Edge Detection

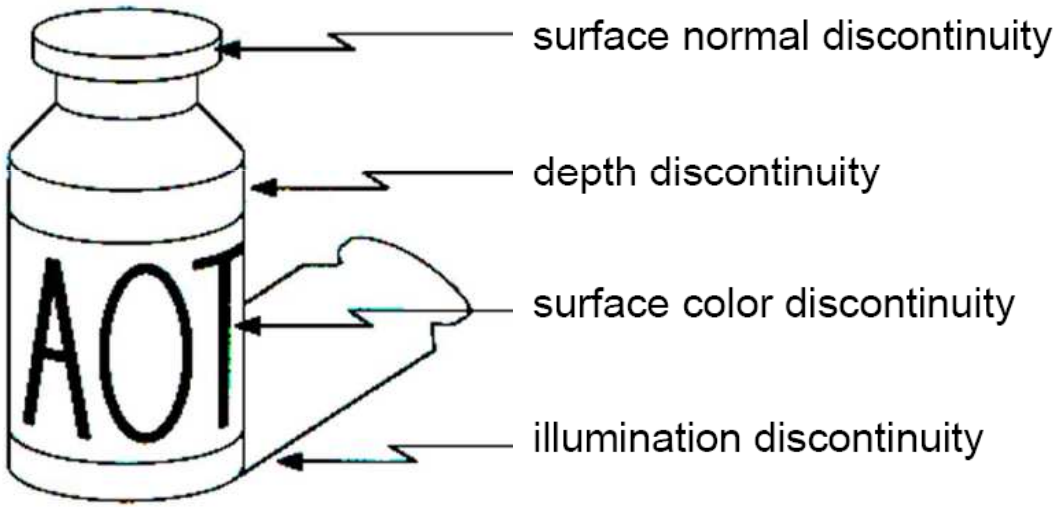
Selim Aksoy
Department of Computer Engineering
Bilkent University
saksoy@cs.bilkent.edu.tr

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.
- Further processing of edges into lines, curves and circular arcs result in useful features for matching and recognition.
- 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



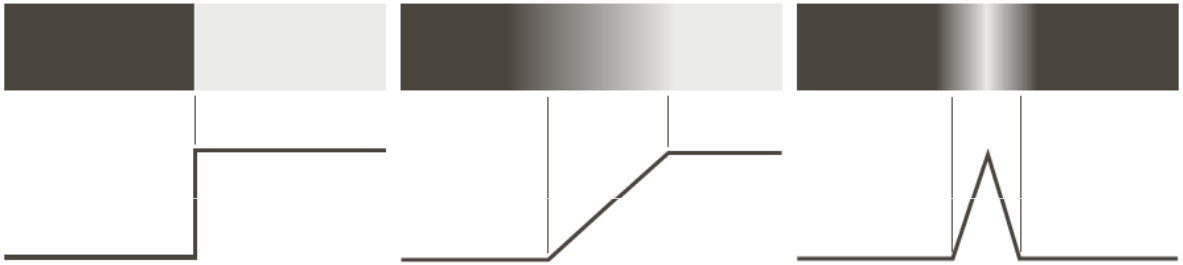
Edge detection

- 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.

Edge models



a b c

FIGURE 10.8
From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

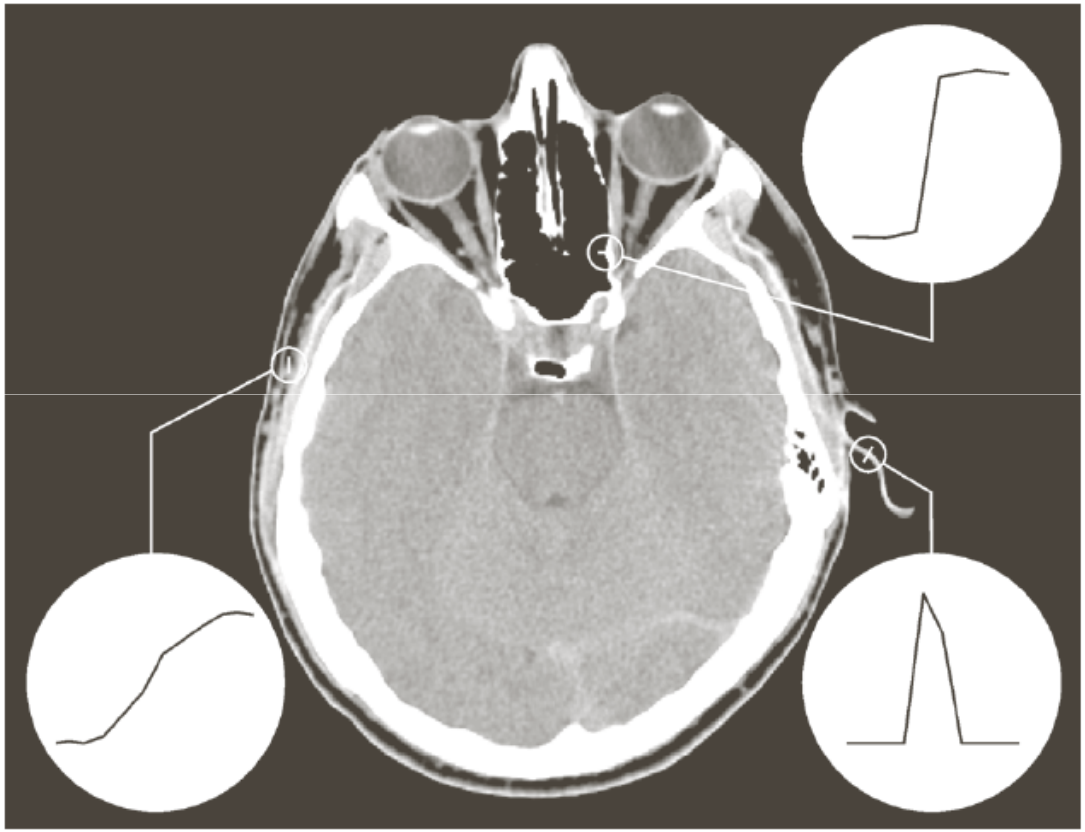


FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and "step" profiles span 9 pixels and 2 pixels, respectively.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

Difference operators for 1D

mask M = [−1, 0, 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 **M** sum to zero, output must be zero on a constant region.

Adapted from Shapiro and Stockman

Difference operators for 1D

mask M $= [-1, 2, -1]$

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 $\mathbf{M} = [-1, 2, -1]$; (a) upward step edge, (b) downward step edge, (c) upward ramp, and (d) bright impulse. Since the coordinates of **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

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.

Difference operators under noise

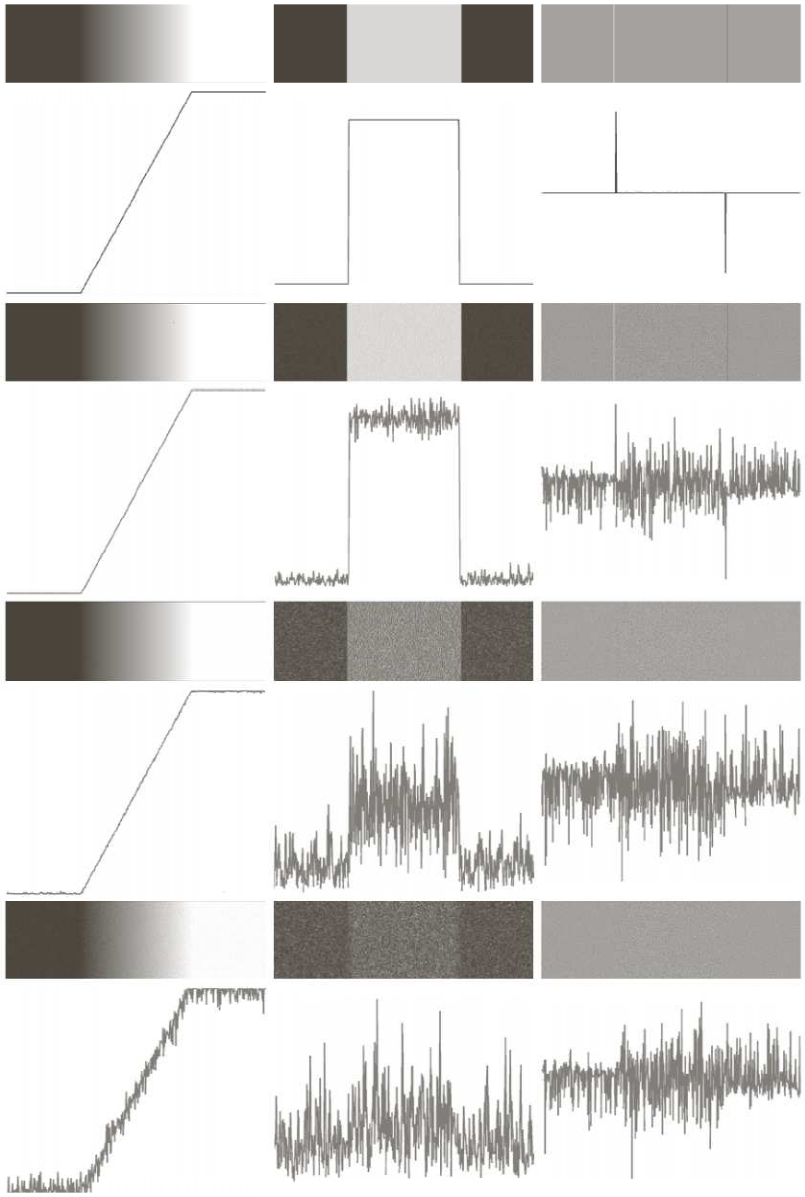


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images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

Adapted from Gonzales and Woods

Difference operators under noise

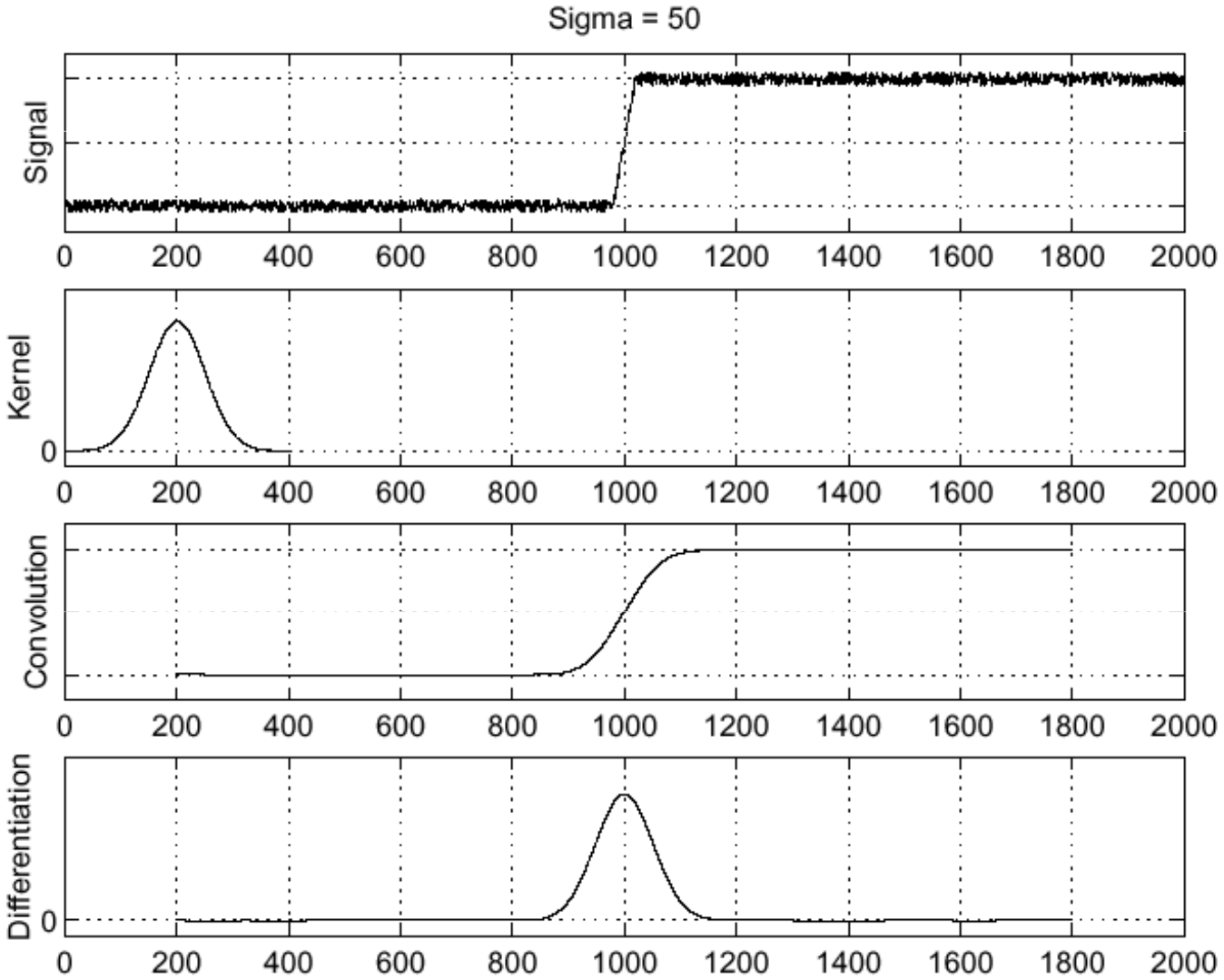
Solution is to smooth first:

f

h

$h \star f$

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



Adapted from Steve Seitz

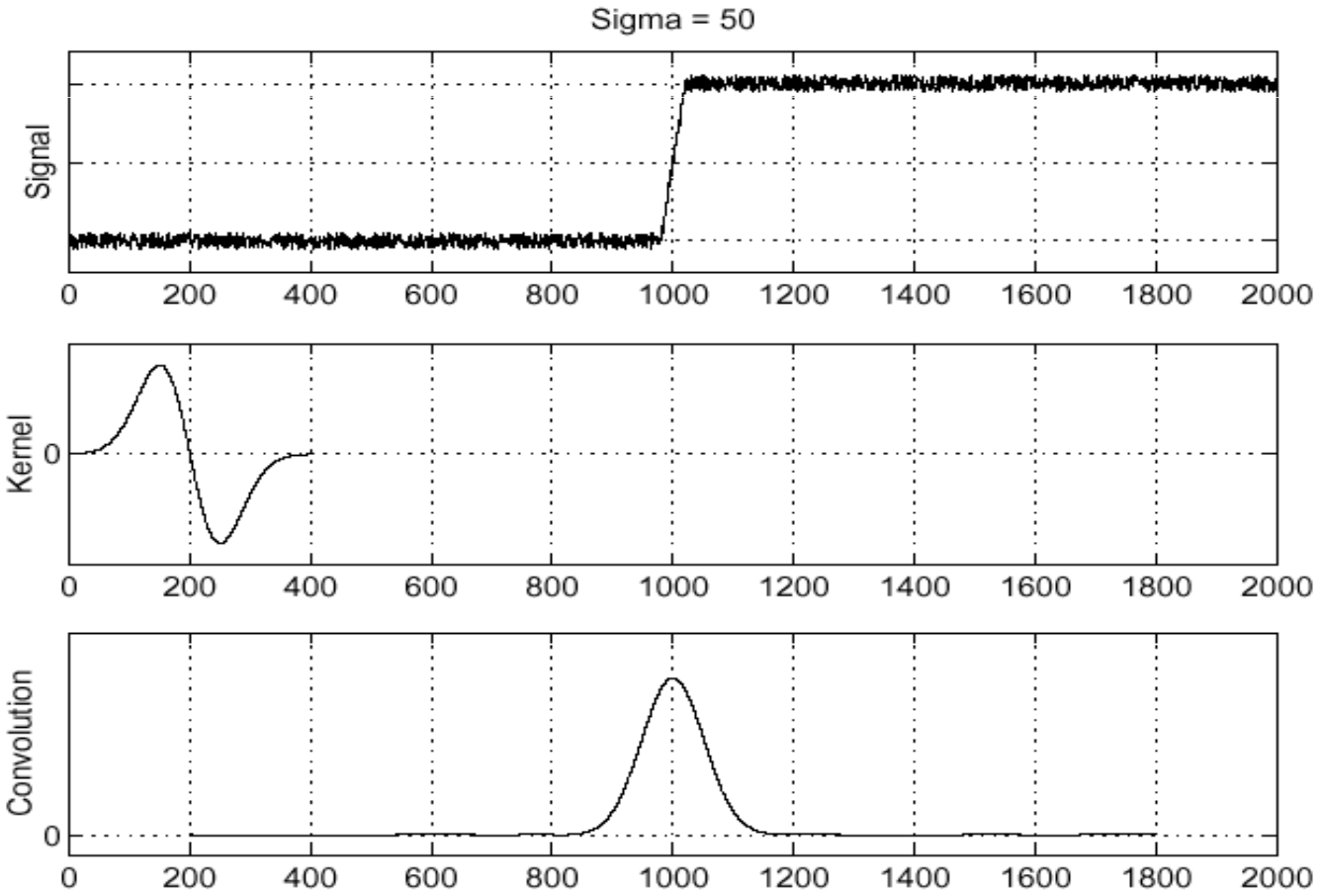
Difference operators under noise

Differentiation property of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

f

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

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



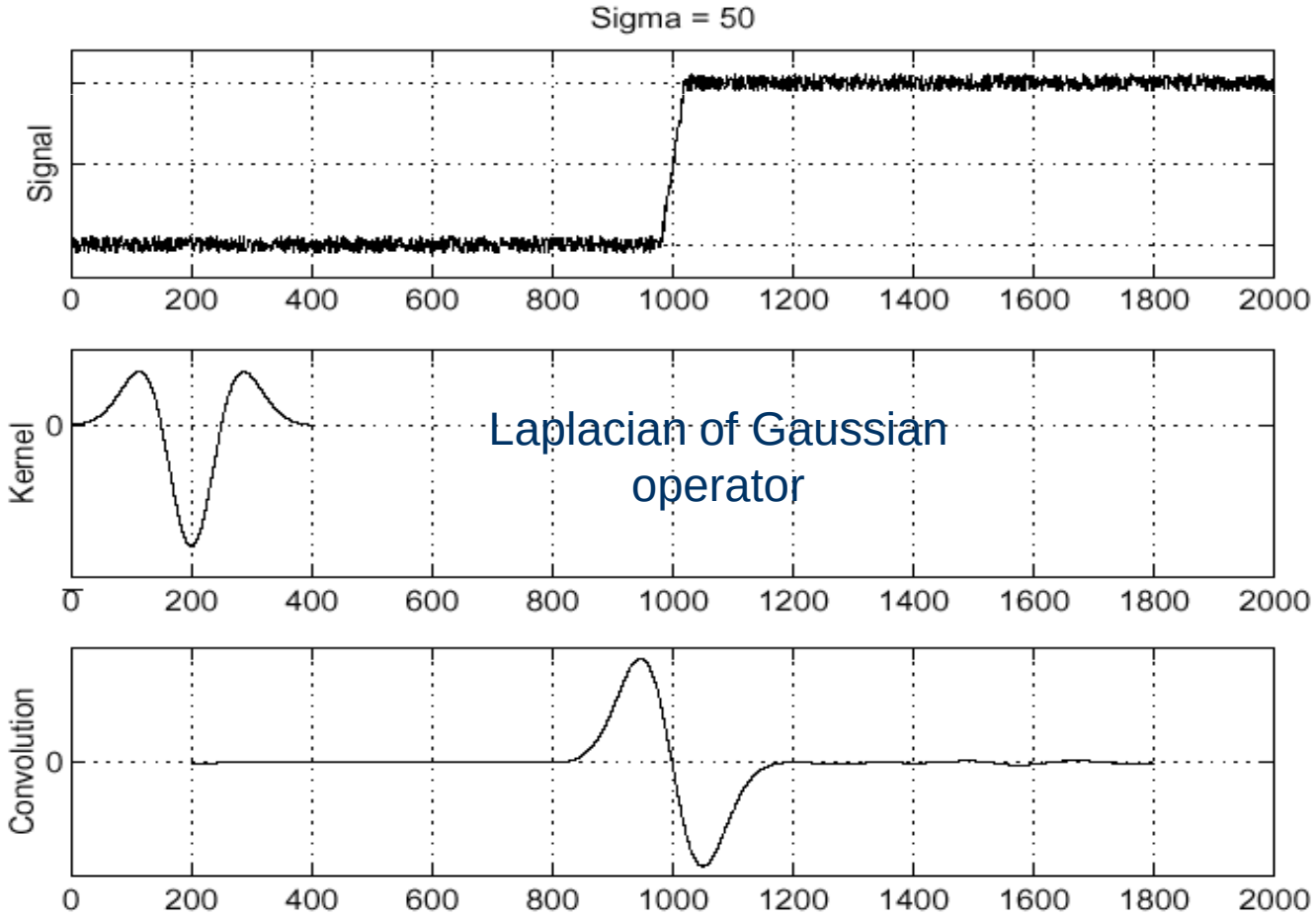
Difference operators under noise

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

f

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

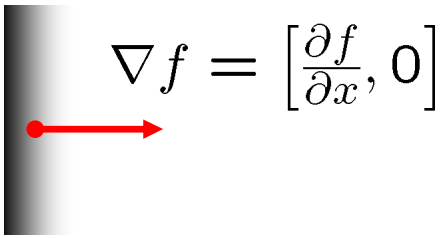
$(\frac{\partial^2}{\partial x^2} h) \star f$

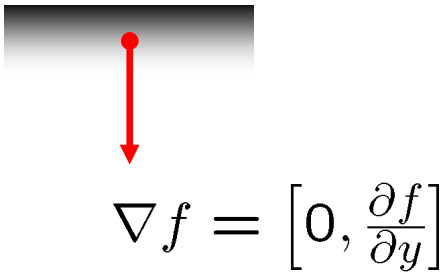


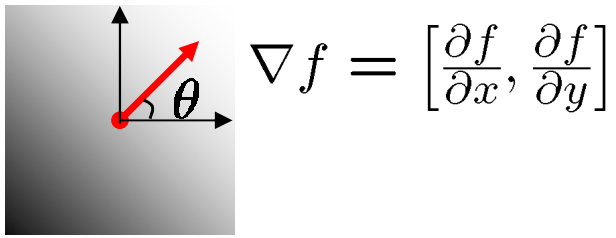
Difference operators for 2D

- 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

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







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*

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

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

Difference operators for 2D

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

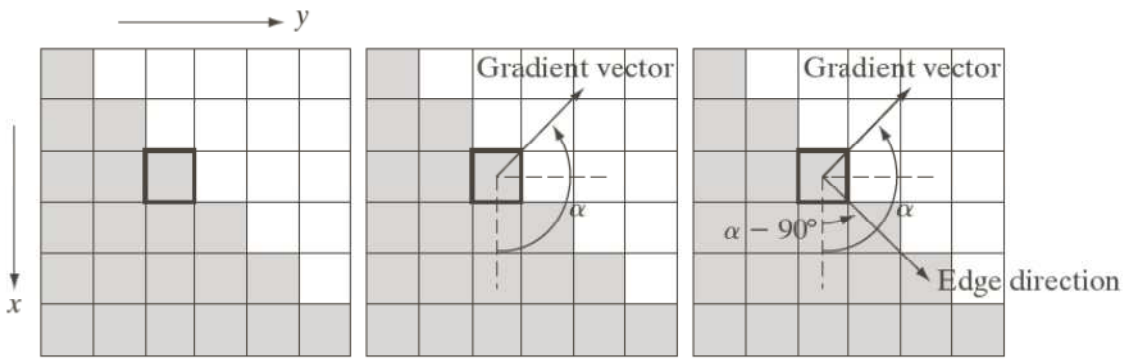
Prewitt

−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 × 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.



a b c

FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.

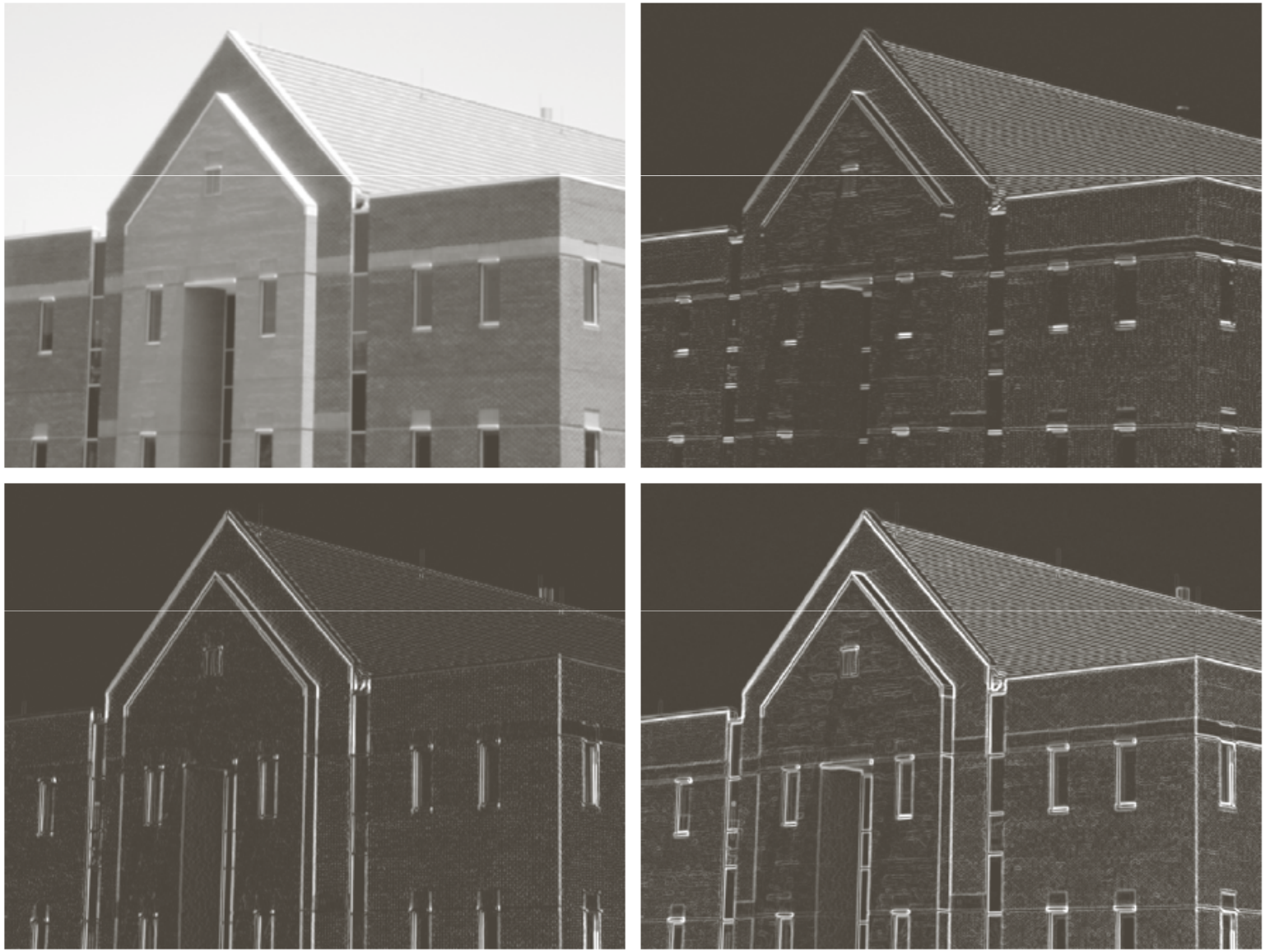
Adapted from Gonzales and Woods

Difference operators for 2D

a b
c d

FIGURE 10.16
(a) Original image of size 834×1114 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, $|g_x| + |g_y|$.

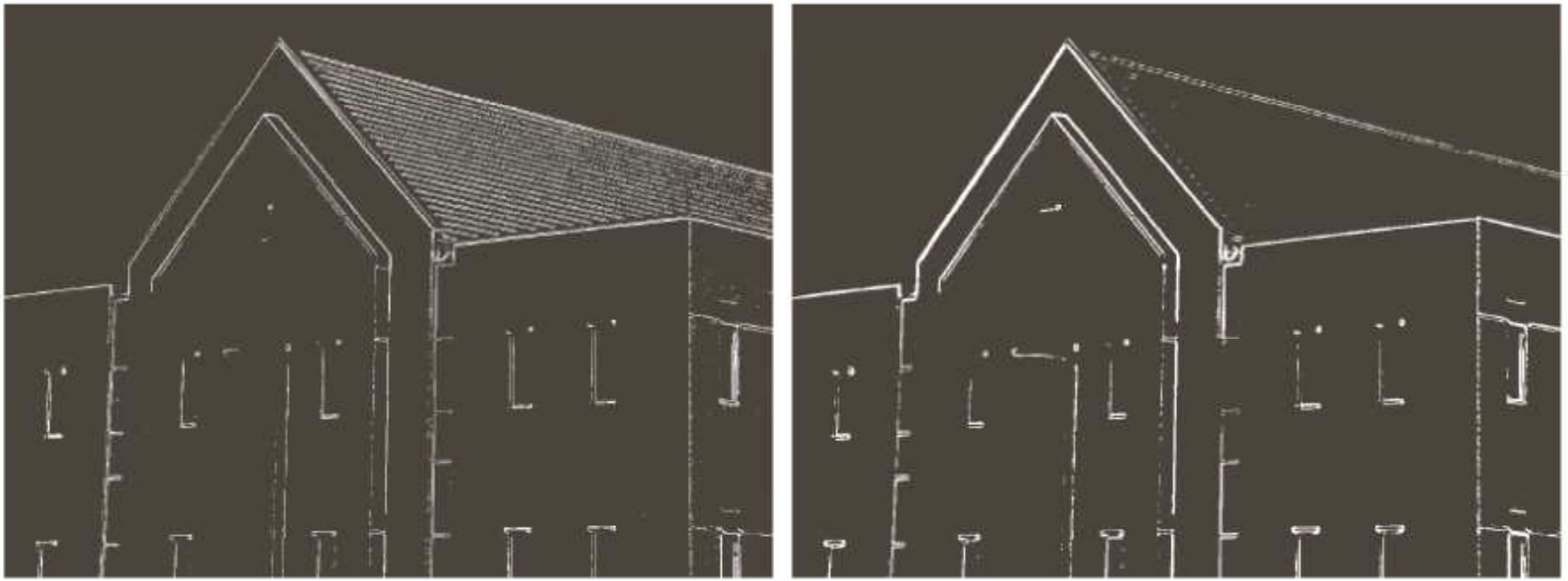
Difference operators for 2D



a b
c d

FIGURE 10.18
Same sequence as in Fig. 10.16, but with the original image smoothed using a 5×5 averaging filter prior to edge detection.

Difference operators for 2D



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.

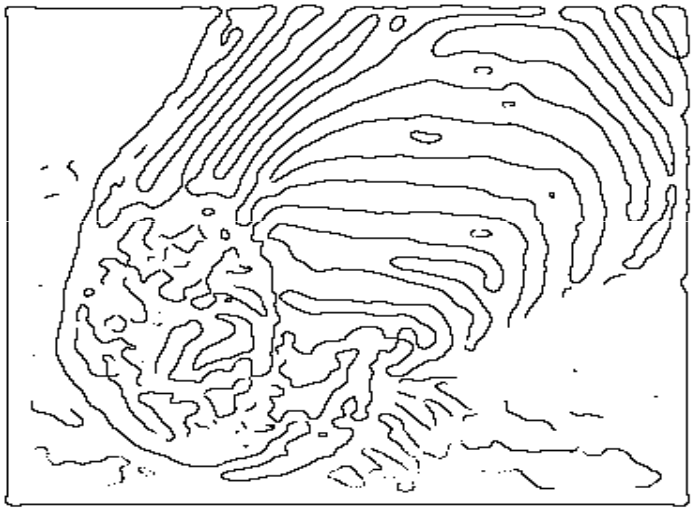
Difference operators for 2D

- 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.

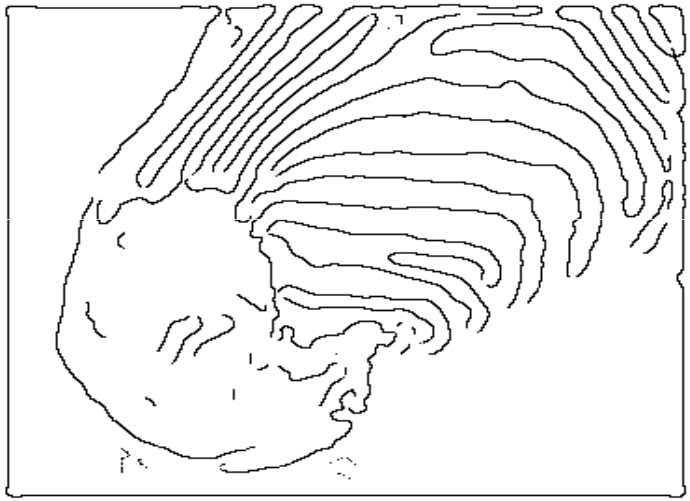
Difference operators for 2D



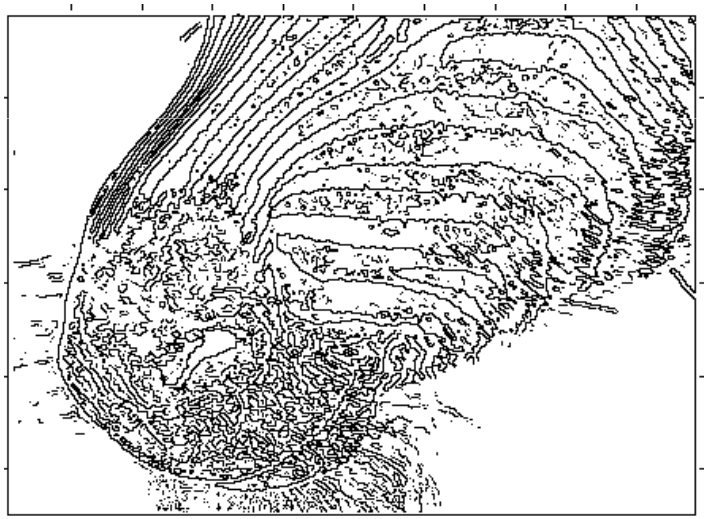
Gradient threshold=1

sigma=4

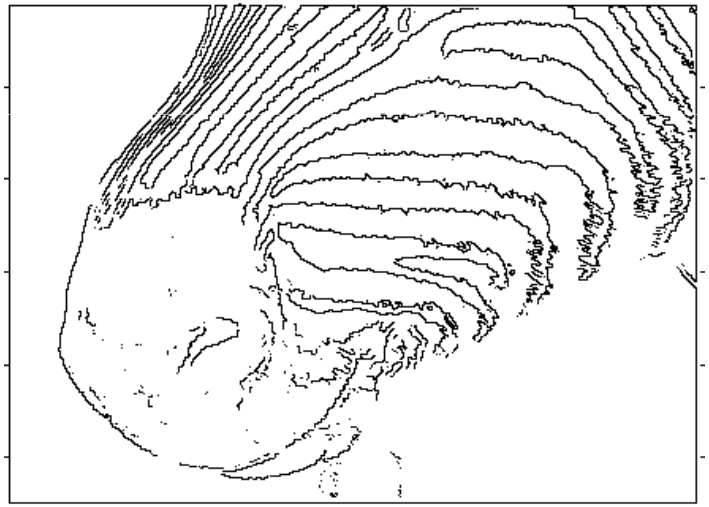
Laplacian of Gaussian zero crossings



Gradient threshold=4

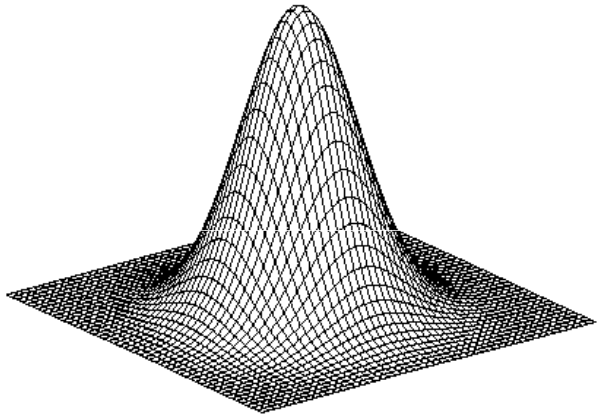


sigma=2



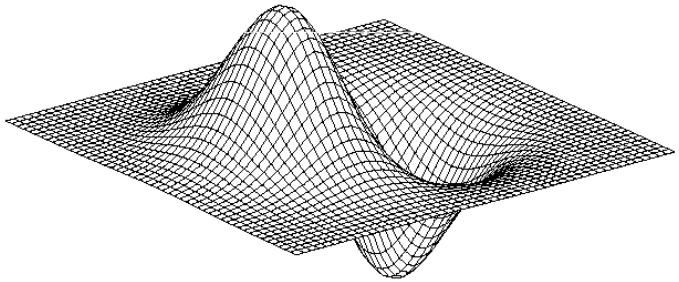
Adapted from David Forsyth, UC Berkeley

Edge detection filters for 2D



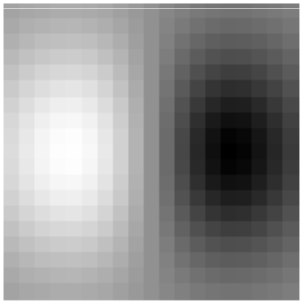
Gaussian

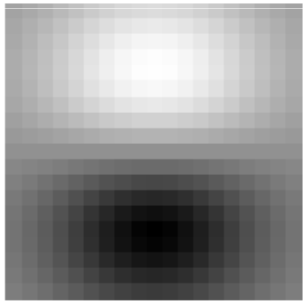
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

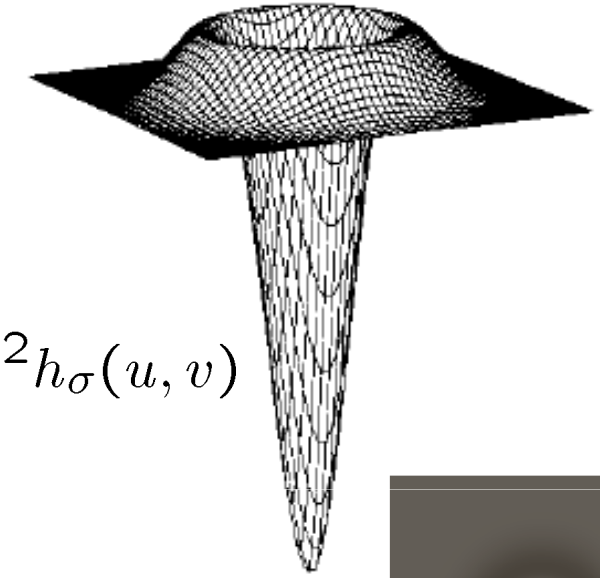
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

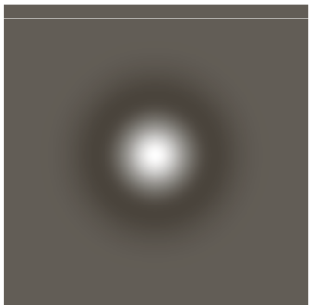




Laplacian of Gaussian

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





Adapted from Steve Seitz, U of Washington

Edge detection

- Three fundamental steps in edge detection:
  1. Image smoothing: to reduce the effects of noise.
  2. Detection of edge points: to find all image points that are potential candidates to become edge points.
  3. Edge localization: to select from the candidate edge points only the points that are true members of an edge.

Canny edge detector

- Canny defined three objectives for edge detection:
  1. Low error rate: All edges should be found and there should be no spurious responses.
  2. Edge points should be well localized: The edges located must be as close as possible to the true edges.
  3. Single edge point response: The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum.

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

Original image (Lena)

Adapted from Steve Seitz, U of Washington

Canny edge detector

Magnitude of the gradient

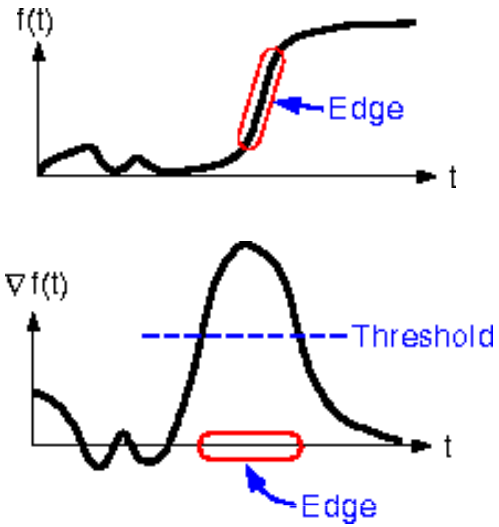
Adapted from Steve Seitz, U of Washington

Canny edge detector

Thresholding

Adapted from Steve Seitz, U of Washington

Canny edge detector

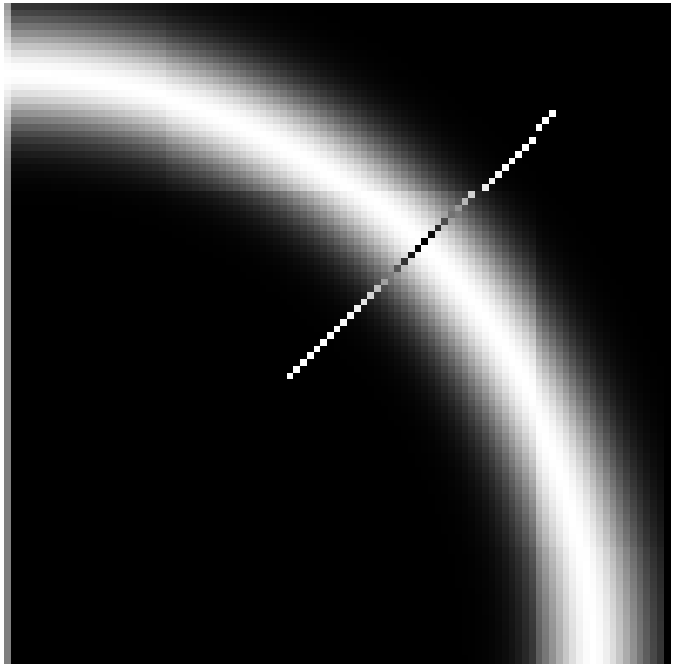


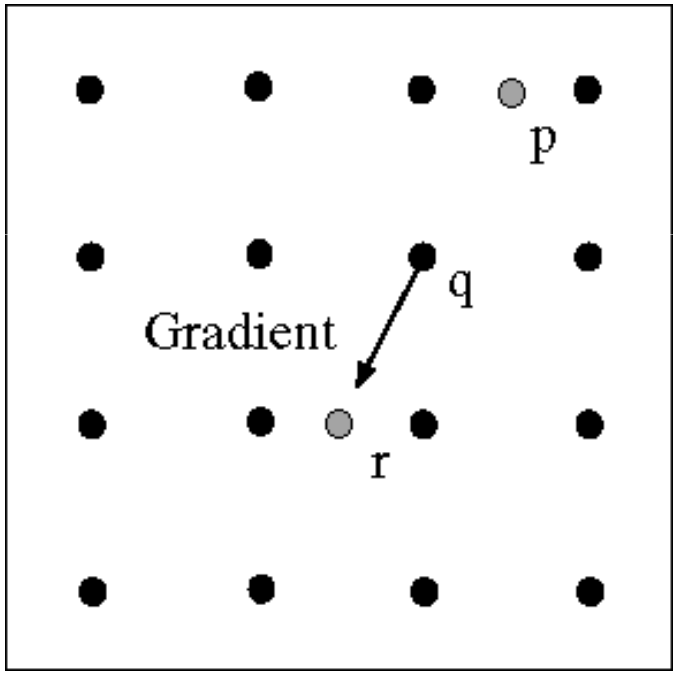
How to turn these thick regions of the gradient into curves?

Adapted from Steve Seitz, U of Washington

Canny edge detector

- Non-maxima suppression:
  - Check if pixel is local maximum along gradient direction.
  - Select single max across width of the edge.
  - Requires checking interpolated pixels p and r.
  - This operation can be used with any edge operator when thin boundaries are wanted.





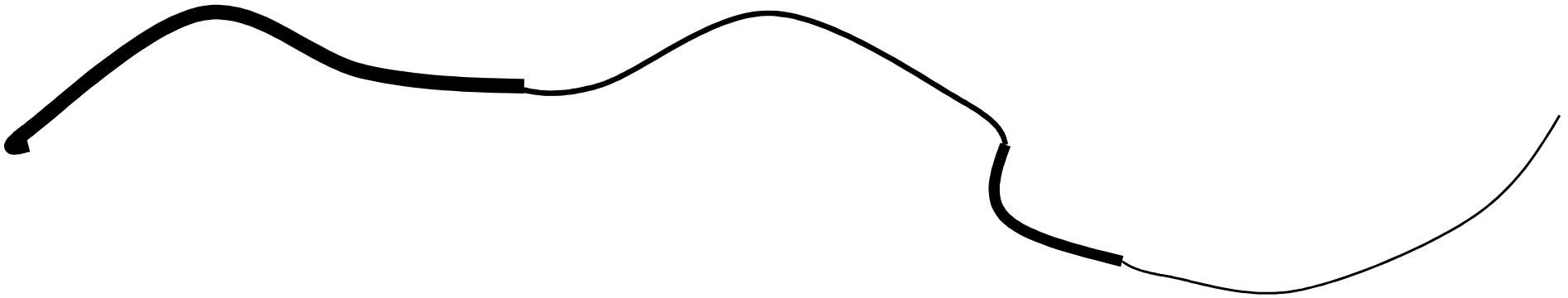
Canny edge detector

Problem: pixels along this edge did not survive the thresholding

Adapted from Steve Seitz, U of Washington

Canny edge detector

- Hysteresis thresholding:
  - Use a high threshold to start edge curves, and a low threshold to continue them.



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

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

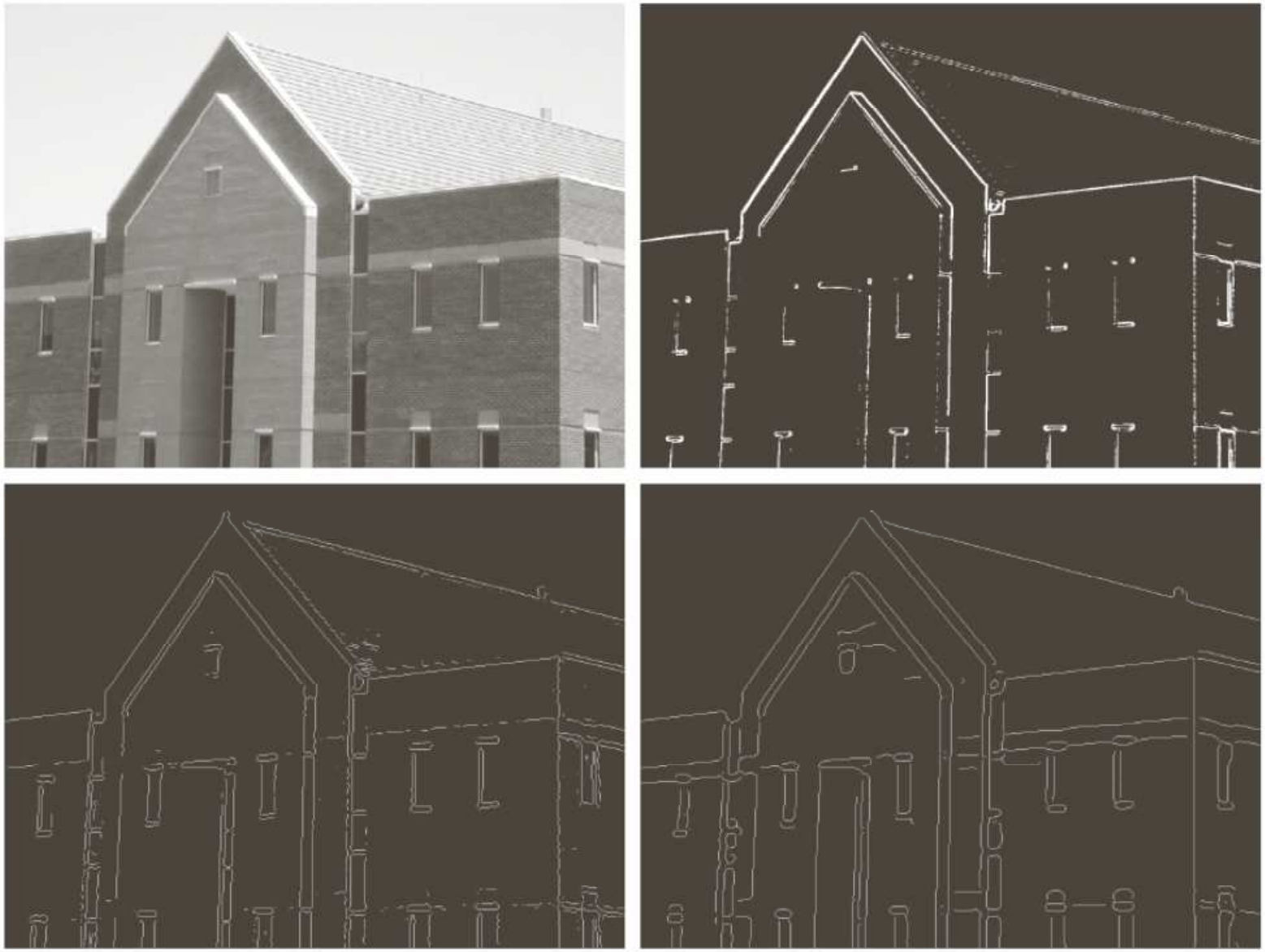
3. Eliminate those pixels that are not local maxima of the magnitude in the direction of the gradient

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

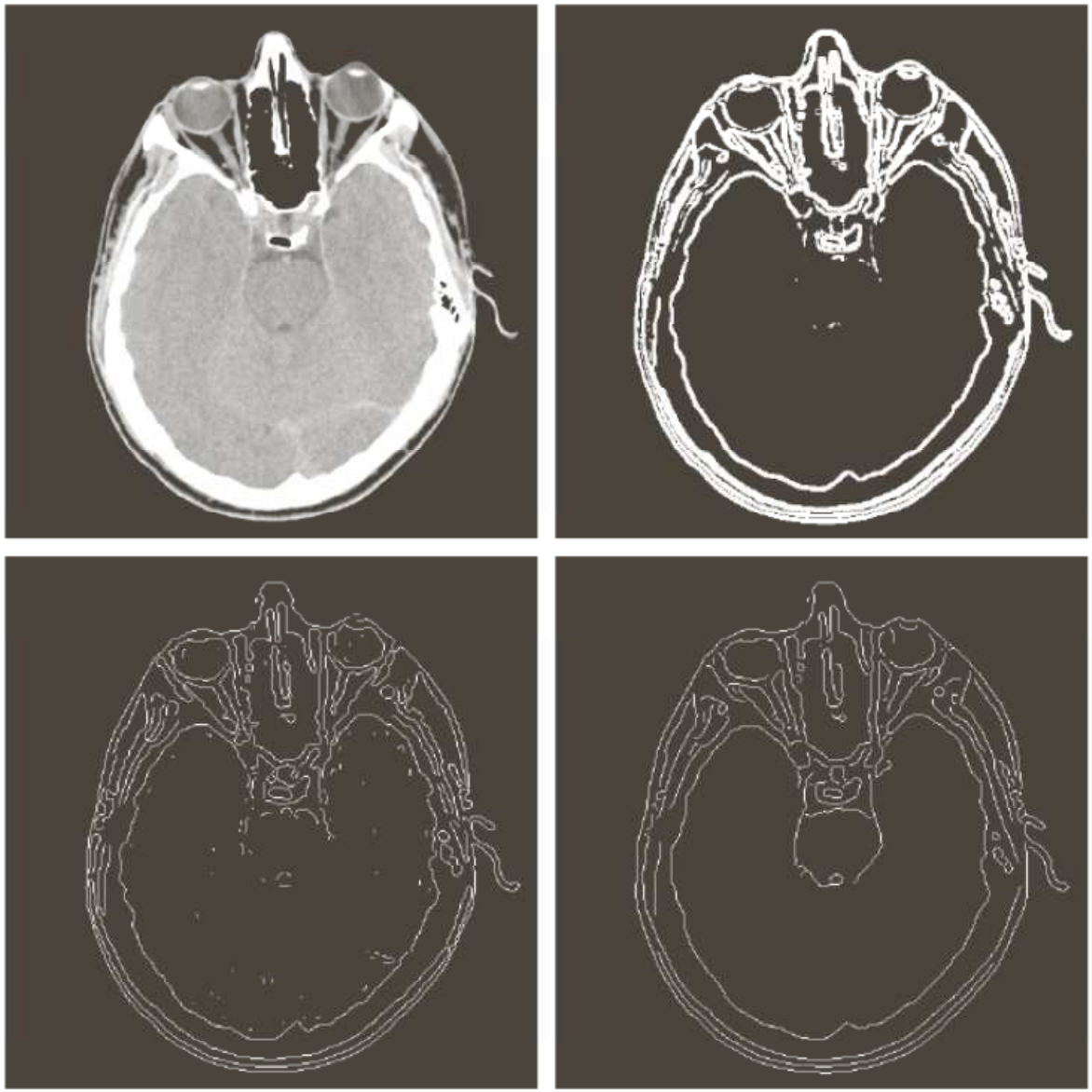
Canny edge detector



a b
c d

FIGURE 10.25
(a) Original image of size 834 × 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.

Canny edge detector



a b
c d

FIGURE 10.26
(a) Original head CT image of size 512 × 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

- 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

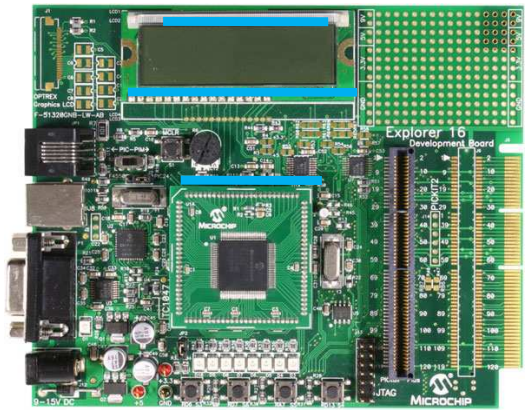
Fitting: main idea

- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - Cannot tell whether a point belongs to a given model just by looking at that point
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets which feature?
  - How many model instances are there?
- Computational complexity is important
  - It is infeasible to examine every possible set of parameters and every possible combination of features

Adapted from Kristen Grauman

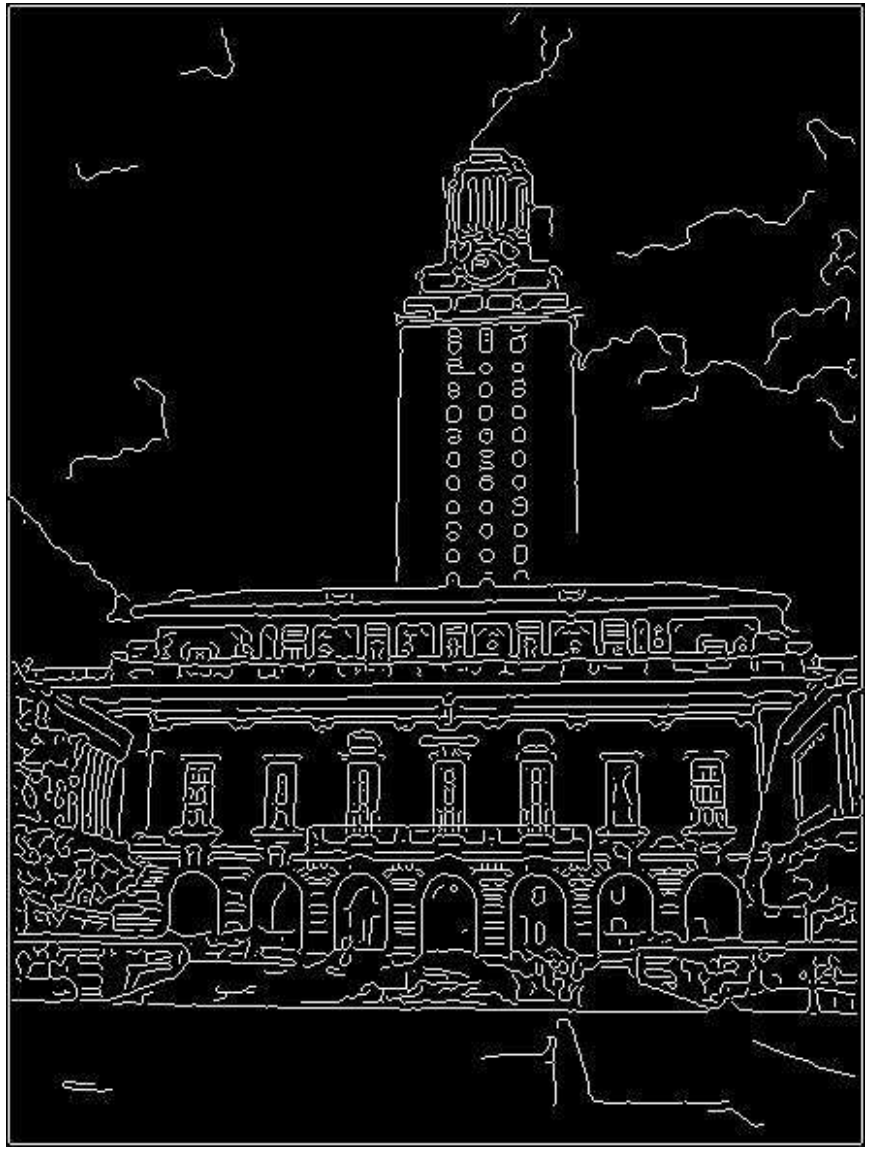
Example: line fitting

- Why fit lines?
  - Many objects characterized by presence of straight lines



Adapted from Kristen Grauman

Difficulty of line fitting



- **Extra** edge points (clutter), multiple models:
  - which points go with which line, if any?
- Only some parts of each line detected, and some parts are **missing**:
  - how to find a line that bridges missing evidence?
- **Noise** in measured edge points, orientations:
  - how to detect true underlying parameters?

Adapted from Kristen Grauman

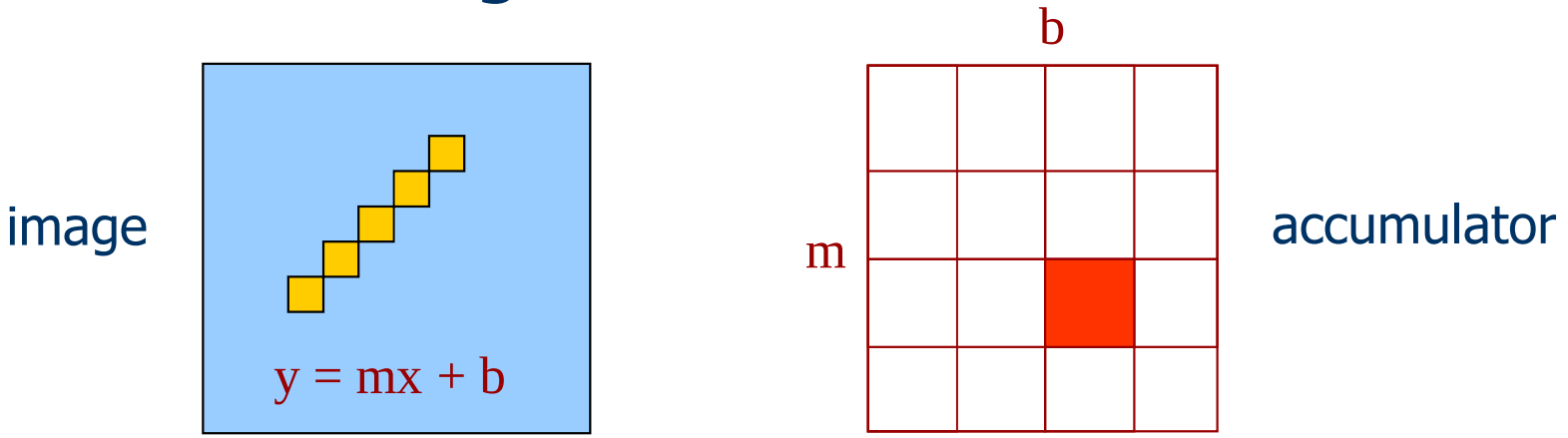
Voting

- It is not feasible to check all combinations of features by fitting a model to each possible subset.
- Voting is a general technique where we let each feature vote for all models that are compatible with it.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Noise and clutter features will cast votes too, but typically their votes should be inconsistent with the majority of "good" features.

Adapted from Kristen Grauman

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

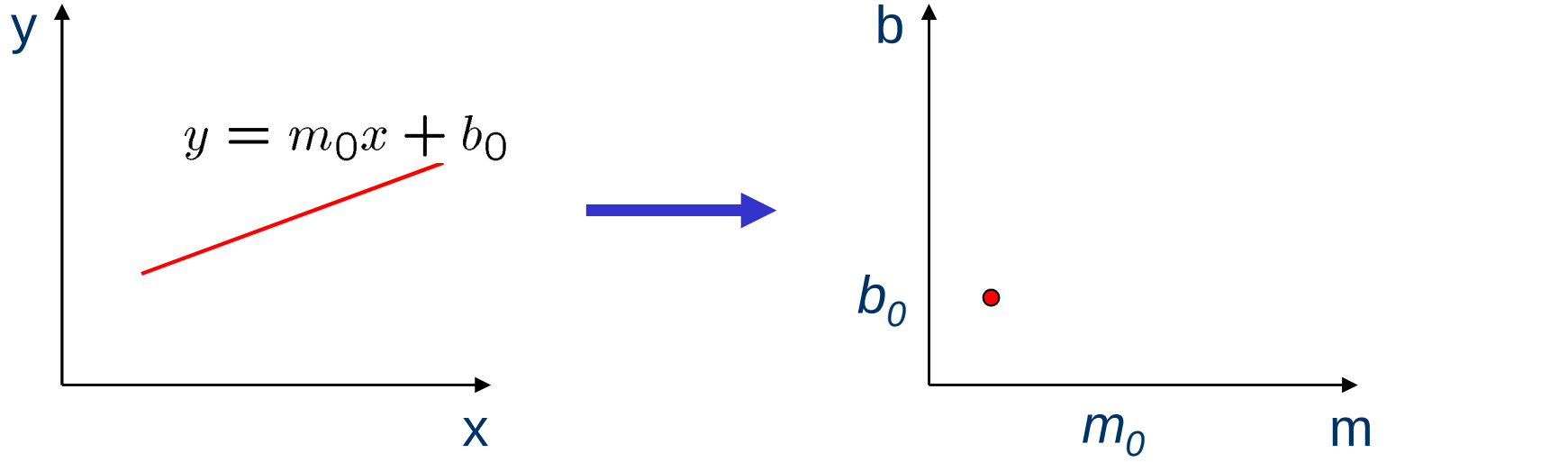


Image space

Hough (parameter) space

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

Hough transform: line segments

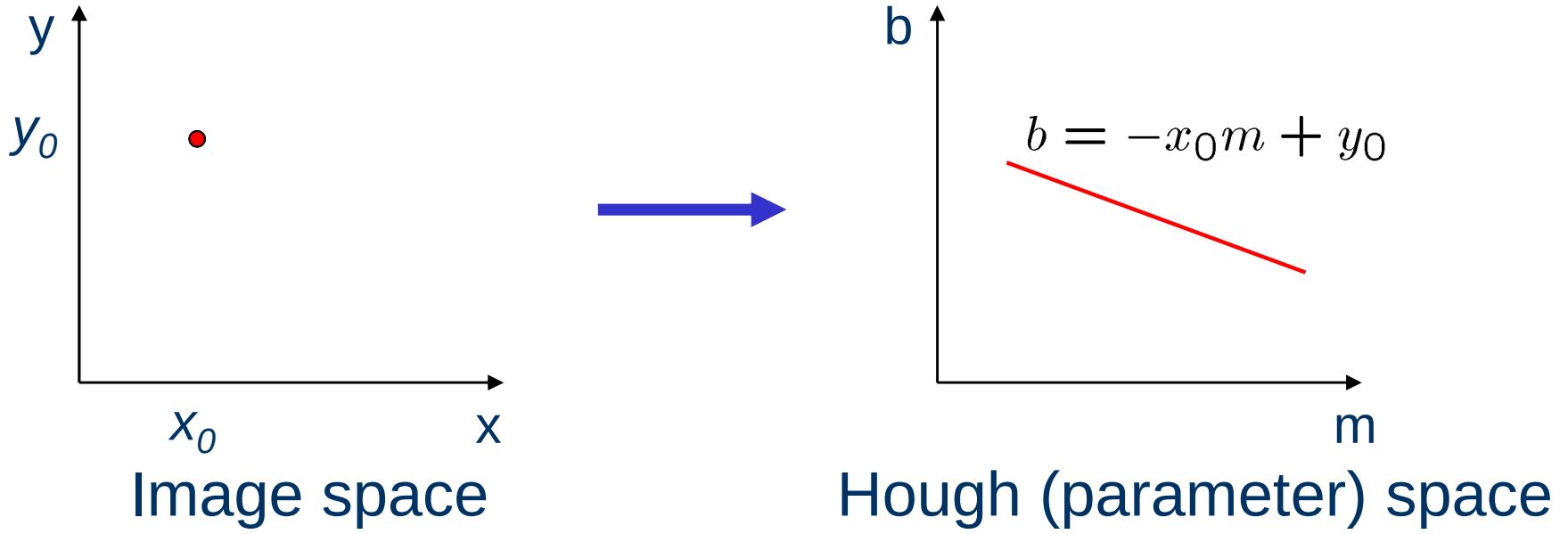


Image space

Hough (parameter) space

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?
  - Answer: the solutions of $b = -x_0 m + y_0$
  - This is a line in Hough space

Adapted from Steve Seitz, U of Washington

Hough transform: line segments

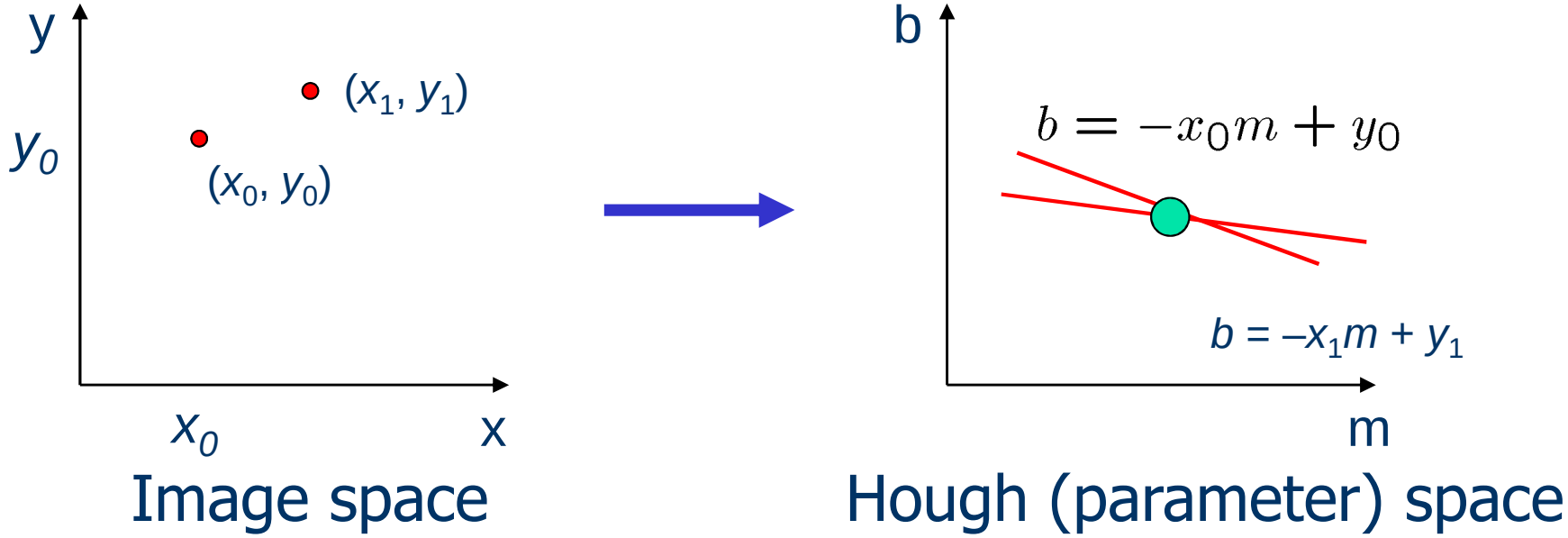


Image space

Hough (parameter) space

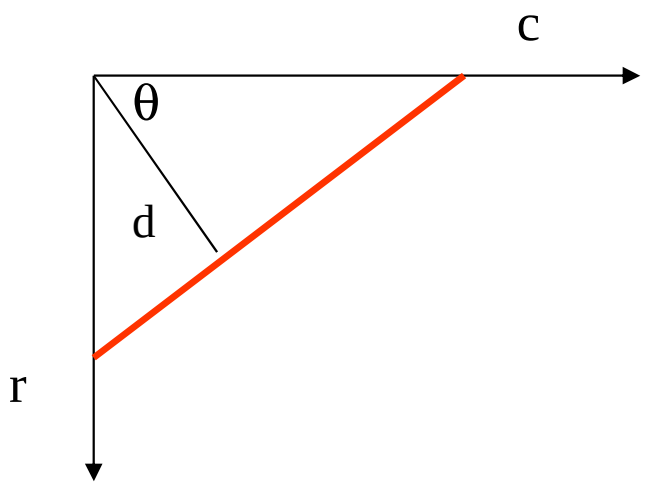
What are the line parameters for the line that contains both (x_0, y_0) and (x_1, y_1)?

- It is the intersection of the lines $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$

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

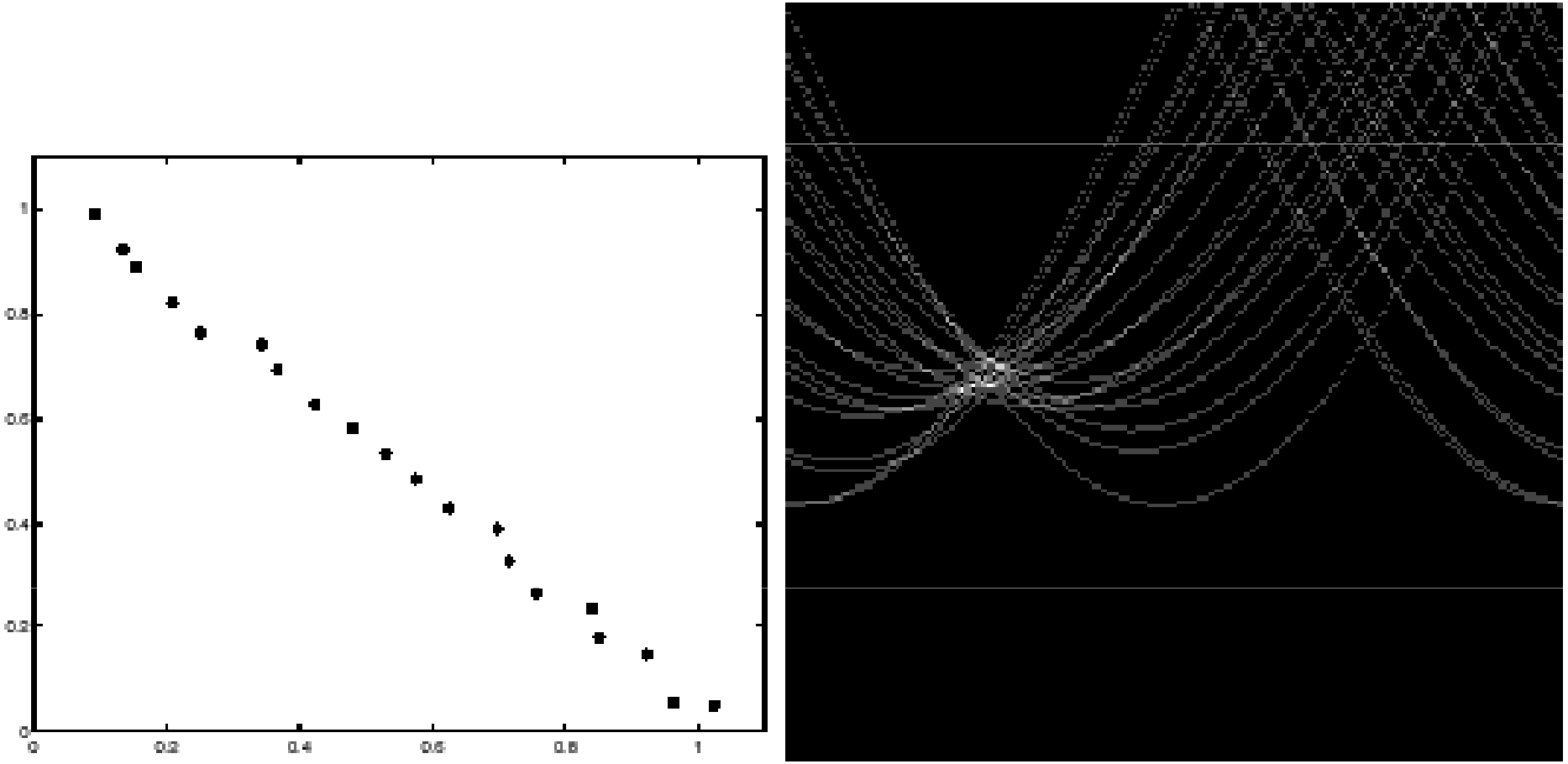
Accumulate the straight line segments in gray-tone image S to accumulator A.
S[**R**, **C**] is the input gray-tone image.
NLINES is the number of rows in the image.
NPIXELS is the number of pixels per row.
A[**DQ**, **THETAQ**] is the accumulator array.
DQ is the quantized distance from a line to the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

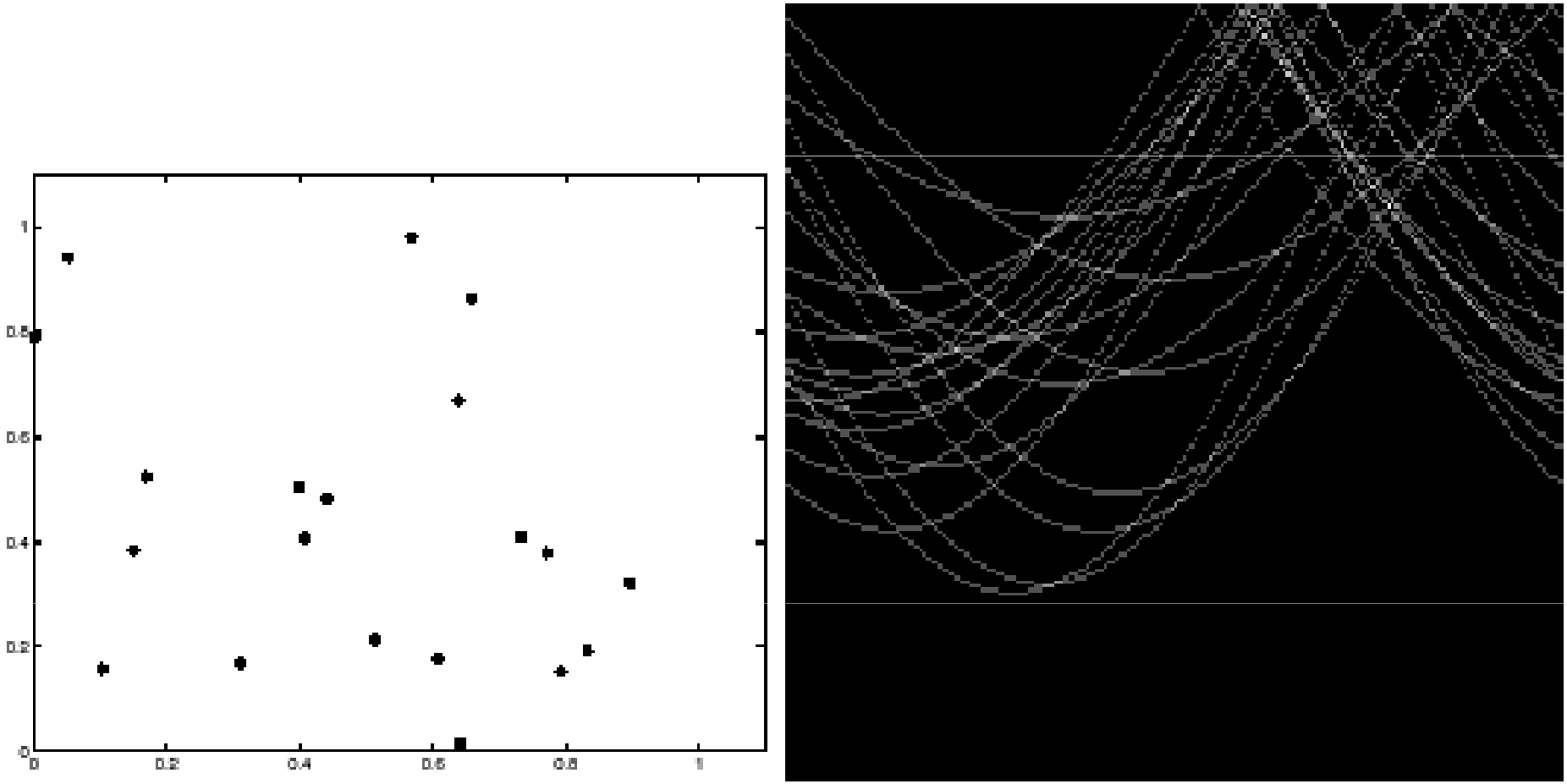
Algorithm 7: Hough Transform for Finding Straight Lines

Adapted from Shapiro and Stockman

Hough transform: line segments



Hough transform: line segments



Hough transform: line segments

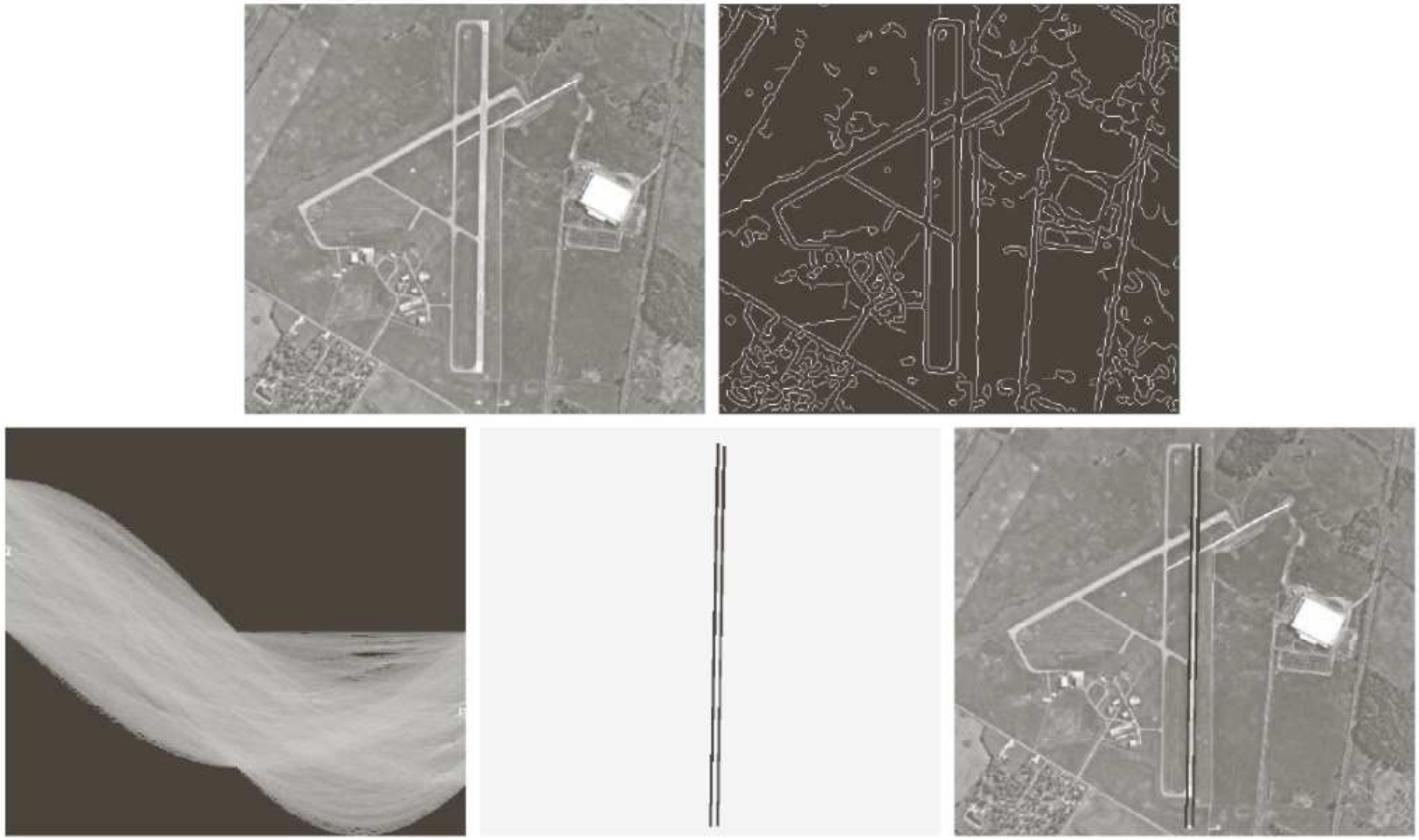
- Extracting the line segments from the accumulators:

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

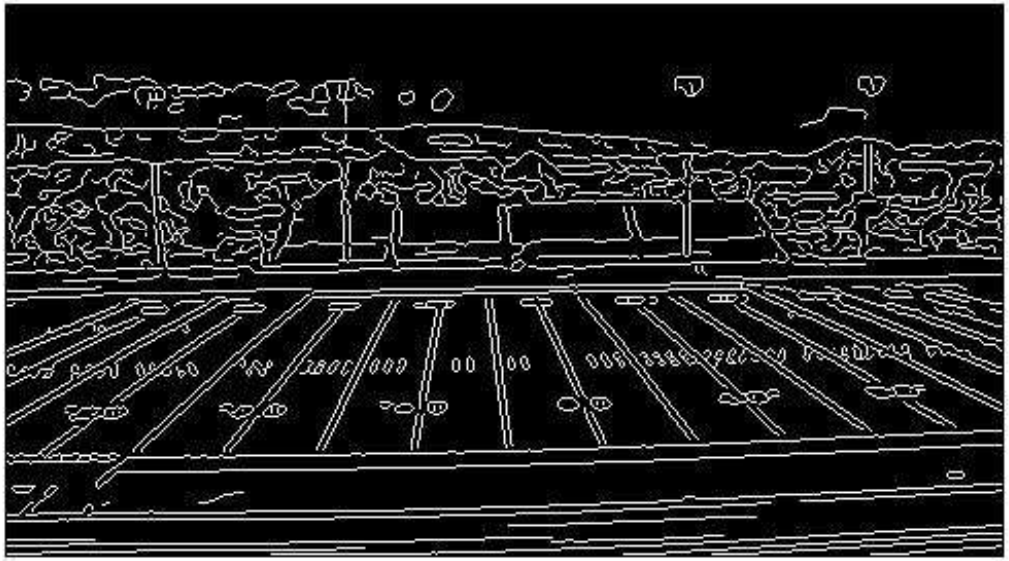
Hough transform: line segments

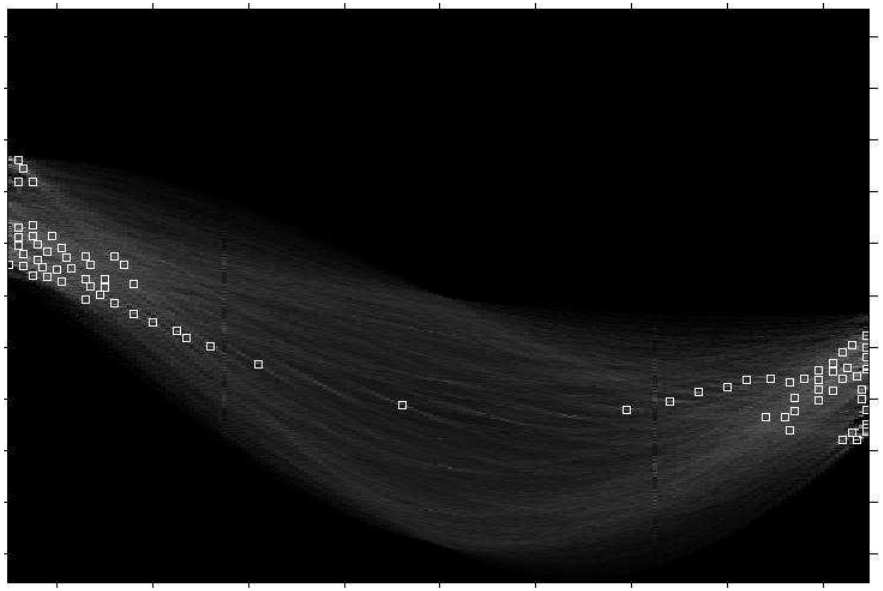


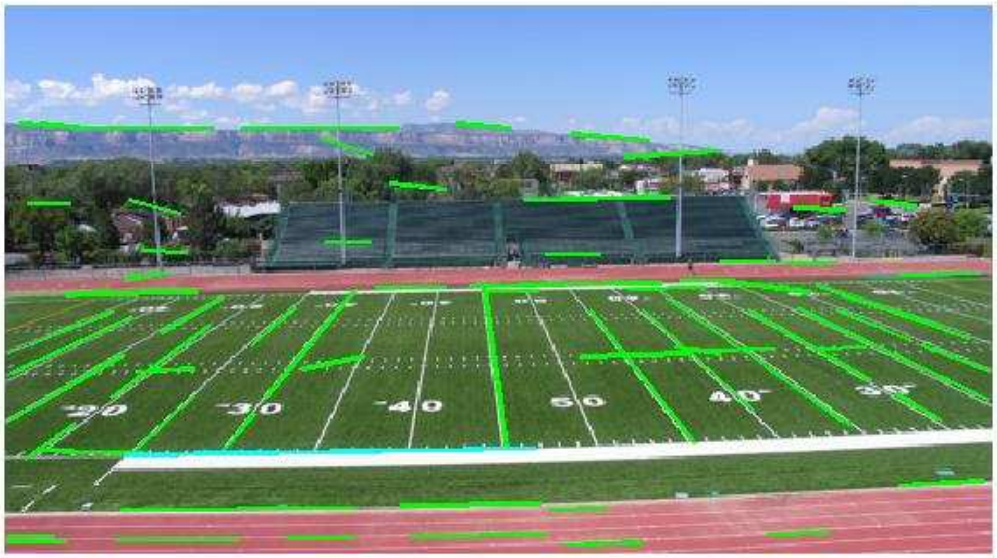
a b
c d e

FIGURE 10.34 (a) A 502 × 564 aerial image of an airport. (b) Edge image obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes). (e) Lines superimposed on the original image.

Hough transform: line segments





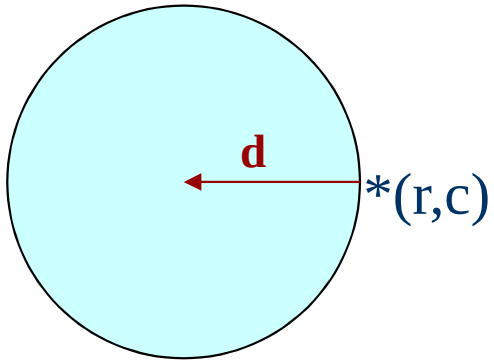


Adapted from Kristen Grauman

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

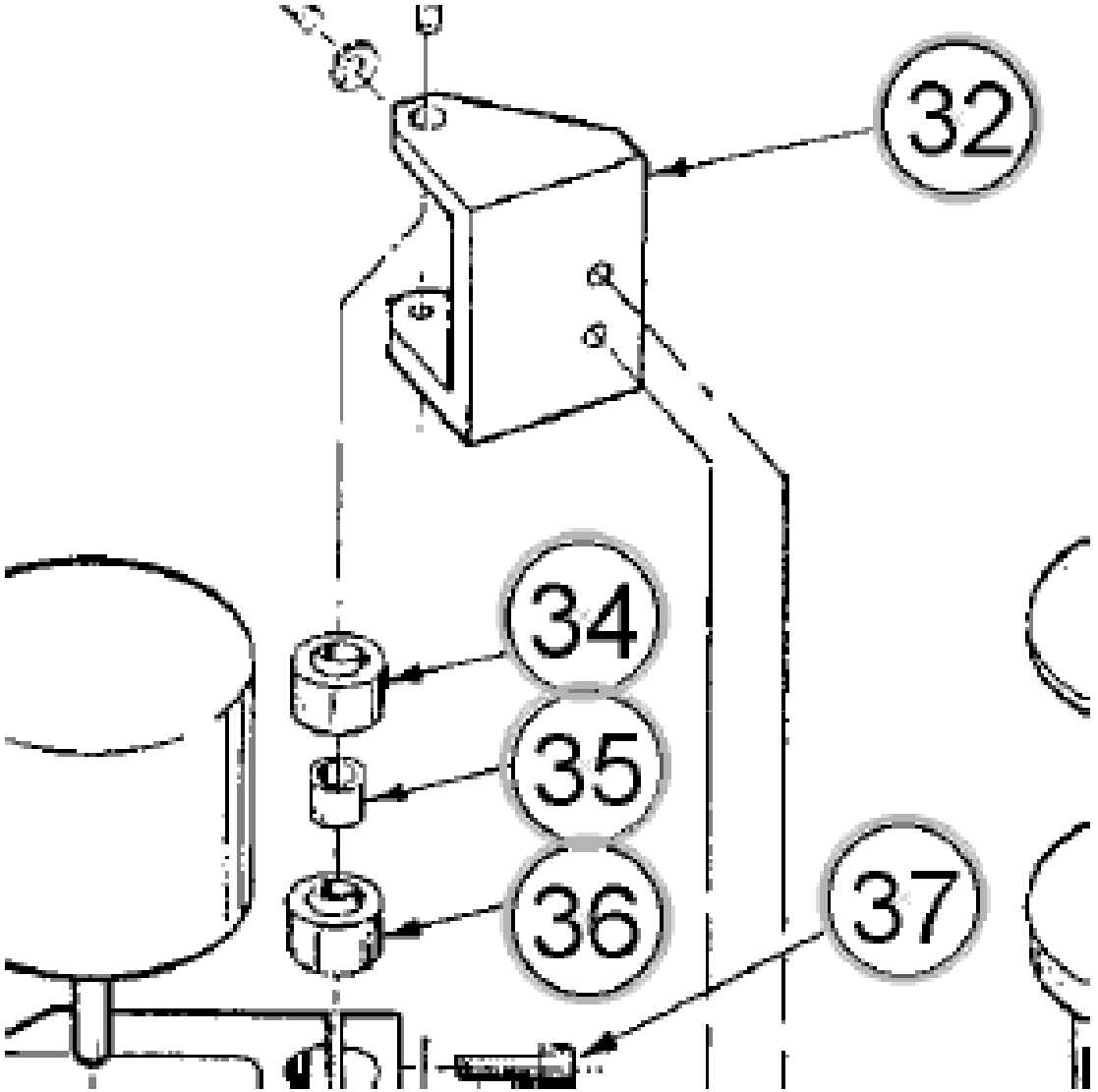
Accumulate the circles in gray-tone image S to accumulator A.
S[R, C] is the input gray-tone image.
NLINES is the number of rows in the image.
NPIXELS is the number of pixels per row.
A[R, C, RAD] is the accumulator array.
R is the row index of the circle center.
C is the column index of the circle center.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

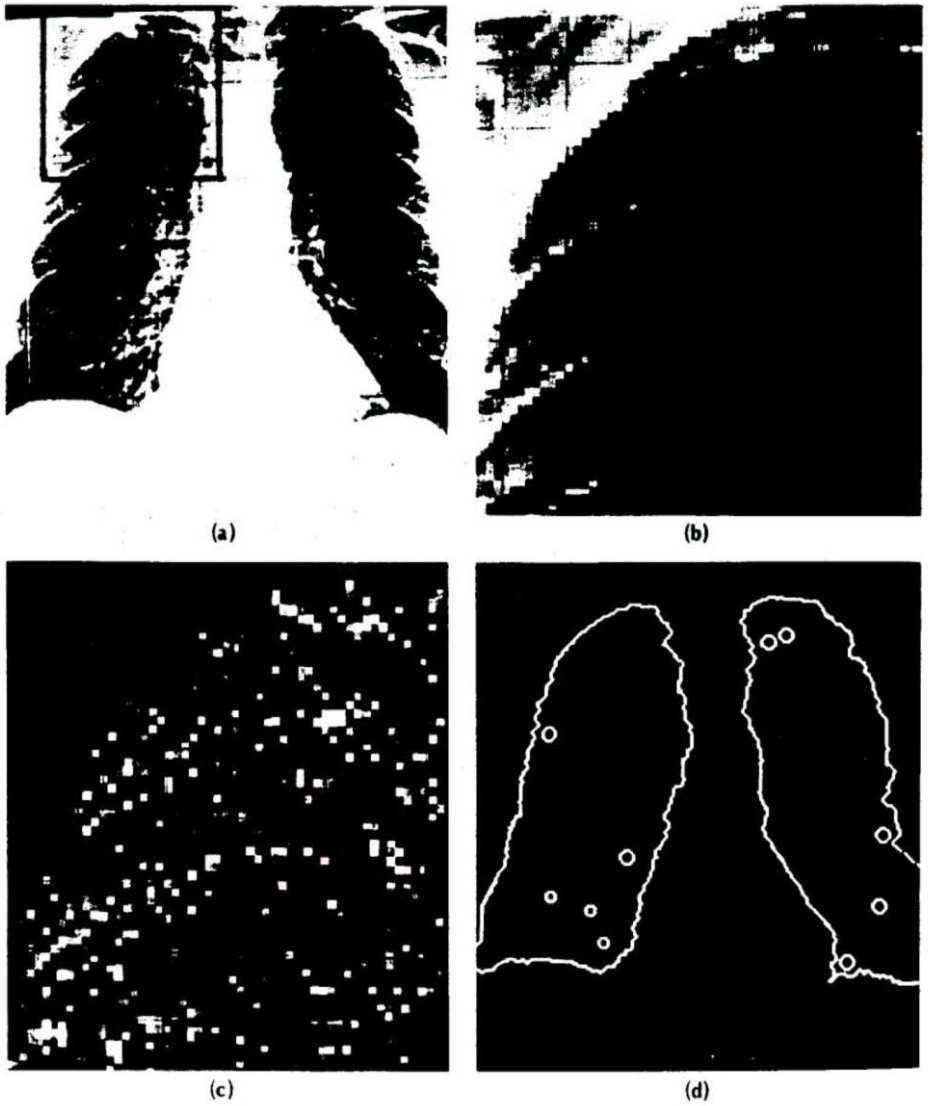
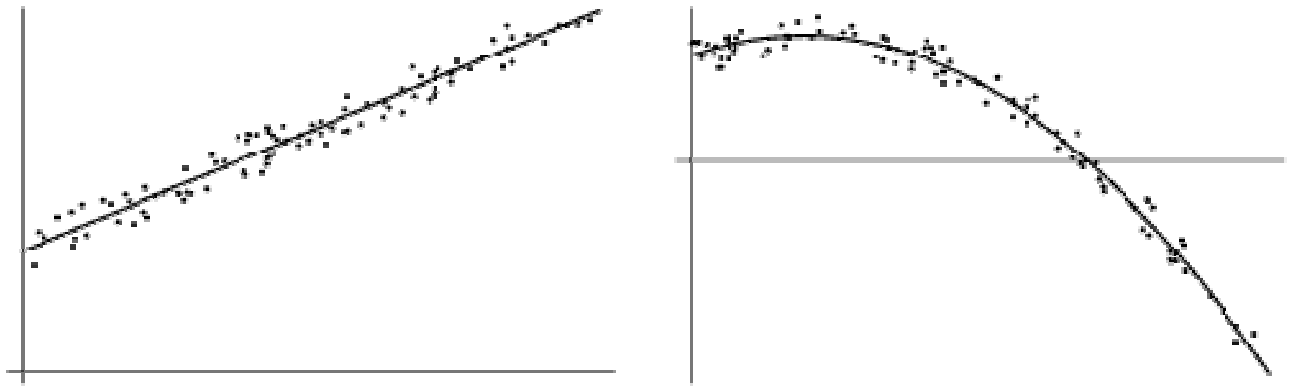


Fig. 4.7 Using the Hough technique for circular shapes. (a) Radiograph. (b) Window. (c) Accumulator array for $r = 3$. (d) Results of maxima detection.

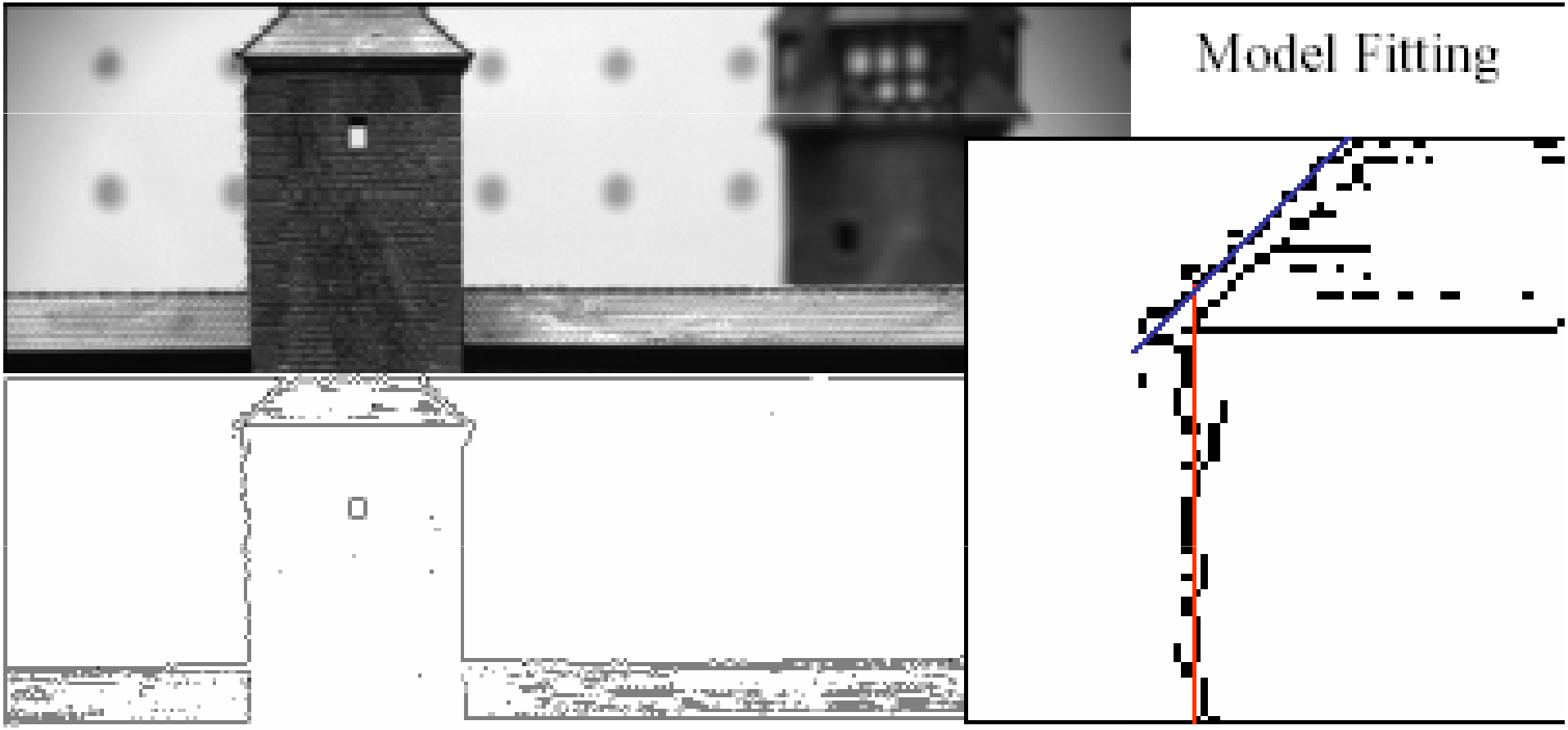
Adapted from Shapiro and Stockman

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 the 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*:

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

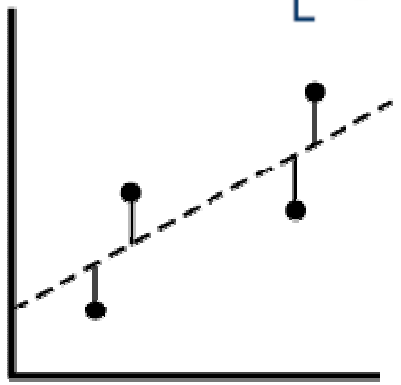
where $ax_i + b - y_i$ is the algebraic distance.

- 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:

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

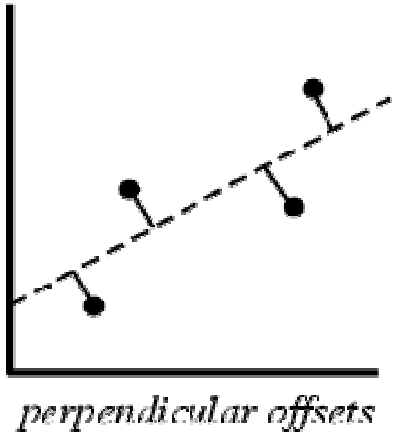


Model fitting: line segments

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



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.

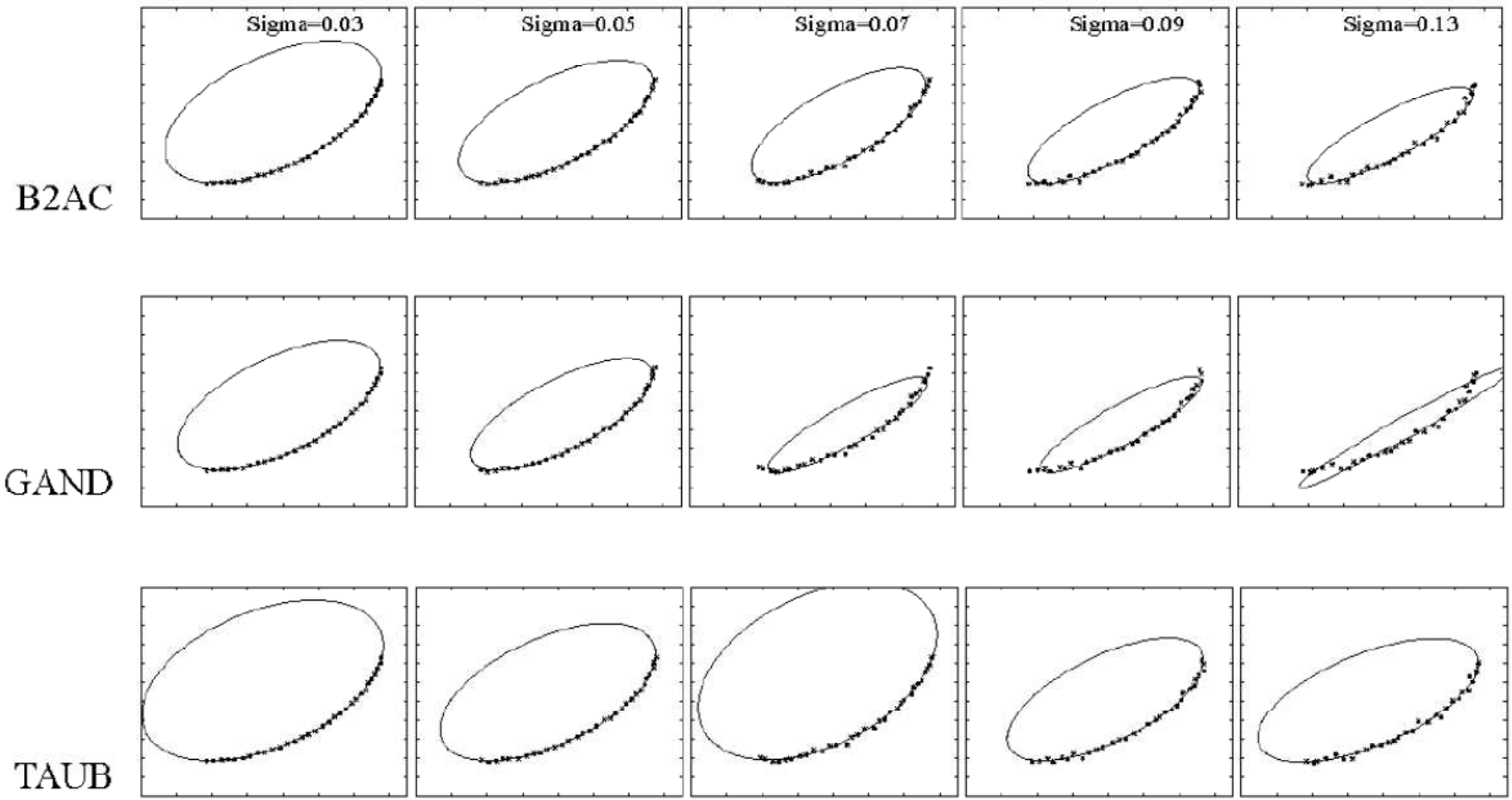
Model fitting: ellipses

```
% 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

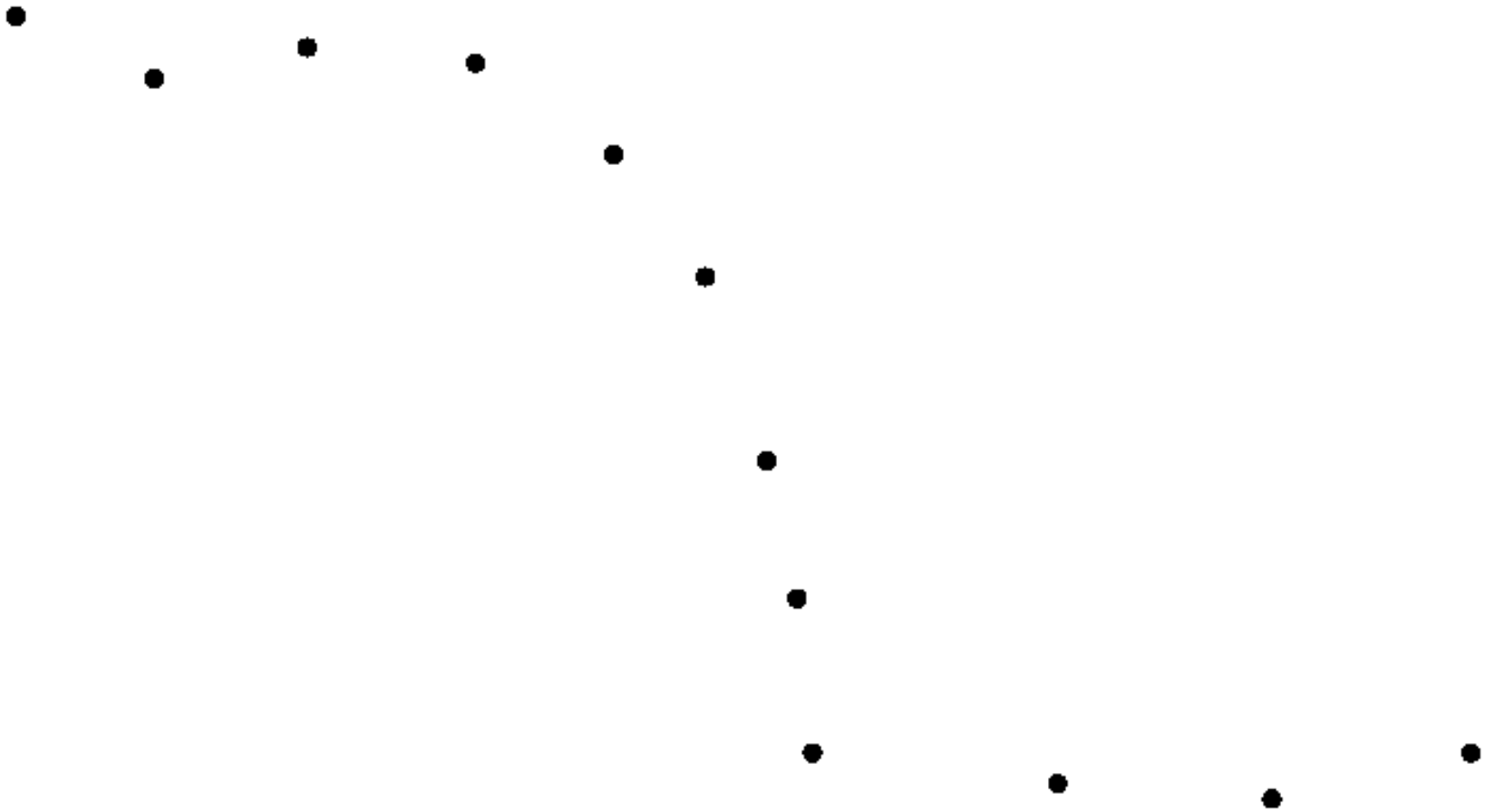
Model fitting: incremental line fitting

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
```

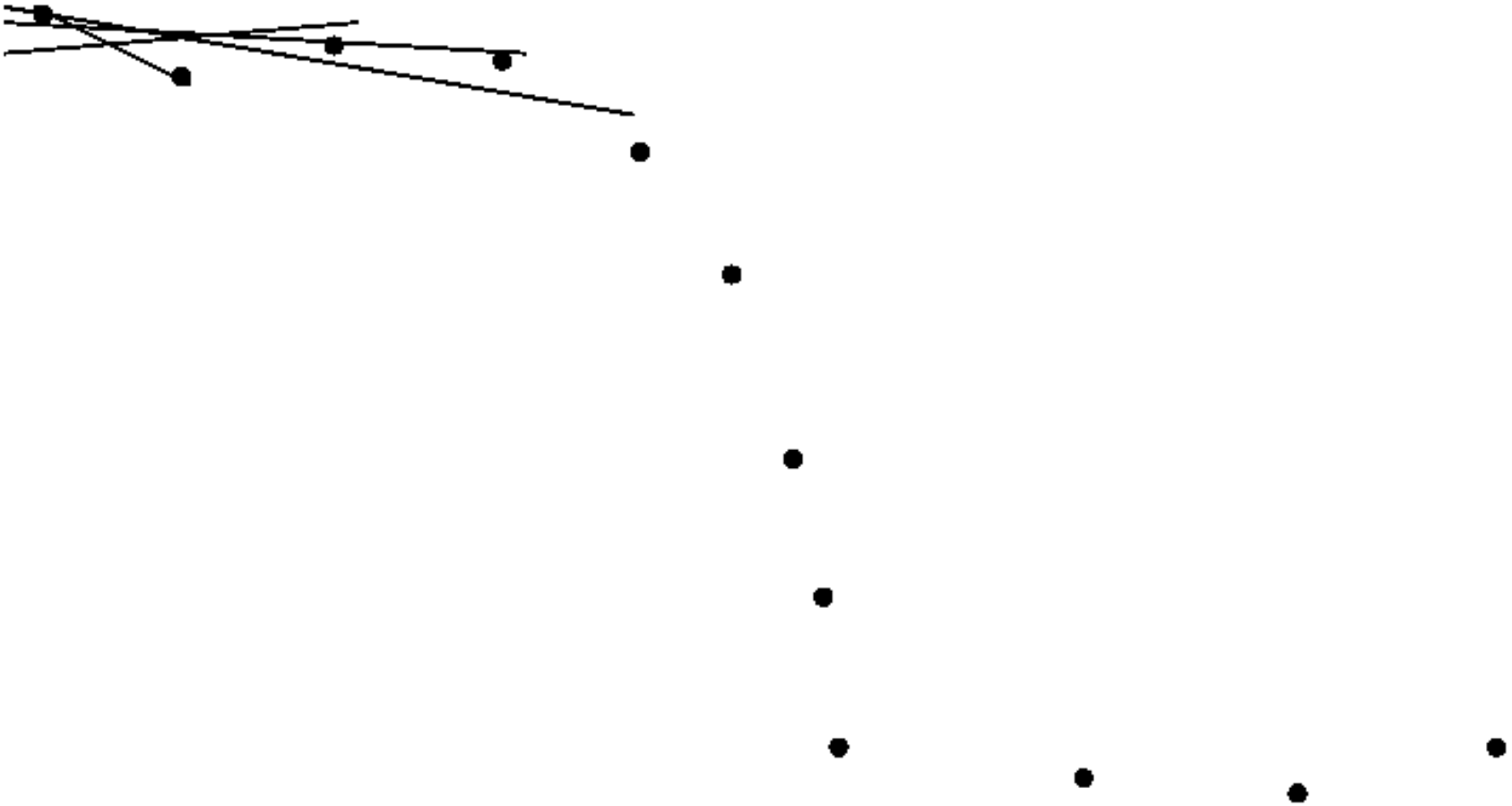
Adapted from David Forsyth, UC Berkeley

Model fitting: incremental line fitting



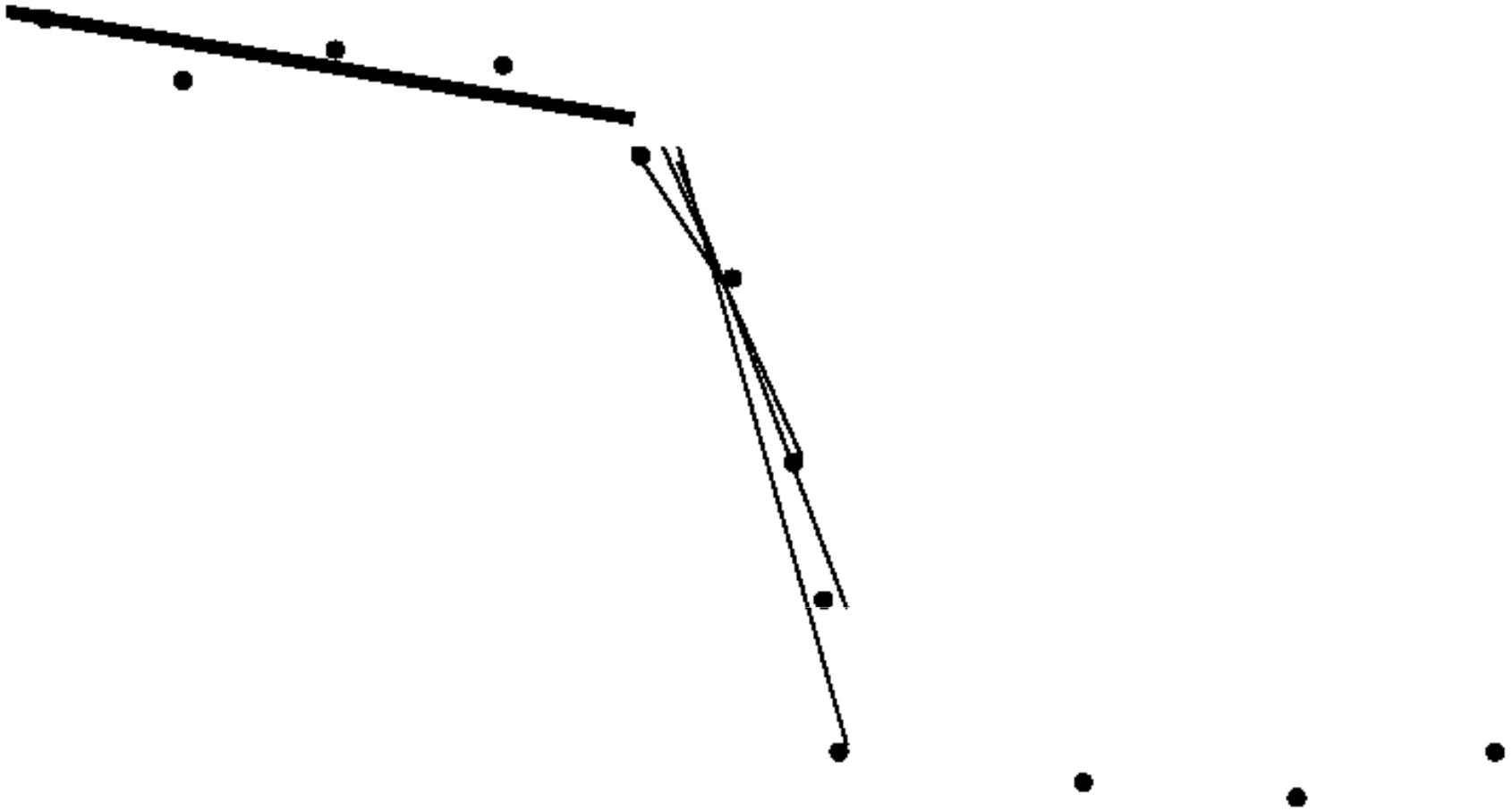
Adapted from Trevor Darrell, MIT

Model fitting: incremental line fitting



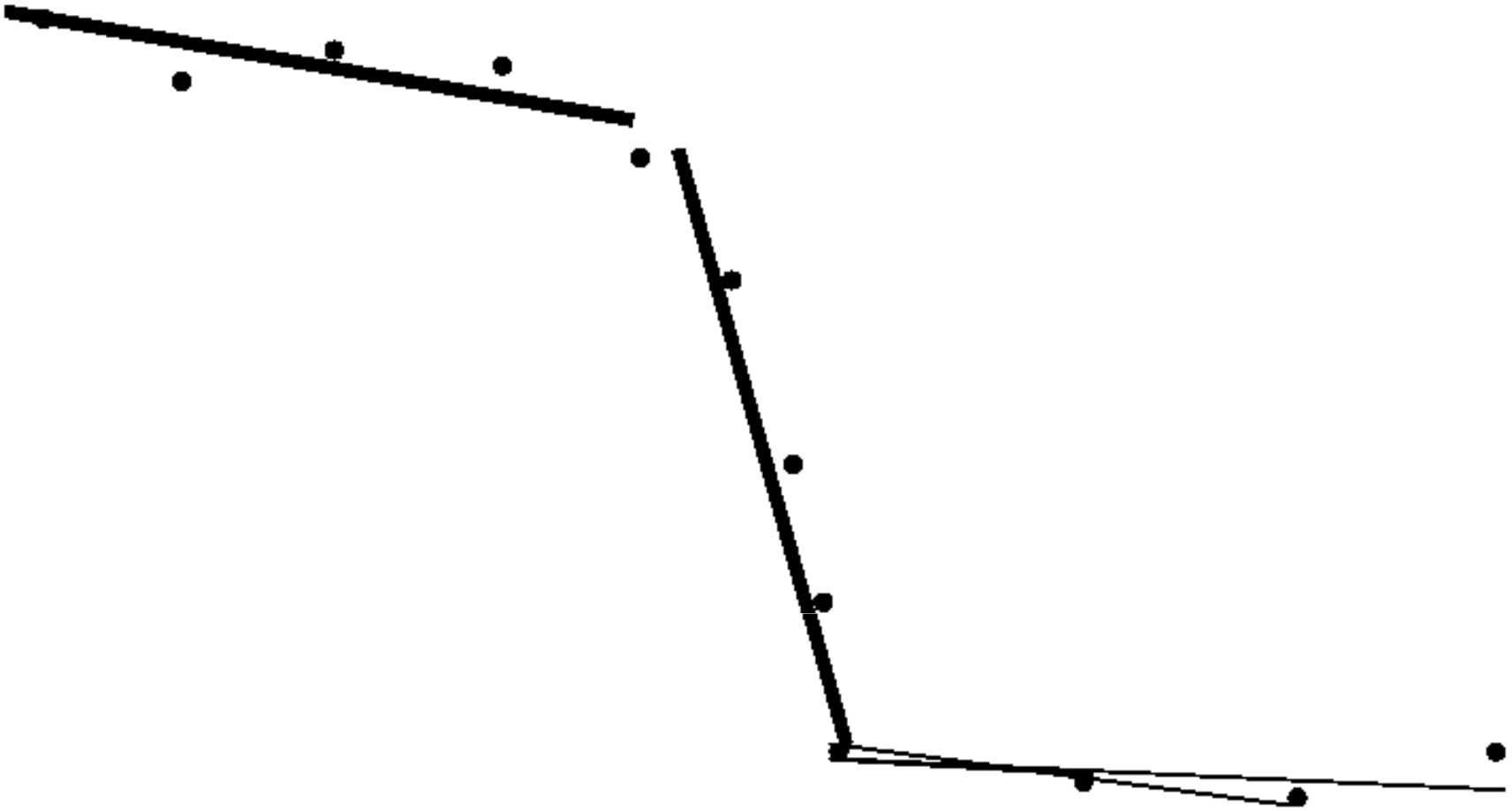
Adapted from Trevor Darrell, MIT

Model fitting: incremental line fitting



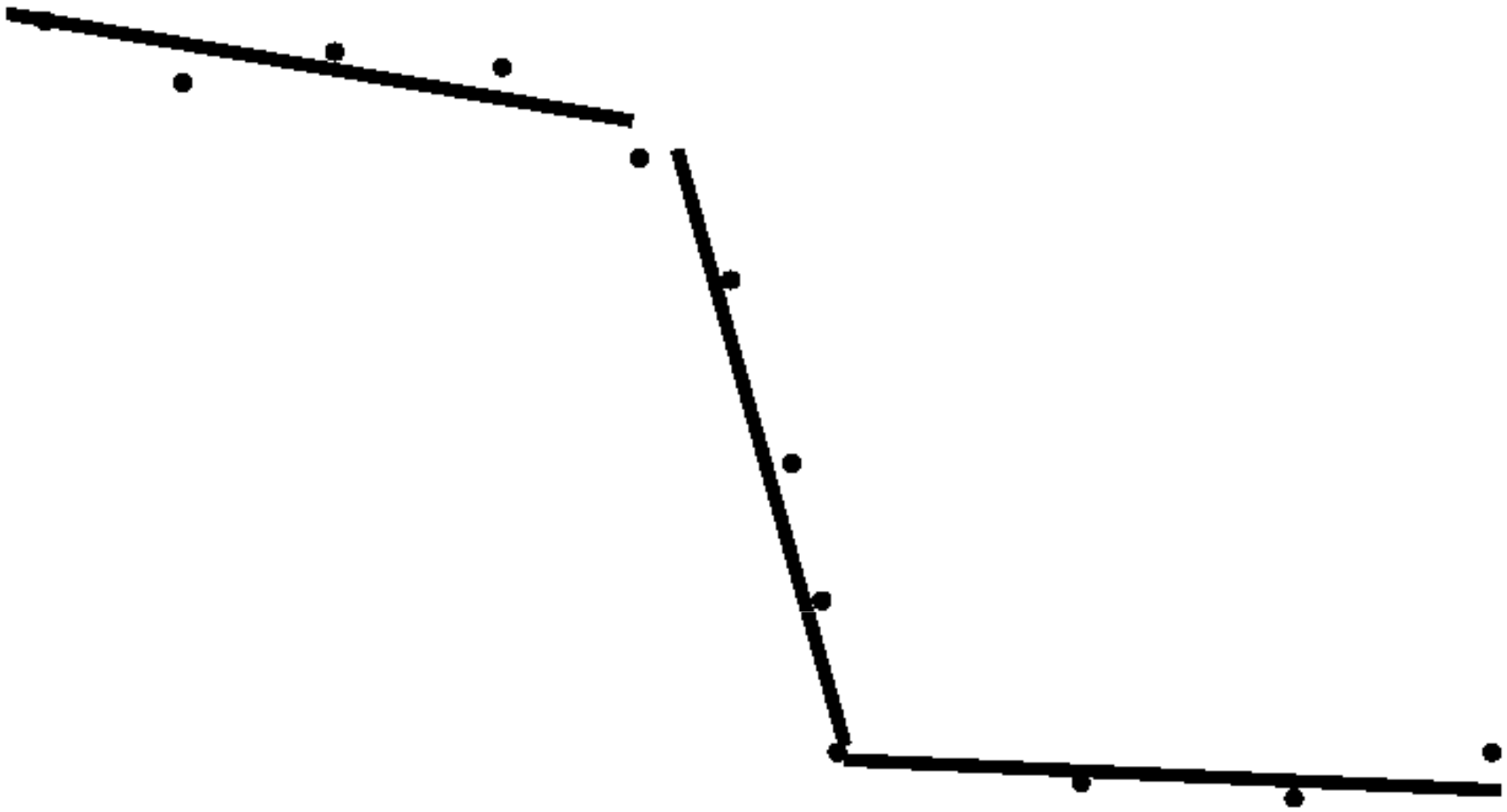
Adapted from Trevor Darrell, MIT

Model fitting: incremental line fitting



Adapted from Trevor Darrell, MIT

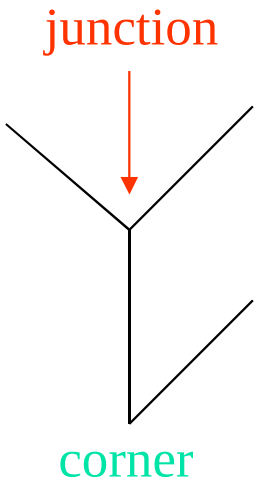
Model fitting: incremental line fitting



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.



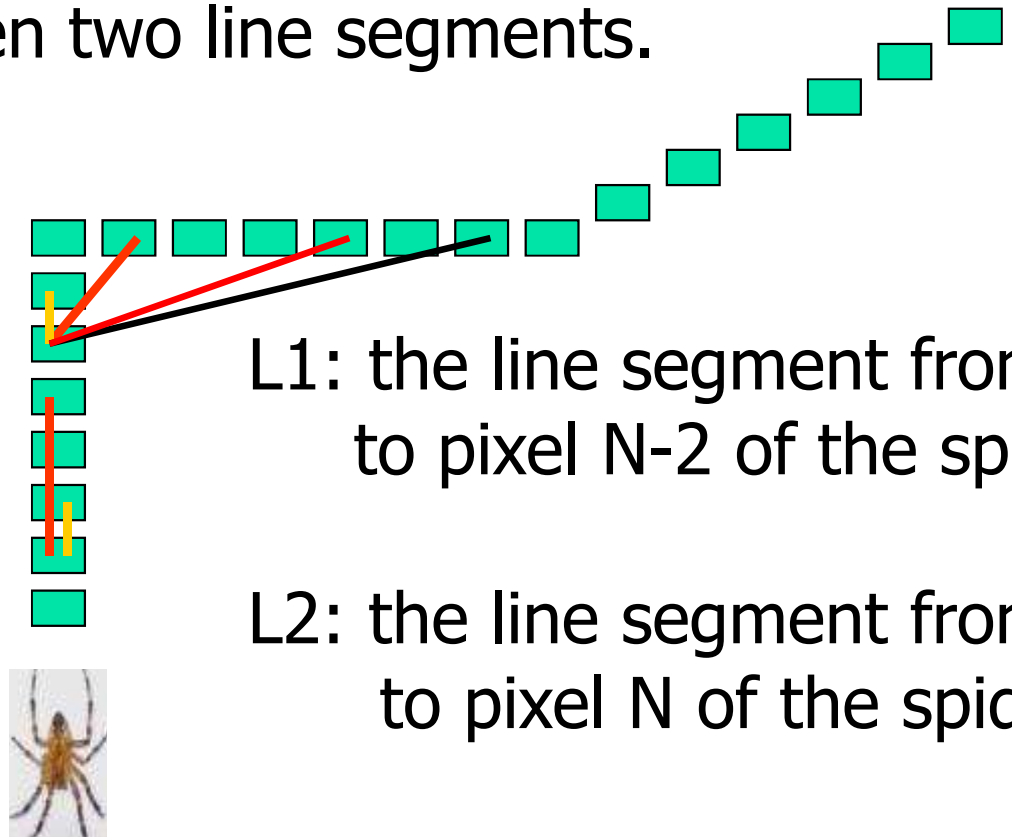
Adapted from Linda Shapiro, U of Washington

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

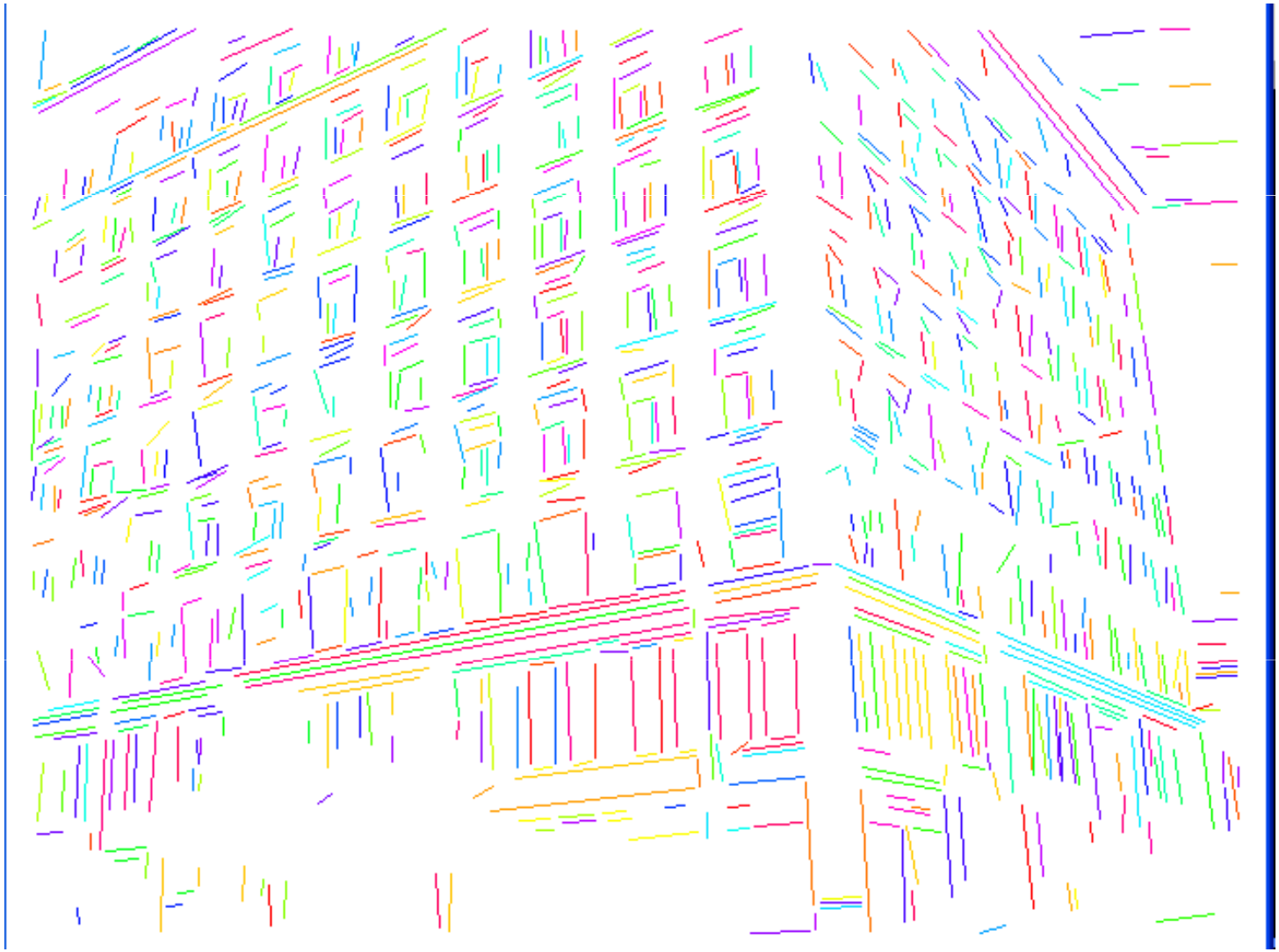
The angle must be <= arctan(2/length(L2))

Longer spiders allow less of an angle.

Edge tracking: ORT Toolkit

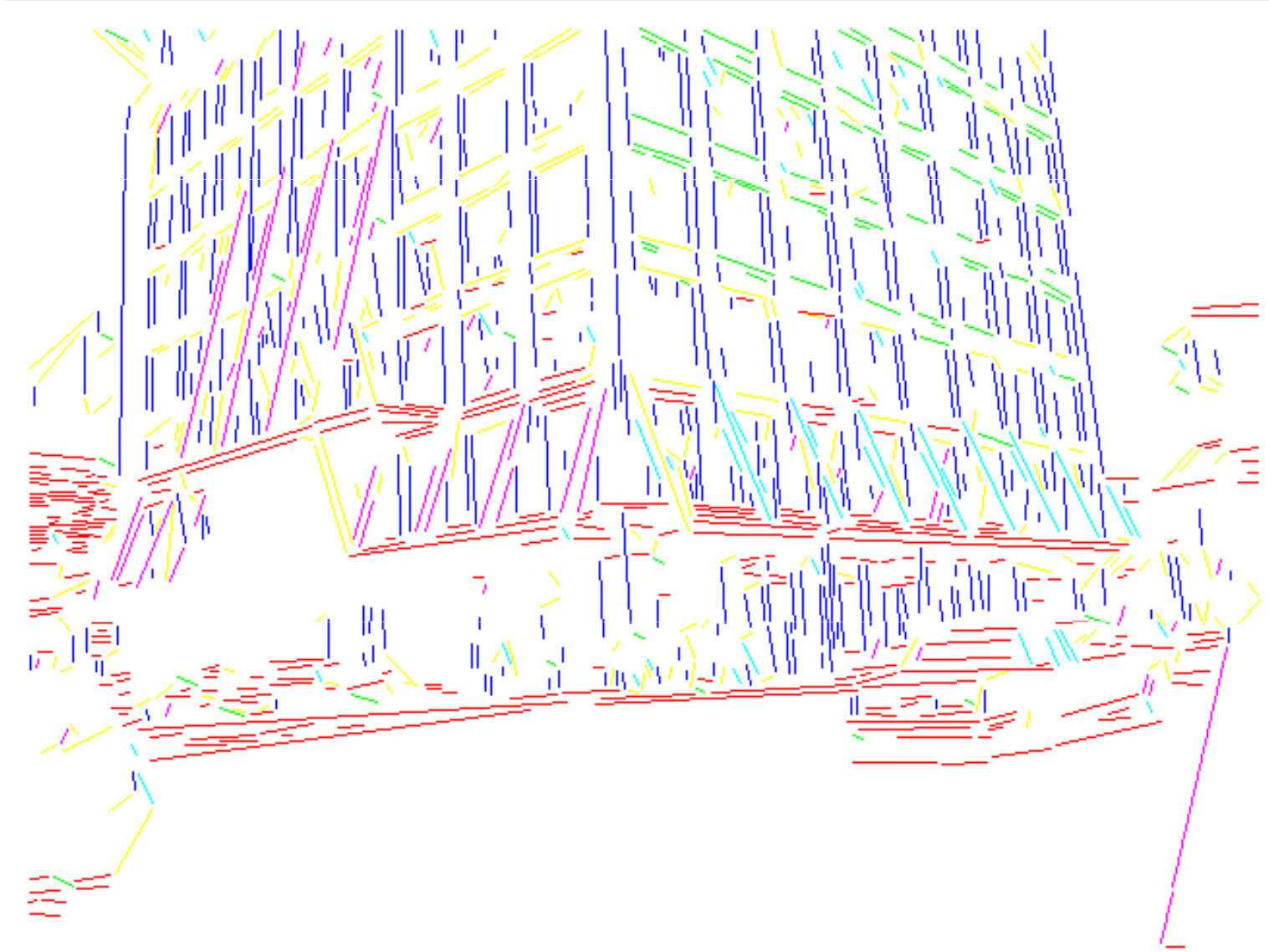
- The parameters are the length of the spider and the number of pixels per step.
- These parameters can be changed to allow for less sensitivity, so that we get longer line segments.
- The algorithm has a final phase in which adjacent segments whose angle differs by less than a given threshold are joined.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

Example: building detection



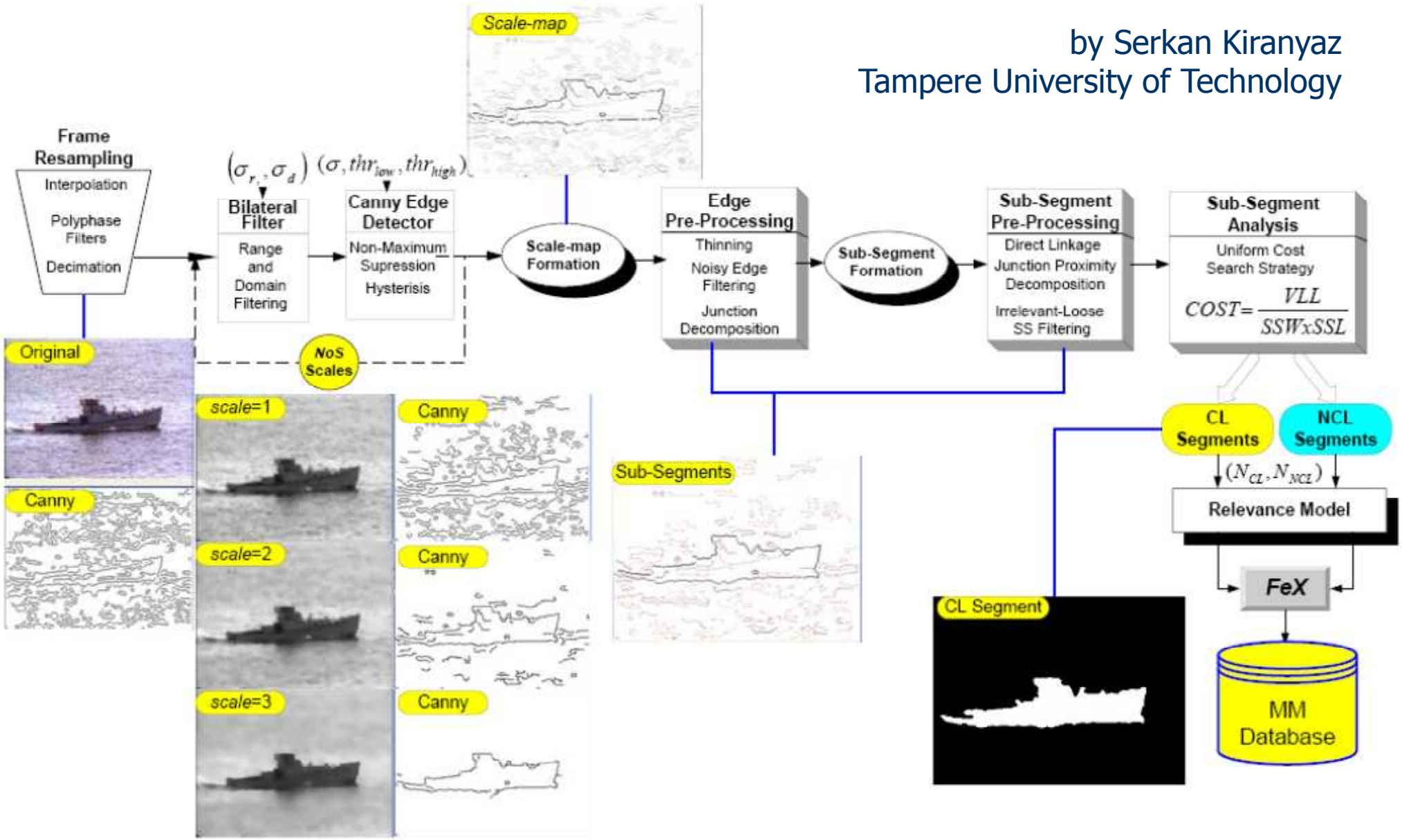
by Yi Li @ University of Washington

Example: building detection



Example: object extraction

by Serkan Kiranyaz
Tampere University of Technology



Example: object extraction

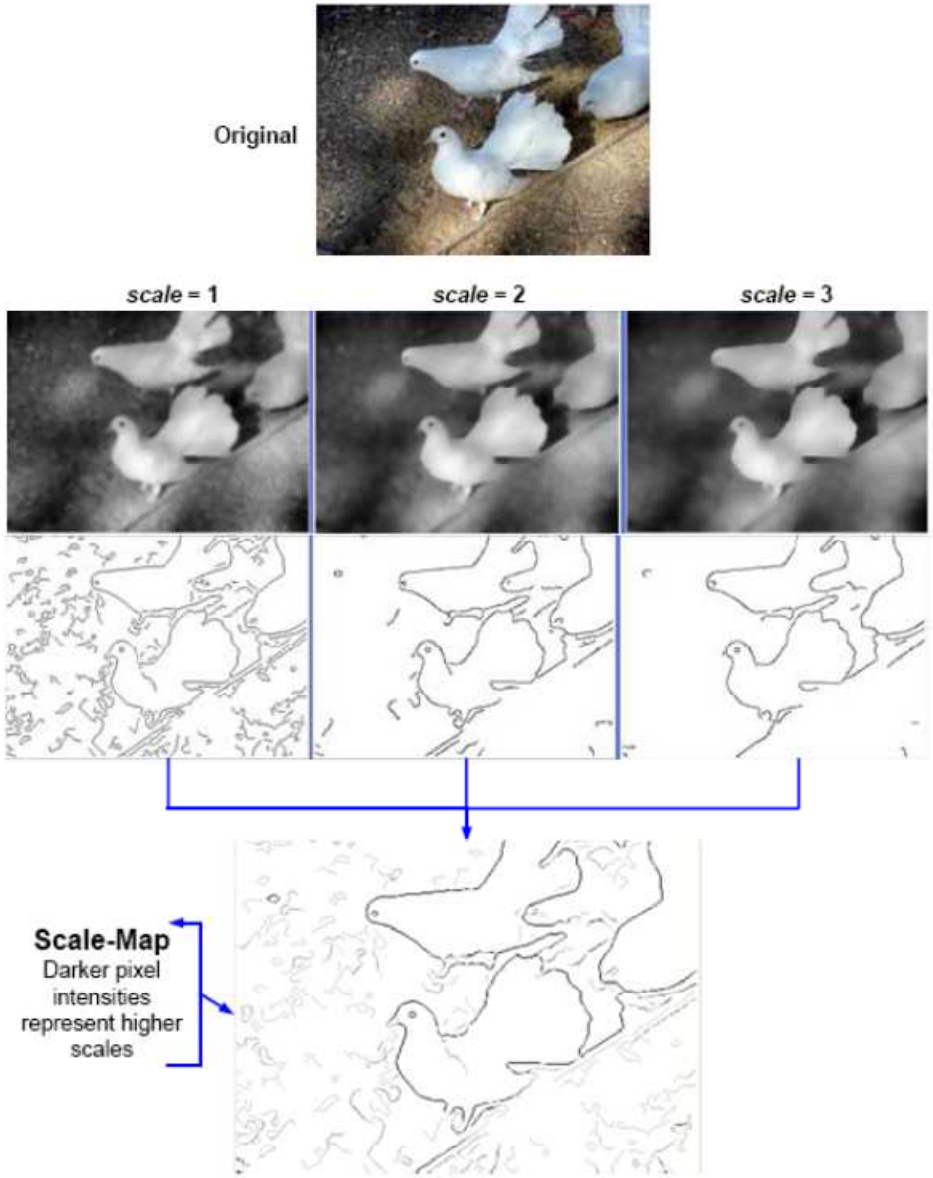


Figure 5: A sample scale-map formation.

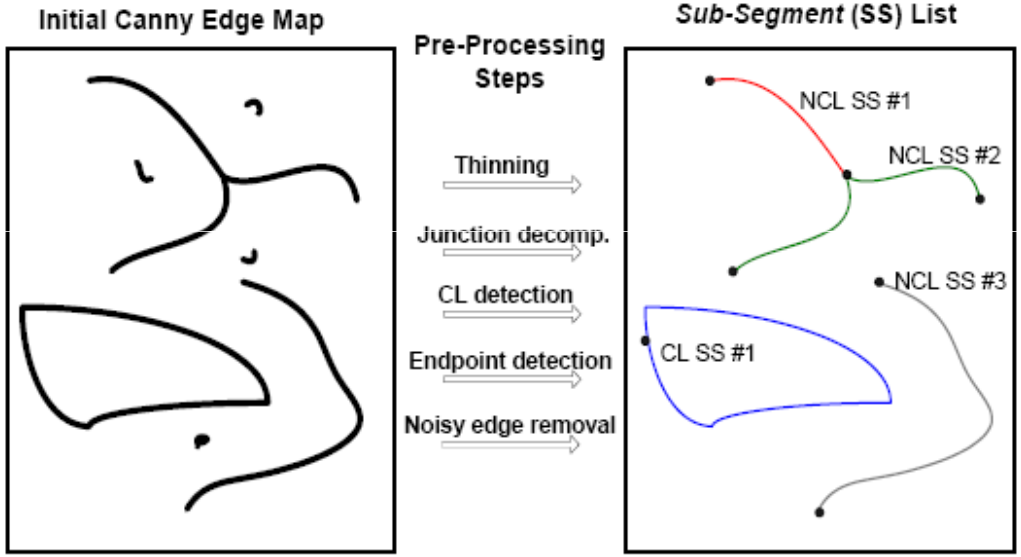


Figure 6: Sub-segment formation from an initial Canny edge field.

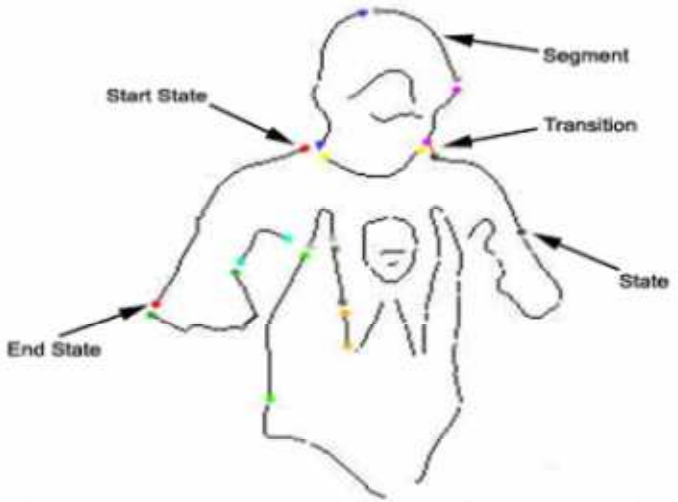


Figure 9: State space for a given sub-segment layout.

Example: object extraction

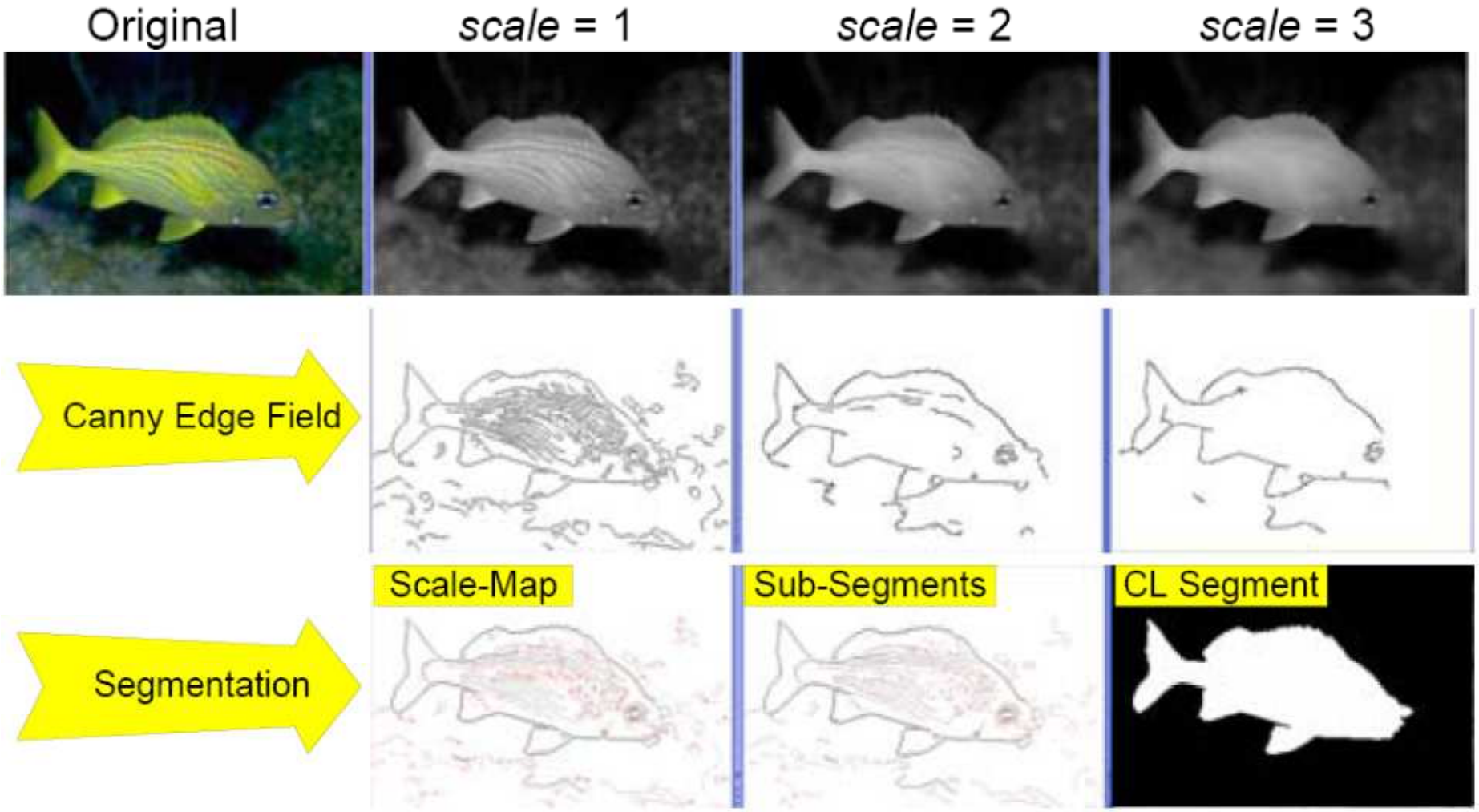
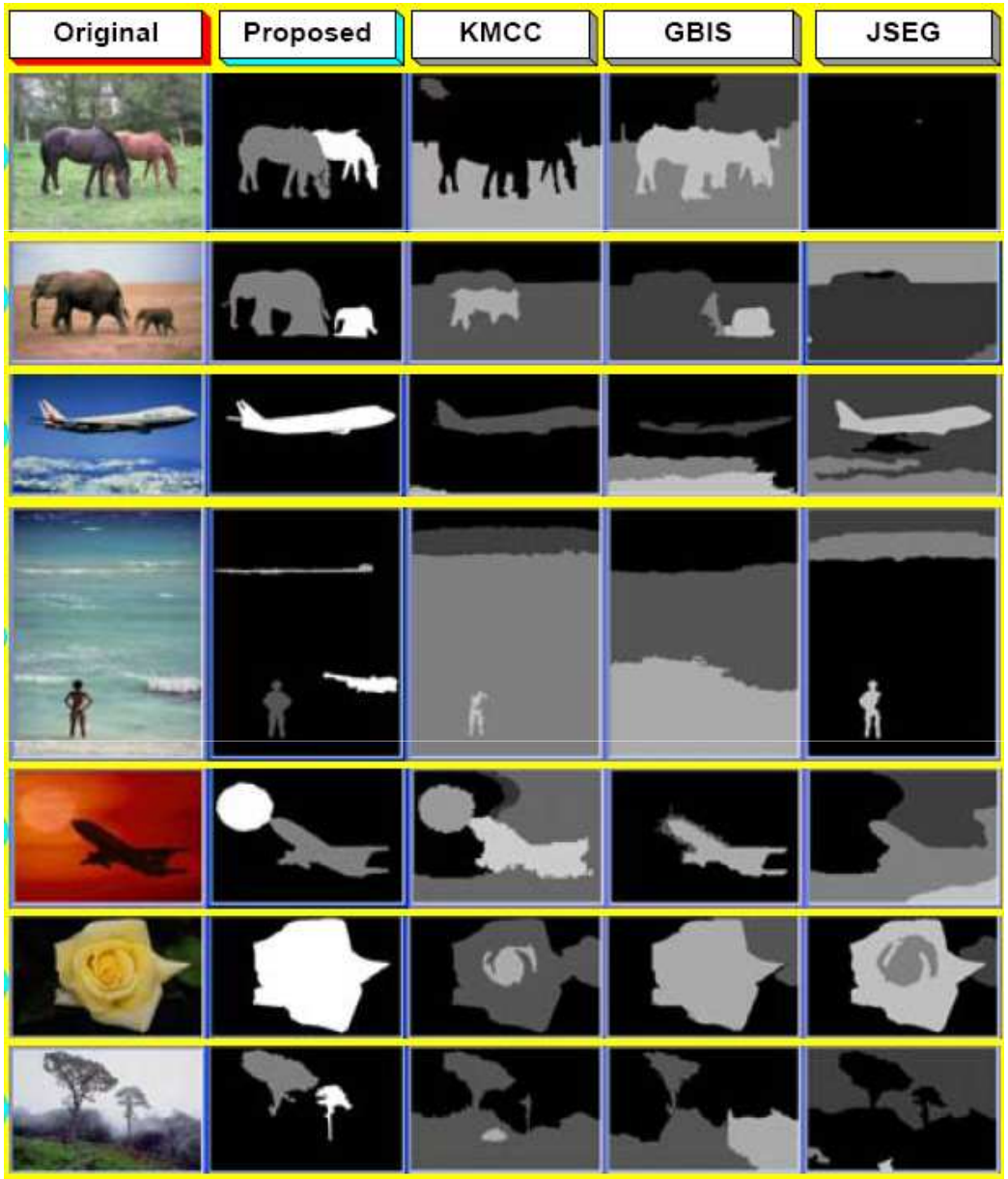
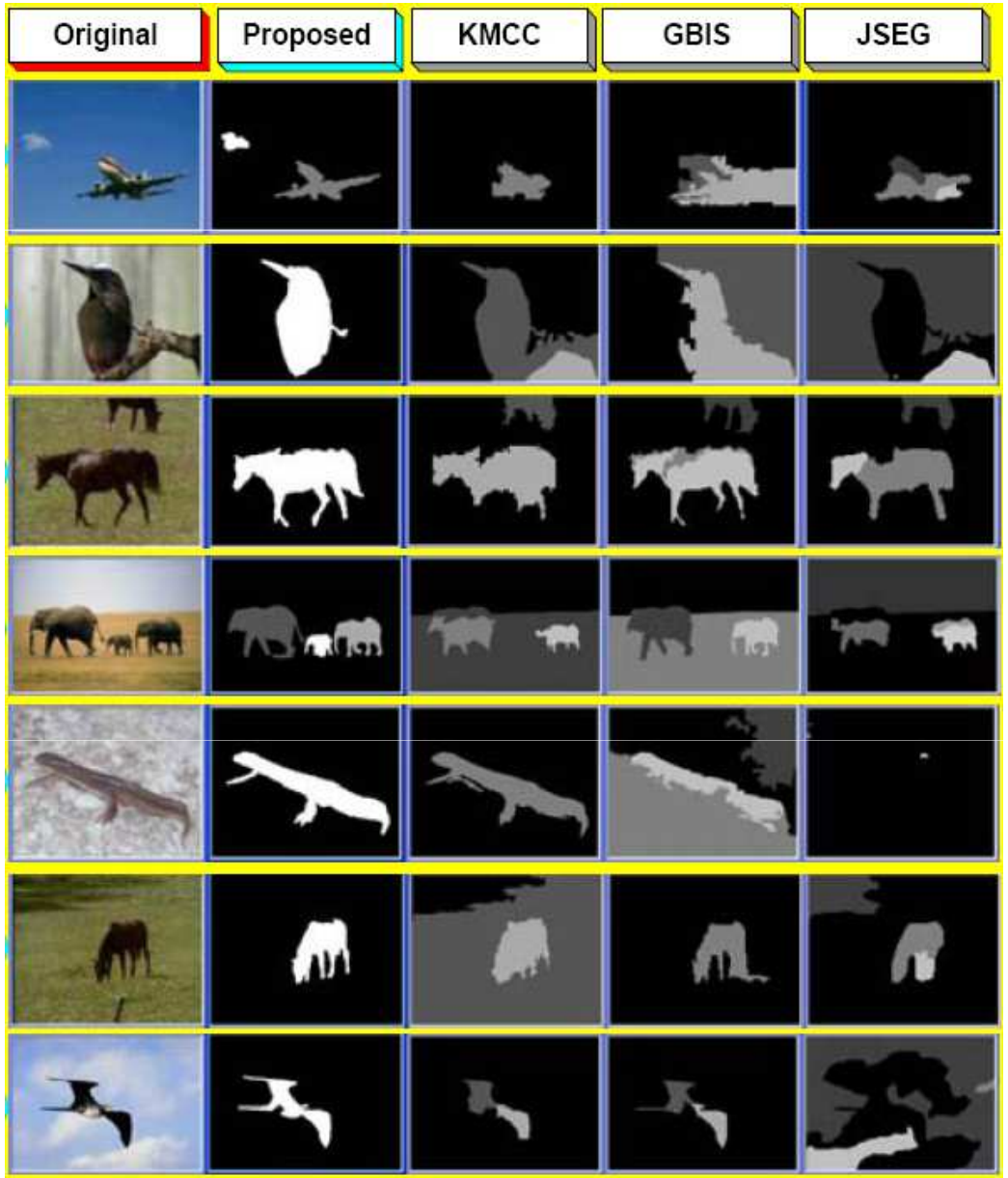


Figure 12: 3-scale simplification process over a natural image and the final CL segment extracted.

Example: object extraction

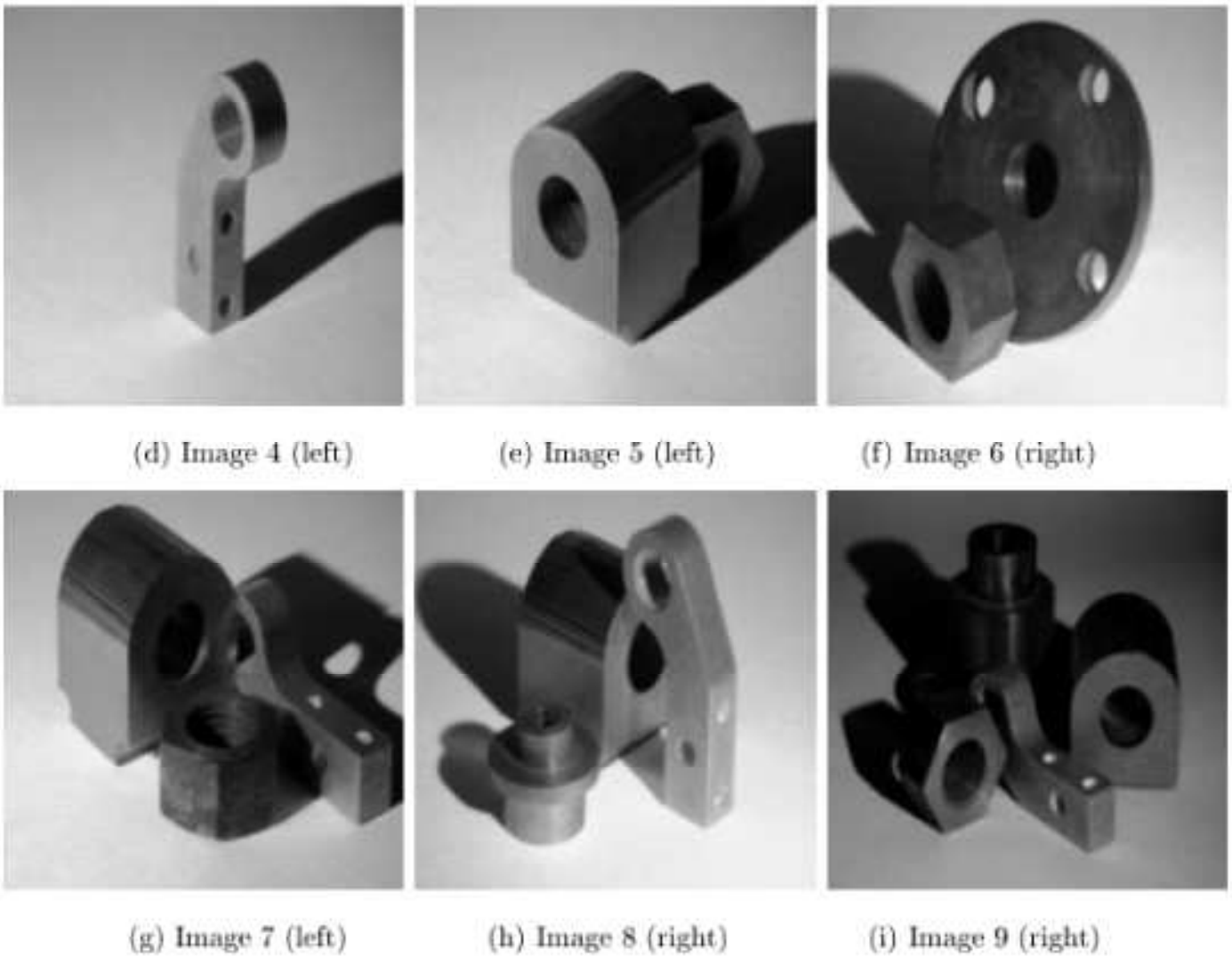




Example: object recognition

- 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



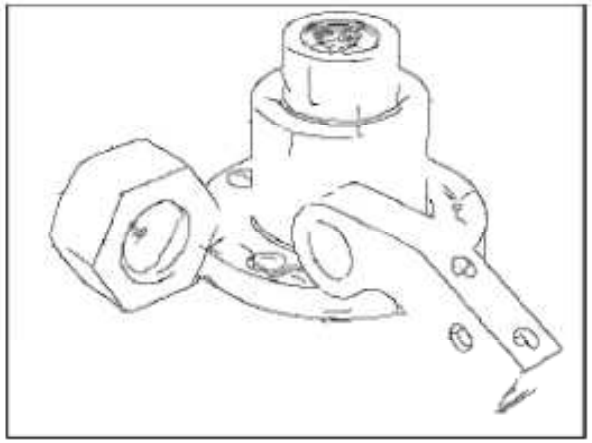
(d) Image 4 (left) (e) Image 5 (left) (f) Image 6 (right)

(g) Image 7 (left) (h) Image 8 (right) (i) Image 9 (right)

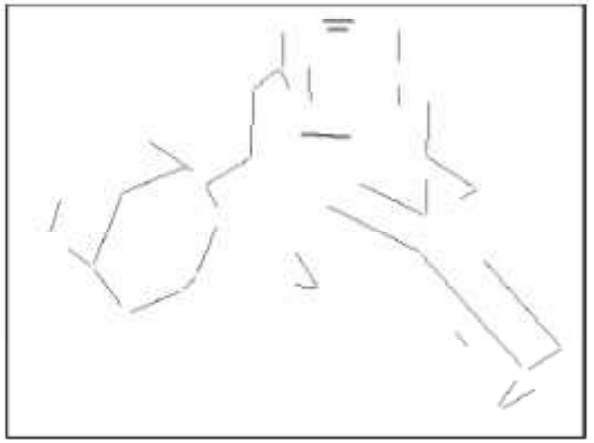
Example scenes used. The labels "left" and "right" indicate the direction of the light source.

(a) Original left image

(b) Original right image

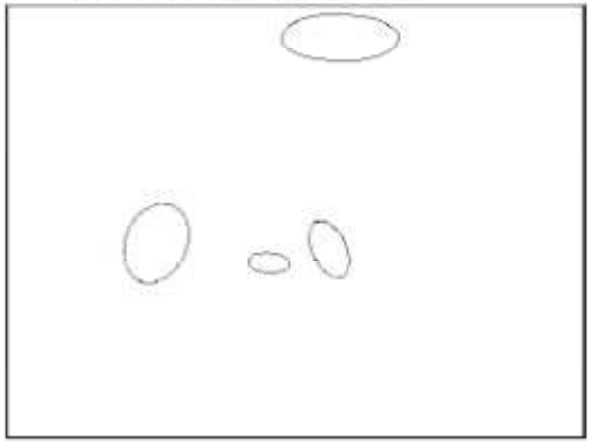


(c) Combined edge image



(d) Linear features detected

(e) Circular arc features detected



(f) Ellipses detected

Figure 22: Sample run of the system.

Example: object recognition

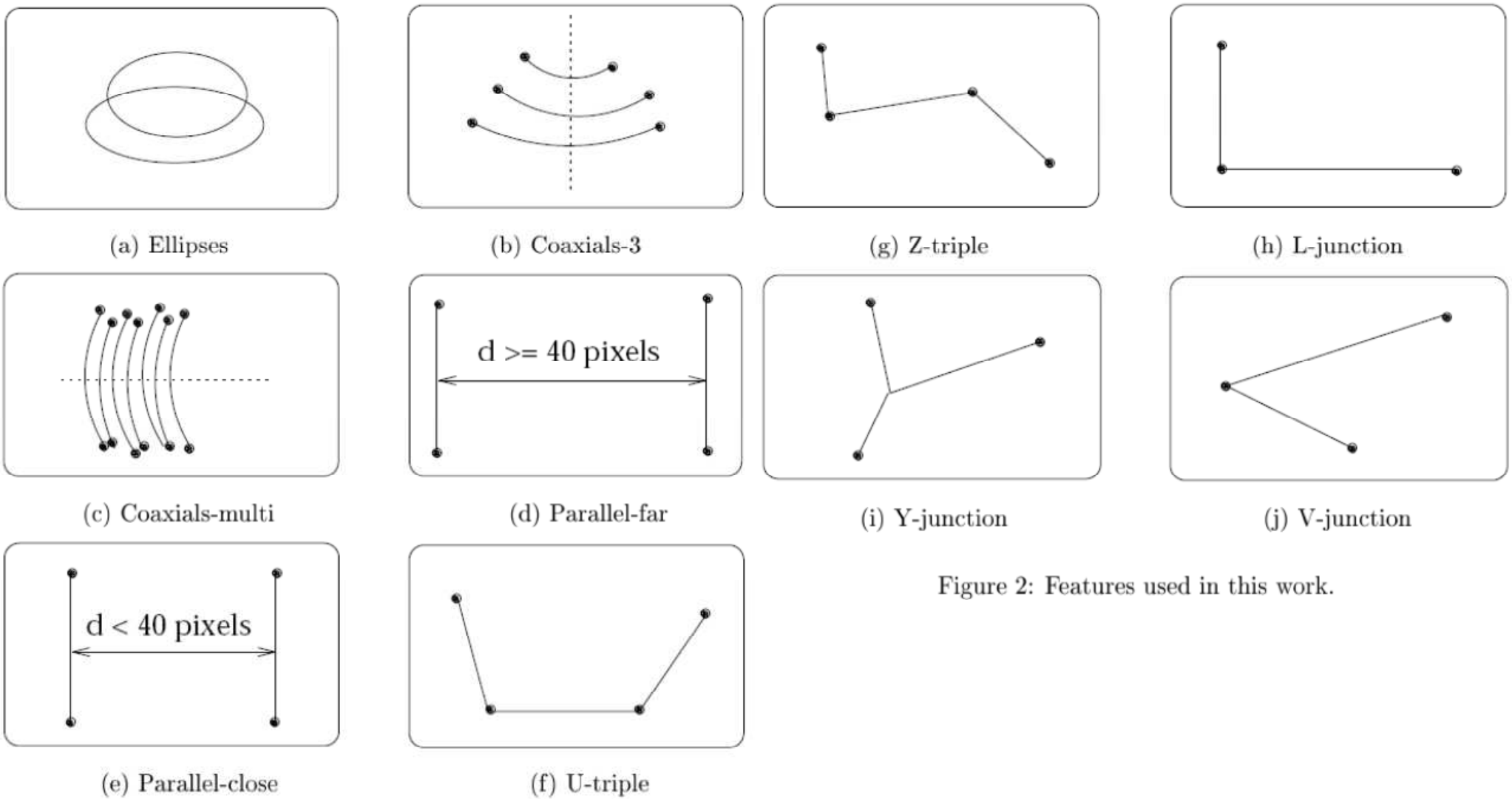
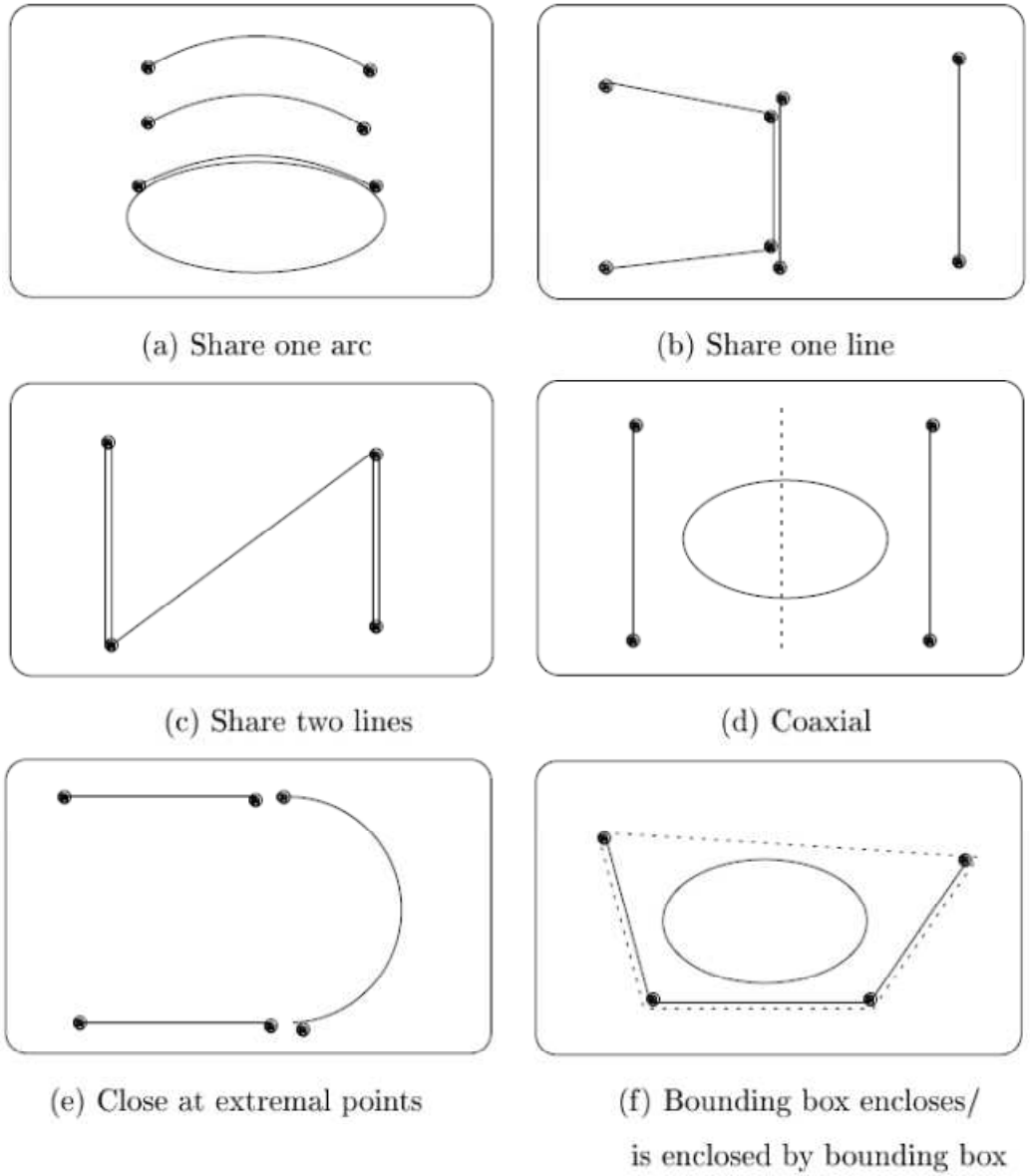


Figure 2: Features used in this work.

Example: object recognition



(a) Share one arc

(b) Share one line

(c) Share two lines

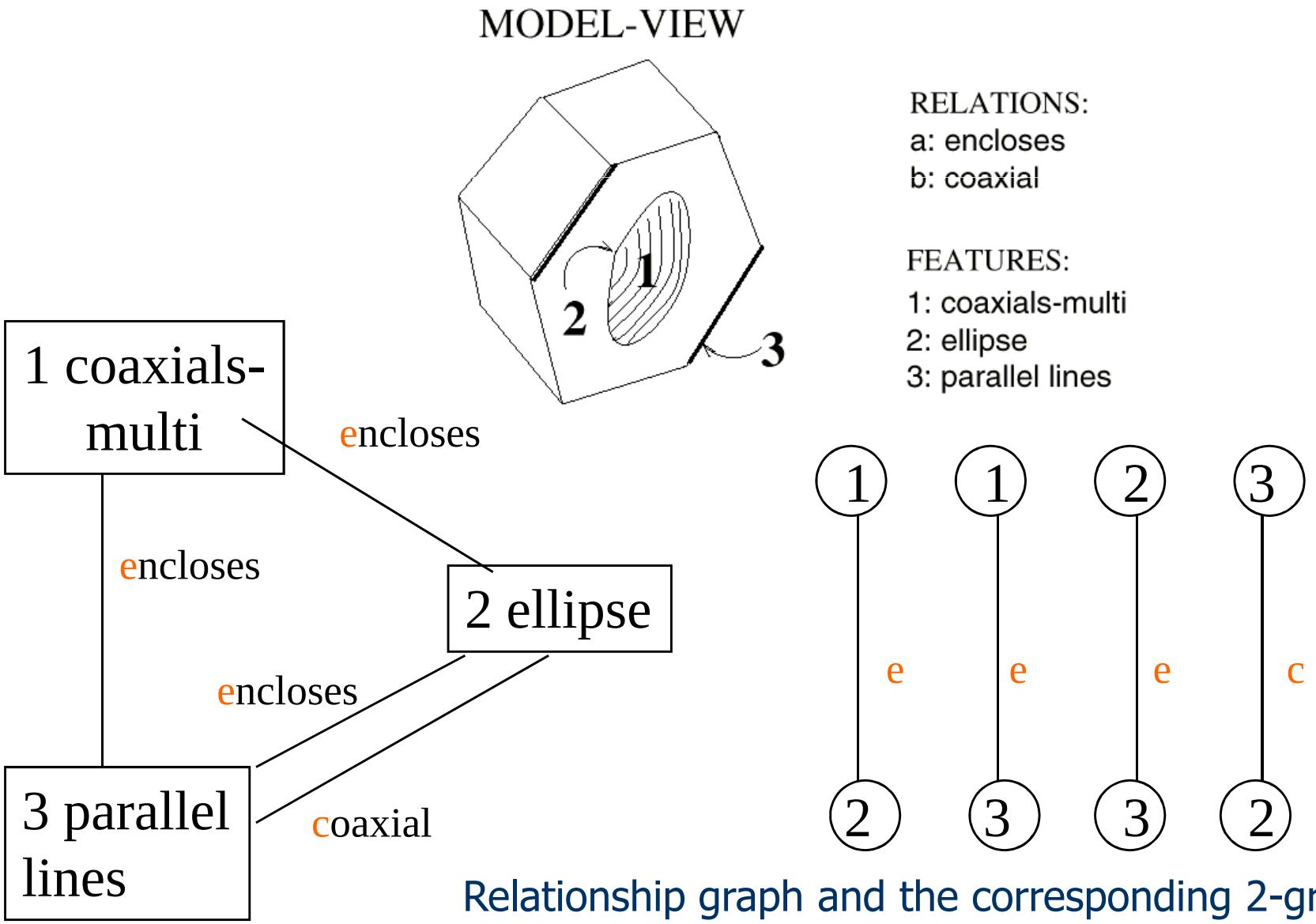
(d) Coaxial

(e) Close at extremal points

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

Figure 3: Relations between sample pairs of features.

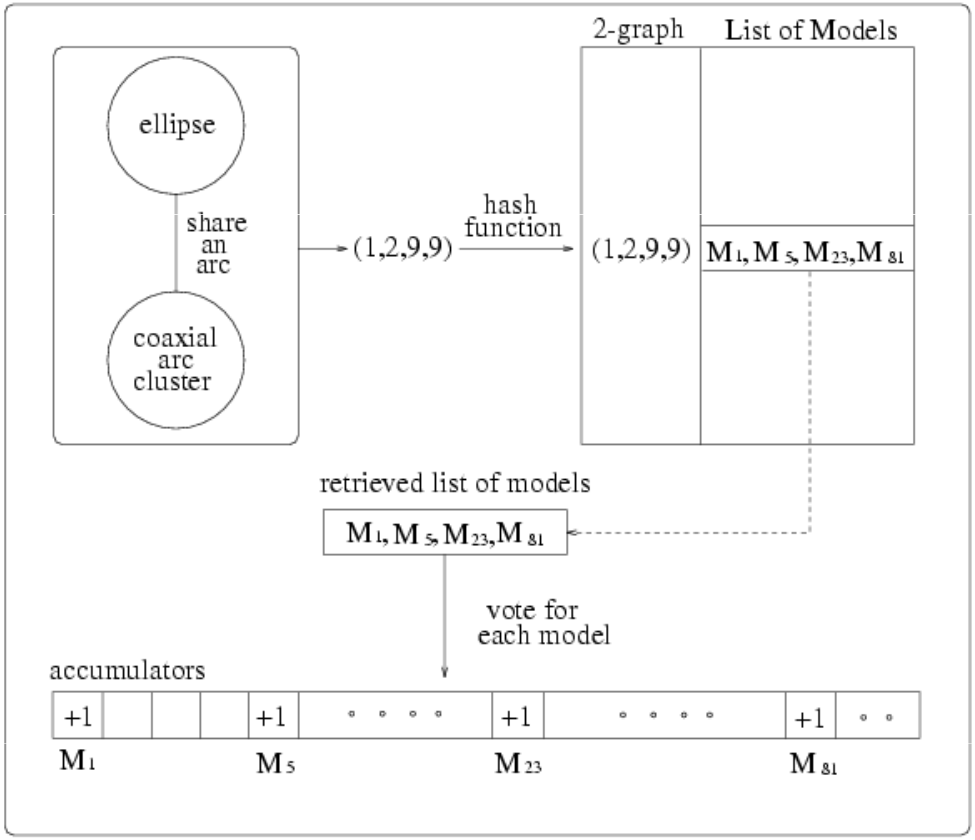
Example: object recognition



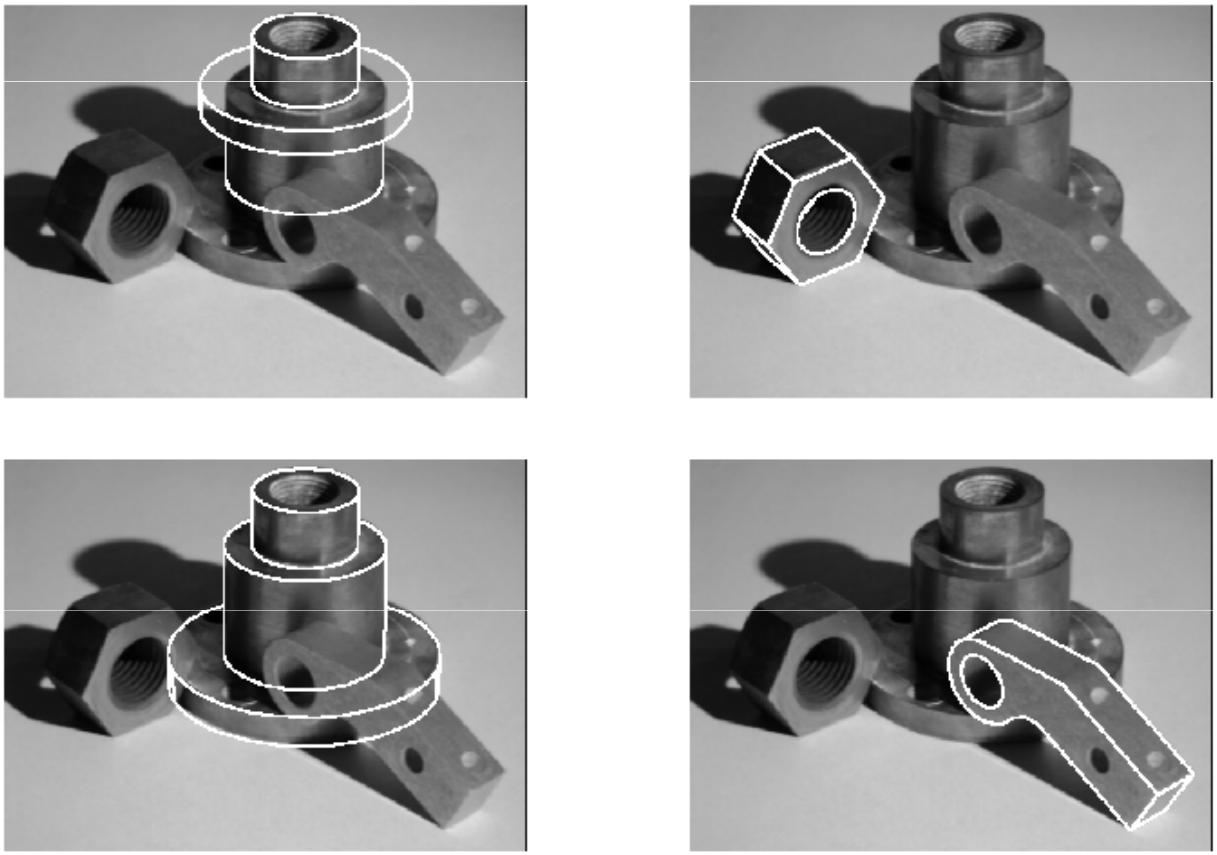
Relationship graph and the corresponding 2-graphs.

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



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.