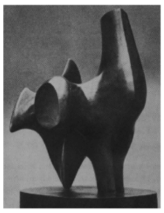
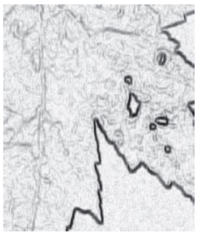
   

DETEKSI TEPI (EDGE DETECTION)

Yeni Herdiyeni
Departemen Ilmu Komputer FMIPA IPB

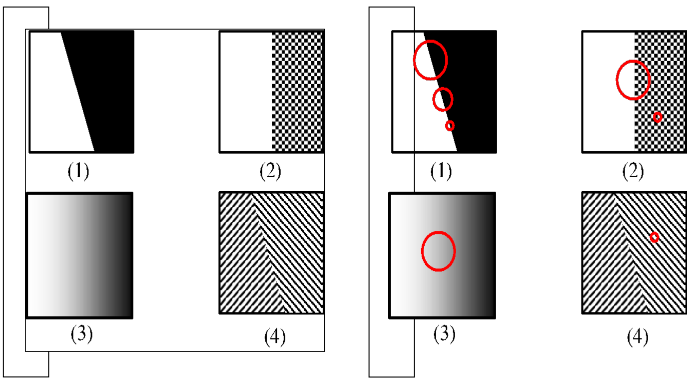
Deteksi Tepi

- Deteksi tepi *(Edge detection)* adalah operasi yang dijalankan untuk mendeteksi garis tepi (*edges*) yang membatasi dua wilayah citra homogen yang memiliki tingkat kecerahan yang berbeda (Pitas 1993).
- Tujuannya adalah untuk mengubah citra 2D menjadi bentuk kurva.
- Edge adalah beberapa bagian dari citra di mana intensitas kecerahan **berubah** secara drastis.

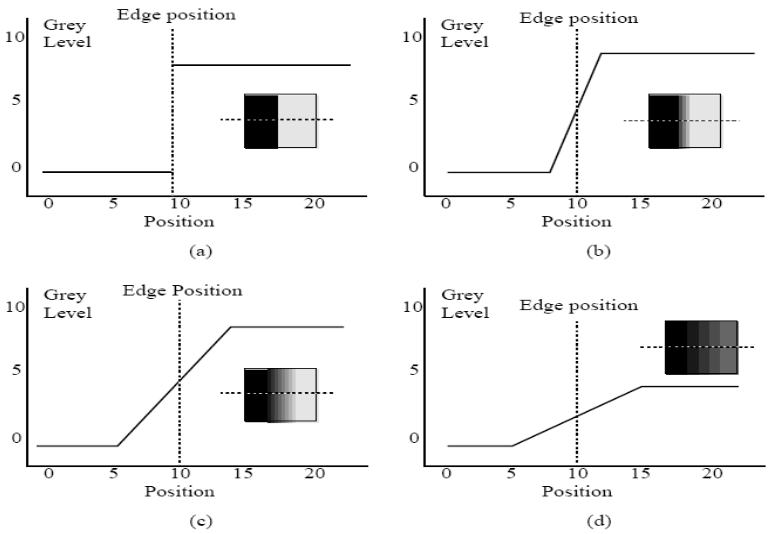
Segmentasi Citra

- Segmentasi citra didasarkan pada dua hal yaitu diskontinuitas (*discontinuity*) dan kemiripan (*similarity*) dari intensitas piksel.
- Pendekatan discontinuity disebut juga dengan pendekatan berbasis edge (*edge-based*). Segmentasi citra menggunakan pendekatan *discontinuity* berdasarkan pada perubahan intensitas warna secara tiba-tiba atau drastis. Pendekatan ini digunakan untuk mendeteksi garis dan edge pada citra.

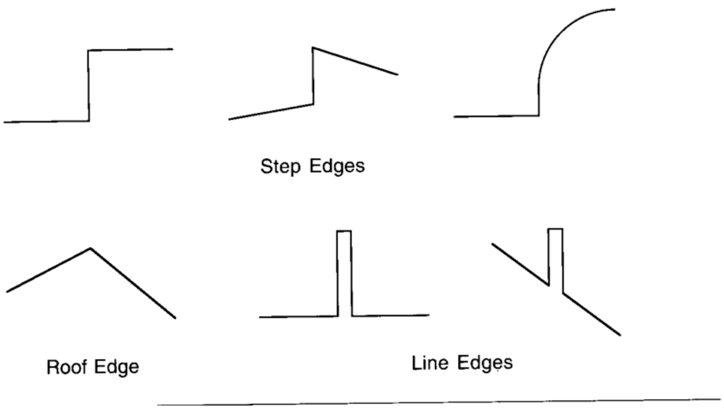
Edge citra



Tipe edge berdasarkan perubahan intensitas keabuan citra



Edge

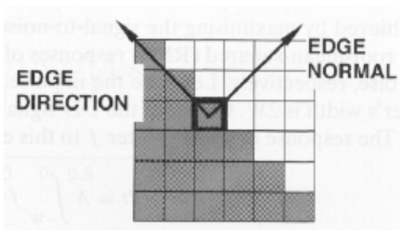


Deteksi Edge

- Edge adalah beberapa bagian dari citra di mana intensitas kecerahan **berubah** secara drastis.
- Dalam objek berdimensi 1, perubahan dapat diukur dengan menggunakan fungsi turunan *(derivative function).*
- Perubahan mencapai maksimum pada saat nilai **turunannya pertamanya mencapai nilai maksimum** atau **nilai turunan kedua (2nd *derivative*) bernilai 0.**

Edge Detection Using the Gradient

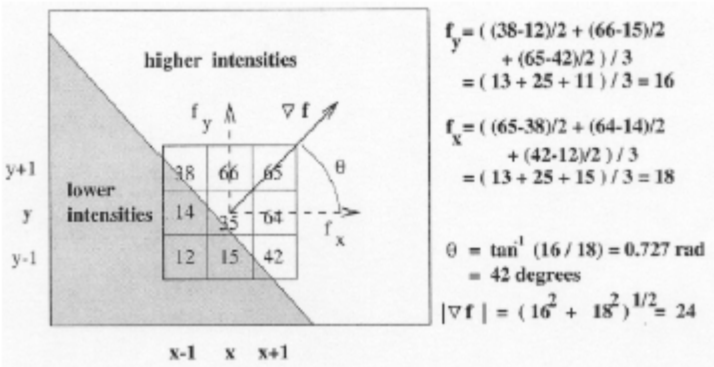
- Properties of the gradient:
  - The magnitude of gradient provides information about the strength of the edge
  - The direction of gradient is always perpendicular to the direction of the edge



- Main idea:
  - Compute derivatives in x and y directions
  - Find gradient magnitude
  - Threshold gradient magnitude

8

Edge Detection Using the Gradient



(an example using the Prewitt edge detector - don't divide by 2)

9

Edge Detection Using the Gradient

- Estimating the gradient with finite differences

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

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

  - Approximation by *finite differences*:

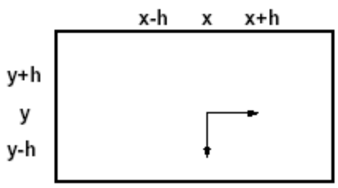
$$\frac{\partial f}{\partial x} = \frac{f(x+h_x, y) - f(x, y)}{h_y} = f(x+1, y) - f(x, y), \ (h_x = 1)$$

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

10

Edge Detection Using the Gradient

- Using pixel-coordinate notation (remember: *j* corresponds to the *x* direction and *i* to the negative *y* direction):



$$\frac{\partial f}{\partial x} = f(i, j+1) - f(i, j)$$

$$\frac{\partial f}{\partial y} = f(i-1, j) - f(i, j) \text{ or } \frac{\partial f}{\partial y} = f(i, j) - f(i+1, j)$$

Edge Detection Using the Gradient

- Example:
  - Suppose we want to approximate the gradient magnitude at z_5

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

$$\frac{\partial I}{\partial x} = z_6 - z_5, \quad \frac{\partial I}{\partial y} = z_5 - z_8$$

$$magn(\nabla I) = \sqrt{(z_6 - z_5)^2 + (z_5 - z_8)^2}$$

- We can implement $\partial I/\partial x$ and $\partial I/\partial y$ using the following masks:

-1	1

1
-1

Note: M_x is the approximation at $(i, j + 1/2)$ and M_y is the approximation at $(i + 1/2, j)$

Edge Detection Using the Gradient

- The Roberts edge detector

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

- This approximation can be implemented by the following masks:

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad M_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Note: M_x and M_y are approximations at (i + 1/2, j + 1/2)

13

Edge Detection Using the Gradient

- The Prewitt edge detector
  - Consider the arrangement of pixels about the pixel (i, j):

$$\begin{matrix} a_0 & a_1 & a_2 \\ a_7 & [i,j] & a_3 \\ a_6 & a_5 & a_4 \end{matrix}$$

  - The partial derivatives can be computed by:

$$M_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$
$$M_y = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2)$$

  - The constant c implies the emphasis given to pixels closer to the center of the mask.
  - Setting c = 1, we get the *Prewitt operator*:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Note: M_x and M_y are approximations at (i, j))

14

Edge Detection Using the Gradient

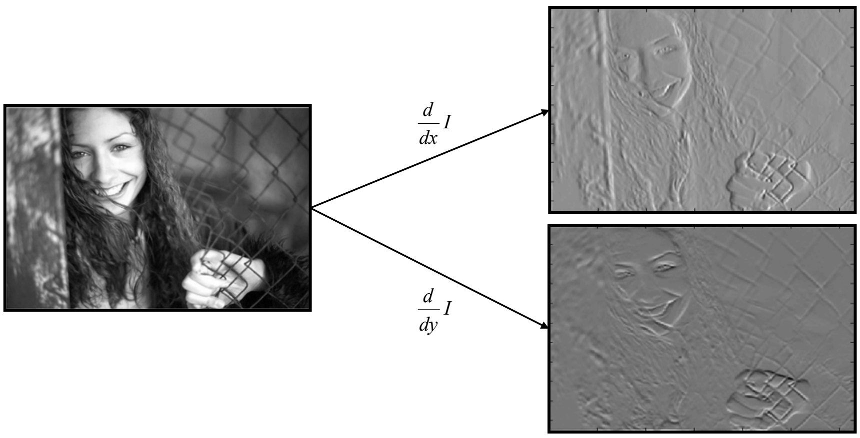
- The Sobel edge detector
  - Setting c = 2, we get the *Sobel operator*:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Note: M_x and M_y are approximations at (i, j))

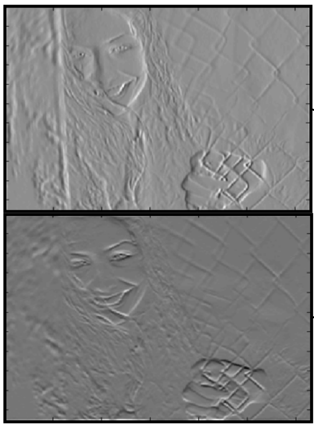
Edge Detection Using the Gradient

- Example:



Edge Detection Using the Gradient

- Example – cont.:



$$\Delta = \sqrt{\left(\frac{d}{dx} I\right)^2 + \left(\frac{d}{dy} I\right)^2}$$

17

Edge Detection Using the Gradient

- Example – cont.:

$$\Delta \geq Threshold = 100$$

18

Edge Detection Using the Gradient

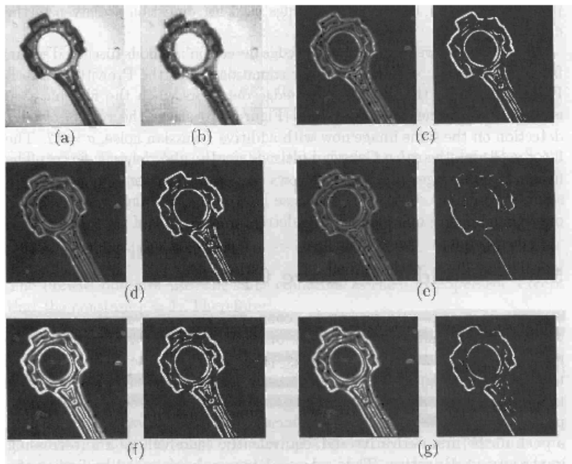


Figure 5.4: A comparison of various edge detectors. (a) Original image. (b) Filtered image. (c) Simple gradient using 1×2 and 2×1 masks, $T = 32$. (d) Gradient using 2×2 masks, $T = 64$. (e) Roberts cross operator, $T = 64$. (f) Sobel operator, $T = 225$. (g) Prewitt operator, $T = 225$.

(with noise filtering)

19

Edge Detection Using the Gradient

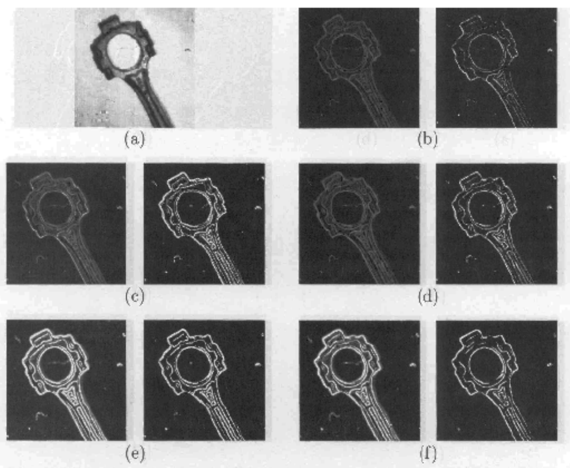


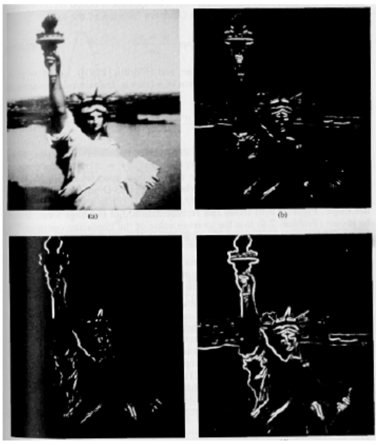
Figure 5.5: A comparison of various edge detectors without filtering. (a) Original image. (b) Simple gradient using 1×2 and 2×1 masks, $T = 64$. (c) Gradient using 2×2 masks, $T = 64$. (d) Roberts cross operator, $T = 64$. (e) Sobel operator, $T = 225$. (f) Prewitt operator, $T = 225$.

(without noise filtering)

20

Edge Detection Using the Gradient

- Isotropic property of gradient magnitude:
  - The magnitude of gradient is an *isotropic* operator (it detects edges in any direction !!)

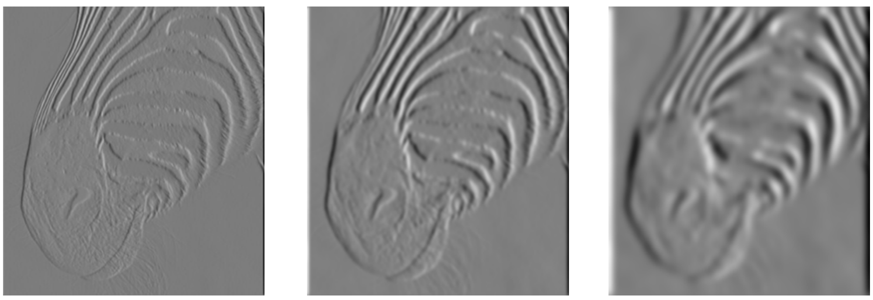


21

Edge Detection

- Practical issues:
  - Differential masks act as high-pass filters – tend to amplify noise.
  - Reduce the effects of noise - first smooth with a low-pass filter.

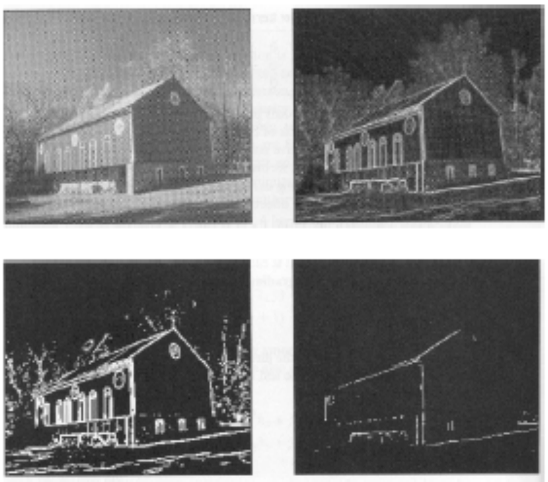
1) <u>The noise suppression-localization tradeoff</u>
  - a larger filter reduces noise, but worsens localization (i.e., it adds uncertainty to the location of the edge) and vice-versa.



22

Edge Detection

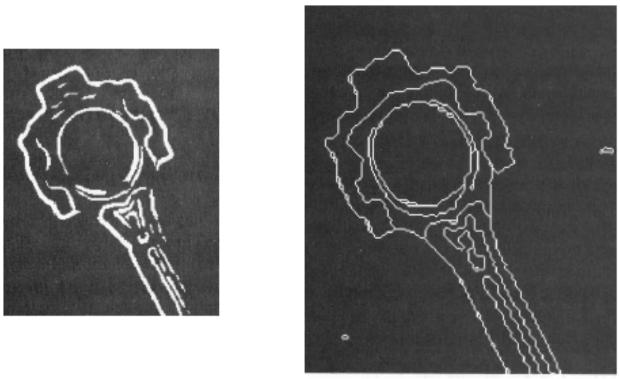
2) How should we choose the threshold?



Edge Detection

3) Edge thinning and linking

- required to obtain good contours



Edge Detection

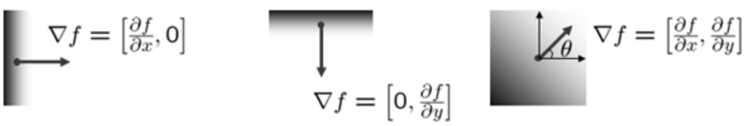
- Criteria for optimal edge detection:
  - <u>Good detection</u>: the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - <u>Good localization</u>: the edges detected must be as close as possible to the true edges.
  - <u>Single response constraint</u>: the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge

25

Image Gradient

- Perubahan intensitas kecerahan dapat dihitung dengan menggunakan gradient citra (*image gradient*).

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$



Gambar 4 Prubahan intensitas kecerahan piksel

Edge Detection Using the Gradient

- Definition of the gradient:

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

$$magn(\nabla f) = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} = \sqrt{M_x{}^2 + M_y{}^2}$$

$$dir(\nabla f) = \tan^{-1}(M_y/M_x)$$

- To save computations, the magnitude of gradient is usually approximated by:

$$magn(\nabla f) \approx |M_x| + |M_y|$$

27

Gradient Magnitude

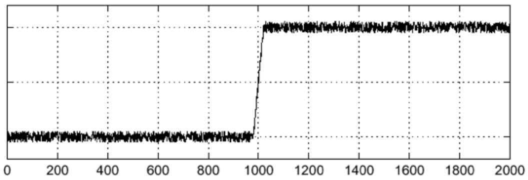
- ***gradient magnitude***. Gradient Magnitude dapat dihitung dengan cara:

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

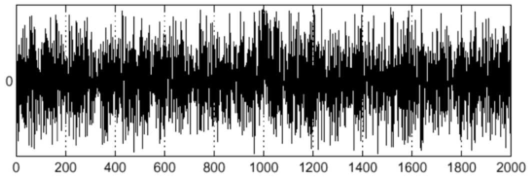
Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

$f(x)$



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



Where is the edge?

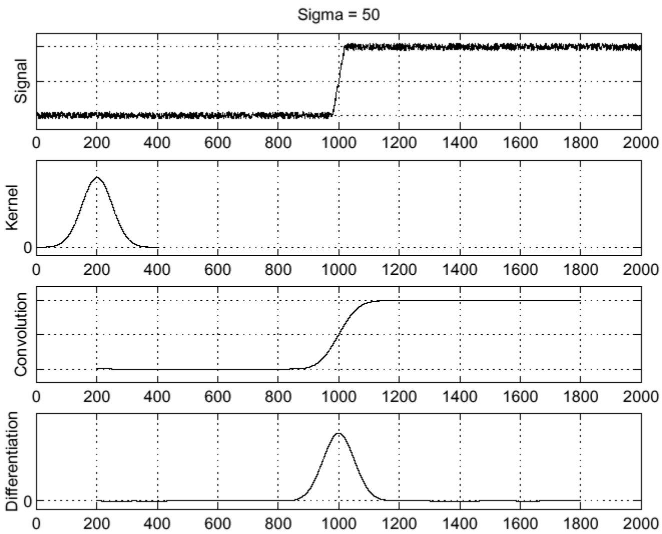
Solution: smooth first

f

h

$h \star f$

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

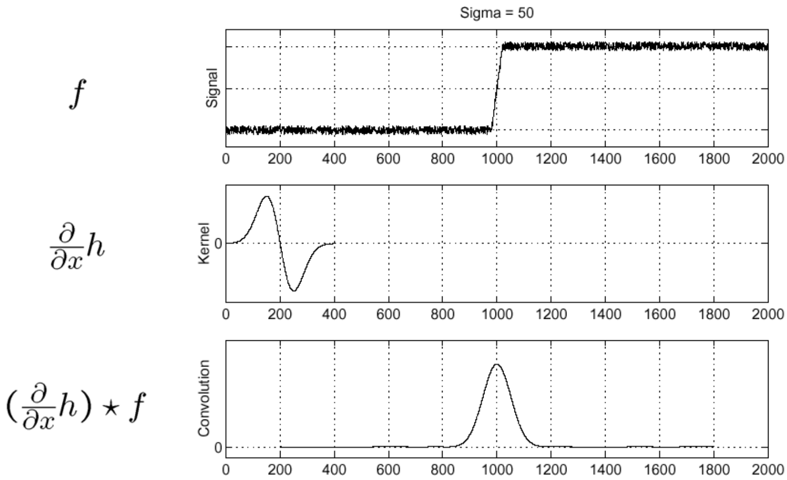


Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

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

- This saves us one operation:



Algoritme Deteksi Tepi

- *Robert Operator*
- *Sobel Operator*
- *Prewitt Operator*
- *Canny Operator*
- *Laplacian operator*
- dan lain-lain.

Operator Robert

- Robert Operator menggunakan operator gradient berukuran 2 x 2 :

1	1
−1	−1

- Gradient magnitude

$$G[f(i,j)] = [f(i,j) - f(i+1,j+1)] + [f(i+1,j) - f(i,j+1)]$$

- Karena operator Robert hanya menggunakan convolution mask berukuran 2 x 2, maka operator Robert sangat sensitive terhadap noise.

Operator Sobel #1

- Operator Gradient 3 x 3

1	2	1
0	0	0
−1	−2	−1

(a)

1	0	−1
2	0	−2
1	0	−1

(b)

- Operator Sobel melakukan deteksi tepi dengan memperhatikan tepi vertical dan horizontal.

Operator Sobel #2

- Gradient Magnitude

$$G_x = [f(i-1,j-1) + 2f(i-1,j) + f(i-1,j+1)] - [f(i+1,j-1) + 2f(i+1,j) + f(i+1,j+1)]$$

$$G_y = [f(i-1,j-1) + 2f(i,j-1) + f(i+1,j-1)] - [f(i-1,j+1) + 2f(i,j+1) + f(i+1,j+1)]$$

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

Operator Prewitt

- Operator Prewitt menggunakan 8 (delapan) buah kernel operator gradient

1	1	1
0	0	0
−1	−1	−1

(a)

0	1	1
−1	0	1
−1	−1	0

(b)

−1	0	1
−1	0	1
−1	0	1

(c)

−1	−1	0
−1	0	1
0	1	1

(d)

Implementasi Matlab

- **Deteksi tepi dengan operator Prewitt**

```
citra=imread('cameraman.tif');
 ic = citra (:,:,1);

px=[-1 0 1;-1 0 1;-1 0 1]; %% Deteksi Vertikal
icx=filter2(px,ic); % convolution
figure,imshow(icx/255);

py=px'; %% Deteksi Horizontal
icy=filter2(py,ic);
figure,imshow(icy/255);

edge_p=sqrt(icx.^2+icy.^2);
figure,imshow(edge_p/255);

edge_t=im2bw(edge_p/255,0.3);
figure, imshow(edge_t);
```

Operator Canny

- Operator Canny menggunakan operator detector Gaussian.
- Ada beberapa kriteria yang harus dipenuhi pada operator Canny yaitu:
  1. ***Good Detection*.** Operator gradient melakukan respon terhadap edge bukan noise
  2. ***Good Localization***. Titik-titik tepi yang ditemukan haruslah terlokalisasi dengan benar. Dengan kata lain, jarak antara titik tepi yang ditemukan dengan tepian yang sebenarnya haruslah minimum.
  3. ***Single Response***. Hanya memiliki satu respon terhadap sebuah garis tepi tunggal. Pendeteksi tepi harus dapat menghilangkan kemungkinan sejumlah respon terhadap sebuah garis tepi tunggal (Green 2002).

Operator Laplacian

- Deteksi tepi dengan menggunakan Canny terdiri dari 4 (empat) tahap yaitu:
  1. Membangkitkan operator gradient (mask/filter)
  2. Melakukan konvolusi
  3. Non Maxima Suppression.
  4. Pengujian threshold

The Canny edge detector

- original image (Lena)

The Canny edge detector

norm of the gradient

The Canny edge detector

thresholding

The Canny edge detector

thinning
(non-maximum suppression)

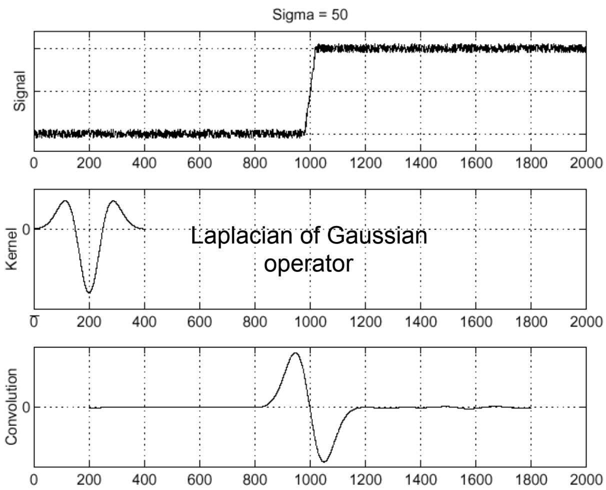
Laplacian of Gaussian

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

f

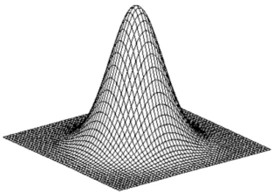
$\frac{\partial^2}{\partial x^2}h$

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



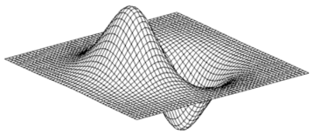
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters



Gaussian

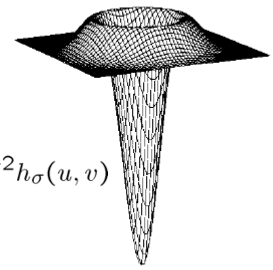
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



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

∇^2 is the **Laplacian** operator:

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

1. **Laplacian-type filters**:

- ✓ Are rotationally invariant, that is they enhance the details in all directions equally
- ✓ Example convolution masks of Laplacian-type filters are:

Filter 1

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

Filter 2

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

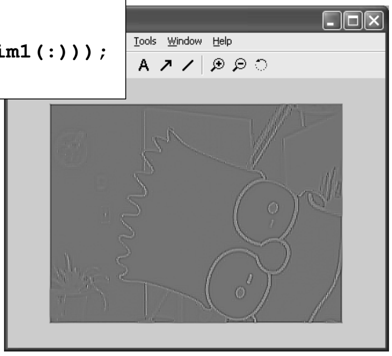
Filter 3

$$\begin{bmatrix} -2 & 1 & -2 \\ 1 & 5 & 1 \\ -2 & 1 & -2 \end{bmatrix}$$

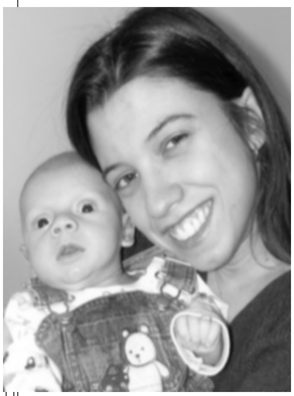
Spatial Filtering

Effect of the Laplacian… (MATLAB demo)

```
im = im/max(im(:));

% create laplacian
L=[1 1 1;1 -8 1; 1 1 1];

% Filter !
im1=conv2(im,L);

% normalize and show
im1=(im1-min(im1(:))) / (max(im1(:))-min(im1(:)));
imshow(im1);
```



Laplacian filter



Original image

Laplacian filtered image

Contrast enhanced version of Laplacian filtered image

Tugas

Buat program segmentasi untuk memisahkan objek jeruk dan daun