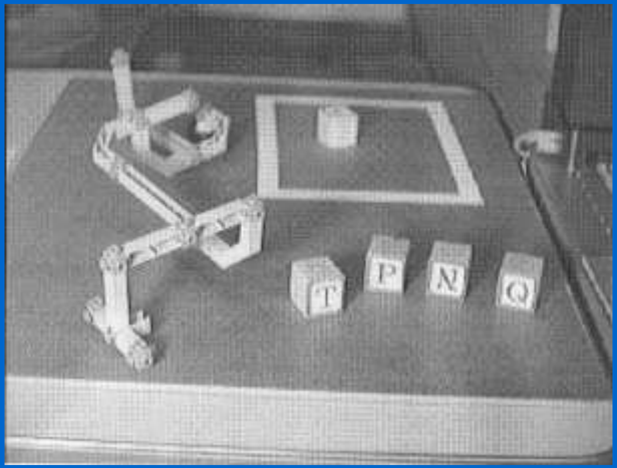
Edge detection is part of segmentation

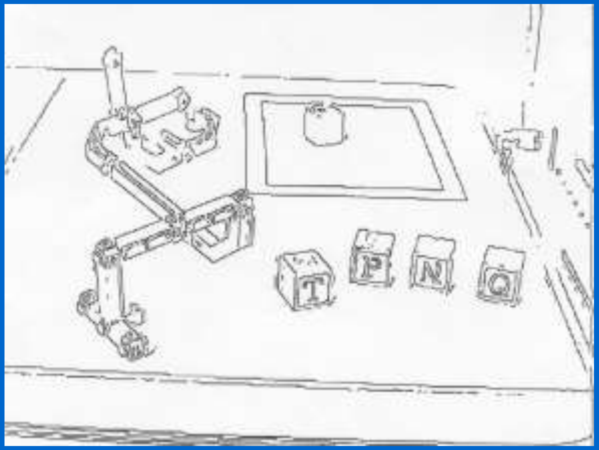
image | human segmentation | gradient magnitude

- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Goal of Edge Detection

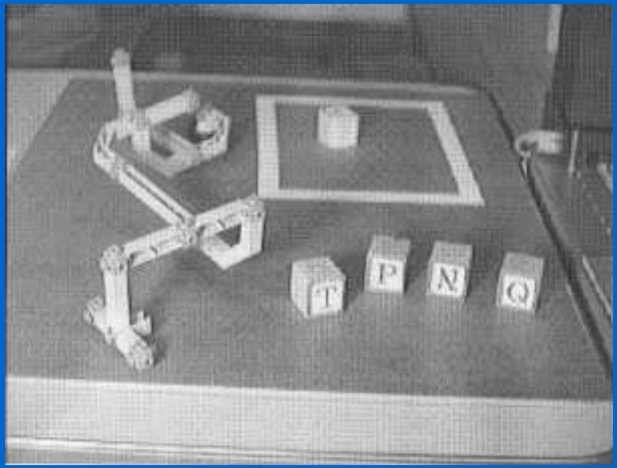
- Produce a line “drawing” of a scene from an image of that scene.

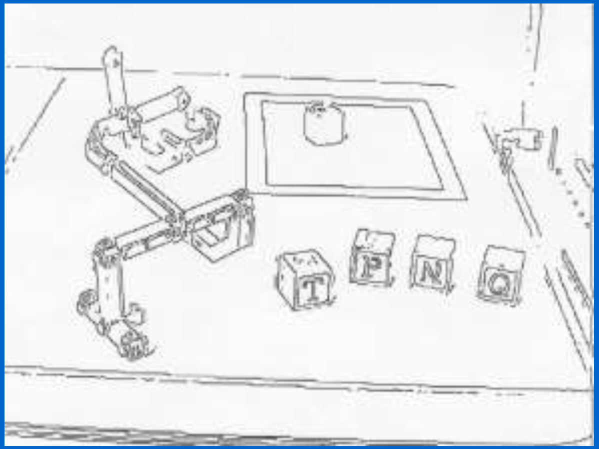




Why is Edge Detection Useful?

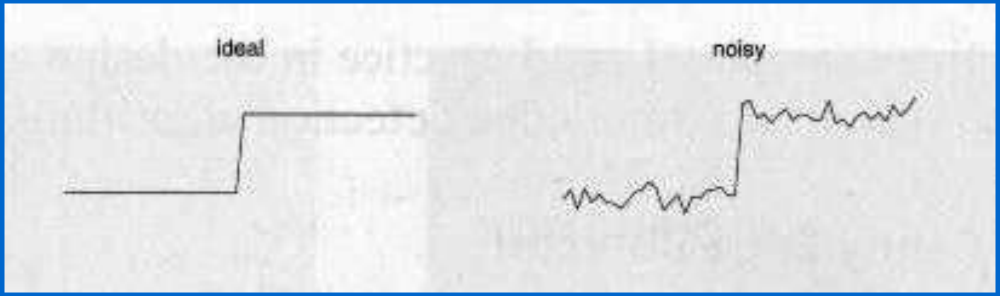
- Important features can be extracted from the edges of an image (e.g., corners, lines, curves).
- These features are used by higher-level computer vision algorithms (e.g., recognition).

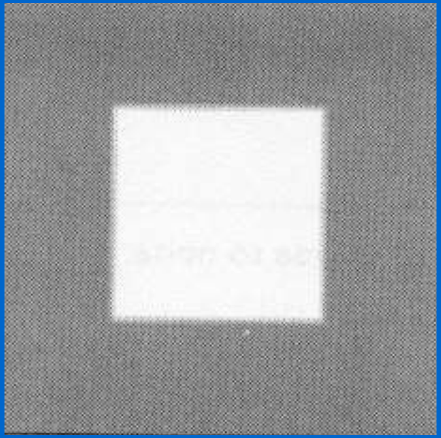




Modeling Intensity Changes

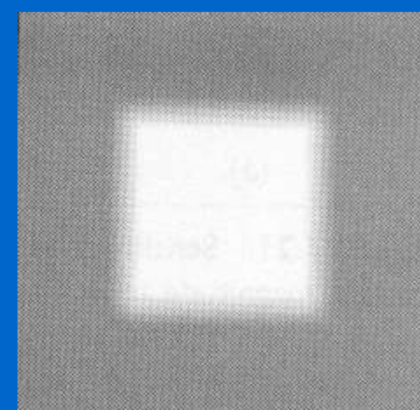
- **Step edge:** the image intensity abruptly changes from one value on one side of the discontinuity to a different value on the opposite side.





Modeling Intensity Changes (cont'd)

- **Ramp edge:** a step edge where the intensity change is not instantaneous but occur over a finite distance.

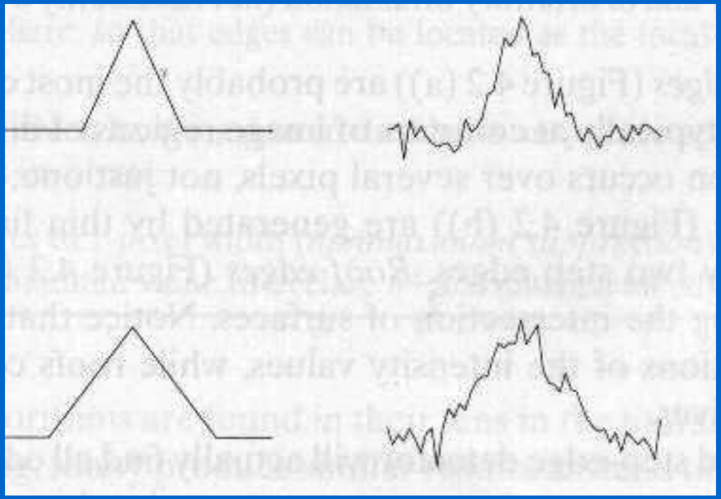


Modeling Intensity Changes (cont'd)

- **Ridge edge:** the image intensity abruptly changes value but then returns to the starting value within some short distance (i.e., usually generated by lines).

Modeling Intensity Changes (cont'd)

- **Roof edge: a** ridge edge where the intensity change is not instantaneous but occur over a finite distance (i.e., usually generated by the intersection of two surfaces).



Edge Detection Using Derivatives

- Often, points that lie on an edge are detected by:

(1) Detecting the local **maxima** or **minima** of the first derivative.

(2) Detecting the **zero-crossings** of the second derivative.

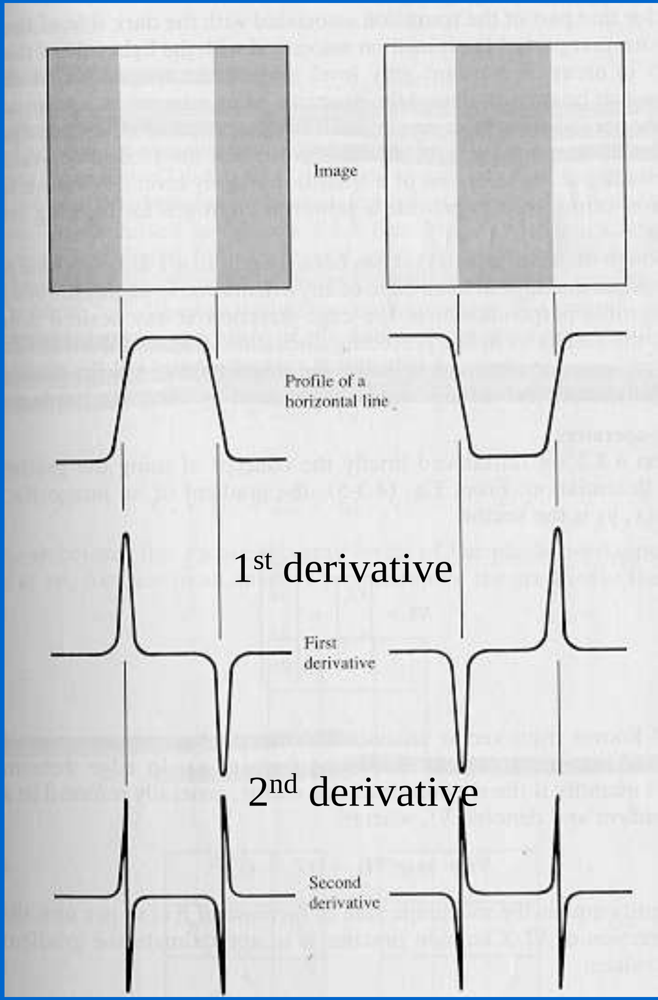


Image Derivatives

- How can we differentiate a *digital* image?
  - **Option 1:** reconstruct a continuous image, *f(x,y)*, then compute the derivative.
  - **Option 2:** take discrete derivative (i.e., finite differences)

Consider this case first!

Edge Detection Using First Derivative

1D functions

(not centered at x)

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x) \quad (h=1)$$

mask: $[-1 \quad 1]$

$$\text{mask M} = [-1, 0, 1]$$

(centered at x)

(upward) step edge

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	0	0	0	0	12	12	0	0	0	0

(downward) step edge

S_2			24	24	24	24	24	12	12	12	12	12
S_2	$\otimes$	M	0	0	0	0	-12	-12	0	0	0	0

Edge Detection Using Second Derivative

- Approximate finding maxima/minima of gradient magnitude by finding places where:

$$\frac{df^2}{dx^2}(x) = 0$$

- Can’t always find discrete pixels where the second derivative is zero – look for zero-crossing instead.

Edge Detection Using Second Derivative (cont'd)

1D functions:

$$f''(x) = \lim_{h \to 0} \frac{f'(x+h) - f'(x)}{h} \approx f'(x+1) - f'(x) =$$

$$f(x+2) - 2f(x+1) + f(x) \quad (h=1)$$

(centered at x+1)

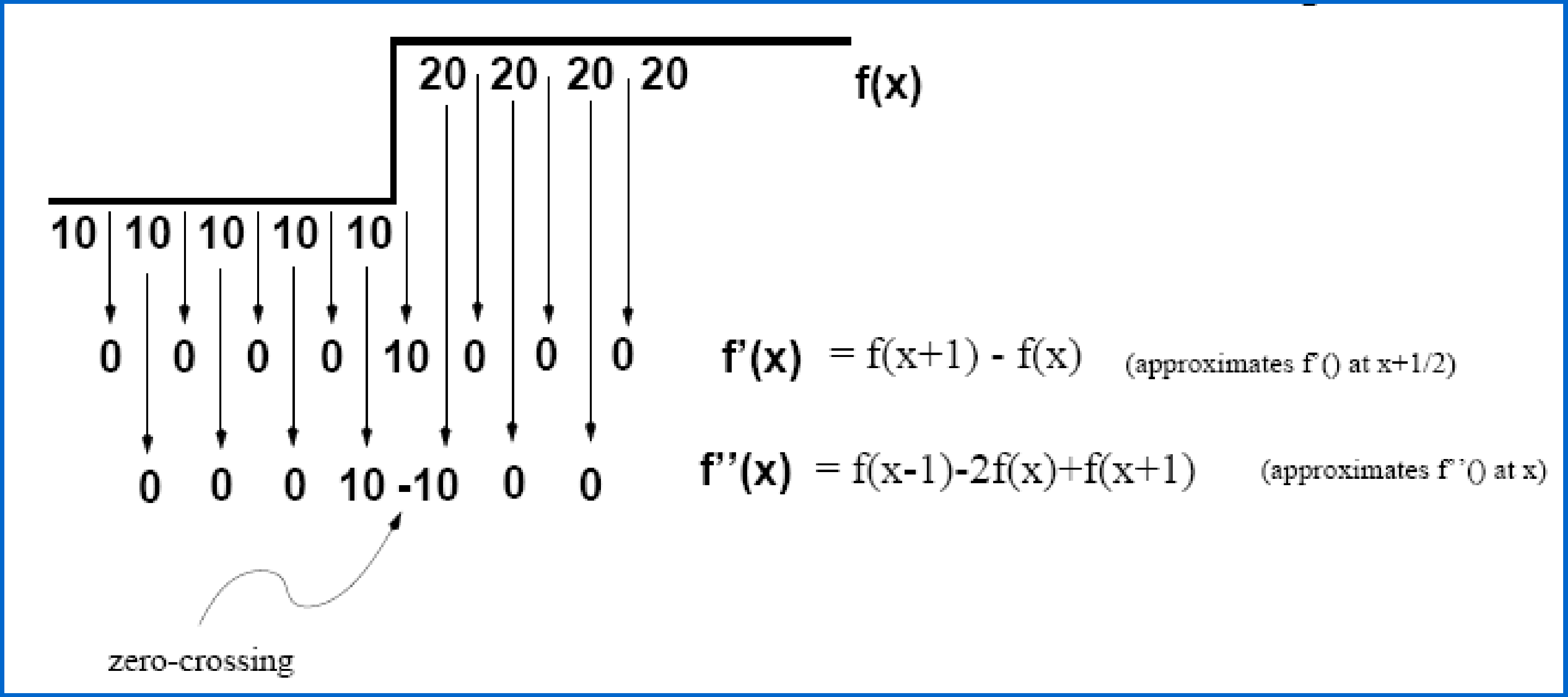
Replace x+1 with x (i.e., centered at x):

$$f''(x) \approx f(x+1) - 2f(x) + f(x-1)$$

$$\Downarrow$$

mask: [1 –2 1]

Edge Detection Using Second Derivative (cont’d)



Edge Detection Using Second Derivative (cont'd)

(upward) step edge

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	0	0	0	0	-12	12	0	0	0	0

(downward) step edge

S_2			24	24	24	24	24	12	12	12	12	12
S_2	$\otimes$	M	0	0	0	0	12	-12	0	0	0	0

Edge Detection Using Second Derivative (cont'd)

(upward) step edge

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	0	0	0	0	-12	12	0	0	0	0

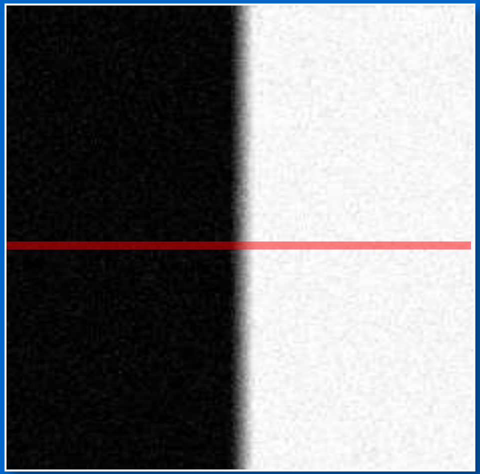
(downward) step edge

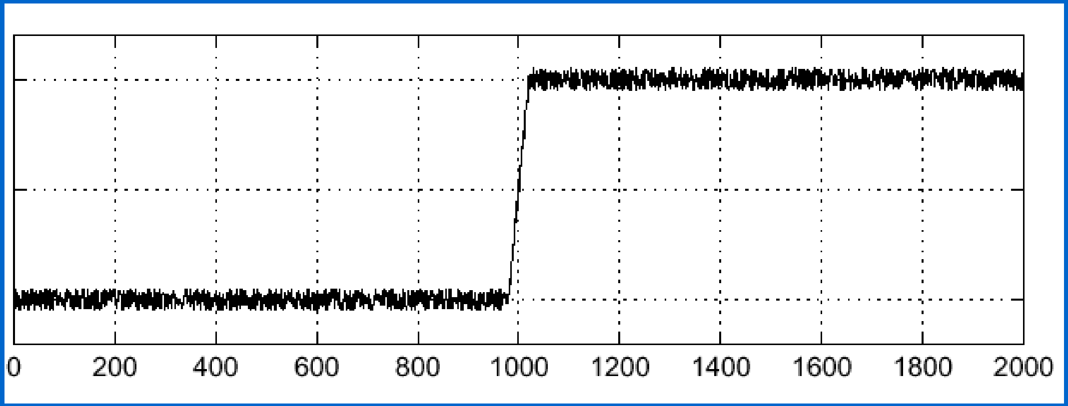
S_2			24	24	24	24	24	12	12	12	12	12
S_2	$\otimes$	M	0	0	0	0	12	-12	0	0	0	0

Zero-crossing

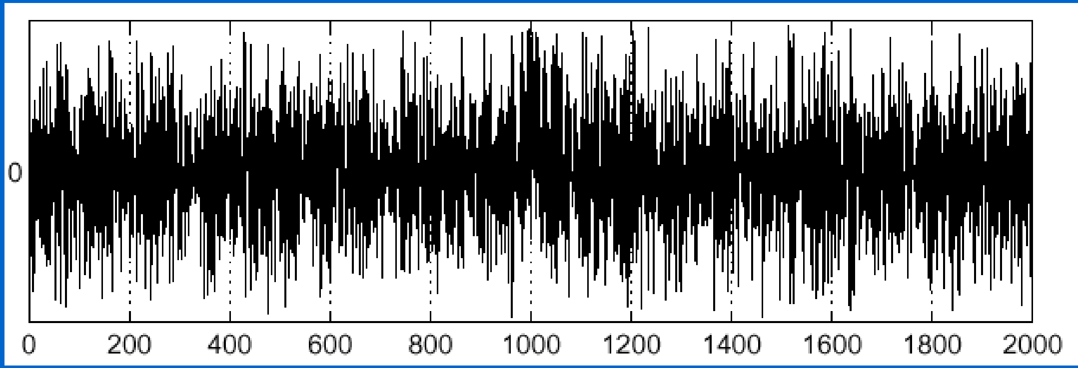
Effect of Noise on Derivatives

$f(x)$





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



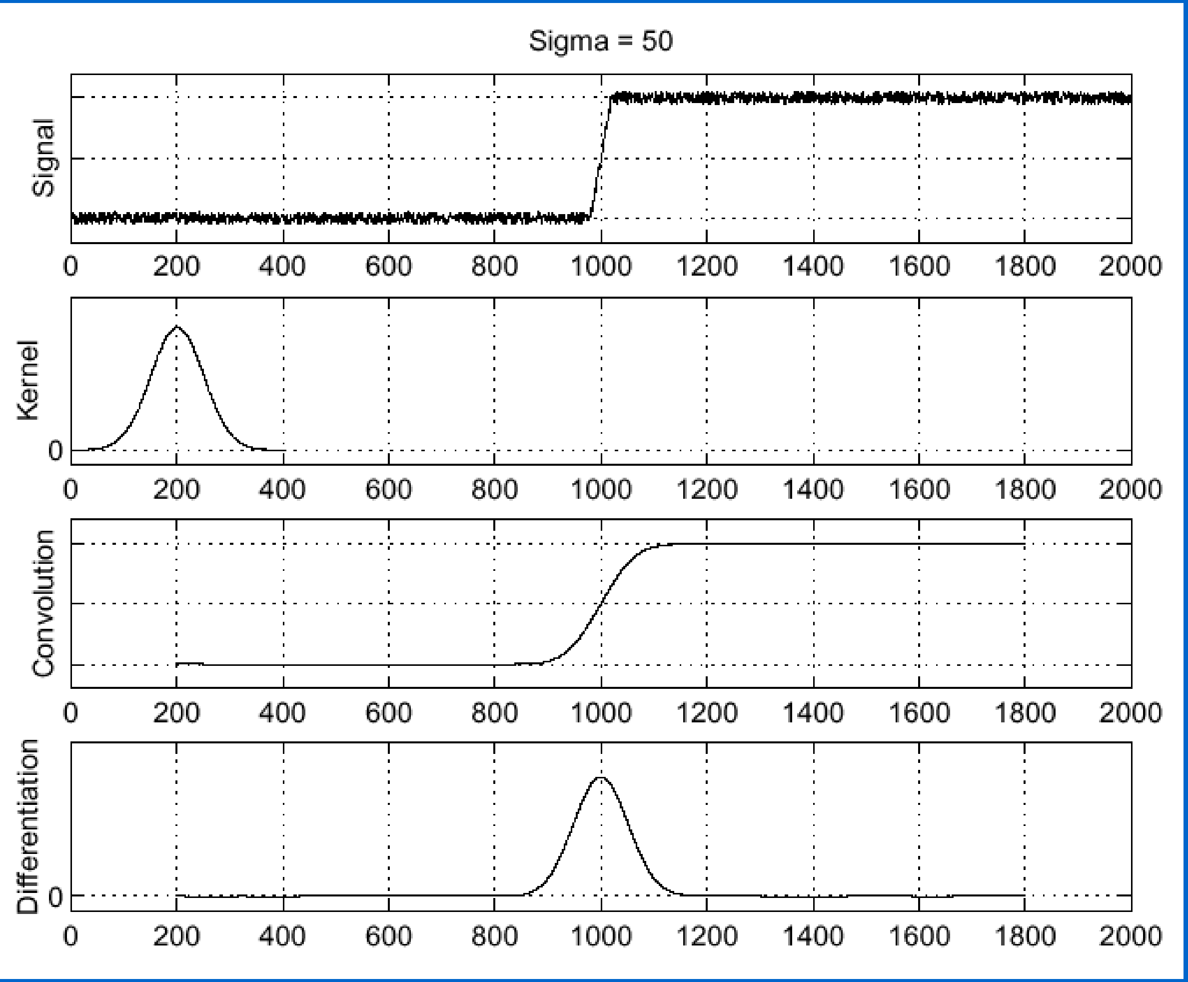
Effect of Smoothing on Derivatives (cont'd)

f

h

$h \star f$

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



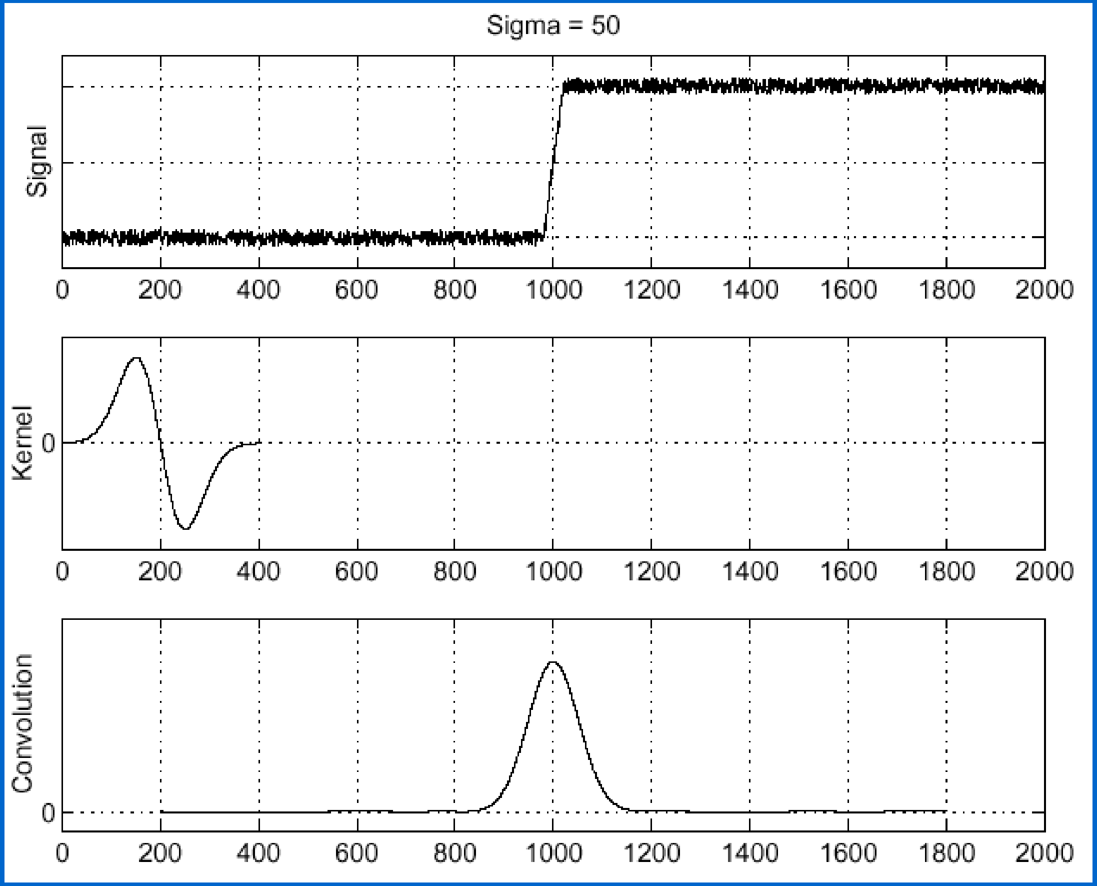
Combine Smoothing with Differentiation

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$ (i.e., saves one operation)

f

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

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

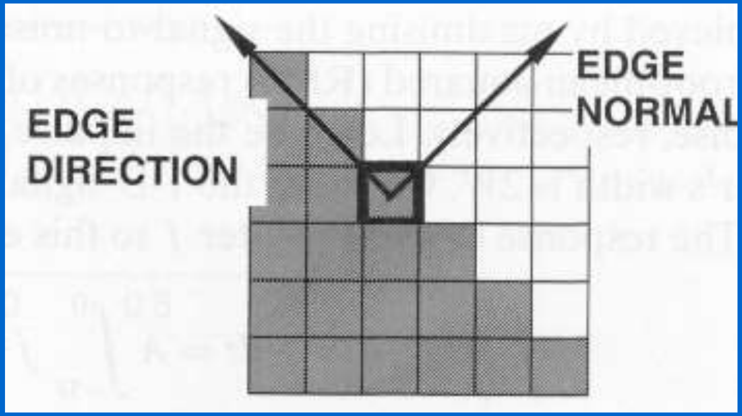


Mathematical Interpretation of combining smoothing with differentiation

- Numerical differentiation is an ill-posed problem.
  - i.e., solution does not exist or it is not unique or it does not depend continuously on initial data)
- Ill-posed problems can be solved using "regularization"
  - i.e., impose additional constraints
- Smoothing performs image interpolation.

Edge Description in 2D

- **Edge direction:** perpendicular to the direction of maximum intensity change (i.e., edge normal)
- **Edge strength:** related to the local image contrast along the normal.
- **Edge position:** the image position at which the edge is located.



Edge Detection Using First Derivative (Gradient)

2D functions:

- The first derivate of an image can be computed using the gradient:

$$\nabla f$$

$$grad(f) = \begin{pmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{pmatrix}$$

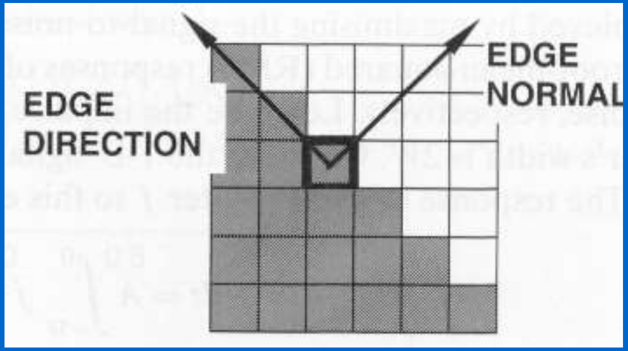
Gradient Representation

- The gradient is a vector which has magnitude and direction:

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

$$direction(grad(f)) = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- **Magnitude:** indicates edge strength.
- **Direction:** indicates edge direction.
  – i.e., perpendicular to edge direction



Approximate Gradient

- Approximate gradient using finite differences:

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

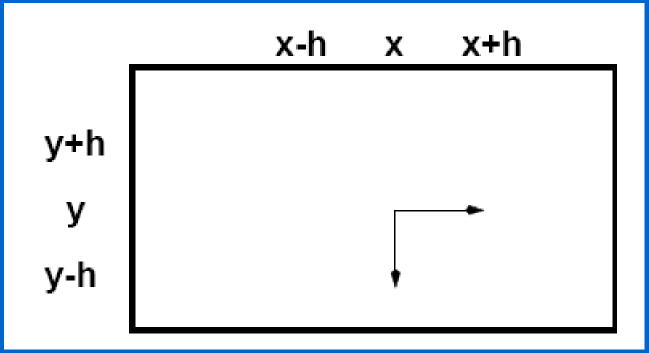
$$\frac{\partial f}{\partial y} = \lim_{h \to 0} \frac{f(x,y+h) - f(x,y)}{h}$$

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

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

Approximate Gradient (cont'd)

- Cartesian vs pixel-coordinates:
  - *j* corresponds to *x* direction
  - *i* to -*y* direction



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

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

Approximating Gradient (cont'd)

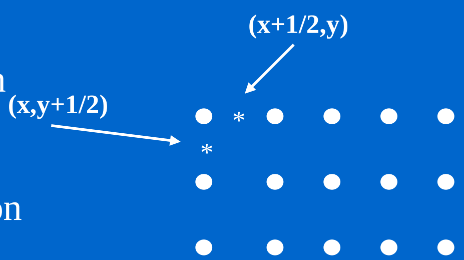
- We can implement $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using the following masks:

-1	1

$\frac{\partial f}{\partial x}$ good approximation at (x+1/2,y)

1
-1

$\frac{\partial f}{\partial y}$ good approximation at (x,y+1/2)

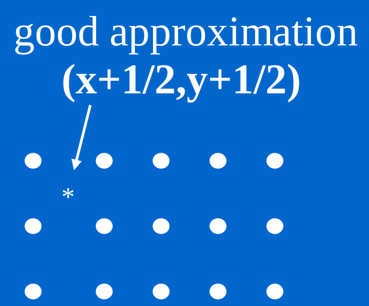


Approximating Gradient (cont'd)

- A different approximation of the gradient:

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

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



- $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ can be implemented using the following masks:

1	0
0	–1

0	1
–1	0

(b) Roberts

Another Approximation

- Consider the arrangement of pixels about the pixel (i, j):

3 x 3 neighborhood:

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

- The partial derivatives $\frac{\partial f}{\partial x}$ $\frac{\partial f}{\partial y}$ 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.

Prewitt Operator

- Setting $c = 1$, we get the Prewitt operator:

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

M_x and M_y are approximations at (i, j)

Sobel Operator

- Setting $c = 2$, we get the Sobel operator:

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

M_x and M_y are approximations at (i, j)

Edge Detection Steps Using Gradient

(1) Smooth the input image ($\hat{f}(x, y) = f(x, y) * G(x, y)$)

(2) $\hat{f}_x = \hat{f}(x, y) * M_x(x, y) \longrightarrow \dfrac{\partial f}{\partial x}$

(3) $\hat{f}_y = \hat{f}(x, y) * M_y(x, y) \longrightarrow \dfrac{\partial f}{\partial y}$

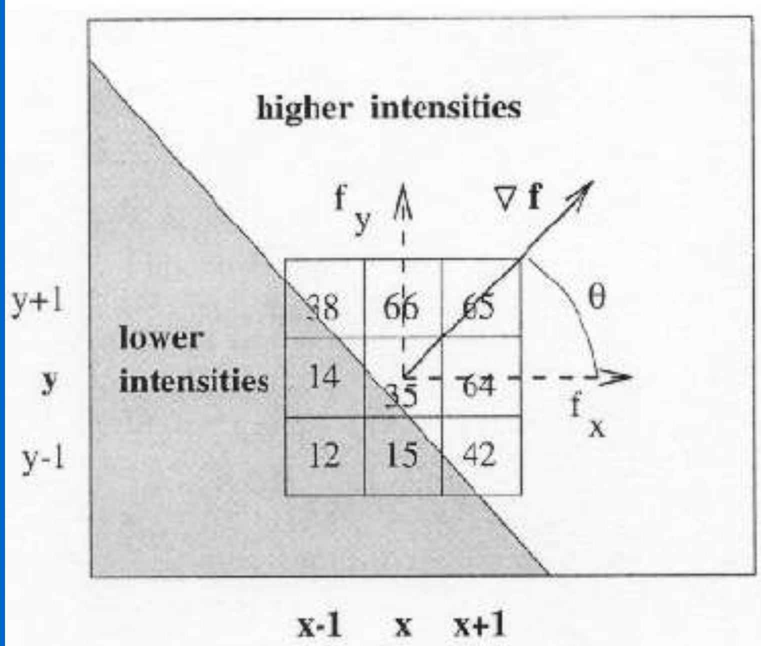
(4) $magn(x, y) = \sqrt{\dfrac{\partial f}{\partial x}^2 + \dfrac{\partial f}{\partial y}^2}$

(5) $dir(x, y) = \tan^{-1}(\hat{f}_y / \hat{f}_x)$

(6) If $magn(x, y) > T$, then possible edge point

Example (using Prewitt operator)

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



f_y = ((38-12)/2 + (66-15)/2 + (65-42)/2) / 3 = (13 + 25 + 11) / 3 = 16

f_x = ((65-38)/2 + (64-14)/2 + (42-12)/2) / 3 = (13 + 25 + 15) / 3 = 18

$\theta = \tan^{-1}(16 / 18) = 0.727$ rad = 42 degrees

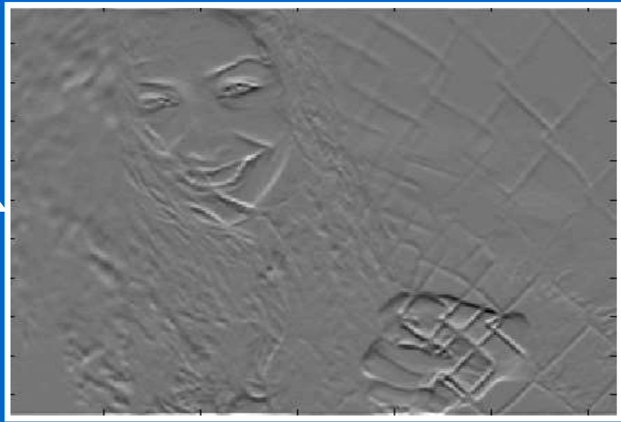
$|\nabla f| = (16^2 + 18^2)^{1/2} = 24$

Note: in this example, the divisions by 2 and 3 in the computation of f_x and f_y are done for normalization purposes only

Another Example

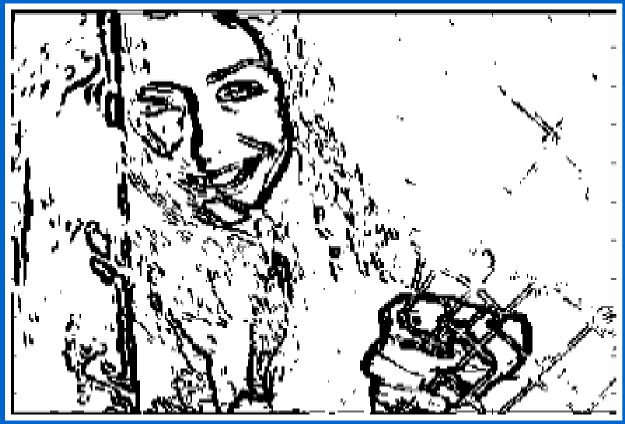
$$\frac{d}{dx}I$$

$$\frac{d}{dy}I$$



Another Example (cont'd)

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



$$\nabla \geq Threshold = 100$$

Isotropic property of gradient magnitude

- The magnitude of the gradient detects edges in all directions.

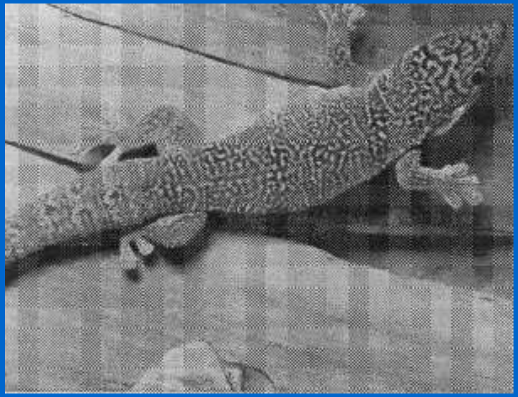
$$\frac{d}{dx}I$$

$$\frac{d}{dy}I$$

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

Practical Issues

- Noise suppression-localization tradeoff.
  - Smoothing depends on mask size (e.g., depends on σ for Gaussian filters).
  - Larger mask sizes reduce noise, but worsen localization (i.e., add uncertainty to the location of the edge) and vice versa.



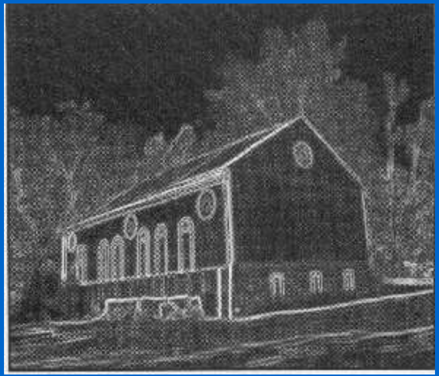
smaller mask

larger mask

Practical Issues (cont'd)

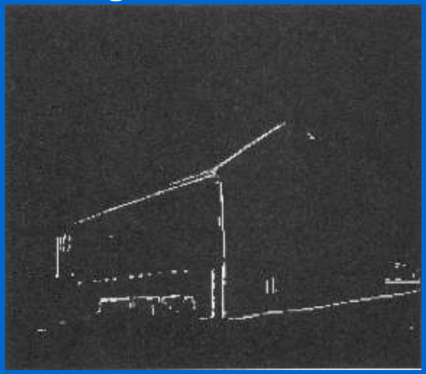
- Choice of threshold.

gradient magnitude



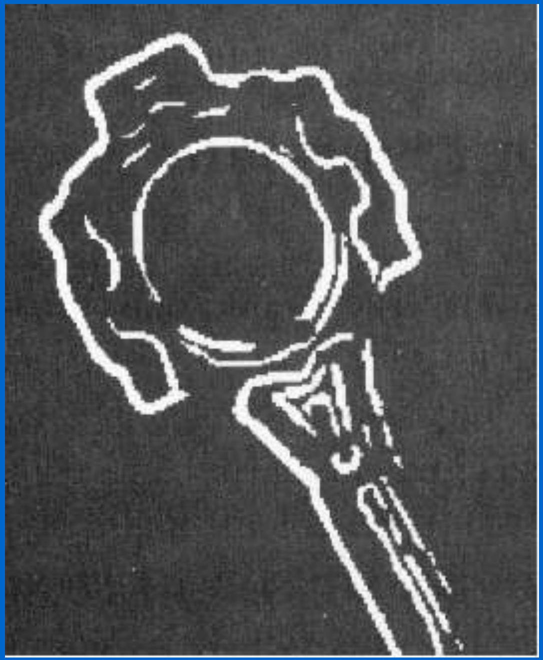
low threshold

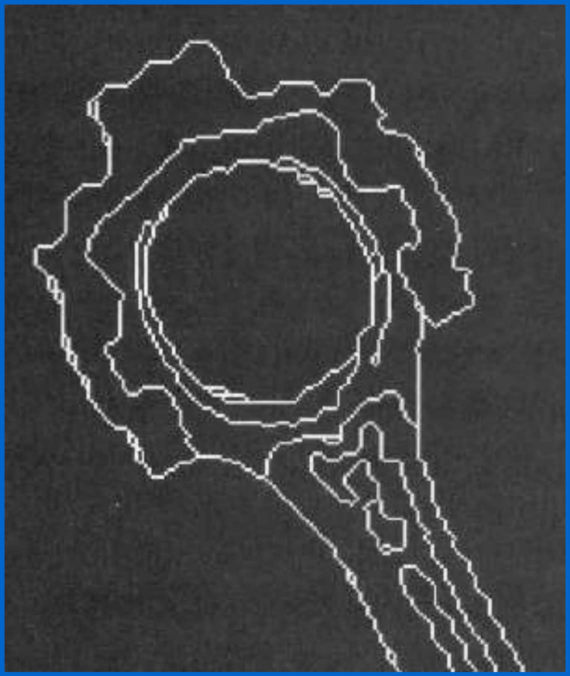
high threshold



Practical Issues (cont'd)

- Edge thinning and linking.





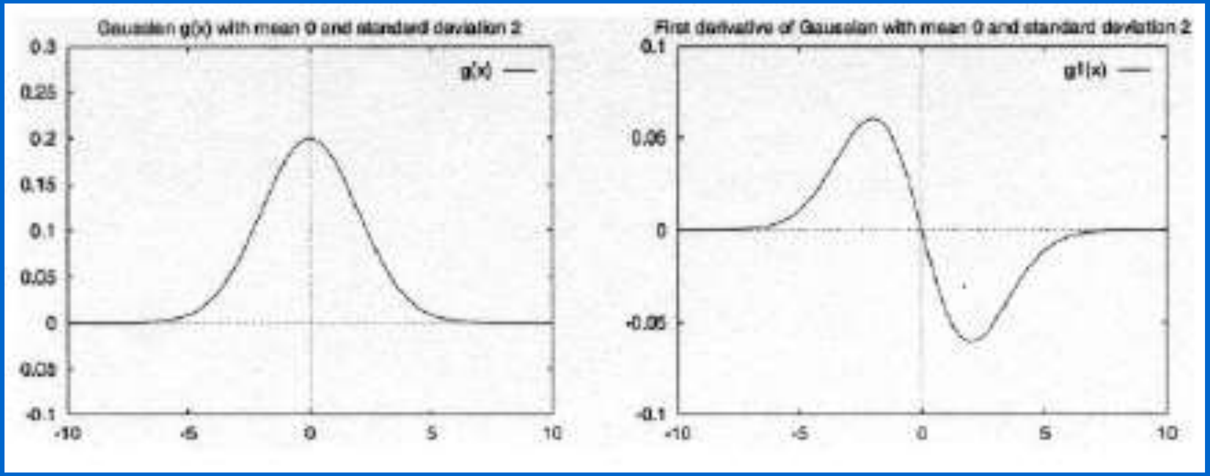
Criteria for Optimal Edge Detection

- **(1) Good detection**
  - Minimize the probability of false positives (i.e., spurious edges).
  - Minimize the probability of false negatives (i.e., missing real edges).
- **(2) Good localization**
  - Detected edges must be as close as possible to the true edges.
- **(3) Single response**
  - Minimize the number of local maxima around the true edge.

Canny edge detector

- Canny has shown that the **first derivative of the Gaussian** closely approximates the operator that optimizes the product of <u>*signal-to-noise* ratio</u> and <u>localization</u>.

(i.e., analysis based on "step-edges" corrupted by "Gaussian noise“)



J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Steps of Canny edge detector

Algorithm

1. Compute f_x and f_y

$$f_x = \frac{\partial}{\partial x}(f * G) = f * \frac{\partial}{\partial x} G = f * G_x$$

$$f_y = \frac{\partial}{\partial y}(f * G) = f * \frac{\partial}{\partial y} G = f * G_y$$

$G(x, y)$ is the Gaussian function

$G_x(x, y)$ is the derivate of $G(x, y)$ with respect to x: $G_x(x, y) = \frac{-x}{\sigma^2} G(x, y)$

$G_y(x, y)$ is the derivate of $G(x, y)$ with respect to y: $G_y(x, y) = \frac{-y}{\sigma^2} G(x, y)$

Steps of Canny edge detector (cont'd)

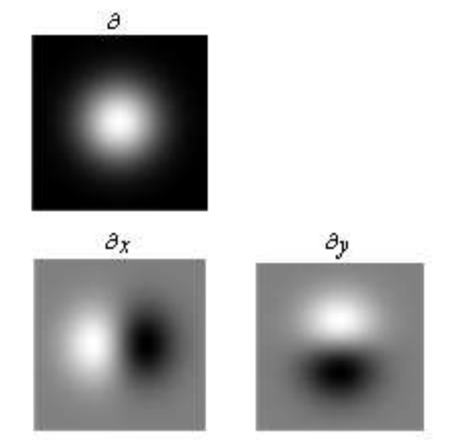
2. Compute the gradient magnitude (and direction)

$$magnitude(grad(f)) = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2} \qquad dir(x, y) = \tan^{-1}(\hat{f}_y / \hat{f}_x)$$

3. Apply non-maxima suppression.

4. Apply hysteresis thresholding/edge linking.

2D Edge Filter: Output at different scales



1[st] order Gaussian Derivatives

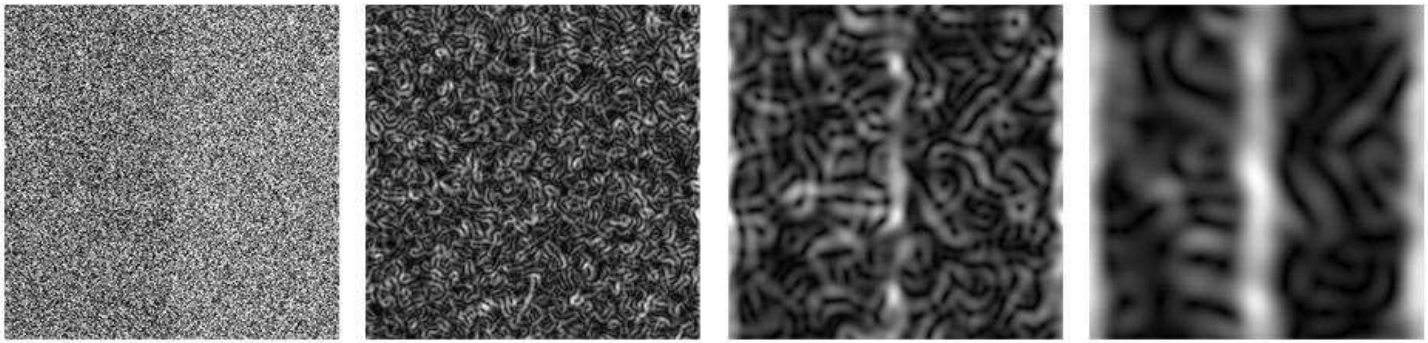


Figure 5.11 Detection of a very low contrast step-edge in noise. Left: original image, the step-edge is barely visible. At small scales (second image, $\sigma = 2$ pixels) the edge is not detected. We see the edges of the noise itself, cluttering the edge of the step-edge. Only at large scale (right, $\sigma = 12$ pixels) the edge is clearly found. At this scale the large scale structure of the edge emerges from the small scale structure of the noise.

Response at different scales

Figure 5.11 Gradient edges detected at different scales (σ = 0.5, 2, 5 pixels resp.).
coarser edges (right) indicate hierarchically more 'important' edges.

Canny edge detector - example

original image

Canny edge detector – example (cont'd)

Gradient magnitude

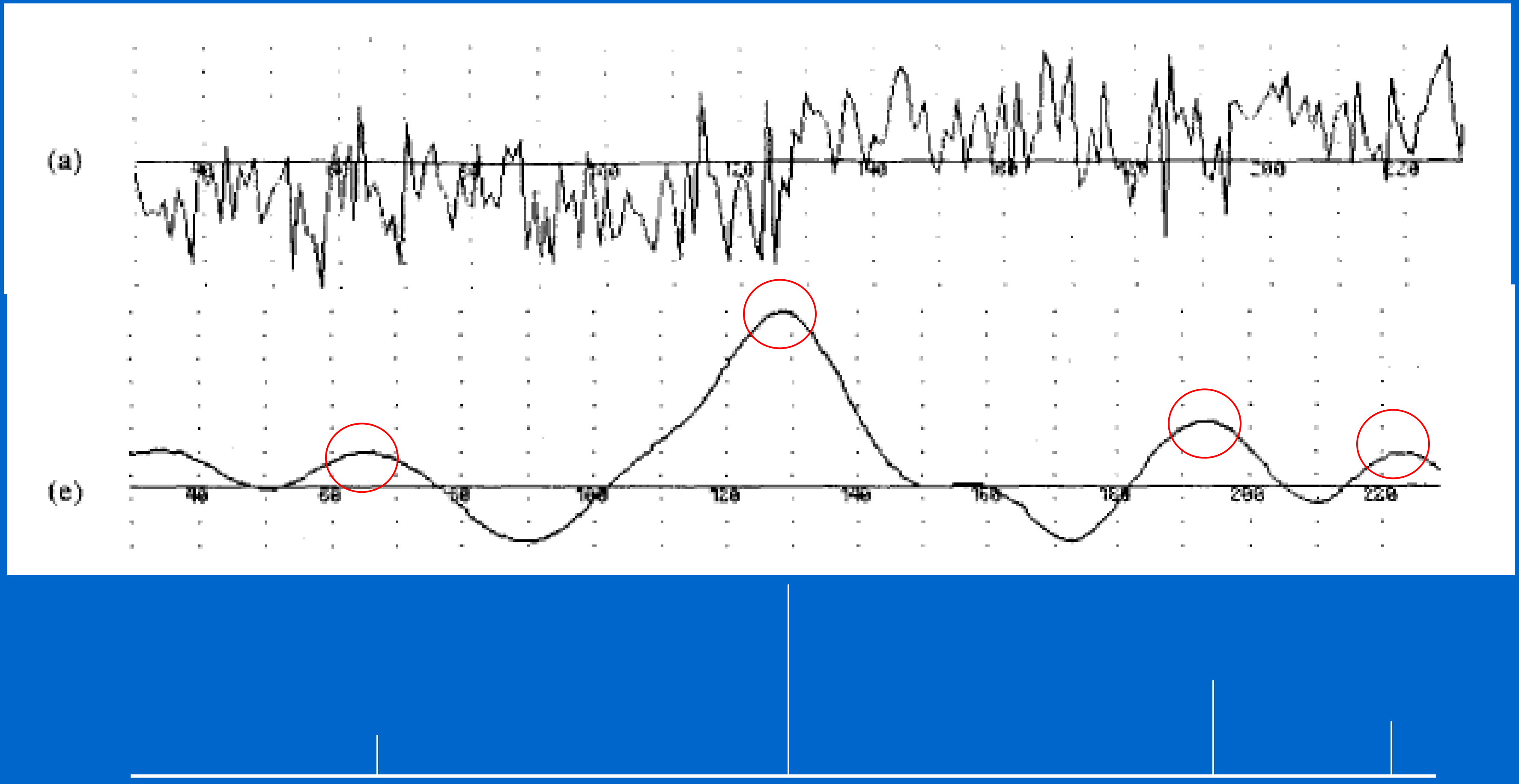
Canny edge detector – example (cont'd)

Thresholded gradient magnitude

Canny edge detector – example (cont'd)

Thinning (non-maxima suppression)

Non-Maximum Suppression

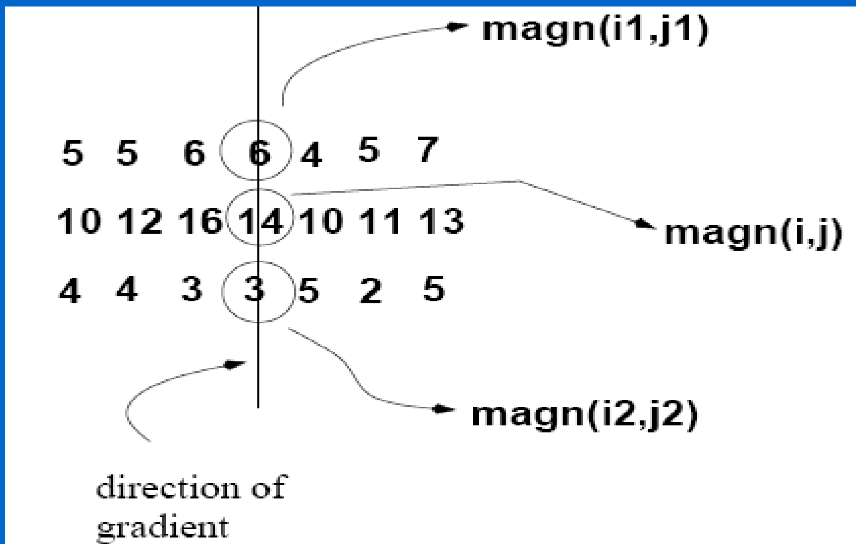


Detect local maxima and suppress all other signals.

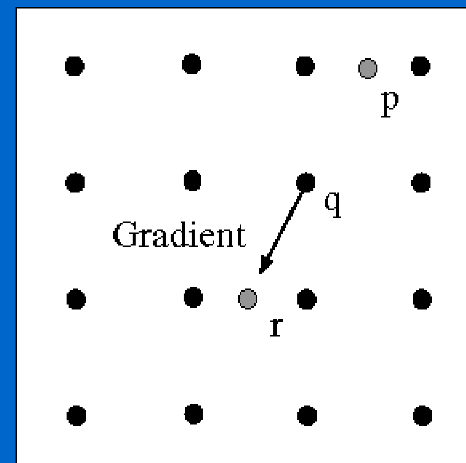
Non-maxima suppression

- Check if gradient magnitude at pixel location (i,j) is local maximum along gradient direction

Non-maxima suppression (cont'd)



Warning: requires checking interpolated pixels p and r



Algorithm

For each pixel (i,j) do:

if $magn(i, j) < magn(i_1, j_1)$ or $magn(i, j) < magn(i_2, j_2)$
    then $I_N(i, j) = 0$
else $I_N(i, j) = magn(i, j)$

Hysteresis thresholding

- Standard thresholding:

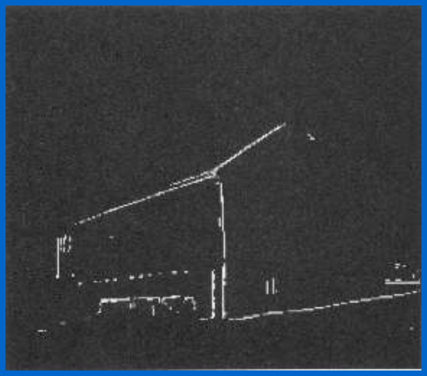
$$E(x,y) = \begin{cases} 1 & \text{if } \|\nabla f(x,y)\| > T \text{ for some threshold } T \\ 0 & \text{otherwise} \end{cases}$$

- Can only select "strong" edges.
- Does not guarantee "continuity".

gradient magnitude

low threshold

high threshold



Hysteresis thresholding (cont'd)

- Hysteresis thresholding uses two thresholds:

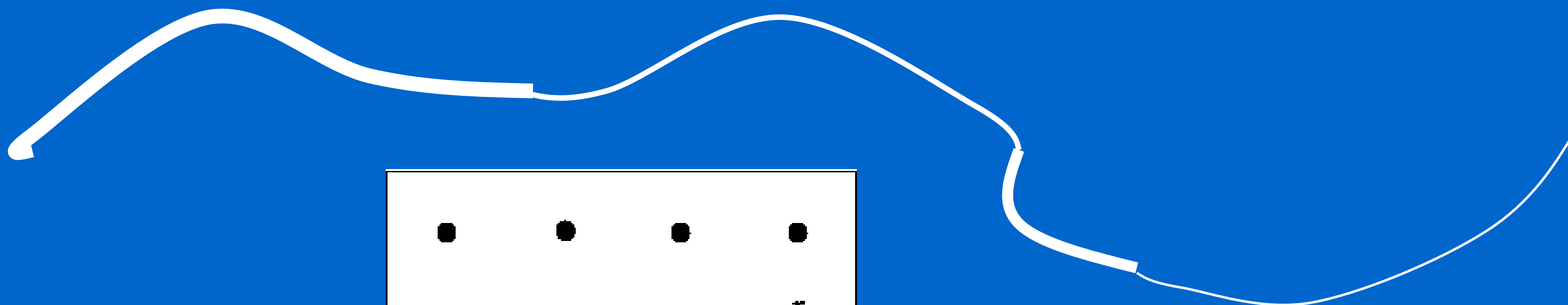
  - low threshold t_l
  - high threshold t_h (usually, $t_h = 2t_l$)

$$
\begin{array}{rll}
\|\nabla f(x,y)\| \geq t_h & & \text{definitely an edge} \\
t_l \geq \|\nabla f(x,y)\| < t_h & & \text{maybe an edge, depends on context} \\
\|\nabla f(x,y)\| < t_l & & \text{definitely not an edge}
\end{array}
$$

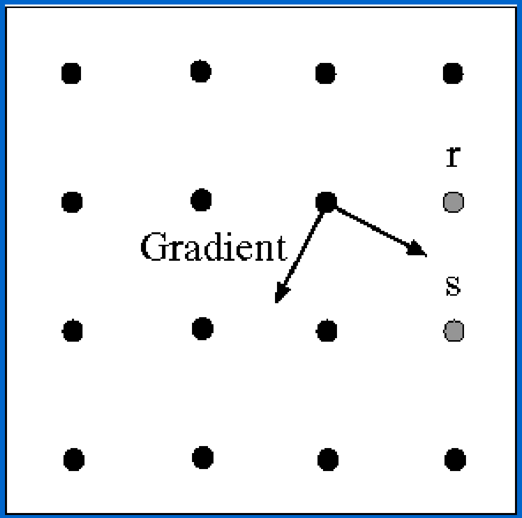
- For "maybe" edges, decide on the edge if <u>neighboring</u> pixel is a strong edge.

Hysteresis thresholding/Edge Linking

Idea: use a **high** threshold to start edge curves and a **low** threshold to continue them.



Use edge “direction” for linking edges



Hysteresis Thresholding/Edge Linking (cont'd)

Algorithm

1. Produce two thresholded images $I_1(i, j)$ and $I_2(i, j)$. (using t_l and t_h)

(note: since $I_2(i, j)$ was formed with a high threshold, it will contain fewer false edges but there might be gaps in the contours)

2. Link the edges in $I_2(i, j)$ into contours

   2.1 Look in $I_1(i, j)$ when a gap is found.

   2.2 By examining the 8 neighbors in $I_1(i, j)$, gather edge points from $I_1(i, j)$ until the gap has been bridged to an edge in $I_2(i, j)$.

Note: large gaps are still difficult to bridge.
(i.e., more sophisticated algorithms are required)