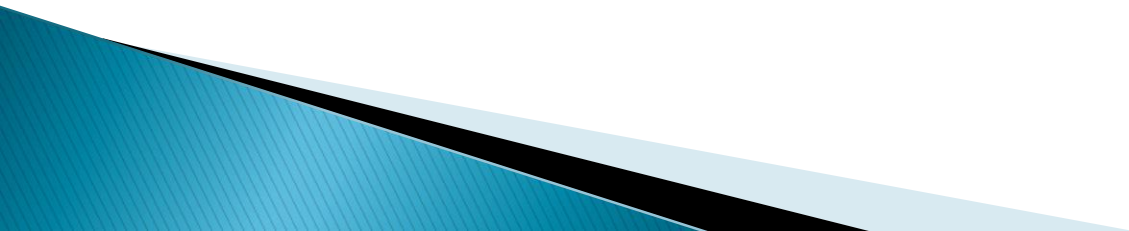


# Edge Detection

# Edge detection and image gradient



# Detection of Isolated Points

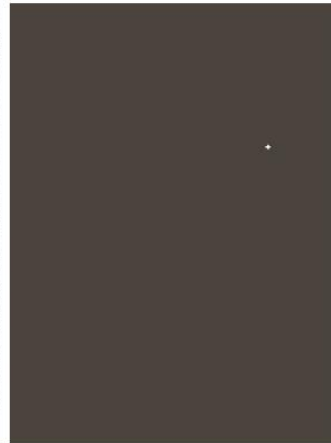
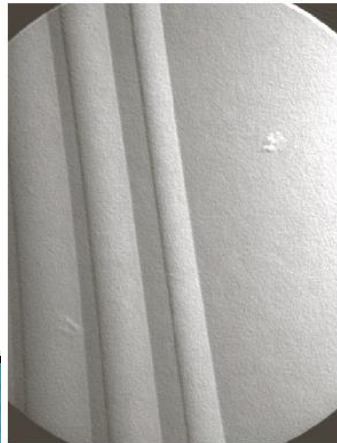
$$|R| \geq T$$

T: Threshold

$$R = w_1z_1 + w_2z_2 + \dots + w_9z_9$$

$w_1$	$w_2$	$w_3$
$w_4$	$w_5$	$w_6$
$w_7$	$w_8$	$w_9$

1	1	1
1	-8	1
1	1	1

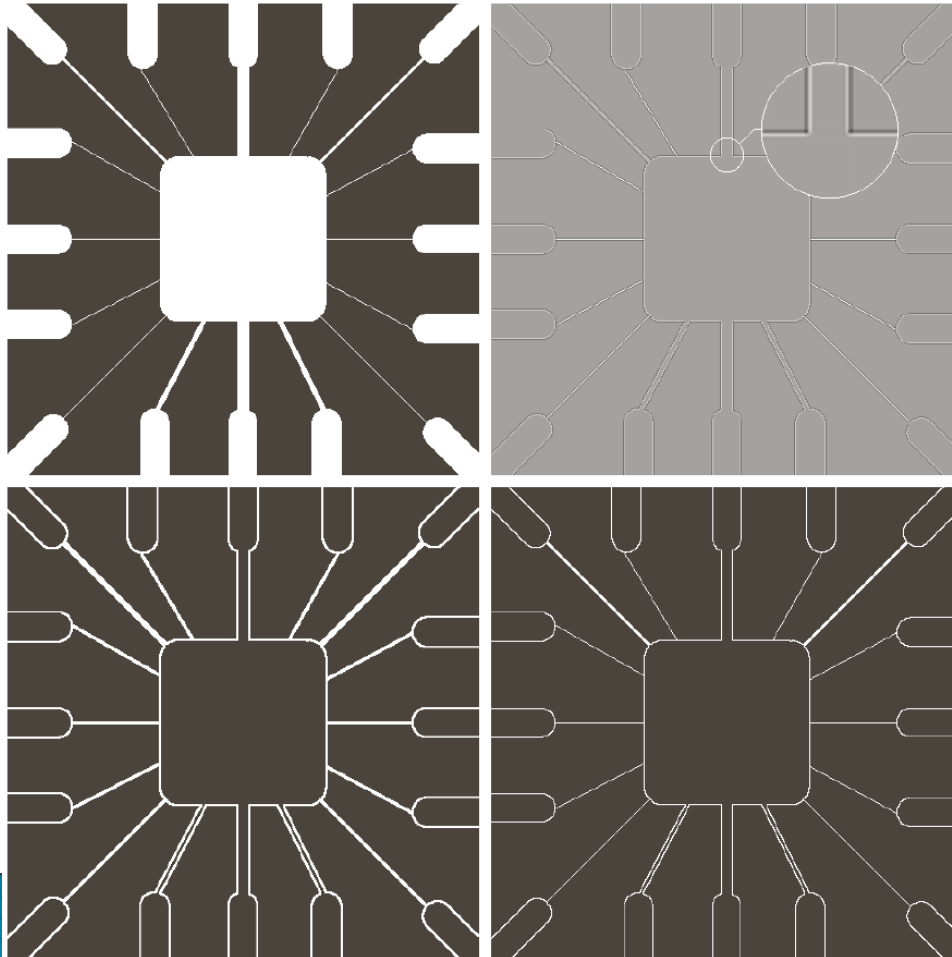


a  
b c d

**FIGURE 10.4**

(a) Point detection (Laplacian) mask.  
(b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel.  
(c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

# Line Detection: Laplacian Image



a b  
c d

**FIGURE 10.5**

(a) Original image.

(b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian.

(c) Absolute value of the Laplacian.

(d) Positive values of the Laplacian.

Laplacian  
detector is  
Isotropic

# Line Detector

- ▶ Horizontal mask will result with max response when a line passed through the middle row of the mask with a constant background.
- ▶ The similar idea is used with other masks.

-1	-1	-1	2	-1	-1	-1	2	-1	-1	-1	2
2	2	2	-1	2	-1	-1	2	-1	-1	2	-1
-1	-1	-1	-1	-1	2	-1	2	-1	2	-1	-1
Horizontal			+45°			Vertical			-45°		

**FIGURE 10.6** Line detection masks. Angles are with respect to the axis system in Fig. 2.18(b).

# 2D discrete derivative – example

What happens when we apply this filter?

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

# 2D discrete derivative – example

What happens when we apply this filter?

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

# 2D discrete derivative – example

Now let's try the other filter!

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$\frac{1}{3} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$



# 2D discrete derivative – example

What happens when we apply this filter?

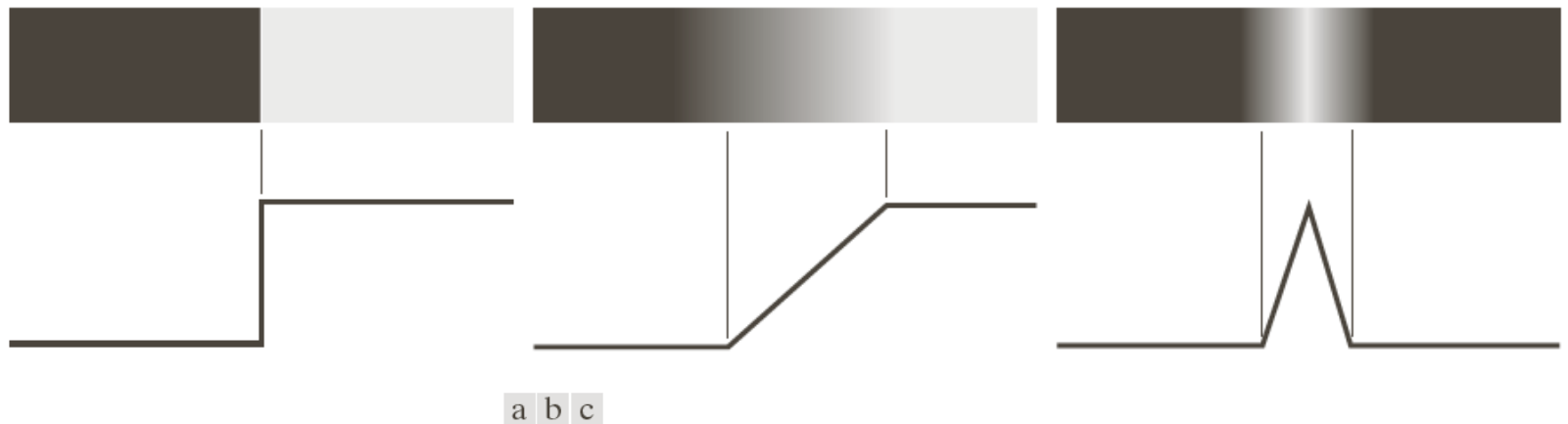
$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$\frac{1}{3} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

# Edge Models

- ▶ Edge detection is the approach used most frequently for segmentation image based on abrupt changes in intensity.



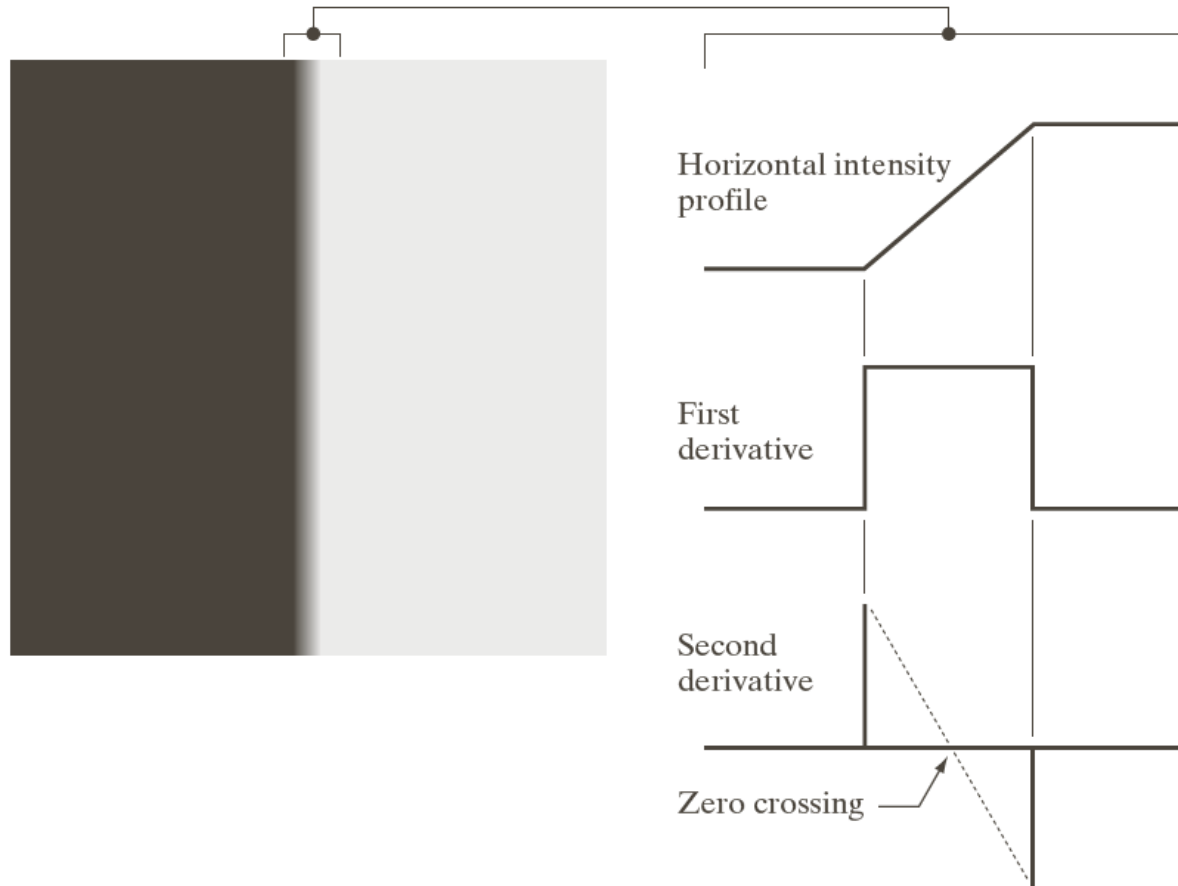
**FIGURE 10.8**

From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.



**FIGURE 10.9** A  $1508 \times 1970$  image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and “step” profiles span 9 pixels and 2 pixels, respectively. The base of the roof edge is 3 pixels. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

# First and Second Derivatives



a b

**FIGURE 10.10**

(a) Two regions of constant intensity separated by an ideal vertical ramp edge.

(b) Detail near the edge, showing a horizontal intensity profile, together with its first and second derivatives.

# Image Gradient and its Property

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad \nabla f = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

$$\alpha(x, y) = \tan^{-1} \left[ \frac{g_y}{g_x} \right]$$

# Various Masks

$z_1$	$z_2$	$z_3$
$z_4$	$z_5$	$z_6$
$z_7$	$z_8$	$z_9$

-1	0	0	-1
0	1	1	0

Roberts

-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

Prewitt

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

Sobel

a
b c
d e
f g

**FIGURE 10.14**

A  $3 \times 3$  region of an image (the  $z$ 's are intensity values) and various masks used to compute the gradient at the point labeled  $z_5$ .

$$\nabla f = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$



Approximation

$$\nabla f \approx |g_x| + |g_y|$$

# Prewitt and Sobel Masks for Detecting Diagonal Edges

0	1	1	-1	-1	0
-1	0	1	-1	0	1
-1	-1	0	0	1	1

Prewitt

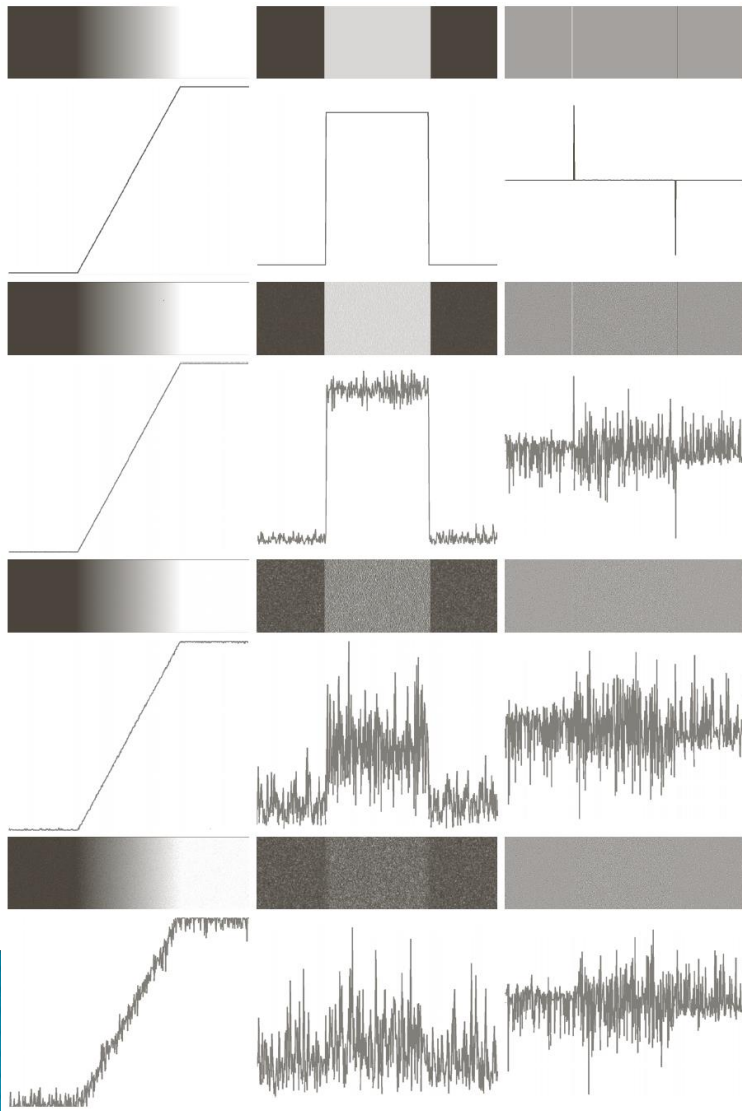
0	1	2	-2	-1	0
-1	0	1	-1	0	1
-2	-1	0	0	1	2

Sobel

a	b
c	d

**FIGURE 10.15**  
Prewitt and Sobel  
masks for  
detecting diagonal  
edges.

# Edge Detection: Effect of Noise



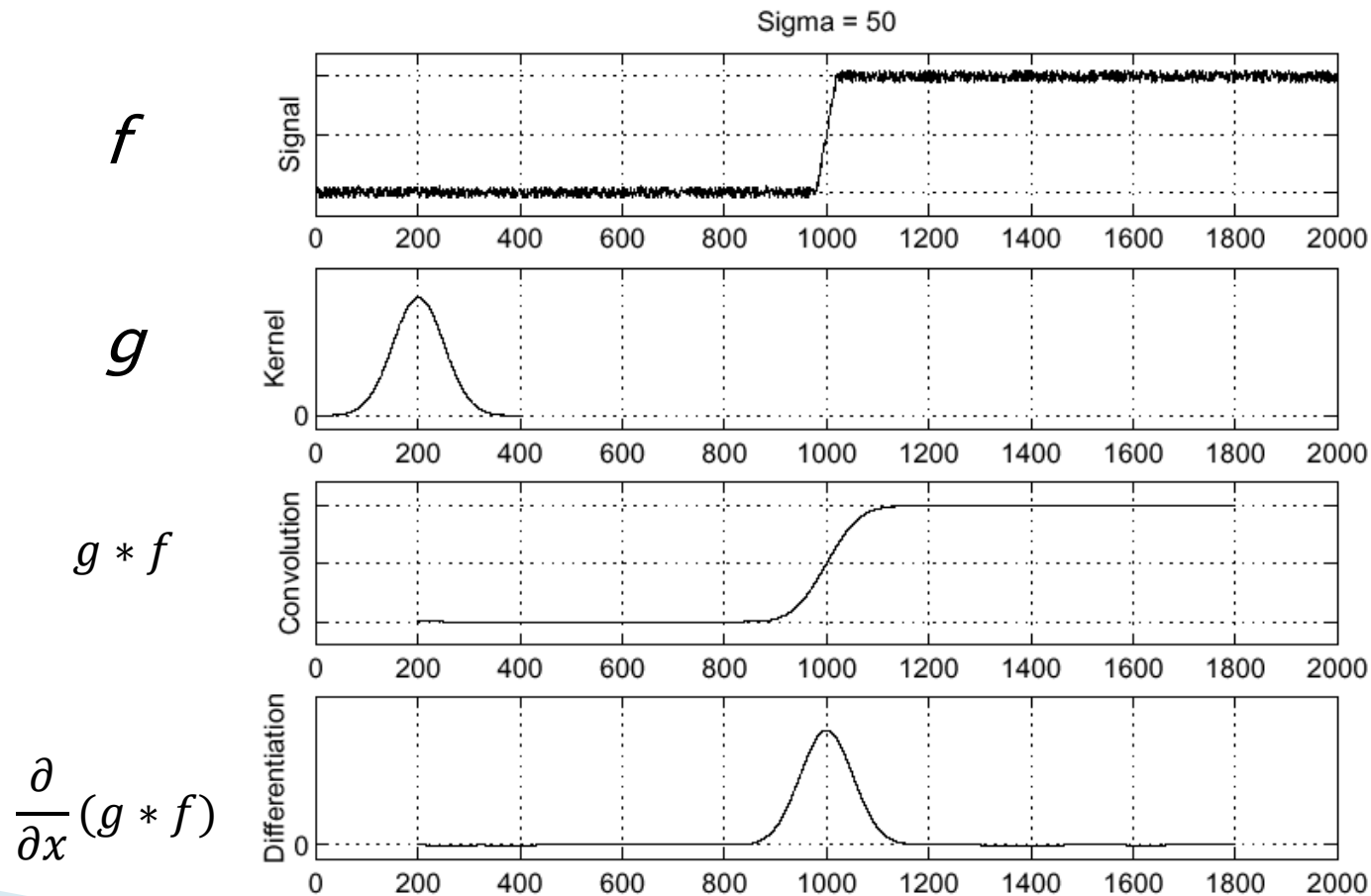
**FIGURE 10.11** First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.



# Three Fundamental Steps

- ▶ Image smoothing for noise reduction
- ▶ Detection of edge points
- ▶ Edge localization

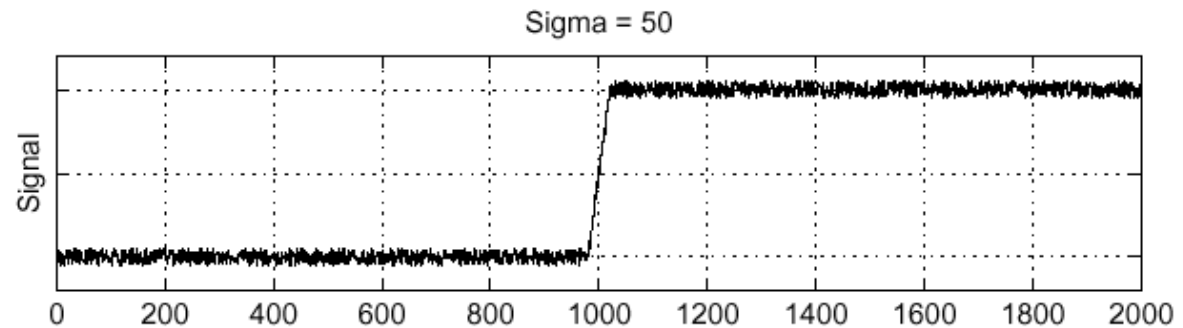
# Solution: smooth first



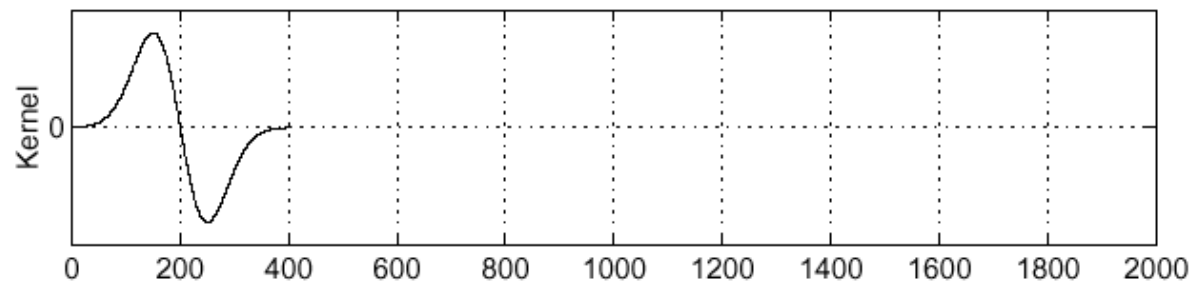
# Derivative theorem of convolution

$$\frac{\partial}{\partial x}(g * f) = \left(\frac{\partial}{\partial x}g\right) * f$$

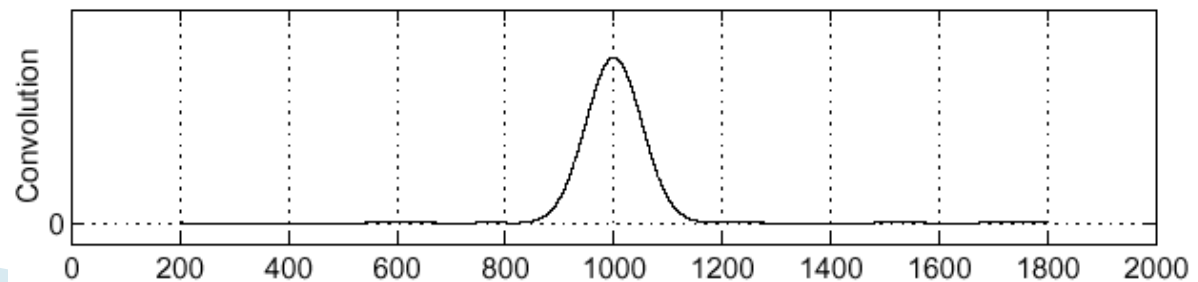
$f$



$\frac{\partial}{\partial x}g$

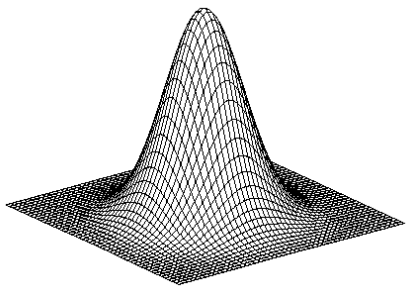


$\left(\frac{\partial}{\partial x}g\right) * f$



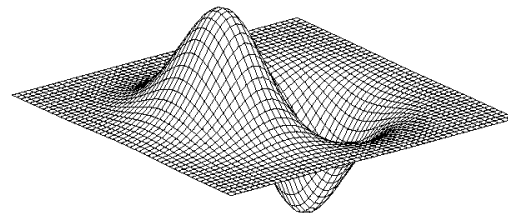
# Implementation of Derivative

$$\left(\frac{\partial}{\partial x} g\right) * f = h * g * f = (h * g) * f$$



2D-gaussian

$$* [1 \ 0 \ -1] =$$



x - derivative

$z_1$	$z_2$	$z_3$
$z_4$	$z_5$	$z_6$
$z_7$	$z_8$	$z_9$

-1	0	0	-1
0	1	1	0

Roberts

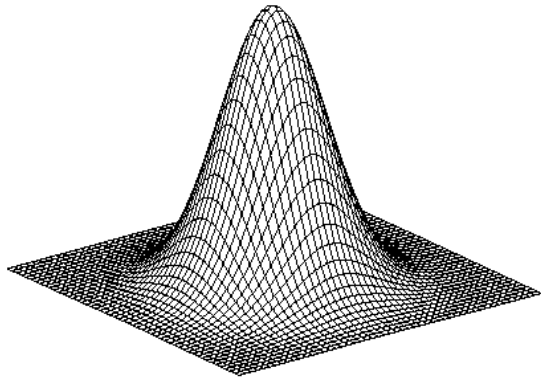
-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

Prewitt

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

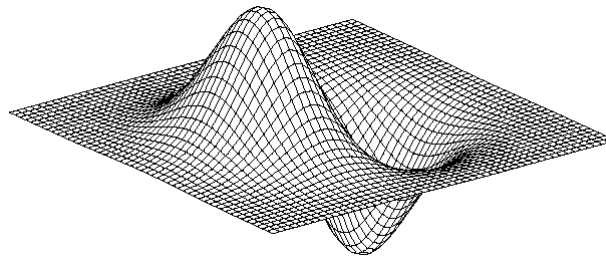
Sobel

# 2D edge detection filters



**Gaussian**

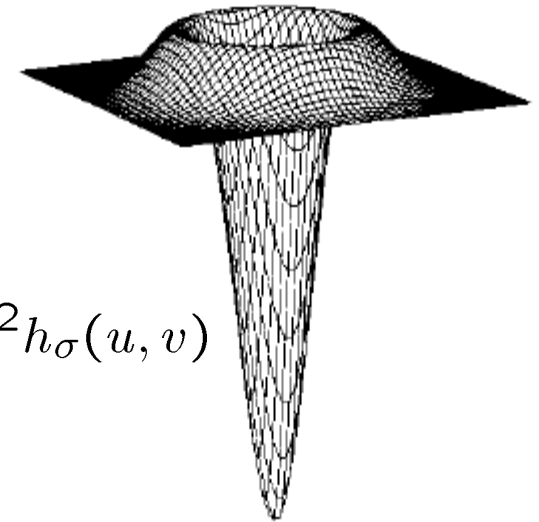
$$h_{\sigma}(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



**derivative of Gaussian**

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

**Laplacian of Gaussian**



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

- $\nabla^2$  is the Laplacian operator:

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

# Example



a	b
c	d

**FIGURE 10.16**

(a) Original image of size  $834 \times 1114$  pixels, with intensity values scaled to the range  $[0, 1]$ .

(b)  $|g_x|$ , the component of the gradient in the  $x$ -direction, obtained using the Sobel mask in Fig. 10.14(f) to filter the image.

(c)  $|g_y|$ , obtained using the mask in Fig. 10.14(g).

(d) The gradient image,  $|g_x| + |g_y|$ .



# Example



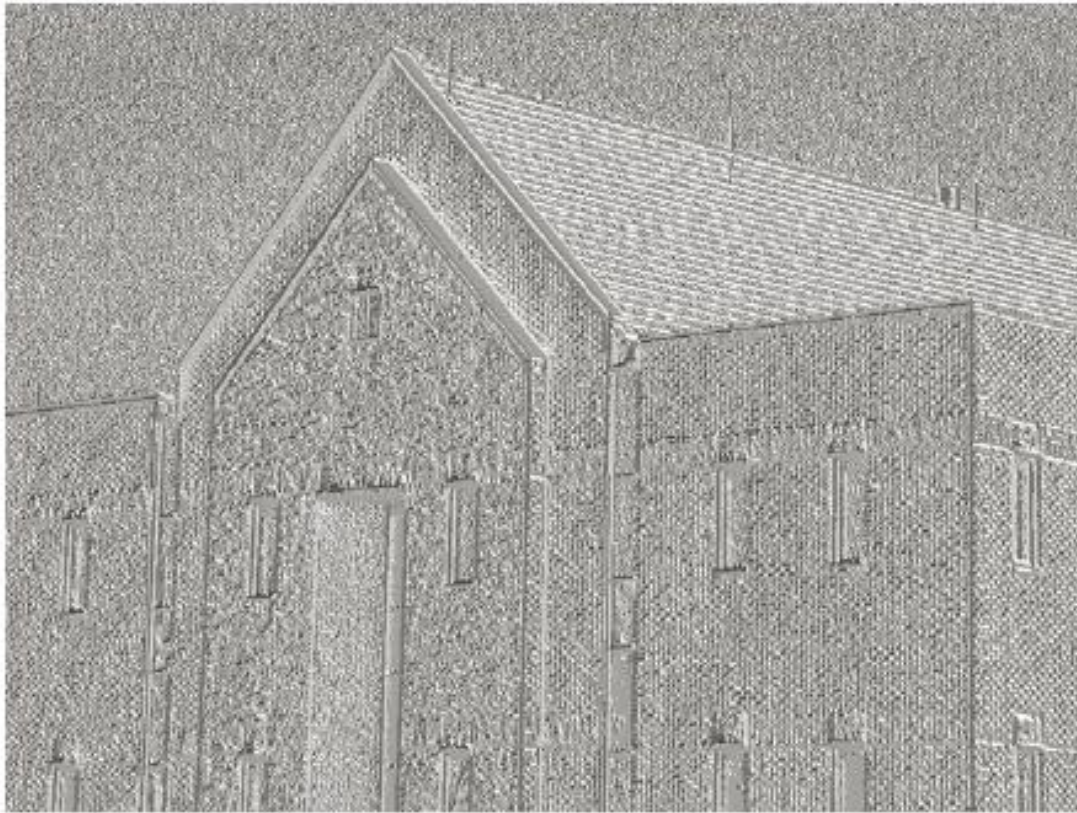
a	b
c	d

**FIGURE 10.18**

Same sequence as in Fig. 10.16, but with the original image smoothed using a  $5 \times 5$  averaging filter prior to edge detection.



# Example



**FIGURE 10.17**  
Gradient angle  
image computed  
using  
Eq. (10.2-11).  
Areas of constant  
intensity in this  
image indicate  
that the direction  
of the gradient  
vector is the same  
at all the pixel  
locations in those  
regions.

---



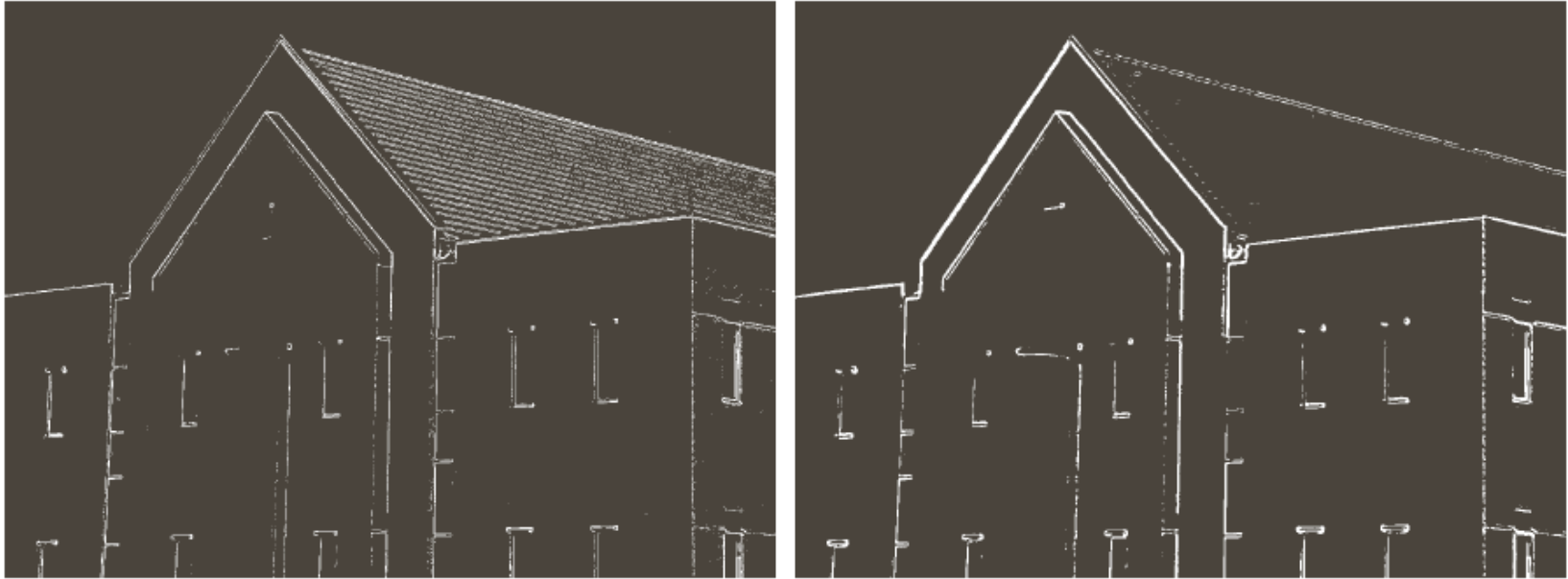
# Example



a b

**FIGURE 10.19**  
Diagonal edge detection.  
(a) Result of using the mask in Fig. 10.15(c).  
(b) Result of using the mask in Fig. 10.15(d). The input image in both cases was Fig. 10.18(a).

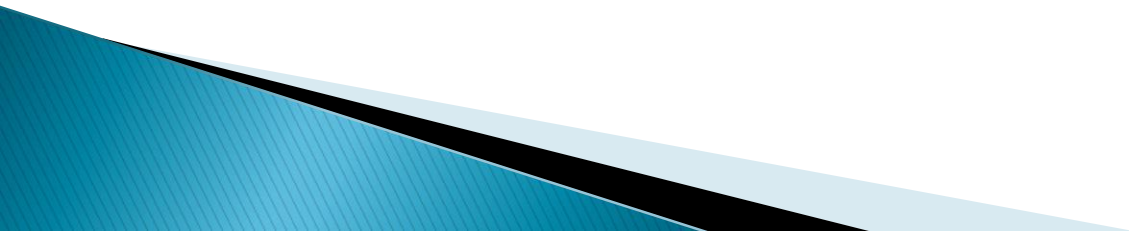
# Combining the Gradient with Thresholding



a b

**FIGURE 10.20** (a) Thresholded version of the image in Fig. 10.16(d), with the threshold selected as 33% of the highest value in the image; this threshold was just high enough to eliminate most of the brick edges in the gradient image. (b) Thresholded version of the image in Fig. 10.18(d), obtained using a threshold equal to 33% of the highest value in that image.

# Canny edge detector



# Canny Edge Detector

- ▶ Although the algorithm is complex, the performance of the Canny edge detector is superior in general to the edge detectors discussed thus far.
- ▶ This is probably the most widely used edge detector in computer vision.

J. Canny, [A Computational Approach To Edge Detection](#), IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.



# Canny Edge Detector

- ▶ Three basic objectives
  - Low error rate: Finding all edges and nothing but edges.
  - Edge points should be well localized – Distance between edges found and actual edges should be minimized.
  - Single edge point response: No multiple edge pixels when only a single edge exists.

# Examples



a	b
c	d

**FIGURE 10.25**

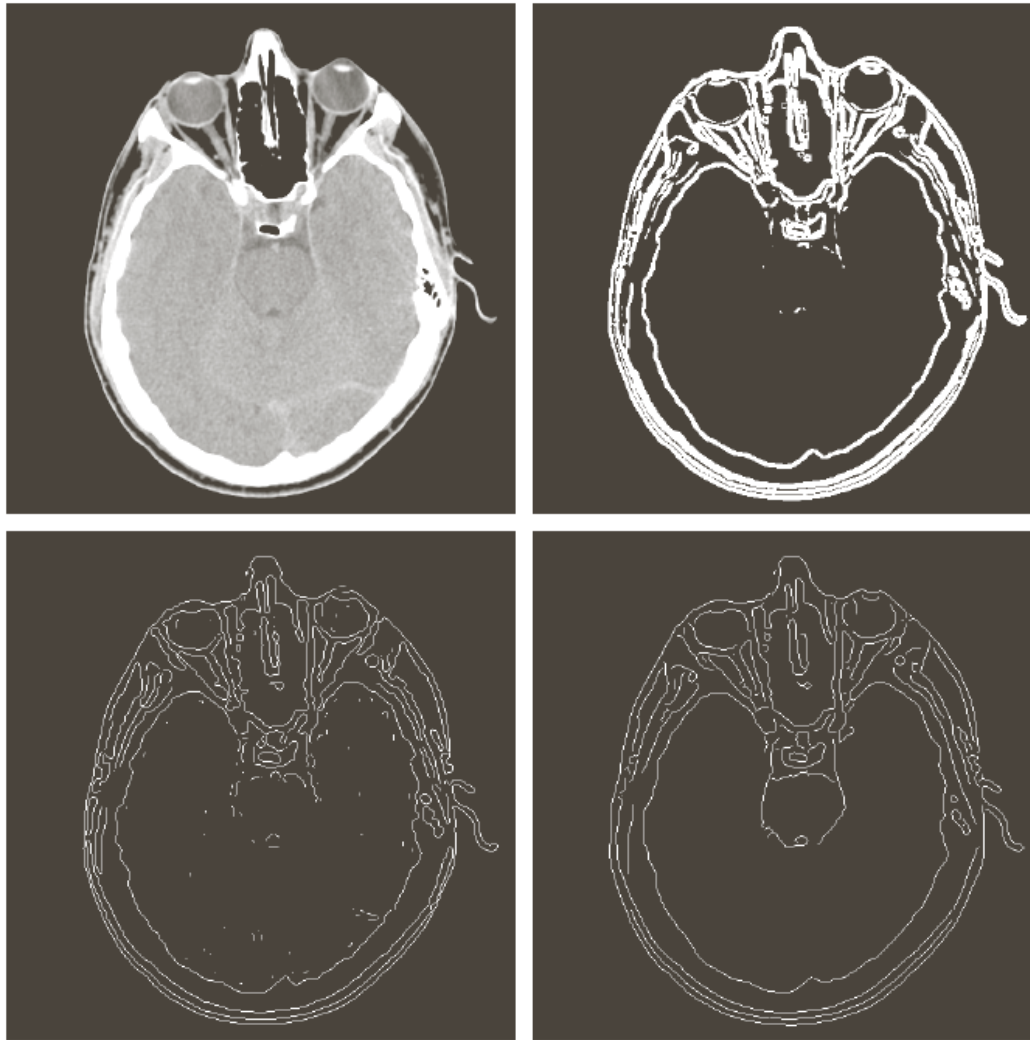
(a) Original image of size  $834 \times 1114$  pixels, with intensity values scaled to the range  $[0, 1]$ .

(b) Thresholded gradient of smoothed image.

(c) Image obtained using the Marr-Hildreth algorithm.

(d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

# Examples



a	b
c	d

**FIGURE 10.26**

(a) Original head CT image of size  $512 \times 512$  pixels, with intensity values scaled to the range  $[0, 1]$ .

(b) Thresholded gradient of smoothed image.

(c) Image obtained using the Marr-Hildreth algorithm.

(d) Image obtained using the Canny algorithm.

(Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)



# Canny edge detector

- ▶ Filter image with derivative of Gaussian
- ▶ Find magnitude and orientation of gradient
- ▶ **Non-maximum suppression**
- ▶ **Linking and thresholding (hysteresis)**
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them



# The Canny edge detector



original image (Lena)

# The Canny edge detector



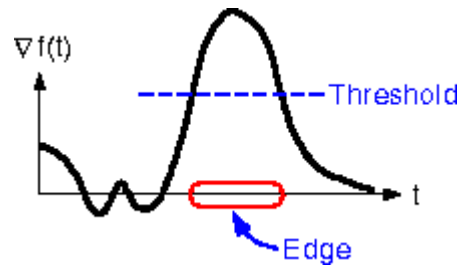
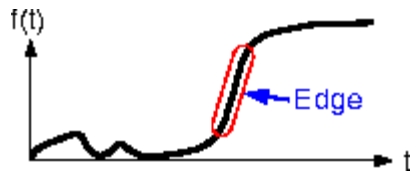
norm of the gradient

# The Canny edge detector



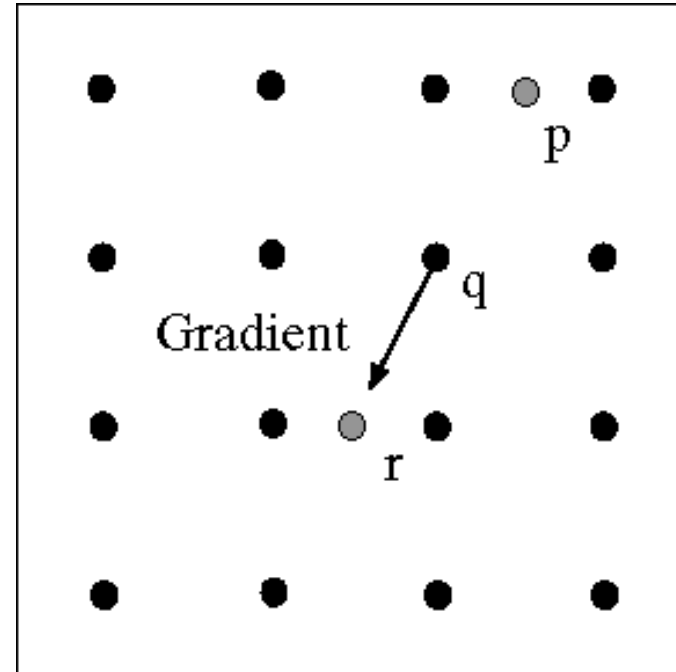
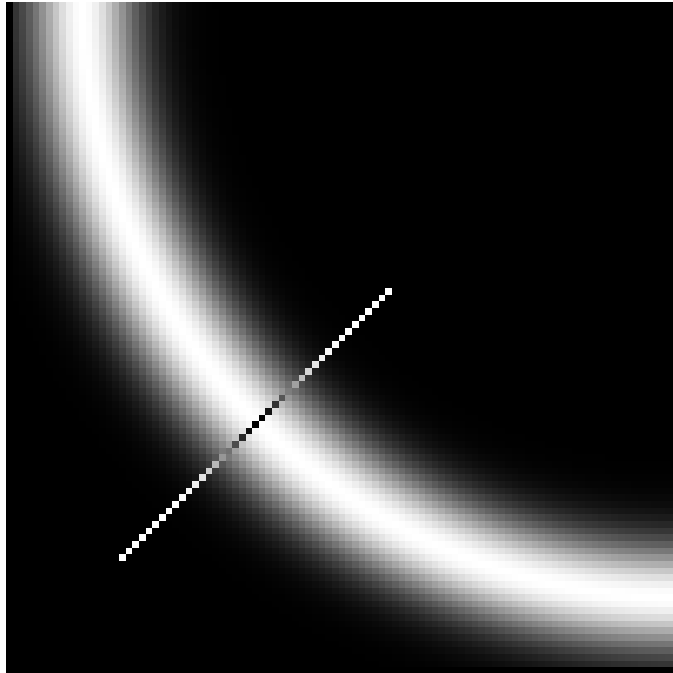
thresholding

# The Canny edge detector



How to turn these thick regions of the gradient into curves?

# Non-maximum suppression



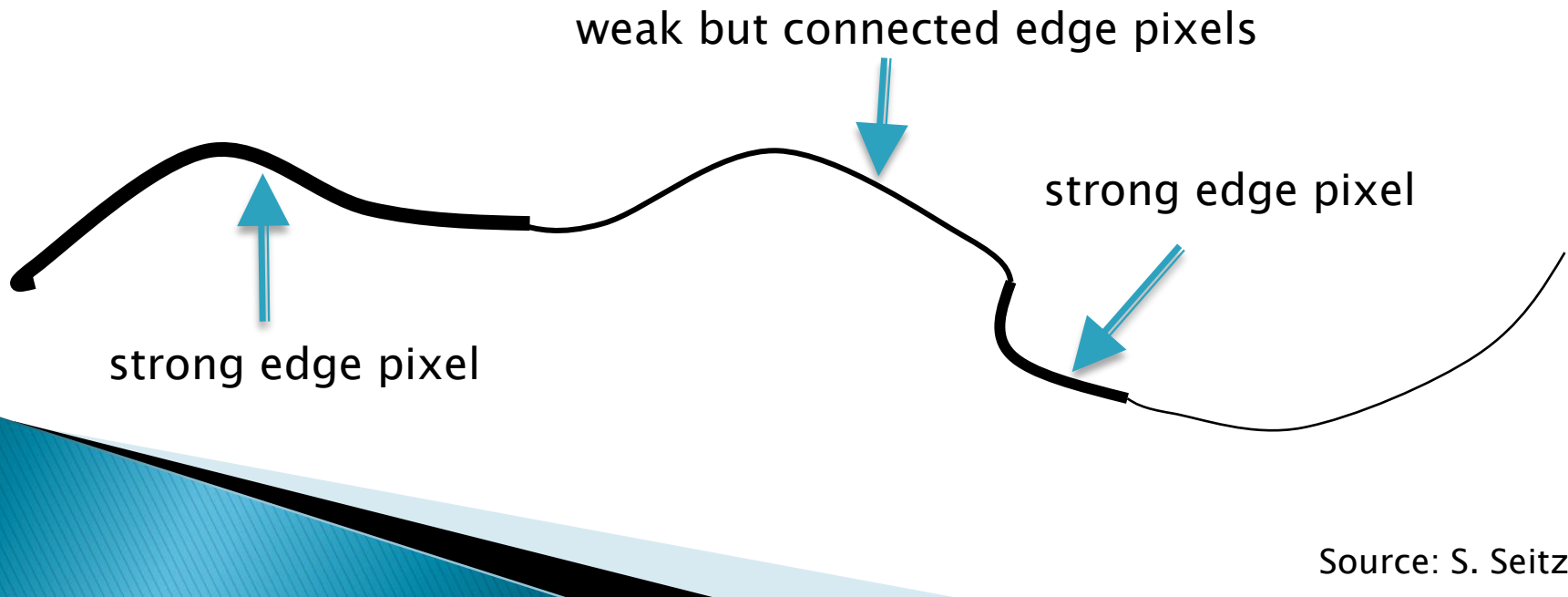
Check if pixel is local maximum along gradient direction, select single max across width of the edge

# Non-max Suppression



# Hysteresis thresholding

- ▶ Define two thresholds: Low and High
  - If less than Low, not an edge
  - If greater than High, strong edge
  - If between Low and High, weak edge





# Final Canny Edges





# Effect of $\sigma$ (Gaussian kernel spread/size)

- ▶ The choice of  $\sigma$  depends on desired behavior
  - large  $\sigma$  detects large scale edges
  - small  $\sigma$  detects fine features



original

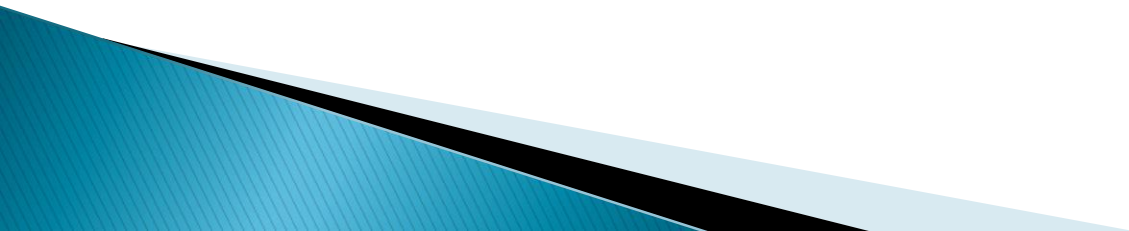


Canny with  $\sigma = 1$

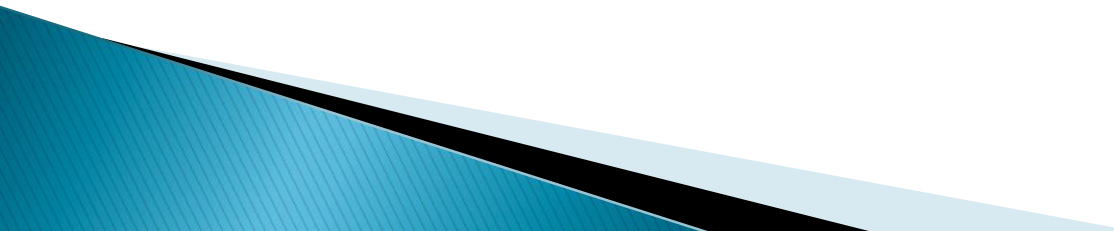


Canny with  $\sigma = 2$

# Hough transform



# Detecting lines

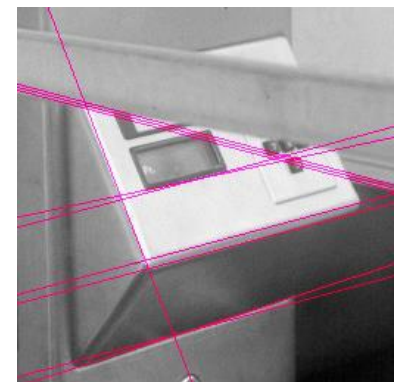
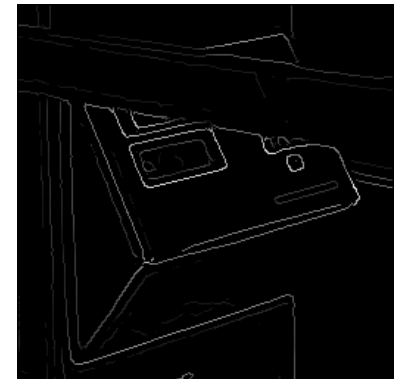
- ▶ We can perform some edge detector.
  - ▶ Then, we have a map of the edge magnitude image.
  - ▶ Some pixels in the map may describe the boundary of some objects.
  - ▶ We wish to find sets of pixels that make up straight lines.
- 

# Fitting lines: Hough transform

- ▶ Given points that belong to a line, what is the line?
- ▶ How many lines are there?
- ▶ Which points belong to which lines?
- ▶ **Hough Transform** is a voting technique that can be used to answer all of these questions.

Main idea:

1. Record vote for each possible line on which each edge point lies.
2. Look for lines that get many votes.



# Finding lines in an image: Hough space

- ▶ Transform from image  $(x,y)$  to Hough  $(a,b)$  spaces

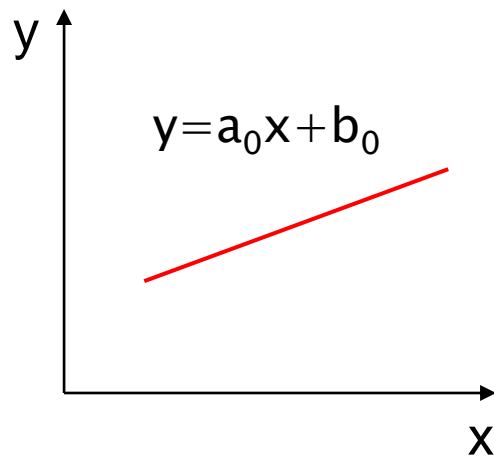
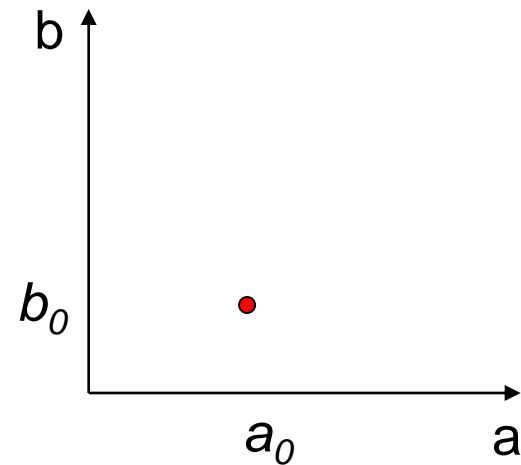


image space



Hough (parameter) space

# Finding lines in an image: Hough space

- ▶ Transform from image  $(x,y)$  to Hough  $(a,b)$  spaces

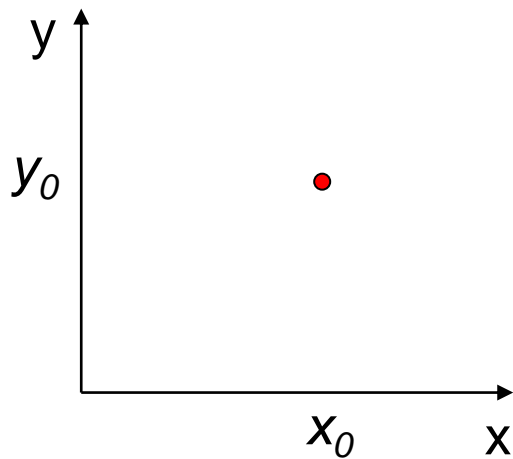
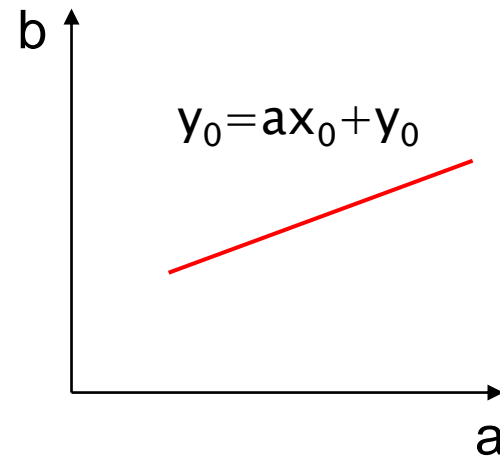


image space



Hough (parameter) space

# Finding lines in an image: Hough space

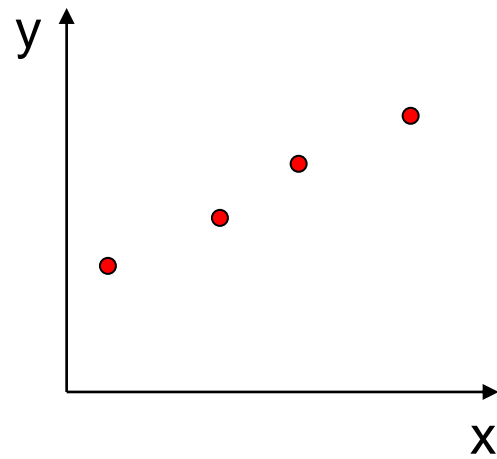
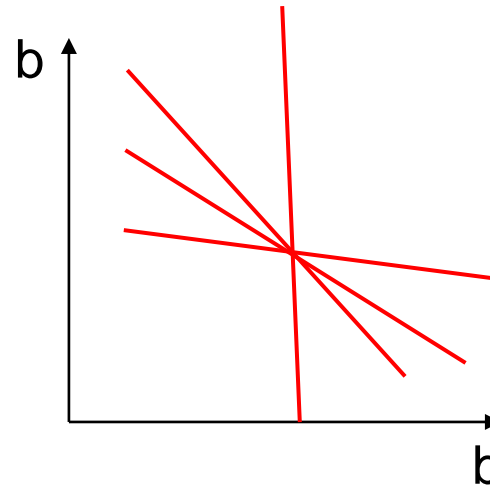


image space



Hough (parameter) space

# Example

