Digital Image Processing

Prof. Tiago Vieira, PhD

Universidade Federal de Alagoas

tvieira@ic.ufal.br

August 13, 2019

Contents

1. Introduction
2. Background
3. Point, line and edge detection
4. Thresholding
5. Region-Based Segmentation
6. Watershed Segmentation Using The Watershed Transform

- The segmentation of an image divides it into parts (or objects).
- Segmentation is crucial to a successful image analysis.

Segmentation of monochrome images are based on pixel intensity properties:

- Discontinuities.
- Similarity.

Discontinuity:

- Abrupt changes in intensity levels.
- Detection of isolated points, lines and borders.

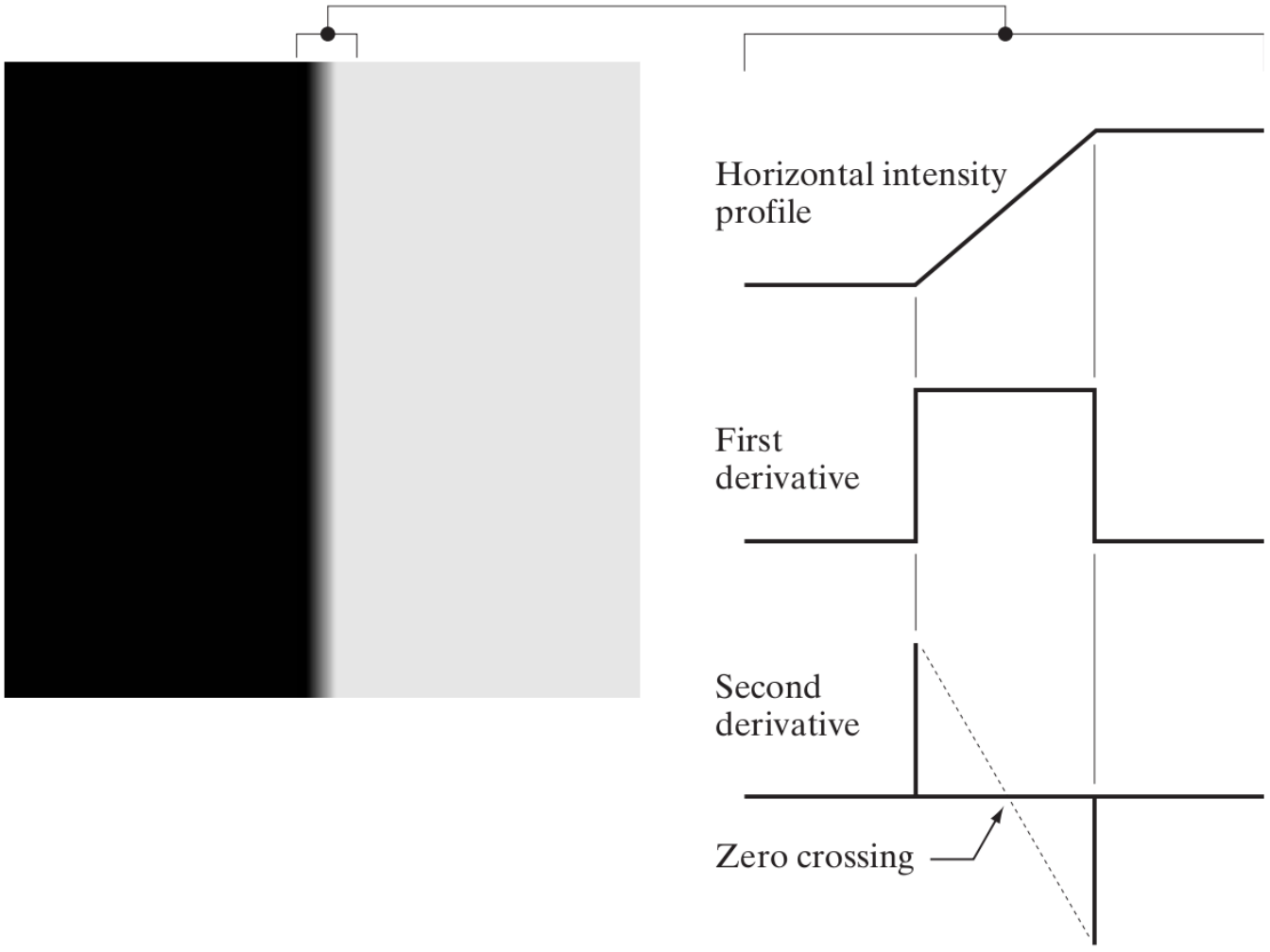
Similarity:

- Thresholding.
- Region growing.
- Region division and fusion.

- Three discontinuities:
  1. Points;
  2. Lines;
  3. Borders.
- Local changes in intensity can be detected using derivatives.
- Derivatives of a digital function are defined in terms of differences.

- Approximations of first derivative must be:
  1. Zero in regions of constant intensity;
  2. Nonzero on the onset of an intensity step or ramp;
  3. Nonzero at points along an intensity ramp.

- Approximations of second derivatives must be:
  1. Zero in regions of constant intensity;
  2. Nonzero at the onset *and* end of an intensity step or ramp;
  3. Zero along intensity ramps.



a b

FIGURE 10.10
(a) Two regions of constant intensity separated by an ideal vertical ramp edge. (b) Detail near the edge, showing a horizontal intensity profile, together with its first and second derivatives.

First order derivative approximation:

$$\frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x). \tag{1}$$

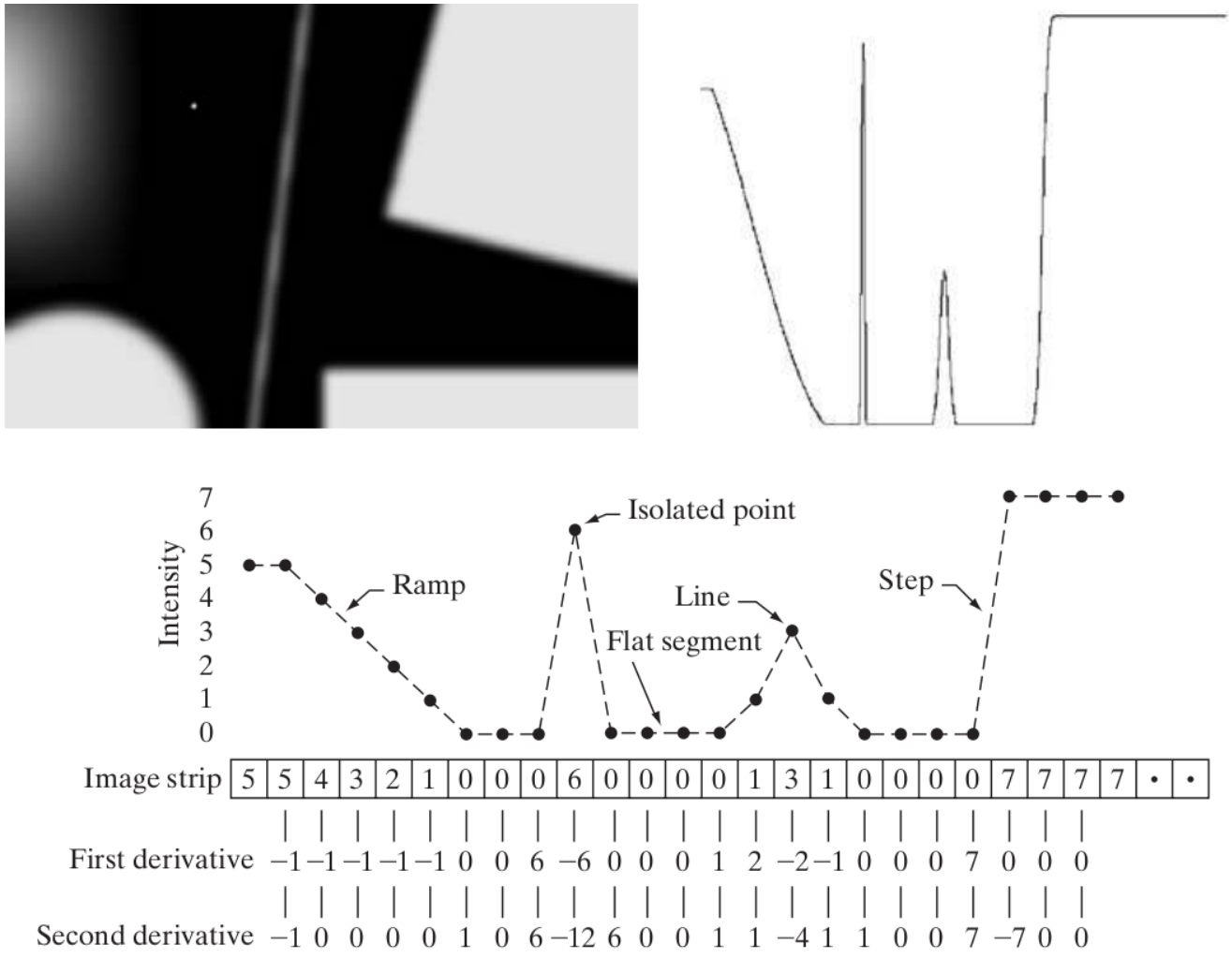
Second order derivative approximation:

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial f'(x)}{\partial x} = f'(x+1) - f'(x)$$

$$= f(x+2) - f(x+1) - f(x+1) + f(x)$$

$$= f(x+2) - 2f(x+1) + f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f''(x) = f(x+1) + f(x-1) - 2f(x). \tag{2}$$



a b
c

FIGURE 10.2 (a) Image. (b) Horizontal intensity profile through the center of the image, including the isolated noise point. (c) Simplified profile (the points are joined by dashes for clarity). The image strip corresponds to the intensity profile, and the numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10.2-1) and (10.2-2).

To find derivatives we scan the image using a mask.

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9$$

$$R = \sum_{k=1}^{9} w_k z_k$$

z_i is the gray-scale intensity of *pixel* associated with the coefficient w_i of the mask.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 10.3
A general 3×3 spatial filter mask.

Points are detected based on the Laplacian:

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

$$\nabla^2 f(x,y) = f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)-4f(x,y). \quad (3)$$

which can be implemented by following masks:

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

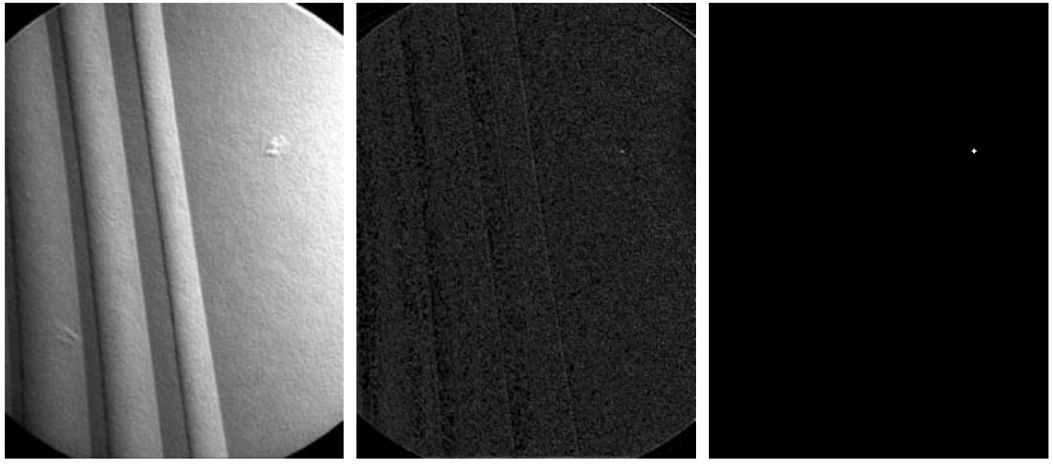
−1	−1	−1
−1	8	−1
−1	−1	−1

a b
c d

FIGURE 3.37
(a) Filter mask used to implement Eq. (3.6-6).
(b) Mask used to implement an extension of this equation that includes the diagonal terms.
(c) and (d) Two other implementations of the Laplacian found frequently in practice.

Example:

1	1	1
1	−8	1
1	1	1



a
b c d

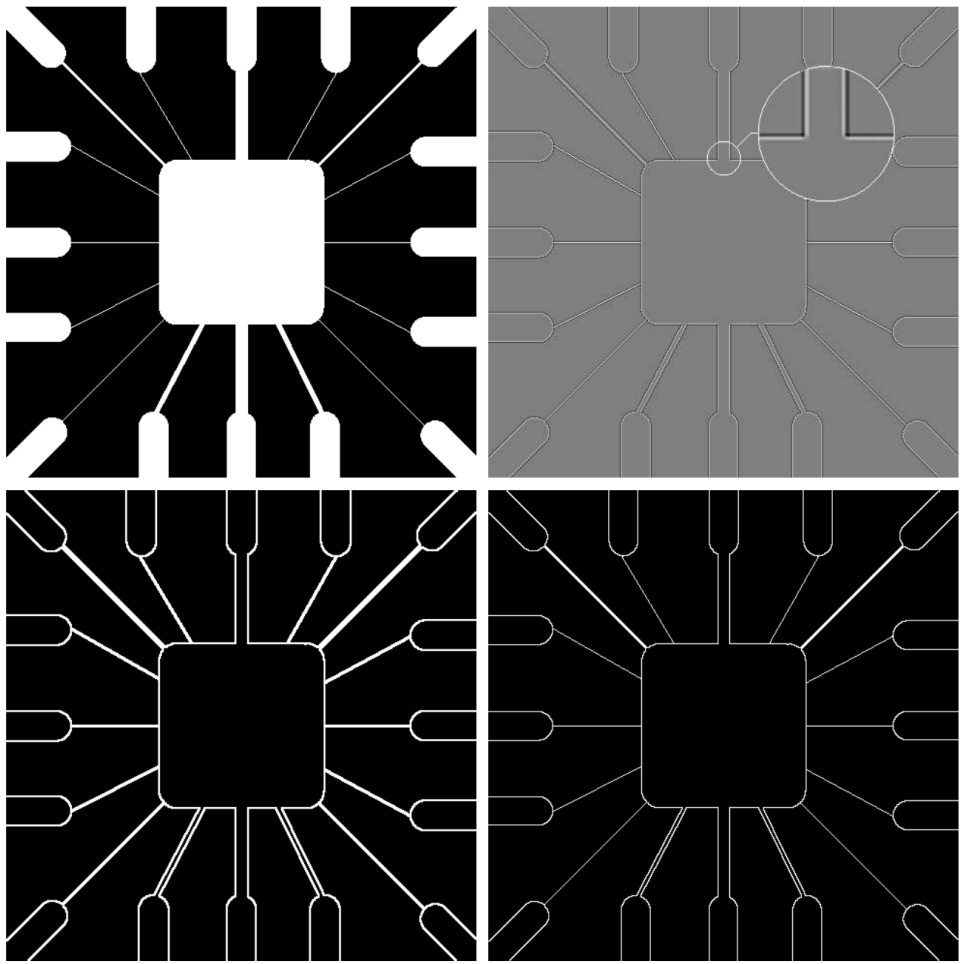
FIGURE 10.4
(a) Point detection (Laplacian) mask. (b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel. (c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

- Point is detected if $R > T$, where T is a threshold.
- Weighed difference between central pixel and its neighbors.

Laplacian for isotropic[1] line detection.

a b
c d

FIGURE 10.5
(a) Original image. (b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian. (c) Absolute value of the Laplacian. (d) Positive values of the Laplacian.

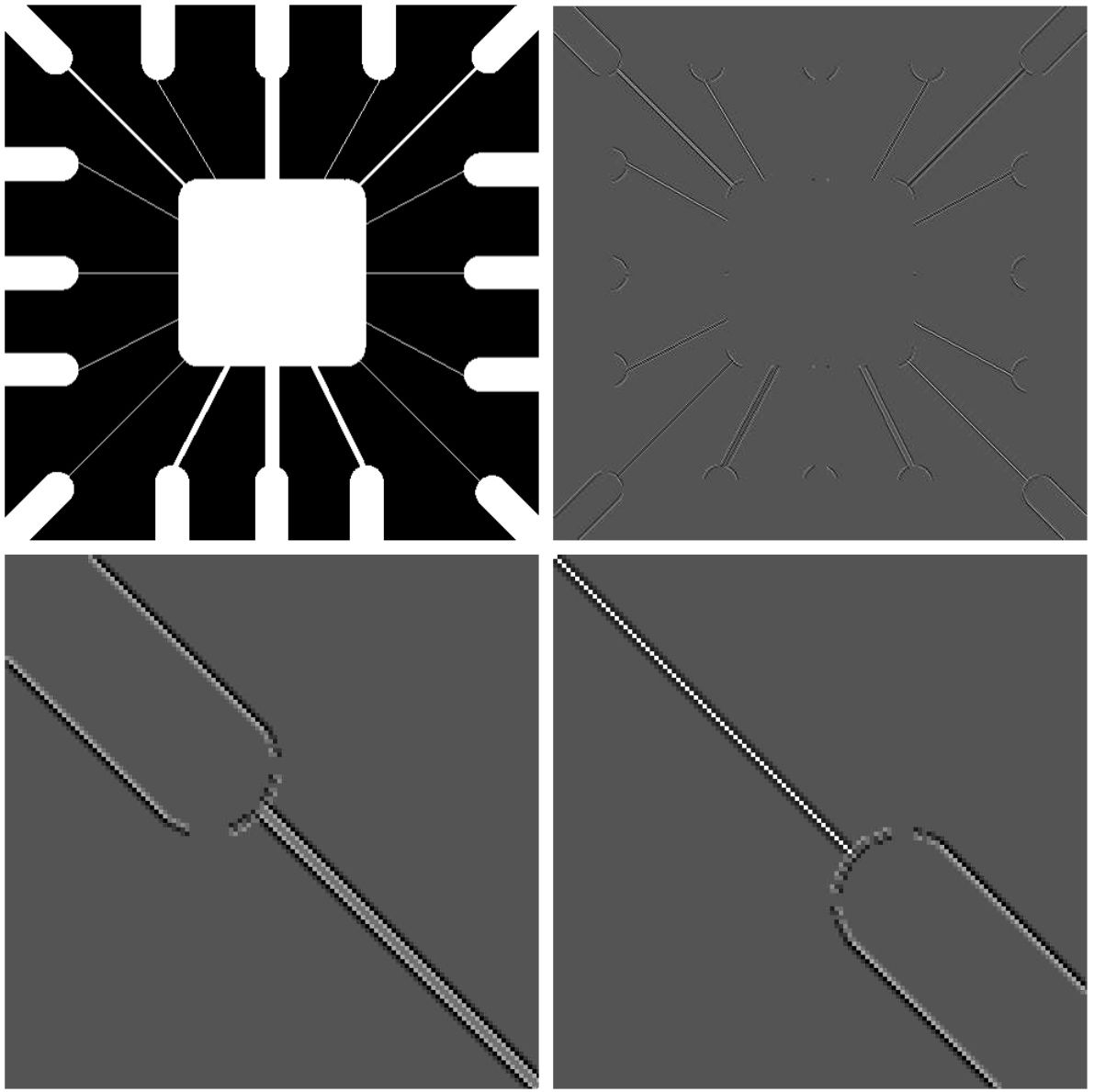


[1] Insensitive to line direction.

- Let $R_k,\ k = 1, \ldots, 4$ be the responses of the individual spatial filtering of the masks below with an image I.
- For a point, if $|R_k| > |R_j|, \forall k \neq j$, then the point is more likely associated with a line with direction k.

−1	−1	−1	2	−1	−1	−1	2	−1	−1	−1	2
2	2	2	−1	2	−1	−1	2	−1	−1	2	−1
−1	−1	−1	−1	−1	2	−1	2	−1	2	−1	−1
Horizontal			+45°			Vertical			−45°		

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



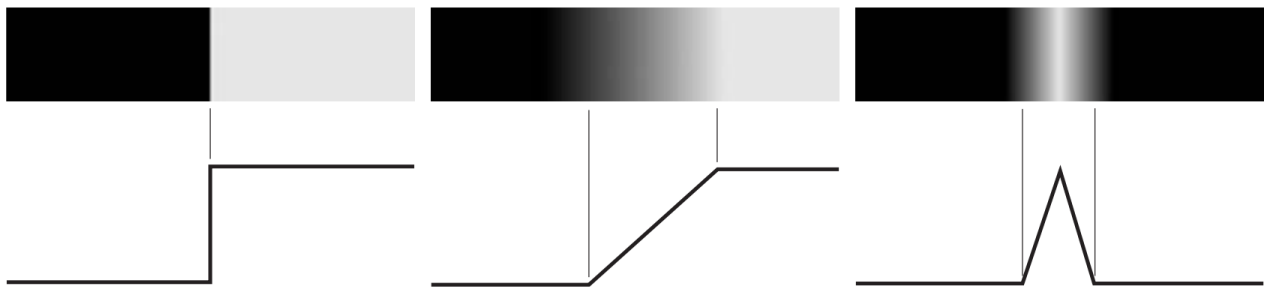
a b
c d
e f

FIGURE 10.7
(a) Image of a wire-bond template.
(b) Result of processing with the +45° line detector mask in Fig. 10.6.
(c) Zoomed view of the top left region of (b).
(d) Zoomed view of the bottom right region of (b). (e) The image in (b) with all negative values set to zero. (f) All points (in white) whose values satisfied the condition $g \geq T$, where g is the image in (e). (The points in (f) were enlarged to make them easier to see.)

- Step edges occur:
  - In computer generated images (animation, solid modeling).
- In practice, edges are blurred due to:
  - Focus (optical apparatus).
  - Noise (electronics).

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.



Images may contain all three types of edges:

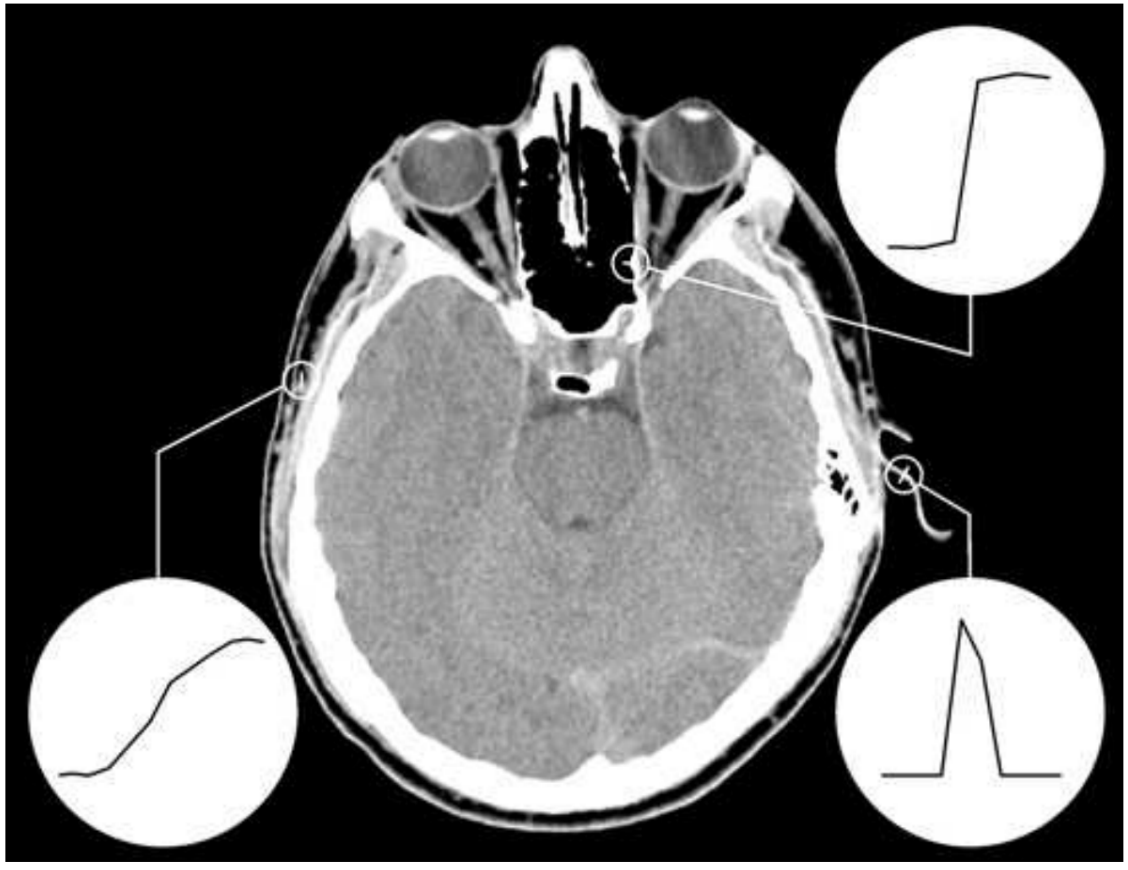
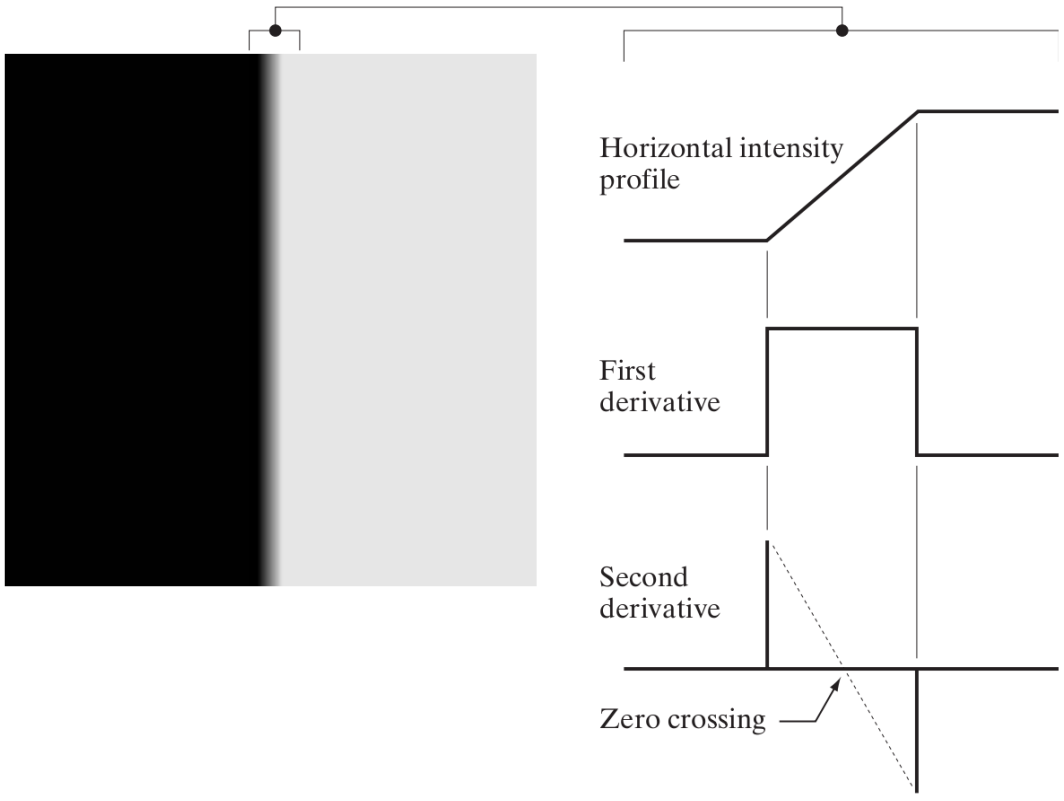


FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and "step" profiles span 9 pixels and 2 pixels, respectively.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

- The magnitude of the 1st der. detects an edge.
- The 2nd derivative:
  1. Has 2 values for an edge.
  2. Its zero crossings locate the center of the edge.



a b

FIGURE 10.10
(a) Two regions of constant intensity separated by an ideal vertical ramp edge. (b) Detail near the edge, showing a horizontal intensity profile, together with its first and second derivatives.

Problems with noisy images:

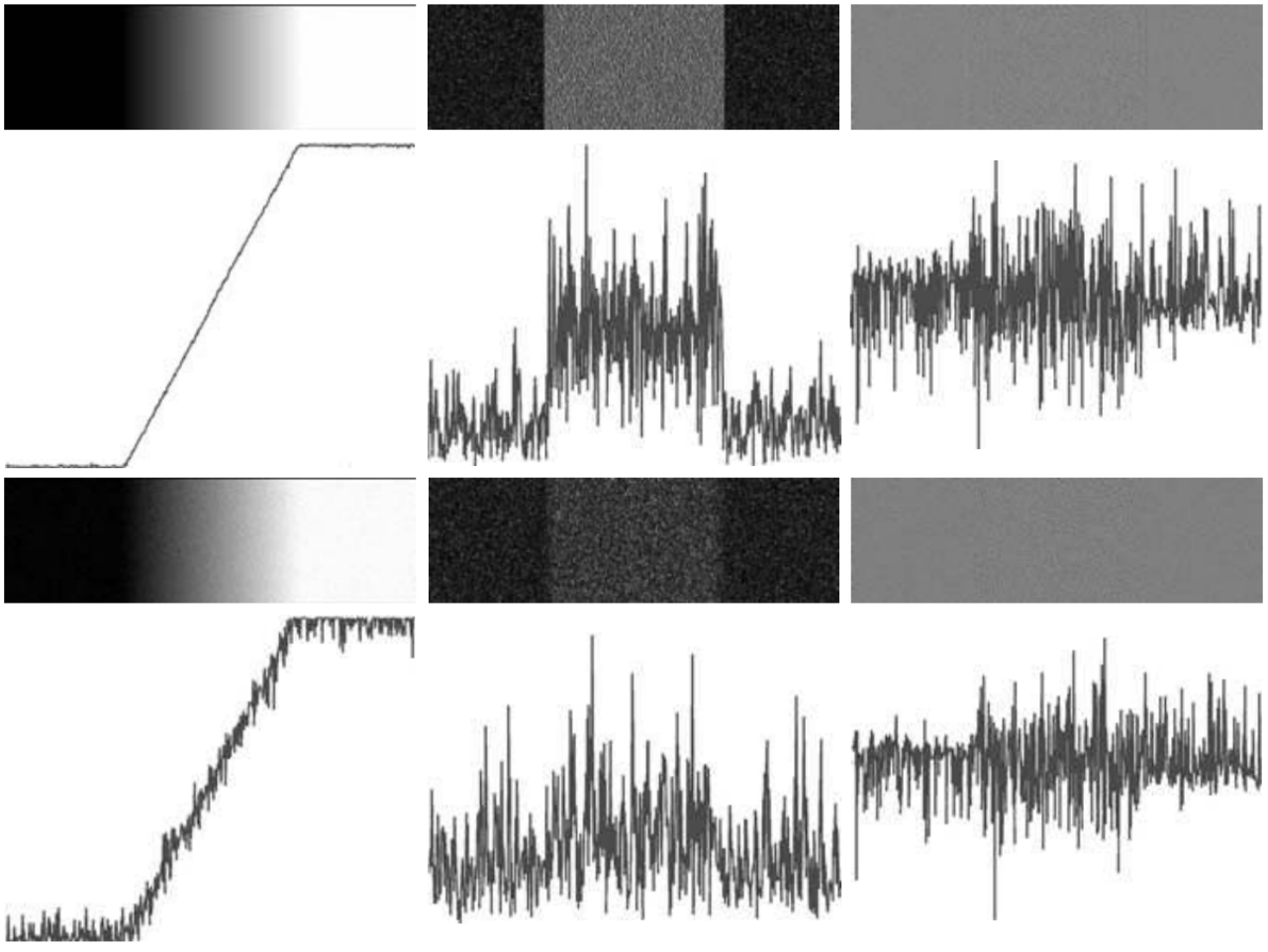


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images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

Fundamental steps in edge detection

1. Image smoothing for noise reduction.
2. Detection of edge points. Local operation that detects potential edge points.
3. Edge localization. Select true members of an edge.

- First order derivative (gradient):

$$\vec{\nabla} f \equiv \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \begin{pmatrix} \partial f/\partial x \\ \partial f/\partial y \end{pmatrix}.$$

- The *magnitude*: rate of change in the direction of the gradient vector:

$$|\vec{\nabla} f| = M(x, y) = \sqrt{g_x^2 + g_y^2}.$$

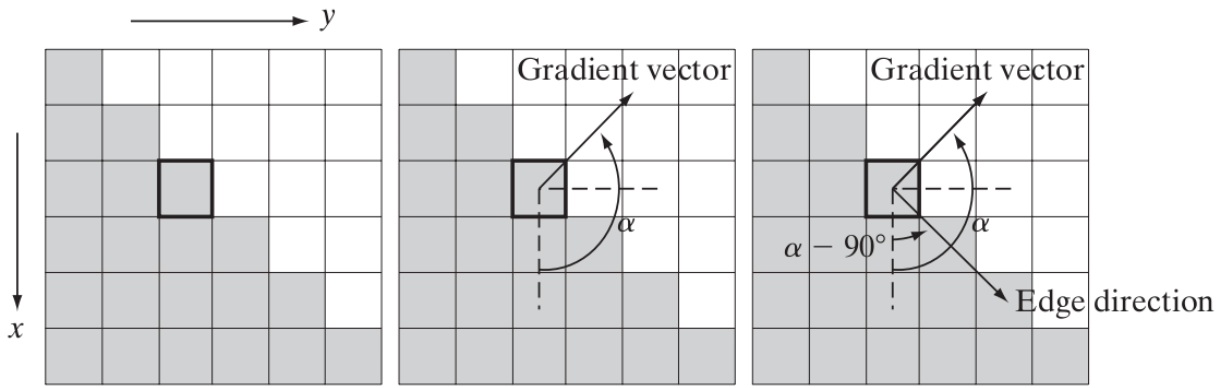
- The *direction* of the gradient is:

$$\alpha(x, y) = \tan^{-1}(g_x/g_y).$$

Note: g_x, g_y and M are images the size of f.

For the image below:

- $\nabla f = \begin{pmatrix} \partial f/\partial x \\ \partial f/\partial y \end{pmatrix} = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$.
- $M(x, y) = 2\sqrt{2}$.
- $\alpha = \tan^{-1}(g_y/g_x) = -45°$.



a b c

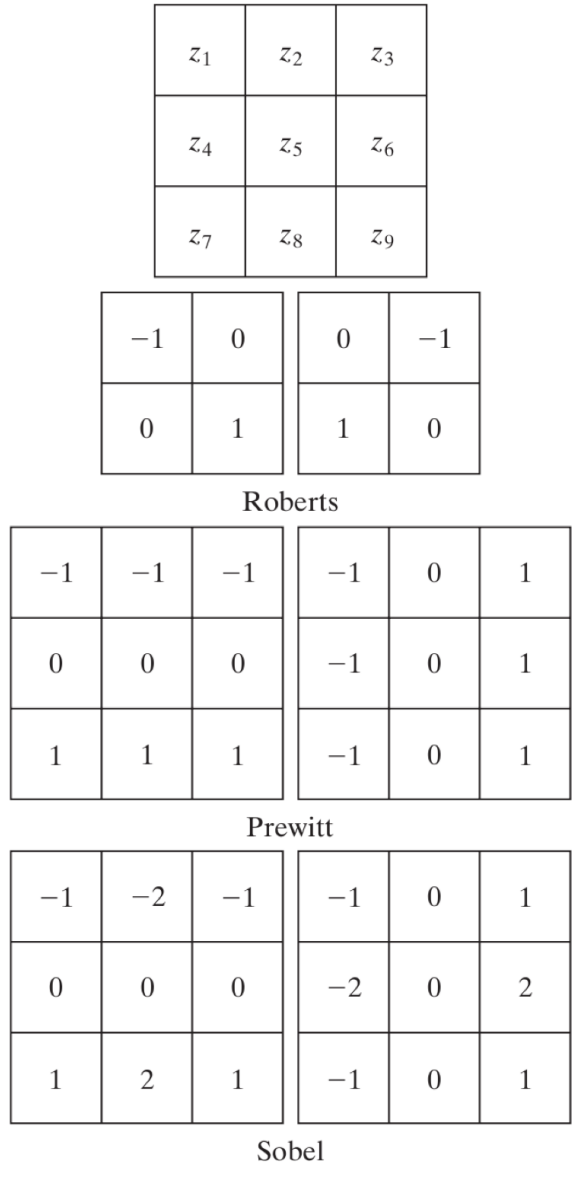
FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.

Digital approximation of first order partial derivatives:

- $g_x = \dfrac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y).$
- $g_y = \dfrac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y).$

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.



Filter masks used to compute the derivatives needed for the gradient are often called *gradient operators, difference operators, edge operators*, or *edge detectors*.

Prewitt and Sobel masks for detecting diagonal edges:

a b
c d

FIGURE 10.15
Prewitt and Sobel masks for detecting diagonal edges.

0	1	1	−1	−1	0
−1	0	1	−1	0	1
−1	−1	0	0	1	1

Prewitt

0	1	2	−2	−1	0
−1	0	1	−1	0	1
−2	−1	0	0	1	2

Sobel

- The presented masks are used to compute g_x and g_y, which give edge strength

$$|\nabla f| = \sqrt{g_x^2 + g_y^2}$$

and direction

$$\alpha = \tan^{-1}(g_y/g_x).$$

- The magnitude can be approximated by

$$M(x, y) \approx |g_x| + |g_y|;$$

cheaper to compute but not isotropic[a].

[a] Invariant to rotation.

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, $|g_x| + |g_y|$.

The angle image ($\alpha = \tan^{-1}(g_y/g_x)$):

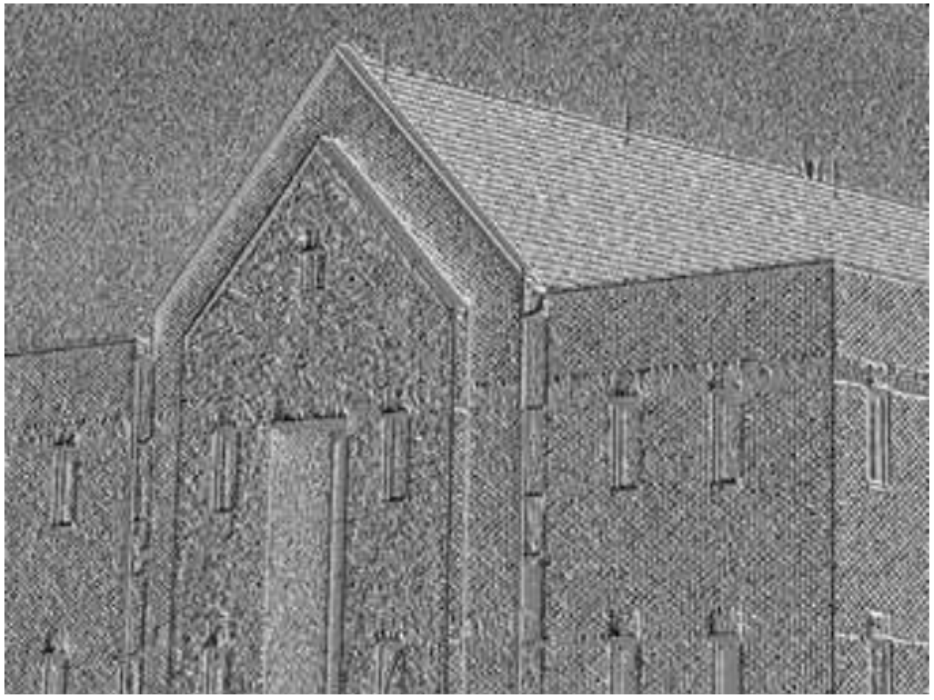
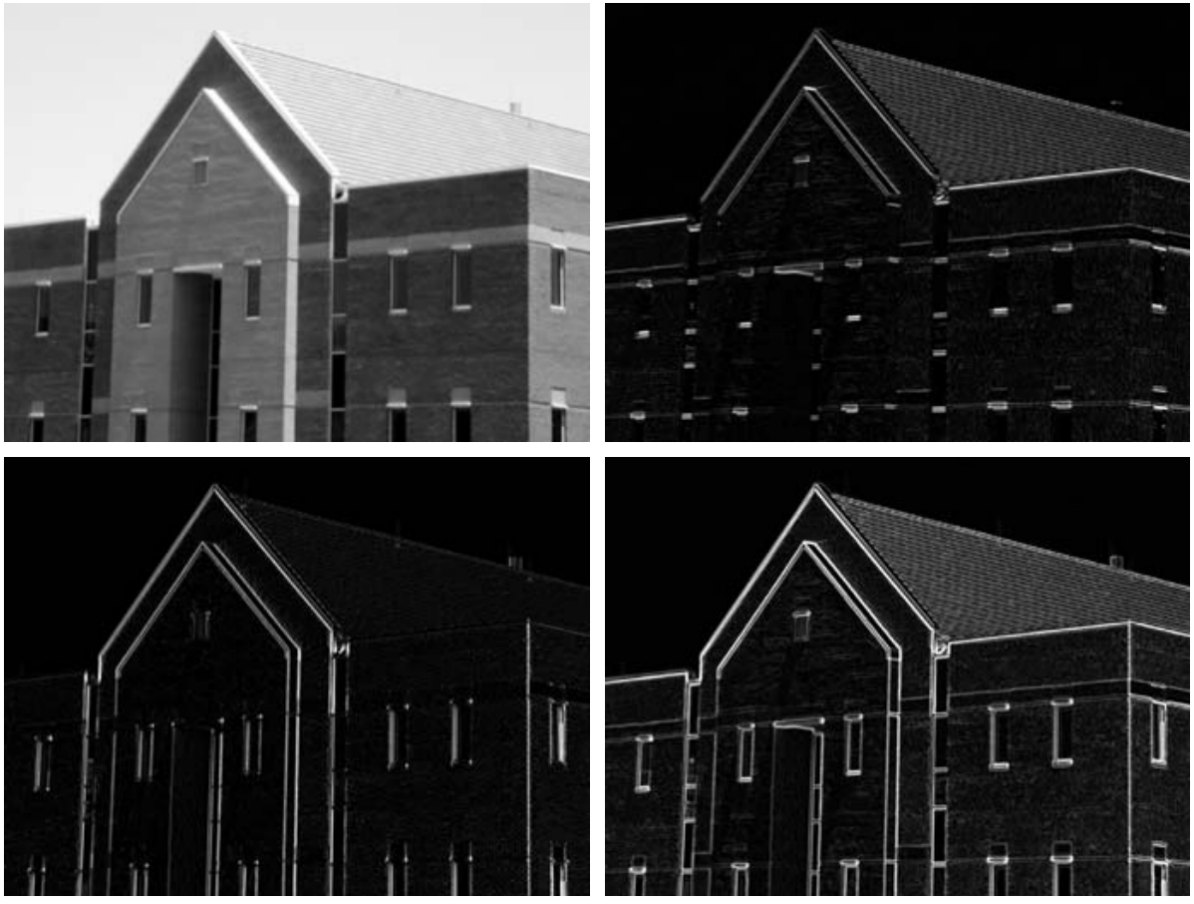


FIGURE 10.17 Gradient angle image computed using Eq. (10.2-11). Areas of constant intensity in this image indicate that the direction of the gradient vector is the same at all the pixel locations in those regions.

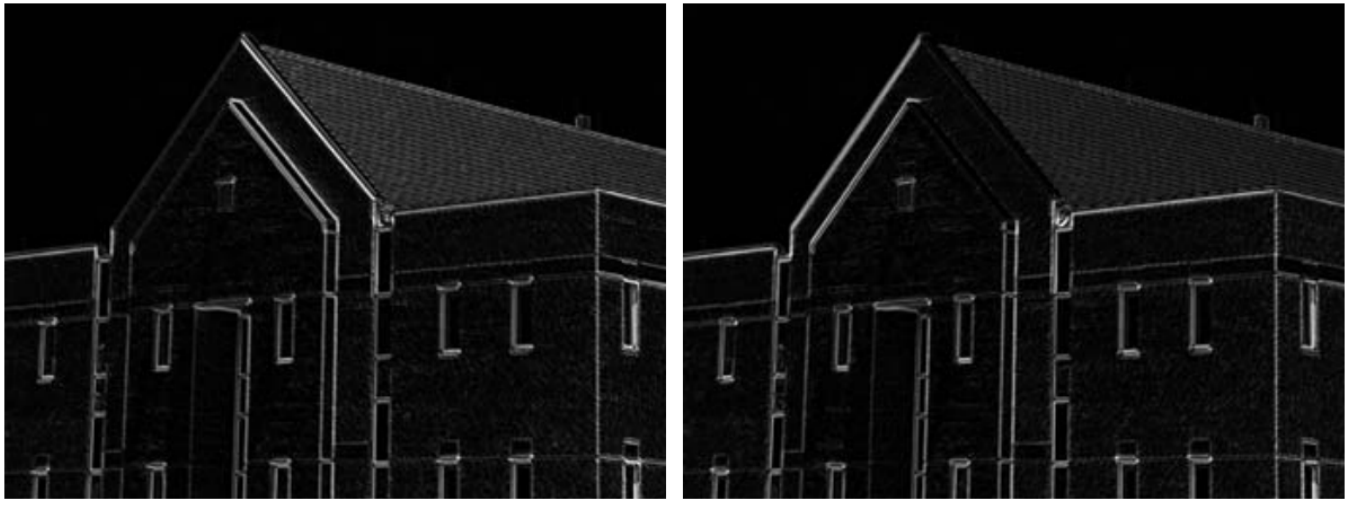
Smoothing the original image to reduce detection of weak edges:

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.



Using diagonal masks:



a b

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

Thresholding the original and smoothed filtered images:

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.

- We've seen edge detection through filtering regardless of edge characteristics and noise content.
- Let's learn more advanced techniques aimed at tackling noise and edge natures.

Marr and Hildreth argued that:

1. Intensity changes depend on image scale. Hence we must use operators of different sizes.
2. Sudden intensity change will give rise to a peak in the 1st derivative (or, equivalently, to a zero-crossing in the 2nd derivative).

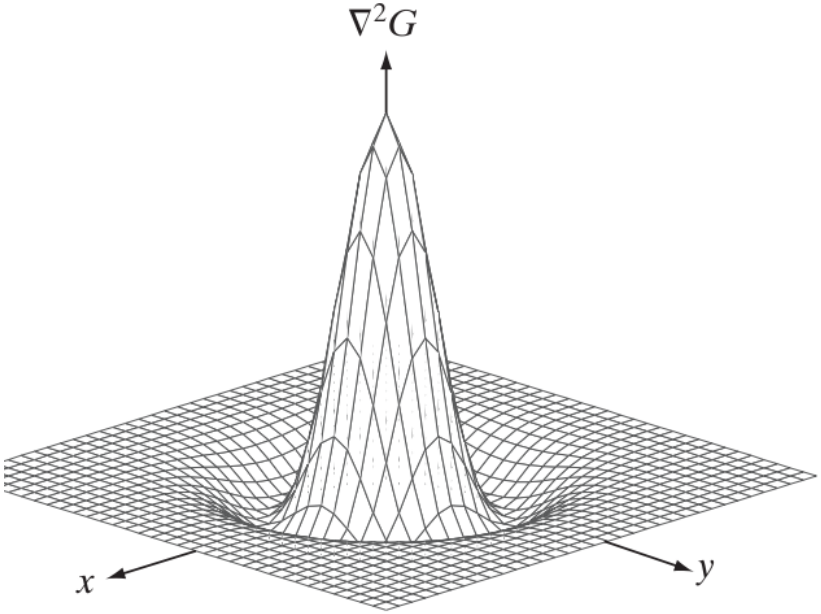
Hence, an operator should;

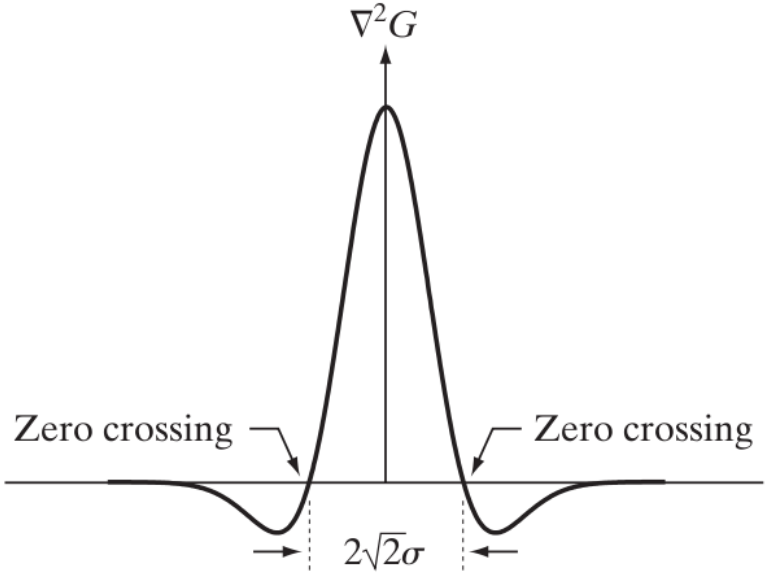
1. be a differential operator for computing the first or second derivative.
2. be adjustable for different scales (blurred and sharp edges).

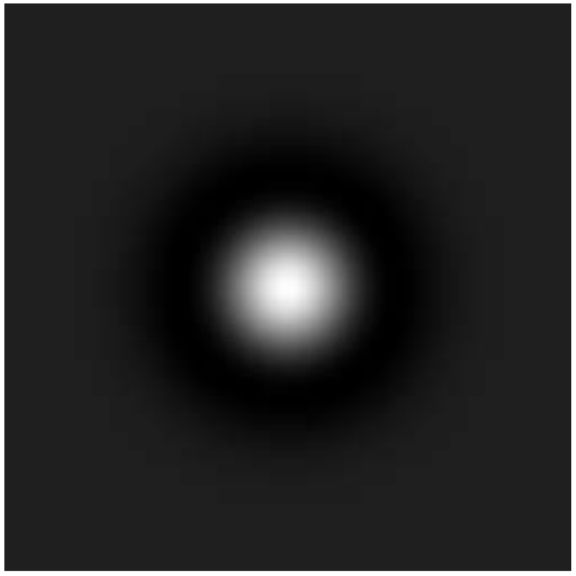
They argued that the best operator is the Laplacian-of-Gaussian $\nabla^2 G(x, y)$, where

$$G(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}}.$$

$$\nabla^2 G(x, y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{x^2 + y^2}{2\sigma^2}}$$







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 10.21 (a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5 × 5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

Two fundamental ideas:

1. Reduce details much smaller than σ by blurring the image with a Gaussian.
2. Detect abrupt changes through the Laplacian (which is isotropic).

The algorithm relies on the Laplacian-of-Gaussian (LoG) operator:
$g(x,y) = \left[\nabla^2 G(x,y)\right] \star f(x,y) = \nabla^2 \left[G(x,y) \star f(x,y)\right]$

The Marr-Hildreth algorithm

1. Filter the input image $f(x,y)$ with an $n \times n$ Guassian lowpass filter[a].
2. Compute the Laplacian of the filtered image.
3. Find zero-crossings of the result.

[a] Choose n as the smallest integer greater than 6σ (since 99.7% of the Guassian volume lies within $\pm 3\sigma$)

a b
c d

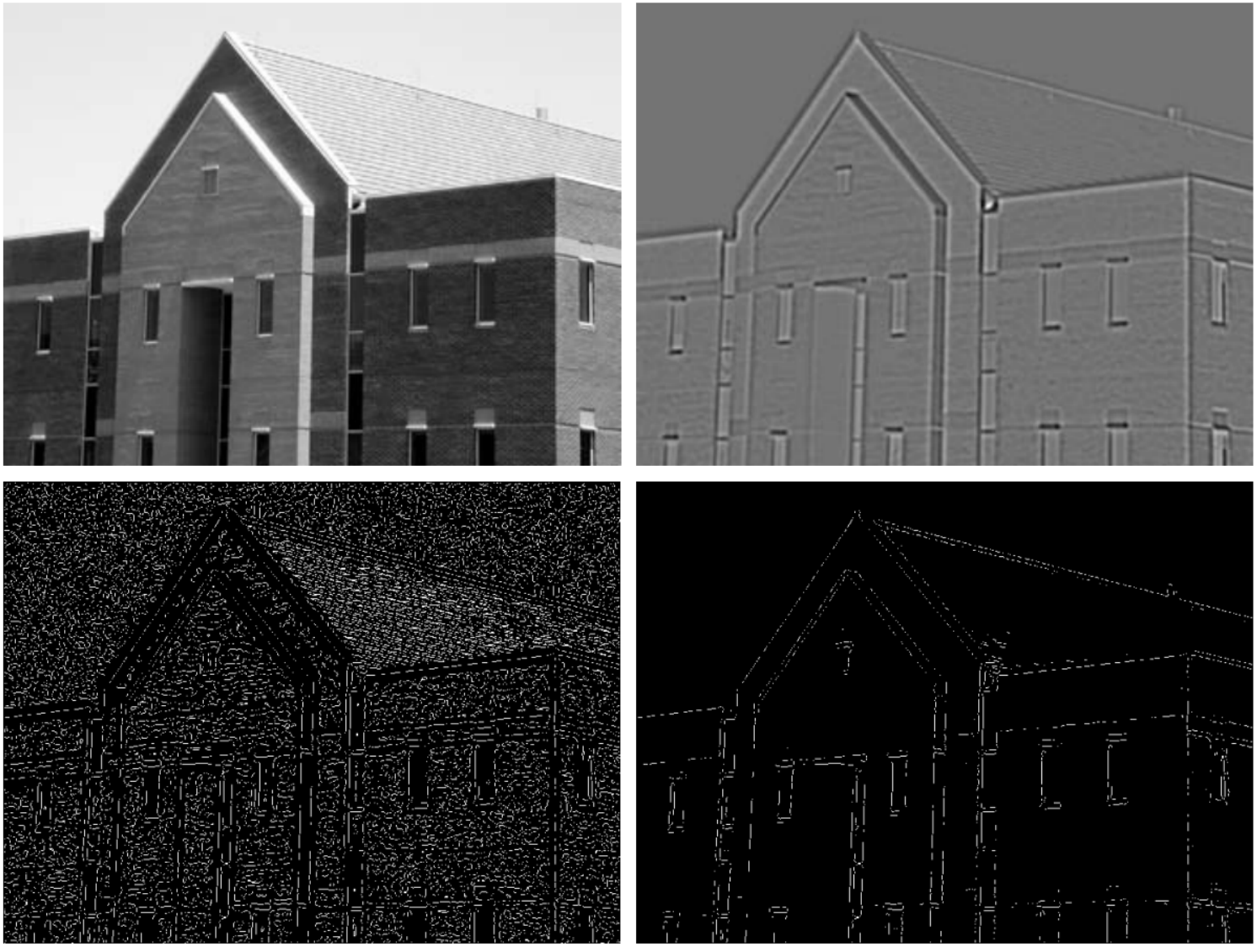


FIGURE 10.22
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

- The Canny edge detector is more complex but better.
- Three main objectives:
  1. Low error rate:
     - All edges should be found (low false negative).
     - There should be no spurious responses (low false positive).
  2. Edge points should be well localized.
  3. Single edge point response (minimum number of local maxima around the true edge).

Overview of the Canny edge detector:

1. Smooth the input image with a Gaussian filter
2. Compute the gradient magnitude and angle images.
3. Apply non-maxima suppression to the gradient magnitude image.
4. Use double thresholding and connectivity analysis to detect and link edges.

To meet these three criteria, Canny used;

1. numerical optimization;
2. 1-D step edges;
3. corruption by additive white Gaussian noise;

to find the optimal edge detector:

$$\frac{d}{dx}e^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}}.$$

Generalizing to 2-D:

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

$$f_s = G(x,y) \star f(x,y).$$

Then we compute the gradient magnitude and direction:

$$M(x,y) = \sqrt{g_x^2 + g_y^2},$$

$$\alpha(x,y) = \tan^{-1}(g_y/g_x).$$

where $g_x = \partial f/\partial x$ and $g_y = \partial f/\partial y$.

- Next step: thin wide ridges resulting from the gradient operation (first derivatives are constant throughout edge limits).
- Use *nonmaxima suppression*.

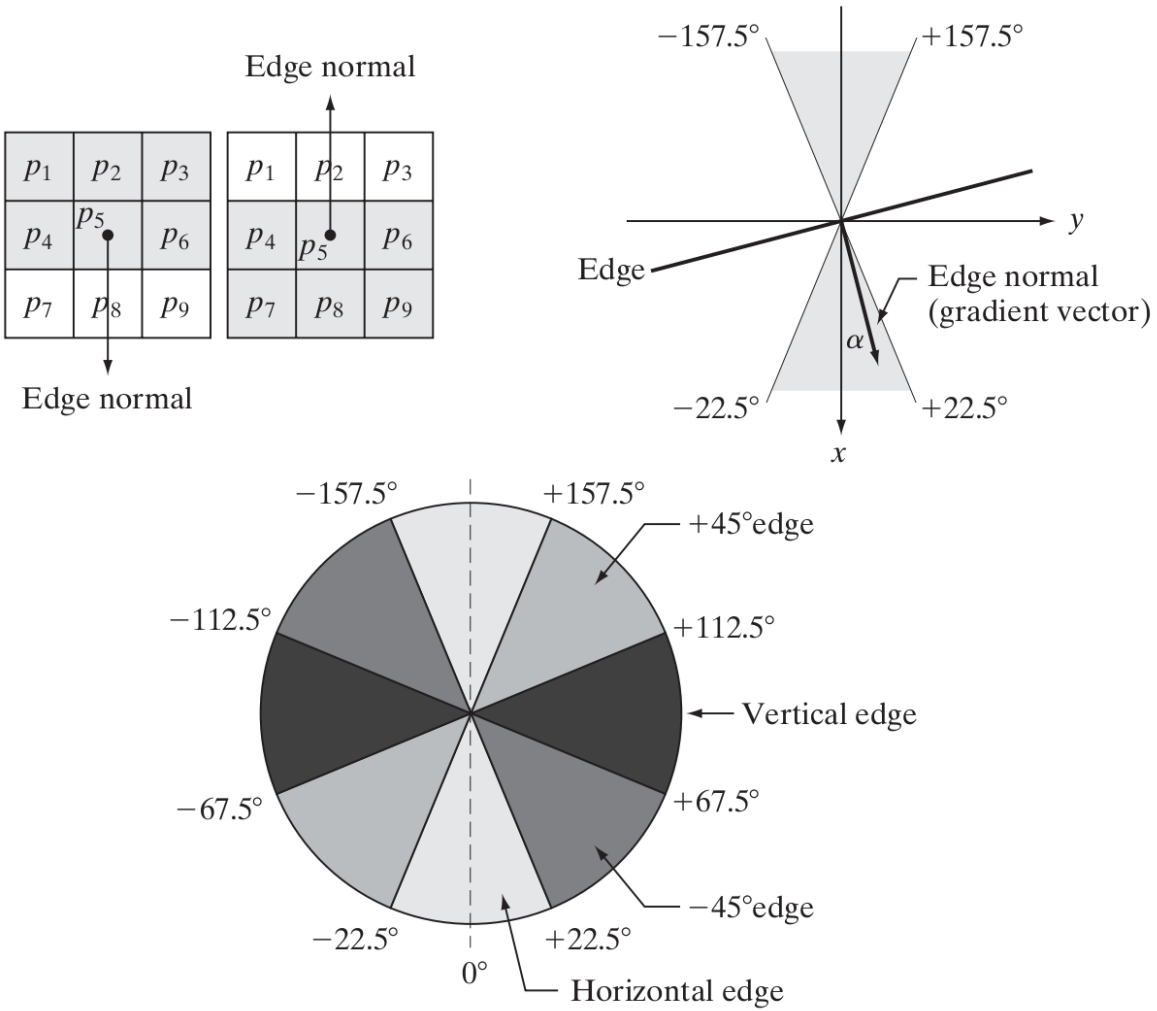
Nonmaxima suppression; analyze neighborhood around each point of $\alpha(x, y)$:

1. Find direction d_k closest to $\alpha(x, y)$.
2. If $M(x, y)$ is less than at least one of its two neighbors along d_k, let $g_N(x, y) = 0$ (suppression), otherwise, $g_N(x, y) = M(x, y)$.

Image g_N:

- Is the non-maxima suppressed image.
- Contains only the thinned edges; it is equal to $M(x, y)$ with the non-maxima edge points suppressed.

Nonmaxima suppression; analyze neighborhood around each point of $\alpha(x, y)$:



a b
c

FIGURE 10.24
(a) Two possible orientations of a horizontal edge (in gray) in a 3×3 neighborhood. (b) Range of values (in gray) of α, the direction angle of the *edge normal*, for a horizontal edge. (c) The angle ranges of the edge normals for the four types of edge directions in a 3×3 neighborhood. Each edge direction has two ranges, shown in corresponding shades of gray.

Next step: thresholding.
Find:

- "Strong" edges using $g_{NH}(x, y) = g_N(x, y) \geq T_H$.
- "Weak" edges using $g_{NL}(x, y) = g_N(x, y) \geq T_L$.

Then:

- Pixels in g_{NH} (typically unconnected) are marked as edge pixels.

In the end, append to $g_{NH}(x, y)$ all nonzero pixels from g_{NL}.

1. Take $p \in g_{NH}(x, y)$.
2. Mark as valid edge pixels all the weak pixels in g_{NL} connected to p (e.g., 8 connectivity).
3. If all nonzero pixels in $g_{NH}(x, y)$ were analyzed go to 4. Else, return to 1.
4. Set to zero all pixels in $g_{NL}(x, y)$ that were not marked as valid edge pixels.

a b
c d

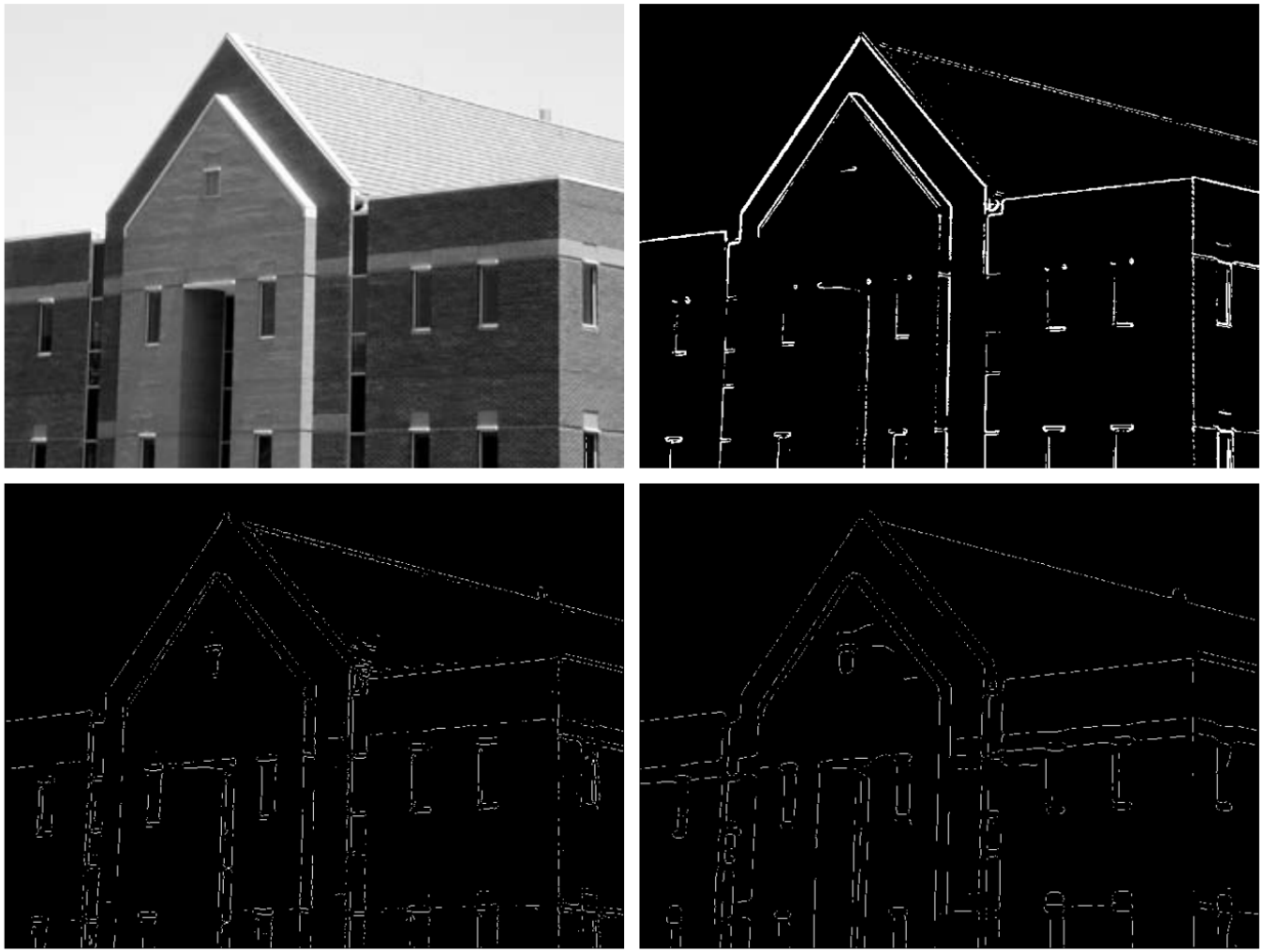
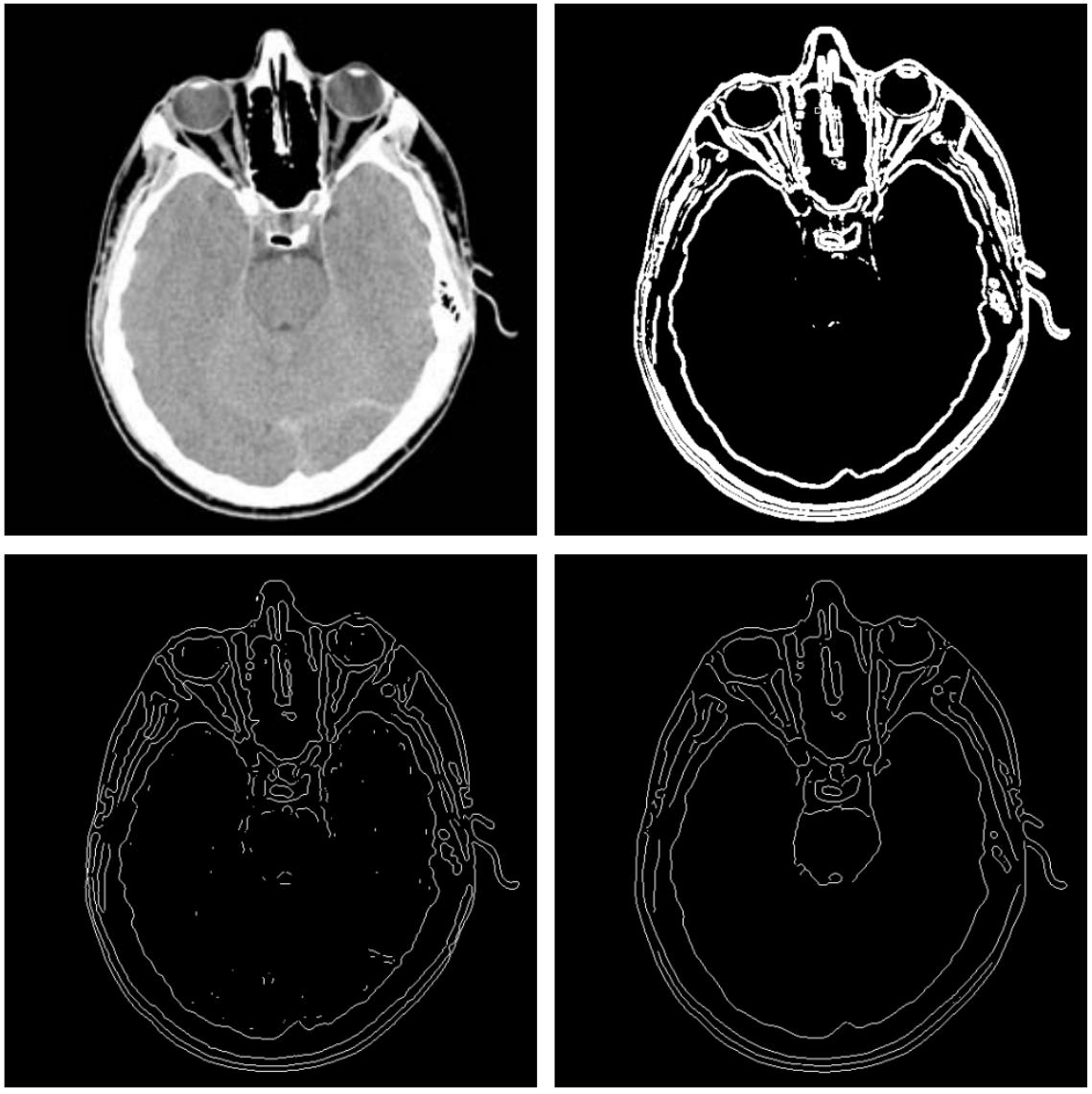


FIGURE 10.25
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].
(b) Thresholded gradient of smoothed image.
(c) Image obtained using the Marr-Hildreth algorithm.
(d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.



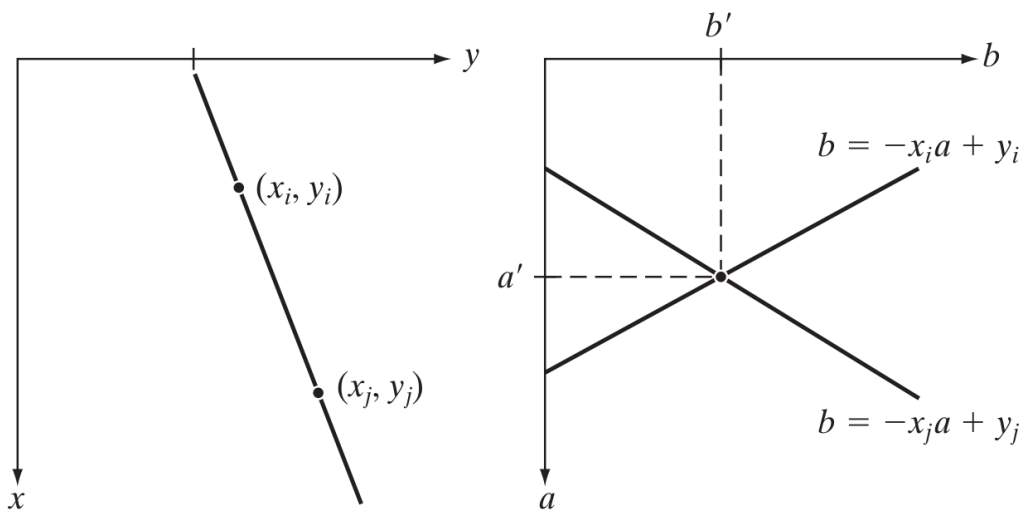
a b
c d

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.)

The Hough transform:

- Developed by Paul Hough in 1962.
- Can be applied to detect curves that:
  - can be described as a mathematical expression, and;
  - have no constrain in their location.
- Advantage:
  - Relatively insensitive to gaps and noise.

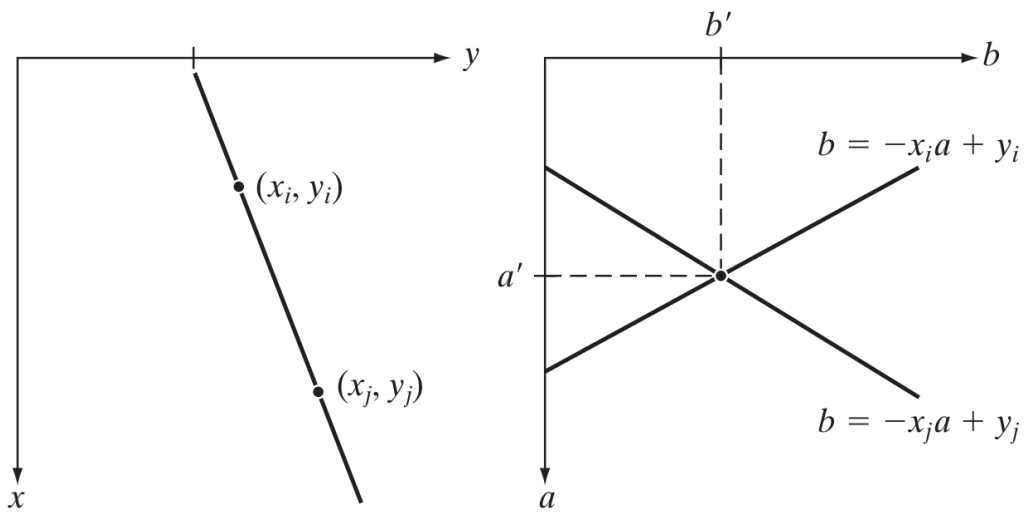
- Consider point $p_i = (x_i, y_i)$ in plane S_{xy}.
- Infinitely many lines pass through p_i for varying values of a and b satisfying $y_i = ax_i + b$.
- In plane S_{ab}, point p_i is transformed into line $b = -x_i a + y_i$.
- Equivalently, point p_j is transformed into line $b = -x_j a + y_j$.



a b

FIGURE 10.31
(a) xy-plane. (b) Parameter space.

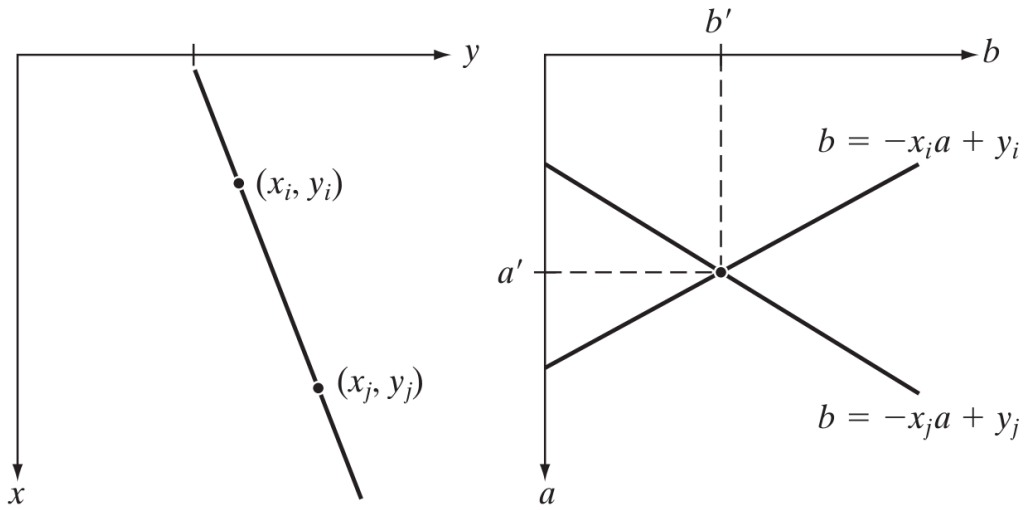
- A line in S_{xy} is transformed into a point in S_{ab} and vice-versa.
- Points p_i and p_j lie on the line given by equation $y = a'x + b'$.
- All points lying on this line on S_{xy} are transformed into lines in S_{ab} that intersect at point (a', b').
- The higher the number of points, the more lines intersect.



a b

FIGURE 10.31 (a) xy-plane. (b) Parameter space.

- The number of lines intersecting at (a', b') in S_{ab} denote the number of points in S_{xy} belonging to line $y = a'x + b'$.
- We can find dense lines in S_{xy} by finding points in S_{ab} where several lines intersect.



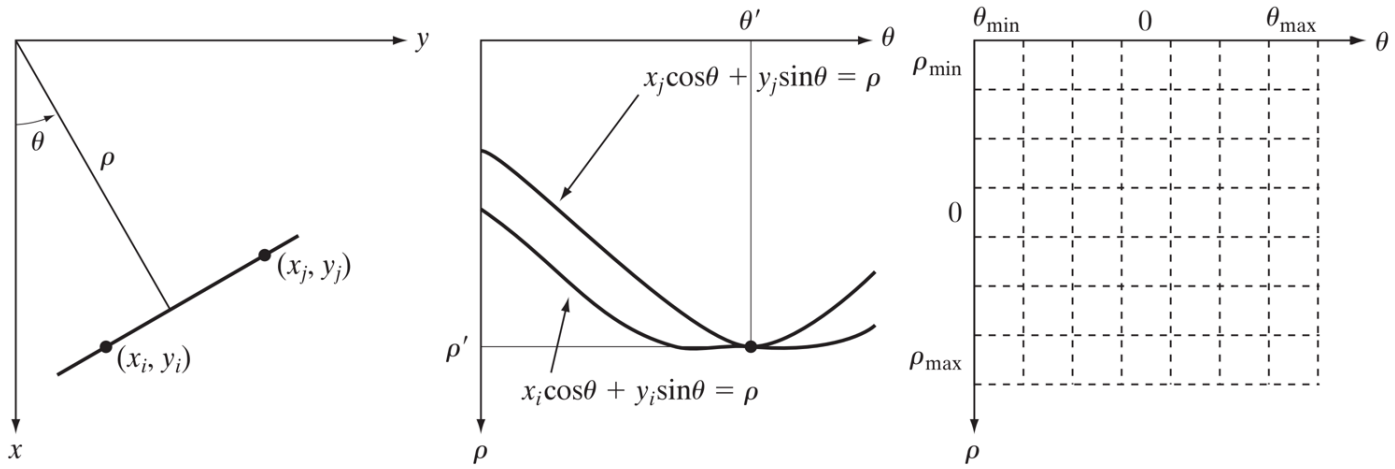
a b

FIGURE 10.31
(a) xy-plane. (b) Parameter space.

- One problem! With vertical lines: $b \mapsto \infty$.
- The solution is to use:

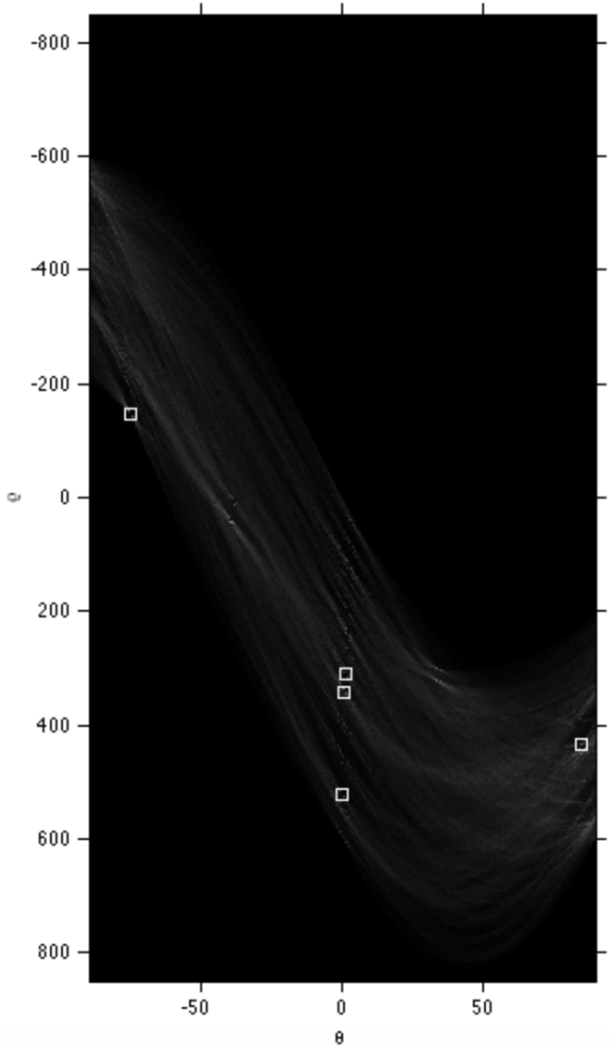
$$x\cos\theta + y\sin\theta = \rho. \tag{4}$$

- In this case, points in S_{xy} map to sinusoids in S_{ab}.



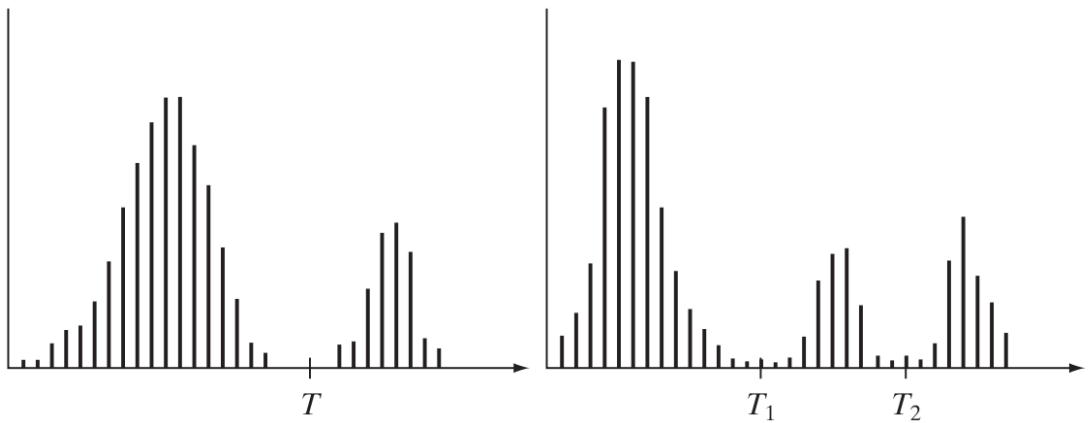
a b c

FIGURE 10.32 (a) (ρ, θ) parameterization of line in the xy-plane. (b) Sinusoidal curves in the $\rho\theta$-plane; the point of intersection (ρ', θ') corresponds to the line passing through points (x_i, y_i) and (x_j, y_j) in the xy-plane. (c) Division of the $\rho\theta$-plane into accumulator cells.


-800
-600
-400
-200
0
200
400
600
800
ρ
-50
0
50
θ

- A grayscale image $f(x, y)$ can be divided into:
  - Background.
  - Foreground.
- Threshold (T) is a grayscale level capable of performing the separation:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases} .$$

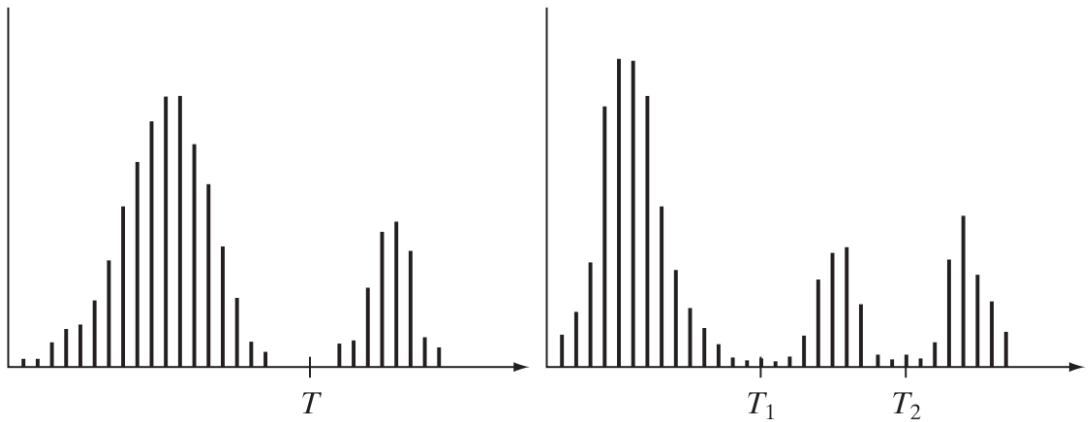


a b

FIGURE 10.35 Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.

- There can be various threshold levels.
- T can be fixed (*global thresholding*) or not (variable thresholding).
- T may also depend on the neighborhood of the pixel (adaptive thresholding).

$$g(x,y) = \begin{cases} a & \text{if } f(x,y) > T_2 \\ b & \text{if } T_1 < f(x,y) \leq T_2 \\ c & \text{if } f(x,y) \leq T_1 \end{cases} .$$

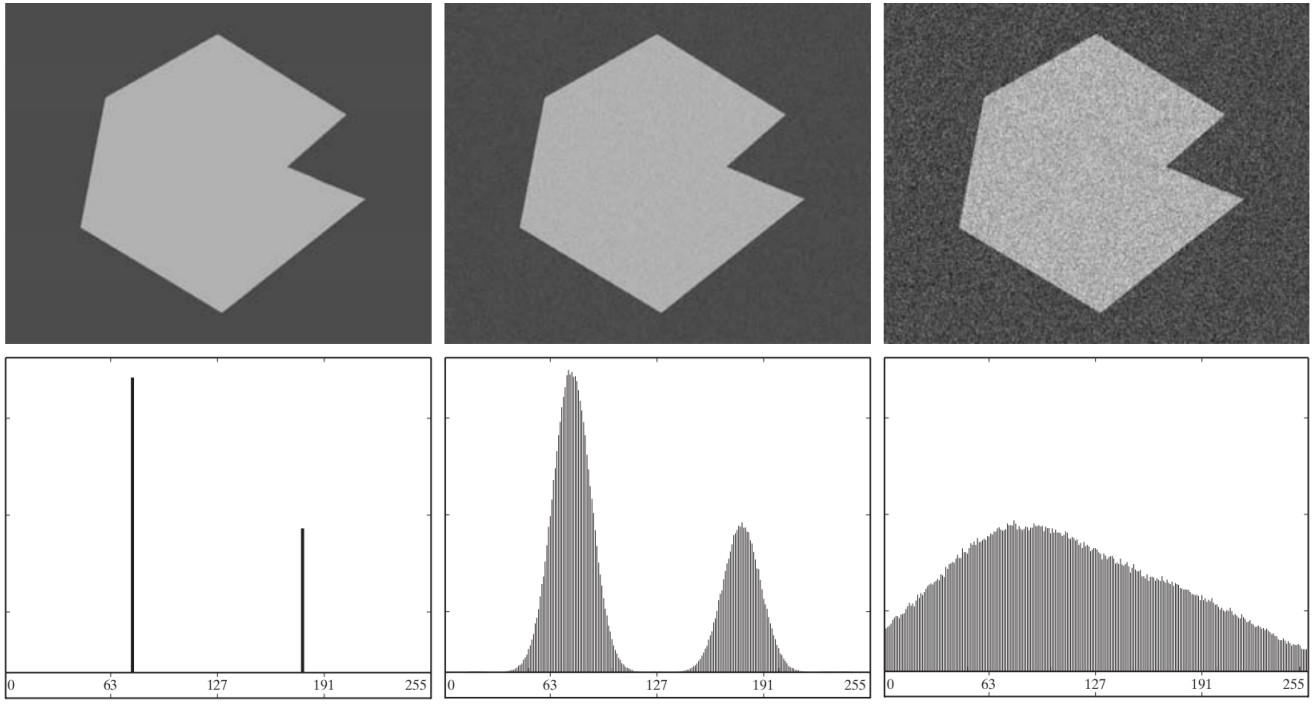


a b

FIGURE 10.35 Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.

- Intensity thresholding depends on width and depth of the valley(s) separating the histogram modes.
- Key factors affecting the properties of the valley(s):
  1. Separation between peaks.
  2. Noise content in the image.
  3. Relative sizes of objects and background.
  4. Uniformity of the illumination source.
  5. Uniformity of the reflectance properties of the image.

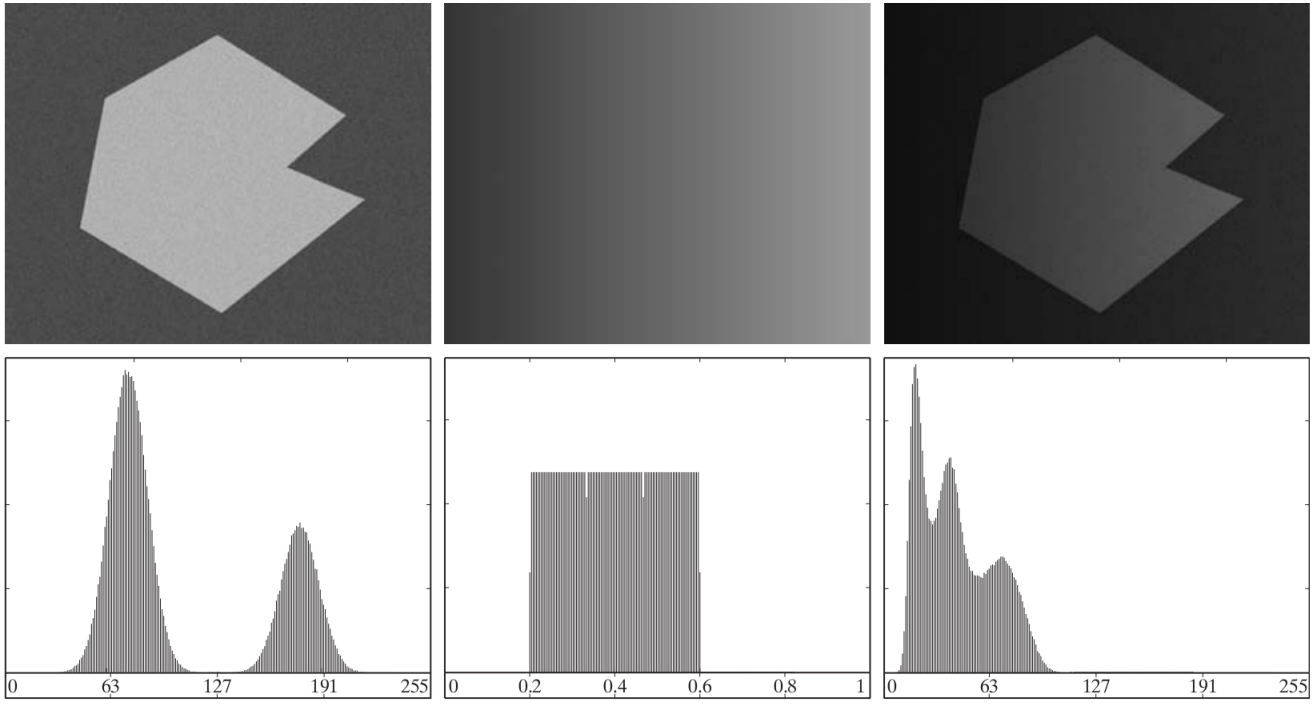
Problem 1: noise.



a b c
d e f

FIGURE 10.36 (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d)–(f) Corresponding histograms.

Effect of non-uniform illumination:



a b c
d e f

FIGURE 10.37 (a) Noisy image. (b) Intensity ramp in the range [0.2, 0.6]. (c) Product of (a) and (b). (d)–(f) Corresponding histograms.

Thresholding

Illumination and reflectance play a central role in the success of image segmentation using thresholding or other segmentation techniques.

Attention

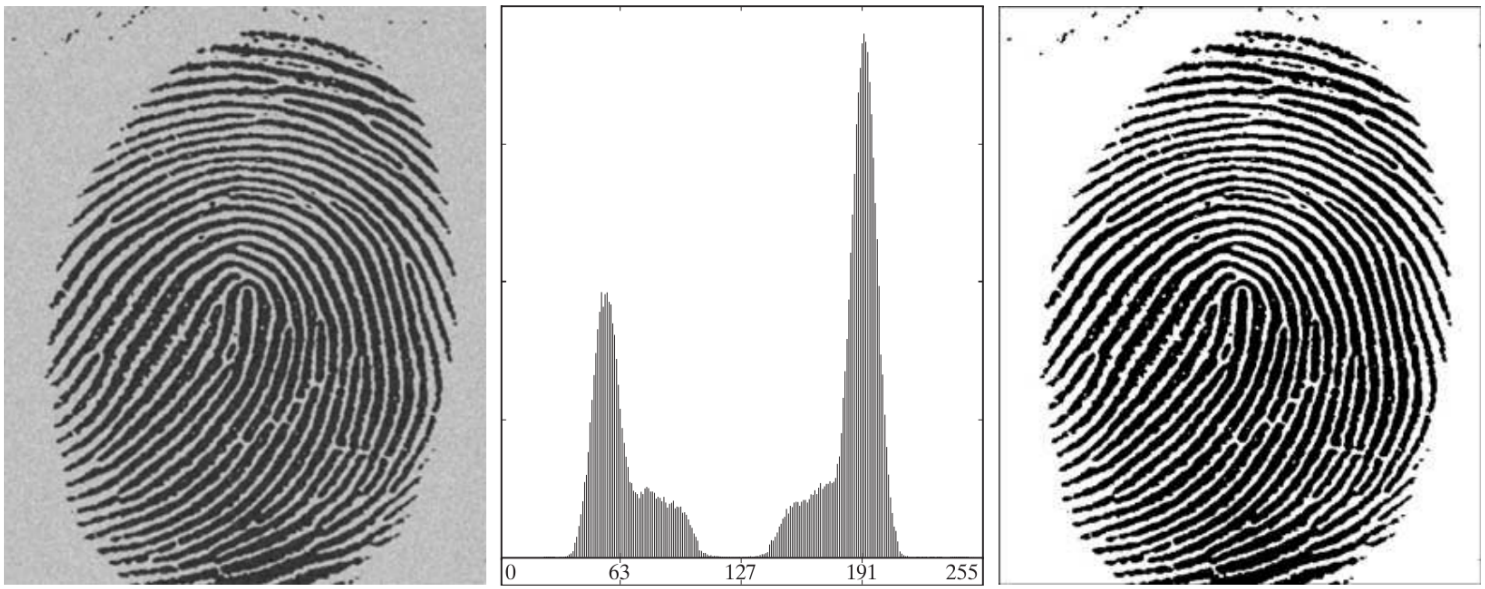
If possible, control these problems when facing a segmentation challenge!!!

- Example: industrial applications.

Three basic approaches to fix non-uniform illumination and/or object reflectance:

1. Find the illumination pattern by imaging a flat surface, then multiply its inverse by an input image.
2. Use the top-hat (or bot-hat) transform.
3. Use variable thresholding.

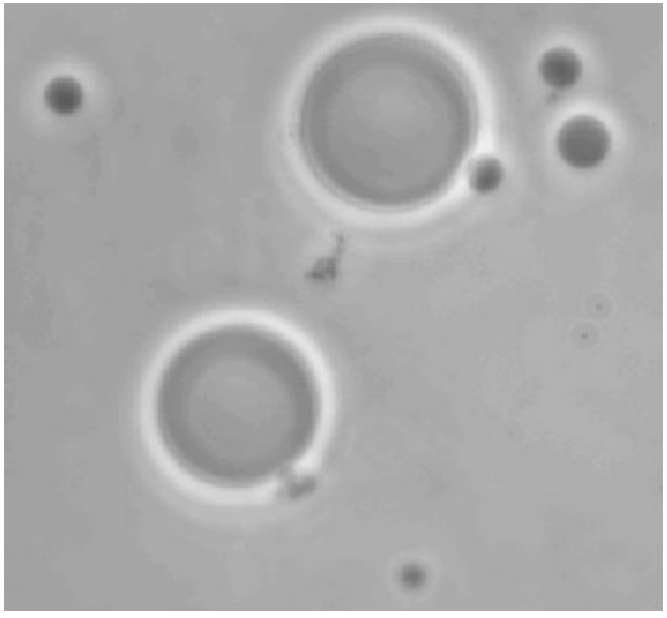
1. Select an initial, T, producing G_1 and G_2.
2. Compute means m_1 and m_2.
3. Update threshold $T = (m_1 + m_2)/2$.
4. Repeat 2-4 until $T_{i+1} - T_i < \delta$.

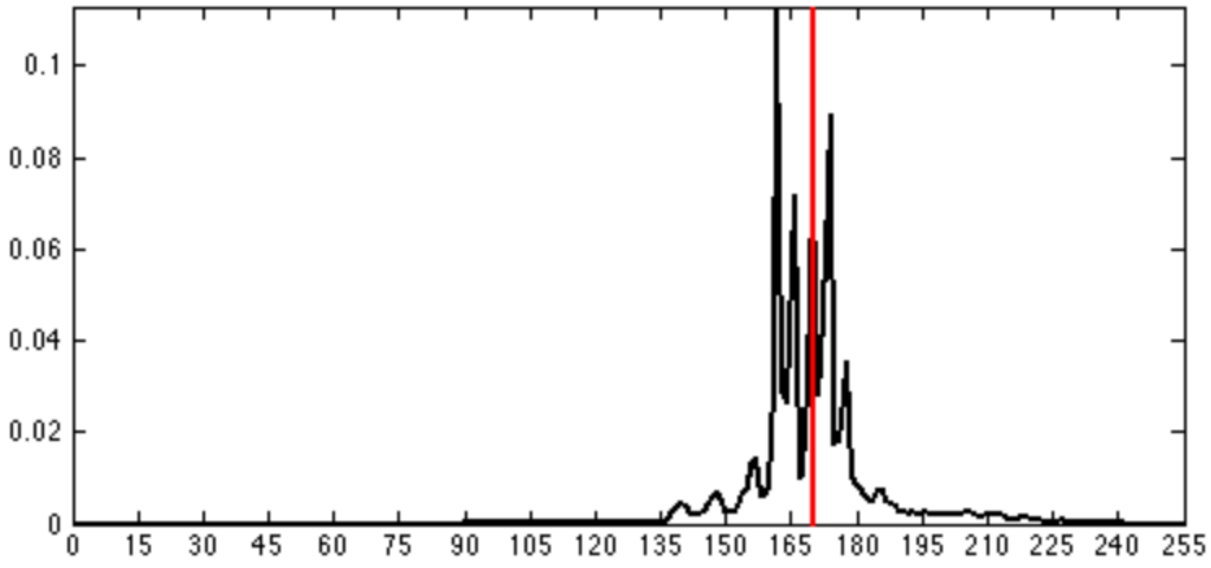


a b c

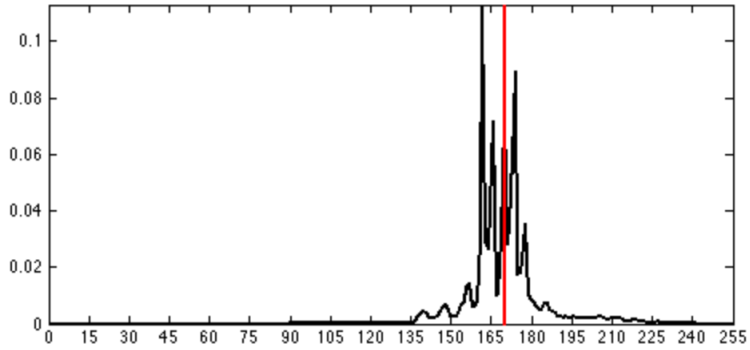
FIGURE 10.38 (a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (the border was added for clarity). (Original courtesy of the National Institute of Standards and Technology.)

Suppose we have an image with N pixels, L intensity levels and the following histogram.





- n_i is the number of pixels with intensity i.
- The normalized histogram is such that $\sum_{i=0}^{L-1} p_i = 1$, where $p_i = n_i/N$.



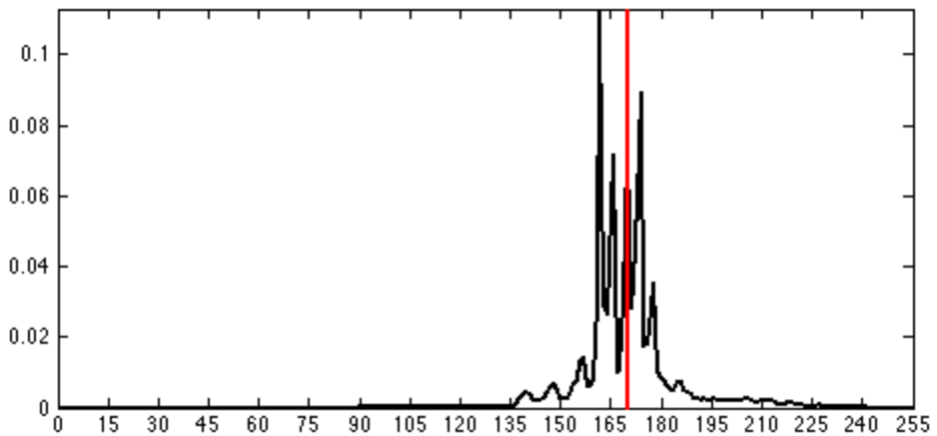
- Segment using a threshold T given by basic global thresholding.
- The probabilities of labeling a pixel to class C_1 and C_2 are

$$P_1(T) = \sum_{i=0}^{T} p_i$$

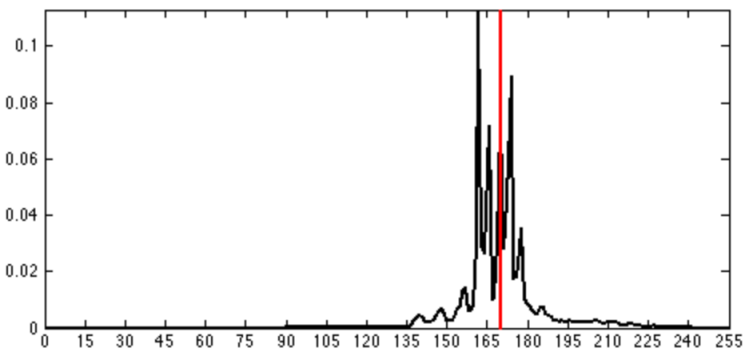
and

$$P_2(T) = \sum_{i=T+1}^{L-1} p_i = 1 - P_1(T).$$

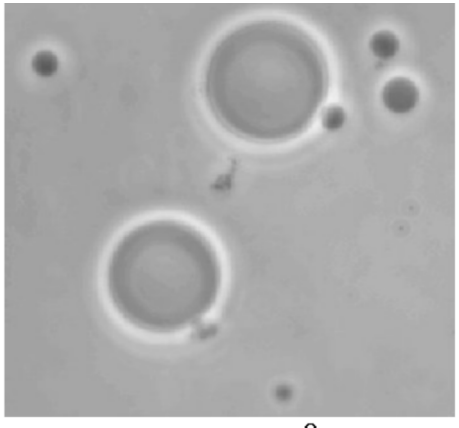
- The mean intensity value of the pixels assigned to C_1 is $m_1(T) = \frac{1}{P_1(T)} \sum_{i=0}^{T} ip_i$.
- Equivalently... $m_2(T) = \frac{1}{P_2(T)} \sum_{i=T+1}^{L-1} ip_i$.

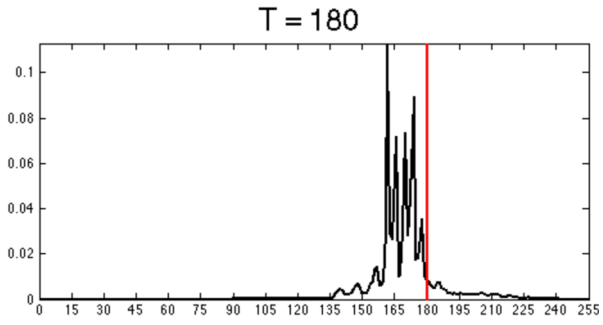


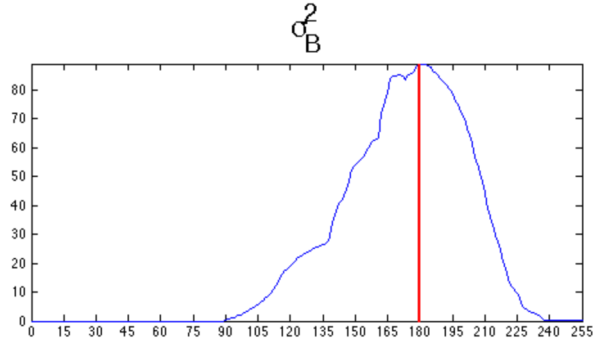
- The cumulative mean is $m(T) = \sum_{i=0}^{T} ip_i$.
- The global mean is $m_G(T) = \sum_{i=0}^{L-1} ip_i$.
- The global variance is given by $\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i$.
- The interclass variance is defined as $\sigma_B^2(T) = [P_1(T)m_1(T) - m_G]^2 + [P_2(T)m_2(T) - m_G]^2$.

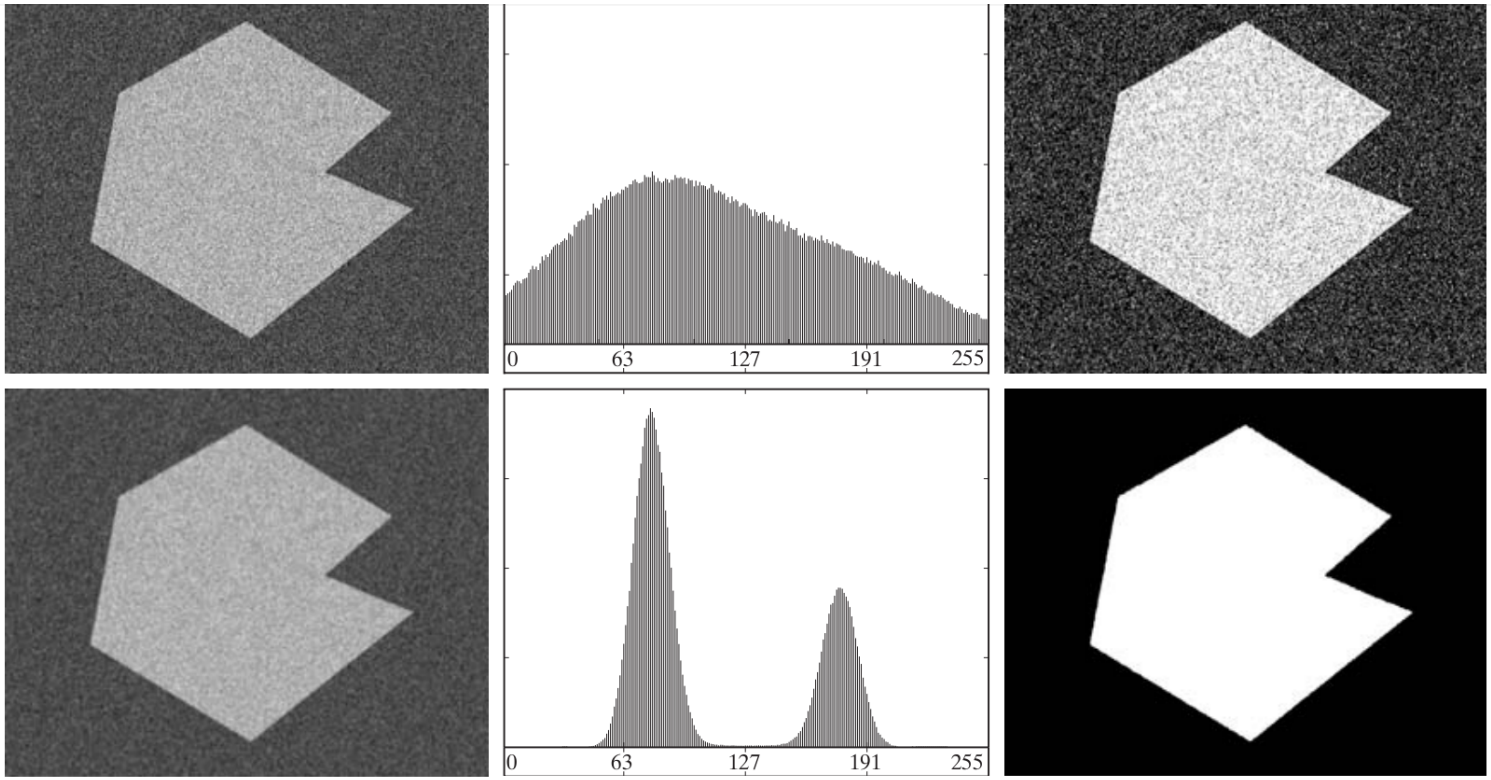


The Otsu's method aims at finding T that maximizes σ_B^2, i. e., best separates the classes:









a b c
d e f

FIGURE 10.40 (a) Noisy image from Fig. 10.36 and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method.

- A good, separable histogram shape has peaks that are;
  - tall;
  - narrow;
  - symmetric, and;
  - separated by deep valleys.
- One approach to improve histogram shapes is to consider pixels lying on or near edges:
  - less dependency on relative sizes of objects and background.
  - approximately equal probabilities of a pixels lying on object or background.
  - Tendency to deepen the valley between peaks.

Region-Based Segmentation

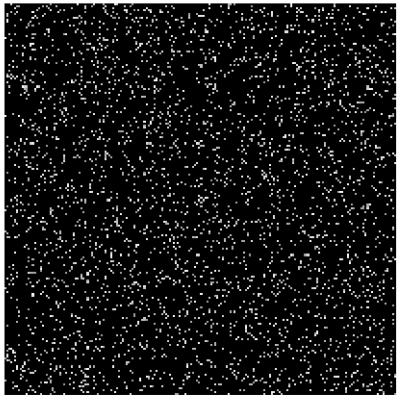
- Region growing is a procedure that groups pixels or sub-regions into larger regions based on predefined criteria for growth.
- The selection of similarity criteria depends not only on the problem but also on the type of image data available:
  - Analysis of satellite imagery depends heavily on the use of color. Otherwise it would be very difficult.
  - For monochrome images:
    - Use descriptors based on intensity levels (such as moments or texture).
    - spatial properties (such as connectivity).

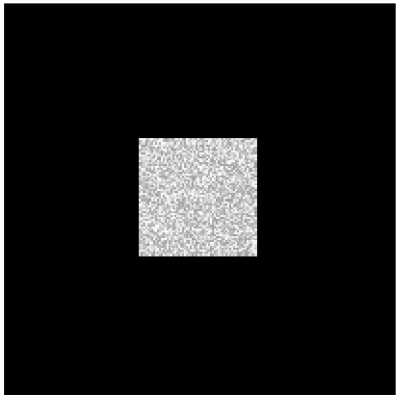
- Descriptors can yield misleading results if connectivity is not used in the region-growing process.

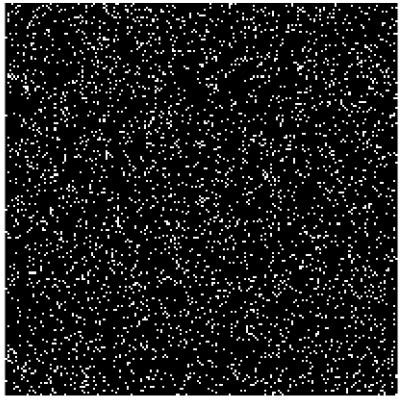
Attention

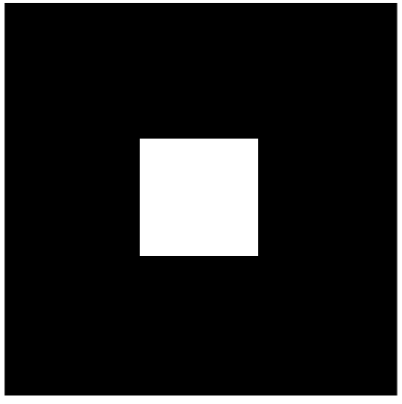
Criteria such as intensity values, texture, and color, are local in nature and do not take into account the "history" of region growth.

```
n = 3600;
sampleSize = 200; % choose an even number.
f = uint8(zeros(sampleSize));
f2 = f;
indexes = randi(numel(f), n, 1);
intensities = uint8(155 + randi(100, n, 1));
f(indexes) = intensities;
f2(sampleSize/2 - sqrt(n)/2:sampleSize/2 + sqrt(n)/2 - 1, ...
    sampleSize/2 - sqrt(n)/2:sampleSize/2 + sqrt(n)/2 - 1) = ...
    reshape(intensities, sqrt(n), sqrt(n));
subplot(2,2,1); imshow(f);
subplot(2,2,2); imshow(f2);
subplot(2,2,3); imshow(f > 150);
subplot(2,2,4); imshow(f2 > 150);
```









```
[g, NR, SI, TI] = regiongrow(f, S, T)
```

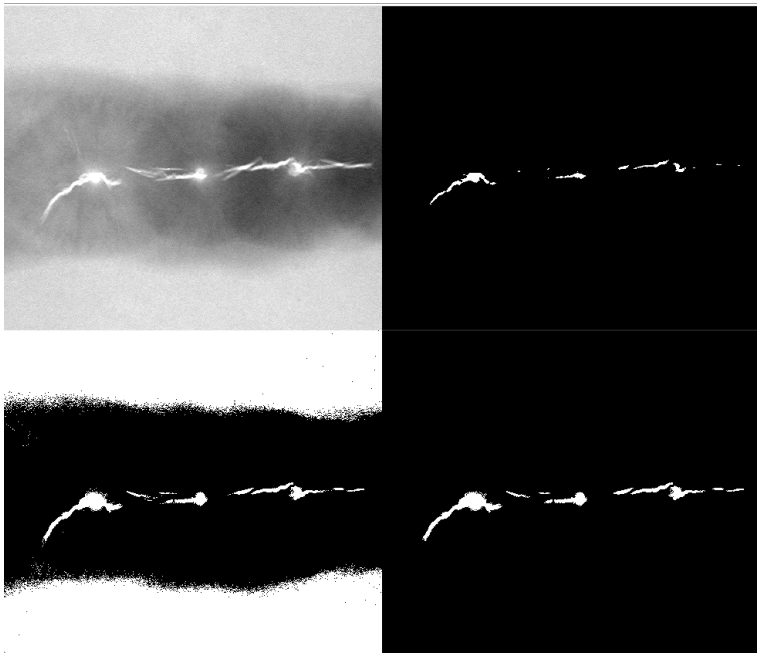
- `f` is the input image to be segmented.
- `S` can be:
  - An array (mask) the same size as `f`, containing the seed points[2].
  - A scalar, indicating the intensity value of the pixels to be considered as seeds.
- `T` can be:
  - An array (same size as `f`) containing the threshold value for each position in `f`.
  - A scalar defining a global threshold.

Attention

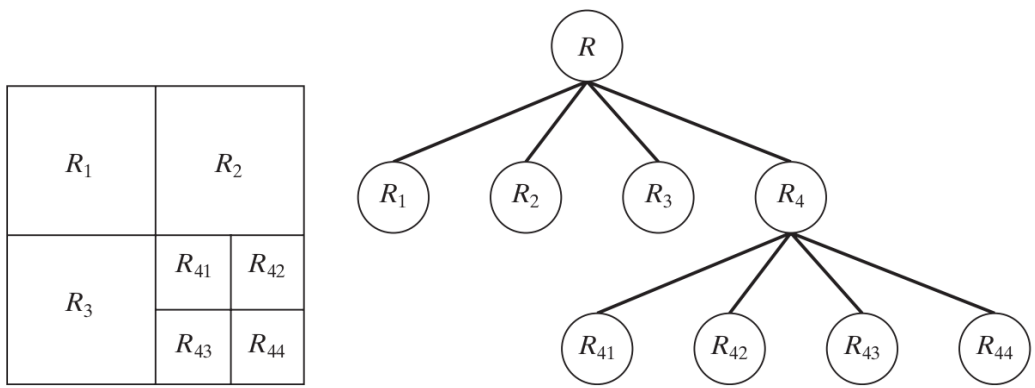
All values of `S` and `T` must be scaled to [0,1].

[2] Seeds can be found by inspection or other function.

```
%% EXAMPLE 11.13: Using region growing to detect weld porosity.
f = imread('Fig1014(a)(defective_weld).tif');
[g, NR, SI, TI] = regiongrow(tofloat(f), 1, 0.26);
```



1. Split into four disjoint quadrants any region R_i for which $Q(R_i) =$ FALSE.
2. When no further splitting is possible, merge any adjacent regions R_j and R_k for which $Q(R_j \cup R_k) =$ TRUE.
3. Stop when no further merging is possible.



a b

FIGURE 10.52 (a) Partitioned image. (b) Corresponding quadtree. R represents the entire image region.

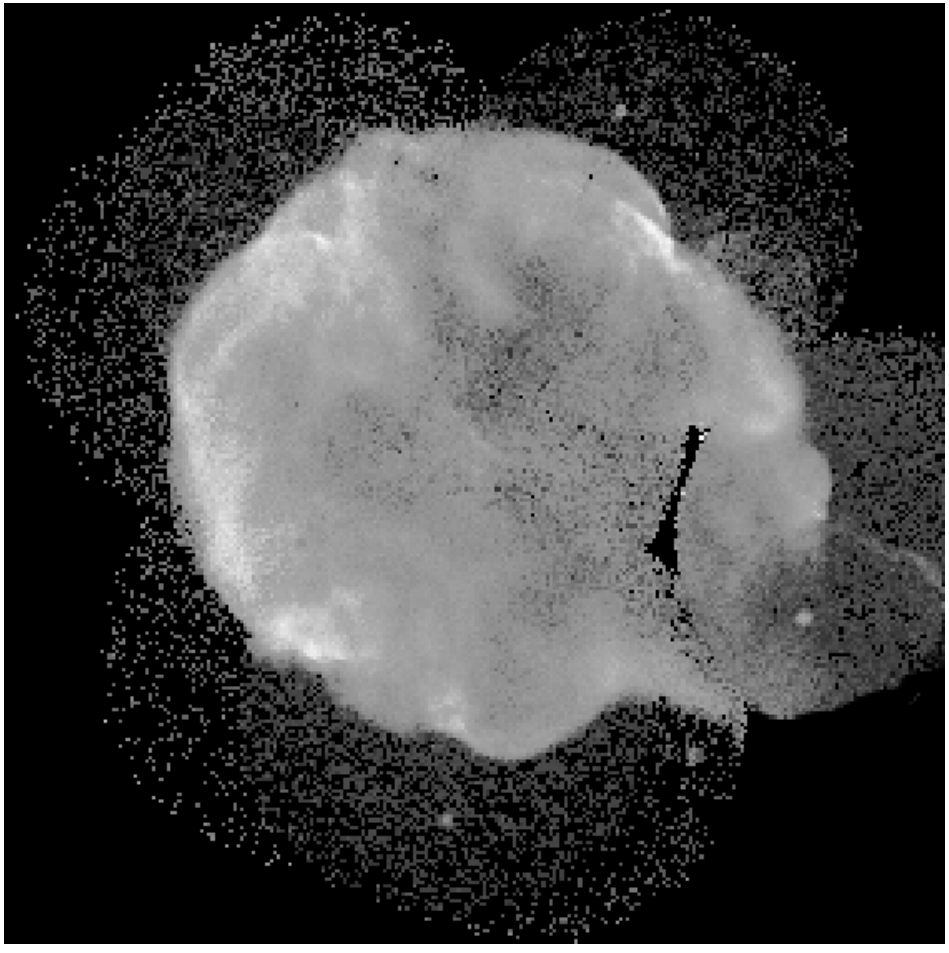
```
Z = qtdecomp(f, @split_test, parameters)
```

- `f` is the input image.
- `Z` is a sparse matrix containing the quadtree structure.
- If $Z(r, c)$ is nonzero, then;
  - (r, c) is the upper-left corner of a block in the decomposition, and;
  - $Z(r, c)$ is the size of the block.
- Function `split_test` defines the predicate, which determines if a region is to be split or not.
- `parameters` are any additional parameters required by the predicate.

To get the actual quadregion pixel values:

```
[vals, r, c] = qtgetblk(f, Z, m)
```

- X-ray band image of the Cygnus Loop:
- The objective is to segment out of the image the "ring" of less dense matter surrounding the dense center.



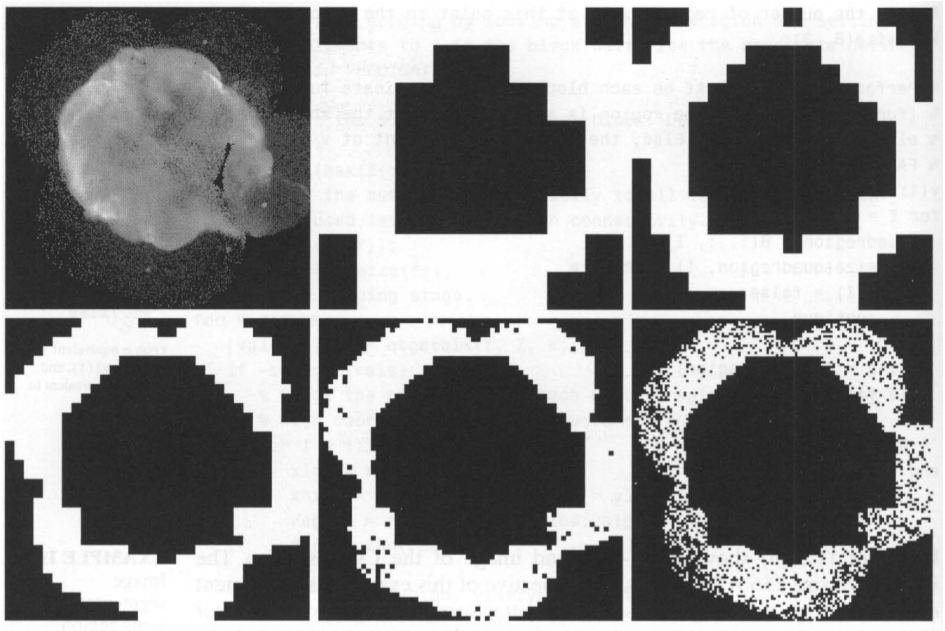
1. The data has a random nature to it.
2. The standard deviation σ of the wanted region is greater than the SD of the background ($\sigma = 0$) and lower than the brighter region.
3. So the predicate is determined by

$$Q = \begin{cases} \text{TRUE} & \text{if } \sigma > a \text{ AND } 0 < m < b \\ \text{FALSE} & \text{otherwise} \end{cases} .$$

```
% X-ray band image of the Cygnus Loop.
f = imread('Fig0938(a)(cygnusloop_Xray_original).tif');
mindim = 32; % 32, 16, 8, 4, 2
g = splitmerge(f, mindim, @predicate);
```

Where the predicate is defined in a different function

```
function flag = predicate(region)
sd = std2(region);
m = mean2(region);
flag = (sd > 10) & (m > 0) & (m < 125);
```



a b c
d e f

FIGURE 11.23 Image segmentation using a split-and-merge algorithm. (a) Original image. (b) through (f) Results of segmentation using function `splitmerge` with values of `mindim` equal to 32, 16, 8, 4, and 2, respectively. (Original image courtesy of NASA.)

Watershed Segmentation Using The Watershed Transform

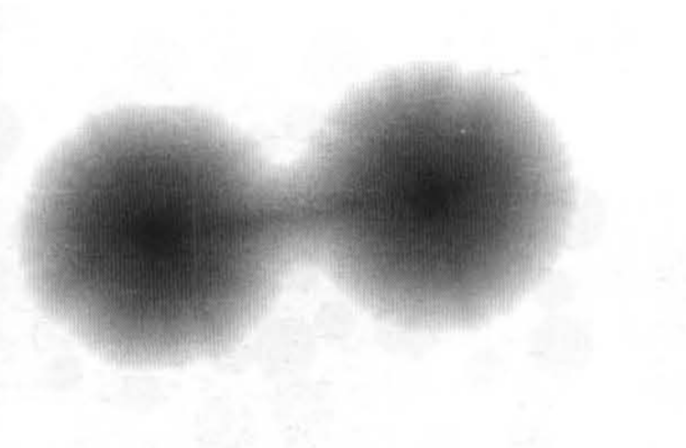
We have discussed segmentation based on three principal concepts:

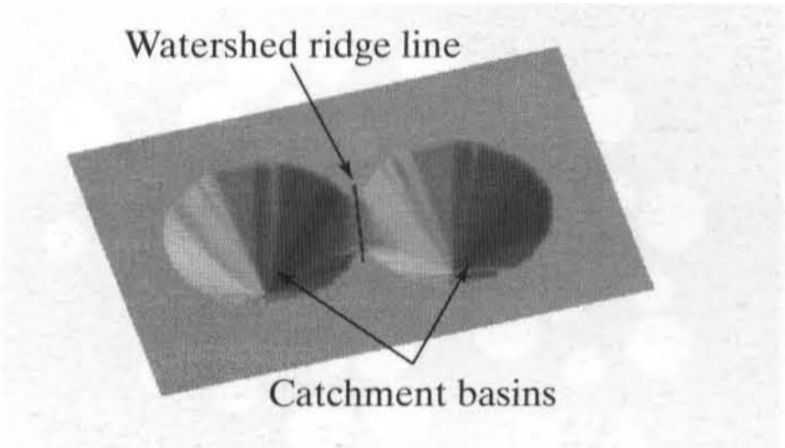
- Edge detection (speed).
- Thresholding (post-processing needed, such as edge linking).
- Region growing.

Let's discuss morphological watersheds. They embody concepts of the previous three approaches.

Visualize the image in three dimensions (topographic interpretation):

1. Points belonging to a regional minimum.
2. Points at which a drop of water, if placed at the location of any of those points, would fall with certainty to a single minimum.
3. Points at which water would be equally likely to more than one than such minimum.





a b

FIGURE 11.24 (a) Gray-scale scale image. (b) Image viewed as a surface, showing a watershed ridge line and catchment basins.

```
g = bwdist(f)
```

1	1	0	0	0	0.00	0.00	1.00	2.00	3.00
1	1	0	0	0	0.00	0.00	1.00	2.00	3.00
0	0	0	0	0	1.00	1.00	1.41	2.00	2.24
0	0	0	0	0	1.41	1.00	1.00	1.00	1.41
0	1	1	1	0	1.00	0.00	0.00	0.00	1.00

a b

FIGURE 11.25
(a) Binary image. (b) Distance transform.

```
g = imread('Fig1020(a)(binary-dowel-image).tif');
gc = ~g;
D = bwdist(gc);
L = watershed(-D);
w = L == 0;
g2 = g & ~w;
```

a b
c d
e

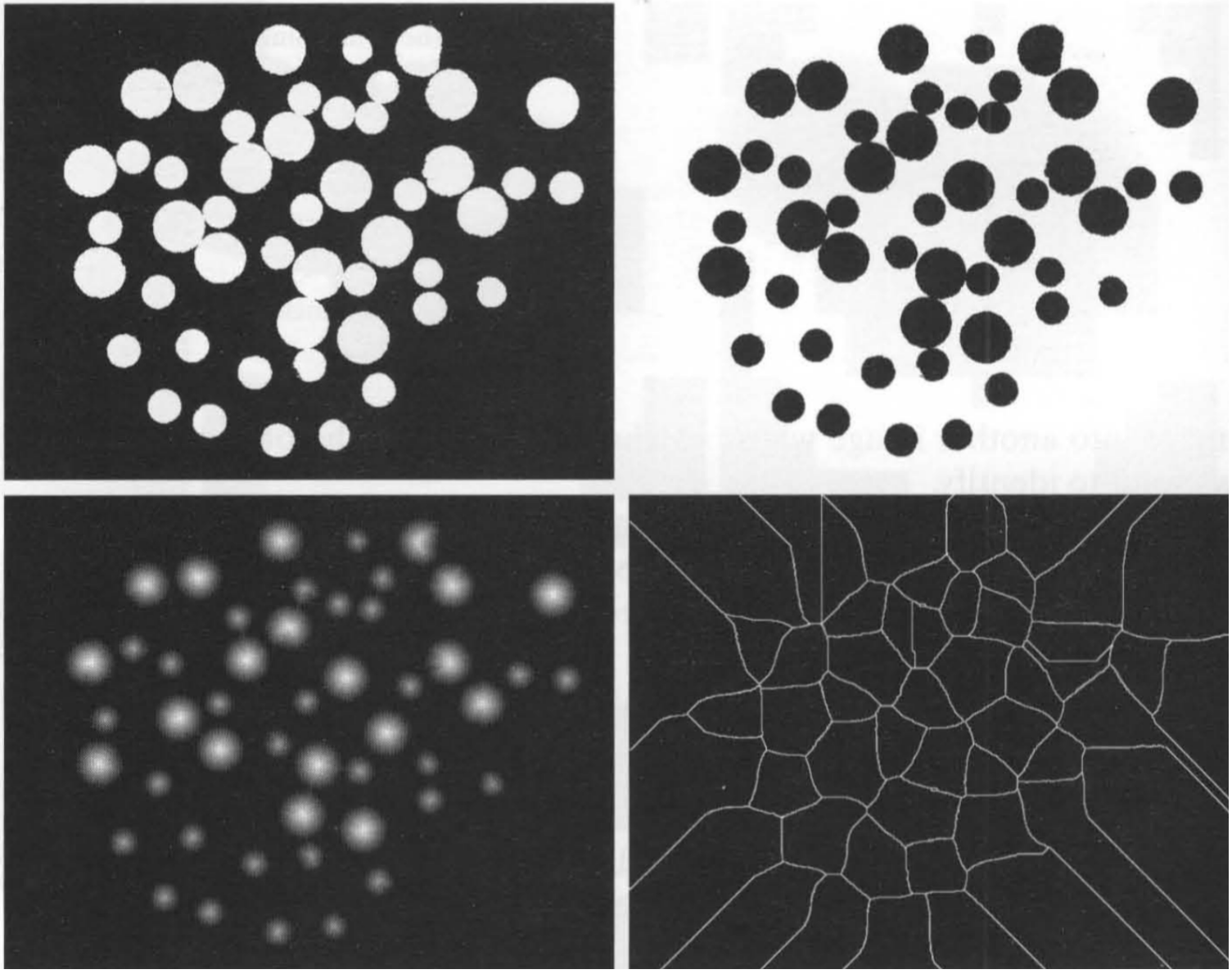
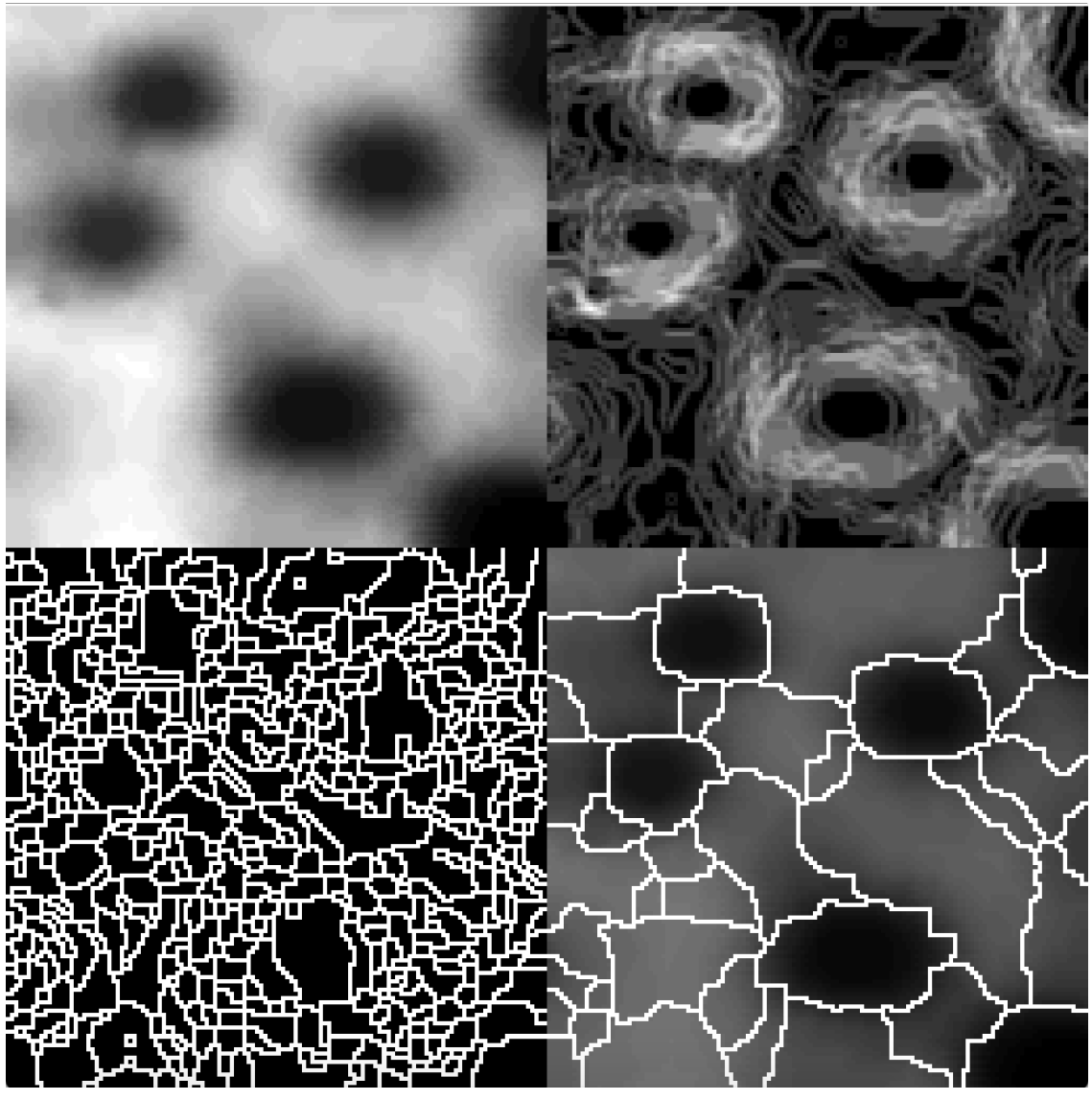


FIGURE 11.26
(a) Binary image. (b) Complement of image in (a). (c) Distance transform. (d) Watershed ridge lines of the negative of the distance transform. (e) Watershed ridge lines superimposed in black over original binary image. Some oversegmentation is evident.

```
f = imread('Fig1021(a)(small-blobs).tif');
h = fspecial('sobel');
fd = tofloat(f);
g = sqrt(imfilter(fd, h, 'replicate') .^ 2 + ...
    imfilter(fd, h', 'replicate') .^ 2);

L = watershed(g);
wr = L == 0;

g2 = imclose(imopen(g, ones(3,3)), ones(3,3));
L2 = watershed(g2);
wr2 = L2 == 0;
f2 = f;
f2(wr2) = 255;
```



```
f0 = imread('Fig1022(a)(gel-image).tif');
h = fspecial('sobel');
fd = tofloat(f0);
g = sqrt(imfilter(fd, h, 'replicate') .^ 2 + ...
    imfilter(fd, h', 'replicate') .^ 2);
L = watershed(g);
wr = L == 0;
rm = imregionalmin(g);
```