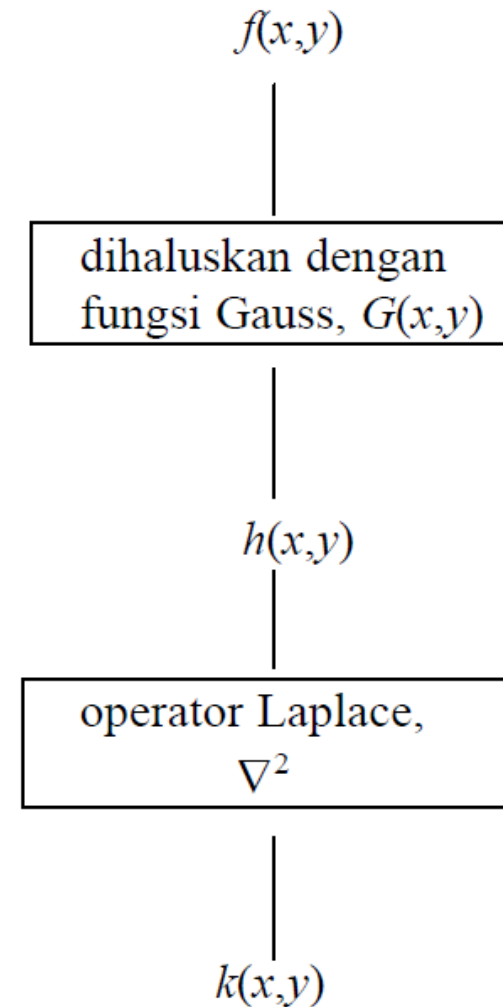
```
I = imread('lena.bmp');  
figure, imshow(I);  
H = [1 1 1; 1 -8 1; 1 1 1];  
J = uint8(convn(double(I), double(H)));  
figure, imshow(J)
```

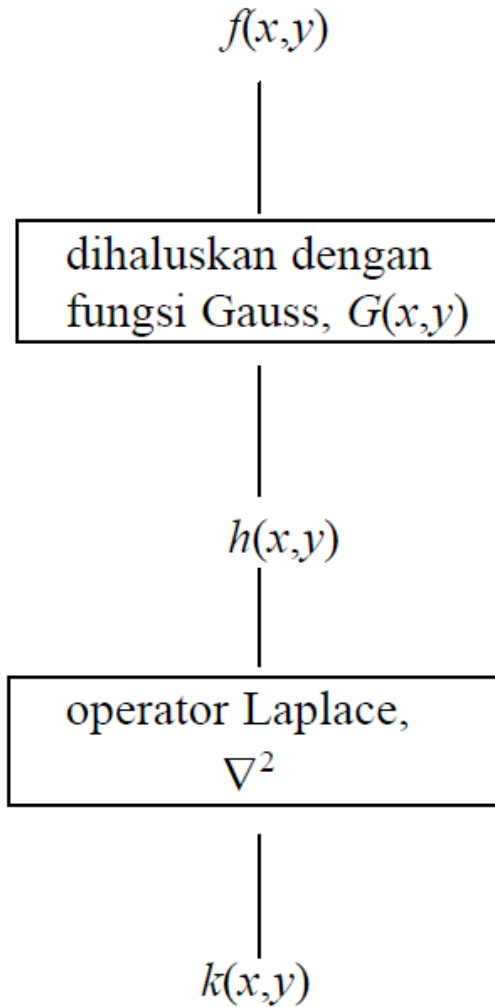
1	1	1
1	-8	1
1	1	1



# Operator Laplace of Gaussian (LoG)

- Kadangkala pendeteksian tepi dengan operator Laplace menghasilkan tepi-tepi palsu yang disebabkan oleh gangguan pada gambar.
- Untuk mengurangi kemunculan tepi palsu, citra ditapis dulu dengan fungsi Gaussian





Berdasarkan gambar di samping:

$$h(x, y) = f(x, y) * G(x, y)$$

$$k(x, y) = \nabla^2 h(x, y)$$

Dapat dibuktikan bahwa

$$\nabla^2 [f(x, y) * G(x, y)] = f(x, y) * \nabla^2 G(x, y)$$

Jadi,

$$k(x, y) = f(x, y) * \nabla^2 G(x, y)$$

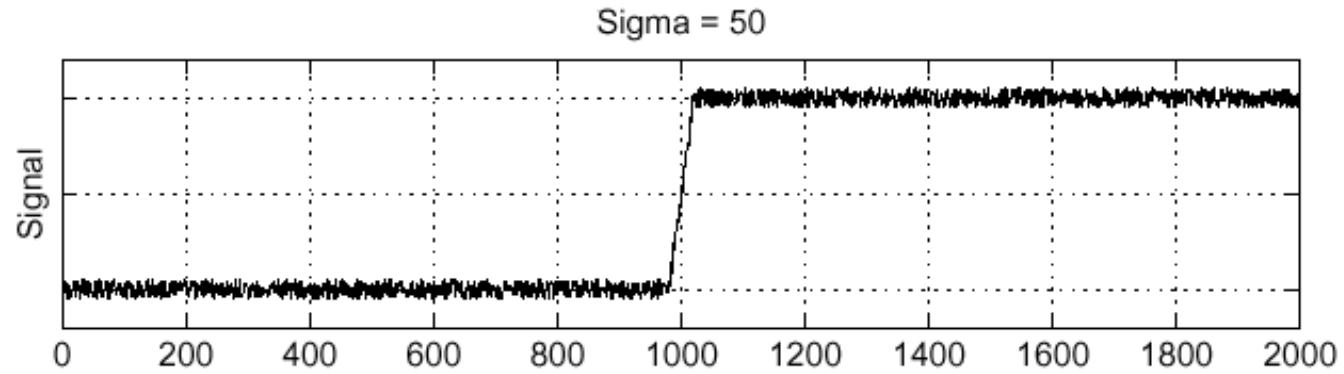
yang dalam hal ini,

$$\nabla^2 G(x, y) = \left( \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{(x^2 + y^2)}{2\sigma^2}}$$

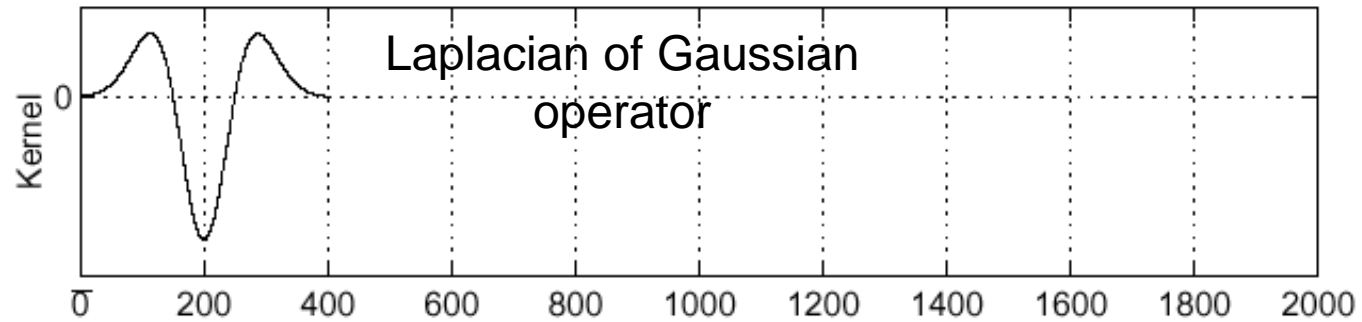


- Tinjau  $\frac{\partial^2}{\partial x^2}(h \star f)$

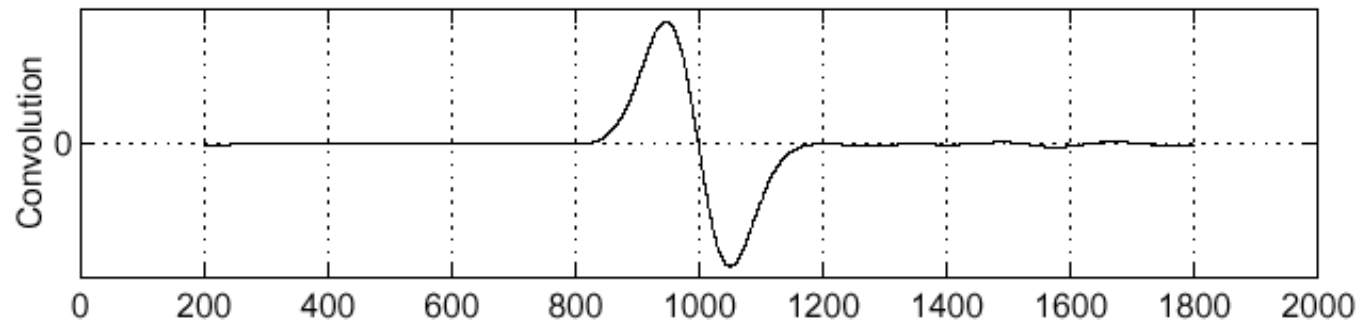
$f$



$\frac{\partial^2}{\partial x^2}h$



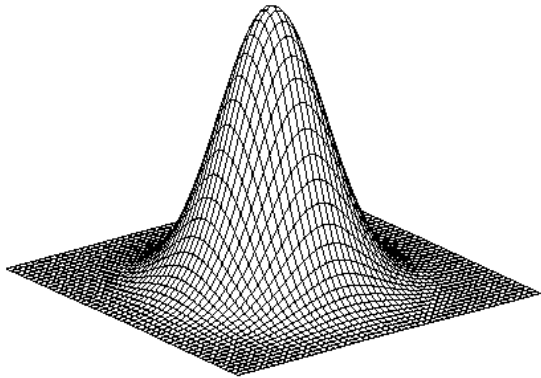
$(\frac{\partial^2}{\partial x^2}h) \star f$



Di mana tepi?

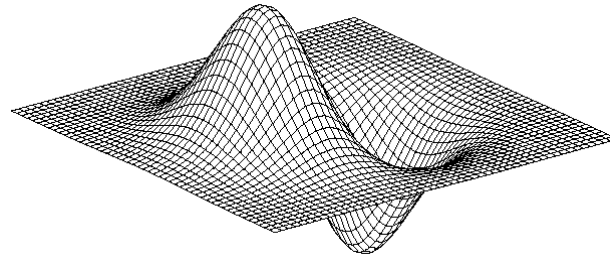
Di tempat *zero-crossing*

- Fungsi  $\nabla^2 G(x,y)$  merupakan turunan kedua dari fungsi Gauss
- Kadang-kadang disebut juga fungsi *Laplacian of Gaussian (LoG)* atau fungsi topi orang Mexico (*Mexican Hat*), karena bentuk kurvanya seperti topi Meksiko.



Gaussian

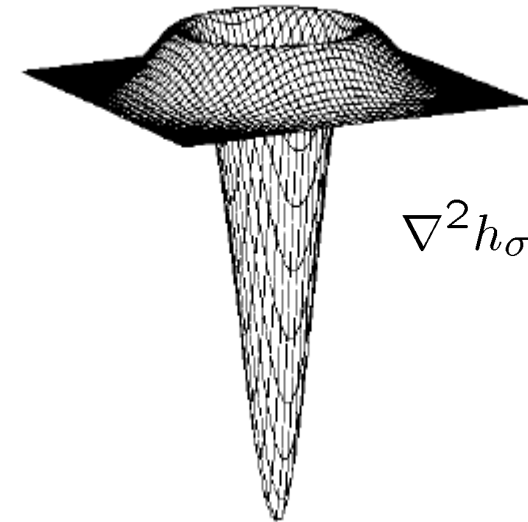
$$h_{\sigma}(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_{\sigma}(u, v)$$

Laplacian of Gaussian



$$\nabla^2 h_{\sigma}(u, v)$$

$\nabla^2$  adalah operator **Laplace**:  $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Jadi, untuk mendeteksi tepi dari citra yang mengalami gangguan, kita dapat melakukan salah satu dari dua operasi ekuivalen di bawah ini:

1. konvolusi citra dengan fungsi Gauss  $G(x,y)$ , kemudian lakukan operasi Laplace terhadap hasilnya, **atau**
2. konvolusi citra langsung dengan penapis  $LoG$ .

Contoh penapis  $LoG$  yang berukuran  $5 \times 5$ :

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

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Mask konvolusi LoG dengan  $\sigma = 1.4$



Citra masukan



Hasil operator Laplace



Hasil operator LoG

```
%Deteksi tepi dengan LoG. Kasus 1: tidak ada derau

I = imread('house.jpg'), title('Original image');

%Deteksi tepi dengan penapis LoG 5 x 5
h = fspecial('log');
J = uint8(convn(double(I), double(h)));
figure, imshow(J), title ('Hasil deteksi tepi dengan LoG');
```

Original image



Hasil deteksi tepi dengan LoG



```
%Deteksi tepi dengan LoG. Kasus 2: Ada derau

I = imread('house.jpg');
I_noise = imnoise(I, 'salt & pepper', 0.02);
imshow(I), title ('Original image');
figure, imshow(I_noise), title('Noisy image');

%Lakukan image smoothing dengan penapis LoG 5 x 5
h = fspecial('log');
Ifiltered = uint8(convn(double(I_noise), double(h)));
figure, imshow(Ifiltered), title ('Hasil deteksi tepi dengan LoG');
```



Noisy image



Hasil deteksi tepi dengan LoG

