Edge Detection

Selim Aksoy
Department of Computer Engineering
Bilkent University
saksoy@cs.bilkent.edu.tr

Edge detection

- Edge detection is the process of finding meaningful transitions in an image.
- The points where sharp changes in the brightness occur typically form the border between different objects or scene parts.
- These points can be detected by computing intensity differences in local image regions.
- Initial stages of mammalian vision systems also involve detection of edges and local features.

Edge detection

- Sharp changes in the image brightness occur at:
  - Object boundaries
    - A light object may lie on a dark background or a dark object may lie on a light background.
  - Reflectance changes
    - May have quite different characteristics - zebras have stripes, and leopards have spots.
  - Cast shadows
  - Sharp changes in surface orientation
- Further processing of edges into lines, curves and circular arcs result in useful features for matching and recognition.

Edge detection

- Basic idea: look for a neighborhood with strong signs of change.
- Problems:
  - Neighborhood size
  - How to detect change

81	82	26	24
82	33	25	25
81	82	26	24

- Differential operators:
  - Attempt to approximate the gradient at a pixel via masks.
  - Threshold the gradient to select the edge pixels.

Difference operators for 1D

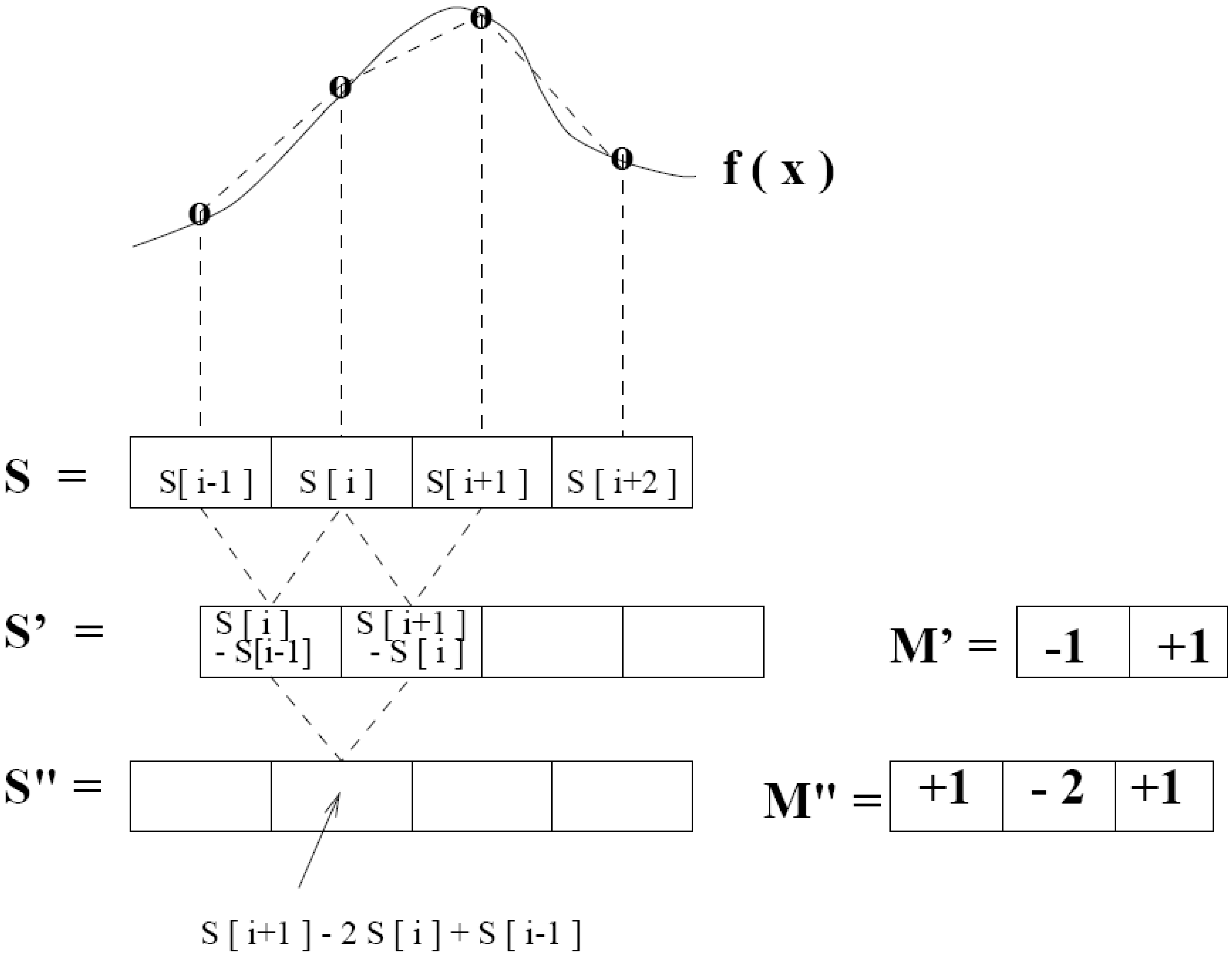


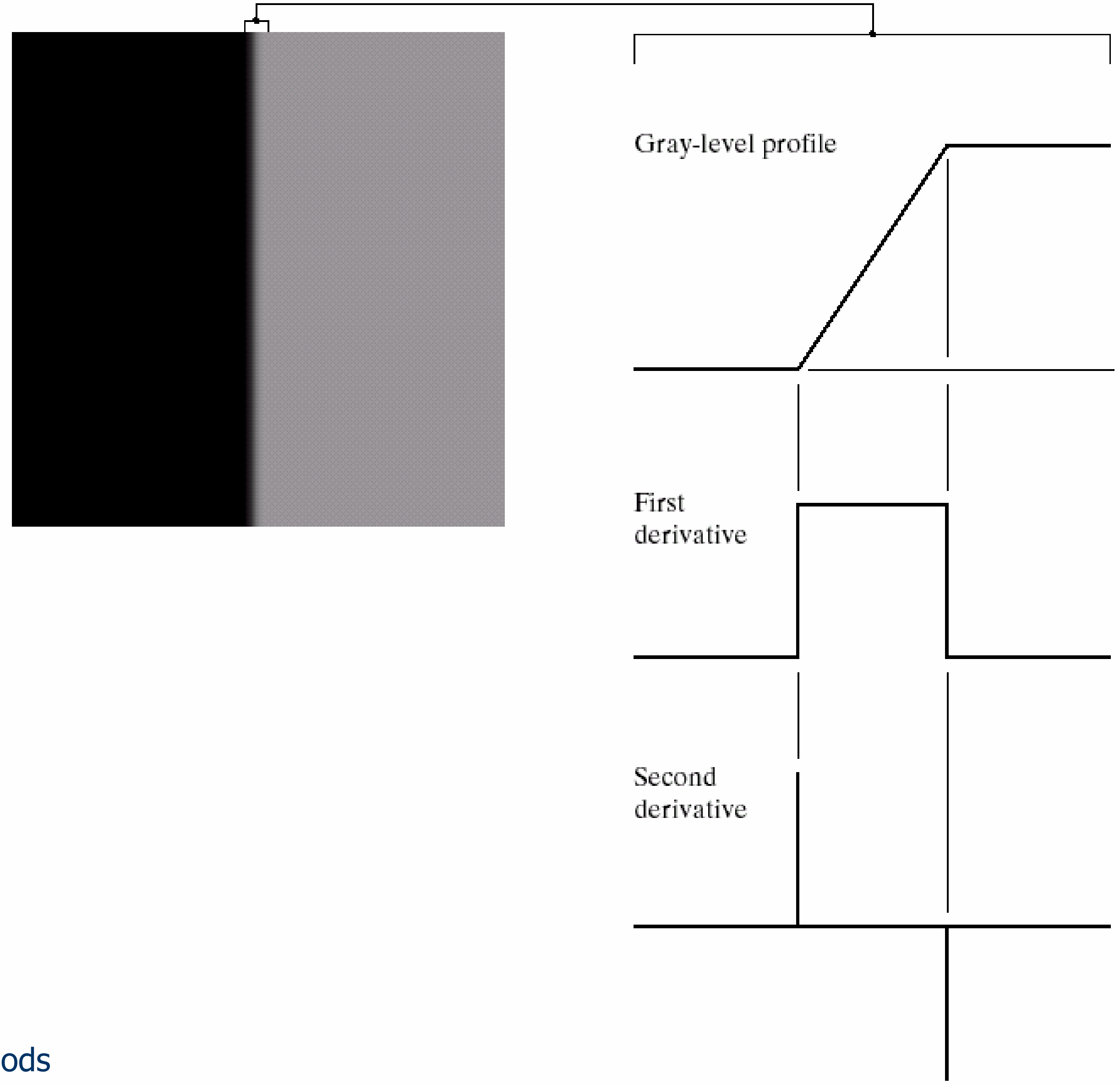
Figure 5.10: (Left) The first (**S'**) and second (**S"**) difference signals are scaled approximations to the first and second derivatives of the signal **S**. (Right) Masks **M'** and **M"** represent the derivative operations.

Adapted from Shapiro and Stockman

Difference operators for 1D

a b

FIGURE 10.6 (a) Two regions separated by a vertical edge. (b) Detail near the edge, showing a gray-level profile, and the first and second derivatives of the profile.



Adapted from Gonzales and Woods

Difference operators for 1D

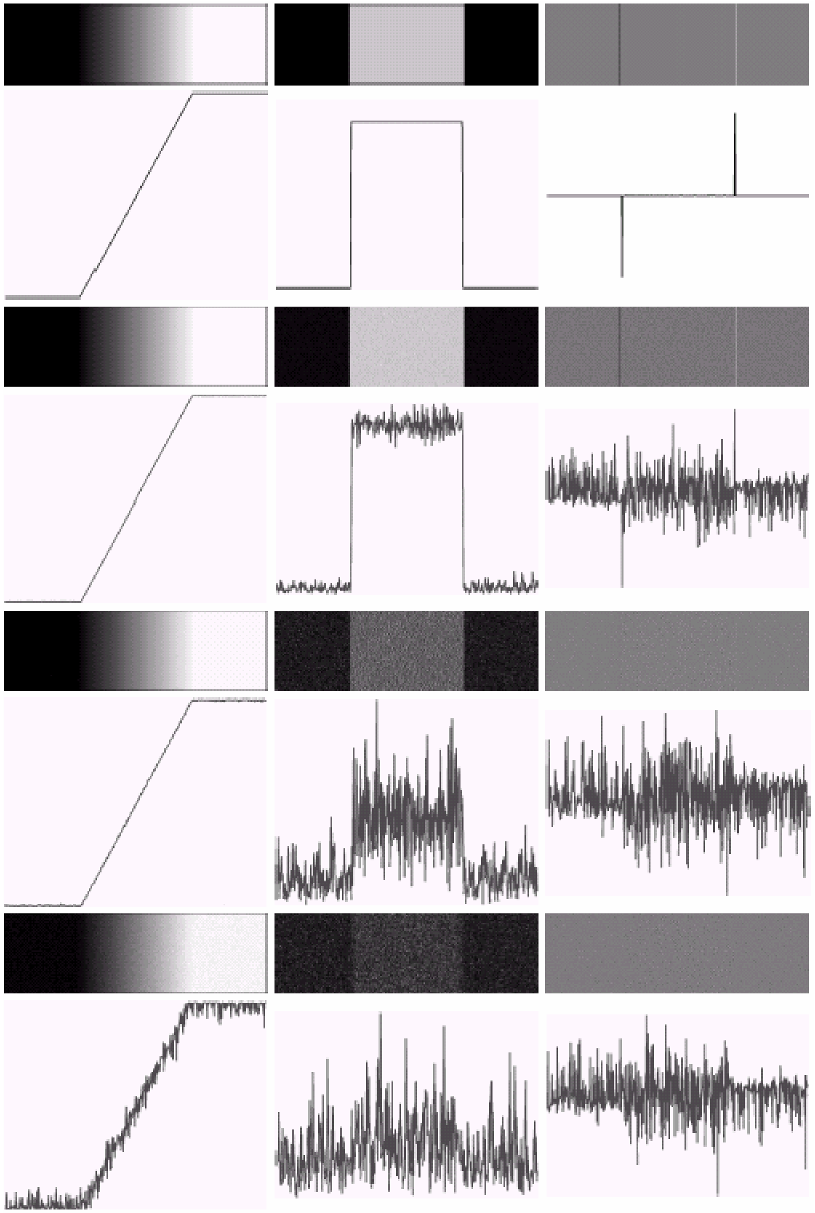
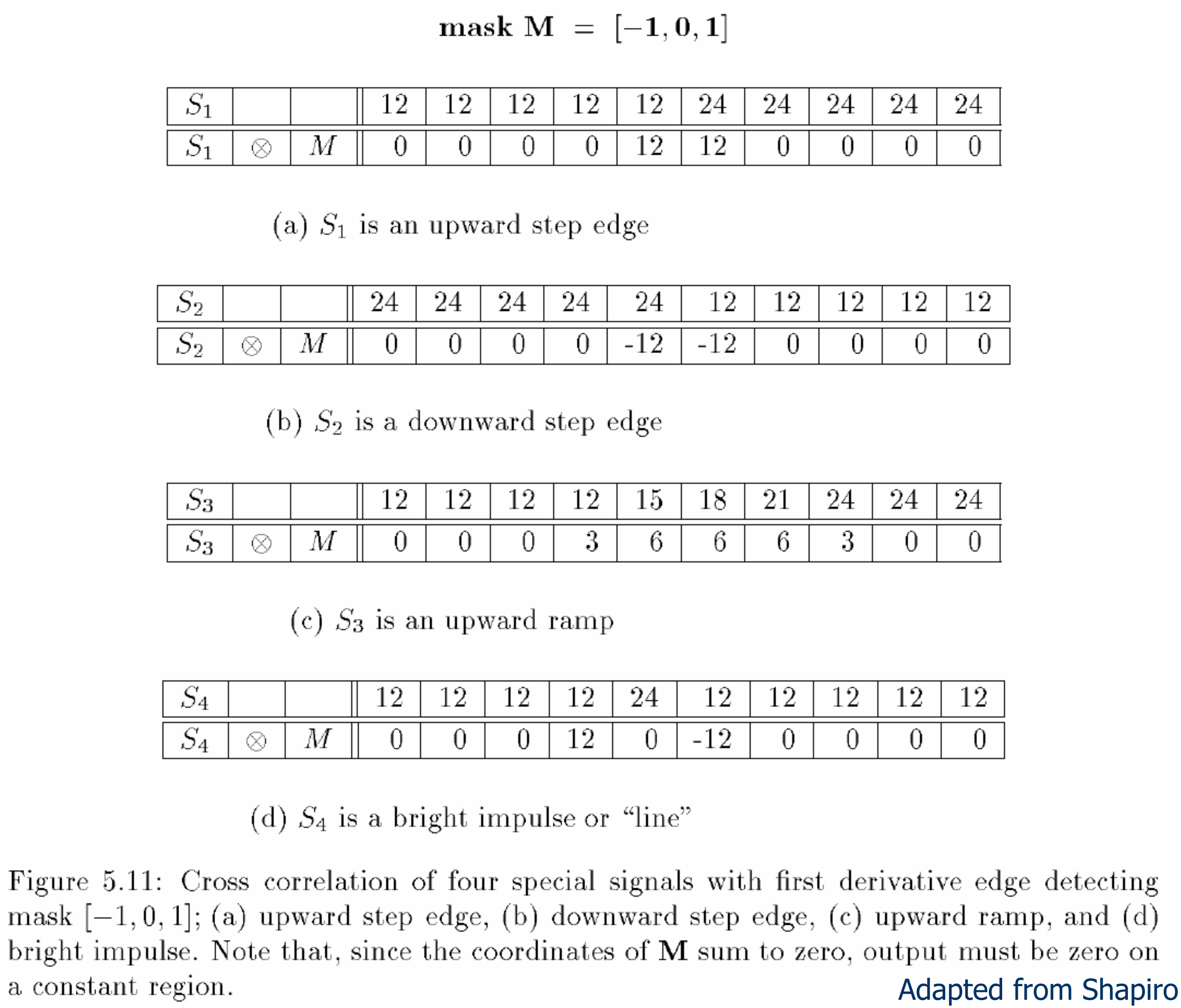


FIGURE 10.7 First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and $\sigma = 0.0, 0.1, 1.0$, and 10.0, respectively. Second column: first-derivative images and gray-level profiles. Third column: second-derivative images and gray-level profiles.

Adapted from Gonzales and Woods

Difference operators for 1D



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

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

(a) S_1 is an upward 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

(b) S_2 is a downward step edge

S_3			12	12	12	12	15	18	21	24	24	24
S_3	$\otimes$	M	0	0	0	3	6	6	6	3	0	0

(c) S_3 is an upward ramp

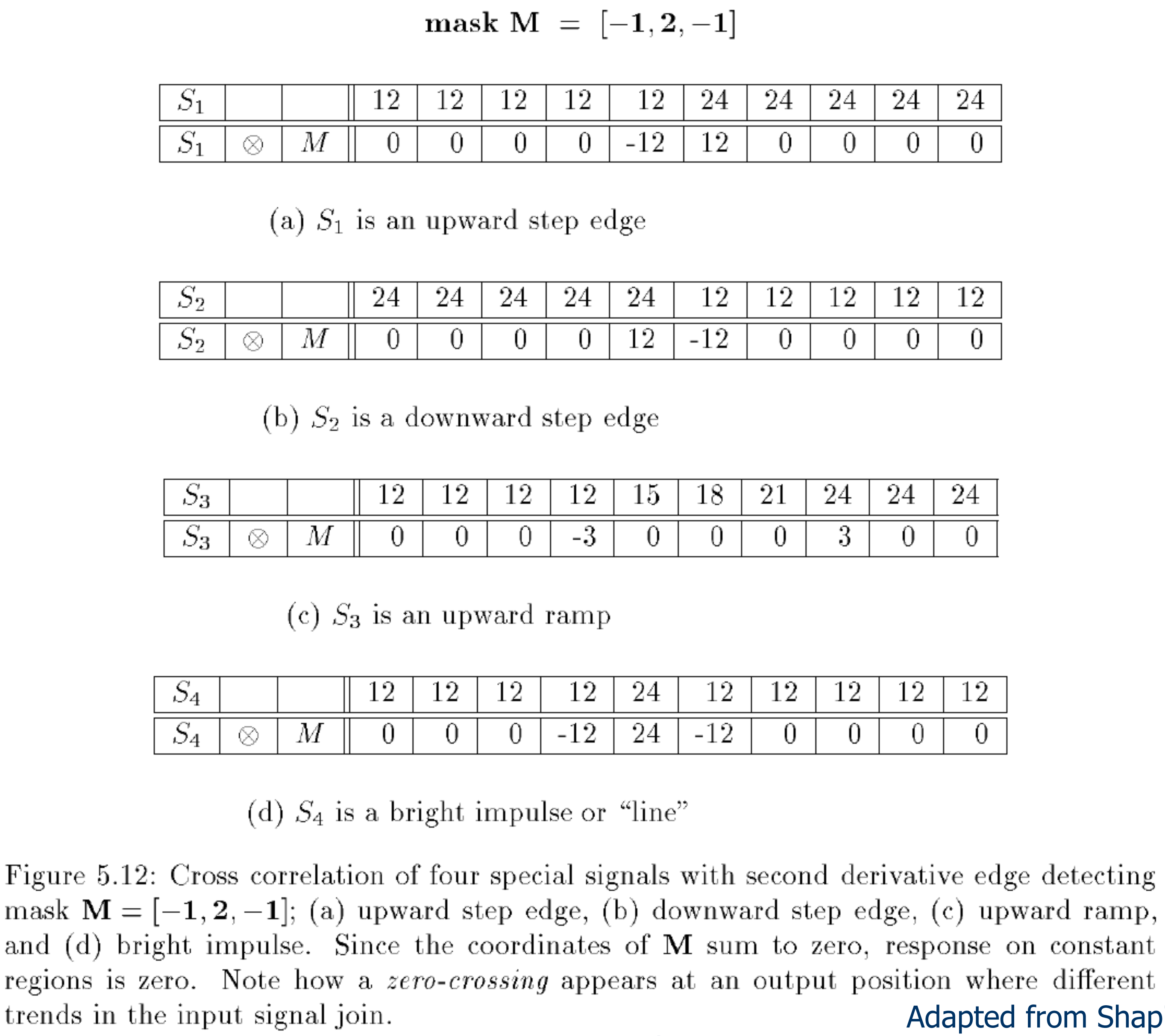
S_4			12	12	12	12	24	12	12	12	12	12
S_4	$\otimes$	M	0	0	0	12	0	-12	0	0	0	0

(d) S_4 is a bright impulse or “line”

Figure 5.11: Cross correlation of four special signals with first derivative edge detecting mask $[-1, 0, 1]$; (a) upward step edge, (b) downward step edge, (c) upward ramp, and (d) bright impulse. Note that, since the coordinates of $\mathbf{M}$ sum to zero, output must be zero on a constant region.

Adapted from Shapiro and Stockman

Difference operators for 1D



mask M $= [-\mathbf{1}, \mathbf{2}, -\mathbf{1}]$

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

(a) S_1 is an upward 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

(b) S_2 is a downward step edge

S_3			12	12	12	12	15	18	21	24	24	24
S_3	$\otimes$	M	0	0	0	-3	0	0	0	3	0	0

(c) S_3 is an upward ramp

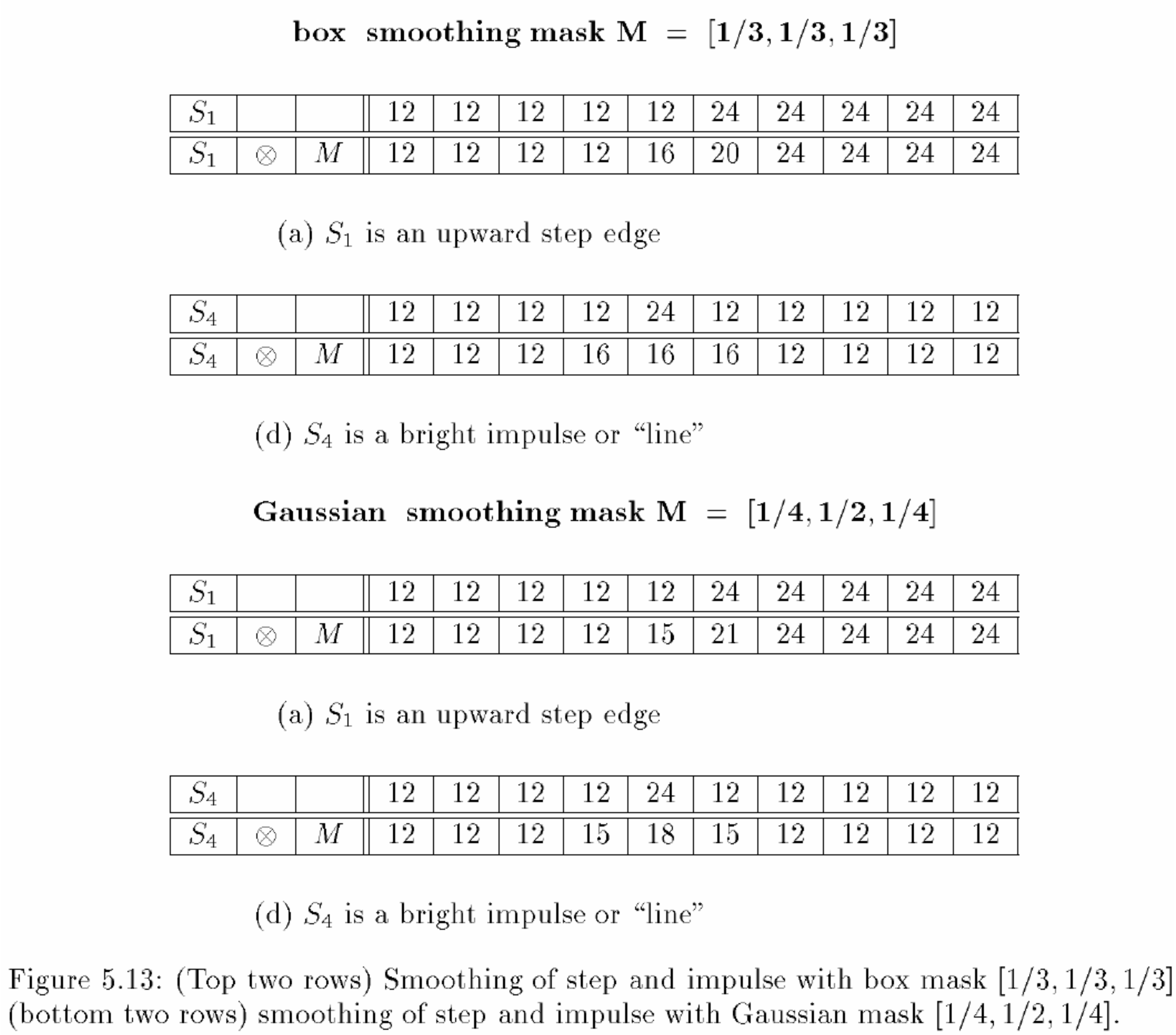
S_4			12	12	12	12	24	12	12	12	12	12
S_4	$\otimes$	M	0	0	0	-12	24	-12	0	0	0	0

(d) S_4 is a bright impulse or "line"

Figure 5.12: Cross correlation of four special signals with second derivative edge detecting mask $\mathbf{M} = [-\mathbf{1}, \mathbf{2}, -\mathbf{1}]$; (a) upward step edge, (b) downward step edge, (c) upward ramp, and (d) bright impulse. Since the coordinates of $\mathbf{M}$ sum to zero, response on constant regions is zero. Note how a *zero-crossing* appears at an output position where different trends in the input signal join.

Adapted from Shapiro and Stockman

Difference operators for 1D



box smoothing mask M $= [\mathbf{1/3, 1/3, 1/3}]$

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	12	12	12	12	16	20	24	24	24	24

(a) S_1 is an upward step edge

S_4			12	12	12	12	24	12	12	12	12	12
S_4	$\otimes$	M	12	12	12	16	16	16	12	12	12	12

(d) S_4 is a bright impulse or "line"

Gaussian smoothing mask M $= [\mathbf{1/4, 1/2, 1/4}]$

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	12	12	12	12	15	21	24	24	24	24

(a) S_1 is an upward step edge

S_4			12	12	12	12	24	12	12	12	12	12
S_4	$\otimes$	M	12	12	12	15	18	15	12	12	12	12

(d) S_4 is a bright impulse or "line"

Figure 5.13: (Top two rows) Smoothing of step and impulse with box mask $[1/3, 1/3, 1/3]$ (bottom two rows) smoothing of step and impulse with Gaussian mask $[1/4, 1/2, 1/4]$.

Observations

- Properties of derivative masks:
  - Coordinates of derivative masks have opposite signs in order to obtain a high response in signal regions of high contrast.
  - The sum of coordinates of derivative masks is zero so that a zero response is obtained on constant regions.
  - First derivative masks produce high absolute values at points of high contrast.
  - Second derivative masks produce zero-crossings at points of high contrast.

Observations

- Properties of smoothing masks:
  - Coordinates of smoothing masks are positive and sum to one so that output on constant regions is the same as the input.
  - The amount of smoothing and noise reduction is proportional to the mask size.
  - Step edges are blurred in proportion to the mask size.

Difference operators for 2D

- Contrast in the 2D picture function $f(x,y)$ can occur in any direction.
- From calculus, we know that the maximum change occurs along the direction of the *gradient*.
- The gradient of an image $f(x,y)$ at location (x,y) is defined as the vector

$$\nabla f = \left[\frac{\partial f}{\partial x} \; \frac{\partial f}{\partial y} \right]^T.$$

Difference operators for 2D

- The *magnitude of the gradient*

$$|\nabla f| = \left(\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right)^{1/2}$$

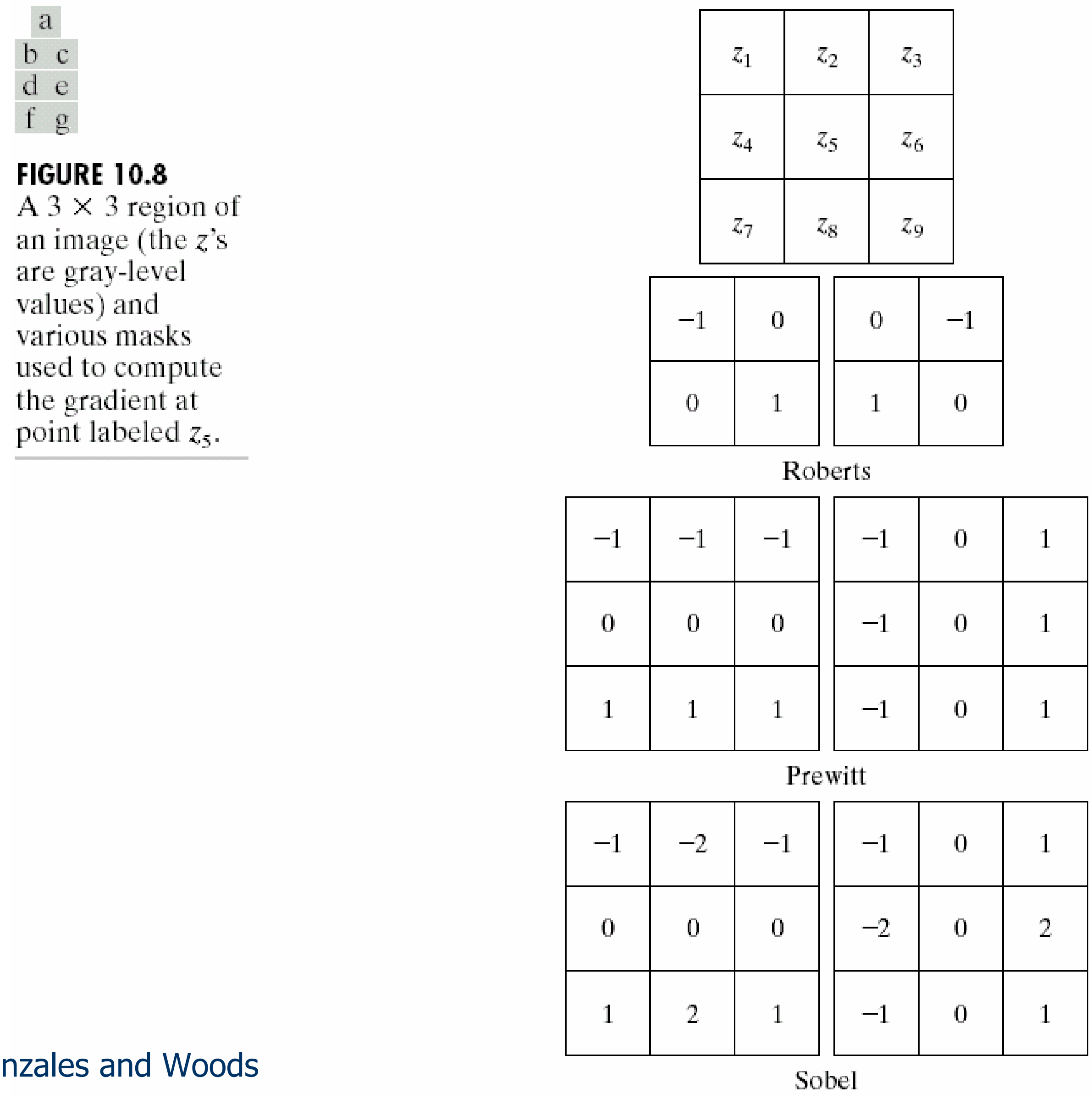
gives the maximum rate of increase of $f(x, y)$ per unit distance in the direction of ∇f.

- The *direction of the gradient*

$$\measuredangle(\nabla f) = \tan^{-1} \left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x} \right)$$

represents the direction of this change with respect to the x-axis.

Difference operators for 2D



a

b c

d e

f g

FIGURE 10.8 A 3 × 3 region of an image (the z's are gray-level values) and various masks used to compute the gradient at point labeled z_5.

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
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

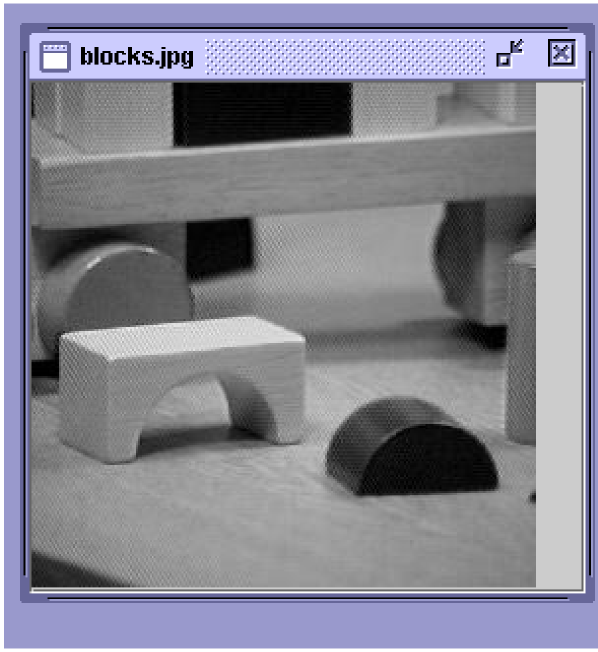
−1	−2	−1
0	0	0
1	2	1

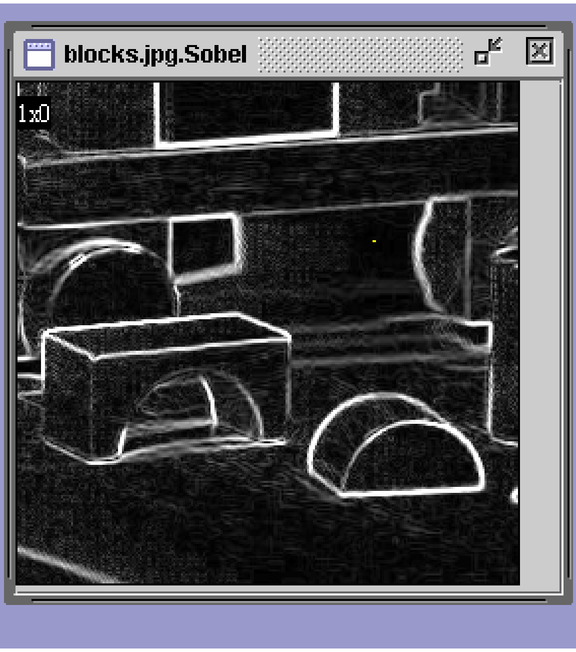
−1	0	1
−2	0	2
−1	0	1

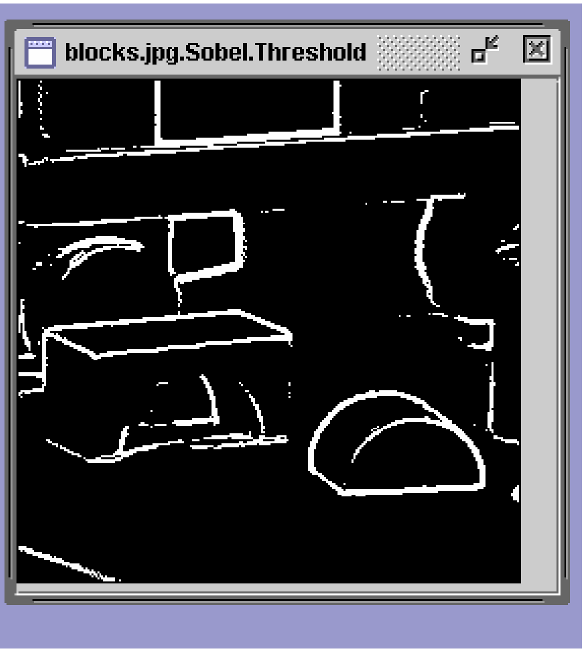
Sobel

Adapted from Gonzales and Woods

Difference operators for 2D







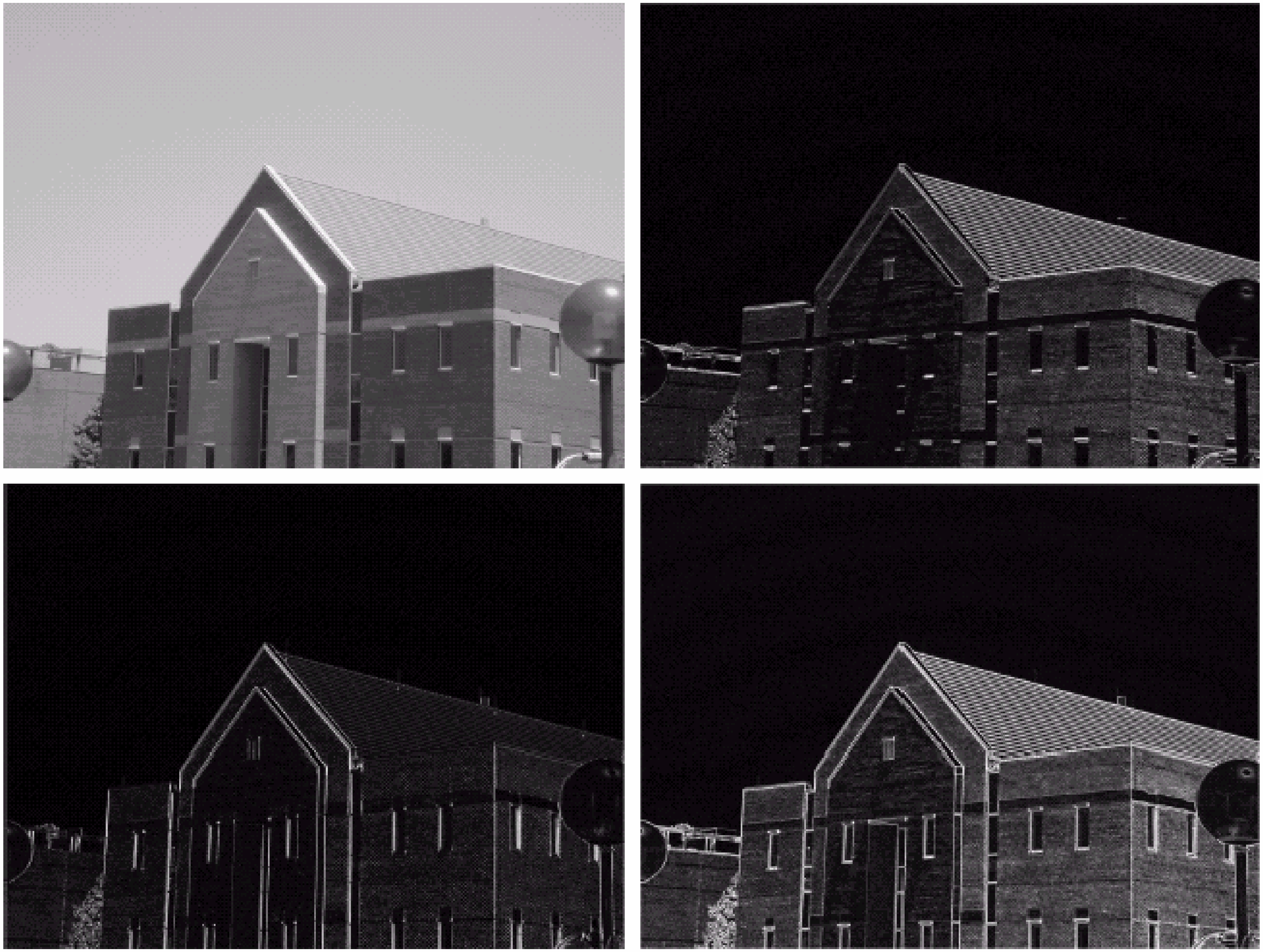
original image

gradient magnitude

thresholded gradient magnitude

Adapted from Linda Shapiro, U of Washington

Difference operators for 2D



a b
c d

FIGURE 10.10 (a) Original image. (b) $|G_x|$, component of the gradient in the x-direction. (c) $|G_y|$, component in the y-direction. (d) Gradient image, $|G_x| + |G_y|$.

Difference operators for 2D

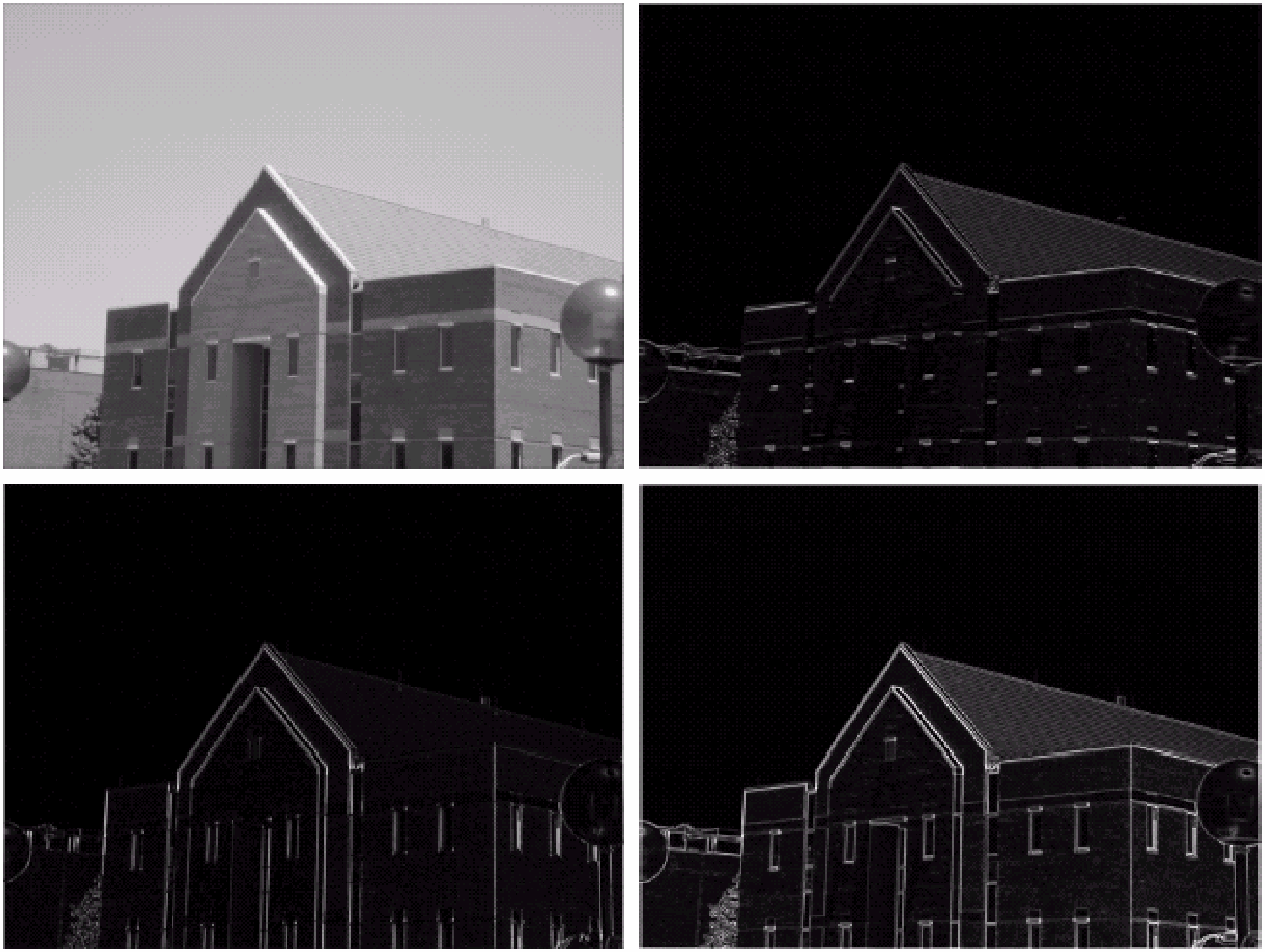
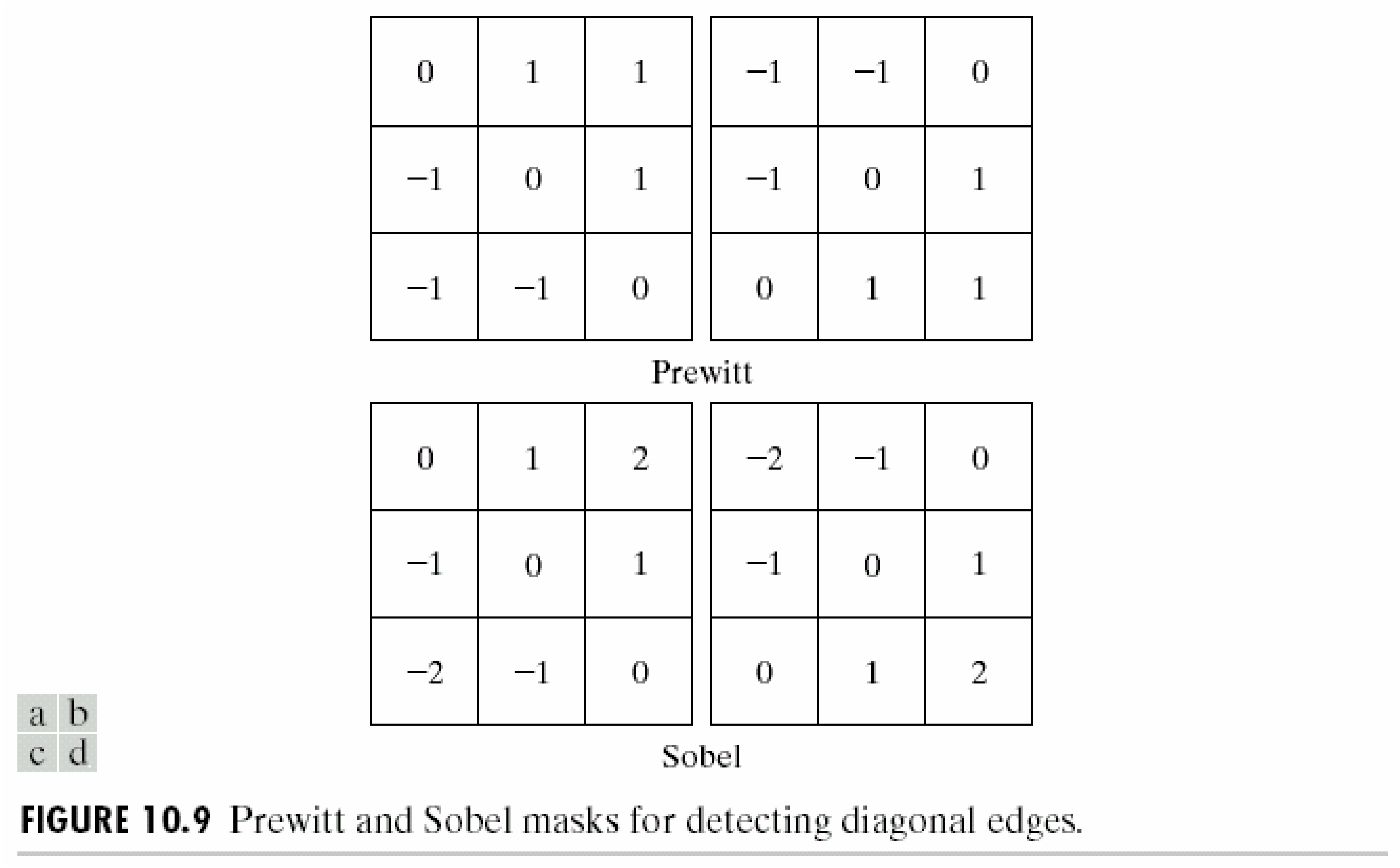


FIGURE 10.11 Same sequence as in Fig. 10.10, but with the original image smoothed with a 5 × 5 averaging filter.

Difference operators for 2D



0	1	1
−1	0	1
−1	−1	0

−1	−1	0
−1	0	1
0	1	1

Prewitt

0	1	2
−1	0	1
−2	−1	0

−2	−1	0
−1	0	1
0	1	2

Sobel

a b
c d

FIGURE 10.9 Prewitt and Sobel masks for detecting diagonal edges.

Adapted from Gonzales and Woods

Difference operators for 2D

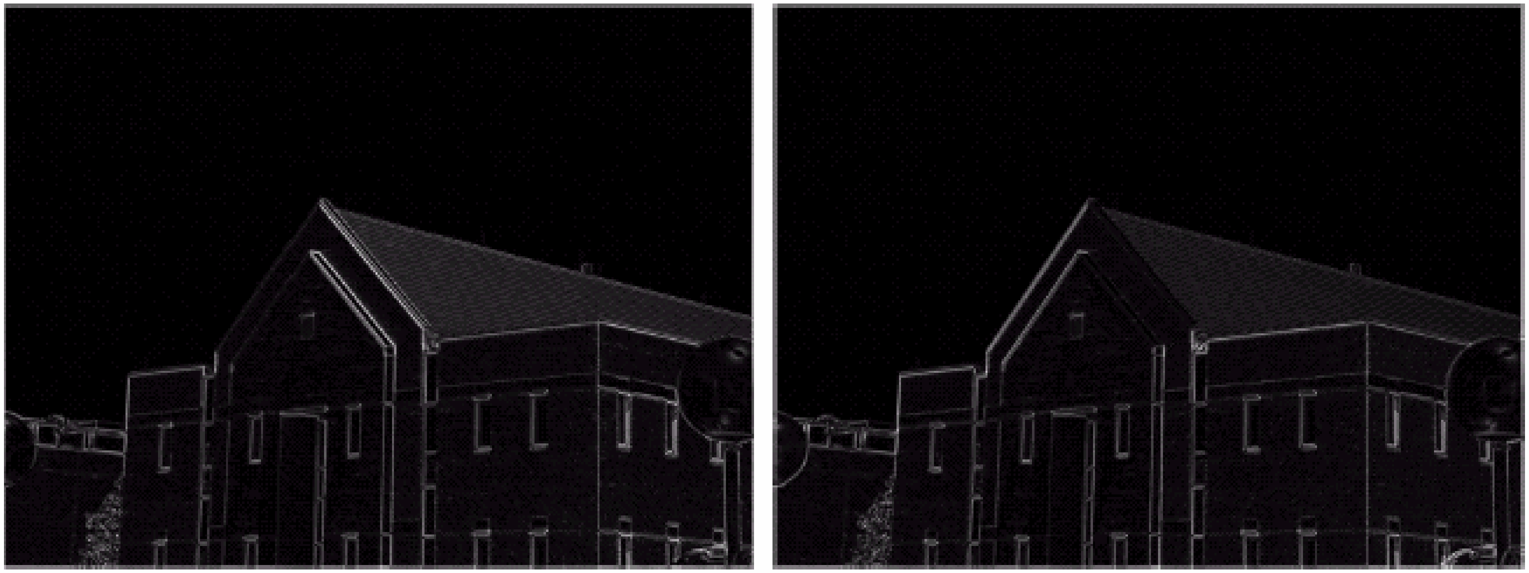


FIGURE 10.12 Diagonal edge detection. (a) Result of using the mask in Fig. 10.9(c). (b) Result of using the mask in Fig. 10.9(d). The input in both cases was Fig. 10.11(a).

Adapted from Gonzales and Woods

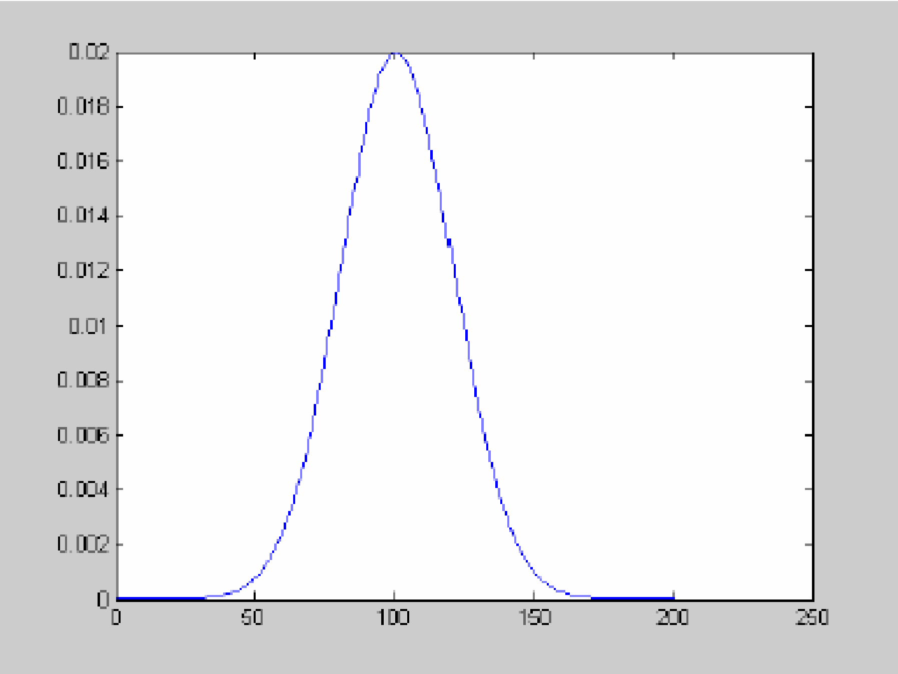
Gaussian smoothing and edge detection

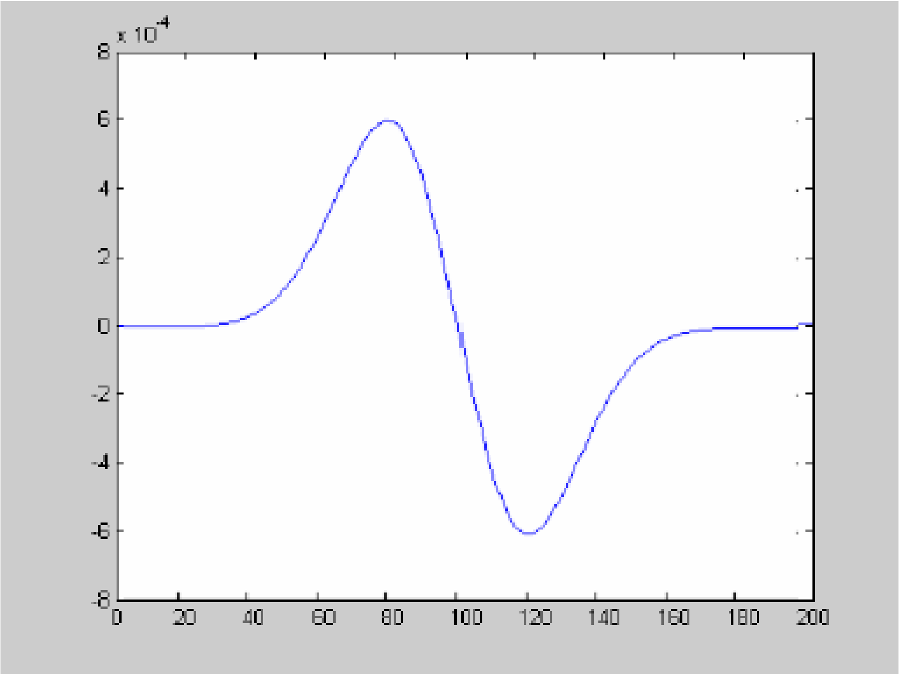
- We can smooth the image using a Gaussian filter and then compute the derivative.
- Two convolutions: one to smooth, then another one to differentiate?

  → Actually, no - we can use a derivative of Gaussian filter because differentiation is convolution and convolution is associative.

Derivative of Gaussian

$$D_x \otimes (G \otimes I) = (D_x \otimes G) \otimes I$$

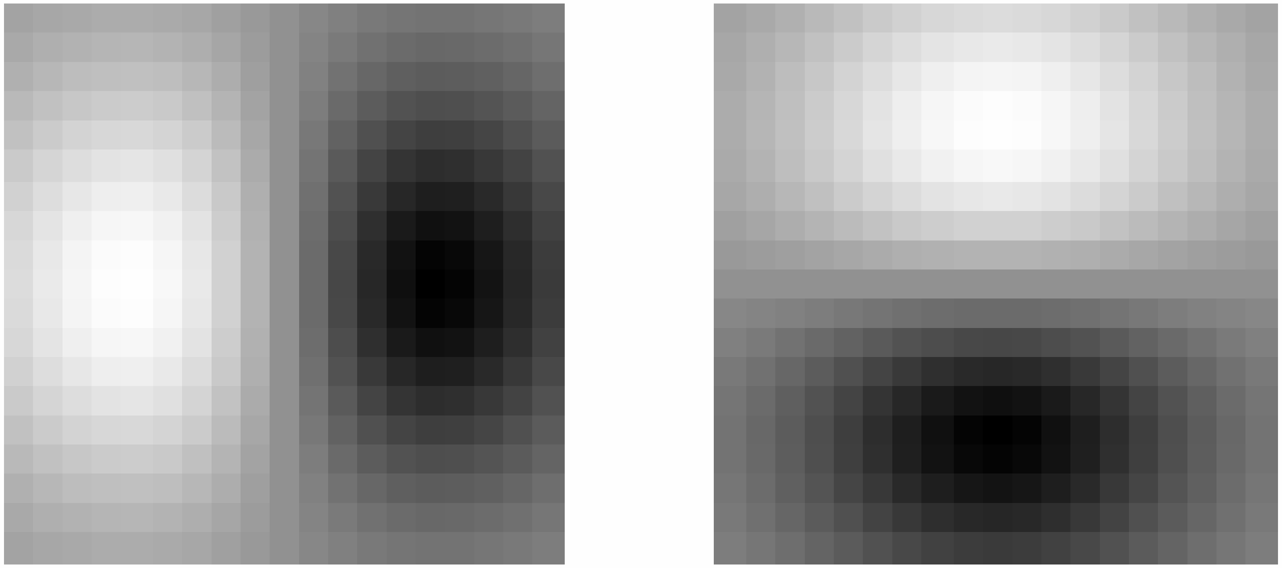




Adapted from Michael Black, Brown University

Derivative of Gaussian

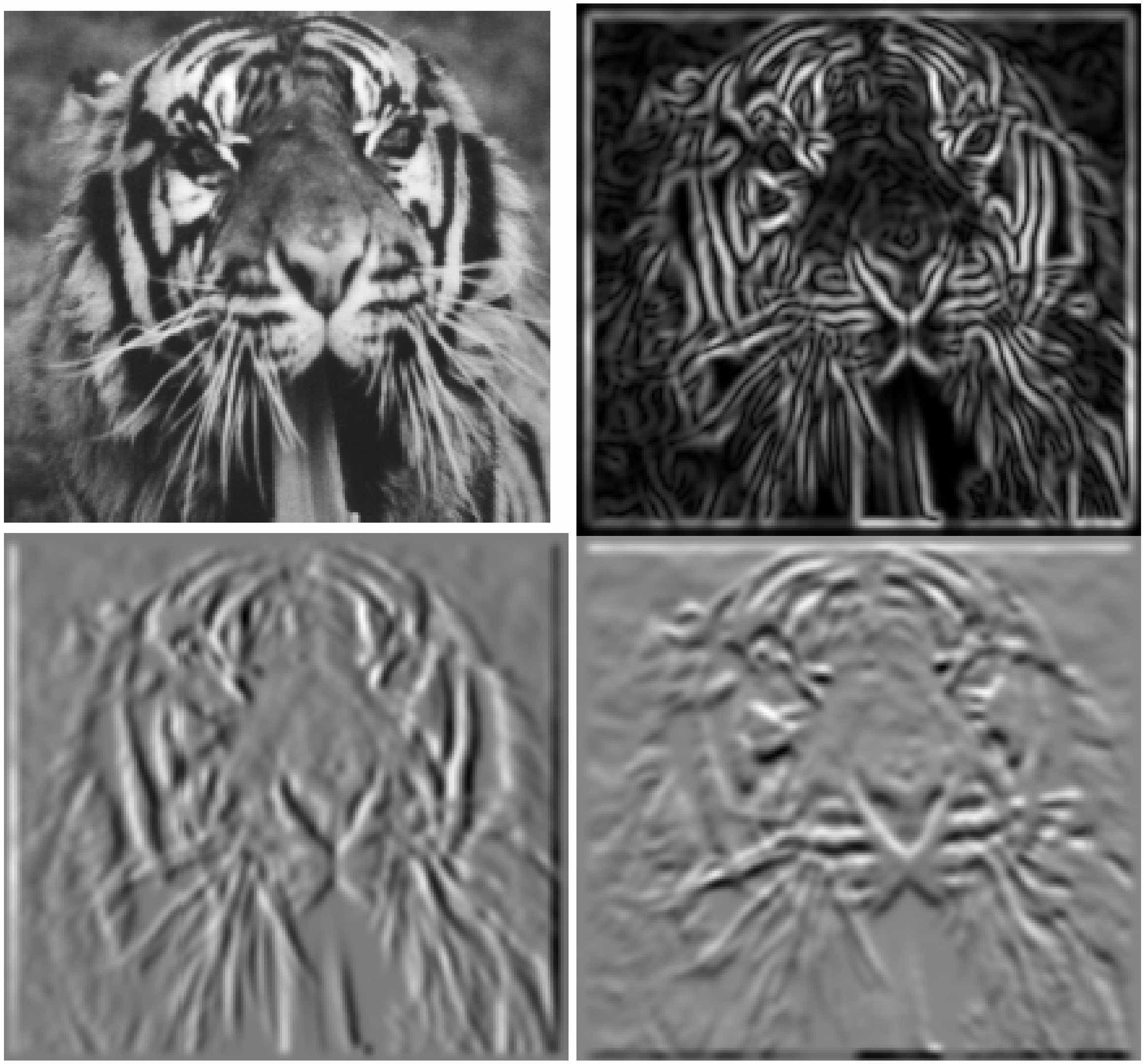
$$D_x \otimes (G \otimes I) = (D_x \otimes G) \otimes I$$



Adapted from Michael Black, Brown University

Derivative of Gaussian

Applying the first derivative of Gaussian



Adapted from Martial Hebert, CMU

Difference operators for 2D

- The *Laplacian* of a 2D function $f(x, y)$ is a second-order derivative defined as

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

- The Laplacian generally is not used in its original form for edge detection because:
  - It is sensitive to noise.
  - Its magnitude produces double edges.
  - It is unable to detect edge direction.
- However, its zero-crossing property can be used for edge localization.

Gaussian smoothing and edge detection

- 1D Gaussian function:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}.$$

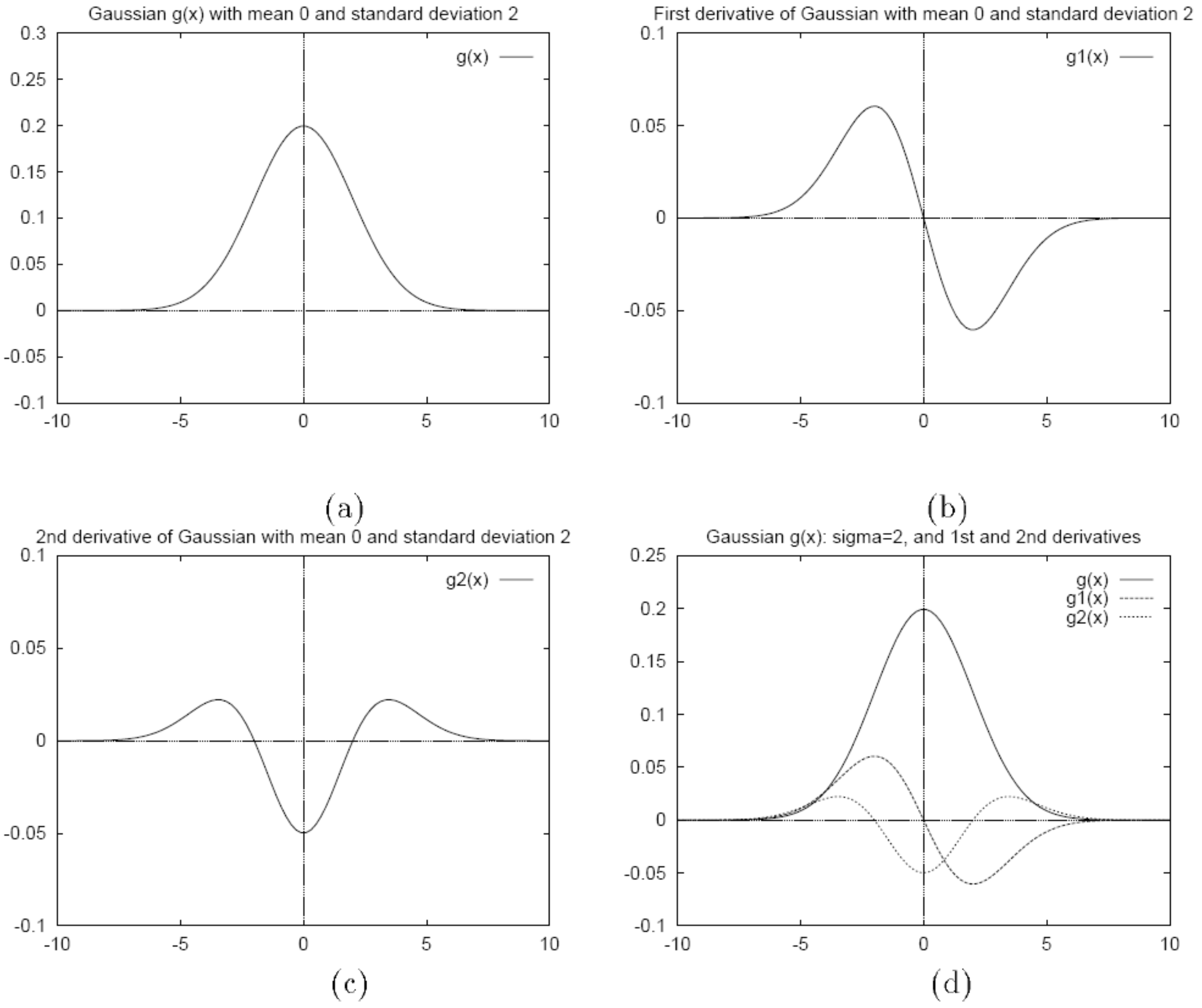
- *Derivative of Gaussian (DoG)*:

$$g'(x) = \frac{-x}{\sigma^2} g(x).$$

- *Laplacian of Gaussian (LoG)*:

$$g''(x) = \left(\frac{x^2}{\sigma^4} - \frac{-1}{\sigma^2} \right) g(x).$$

Gaussian smoothing and edge detection



(a)

(b)

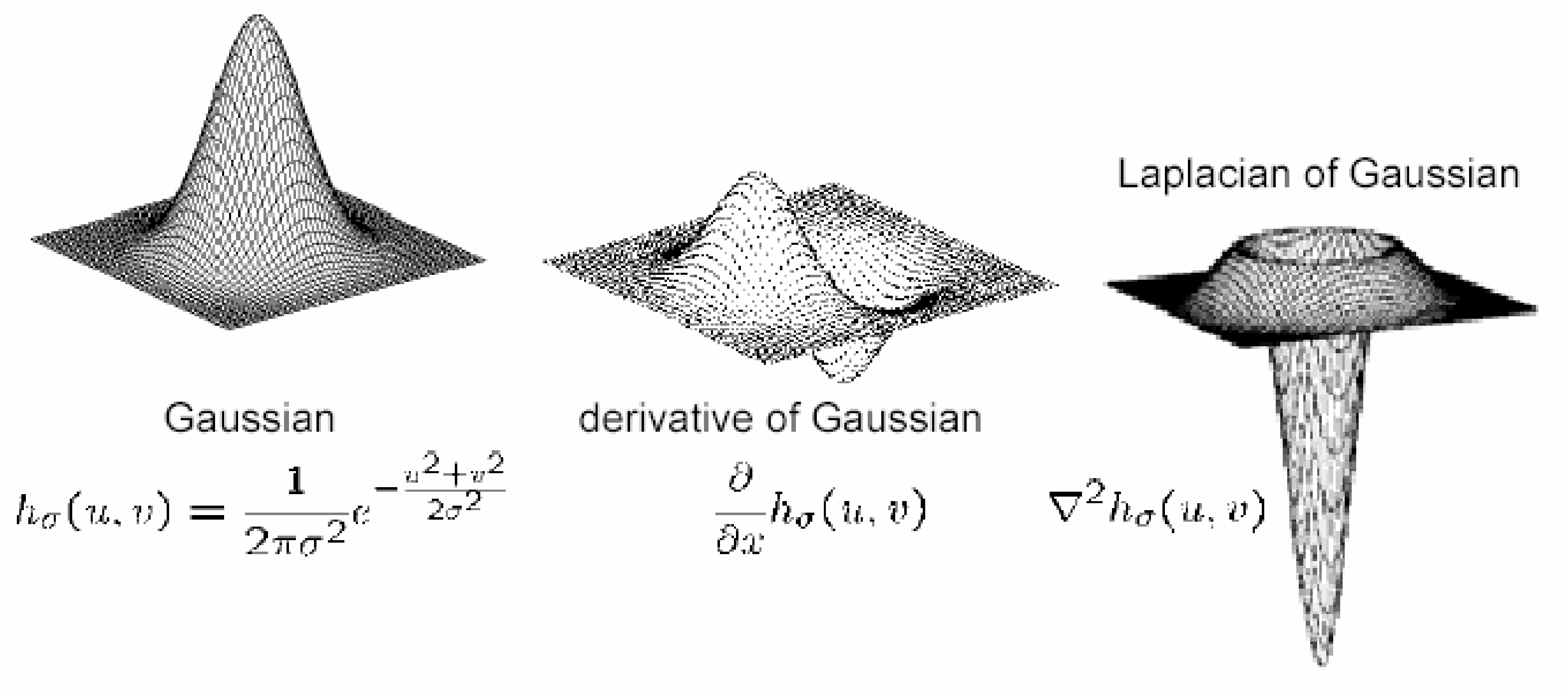
(c)

(d)

Figure 5.19: (a) Gaussian $g(x)$ with spread $\sigma = 2$; (b) first derivative $g'(x)$; (c) second derivative $g''(x)$, which looks like the cross section of a sombrero upside down from how it would be worn; (d) all three plots superimposed to show how the extreme slopes of $g(x)$ align with the extremas of $g'(x)$ and the zero crossings of $g''(x)$.

Adapted from Shapiro and Stockman

Gaussian smoothing and edge detection



Gaussian

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

derivative of Gaussian

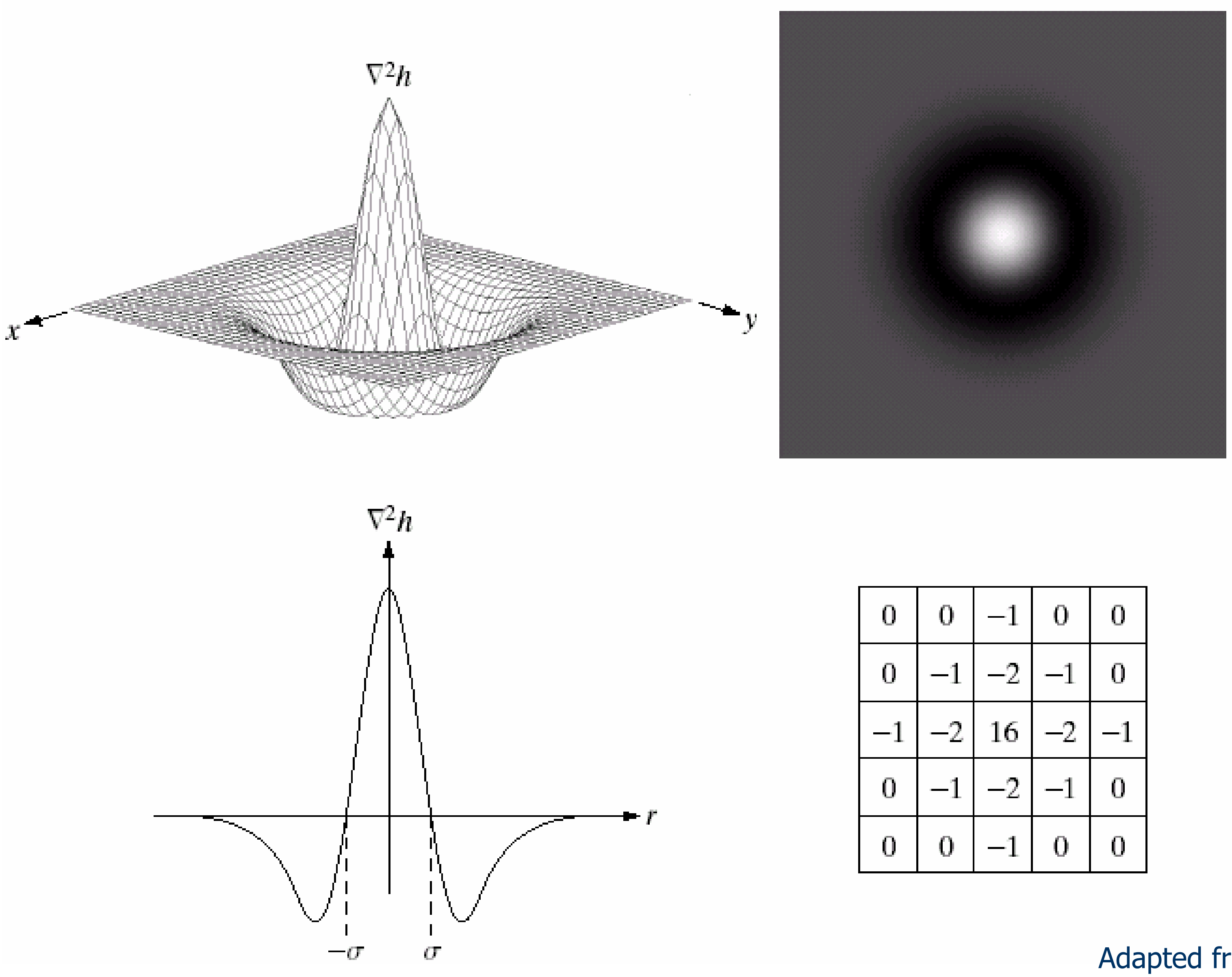
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian

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

Adapted from Steve Seitz, U of Washington

Laplacian of Gaussian



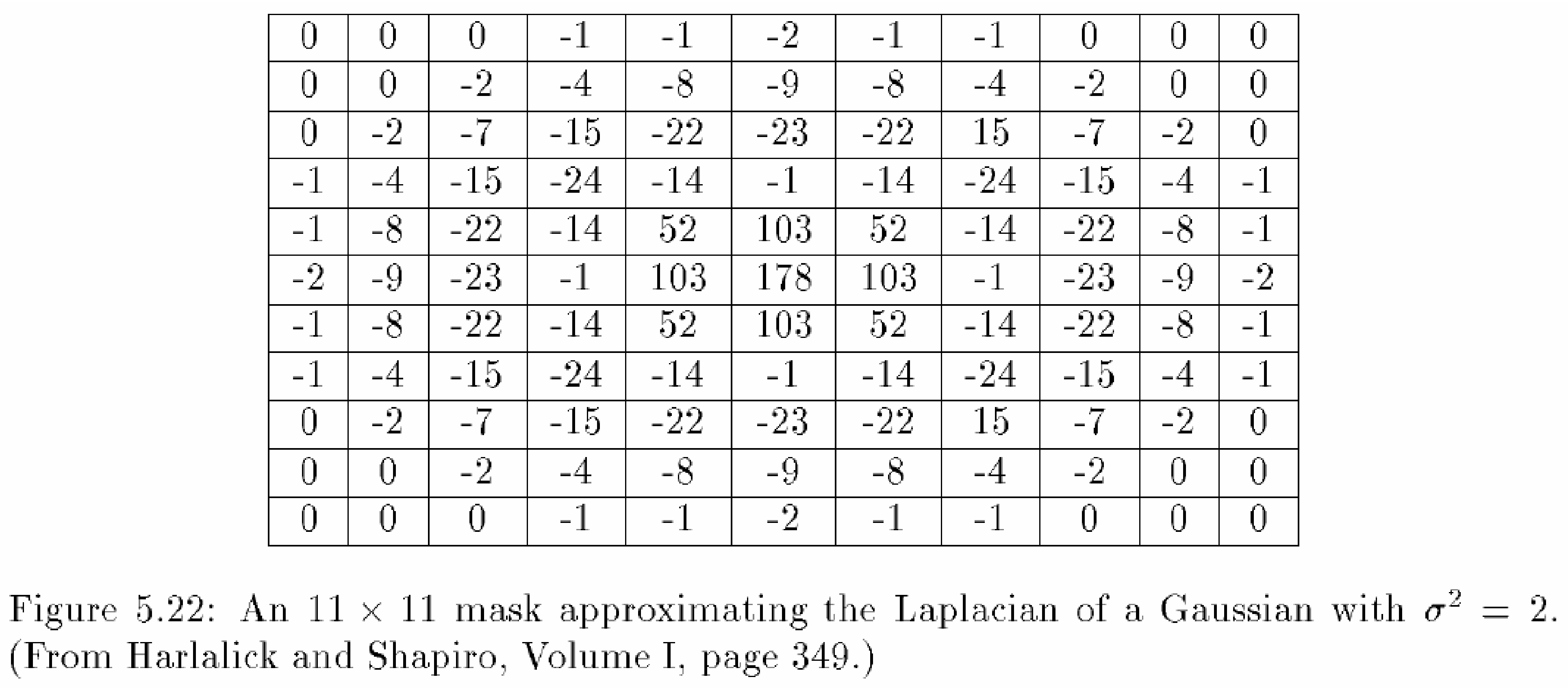
0	0	–1	0	0
0	–1	–2	–1	0
–1	–2	16	–2	–1
0	–1	–2	–1	0
0	0	–1	0	0

a b
c d

FIGURE 10.14 Laplacian of a Gaussian (LoG). (a) 3-D plot. (b) Image (black is negative, gray is the zero plane, and white is positive). (c) Cross section showing zero crossings. (d) 5 × 5 mask approximation to the shape of (a).

Adapted from Gonzales and Woods

Laplacian of Gaussian

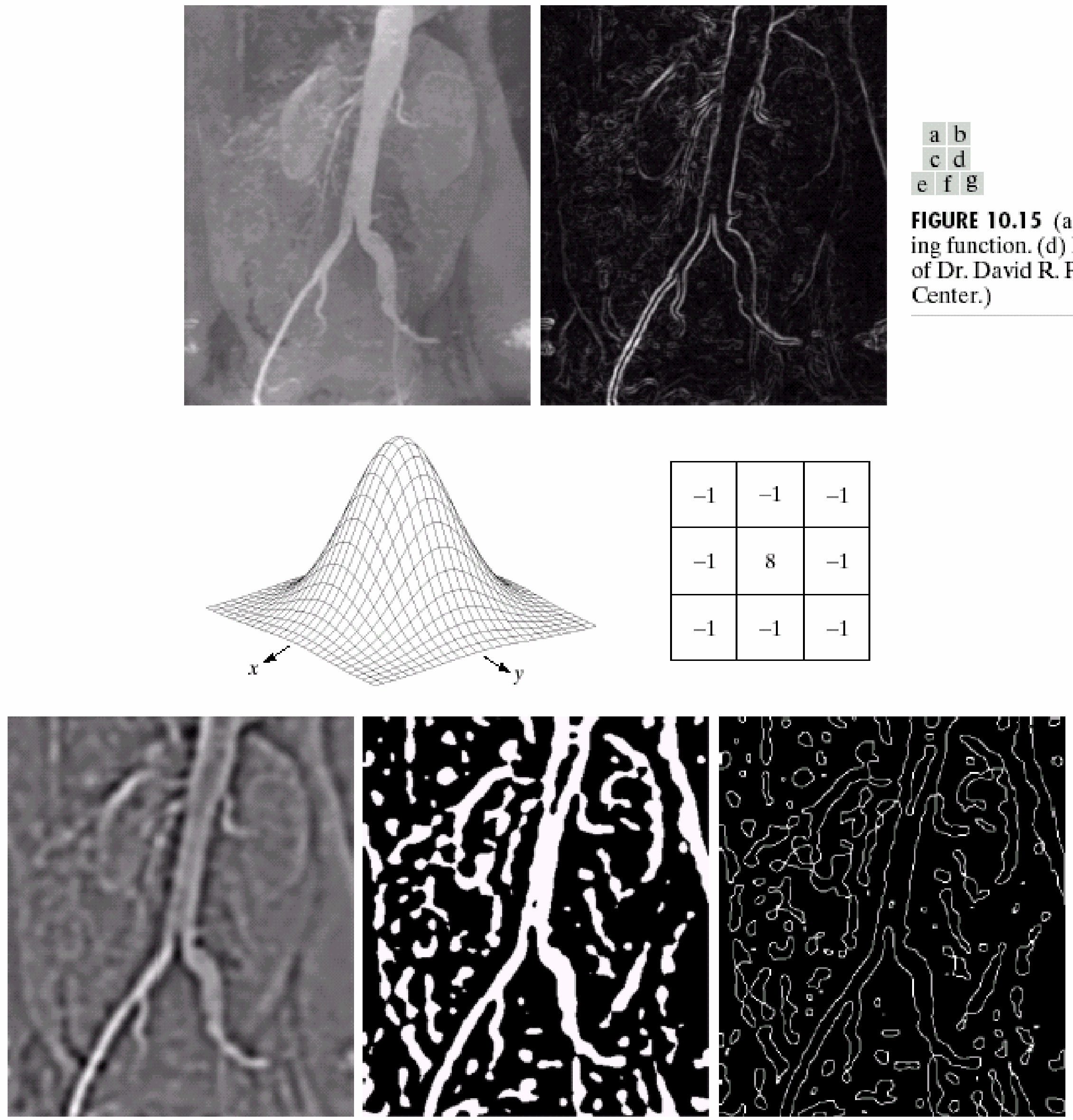


0	0	0	-1	-1	-2	-1	-1	0	0	0
0	0	-2	-4	-8	-9	-8	-4	-2	0	0
0	-2	-7	-15	-22	-23	-22	15	-7	-2	0
-1	-4	-15	-24	-14	-1	-14	-24	-15	-4	-1
-1	-8	-22	-14	52	103	52	-14	-22	-8	-1
-2	-9	-23	-1	103	178	103	-1	-23	-9	-2
-1	-8	-22	-14	52	103	52	-14	-22	-8	-1
-1	-4	-15	-24	-14	-1	-14	-24	-15	-4	-1
0	-2	-7	-15	-22	-23	-22	15	-7	-2	0
0	0	-2	-4	-8	-9	-8	-4	-2	0	0
0	0	0	-1	-1	-2	-1	-1	0	0	0

Figure 5.22: An 11 × 11 mask approximating the Laplacian of a Gaussian with $\sigma^2 = 2$. (From Harlalick and Shapiro, Volume I, page 349.)

Adapted from Shapiro and Stockman

Laplacian of Gaussian



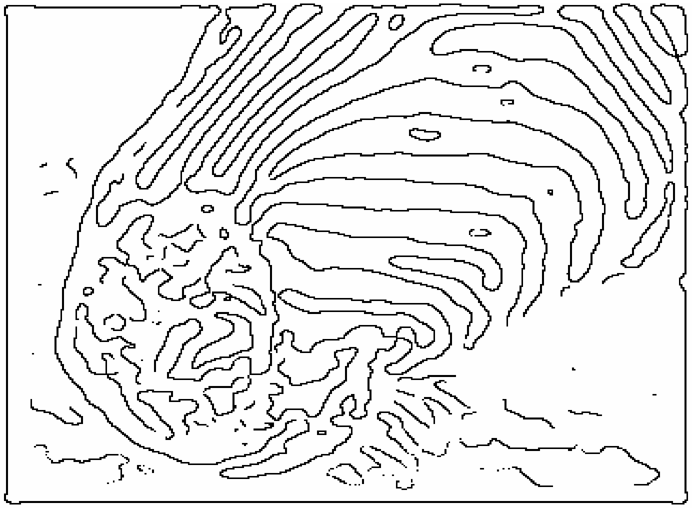
–1	–1	–1
–1	8	–1
–1	–1	–1

a b
c d
e f g

FIGURE 10.15 (a) Original image. (b) Sobel gradient (shown for comparison). (c) Spatial Gaussian smoothing function. (d) Laplacian mask. (e) LoG. (f) Thresholded LoG. (g) Zero crossings. (Original image courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

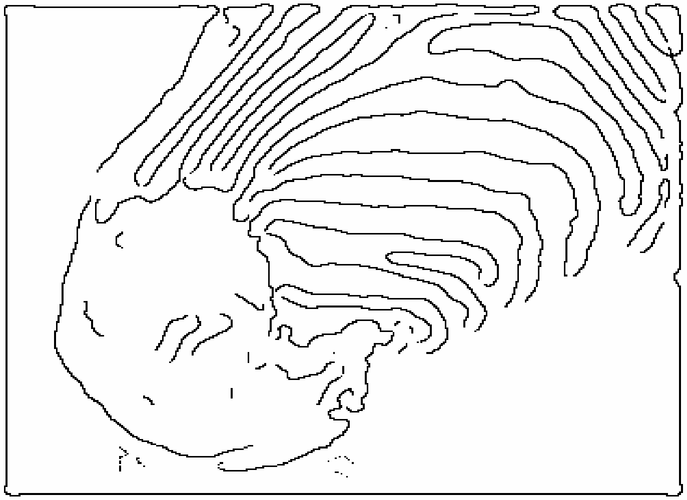
Adapted from Gonzales and Woods

Laplacian of Gaussian



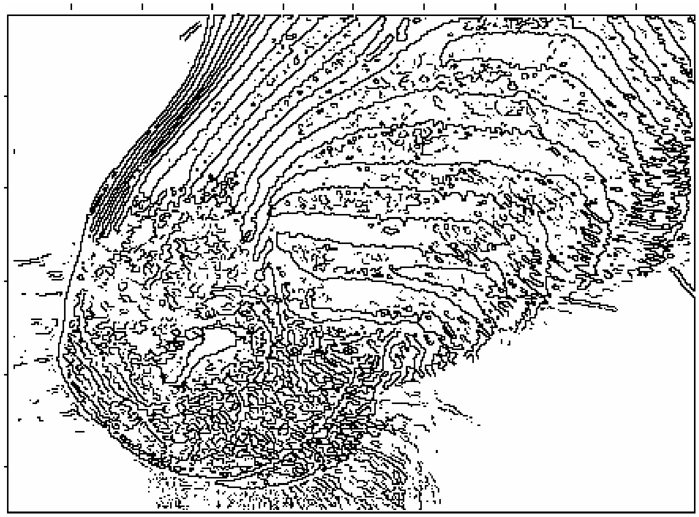
sigma=4

LoG zero crossings

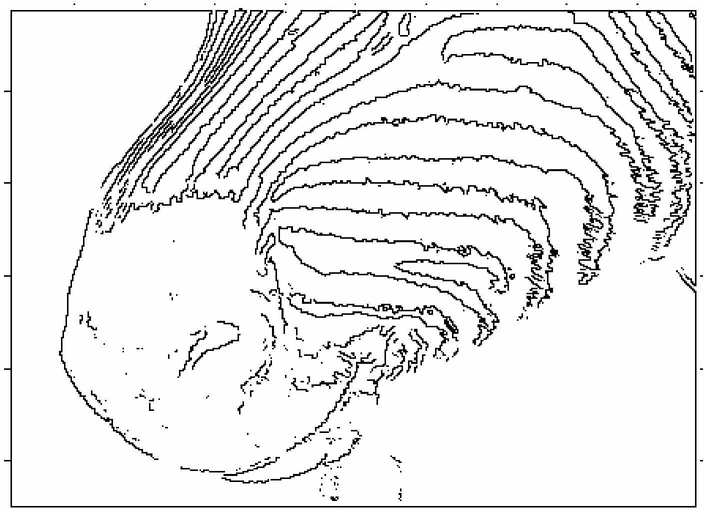


Gradient threshold=1

Gradient threshold=4



sigma=2



Adapted from David Forsyth, UC Berkeley

Marr/Hildreth edge detector

1. First smooth the image via a Gaussian convolution.
2. Apply a Laplacian filter (estimate 2nd derivative).
3. Find zero crossings of the Laplacian of the Gaussian.

This can be done at multiple resolutions.

Haralick edge detector

1. Fit the gray-tone intensity surface to a piecewise cubic polynomial approximation.
2. Use the approximation to find zero crossings of the second directional derivative in the direction that maximizes the first directional derivative.

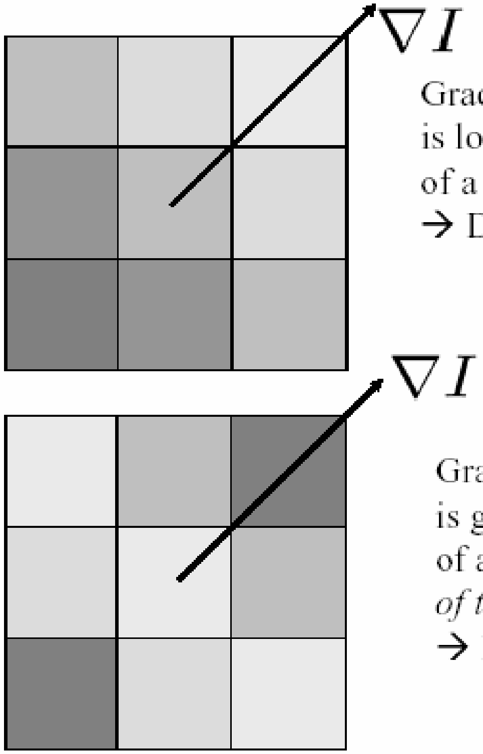
The derivatives here are calculated from direct mathematical expressions with respect to the cubic polynomial.

Canny edge detector

1. Smooth the image with a Gaussian filter with spread σ.
2. Compute gradient magnitude and direction at each pixel of the smoothed image.
3. Zero out any pixel response less than or equal to the two neighboring pixels on either side of it, along the direction of the gradient (non-maxima suppression).
4. Track high-magnitude contours using thresholding (hysteresis thresholding).
5. Keep only pixels along these contours, so weak little segments go away.

Canny edge detector

- Goal: good localization and good detection.
- Non-maxima suppression:
  - Gradient direction is used to thin edges by suppressing any pixel response that is not higher than the two neighboring pixels on either side of it along the direction of the gradient.
  - This operation can be used with any edge operator when thin boundaries are wanted.



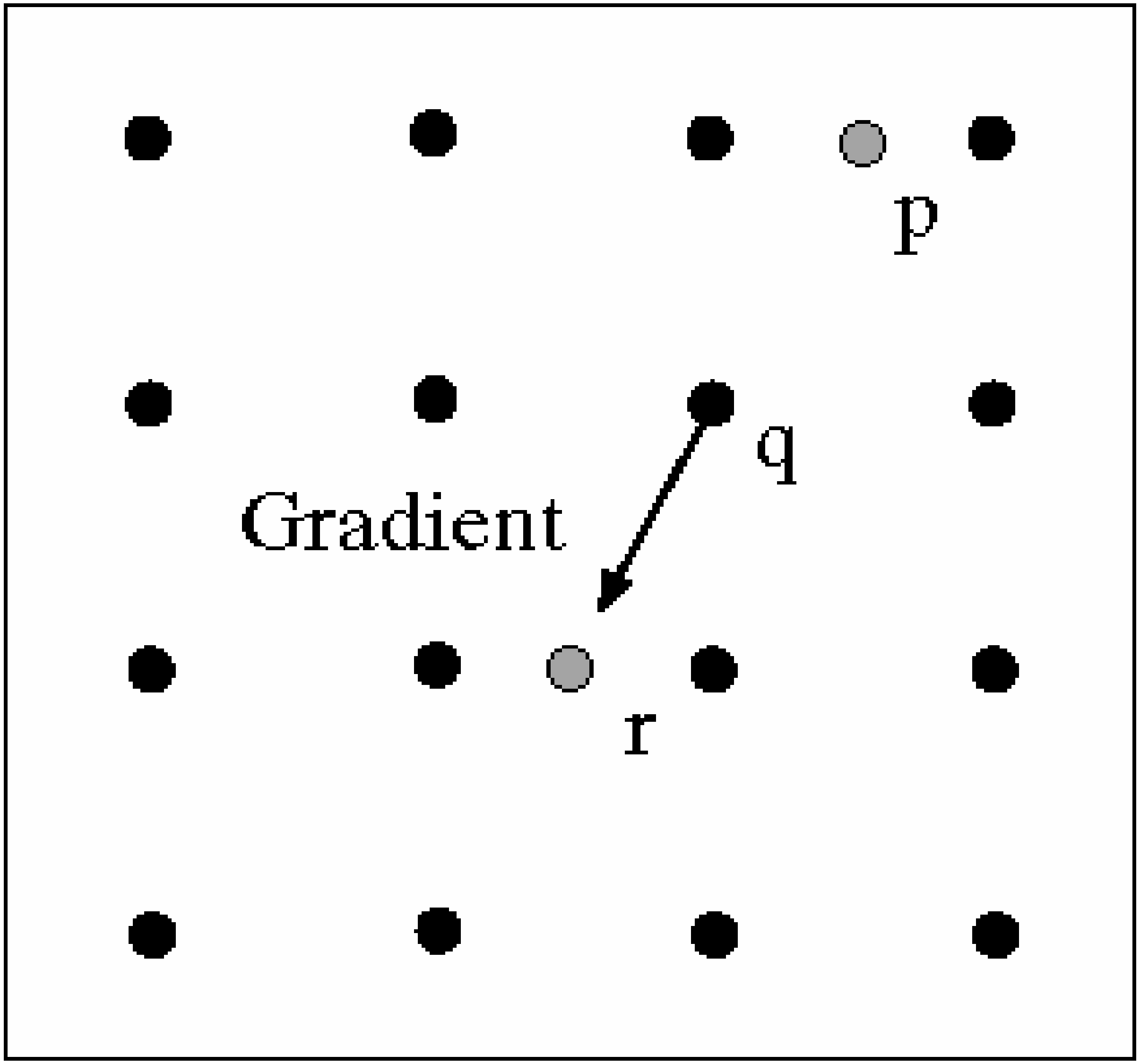
Gradient magnitude at center pixel is lower than the gradient magnitude of a neighbor *in the direction of the gradient*
→ Discard center pixel (set magnitude to 0)

Gradient magnitude at center pixel is greater than gradient magnitude of all the neighbors *in the direction of the gradient*
→ Keep center pixel unchanged

Note: Brighter squares illustrate stronger edge response.

Adapted from Martial Hebert, CMU

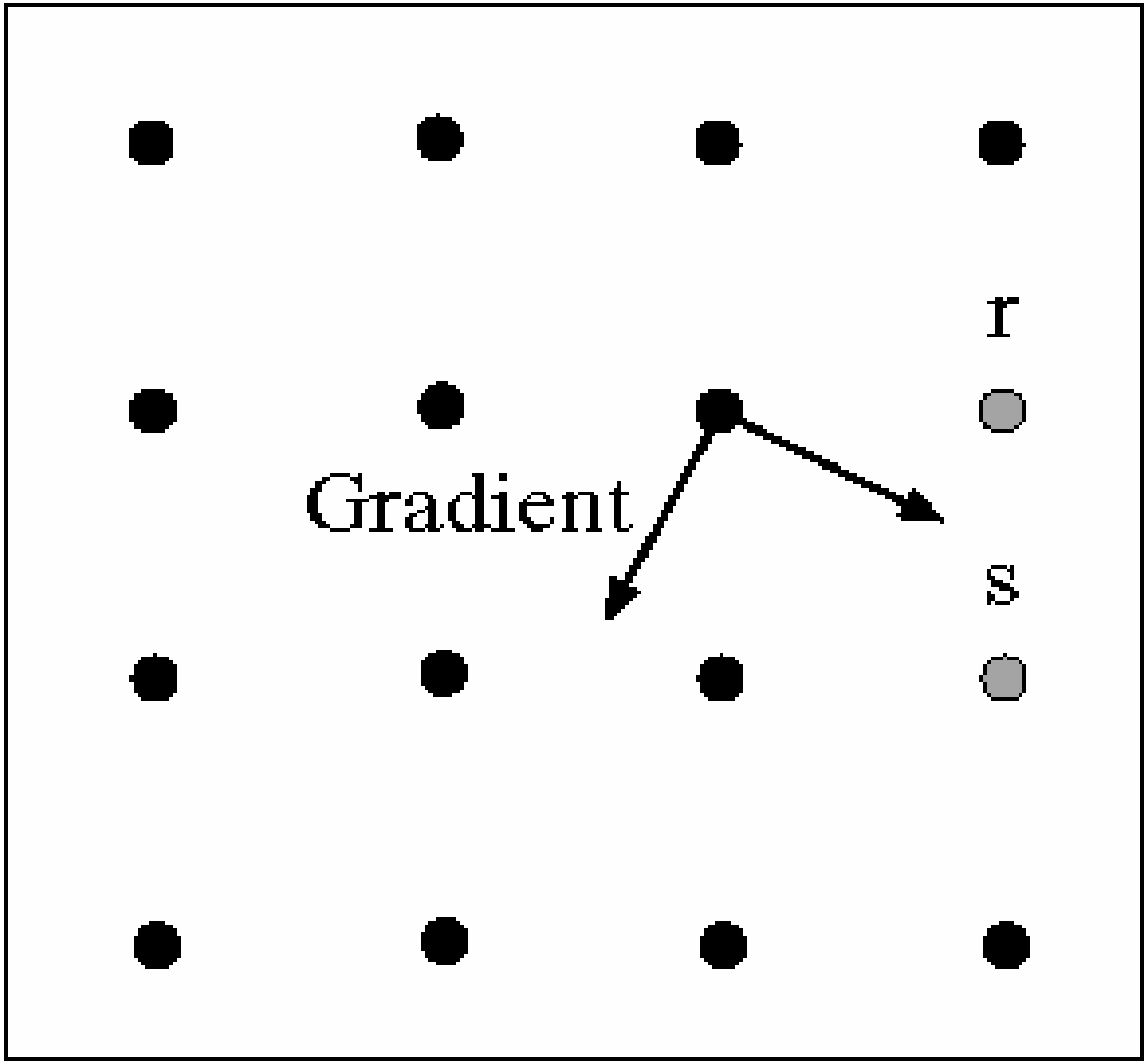
Canny edge detector



At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

Adapted from David Forsyth, UC Berkeley

Canny edge detector



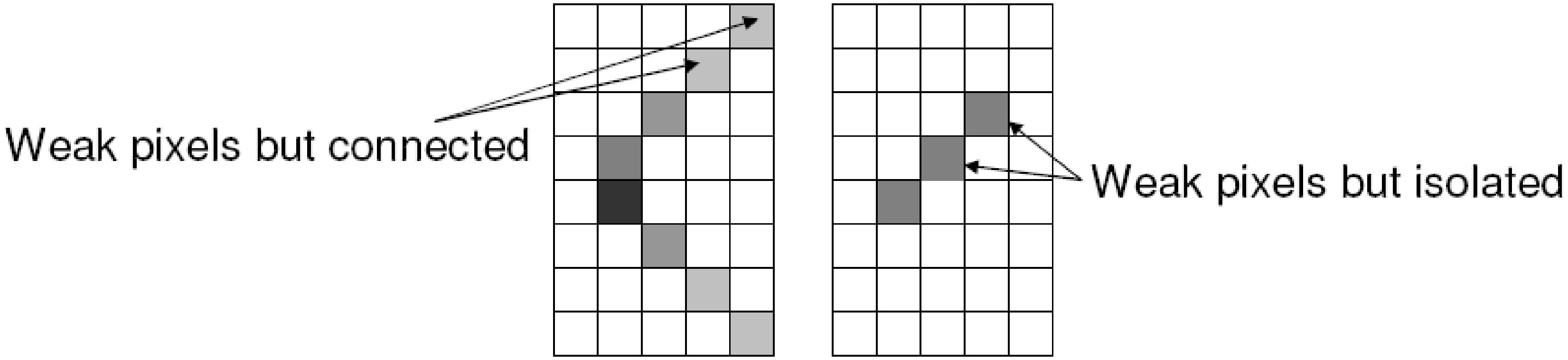
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).

Adapted from David Forsyth, UC Berkeley

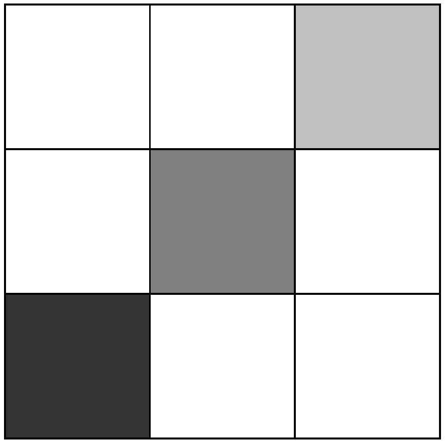
Canny edge detector

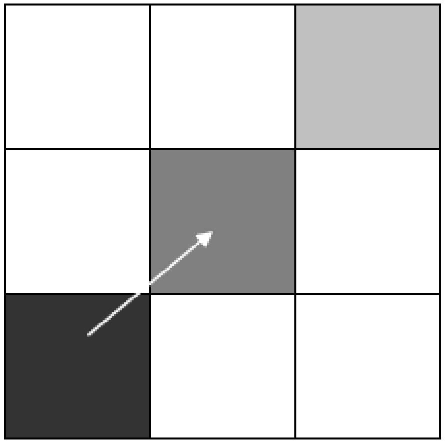
- Hysteresis thresholding:
  - Once the gradient magnitudes are thinned, high magnitude contours are tracked.
  - In the final aggregation phase, continuous contour segments are sequentially followed.
  - Contour following is initiated only on edge pixels where the gradient magnitude meets a high threshold.
  - However, once started, a contour may be followed through pixels whose gradient magnitude meet a lower threshold (usually about half of the higher starting threshold).

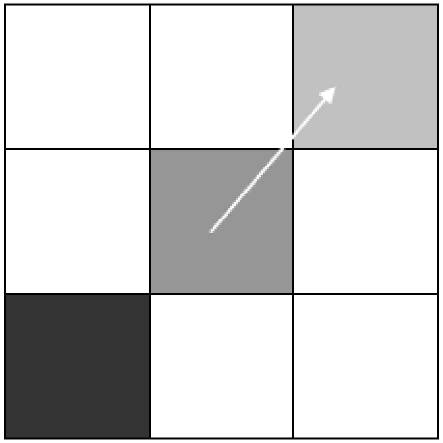
Canny edge detector


Weak pixels but connected

Weak pixels but isolated


Very strong edge response.
Let's start here

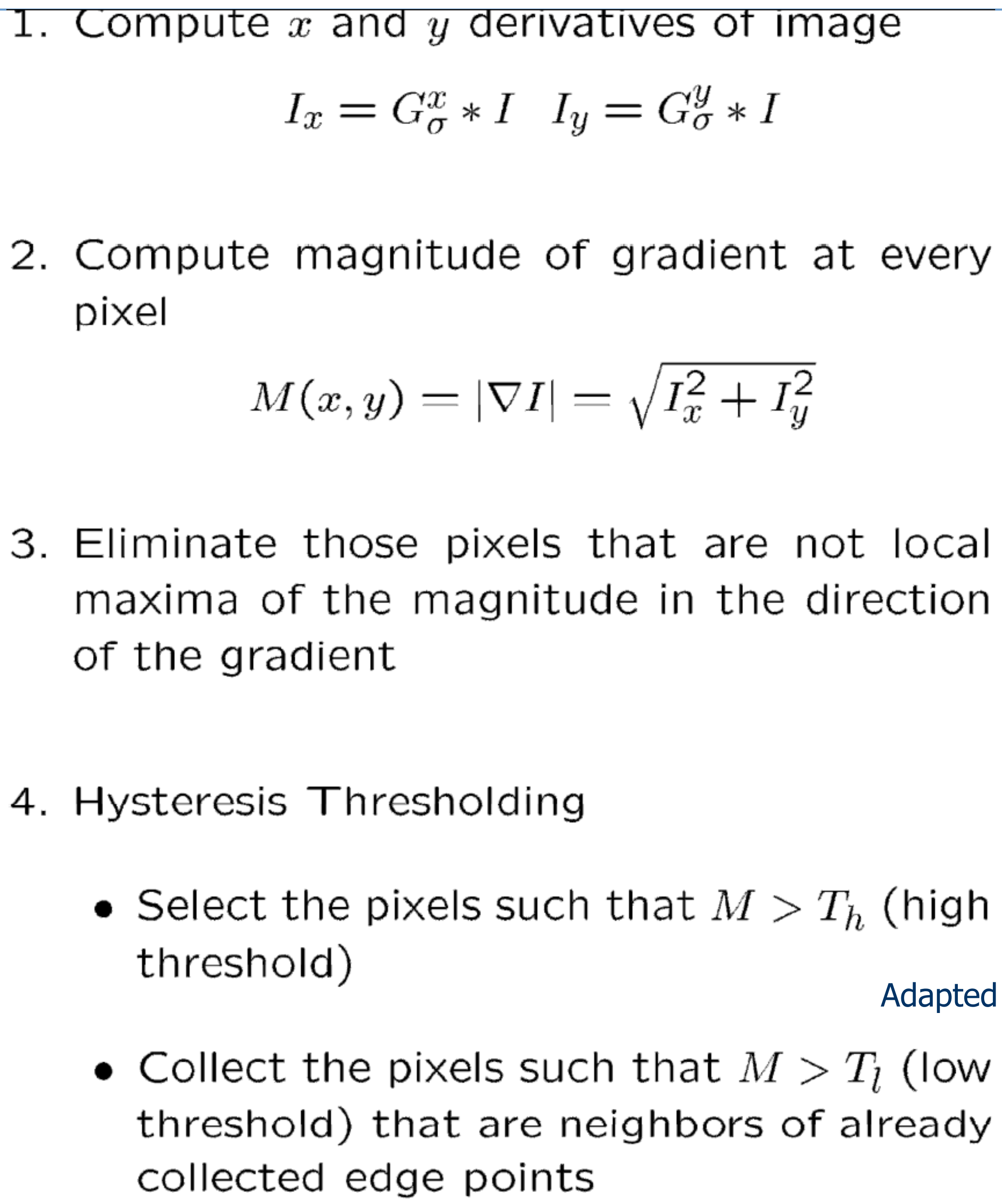

Weaker response but it is connected to a confirmed edge point. Let's keep it.


Continue….

Note: Darker squares illustrate stronger edge response (larger *M*)

Adapted from Martial Hebert, CMU

Canny edge detector



1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute magnitude of gradient at every pixel

$$M(x,y) = |\nabla I| = \sqrt{I_x^2 + I_y^2}$$

3. Eliminate those pixels that are not local maxima of the magnitude in the direction of the gradient

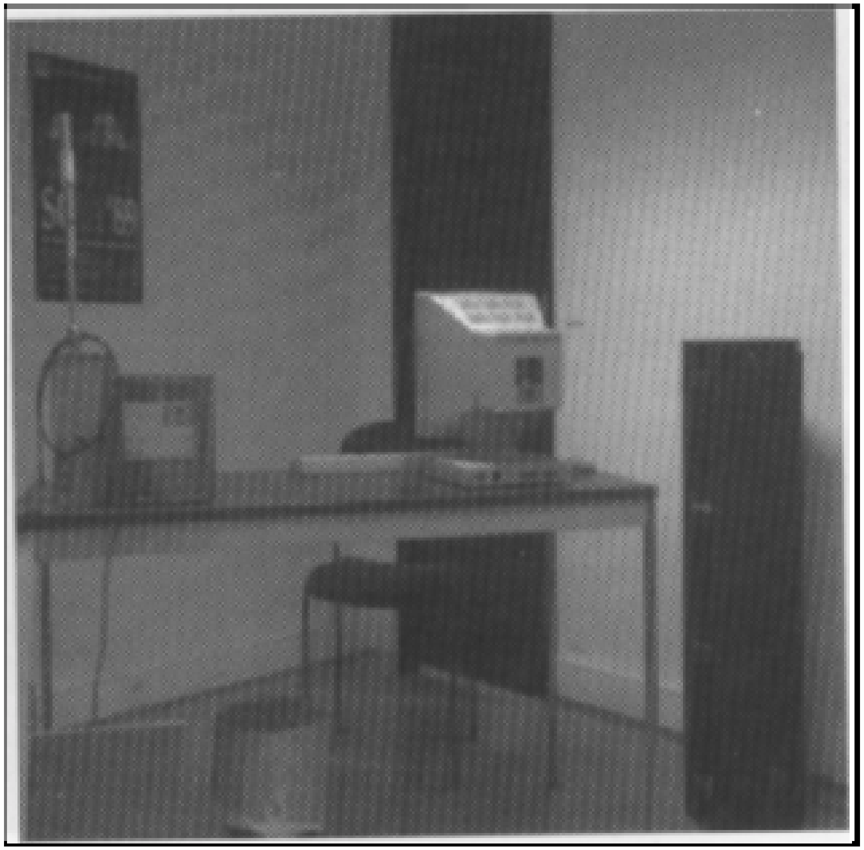
4. Hysteresis Thresholding

   - Select the pixels such that $M > T_h$ (high threshold)

   - Collect the pixels such that $M > T_l$ (low threshold) that are neighbors of already collected edge points

Adapted from Martial Hebert, CMU

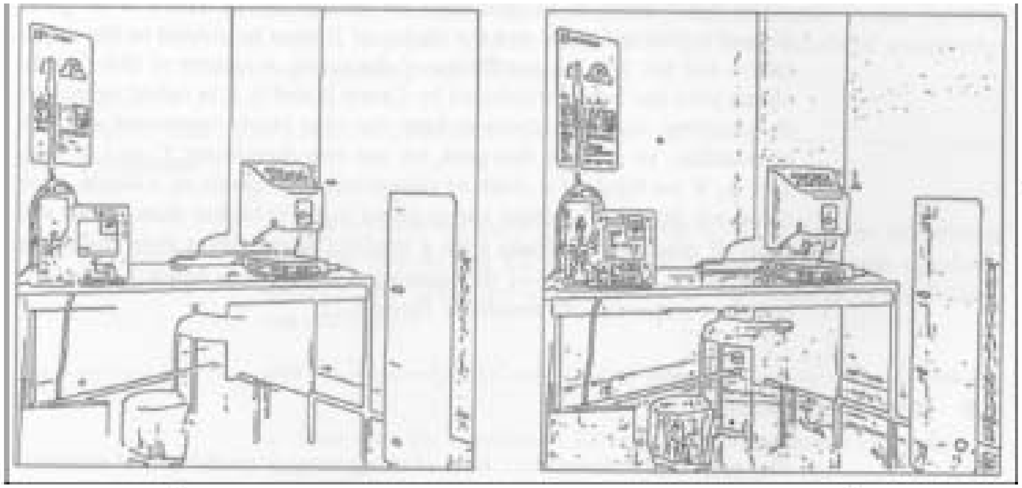
Canny edge detector



Input image

Adapted from Martial Hebert, CMU

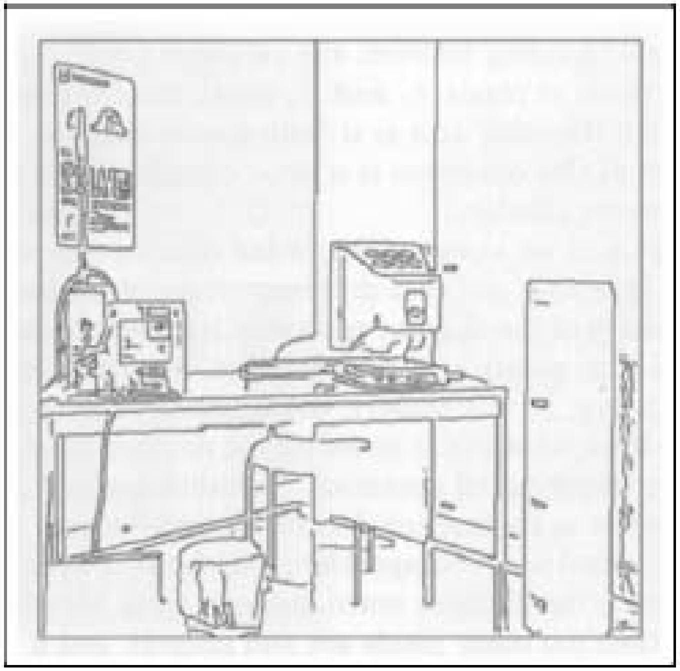
Canny edge detector



T=15

T=5

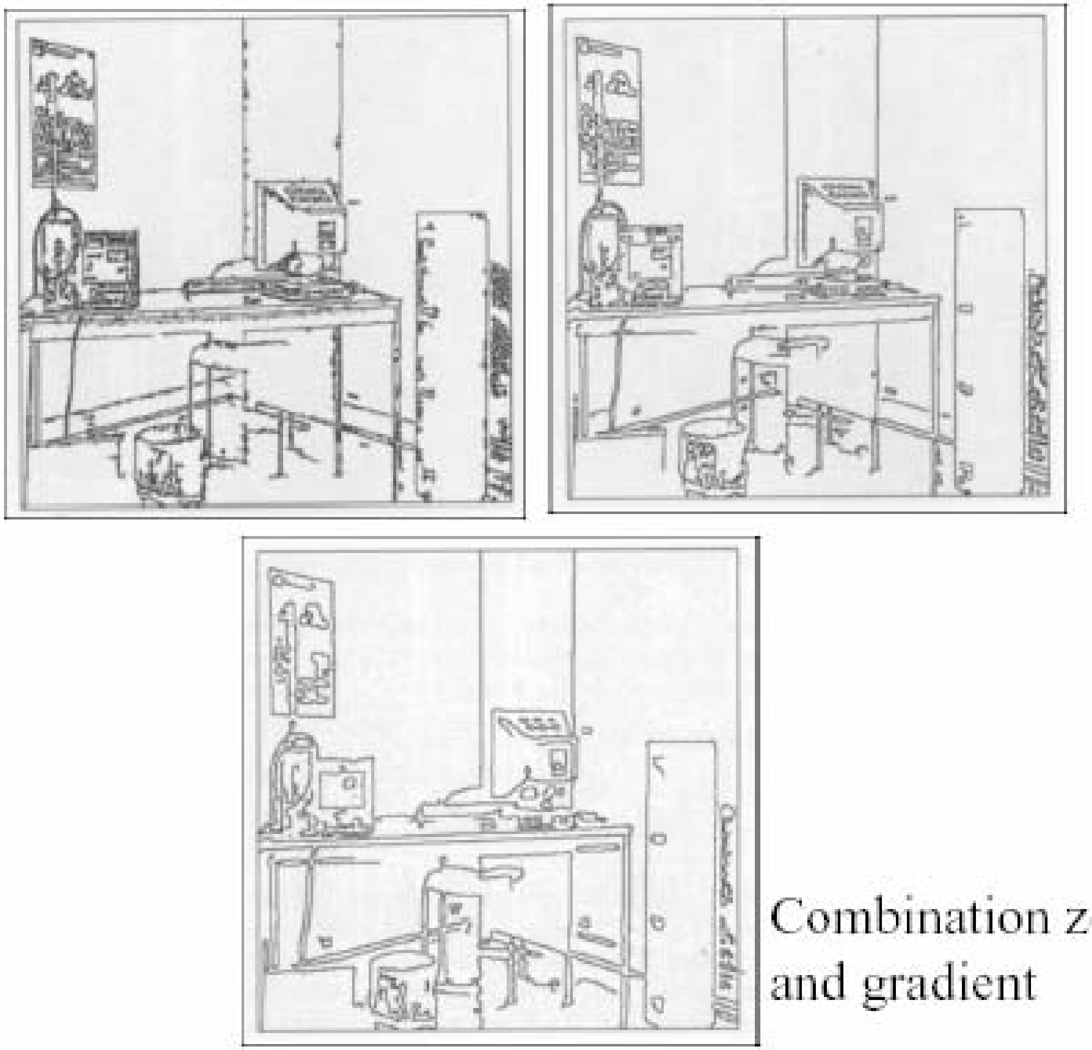
Hysteresis thresholding



Hysteresis $T_h=15$ $T_l = 5$

Adapted from Martial Hebert, CMU

Canny edge detector



$\sigma = 1$

$\sigma = 2$

Combination zero-crossings and gradient

$\sigma = 3$

Adapted from Martial Hebert, CMU

Canny edge detector

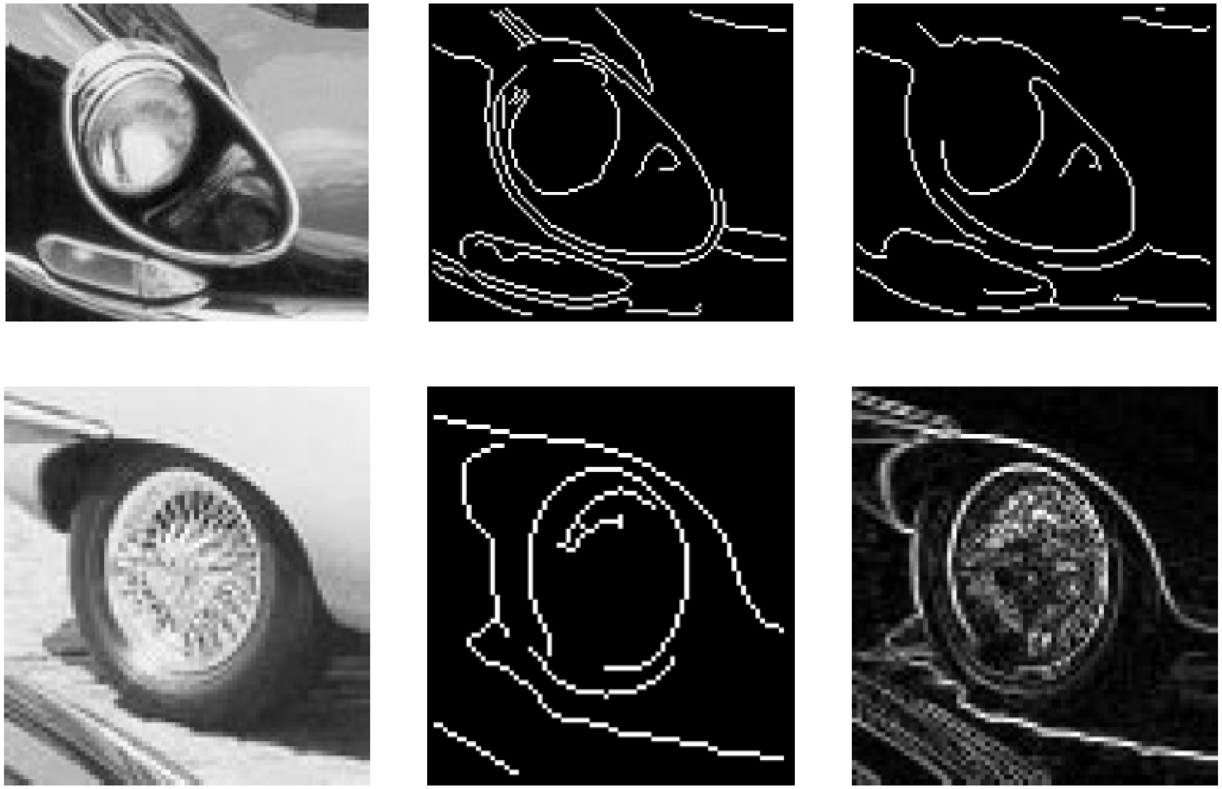


Figure 10.15: (Top left) Image of headlight of a black car; (top center) results of Canny operator with $\sigma = 1$; (top right) results of Canny operator with $\sigma = 4$; (bottom left) image of car wheel; (bottom center) results of Canny operator with $\sigma = 1$; (bottom right) results of Roberts operator. Note in the top row how specular reflection at the top left distracts the edge detector from the boundary of the chrome headlight rim. In the bottom row, note how the shadow of the car connects to the tire which connects to the fender: neither the tire nor the spokes are detected well.

Adapted from Shapiro and Stockman

Canny edge detector

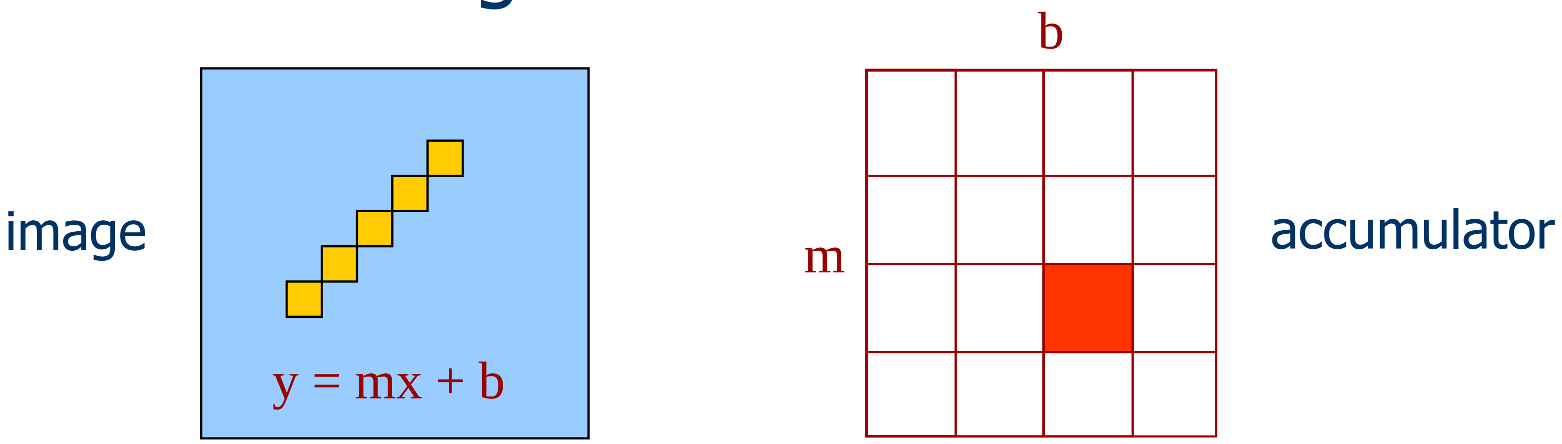
- The Canny operator gives single-pixel-wide images with good continuation between adjacent pixels.
- It is the most widely used edge operator today; no one has done better since it came out in the late 80s. Many implementations are available.
- It is very sensitive to its parameters, which need to be adjusted for different application domains.

Edge linking

- Hough transform
  - Finding line segments
  - Finding circles
- Model fitting
  - Fitting line segments
  - Fitting ellipses
- Edge tracking

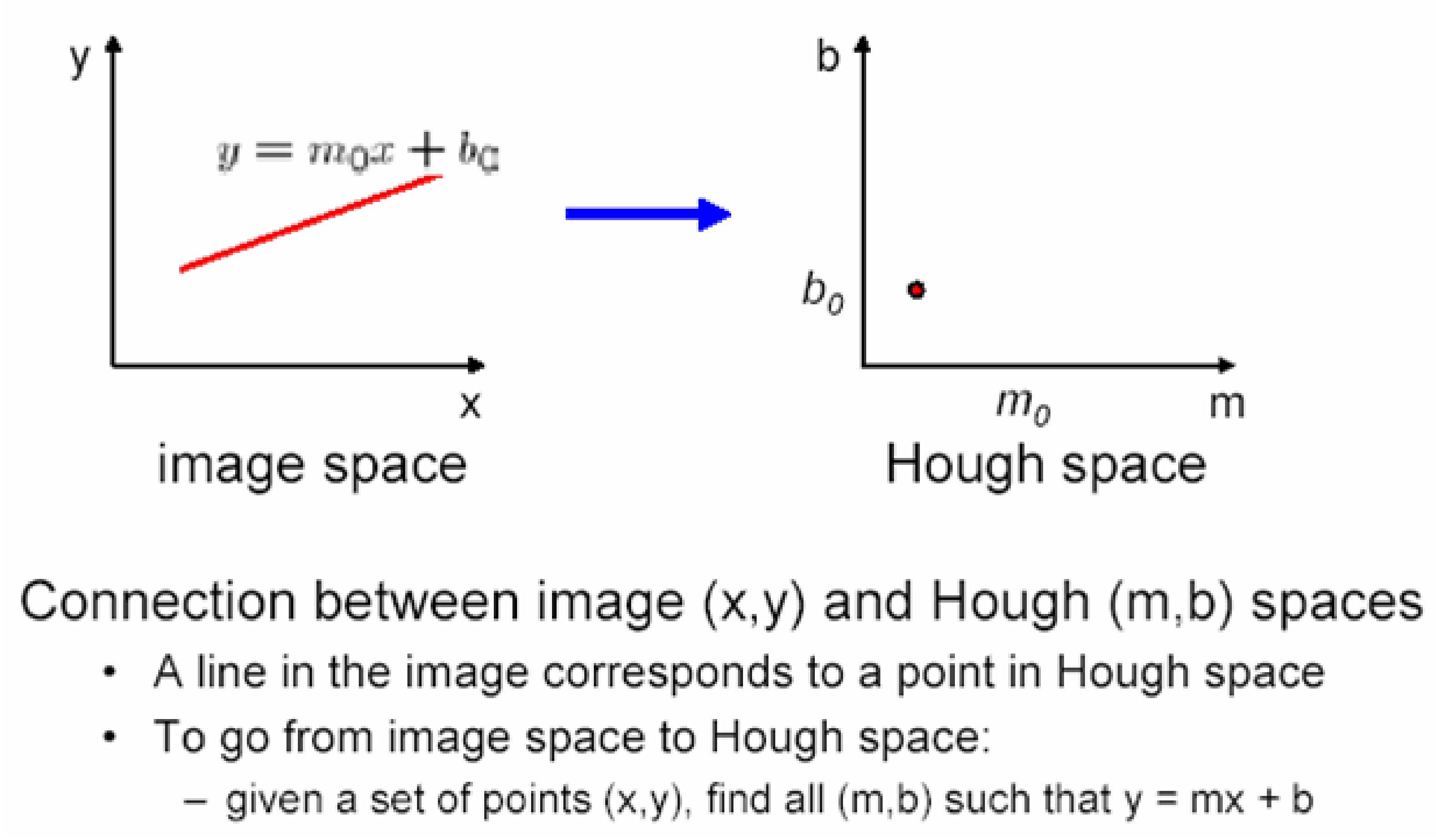
Hough transform

- The Hough transform is a method for detecting lines or curves specified by a parametric function.
- If the parameters are p_1, p_2, … p_n, then the Hough procedure uses an n-dimensional accumulator array in which it accumulates votes for the correct parameters of the lines or curves found on the image.



Adapted from Linda Shapiro, U of Washington

Hough transform: line segments

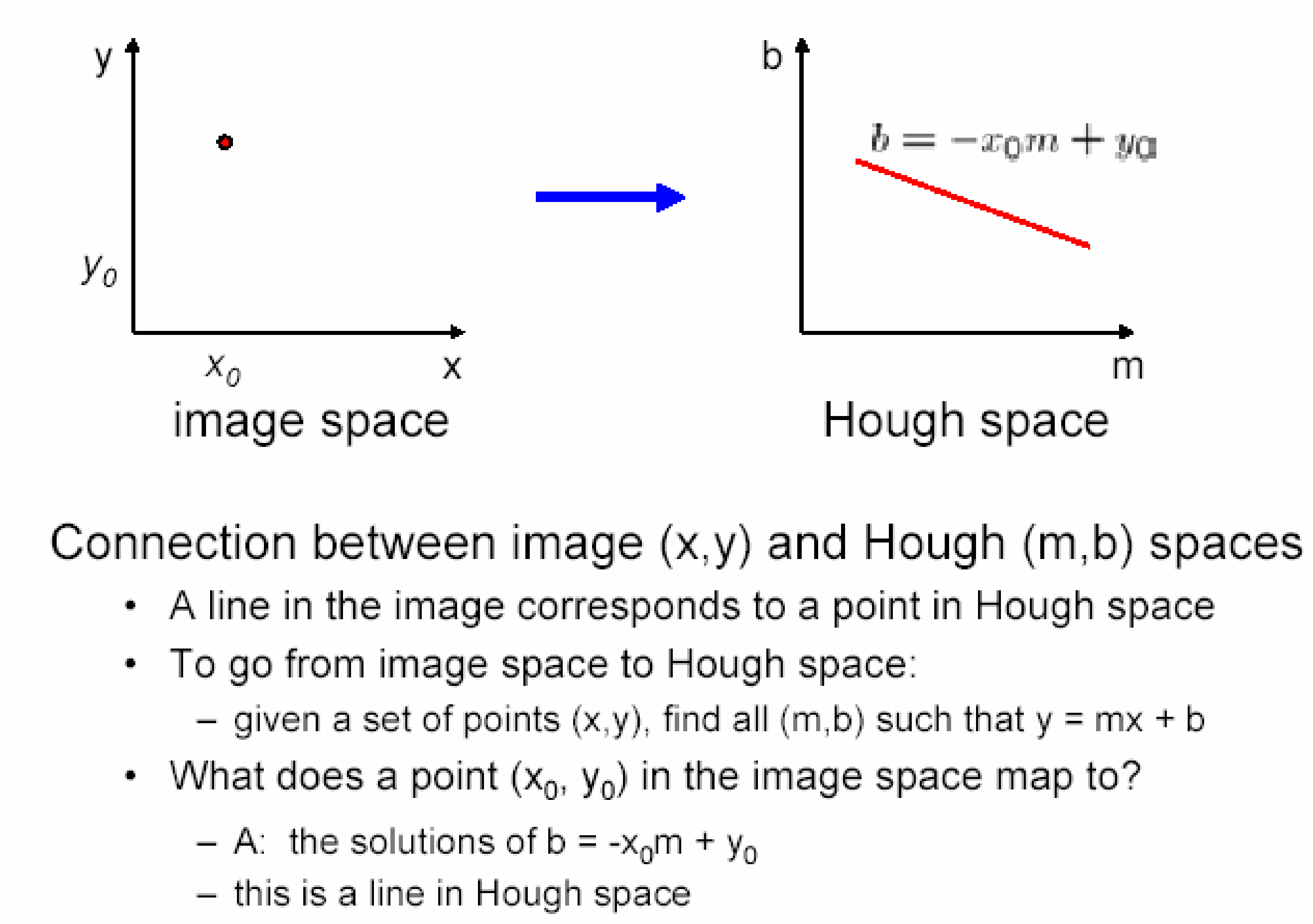


Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b

Adapted from Steve Seitz, U of Washington

Hough transform: line segments



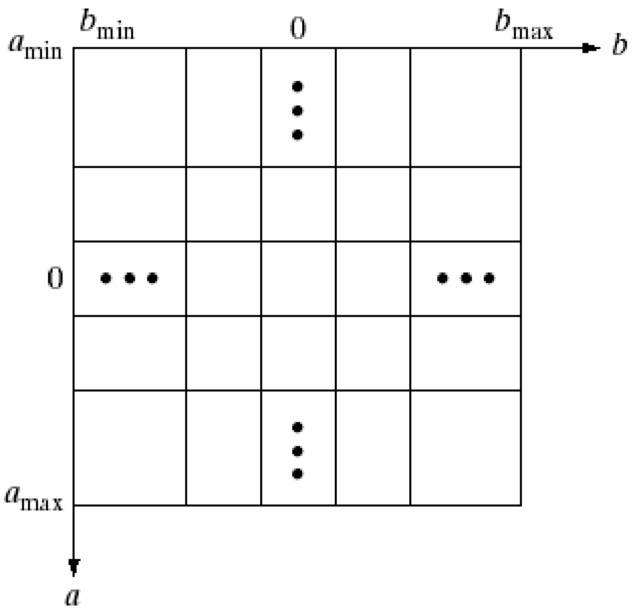
Connection between image (x,y) and Hough (m,b) spaces

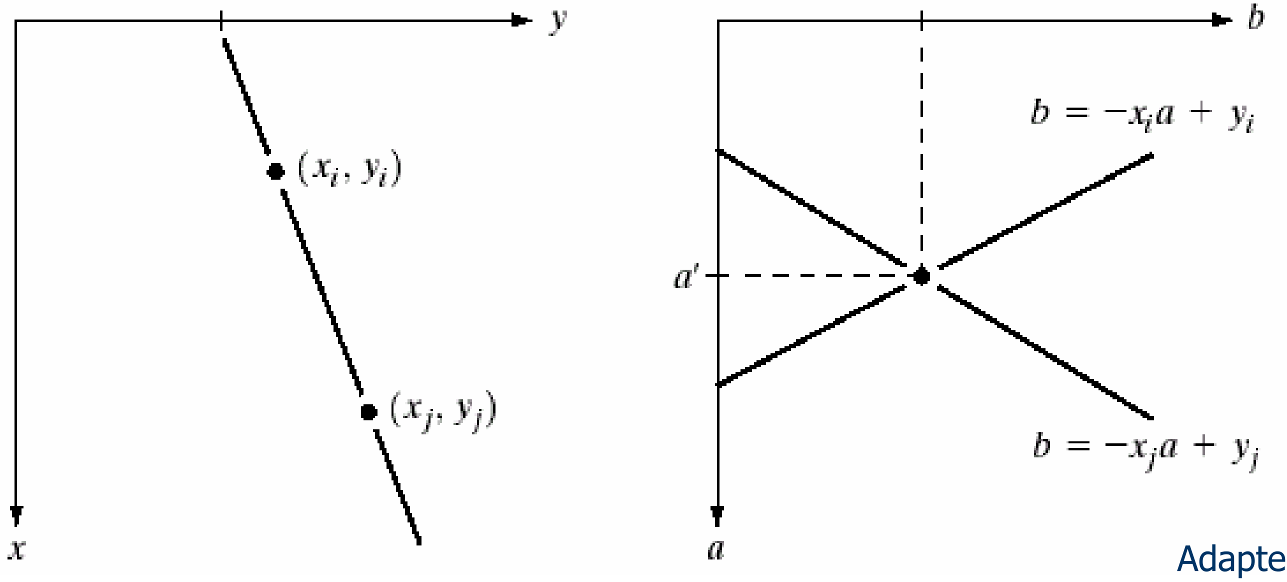
- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b
- What does a point (x_0, y_0) in the image space map to?
  - A: the solutions of $b = -x_0m + y_0$
  - this is a line in Hough space

Adapted from Steve Seitz, U of Washington

Hough transform: line segments

FIGURE 10.18 Subdivision of the parameter plane for use in the Hough transform.





a b

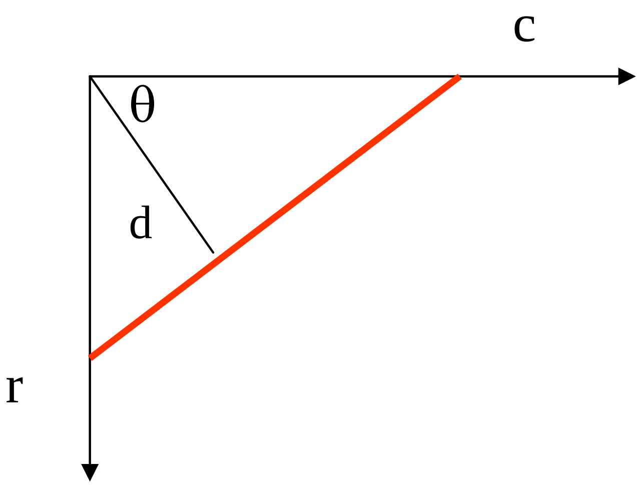
FIGURE 10.17 (a) xy-plane. (b) Parameter space.

Adapted from Gonzales and Woods

Hough transform: line segments

- $y = mx + b$ is not suitable (why?)
- The equation generally used is:

$$d = r \sin(\theta) + c \cos(\theta).$$

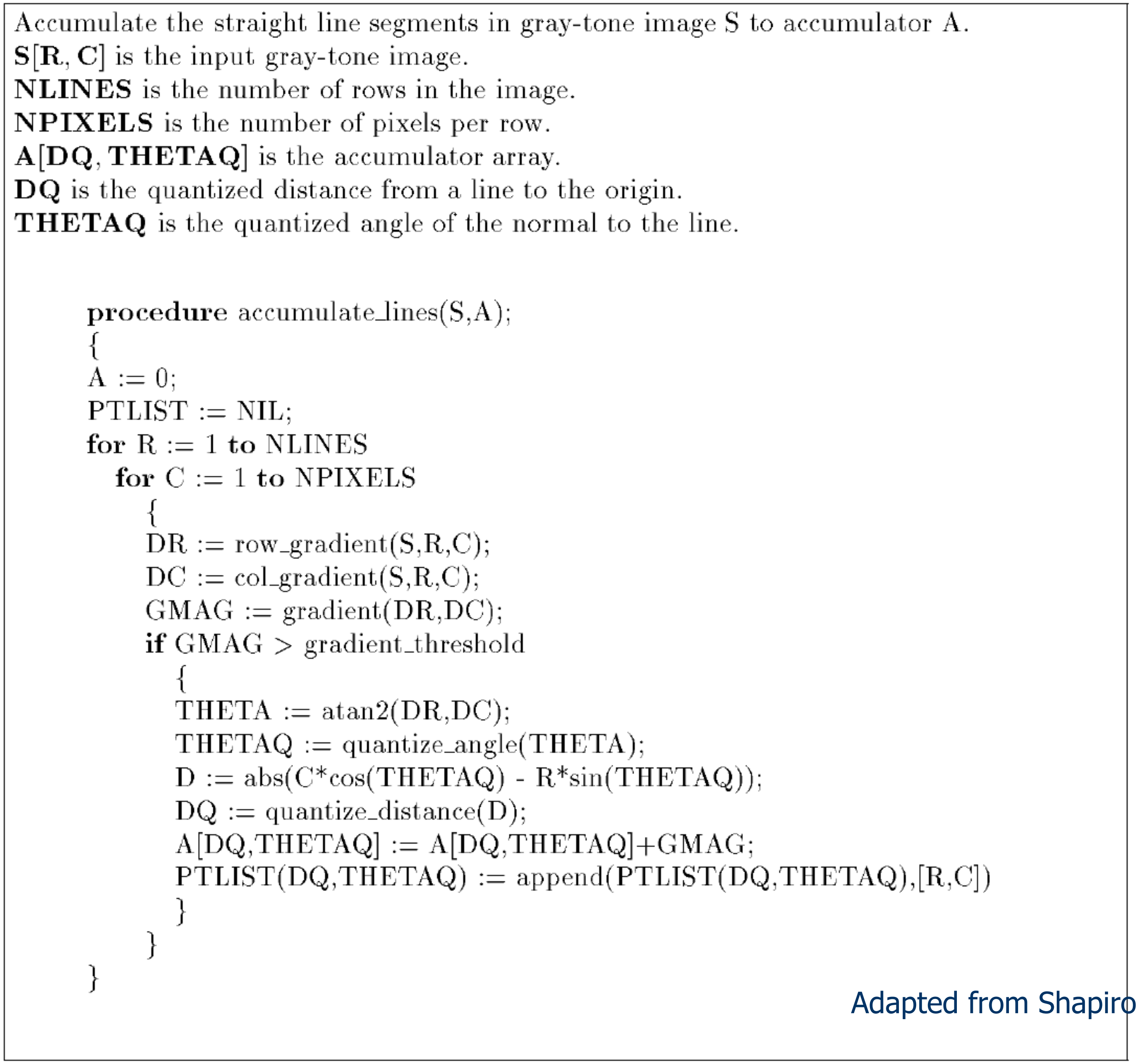


d is the distance from the line to origin.

θ is the angle the perpendicular makes with the column axis.

Adapted from Linda Shapiro, U of Washington

Hough transform: line segments



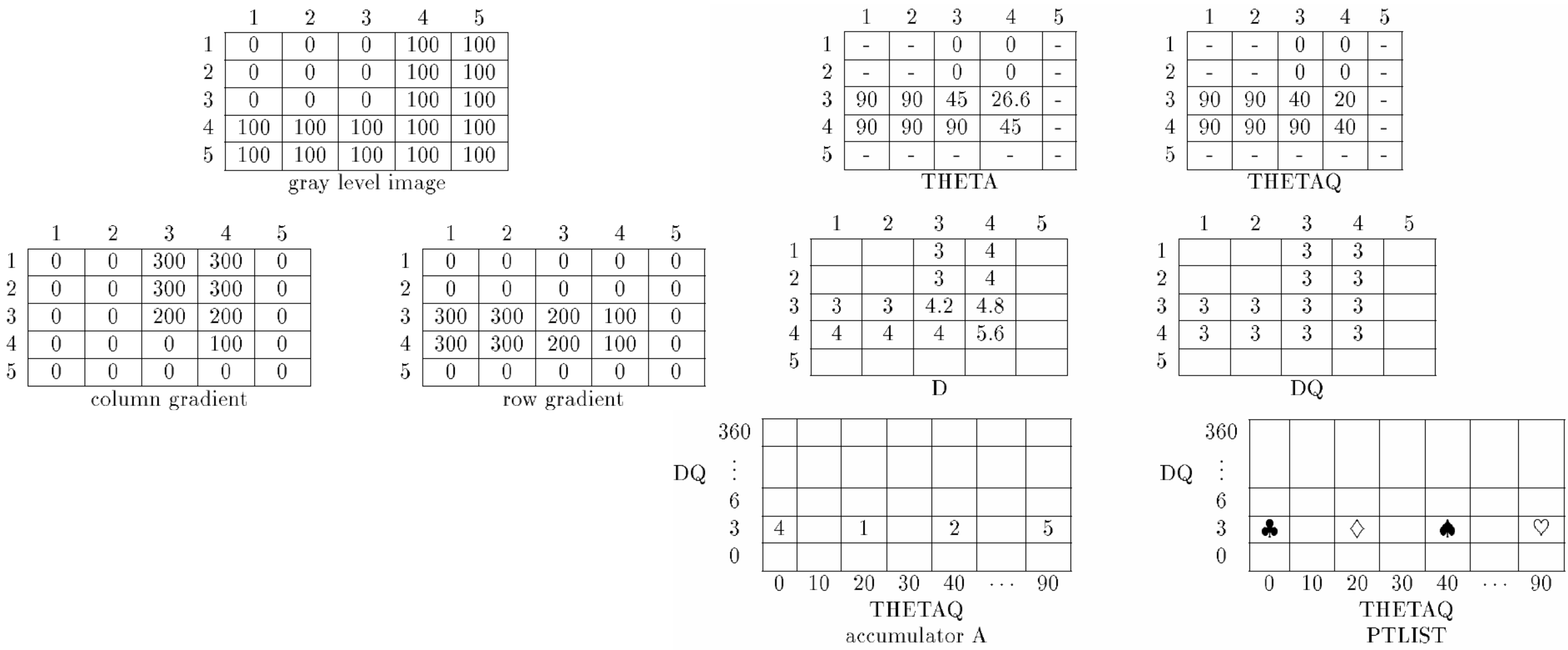
Accumulate the straight line segments in gray-tone image S to accumulator A.
S[**R**, **C**] is the input gray-tone image.
NLINES is the number of rows in the image.
NPIXELS is the number of pixels per row.
A[**DQ**, **THETAQ**] is the accumulator array.
DQ is the quantized distance from a line to the origin.
THETAQ is the quantized angle of the normal to the line.

```
procedure accumulate_lines(S,A);
{
A := 0;
PTLIST := NIL;
for R := 1 to NLINES
  for C := 1 to NPIXELS
    {
    DR := row_gradient(S,R,C);
    DC := col_gradient(S,R,C);
    GMAG := gradient(DR,DC);
    if GMAG > gradient_threshold
      {
      THETA := atan2(DR,DC);
      THETAQ := quantize_angle(THETA);
      D := abs(C*cos(THETAQ) - R*sin(THETAQ));
      DQ := quantize_distance(D);
      A[DQ,THETAQ] := A[DQ,THETAQ]+GMAG;
      PTLIST(DQ,THETAQ) := append(PTLIST(DQ,THETAQ),[R,C])
      }
    }
}
```

Algorithm 7: Hough Transform for Finding Straight Lines

Adapted from Shapiro and Stockman

Hough transform: line segments



gray level image

	1	2	3	4	5
1	0	0	0	100	100
2	0	0	0	100	100
3	0	0	0	100	100
4	100	100	100	100	100
5	100	100	100	100	100

THETA

	1	2	3	4	5
1	-	-	0	0	-
2	-	-	0	0	-
3	90	90	45	26.6	-
4	90	90	90	45	-
5	-	-	-	-	-

THETAQ

	1	2	3	4	5
1	-	-	0	0	-
2	-	-	0	0	-
3	90	90	40	20	-
4	90	90	90	40	-
5	-	-	-	-	-

column gradient

	1	2	3	4	5
1	0	0	300	300	0
2	0	0	300	300	0
3	0	0	200	200	0
4	0	0	0	100	0
5	0	0	0	0	0

row gradient

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	300	300	200	100	0
4	300	300	200	100	0
5	0	0	0	0	0

D

	1	2	3	4	5
1			3	4	
2			3	4	
3	3	3	4.2	4.8	
4	4	4	4	5.6	
5					

DQ

	1	2	3	4	5
1			3	3	
2			3	3	
3	3	3	3	3	
4	3	3	3	3	
5					

accumulator A

DQ \ THETAQ	0	10	20	30	40	⋯	90
360							
⋮							
6							
3	4		1		2		5
0							

PTLIST

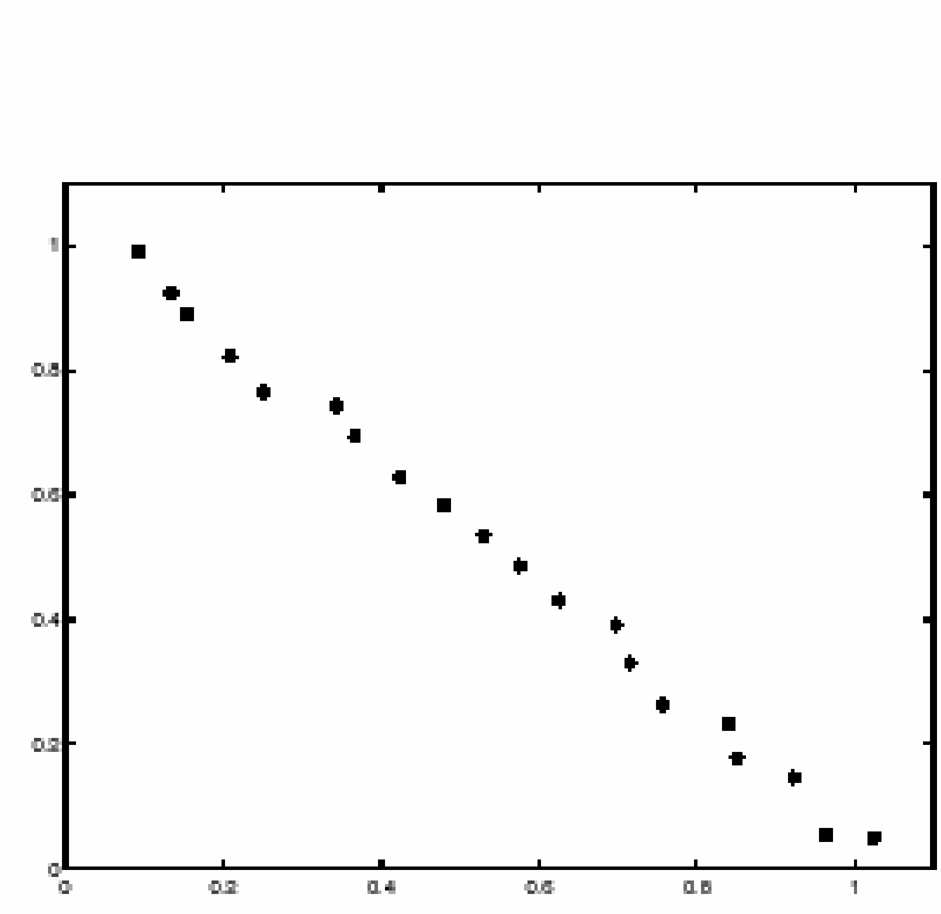
DQ \ THETAQ	0	10	20	30	40	⋯	90
360							
⋮							
6							
3	♣		♢		♠		♡
0							

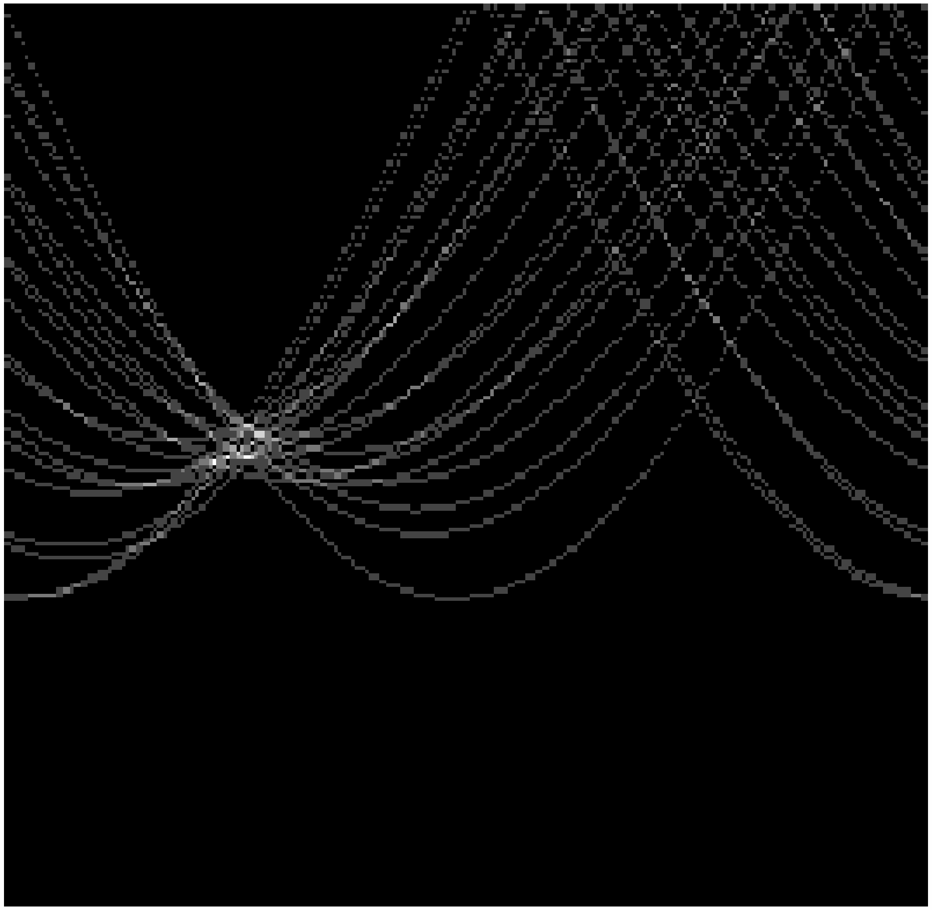
♣ (1,3)(1,4)(2,3)(2,4)
♢ (3,4)
♠ (3,3)(4,4)
♡ (3,1)(3,2)(4,1)(4,2)(4,3)

Figure 10.22: The results of the operation of procedure *accumulate* on a simple gray level image using Prewitt masks. For this small example, the evidence for correct detections is not much larger than that for incorrect ones, but in real images with long segments, the difference will be much more pronounced.

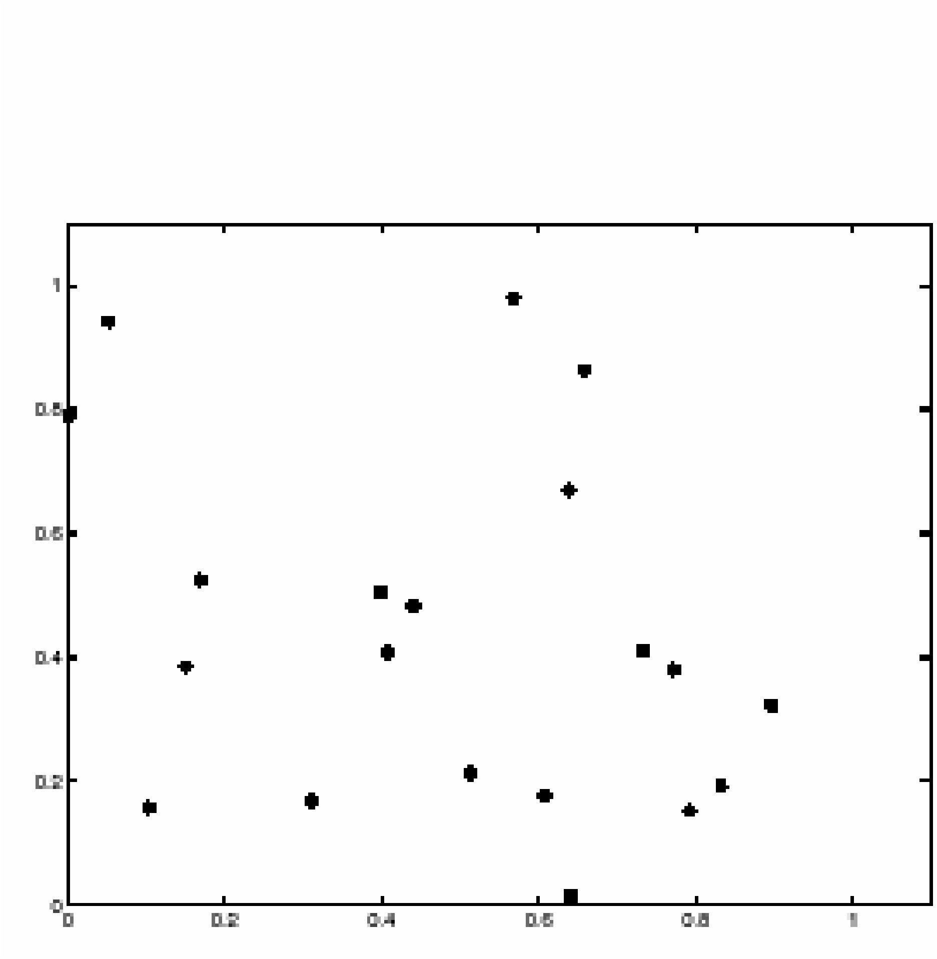
Adapted from Shapiro and Stockman

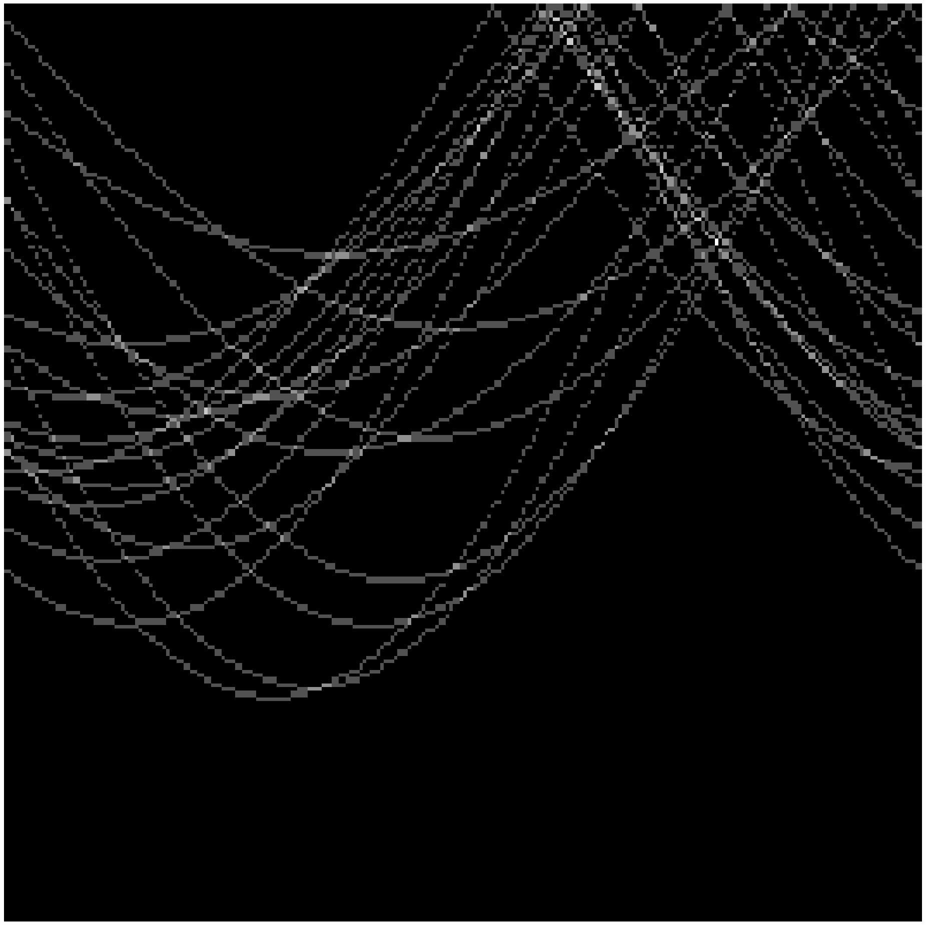
Hough transform: line segments





Hough transform: line segments





Hough transform: line segments

- Extracting the line segments from the accumulators:

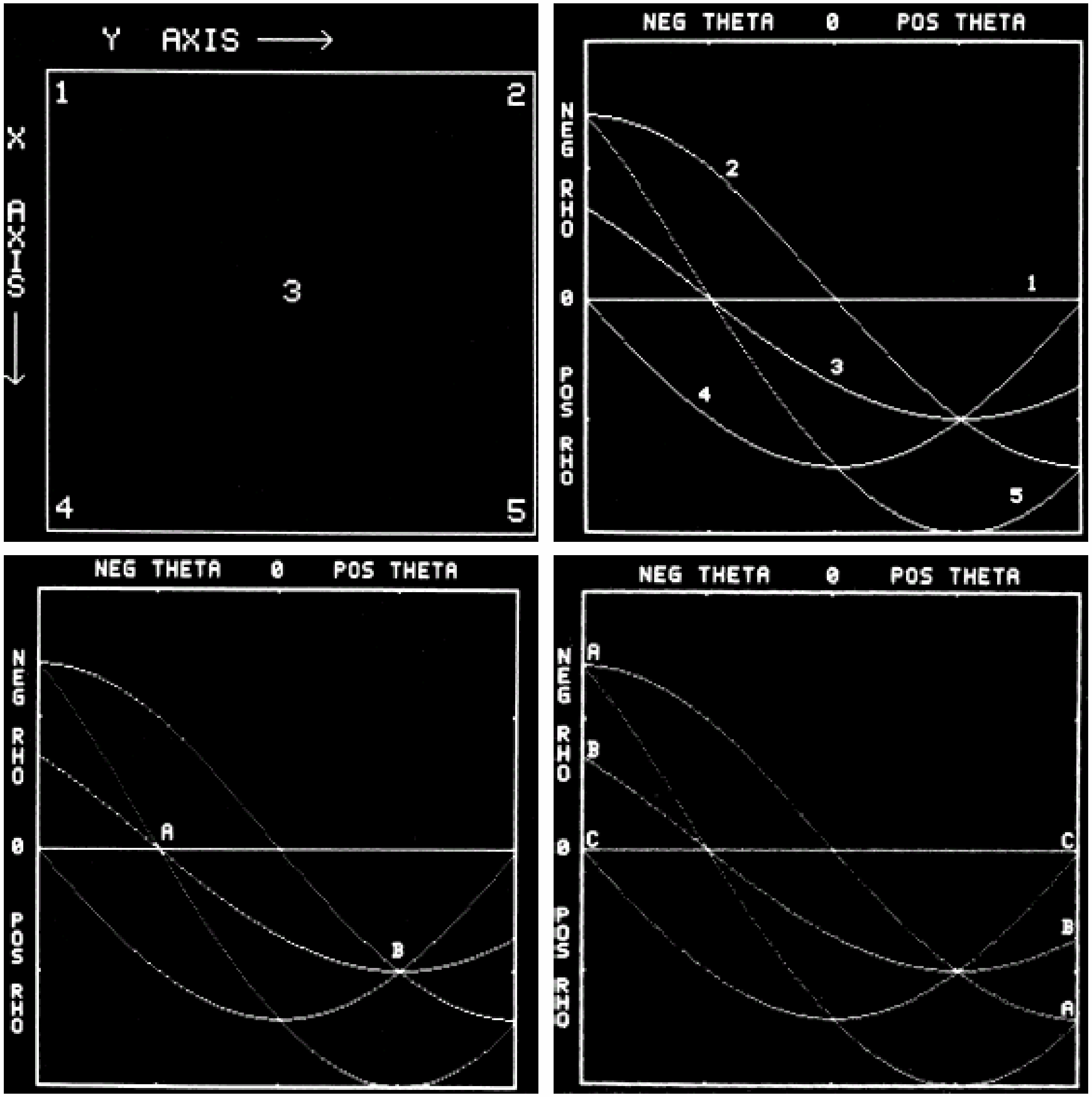
```
1. Pick the bin of A with highest value V
2. While V > value_threshold {
    1. order the corresponding pointlist from PTLIST
    2. merge in high gradient neighbors within 10 degrees
    3. create line segment from final point list
    4. zero out that bin of A
    5. pick the bin of A with highest value V
   }
```

Adapted from Linda Shapiro, U of Washington

Hough transform: line segments

a b
c d

FIGURE 10.20 Illustration of the Hough transform. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)



Hough transform: line segments

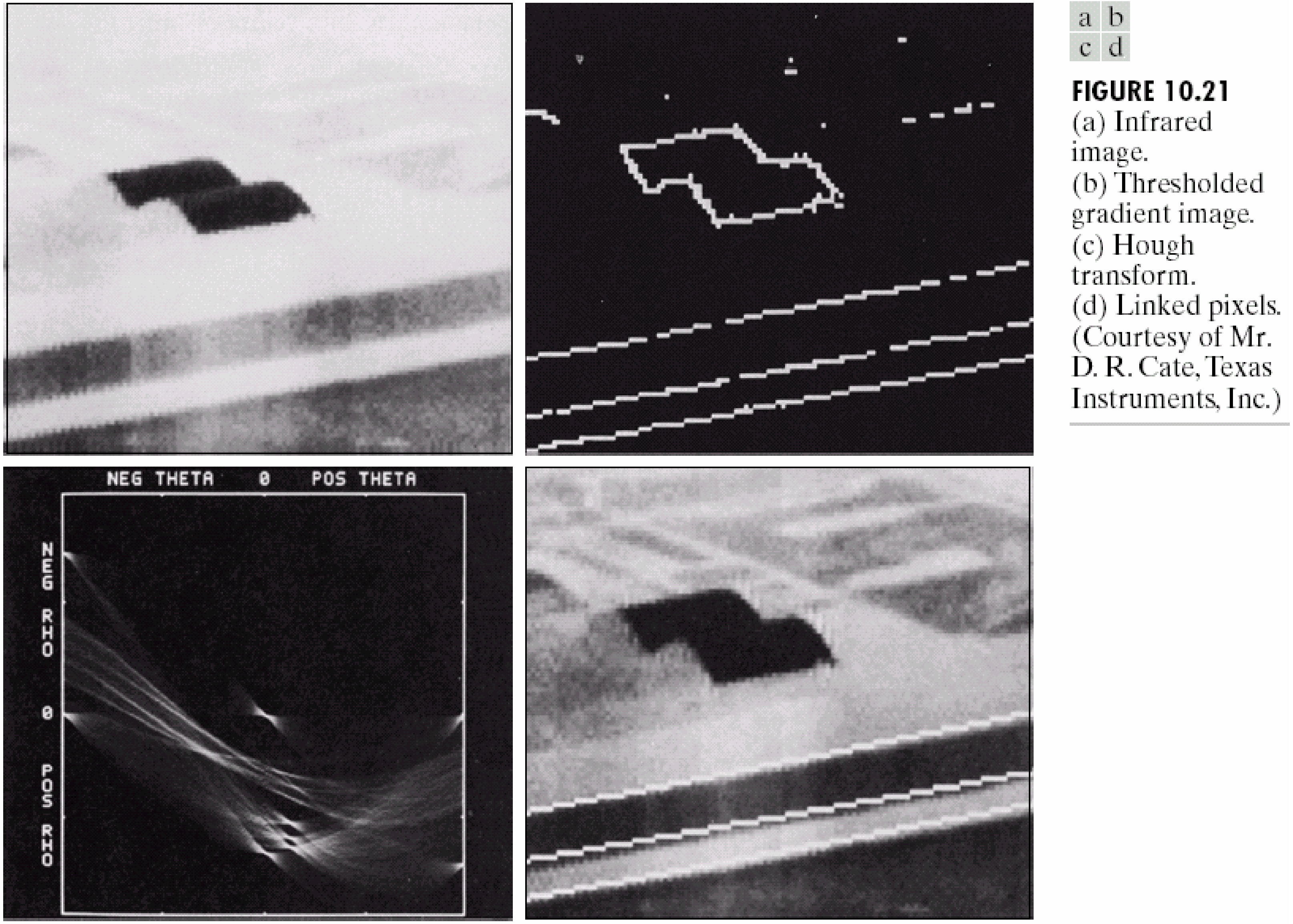
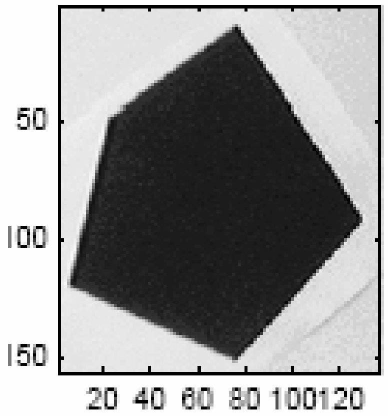
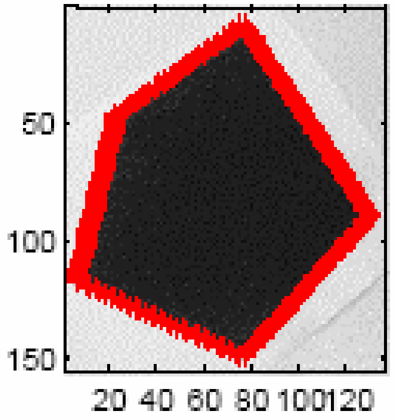


FIGURE 10.21 (a) Infrared image. (b) Thresholded gradient image. (c) Hough transform. (d) Linked pixels. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

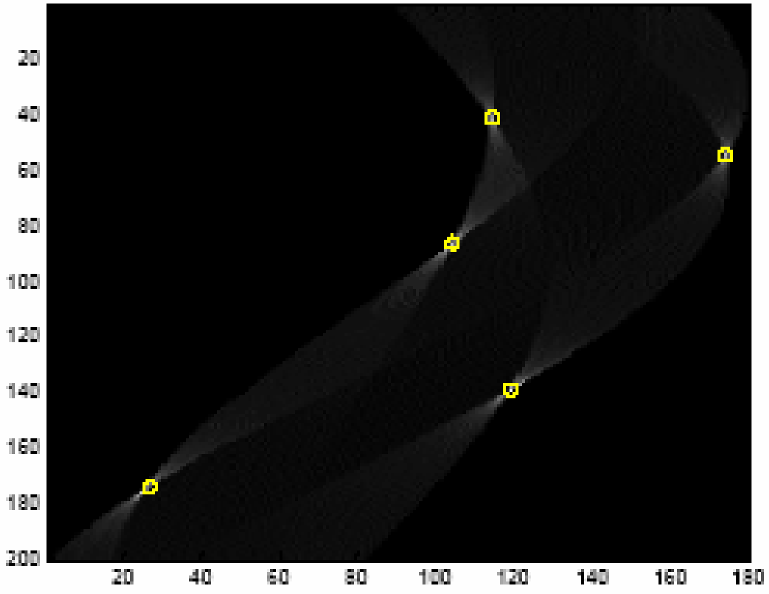
Hough transform: line segments



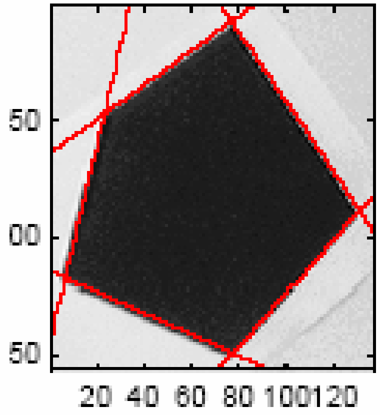
Original Image



Detected Edges



The vote histogram with the detected lines marked with ‘o’



Detected lines

Adapted from Linda Shapiro, U of Washington

Hough transform: line segments

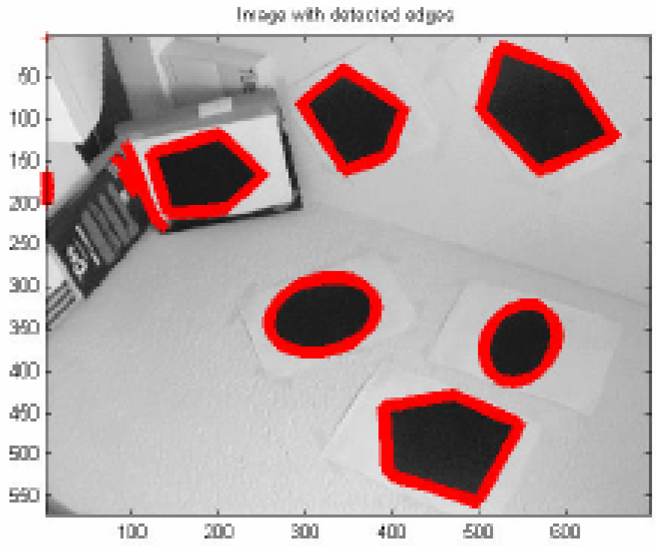
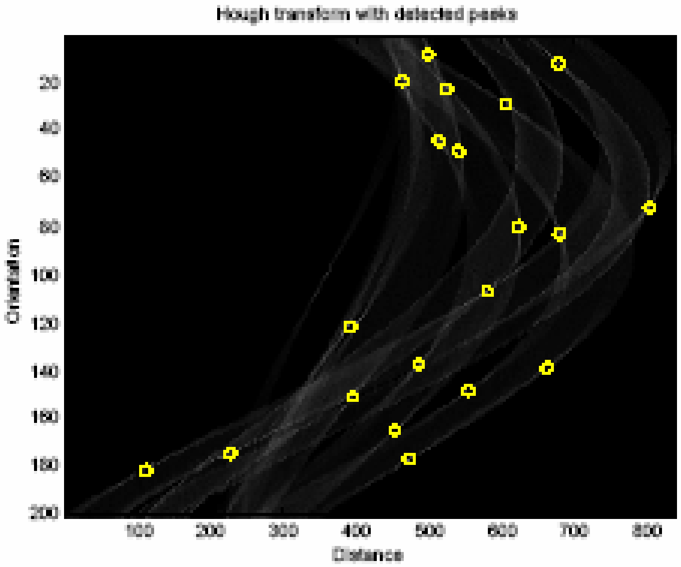


Image with detected edges



The vote histogram with the selected lines

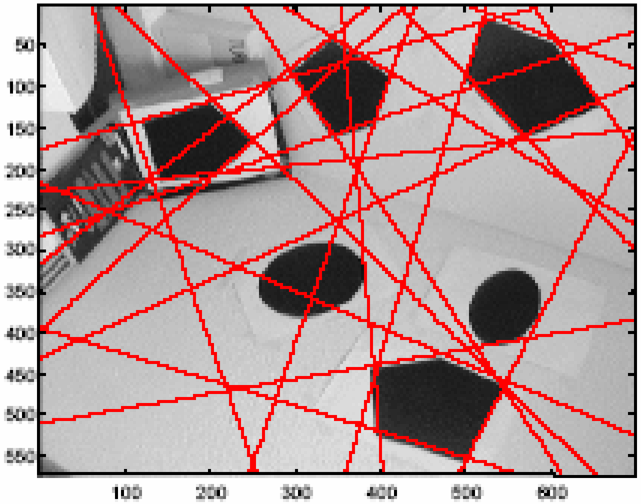


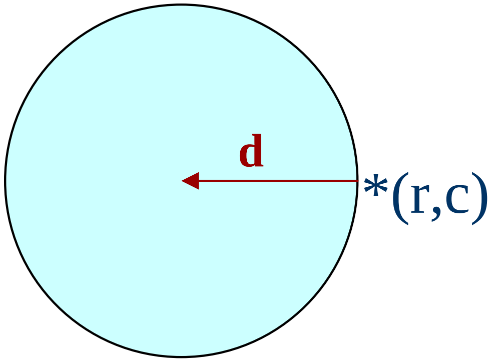
Image with the detected lines

Adapted from Linda Shapiro, U of Washington

Hough transform: circles

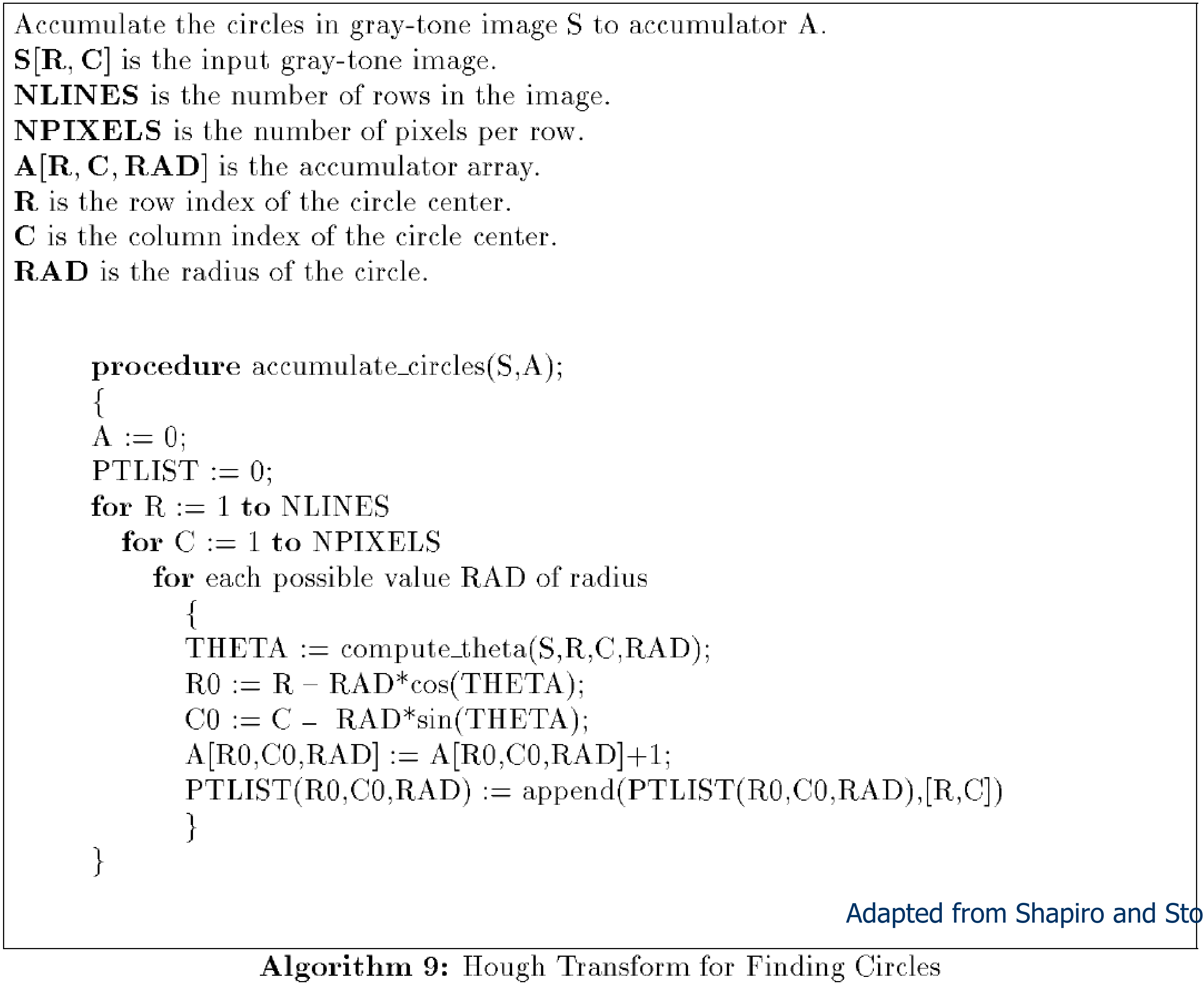
- Main idea: The gradient vector at an edge pixel points the center of the circle.
- Circle equations:
  - $r = r_0 + d \sin(\theta)$
  - $c = c_0 + d \cos(\theta)$

r_0, c_0, d are parameters



Adapted from Linda Shapiro, U of Washington

Hough transform: circles



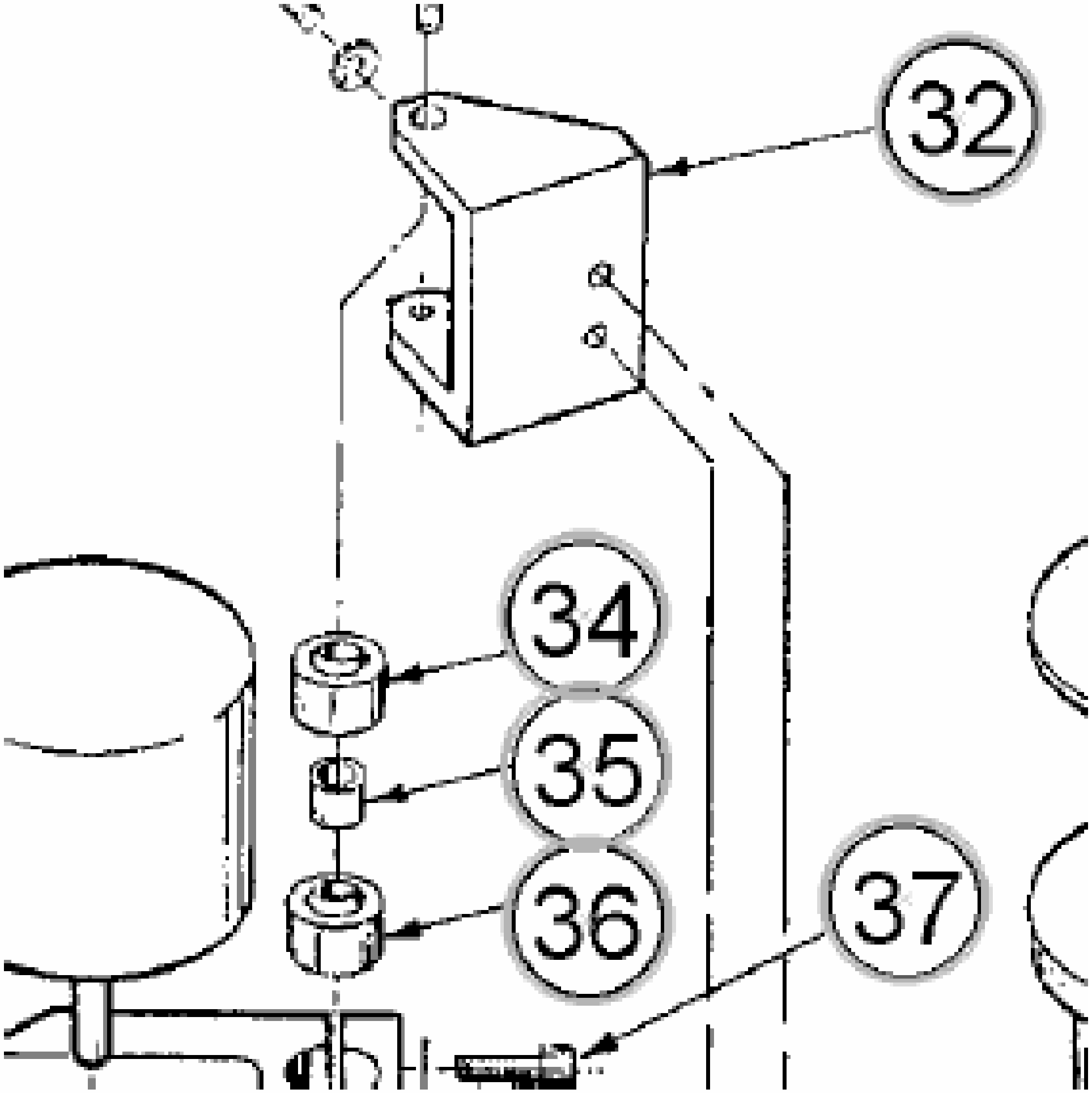
Accumulate the circles in gray-tone image S to accumulator A.
S[**R**, **C**] is the input gray-tone image.
NLINES is the number of rows in the image.
NPIXELS is the number of pixels per row.
A[**R**, **C**, **RAD**] is the accumulator array.
R is the row index of the circle center.
C is the column index of the circle center.
RAD is the radius of the circle.

```
procedure accumulate_circles(S,A);
{
A := 0;
PTLIST := 0;
for R := 1 to NLINES
   for C := 1 to NPIXELS
      for each possible value RAD of radius
         {
         THETA := compute_theta(S,R,C,RAD);
         R0 := R – RAD*cos(THETA);
         C0 := C – RAD*sin(THETA);
         A[R0,C0,RAD] := A[R0,C0,RAD]+1;
         PTLIST(R0,C0,RAD) := append(PTLIST(R0,C0,RAD),[R,C])
         }
}
```

Algorithm 9: Hough Transform for Finding Circles

Adapted from Shapiro and Stockman

Hough transform: circles



Adapted from Shapiro and Stockman

Hough transform: circles

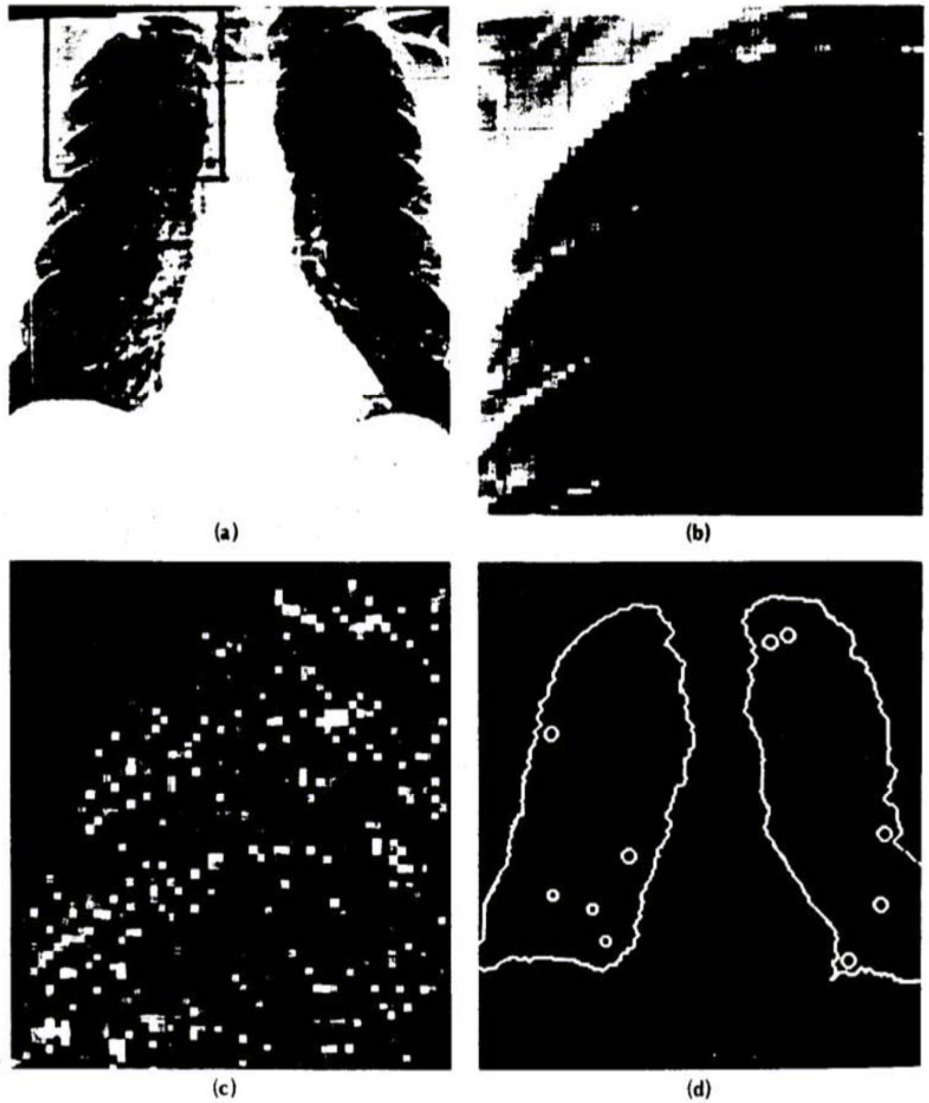
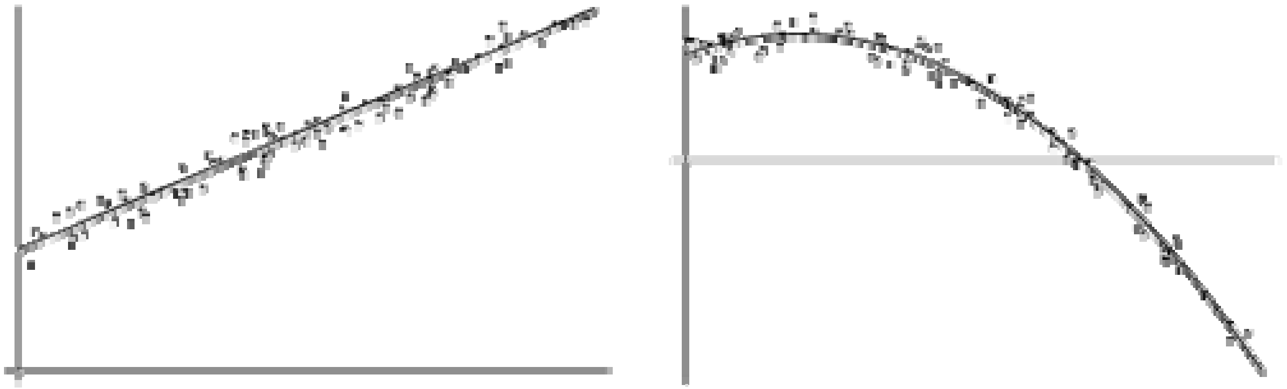


Fig. 4.7 Using the Hough technique for circular shapes. (a) Radiograph. (b) Window. (c) Accumulator array for $r = 3$. (d) Results of maxima detection.

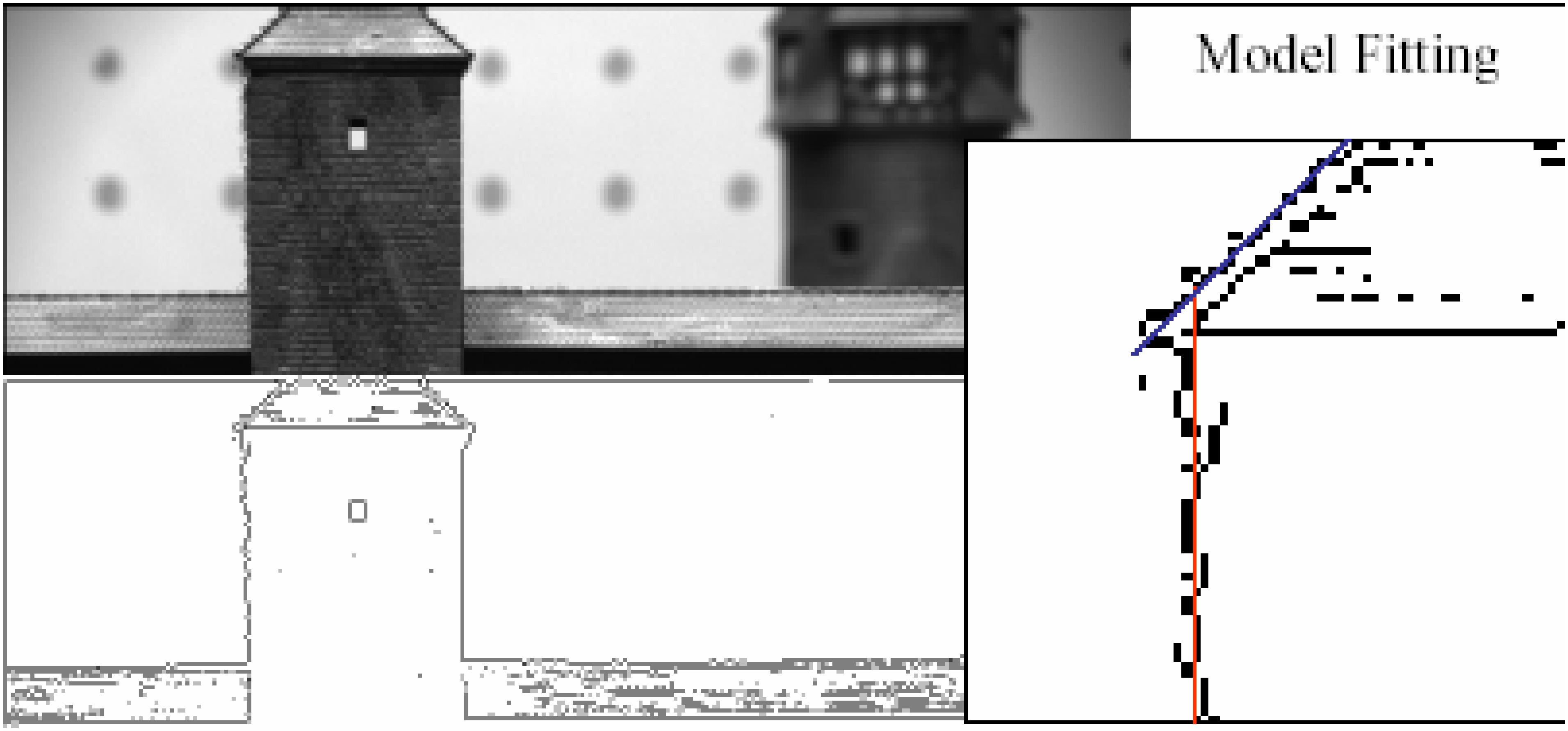
Adapted from Shapiro and Stockman

Model fitting

- Mathematical models that fit data not only reveal important structure in the data, but also can provide efficient representations for further analysis.
- Mathematical models exist for lines, circles, cylinders, and many other shapes.
- We can use the method of least squares for determining the parameters of the best mathematical model fitting observed data.



Model fitting: line segments



Adapted from Martial Hebert, CMU

Model fitting: line segments

- Given a set of observed points $\{(x_i, y_i), i = 1, \ldots, n\}$.
- A straight line can be modeled as a function with two parameters:

$$y = ax + b.$$

- To measure how well a model fits a set of n observations can be computed using the *least-squares error criteria*:

$$LSE = \sum_{i=1}^{n} (ax_i + b - y_i)^2$$

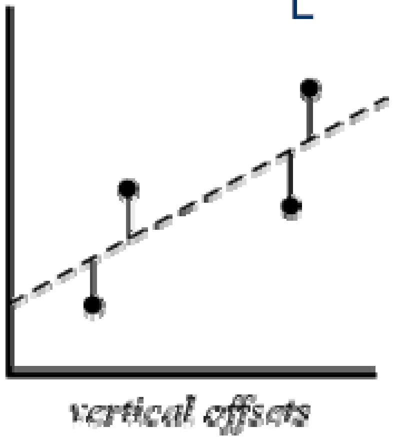
where $ax_i + b - y_i$ is the algebraic distance.

- The best model is the model with the paramaters minimizing this criteria.

Model fitting: line segments

- For the model $y = ax + b$, the parameters that minimize LSE can be found by taking partial derivatives and solving for the unknowns.
- The parameters of the best line are:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} 1 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix}.$$

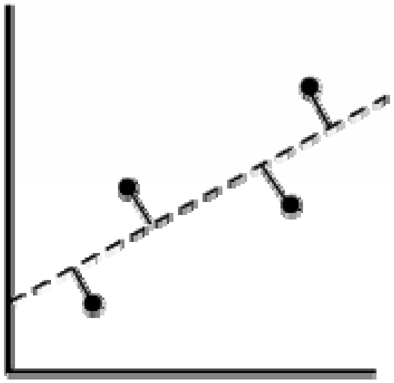


Model fitting: line segments

- If we use the geometric distance where $ax + by + c = 0$ and $a^2 + b^2 = 1$, the solution for $[a\ b]^T$ is the eigenvector corresponding to the smallest eigenvalue of

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 & \sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right) \\ \sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right) & \sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2 \end{bmatrix}$$

and $c = -a \sum_{i=1}^{n} x_i - b \sum_{i=1}^{n} y_i$.

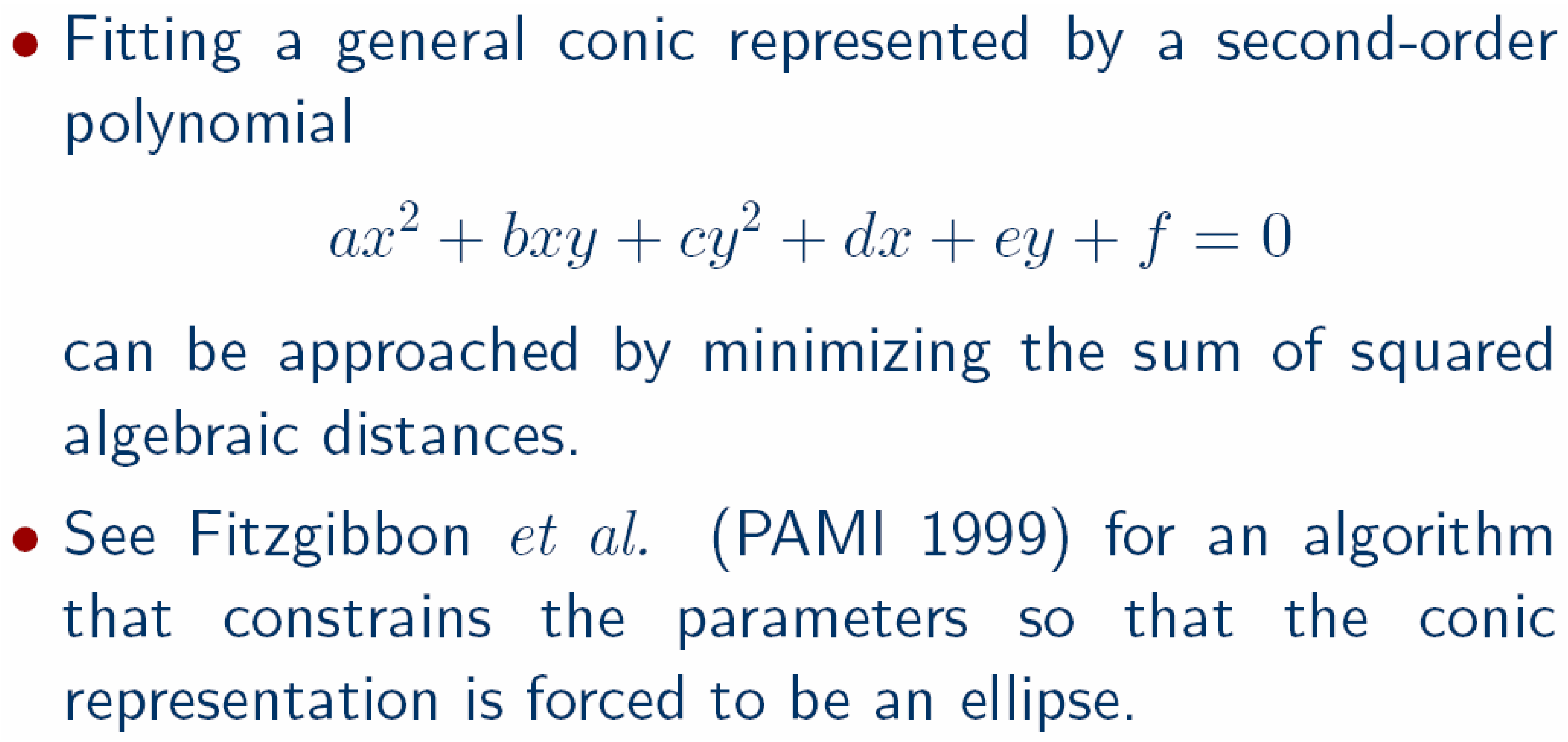


perpendicular offsets

Model fitting: line segments

- Problems in fitting:
  - Outliers
  - Error definition (algebraic vs. geometric distance)
  - Statistical interpretation of the error (hypothesis testing)
  - Nonlinear optimization
  - High dimensionality (of the data and/or the number of model parameters)
  - Additional fit constraints

Model fitting: ellipses



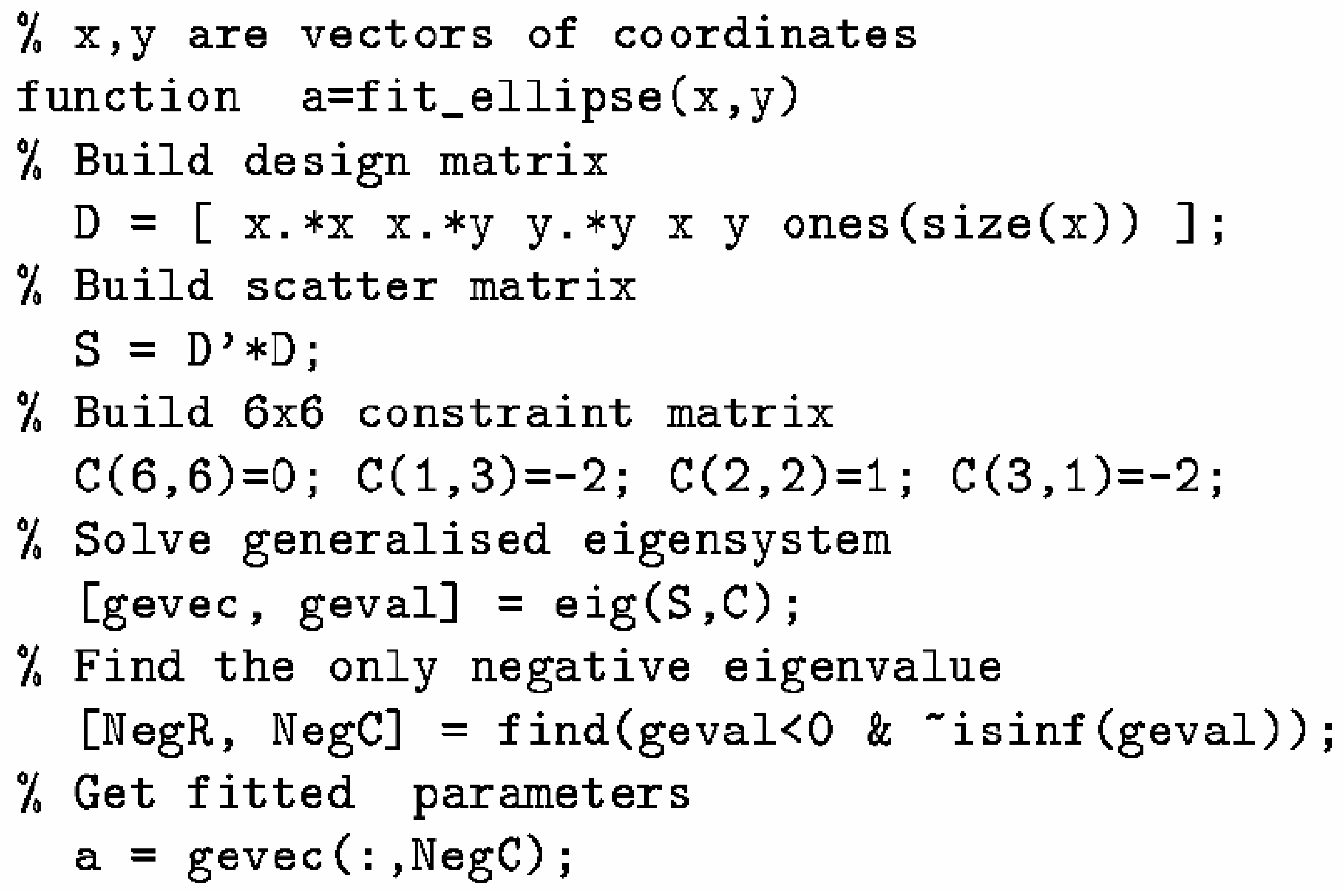
- Fitting a general conic represented by a second-order polynomial

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

can be approached by minimizing the sum of squared algebraic distances.

- See Fitzgibbon *et al.* (PAMI 1999) for an algorithm that constrains the parameters so that the conic representation is forced to be an ellipse.

Model fitting: ellipses

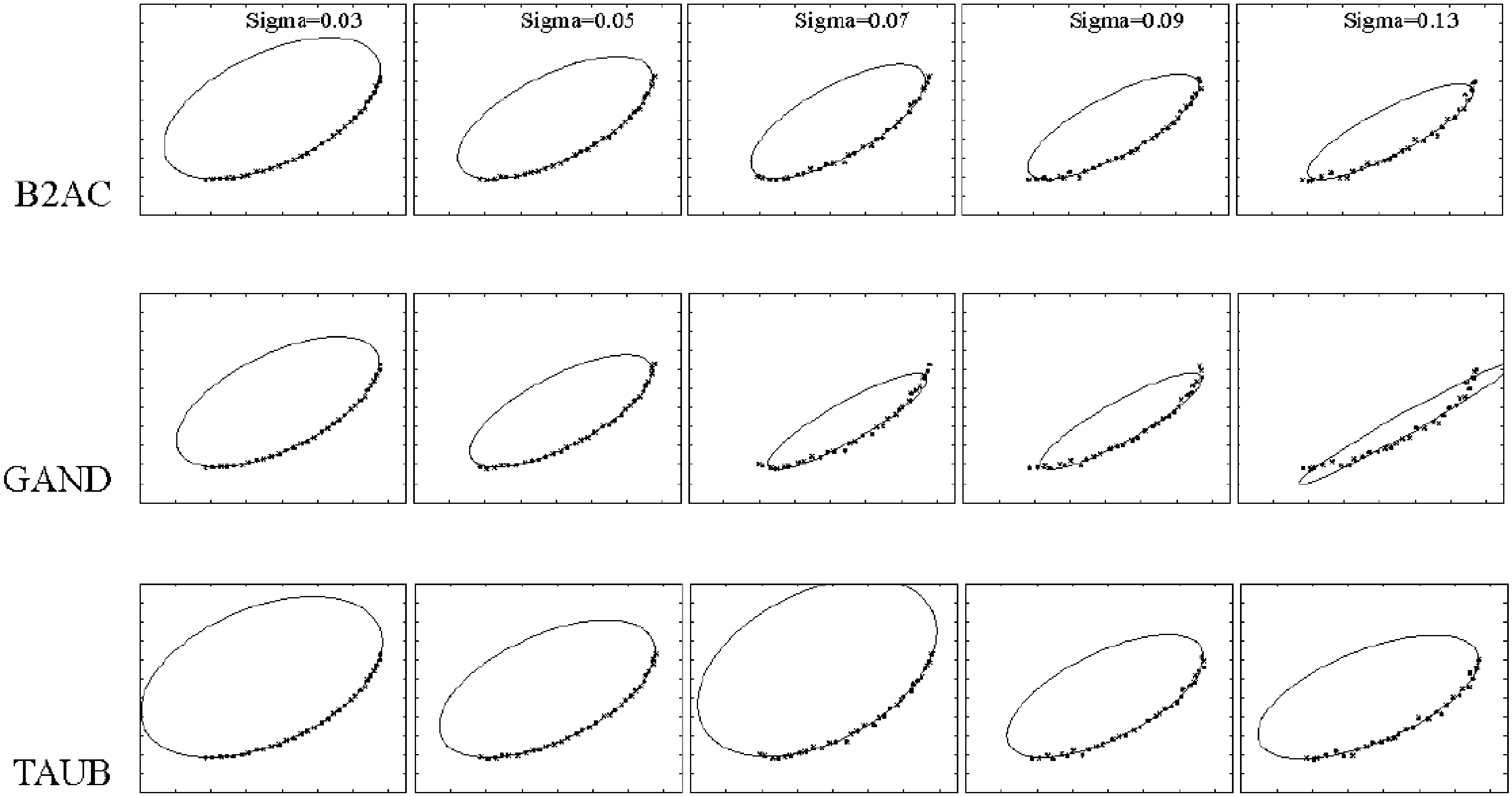


```
% x,y are vectors of coordinates
function  a=fit_ellipse(x,y)
% Build design matrix
  D = [ x.*x x.*y y.*y x y ones(size(x)) ];
% Build scatter matrix
  S = D'*D;
% Build 6x6 constraint matrix
  C(6,6)=0; C(1,3)=-2; C(2,2)=1; C(3,1)=-2;
% Solve generalised eigensystem
  [gevec, geval] = eig(S,C);
% Find the only negative eigenvalue
  [NegR, NegC] = find(geval<0 & ~isinf(geval));
% Get fitted  parameters
  a = gevec(:,NegC);
```

Simple six-line Matlab implementation of the ellipse fitting method.

Adapted from Andrew Fitzgibbon, PAMI 1999

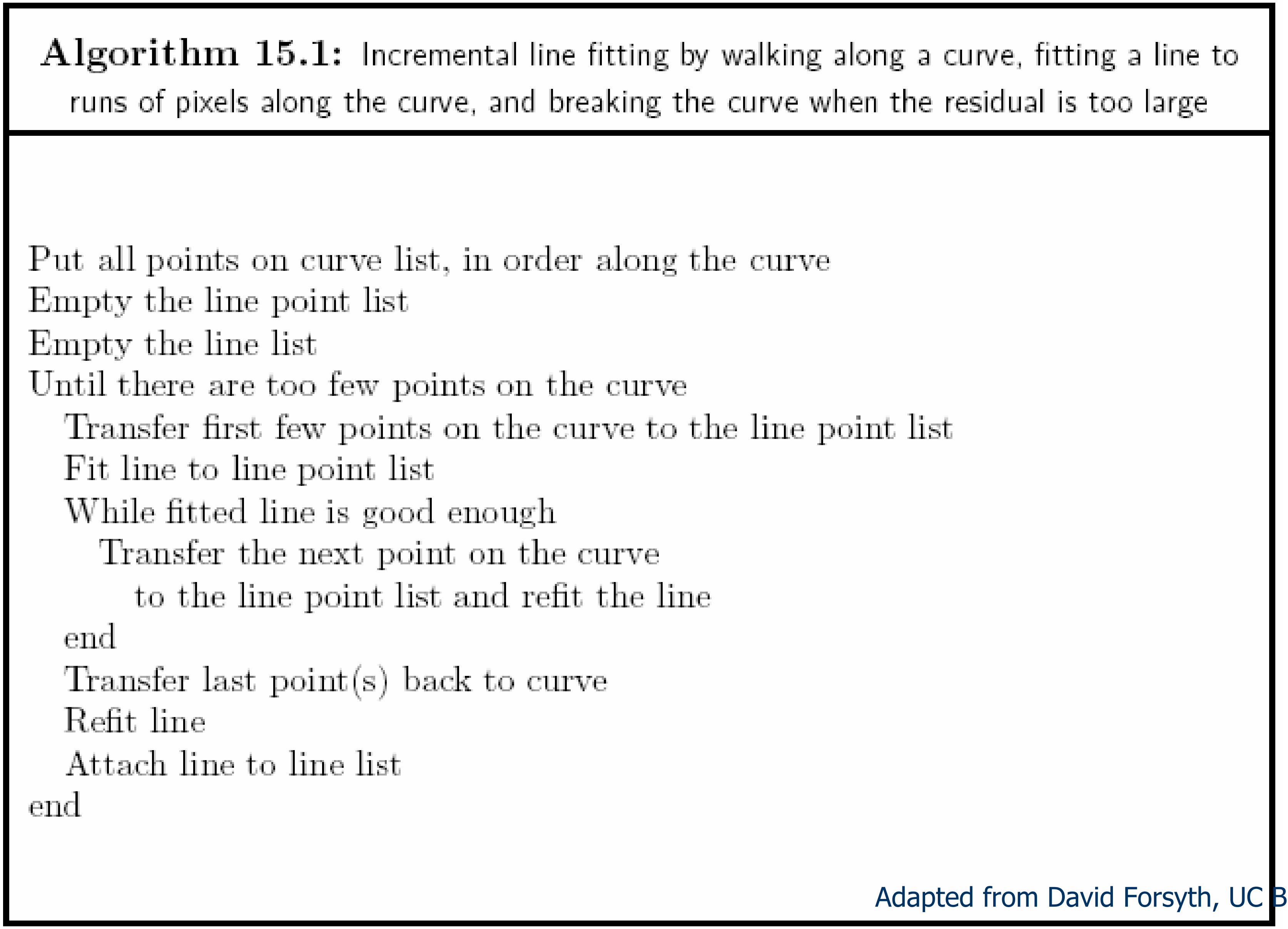
Model fitting: ellipses



Fits to arc of ellipse with increasing noise level.

Adapted from Andrew Fitzgibbon, PAMI 1999

Model fitting: incremental line fitting

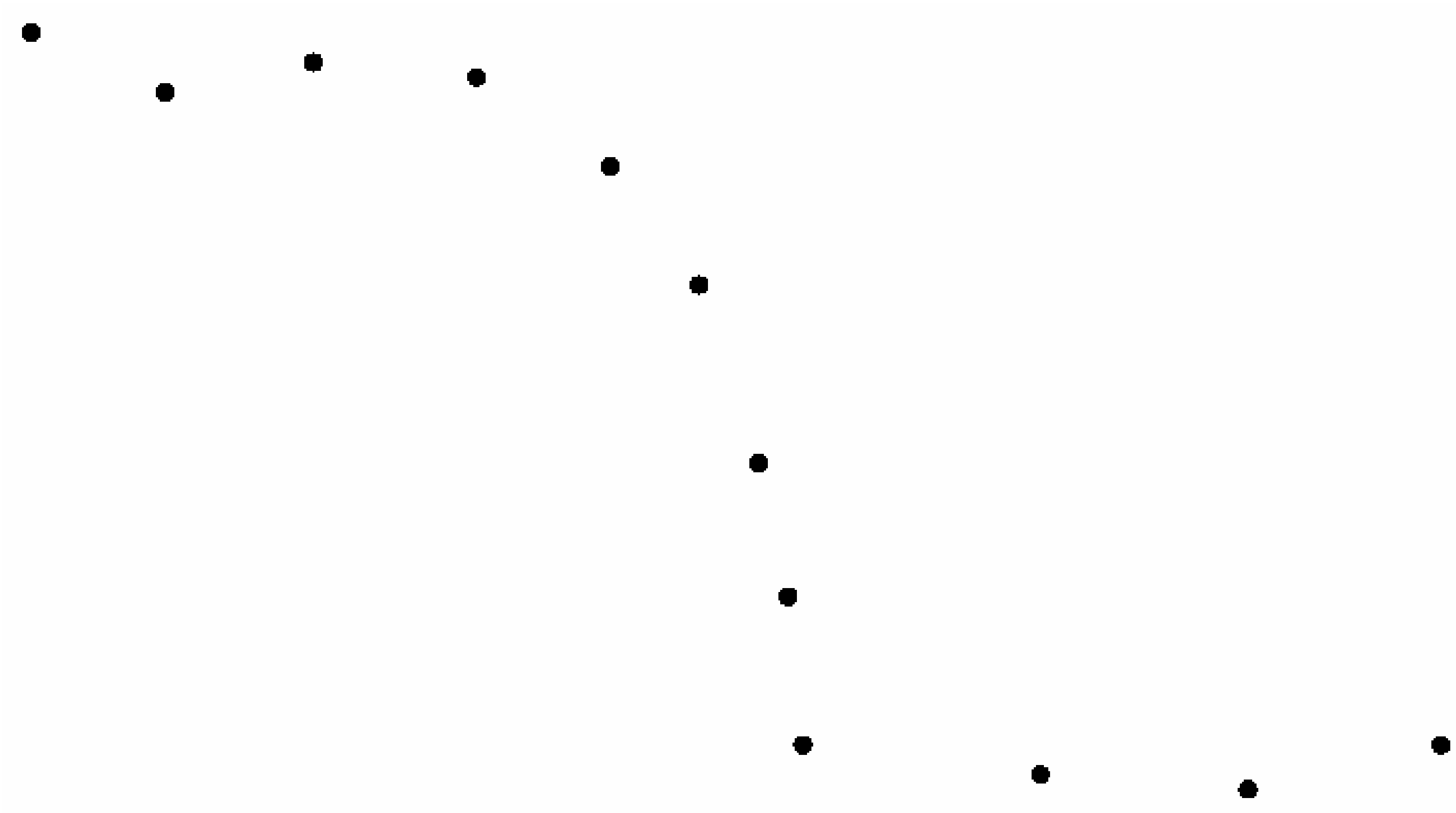


Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
   Transfer first few points on the curve to the line point list
   Fit line to line point list
   While fitted line is good enough
      Transfer the next point on the curve
         to the line point list and refit the line
   end
   Transfer last point(s) back to curve
   Refit line
   Attach line to line list
end
```

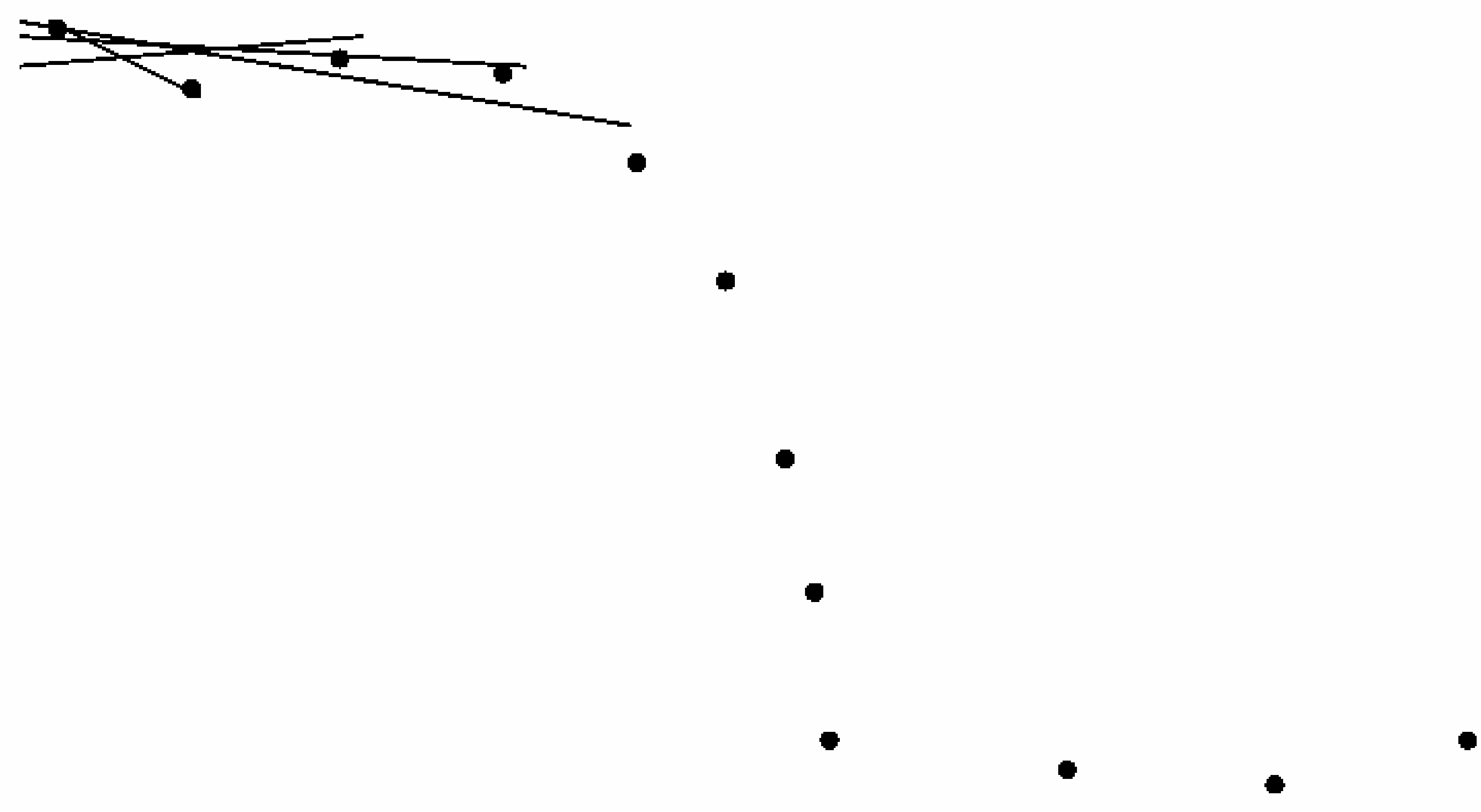
Adapted from David Forsyth, UC Berkeley

Model fitting: incremental line fitting



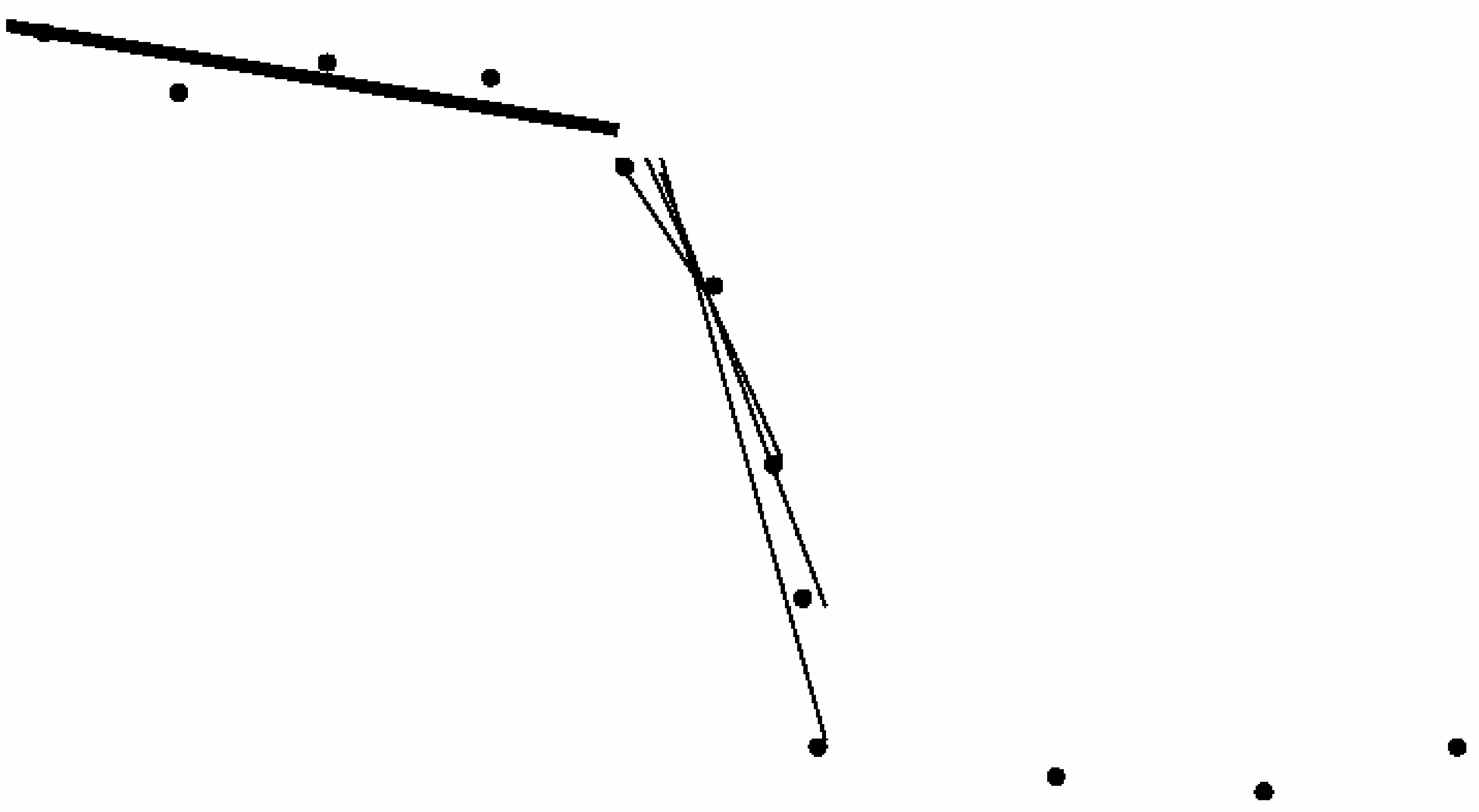
Adapted from Trevor Darrell, MIT

Model fitting: incremental line fitting



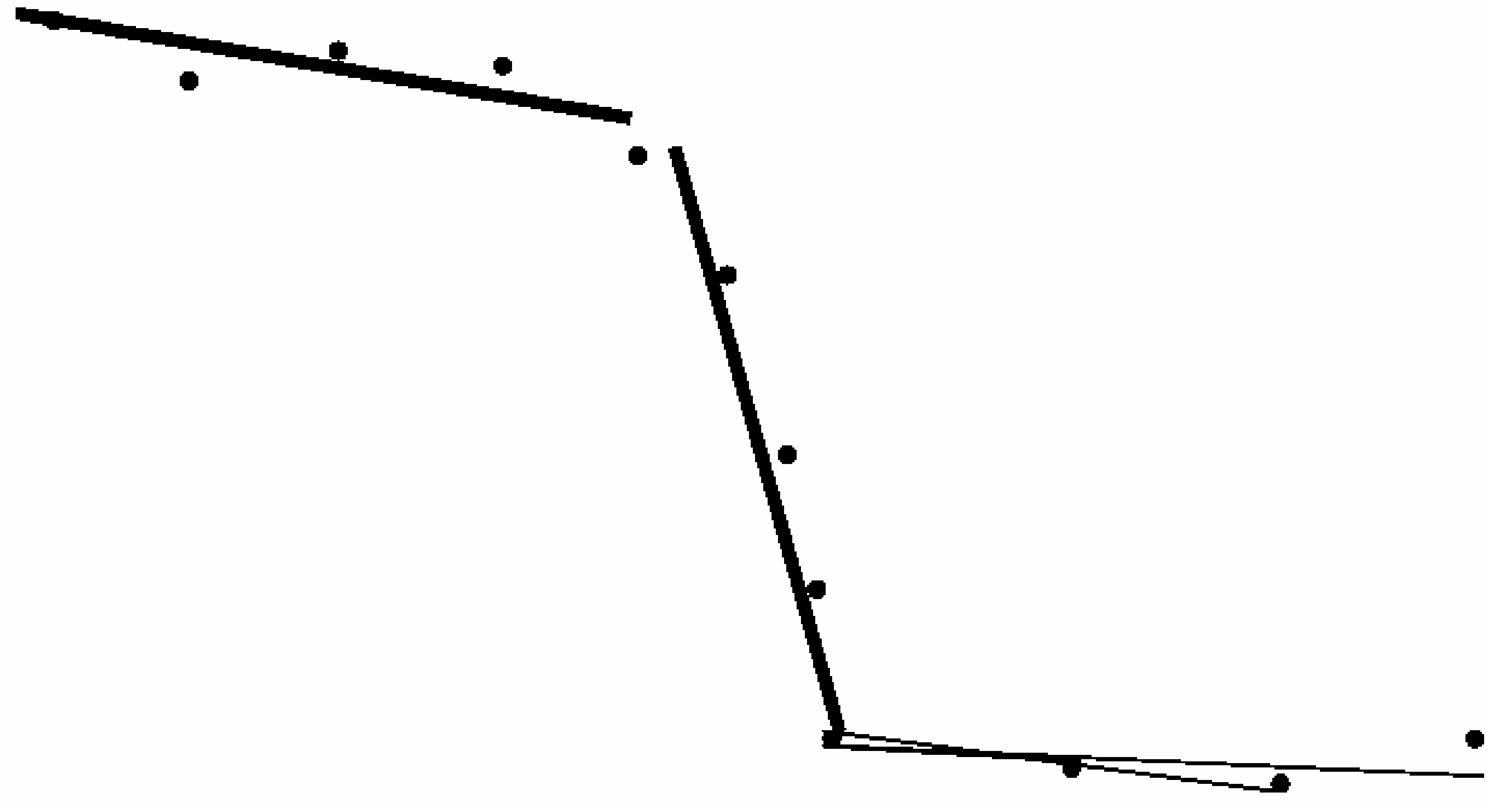
Adapted from Trevor Darrell, MIT

Model fitting: incremental line fitting



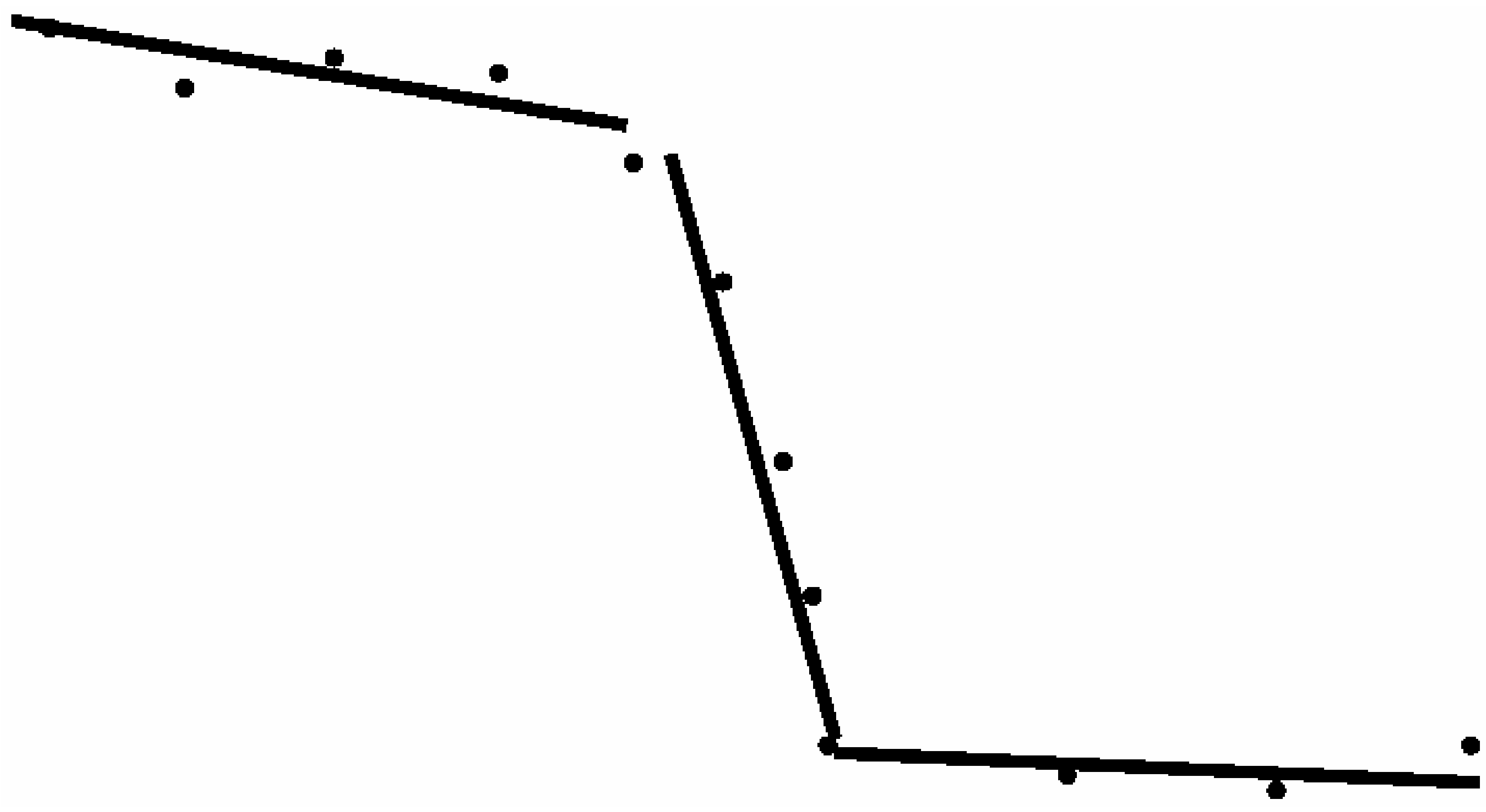
Adapted from Trevor Darrell, MIT

Model fitting: incremental line fitting



Adapted from Trevor Darrell, MIT

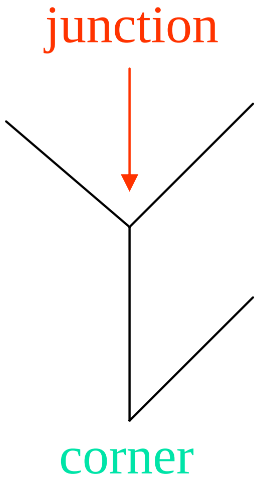
Model fitting: incremental line fitting



Adapted from Trevor Darrell, MIT

Edge tracking

- Mask-based approach uses masks to identify the following events:
  - start of a new segment,
  - interior point continuing a segment,
  - end of a segment,
  - junction between multiple segments,
  - corner that breaks a segment into two.



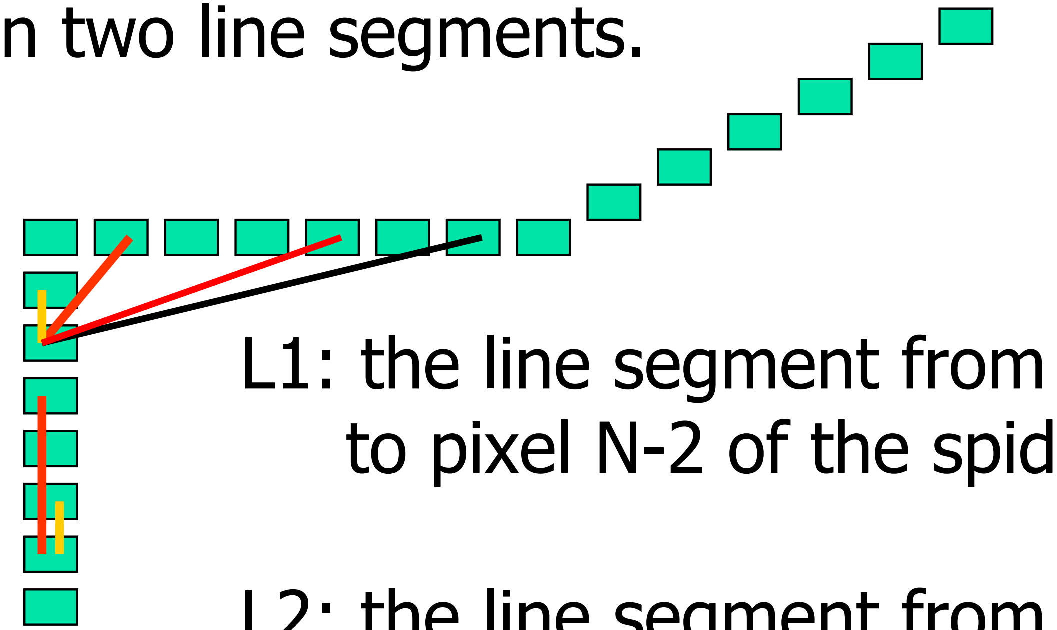
Adapted from Linda Shapiro, U of Washington

Edge tracking: ORT Toolkit

- Designed by Ata Etemadi.
- The algorithm is called Strider and is like a spider moving along pixel chains of an image, looking for junctions and corners.
- It identifies them by a measure of local asymmetry.
  - When it is moving along a straight or curved segment with no interruptions, its legs are symmetric about its body.
  - When it encounters an obstacle (i.e., a corner or junction) its legs are no longer symmetric.
  - If the obstacle is small (compared to the spider), it soon becomes symmetrical.
  - If the obstacle is large, it will take longer.
- The accuracy depends on the length of the spider and the size of its stride.
  - The larger they are, the less sensitive it becomes.

Edge tracking: ORT Toolkit

The measure of asymmetry is the angle between two line segments.



angle 0 here

L1: the line segment from pixel 1 of the spider
to pixel N-2 of the spider

L2: the line segment from pixel 1 of the spider
to pixel N of the spider

The angle must be <= arctan(2/length(L2))

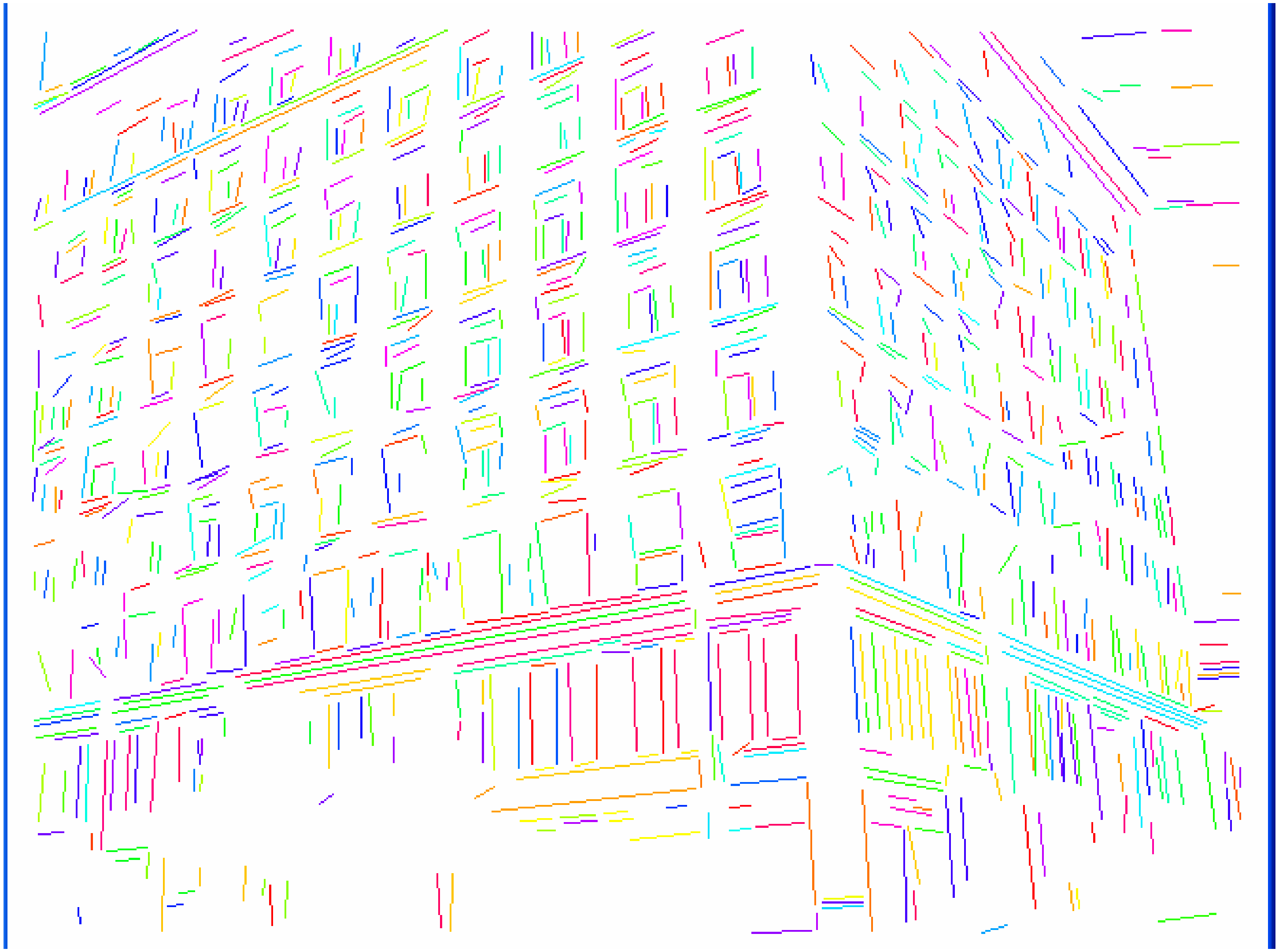
Longer spiders allow less of an angle.

Adapted from Linda Shapiro, U of Washington

Edge tracking: ORT Toolkit

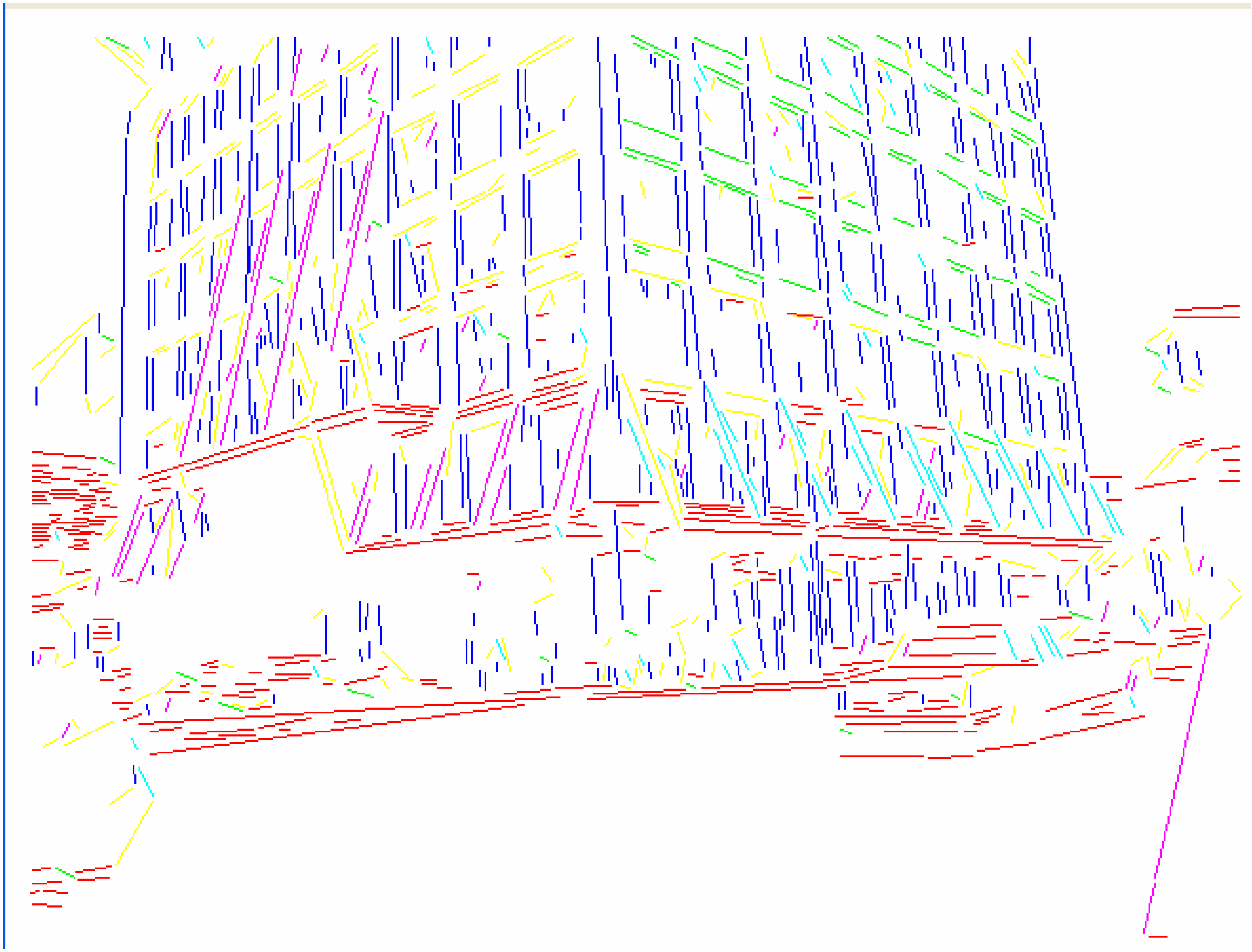
- The parameters are the length of the spider and the number of pixels per step.
- These parameters can be changed to allow for less sensitivity, so that we get longer line segments.
- The algorithm has a final phase in which adjacent segments whose angle differs by less than a given threshold are joined.
- Advantages:
  - Works on pixel chains of arbitrary complexity.
  - Can be implemented in parallel.
  - No assumptions and parameters are well understood.

Example: building detection



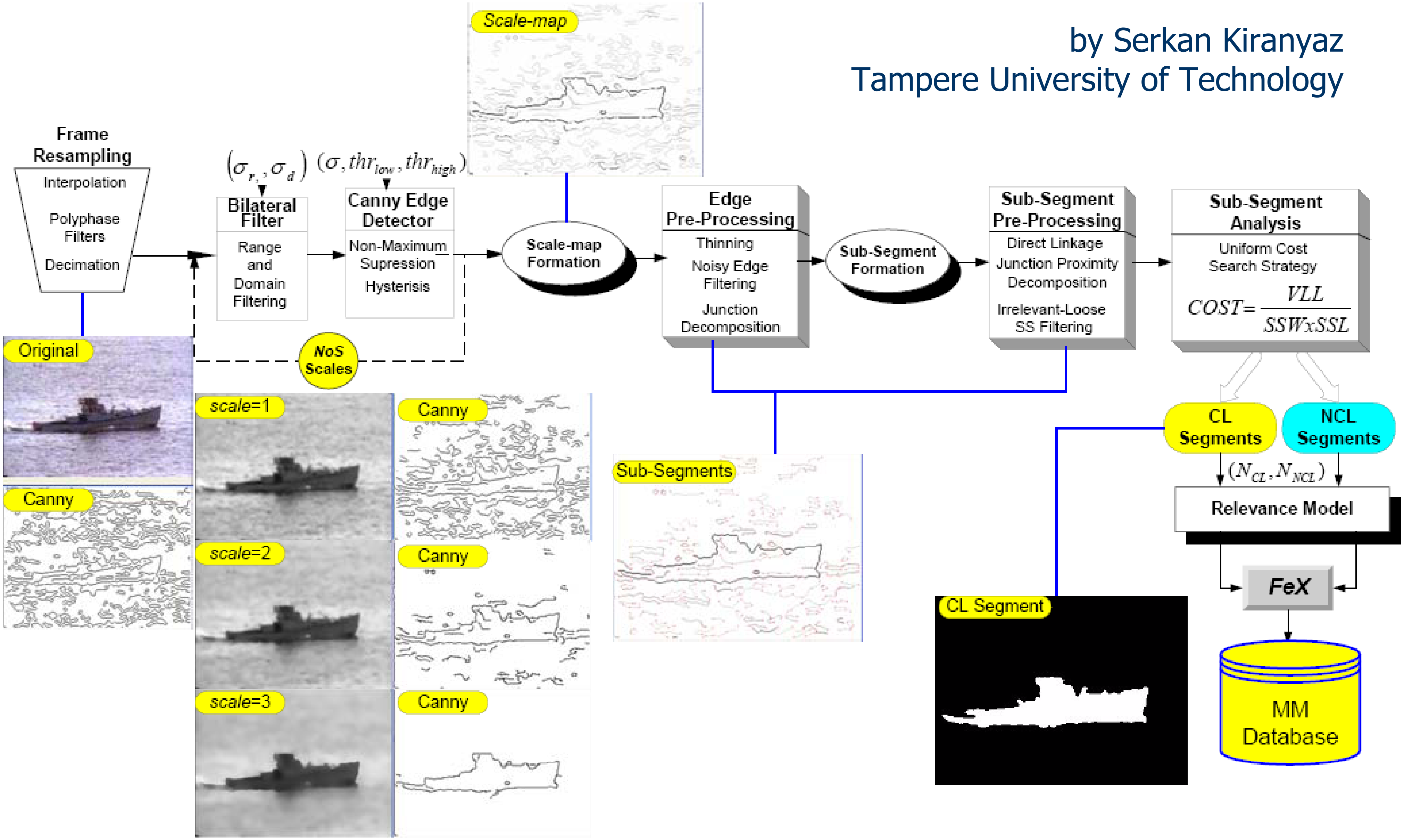
by Yi Li @ University of Washington

Example: building detection



Example: object extraction

by Serkan Kiranyaz
Tampere University of Technology



Frame Resampling: Interpolation, Polyphase Filters, Decimation

(σ_r, σ_d) Bilateral Filter: Range and Domain Filtering

$(\sigma, thr_{low}, thr_{high})$ Canny Edge Detector: Non-Maximum Supression, Hysterisis

NoS Scales

Scale-map Formation

Edge Pre-Processing: Thinning, Noisy Edge Filtering, Junction Decomposition

Sub-Segment Formation

Sub-Segment Pre-Processing: Direct Linkage, Junction Proximity Decomposition, Irrelevant-Loose SS Filtering

Sub-Segment Analysis: Uniform Cost Search Strategy

$$COST = \frac{VLL}{SSWxSSL}$$

CL Segments, NCL Segments (N_{CL}, N_{NCL})

Relevance Model

FeX

MM Database

Example: object extraction

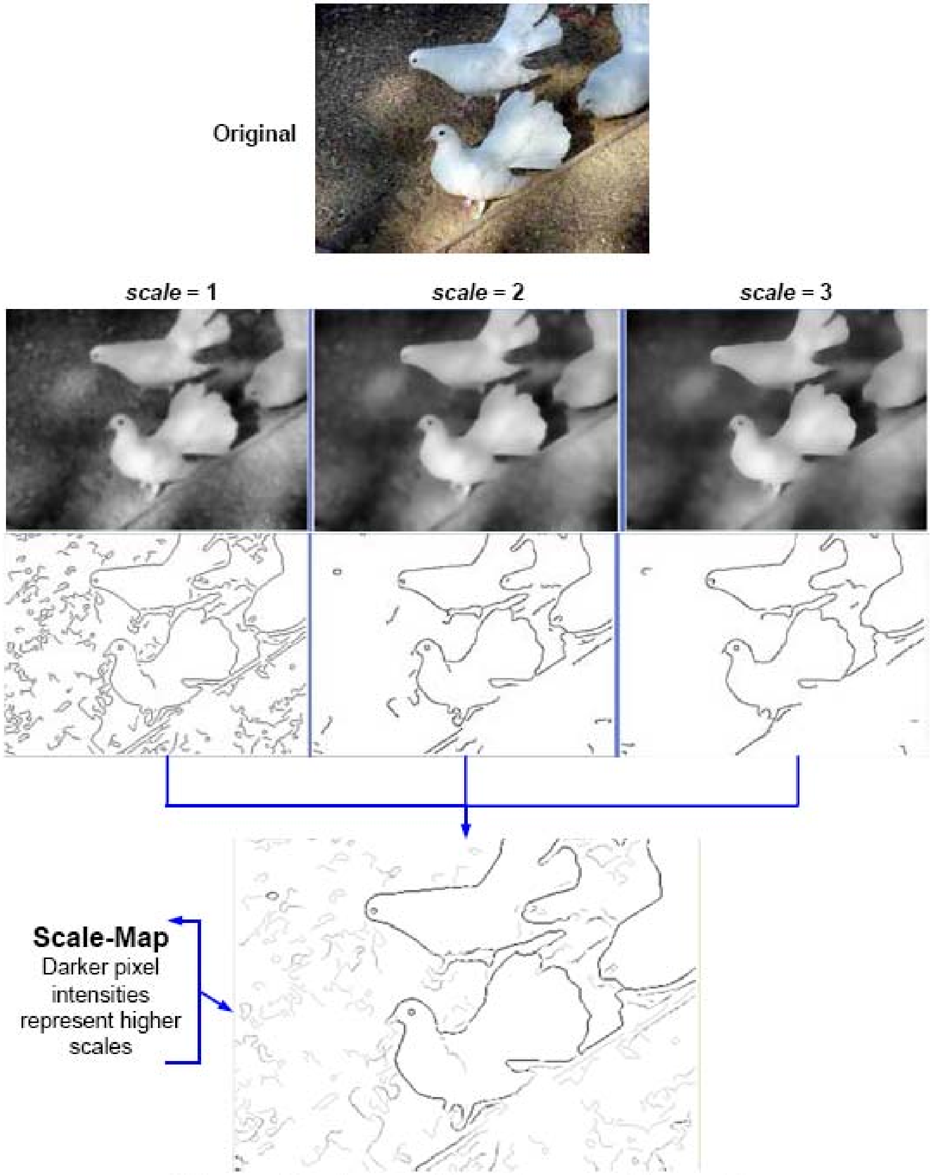


Figure 5: A sample scale-map formation.

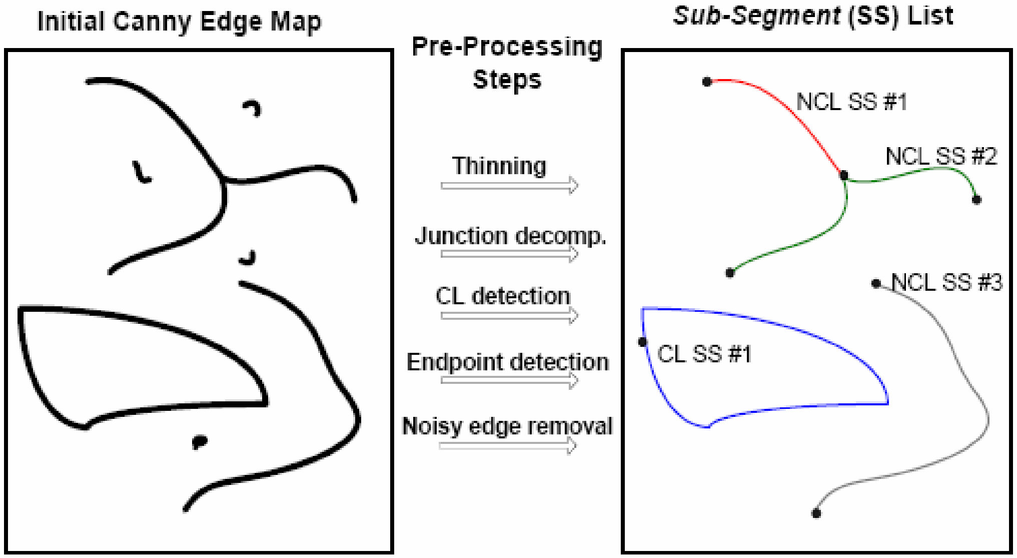


Figure 6: Sub-segment formation from an initial Canny edge field.

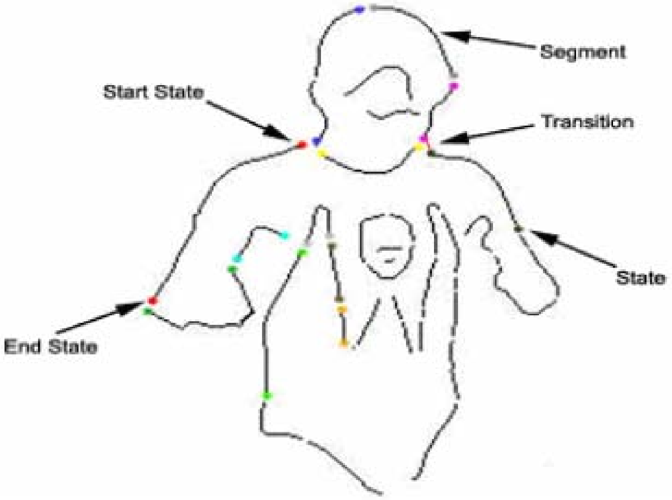


Figure 9: State space for a given sub-segment layout.

Example: object extraction

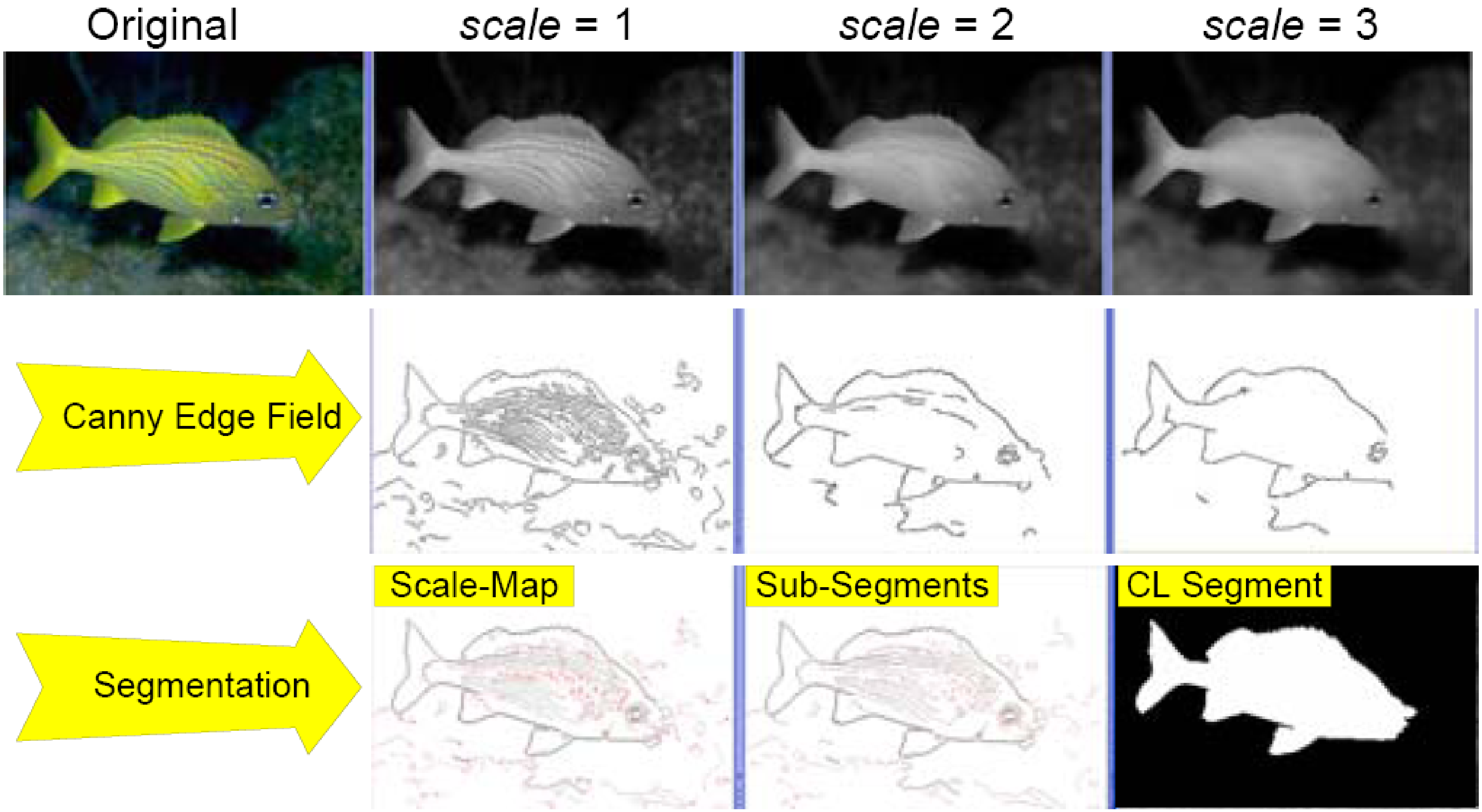
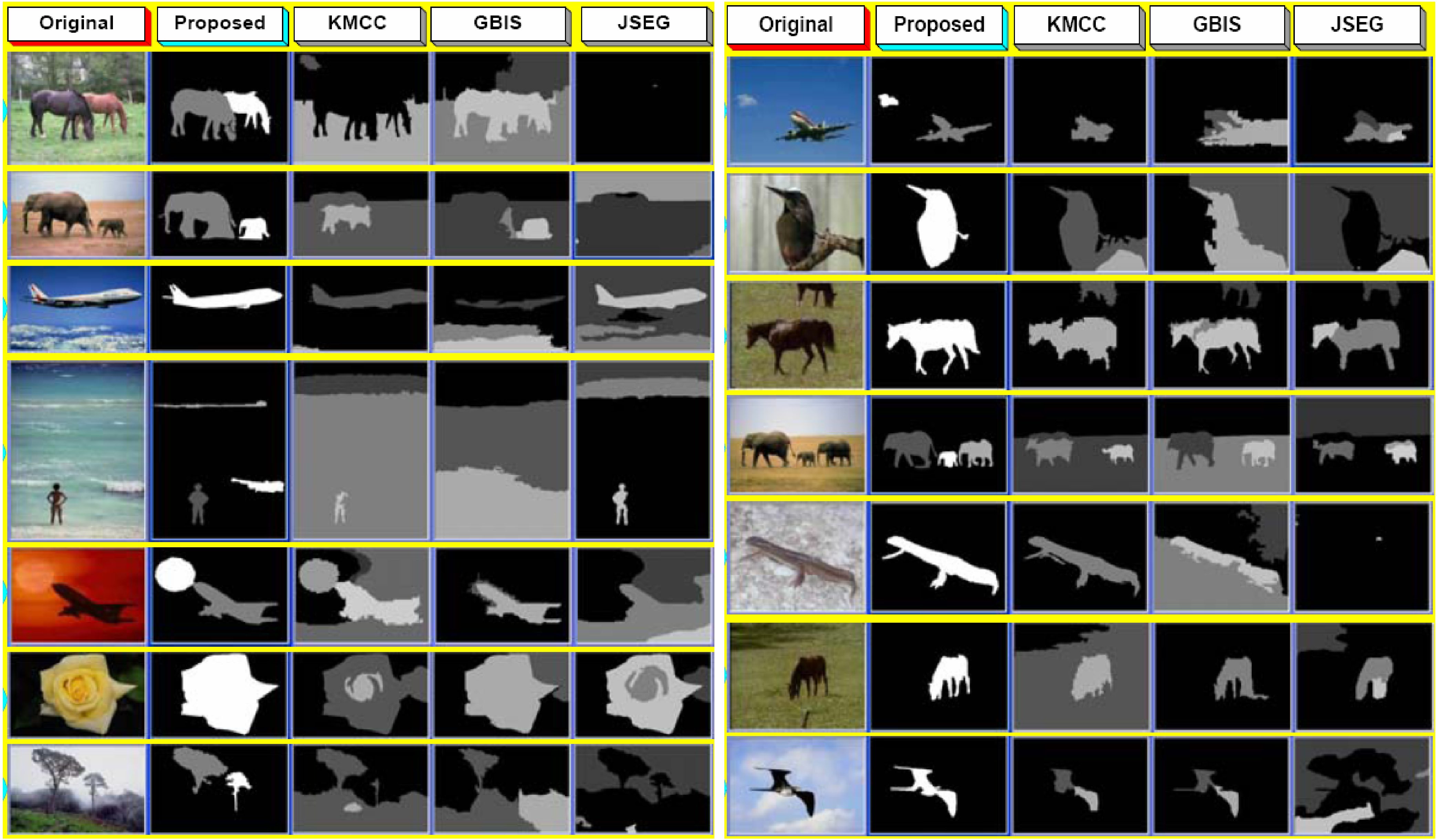


Figure 12: 3-scale simplification process over a natural image and the final CL segment extracted.

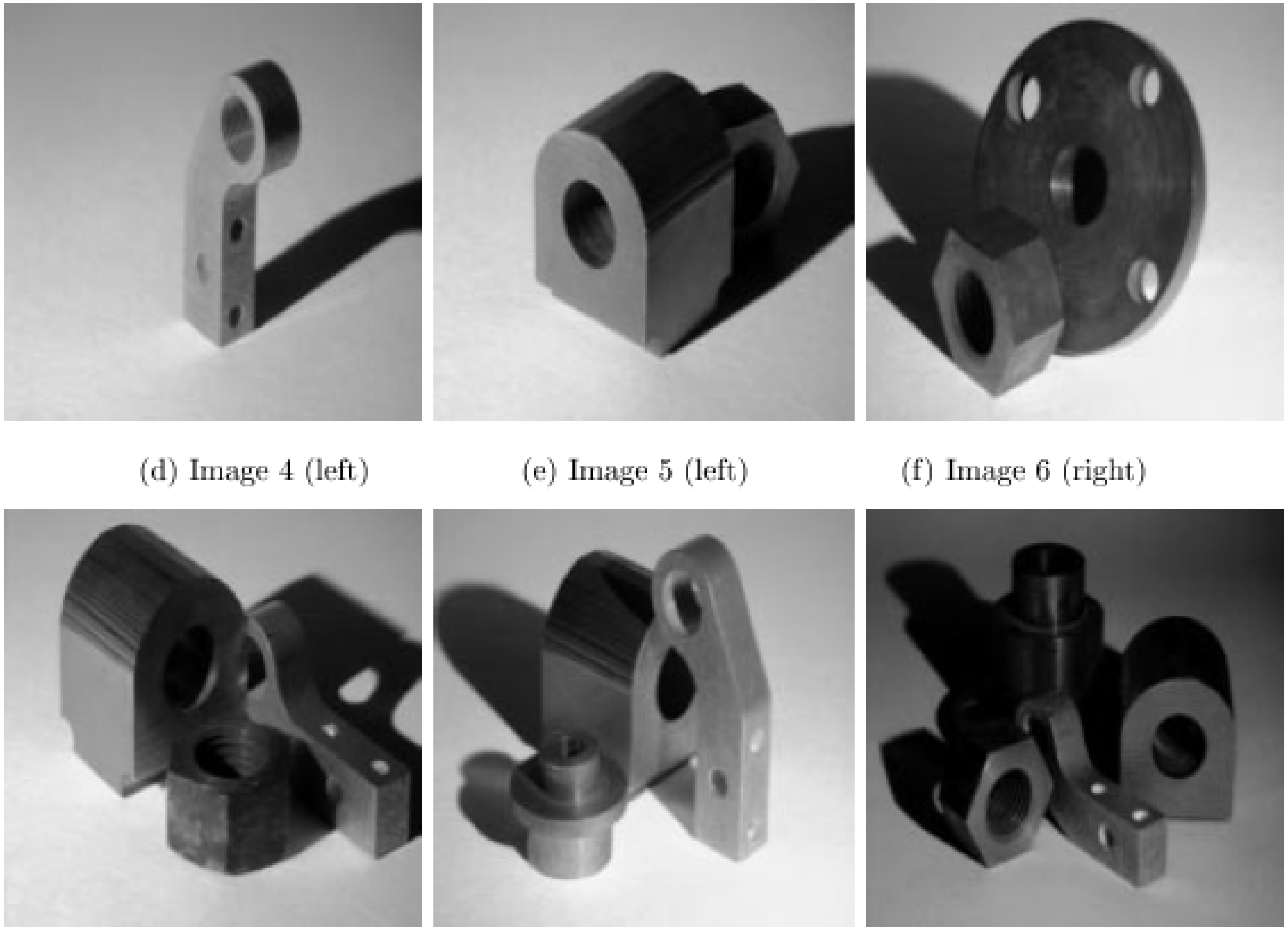
Example: object extraction



Example: object recognition

- Mauro Costa's dissertation at the University of Washington for recognizing 3D objects having planar, cylindrical, and threaded surfaces:
  - Detects edges from two intensity images.
  - From the edge image, finds a set of high-level features and their relationships.
  - Hypothesizes a 3D model using relational indexing.
  - Estimates the pose of the object using point pairs, line segment pairs, and ellipse/circle pairs.
  - Verifies the model after projecting to 2D.

Example: object recognition



(d) Image 4 (left)

(e) Image 5 (left)

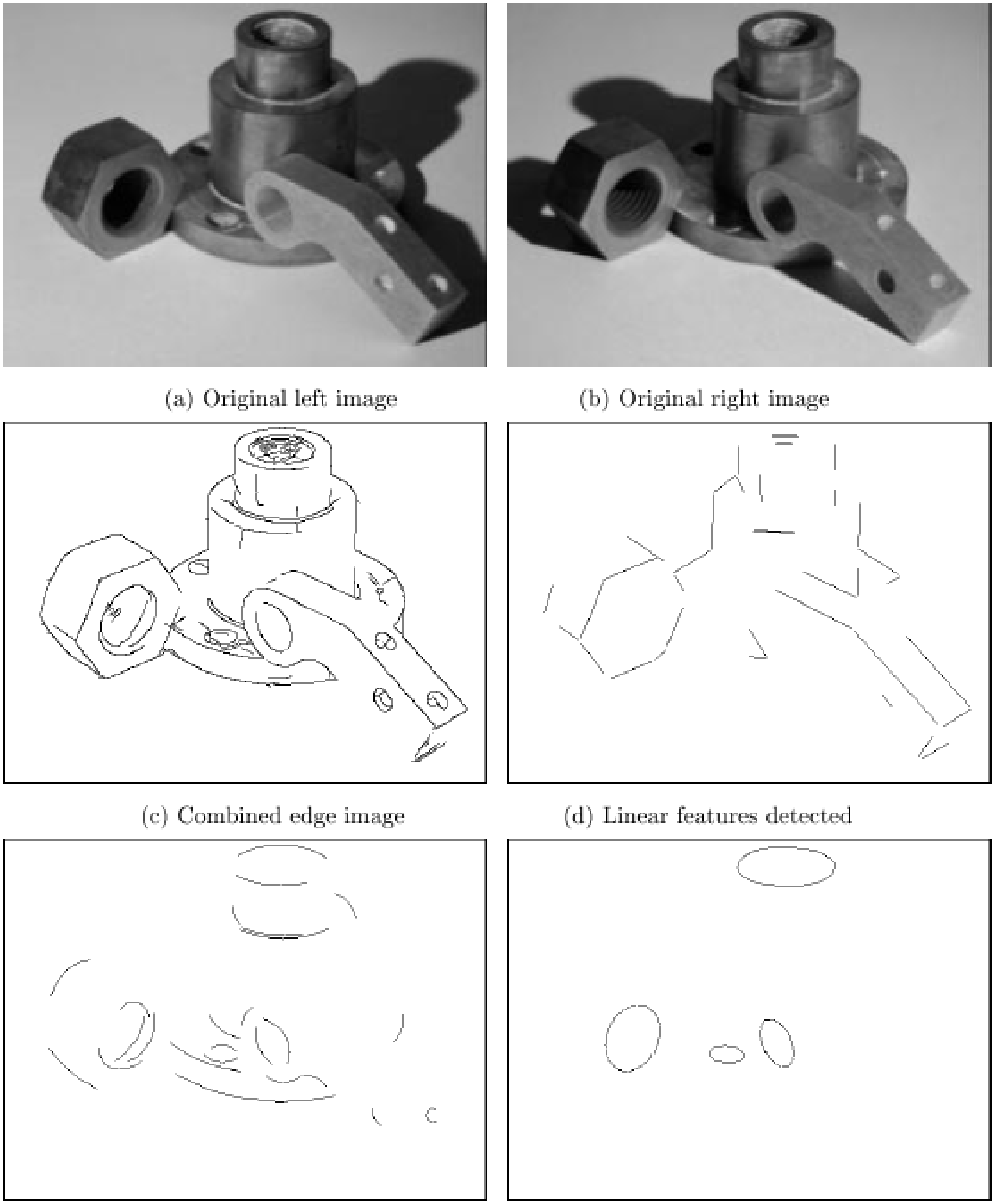
(f) Image 6 (right)

(g) Image 7 (left)

(h) Image 8 (right)

(i) Image 9 (right)

Example scenes used. The labels “left” and “right” indicate the direction of the light source.



(a) Original left image

(b) Original right image

(c) Combined edge image

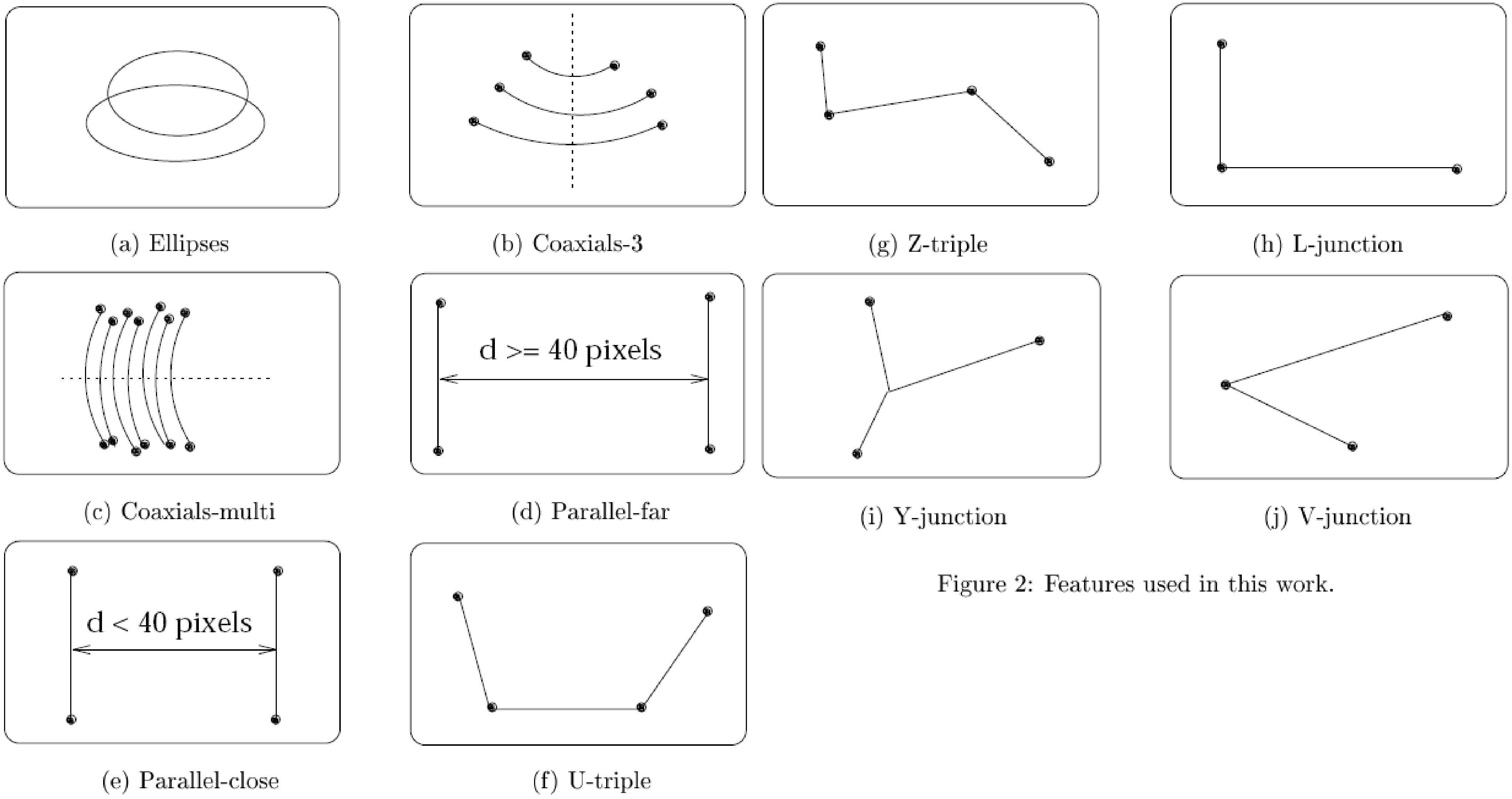
(d) Linear features detected

(e) Circular arc features detected

(f) Ellipses detected

Figure 22: Sample run of the system.

Example: object recognition



(a) Ellipses

(b) Coaxials-3

(g) Z-triple

(h) L-junction

(c) Coaxials-multi

(d) Parallel-far

(i) Y-junction

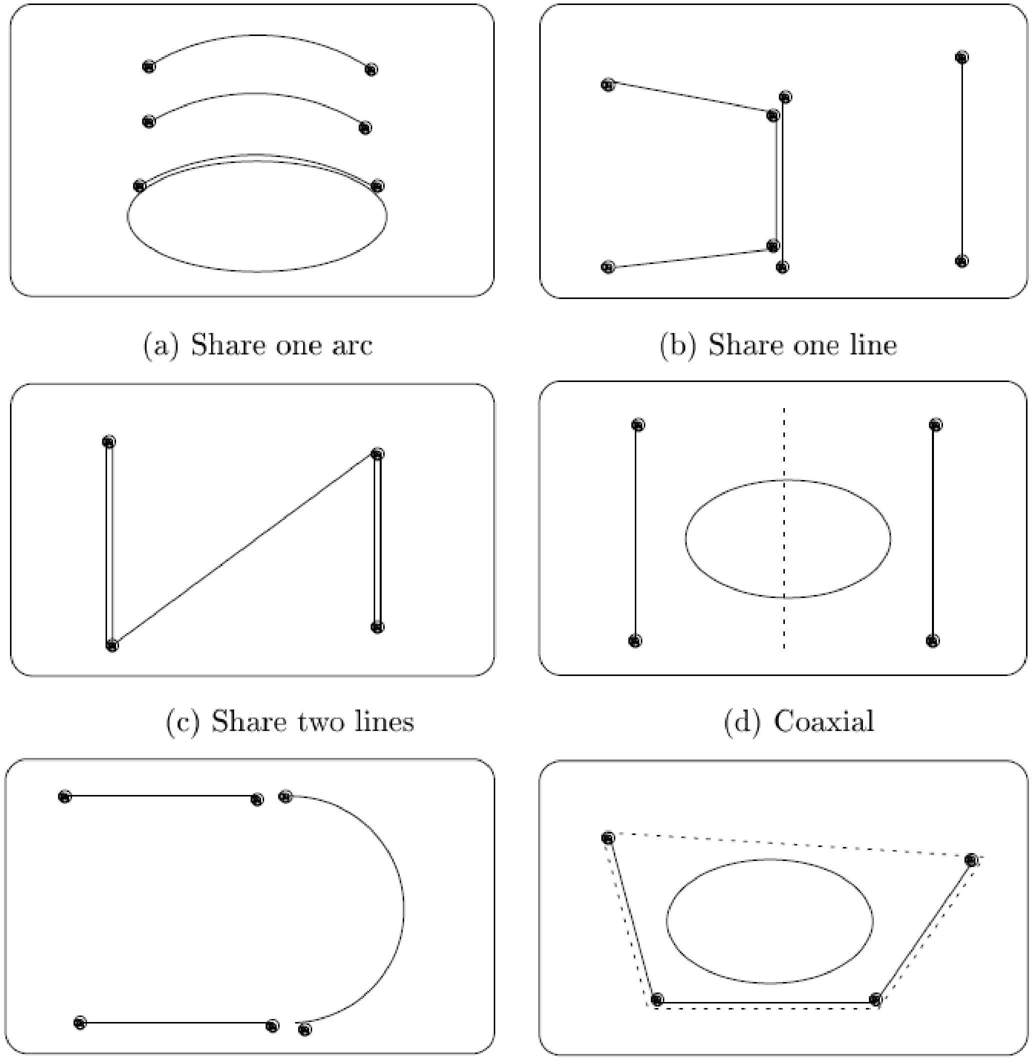
(j) V-junction

(e) Parallel-close

(f) U-triple

Figure 2: Features used in this work.

Example: object recognition



(a) Share one arc

(b) Share one line

(c) Share two lines

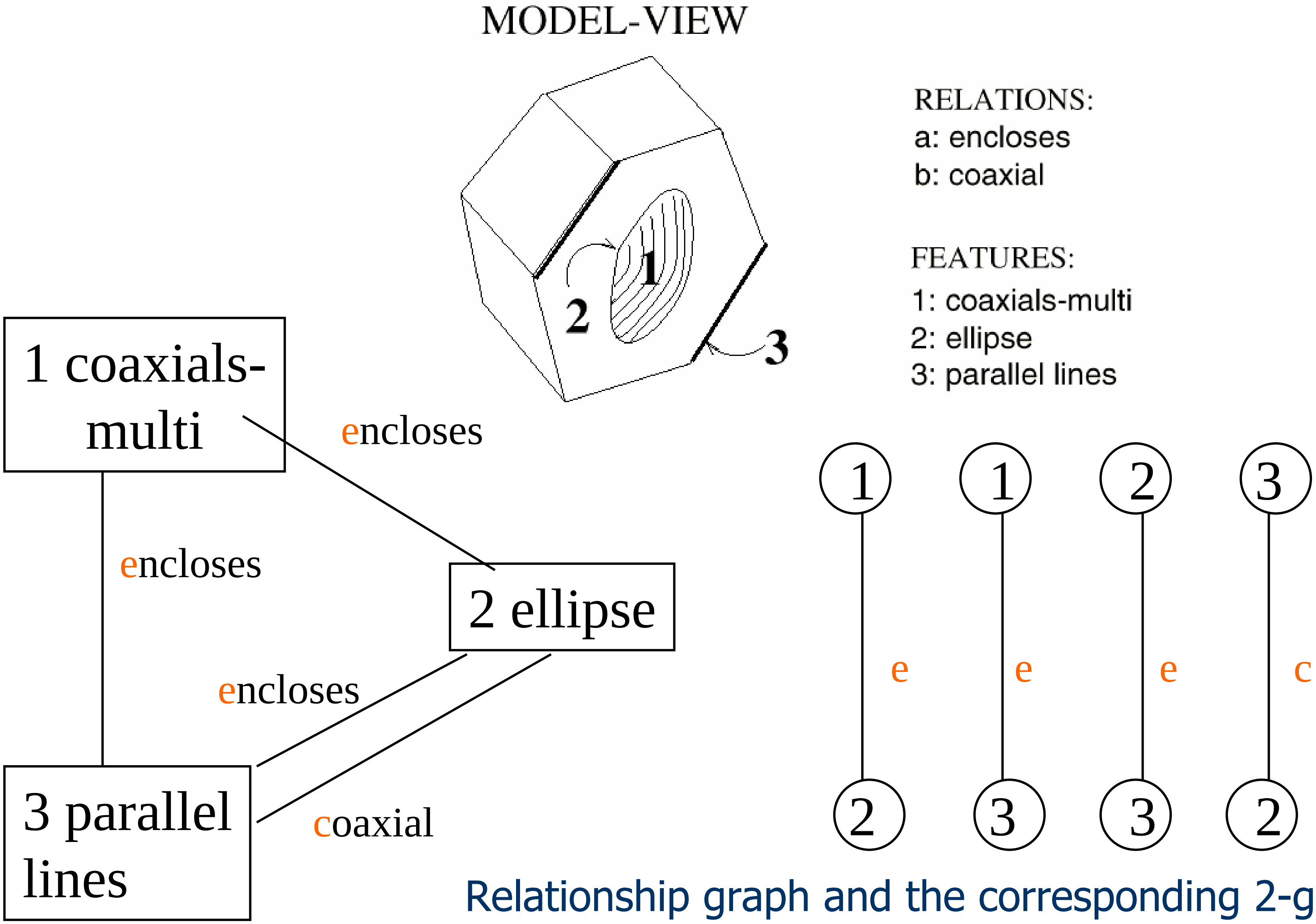
(d) Coaxial

(e) Close at extremal points

(f) Bounding box encloses/ is enclosed by bounding box

Figure 3: Relations between sample pairs of features.

Example: object recognition



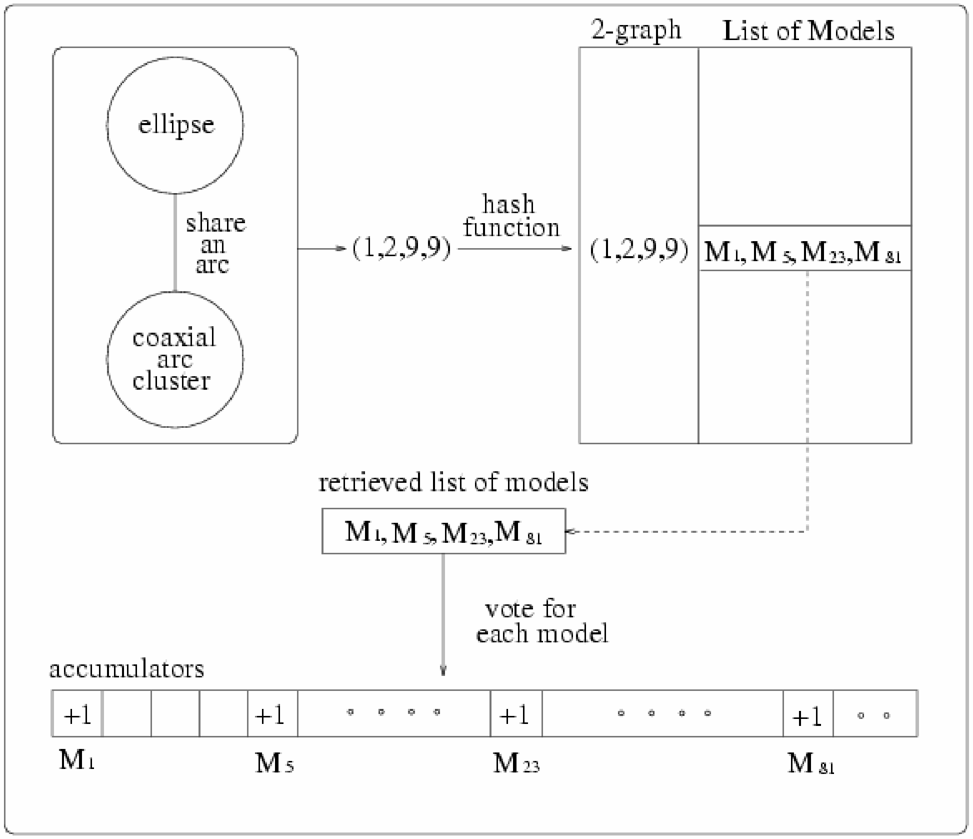
RELATIONS:
a: encloses
b: coaxial

FEATURES:
1: coaxials-multi
2: ellipse
3: parallel lines

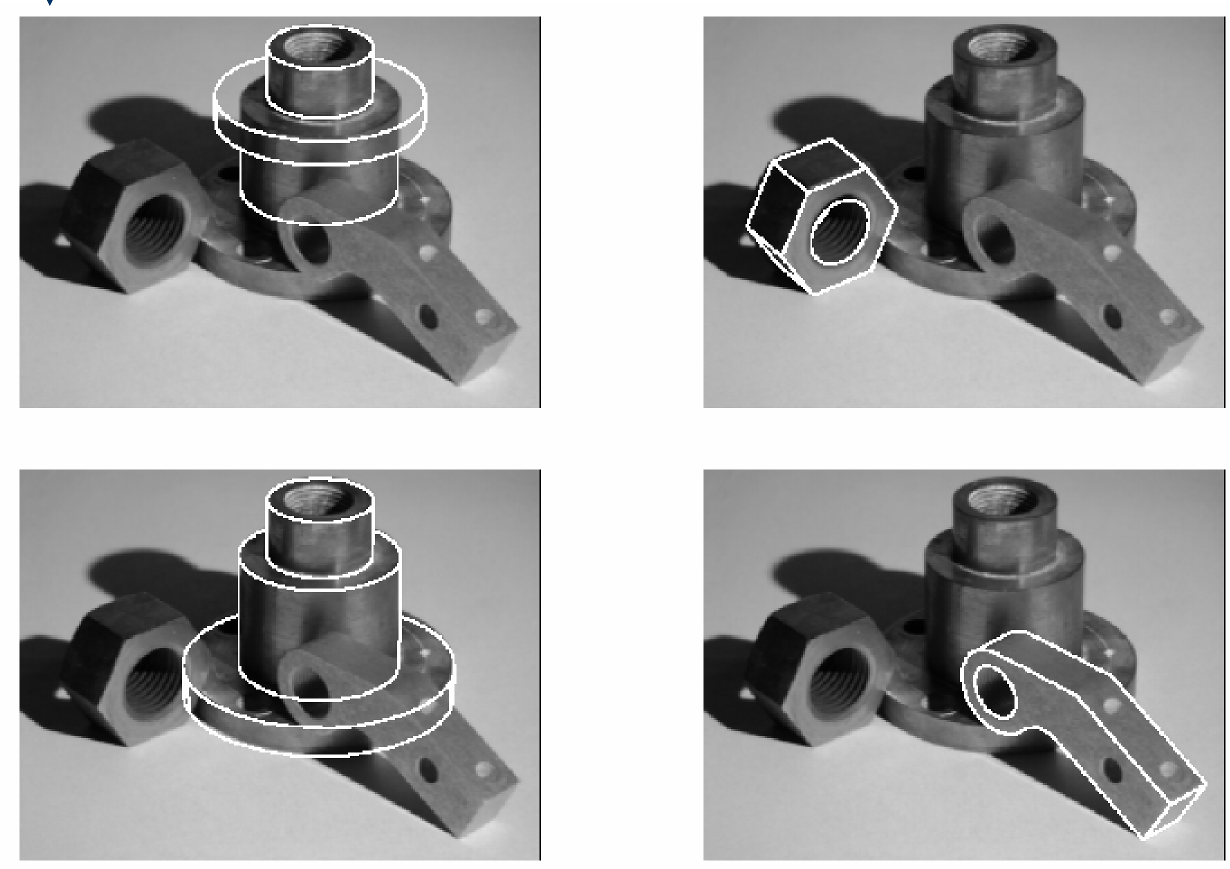
Relationship graph and the corresponding 2-graphs.

Example: object recognition

- Learning phase: relational indexing by encoding each 2-graph and storing in a hash table.
- Matching phase: voting by each 2-graph observed in the image.



Example: object recognition



Incorrect hypothesis

1. The matched features of the hypothesized object are used to determine its **pose**.
2. The **3D mesh** of the object is used to project all its features onto the image.
3. A **verification procedure** checks how well the object features line up with edges on the image.