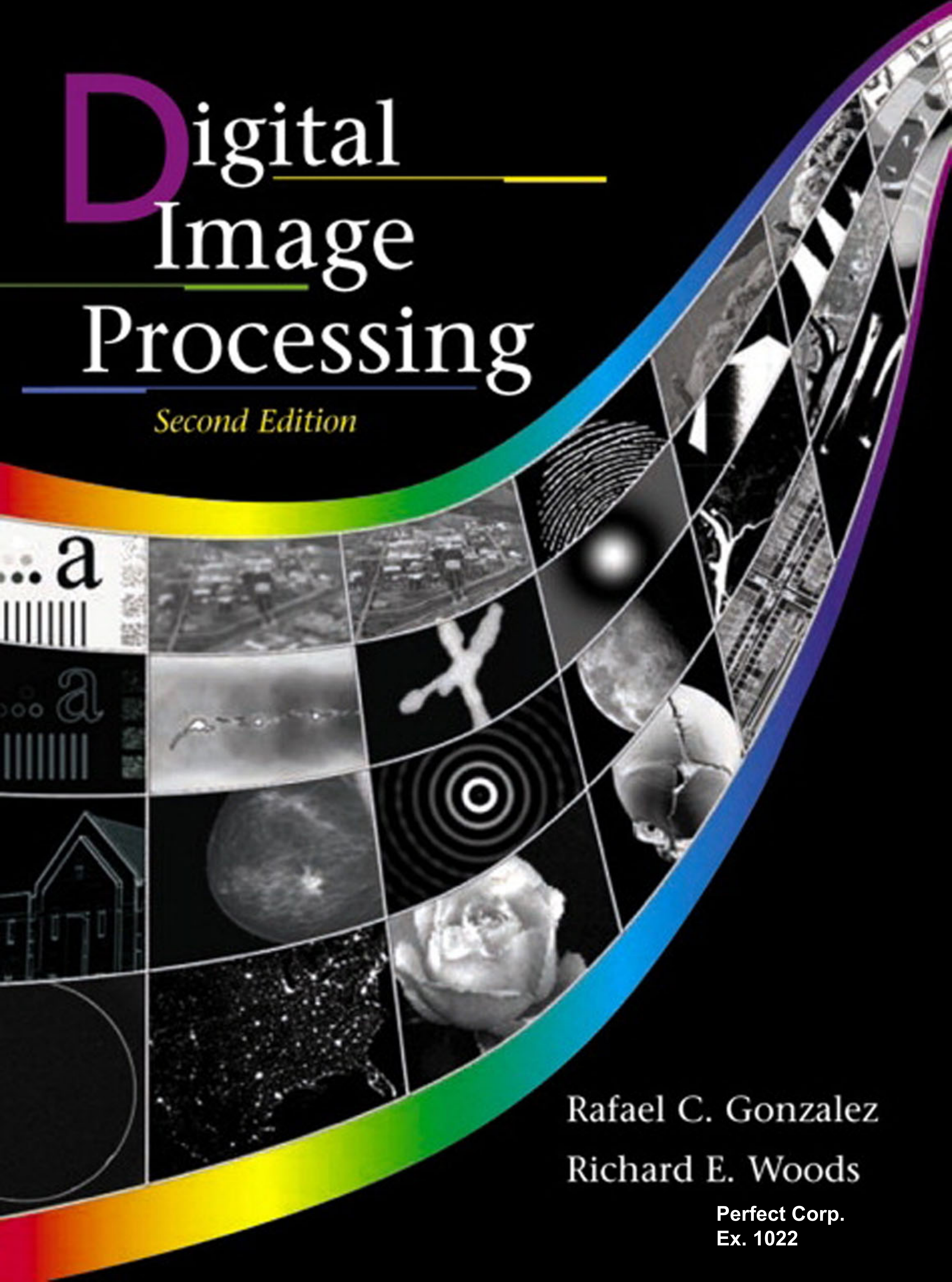
Digital Image Processing

Second Edition



Rafael C. Gonzalez
Richard E. Woods

Perfect Corp.
Ex. 1022

Digital Image Processing

Second Edition

Rafael C. Gonzalez
University of Tennessee

Richard E. Woods
MedData Interactive

Prentice Hall
Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Pubblication Data

Gonzalez, Rafael C.
Digital Image Processing / Richard E. Woods
p. cm.
Includes bibliographical references
ISBN 0-201-18075-8
1. Digital Imaging. 2. Digital Techniques. I. Title.

TA1632.G66 2001
621.3—dc21 2001035846
CIP

Vice-President and Editorial Director, ECS: *Marcia J. Horton*
Publisher: *Tom Robbins*
Associate Editor: *Alice Dworkin*
Editorial Assistant: *Jody McDonnell*
Vice President and Director of Production and Manufacturing, ESM: *David W. Riccardi*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *David A. George*
Production Editor: *Rose Kernan*
Composition: *Prepare, Inc.*
Director of Creative Services: *Paul Belfanti*
Creative Director: *Carole Anson*
Art Director and Cover Designer: *Heather Scott*
Art Editor: *Greg Dulles*
Manufacturing Manager: *Trudy Pisciotti*
Manufacturing Buyer: *Lisa McDowell*
Senior Marketing Manager: *Jennie Burger*

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN: 0-201-18075-8

Pearson Education Ltd., *London*
Pearson Education Australia Pty., Limited, *Sydney*
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., *Hong Kong*
Pearson Education Canada, Ltd., *Toronto*
Pearson Education de Mexico, S.A. de C.V.
Pearson Education—Japan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, *Upper Saddle River, New Jersey*

Contents

Preface xv
Acknowledgements xviii
About the Authors xix

1 *Introduction 15*

1.1 What Is Digital Image Processing? 15
1.2 The Origins of Digital Image Processing 17
1.3 Examples of Fields that Use Digital Image Processing 21
1.3.1 Gamma-Ray Imaging 22
1.3.2 X-ray Imaging 23
1.3.3 Imaging in the Ultraviolet Band 25
1.3.4 Imaging in the Visible and Infrared Bands 26
1.3.5 Imaging in the Microwave Band 32
1.3.6 Imaging in the Radio Band 34
1.3.7 Examples in which Other Imaging Modalities Are Used 34
1.4 Fundamental Steps in Digital Image Processing 39
1.5 Components of an Image Processing System 42
Summary 44
References and Further Reading 45

2 *Digital Image Fundamentals 34*

2.1 Elements of Visual Perception 34
2.1.1 Structure of the Human Eye 35
2.1.2 Image Formation in the Eye 37
2.1.3 Brightness Adaptation and Discrimination 38
2.2 Light and the Electromagnetic Spectrum 42
2.3 Image Sensing and Acquisition 45
2.3.1 Image Acquisition Using a Single Sensor 47
2.3.2 Image Acquisition Using Sensor Strips 48
2.3.3 Image Acquisition Using Sensor Arrays 49
2.3.4 A Simple Image Formation Model 50
2.4 Image Sampling and Quantization 52
2.4.1 Basic Concepts in Sampling and Quantization 52
2.4.2 Representing Digital Images 54
2.4.3 Spatial and Gray-Level Resolution 57
2.4.4 Aliasing and Moiré Patterns 62
2.4.5 Zooming and Shrinking Digital Images 64

2.5 Some Basic Relationships Between Pixels 66
2.5.1 Neighbors of a Pixel 66
2.5.2 Adjacency, Connectivity, Regions, and Boundaries 66
2.5.3 Distance Measures 68
2.5.4 Image Operations on a Pixel Basis 69
2.6 Linear and Nonlinear Operations 70
Summary 70
References and Further Reading 70
Problems 71

3 *Image Enhancement in the Spatial Domain 75*

3.1 Background 76
3.2 Some Basic Gray Level Transformations 78
3.2.1 Image Negatives 78
3.2.2 Log Transformations 79
3.2.3 Power-Law Transformations 80
3.2.4 Piecewise-Linear Transformation Functions 85
3.3 Histogram Processing 88
3.3.1 Histogram Equalization 91
3.3.2 Histogram Matching (Specification) 94
3.3.3 Local Enhancement 103
3.3.4 Use of Histogram Statistics for Image Enhancement 103
3.4 Enhancement Using Arithmetic/Logic Operations 108
3.4.1 Image Subtraction 110
3.4.2 Image Averaging 112
3.5 Basics of Spatial Filtering 116
3.6 Smoothing Spatial Filters 119
3.6.1 Smoothing Linear Filters 119
3.6.2 Order-Statistics Filters 123
3.7 Sharpening Spatial Filters 125
3.7.1 Foundation 125
3.7.2 Use of Second Derivatives for Enhancement–
The Laplacian 128
3.7.3 Use of First Derivatives for Enhancement—The Gradient 134
3.8 Combining Spatial Enhancement Methods 137
Summary 141
References and Further Reading 142
Problems 142

4 *Image Enhancement in the Frequency Domain 147*

4.1 Background 148

4.2 Introduction to the Fourier Transform and the Frequency Domain 149
- 4.2.1 The One-Dimensional Fourier Transform and its Inverse 150
- 4.2.2 The Two-Dimensional DFT and Its Inverse 154
- 4.2.3 Filtering in the Frequency Domain 156
- 4.2.4 Correspondence between Filtering in the Spatial and Frequency Domains 161

4.3 Smoothing Frequency-Domain Filters 167
- 4.3.1 Ideal Lowpass Filters 167
- 4.3.2 Butterworth Lowpass Filters 173
- 4.3.3 Gaussian Lowpass Filters 175
- 4.3.4 Additional Examples of Lowpass Filtering 178

4.4 Sharpening Frequency Domain Filters 180
- 4.4.1 Ideal Highpass Filters 182
- 4.4.2 Butterworth Highpass Filters 183
- 4.4.3 Gaussian Highpass Filters 184
- 4.4.4 The Laplacian in the Frequency Domain 185
- 4.4.5 Unsharp Masking, High-Boost Filtering, and High-Frequency Emphasis Filtering 187

4.5 Homomorphic Filtering 191

4.6 Implementation 194
- 4.6.1 Some Additional Properties of the 2-D Fourier Transform 194
- 4.6.2 Computing the Inverse Fourier Transform Using a Forward Transform Algorithm 198
- 4.6.3 More on Periodicity: the Need for Padding 199
- 4.6.4 The Convolution and Correlation Theorems 205
- 4.6.5 Summary of Properties of the 2-D Fourier Transform 208
- 4.6.6 The Fast Fourier Transform 208
- 4.6.7 Some Comments on Filter Design 213

Summary 214
References 214
Problems 215

5 *Image Restoration 220*

5.1 A Model of the Image Degradation/Restoration Process 221

5.2 Noise Models 222
- 5.2.1 Spatial and Frequency Properties of Noise 222
- 5.2.2 Some Important Noise Probability Density Functions 222
- 5.2.3 Periodic Noise 227
- 5.2.4 Estimation of Noise Parameters 227

5.3 Restoration in the Presence of Noise Only–Spatial Filtering 230
- 5.3.1 Mean Filters 231
- 5.3.2 Order-Statistics Filters 233
- 5.3.3 Adaptive Filters 237

5.4 Periodic Noise Reduction by Frequency Domain Filtering 243
5.4.1 Bandreject Filters 244
5.4.2 Bandpass Filters 245
5.4.3 Notch Filters 246
5.4.4 Optimum Notch Filtering 248
5.5 Linear, Position-Invariant Degradations 254
5.6 Estimating the Degradation Function 256
5.6.1 Estimation by Image Observation 256
5.6.2 Estimation by Experimentation 257
5.6.3 Estimation by Modeling 258
5.7 Inverse Filtering 261
5.8 Minimum Mean Square Error (Wiener) Filtering 262
5.9 Constrained Least Squares Filtering 266
5.10 Geometric Mean Filter 270
5.11 Geometric Transformations 270
5.11.1 Spatial Transformations 271
5.11.2 Gray-Level Interpolation 272
Summary 276
References and Further Reading 277
Problems 278

6 *Color Image Processing 282*

6.1 Color Fundamentals 283
6.2 Color Models 289
6.2.1 The RGB Color Model 290
6.2.2 The CMY and CMYK Color Models 294
6.2.3 The HSI Color Model 295
6.3 Pseudocolor Image Processing 302
6.3.1 Intensity Slicing 303
6.3.2 Gray Level to Color Transformations 308
6.4 Basics of Full-Color Image Processing 313
6.5 Color Transformations 315
6.5.1 Formulation 315
6.5.2 Color Complements 318
6.5.3 Color Slicing 320
6.5.4 Tone and Color Corrections 322
6.5.5 Histogram Processing 326
6.6 Smoothing and Sharpening 327
6.6.1 Color Image Smoothing 328
6.6.2 Color Image Sharpening 330
6.7 Color Segmentation 331
6.7.1 Segmentation in HSI Color Space 331
6.7.2 Segmentation in RGB Vector Space 333
6.7.3 Color Edge Detection 335

6.8 Noise in Color Images 339
6.9 Color Image Compression 342
Summary 343
References and Further Reading 344
Problems 344

7 *Wavelets and Multiresolution Processing 349*

7.1 Background 350
7.1.1 Image Pyramids 351
7.1.2 Subband Coding 354
7.1.3 The Haar Transform 360
7.2 Multiresolution Expansions 363
7.2.1 Series Expansions 364
7.2.2 Scaling Functions 365
7.2.3 Wavelet Functions 369
7.3 Wavelet Transforms in One Dimension 372
7.3.1 The Wavelet Series Expansions 372
7.3.2 The Discrete Wavelet Transform 375
7.3.3 The Continuous Wavelet Transform 376
7.4 The Fast Wavelet Transform 379
7.5 Wavelet Transforms in Two Dimensions 386
7.6 Wavelet Packets 394
Summary 402
References and Further Reading 404
Problems 404

8 *Image Compression 409*

8.1 Fundamentals 411
8.1.1 Coding Redundancy 412
8.1.2 Interpixel Redundancy 414
8.1.3 Psychovisual Redundancy 417
8.1.4 Fidelity Criteria 419
8.2 Image Compression Models 421
8.2.1 The Source Encoder and Decoder 421
8.2.2 The Channel Encoder and Decoder 423
8.3 Elements of Information Theory 424
8.3.1 Measuring Information 424
8.3.2 The Information Channel 425
8.3.3 Fundamental Coding Theorems 430
8.3.4 Using Information Theory 437
8.4 Error-Free Compression 440
8.4.1 Variable-Length Coding 440

8.4.2 LZW Coding 446
8.4.3 Bit-Plane Coding 448
8.4.4 Lossless Predictive Coding 456
8.5 Lossy Compression 459
8.5.1 Lossy Predictive Coding 459
8.5.2 Transform Coding 467
8.5.3 Wavelet Coding 486
8.6 Image Compression Standards 492
8.6.1 Binary Image Compression Standards 493
8.6.2 Continuous Tone Still Image Compression Standards 498
8.6.3 Video Compression Standards 510
Summary 513
References and Further Reading 513
Problems 514

9 *Morphological Image Processing 519*

9.1 Preliminaries 520
9.1.1 Some Basic Concepts from Set Theory 520
9.1.2 Logic Operations Involving Binary Images 522
9.2 Dilation and Erosion 523
9.2.1 Dilation 523
9.2.2 Erosion 525
9.3 Opening and Closing 528
9.4 The Hit-or-Miss Transformation 532
9.5 Some Basic Morphological Algorithms 534
9.5.1 Boundary Extraction 534
9.5.2 Region Filling 535
9.5.3 Extraction of Connected Components 536
9.5.4 Convex Hull 539
9.5.5 Thinning 541
9.5.6 Thickening 541
9.5.7 Skeletons 543
9.5.8 Pruning 545
9.5.9 Summary of Morphological Operations on Binary Images 547
9.6 Extensions to Gray-Scale Images 550
9.6.1 Dilation 550
9.6.2 Erosion 552
9.6.3 Opening and Closing 554
9.6.4 Some Applications of Gray-Scale Morphology 556
Summary 560
References and Further Reading 560
Problems 560

10 *Image Segmentation* *567*

10.1 Detection of Discontinuities 568
10.1.1 Point Detection 569
10.1.2 Line Detection 570
10.1.3 Edge Detection 572
10.2 Edge Linking and Boundary Detection 585
10.2.1 Local Processing 585
10.2.2 Global Processing via the Hough Transform 587
10.2.3 Global Processing via Graph-Theoretic Techniques 591
10.3 Thresholding 595
10.3.1 Foundation 595
10.3.2 The Role of Illumination 596
10.3.3 Basic Global Thresholding 598
10.3.4 Basic Adaptive Thresholding 600
10.3.5 Optimal Global and Adaptive Thresholding 602
10.3.6 Use of Boundary Characteristics for Histogram Improvement and Local Thresholding 608
10.3.7 Thresholds Based on Several Variables 611
10.4 Region-Based Segmentation 612
10.4.1 Basic Formulation 612
10.4.2 Region Growing 613
10.4.3 Region Splitting and Merging 615
10.5 Segmentation by Morphological Watersheds 617
10.5.1 Basic Concepts 617
10.5.2 Dam Construction 620
10.5.3 Watershed Segmentation Algorithm 622
10.5.4 The Use of Markers 624
10.6 The Use of Motion in Segmentation 626
10.6.1 Spatial Techniques 626
10.6.2 Frequency Domain Techniques 630
Summary 634
References and Further Reading 634
Problems 636

11 *Representation and Description* *643*

11.1 Representation 644
11.1.1 Chain Codes 644
11.1.2 Polygonal Approximations 646
11.1.3 Signatures 648
11.1.4 Boundary Segments 649
11.1.5 Skeletons 650

11.2 Boundary Descriptors 653
11.2.1 Some Simple Descriptors 653
11.2.2 Shape Numbers 654
11.2.3 Fourier Descriptors 655
11.2.4 Statistical Moments 659
11.3 Regional Descriptors 660
11.3.1 Some Simple Descriptors 661
11.3.2 Topological Descriptors 661
11.3.3 Texture 665
11.3.4 Moments of Two-Dimensional Functions 672
11.4 Use of Principal Components for Description 675
11.5 Relational Descriptors 683
Summary 687
References and Further Reading 687
Problems 689

12 *Object Recognition 693*

12.1 Patterns and Pattern Classes 693
12.2 Recognition Based on Decision-Theoretic Methods 698
12.2.1 Matching 698
12.2.2 Optimum Statistical Classifiers 704
12.2.3 Neural Networks 712
12.3 Structural Methods 732
12.3.1 Matching Shape Numbers 732
12.3.2 String Matching 734
12.3.3 Syntactic Recognition of Strings 735
12.3.4 Syntactic Recognition of Trees 740
Summary 750
References and Further Reading 750
Problems 750

Bibliography 755

Index 779

10 Image Segmentation

The whole is equal to the sum of its parts.

Euclid

The whole is greater than the sum of its parts.

Max Wertheimer

Preview

The material in the previous chapter began a transition from image processing methods whose input and output are images, to methods in which the inputs are images, but the outputs are attributes extracted from those images (in the sense defined in Section 1.1). Segmentation is another major step in that direction.

Segmentation subdivides an image into its constituent regions or objects. The level to which the subdivision is carried depends on the problem being solved. That is, segmentation should stop when the objects of interest in an application have been isolated. For example, in the automated inspection of electronic assemblies, interest lies in analyzing images of the products with the objective of determining the presence or absence of specific anomalies, such as missing components or broken connection paths. There is no point in carrying segmentation past the level of detail required to identify those elements.

Segmentation of nontrivial images is one of the most difficult tasks in image processing. Segmentation accuracy determines the eventual success or failure of computerized analysis procedures. For this reason, considerable care should be taken to improve the probability of rugged segmentation. In some situations, such as industrial inspection applications, at least some measure of control over the environment is possible at times. The experienced image processing system designer invariably pays considerable attention to such opportunities. In other applications, such as autonomous target acquisition, the system designer has no control of the environment. Then the usual approach is to focus on selecting the types of sensors most likely to enhance the objects of interest while diminishing the contribution of irrelevant image detail. A good example is the use of infrared imaging by the military to detect objects with strong heat signatures, such as equipment and troops in motion.

Image segmentation algorithms generally are based on one of two basic properties of intensity values: discontinuity and similarity. In the first category, the approach is to partition an image based on abrupt changes in intensity, such as edges in an image. The principal approaches in the second category are based on partitioning an image into regions that are similar according to a set of predefined criteria. Thresholding, region growing, and region splitting and merging are examples of methods in this category.

In this chapter we discuss a number of approaches in the two categories just mentioned. We begin the development with methods suitable for detecting gray-level discontinuities such as points, lines, and edges. Edge detection in particular has been a staple of segmentation algorithms for many years. In addition to edge detection per se, we also discuss methods for connecting edge segments and for "assembling" edges into region boundaries. The discussion on edge detection is followed by the introduction of various thresholding techniques. Thresholding also is a fundamental approach to segmentation that enjoys a significant degree of popularity, especially in applications where speed is an important factor. The discussion on thresholding is followed by the development of several region-oriented segmentation approaches. We then discuss a morphological approach to segmentation called *watershed segmentation*. This approach is particularly attractive because it combines several of the positive attributes of segmentation based on the techniques presented in the first part of the chapter. We conclude the chapter with a discussion on the use of motion cues for image segmentation.

10.1 Detection of Discontinuities

In this section we present several techniques for detecting the three basic types of gray-level discontinuities in a digital image: points, lines, and edges. The most common way to look for discontinuities is to run a mask through the image in the manner described in Section 3.5. For the 3 × 3 mask shown in Fig. 10.1, this procedure involves computing the sum of products of the coefficients with the gray

FIGURE 10.1 A general 3 × 3 mask.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

levels contained in the region encompassed by the mask. That is, with reference to Eq. (3.5-3), the response of the mask at any point in the image is given by

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \tag{10.1-1}$$

where z_i is the gray level of the pixel associated with mask coefficient w_i. As usual, the response of the mask is defined with respect to its center location. The details for implementing mask operations are discussed in Section 3.5.

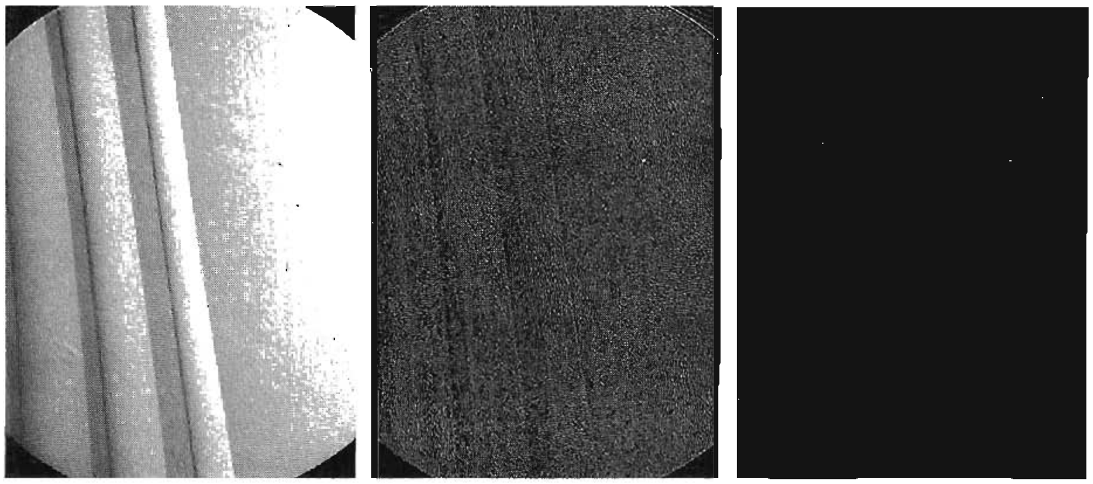
10.1.1 Point Detection

The detection of isolated points in an image is straightforward in principle. Using the mask shown in Fig. 10.2(a), we say that a point has been detected at the location on which the mask is centered if

$$|R| \geq T \tag{10.1-2}$$

where T is a nonnegative threshold and R is given by Eq. (10.1-1). Basically, this formulation measures the weighted differences between the center point and its neighbors. The idea is that an *isolated* point (a point whose gray level is significantly different from its background and which is located in a homogeneous or nearly homogeneous area) will be quite different from its surroundings, and thus be easily detectable by this type of mask. Note that the mask in Fig. 10.2(a) is the same as the mask shown in Fig. 3.39(d) in connection with Laplacian operations. However, the emphasis here is strictly on the detection of points. That is, the only differences that are considered of interest are those

−1	−1	−1
−1	8	−1
−1	−1	−1



a
b c d

FIGURE 10.2 (a) Point detection mask. (b) X-ray image of a turbine blade with a porosity. (c) Result of point detection. (d) Result of using Eq. (10.1-2). (Original image courtesy of X-TEK Systems Ltd.)

large enough (as determined by T) to be considered isolated points. Note that the mask coefficients sum to zero, indicating that the mask response will be zero in areas of constant gray level.

EXAMPLE 10.1: Detection of isolated points in an image.

■ We illustrate segmentation of isolated points from an image with the aid of Fig. 10.2(b), which shows an X-ray image of a jet-engine turbine blade with a porosity in the upper, right quadrant of the image. There is a single black pixel embedded within the porosity. Figure 10.2(c) is the result of applying the point detector mask to the X-ray image, and Fig. 10.2(d) shows the result of using Eq. (10.1-2) with T equal to 90% of the highest absolute pixel value of the image in Fig. 10.2(c). (Threshold selection is discussed in detail in Section 10.3.) The single pixel is clearly visible in this image (the pixel was enlarged manually so that it would be visible after printing). This type of detection process is rather specialized because it is based on single-pixel discontinuities that have a homogeneous background in the area of the detector mask. When this condition is not satisfied, other methods discussed in this chapter are more suitable for detecting gray-level discontinuities. ■

10.1.2 Line Detection

The next level of complexity is line detection. Consider the masks shown in Fig. 10.3. If the first mask were moved around an image, it would respond more strongly to lines (one pixel thick) oriented horizontally. With a constant background, the maximum response would result when the line passed through the middle row of the mask. This is easily verified by sketching a simple array of 1's with a line of a different gray level (say, 5's) running horizontally through the array. A similar experiment would reveal that the second mask in Fig. 10.3 responds best to lines oriented at +45°; the third mask to vertical lines; and the fourth mask to lines in the −45° direction. These directions can be established also by noting that the preferred direction of each mask is weighted with a larger coefficient (i.e., 2) than other possible directions. Note that the coefficients in each mask sum to zero, indicating a zero response from the masks in areas of constant gray level.

Let R_1, R_2, R_3, and R_4 denote the responses of the masks in Fig. 10.3, from left to right, where the R's are given by Eq. (10.1-1). Suppose that the four masks are run individually through an image. If, at a certain point in the image, $|R_i| > |R_j|$, for all $j \neq i$, that point is said to be more likely associated with a line in the direction of mask i. For example, if at a point in the image, $|R_1| > |R_j|$ for

FIGURE 10.3 Line masks.

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

+45°

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

2	−1	−1
−1	2	−1
−1	−1	2

−45°

$j = 2, 3, 4$, that particular point is said to be more likely associated with a horizontal line. Alternatively, we may be interested in detecting lines in a specified direction. In this case, we would use the mask associated with that direction and threshold its output, as in Eq. (10.1-2). In other words, if we are interested in detecting all the lines in an image in the direction defined by a given mask, we simply run the mask through the image and threshold the absolute value of the result. The points that are left are the strongest responses, which, for lines one pixel thick, correspond closest to the direction defined by the mask. The following example illustrates this procedure.

EXAMPLE 10.2: Detection of lines in a specified direction.

Figure 10.4(a) shows a digitized (binary) portion of a wire-bond mask for an electronic circuit. Suppose that we are interested in finding all the lines that are one pixel thick and are oriented at $-45°$. For this purpose, we use the last mask shown in Fig. 10.3. The absolute value of the result is shown in Fig. 10.4(b). Note that all vertical and horizontal components of the image were eliminated, and that the components of the original image that tend toward a $-45°$ direction

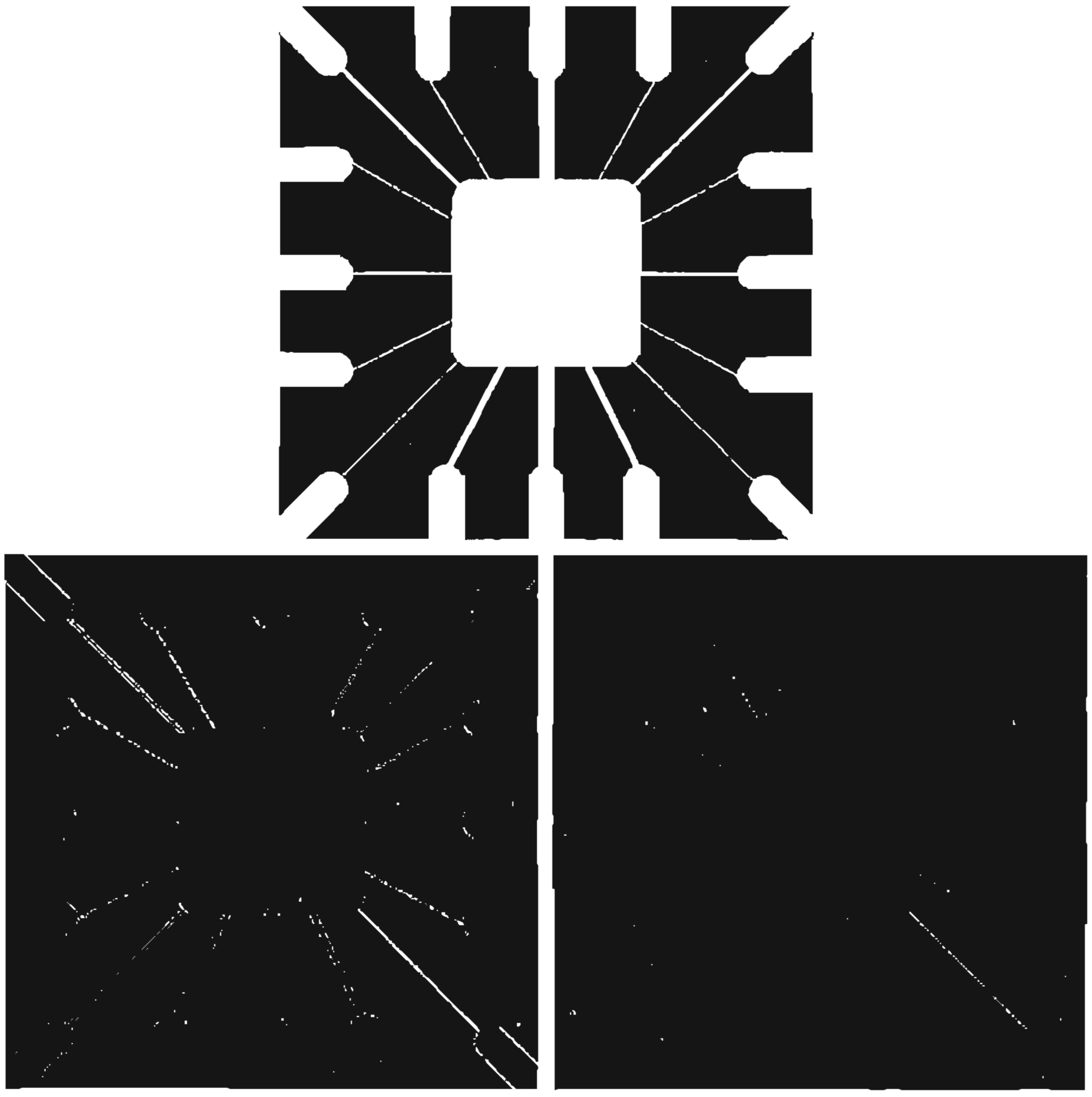

a
b c

FIGURE 10.4 Illustration of line detection. (a) Binary wire-bond mask. (b) Absolute value of result after processing with $-45°$ line detector. (c) Result of thresholding image (b).

produced the strongest responses in Fig. 10.4(b). In order to determine which lines best fit the mask, we simply threshold this image. The result of using a threshold equal to the maximum value in the image is shown in Fig. 10.4(c). The maximum value is a good choice for a threshold in applications such as this because the input image is binary and we are looking for the strongest responses. Figure 10.4(c) shows in white all points that passed the threshold test. In this case, the procedure extracted the only line segment that was one pixel thick and oriented at $-45°$ (the other component of the image oriented in this direction in the top, left quadrant is not one pixel thick). The isolated points shown in Fig. 10.4(c) are points that also had similarly strong responses to the mask. In the original image, these points and their immediate neighbors are oriented in such as way that the mask produced a maximum response at those isolated locations. These isolated points can be detected using the mask in Fig. 10.2(a) and then deleted, or they could be deleted using morphological erosion, as discussed in the last chapter. ■

10.1.3 Edge Detection

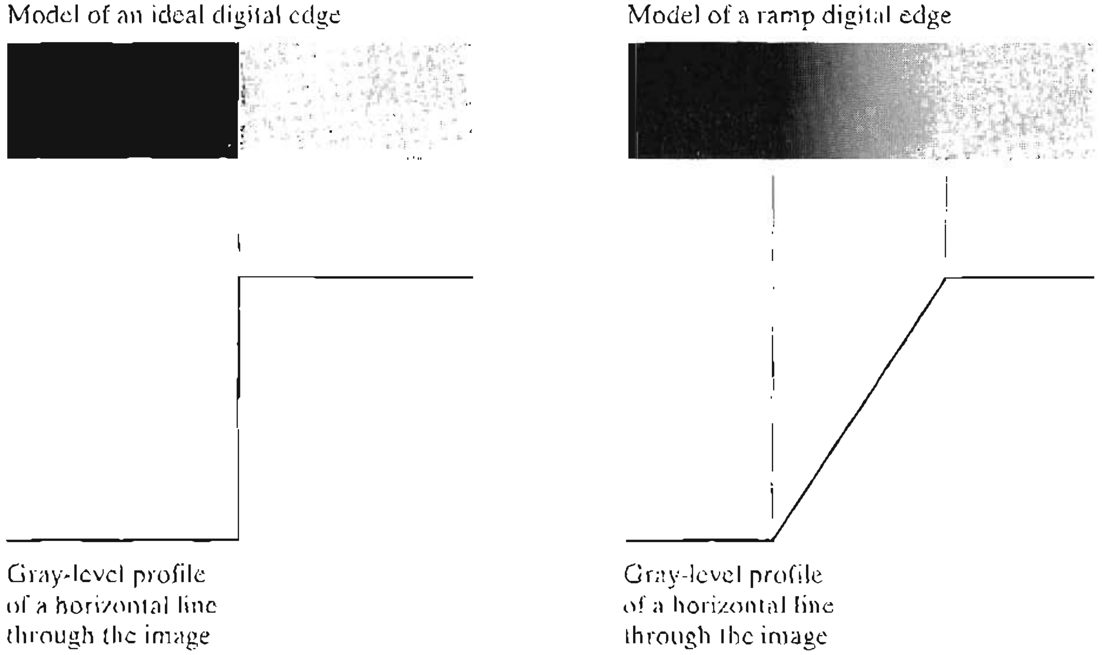
Although point and line detection certainly are important in any discussion on segmentation, edge detection is by far the most common approach for detecting meaningful discontinuities in gray level. In this section we discuss approaches for implementing first- and second-order digital derivatives for the detection of edges in an image. We introduced these derivatives in Section 3.7 in the context of image enhancement. The focus in this section is on their properties for edge detection. Some of the concepts previously introduced are restated briefly here for the sake continuity in the discussion.

Basic formulation

Edges were introduced informally in Section 3.7.1. In this section we look at the concept of a digital edge a little closer. Intuitively, an edge is a set of connected pixels that lie on the boundary between two regions. However, we already went through some length in Section 2.5.2 to explain the difference between an edge and a boundary. Fundamentally, as we shall see shortly, an edge is a "local" concept whereas a region boundary, owing to the way it is defined, is a more global idea. A reasonable definition of "edge" requires the ability to measure gray-level transitions in a meaningful way.

We start by modeling an edge intuitively. This will lead us to a formalism in which "meaningful" transitions in gray levels can be measured. Intuitively, an ideal edge has the properties of the model shown in Fig. 10.5(a). An ideal edge according to this model is a set of connected pixels (in the vertical direction here), each of which is located at an orthogonal step transition in gray level (as shown by the horizontal profile in the figure).

In practice, optics, sampling, and other image acquisition imperfections yield edges that are blurred, with the degree of blurring being determined by factors such as the quality of the image acquisition system, the sampling rate, and illumination conditions under which the image is acquired. As a result, edges are more closely modeled as having a "ramplike" profile, such as the one shown in


Model of an ideal digital edge
Model of a ramp digital edge
Gray-level profile of a horizontal line through the image
Gray-level profile of a horizontal line through the image

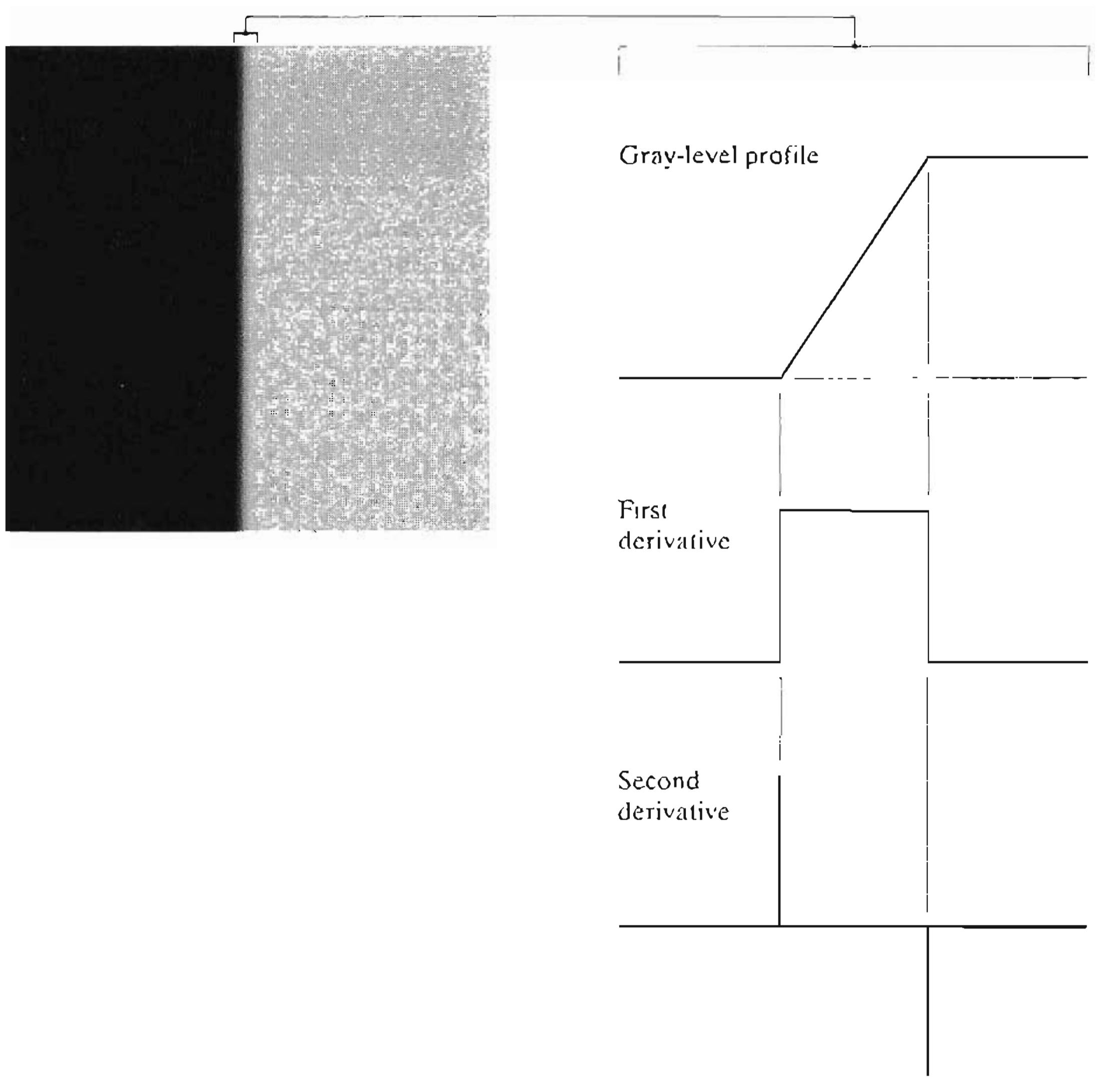
a b

FIGURE 10.5 (a) Model of an ideal digital edge. (b) Model of a ramp edge. The slope of the ramp is proportional to the degree of blurring in the edge.

Fig. 10.5(b). The slope of the ramp is inversely proportional to the degree of blurring in the edge. In this model, we no longer have a thin (one pixel thick) path. Instead, an edge point now is any point contained in the ramp, and an edge would then be a set of such points that are connected. The "thickness" of the edge is determined by the length of the ramp, as it transitions from an initial to a final gray level. This length is determined by the slope, which, in turn, is determined by the degree of blurring. This makes sense: Blurred edges tend to be thick and sharp edges tend to be thin.

Figure 10.6(a) shows the image from which the close-up in Fig. 10.5(b) was extracted. Figure 10.6(b) shows a horizontal gray-level profile of the edge between the two regions. This figure also shows the first and second derivatives of the gray-level profile. The first derivative is positive at the points of transition into and out of the ramp as we move from left to right along the profile; it is constant for points in the ramp; and is zero in areas of constant gray level. The second derivative is positive at the transition associated with the dark side of the edge, negative at the transition associated with the light side of the edge, and zero along the ramp and in areas of constant gray level. The signs of the derivatives in Fig. 10.6(b) would be reversed for an edge that transitions from light to dark.

We conclude from these observations that the magnitude of the first derivative can be used to detect the presence of an edge at a point in an image (i.e., to determine if a point is on a ramp). Similarly, the sign of the second derivative can be used to determine whether an edge pixel lies on the dark or light side of an edge. We note two additional properties of the second derivative around an edge: (1) It produces two values for every edge in an image (an undesirable feature); and (2) an imaginary straight line joining the extreme positive and negative values of the second derivative would cross zero near the midpoint of the edge. This *zero-crossing* property of the second derivative is quite useful



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.

for locating the centers of thick edges, as we show later in this section. Finally, we note that some edge models make use of a smooth transition into and out of the ramp (Problem 10.5). However, the conclusions at which we arrive in the following discussion are the same. Also, it is evident from this discussion that we are dealing here with local measures (thus the comment made in Section 2.5.2 about the local nature of edges).

Although attention thus far has been limited to a 1-D horizontal profile, a similar argument applies to an edge of any orientation in an image. We simply define a profile perpendicular to the edge direction at any desired point and interpret the results as in the preceding discussion.

EXAMPLE 10.3: Behavior of the first and second derivatives around a noisy edge.

■ The edges shown in Fig. 10.5 and 10.6 are free of noise. The image segments in the first column in Fig. 10.7 show close-ups of four ramp edges separating a black region on the left and a white region on the right. It is important to keep in mind that the entire transition from black to white is a *single* edge. The image segment at the top, left is free of noise. The other three images in the first column of Fig. 10.7 are corrupted by additive Gaussian noise with zero mean and

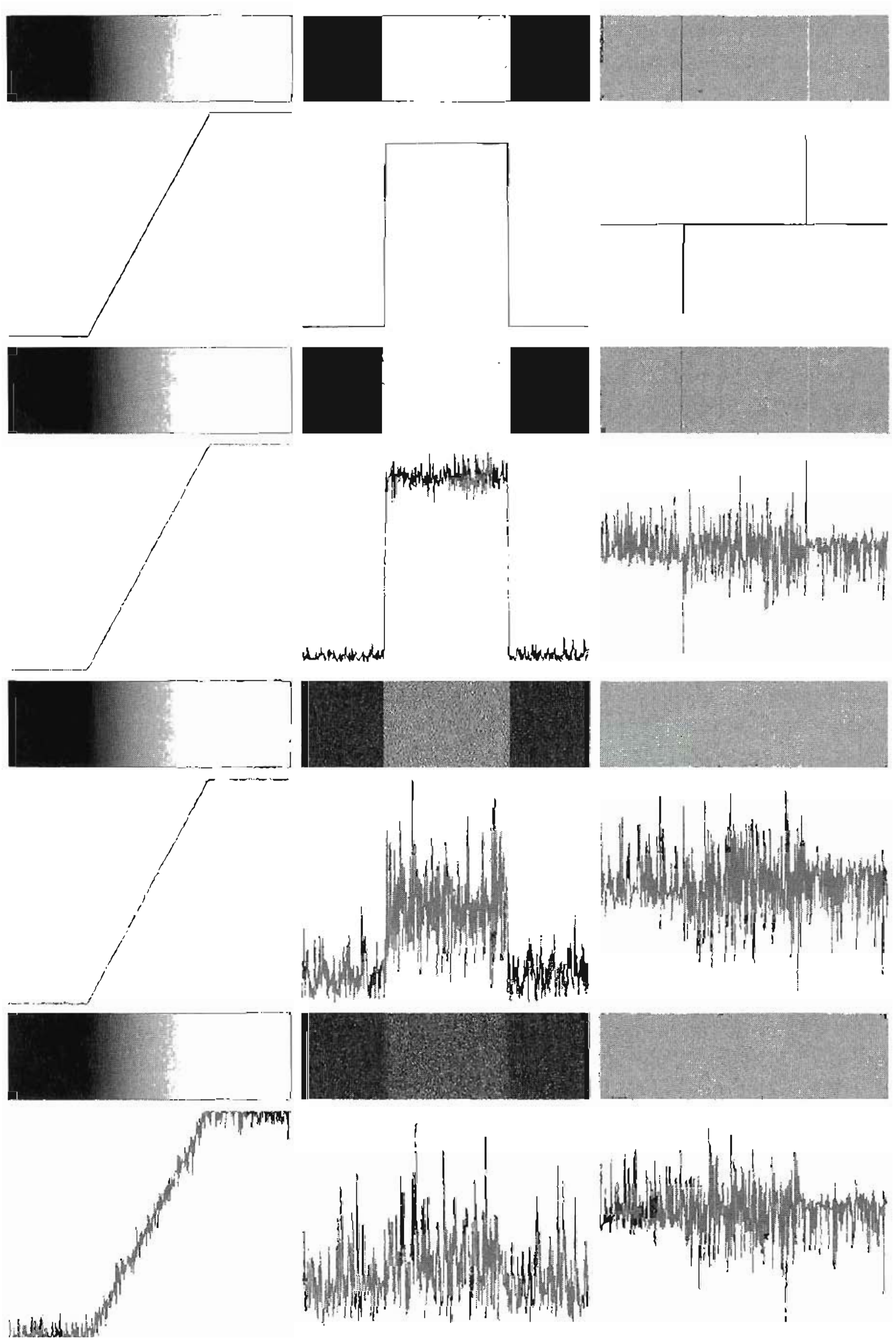


FIGURE 10.7 First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and σ = 0.0, 0.1, 1.0, and 10.0, respectively. Second column: first-derivative images and gray-level profiles. Third column: second-derivative images and gray-level profiles.

a
b
c
d

standard deviation of 0.1, 1.0, and 10.0 gray levels, respectively. The graph shown below each of these images is a gray-level profile of a horizontal scan line passing through the image.

The images in the second column of Fig. 10.7 are the first-order derivatives of the images on the left (we discuss computation of the first and second image derivatives in the following section). Consider, for example, the center image at the top. As discussed in connection with Fig. 10.6(b), the derivative is zero in the constant black and white regions. These are the two black areas shown in the derivative image. The derivative of a constant ramp is a constant, equal to the slope of the ramp. This constant area in the derivative image is shown in gray. As we move down the center column, the derivatives become increasingly different from the noiseless case. In fact, it would be difficult to associate the last profile in that column with a ramp edge. What makes these results interesting is that the noise really is almost invisible in the images on the left column. The last image is a slightly grainy, but this corruption is almost imperceptible. These examples are good illustrations of the sensitivity of derivatives to noise.

As expected, the second derivative is even more sensitive to noise. The second derivative of the noiseless image is shown in the top, right image. The thin black and white lines are the positive and negative components explained in Fig. 10.6. The gray in these images represents zero due to scaling. We note that the only noisy second derivative that resembles the noiseless case is the one corresponding to noise with a standard deviation of 0.1 gray levels. The other two second-derivative images and profiles clearly illustrate that it would be difficult indeed to detect their positive and negative components, which are the truly useful features of the second derivative in terms of edge detection.

The fact that fairly little noise can have such a significant impact on the two key derivatives used for edge detection in images is an important issue to keep in mind. In particular, image smoothing should be a serious consideration prior to the use of derivatives in applications where noise with levels similar to those we have just discussed is likely to be present. ■

Based on this example and on the three paragraphs that precede it, we are led to the conclusion that, to be classified as a meaningful edge point, the transition in gray level associated with that point has to be significantly stronger than the background at that point. Since we are dealing with local computations, the method of choice to determine whether a value is "significant" or not is to use a threshold. Thus, we *define* a point in an image as being an *edge point* if its two-dimensional first-order derivative is greater than a specified threshold. A set of such points that are connected according to a predefined criterion of connectedness (see Section 2.5.2) is by definition an *edge*. The term *edge segment* generally is used if the edge is short in relation to the dimensions of the image. A key problem in segmentation is to assemble edge segments into longer edges, as explained in Section 10.2. An alternate definition if we elect to use the second-derivative is simply to define the edge points in an image as the zero crossings of its second derivative. The definition of an edge in this case is the same as above. It is important to note that these definitions do not guarantee success in finding edges in an image. They simply give us a formalism to look for them.

As in Chapter 3, first-order derivatives in an image are computed using the gradient. Second-order derivatives are obtained using the Laplacian.

Gradient operators

First-order derivatives of a digital image are based on various approximations of the 2-D gradient. The gradient of an image $f(x, y)$ at location (x, y) is defined as the *vector*

$$\nabla\mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}. \tag{10.1-3}$$

See inside front cover
Consult the book web site for a brief review of vector analysis.

It is well known from vector analysis that the gradient vector points in the direction of maximum rate of change of f at coordinates (x, y).

An important quantity in edge detection is the magnitude of this vector, denoted ∇f, where

$$\nabla f = \text{mag}(\nabla\mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2}. \tag{10.1-4}$$

This quantity gives the maximum rate of increase of $f(x, y)$ per unit distance in the direction of $\nabla\mathbf{f}$. It is a common (although not strictly correct) practice to refer to ∇f also as the *gradient*. We will adhere to convention and also use this term interchangeably, differentiating between the vector and its magnitude only in cases in which confusion is likely.

The *direction* of the gradient vector also is an important quantity. Let $\alpha(x, y)$ represent the direction angle of the vector $\nabla\mathbf{f}$ at (x, y). Then, from vector analysis,

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{10.1-5}$$

where the angle is measured with respect to the x-axis. The direction of an edge at (x, y) is *perpendicular* to the direction of the gradient vector at that point.

Computation of the gradient of an image is based on obtaining the partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ at every pixel location. Let the 3×3 area shown in Fig. 10.8(a) represent the gray levels in a neighborhood of an image. As discussed in Section 3.7.3, one of the simplest ways to implement a first-order partial derivative at point z_5 is to use the following *Roberts cross-gradient operators*:

$$G_x = (z_9 - z_5) \tag{10.1-6}$$

and

$$G_y = (z_8 - z_6). \tag{10.1-7}$$

These derivatives can be implemented for an entire image by using the masks shown in Fig. 10.8(b) with the procedure discussed in Section 3.5.

Masks of size 2×2 are awkward to implement because they do not have a clear center. An approach using masks of size 3×3 is given by

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \tag{10.1-8}$$

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

and

$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7). \qquad (10.1\text{-}9)$$

In this formulation, the difference between the first and third rows of the 3 × 3 image region approximates the derivative in the x-direction, and the difference between the third and first columns approximates the derivative in the y-direction. The masks shown in Figs. 10.8(d) and (e), called the *Prewitt operators*, can be used to implement these two equations.

A slight variation of these two equations uses a weight of 2 in the center coefficient:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \qquad (10.1\text{-}10)$$

and

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7). \qquad (10.1\text{-}11)$$

A weight value of 2 is used to achieve some smoothing by giving more importance to the center point (Problem 10.8). Figures 10.8(f) and (g), called the *Sobel* operators, are used to implement these two equations. The Prewitt and Sobel operators are among the most used in practice for computing digital gradients. The Prewitt masks are simpler to implement than the Sobel masks, but the latter have slightly superior noise-suppression characteristics, an important issue when dealing with derivatives. Note that the coefficients in all the masks shown in Fig. 10.8 sum to 0, indicating that they give a response of 0 in areas of constant gray level, as expected of a derivative operator.

The masks just discussed are used to obtain the gradient components G_x and G_y. Computation of the gradient requires that these two components be combined in the manner shown in Eq. (10.1-4). However, this implementation is not always desirable because of the computational burden required by squares and square roots. An approach used frequently is to approximate the gradient by absolute values:

$$\nabla f \approx |G_x| + |G_y|. \tag{10.1-12}$$

This equation is much more attractive computationally, and it still preserves relative changes in gray levels. As discussed in Section 3.7.3, the price paid for this advantage is that the resulting filters will not be isotropic (invariant to rotation) in general. However, this is not an issue when masks such as the Prewitt and Sobel masks are used to compute G_x and G_y. These masks give isotropic results only for vertical and horizontal edges, so even if we used Eq. (10.1-4) to compute the gradient, the results would be isotropic only for edges in those directions. In this case, Eqs. (10.1-4) and (10.1-12) give the same result (Problem 10.6).

It is possible to modify the 3 × 3 masks in Fig. 10.8 so that they have their strongest responses along the diagonal directions. The two additional Prewitt and Sobel masks for detecting discontinuities in the diagonal directions are shown in Fig. 10.9.

EXAMPLE 10.4: Illustration of the gradient and its components.

■ Figure 10.10 illustrates the response of the two components of the gradient, $|G_x|$ and $|G_y|$, as well as the gradient image formed from the sum of these two

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

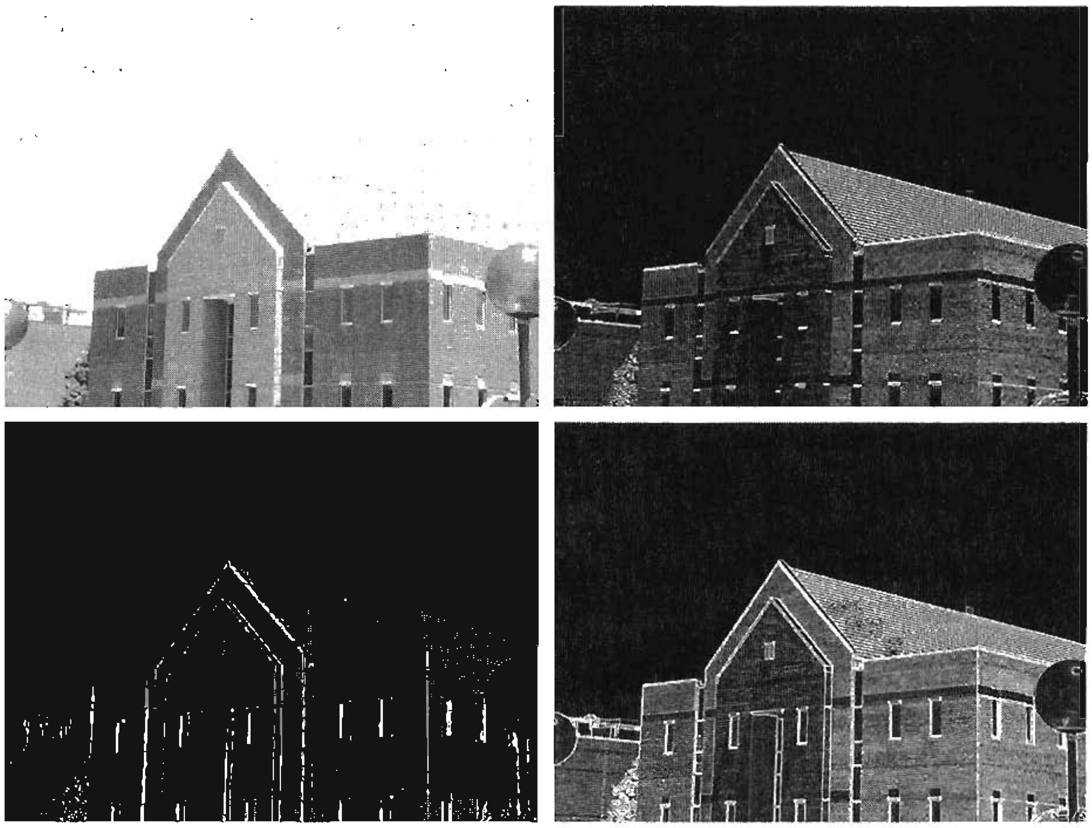
Prewitt

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

Sobel

a b
c d

FIGURE 10.9 Prewitt and Sobel masks for detecting diagonal edges.



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, $|G_x| + |G_y|$.

components. The directionality of the two components is evident in Figs. 10.10(b) and (c). Note in particular how strong the roof tile, horizontal brick joints, and horizontal segments of the windows are in Fig. 10.10(b). By contrast, Fig. 10.10(c) favors the vertical components, such as the corner of the near wall, the vertical components of the window, the vertical joints of the brick, and the lamppost on the right side of the picture.

The original image is of reasonably high resolution (1200×1600 pixels) and, at the distance the image was taken, the contribution made to image detail by the wall bricks is still significant. This level of detail often is undesirable, and one way to reduce it is to smooth the image. Figure 10.11 shows the same sequence of images as in Fig. 10.10, but with the original image being smoothed first using a 5×5 averaging filter. The response of each mask now shows almost no contribution due to the bricks, with the result being dominated mostly by the principal edges. Note that averaging caused the response of all edges to be weaker.

In Figs. 10.10 and 10.11, it is evident that the horizontal and vertical Sobel masks respond about equally well to edges oriented in the minus and plus 45° directions. If it is important to emphasize edges along the diagonal directions, then one of the mask pairs in Fig. 10.9 should be used. The absolute responses of the diagonal Sobel masks are shown in Fig. 10.12. The stronger diagonal response of these masks is evident in this figure. Both diagonal masks have similar response to horizontal and vertical edges but, as expected, their response in these directions is weaker than the response of the horizontal and vertical Sobel masks shown in Figs. 10.10(b) and 10.10(c). ■

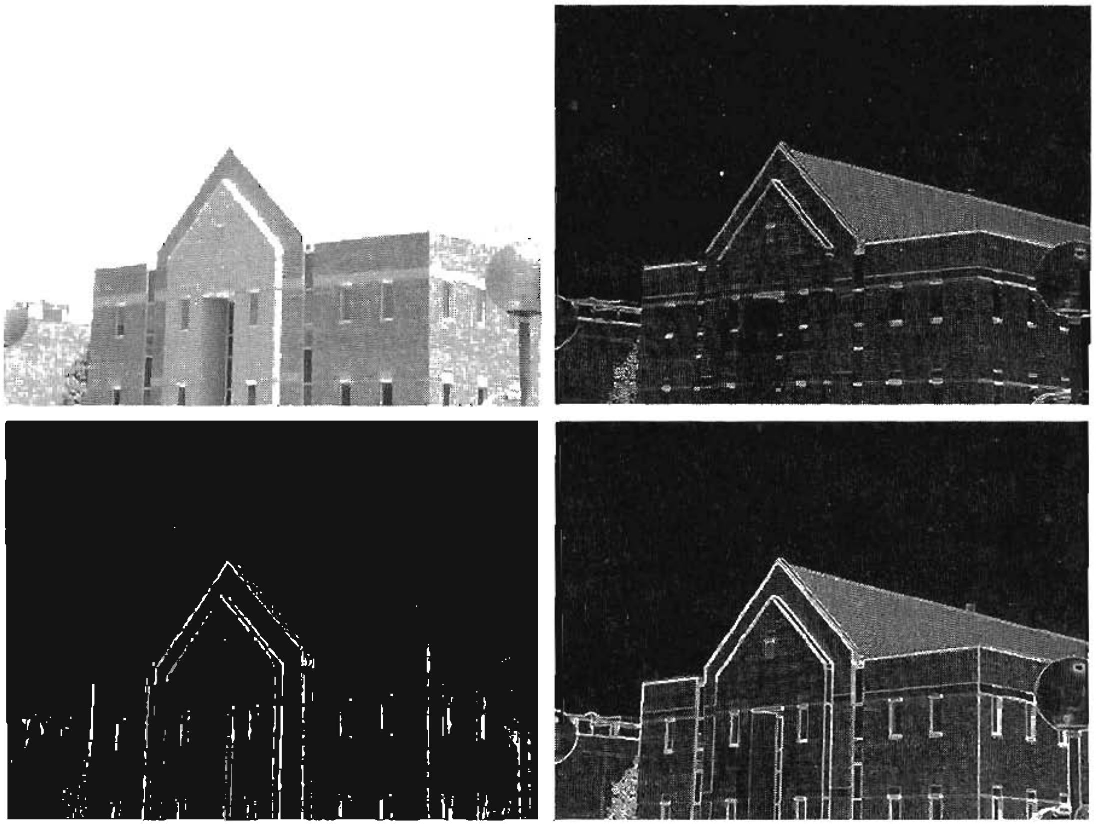

a b
c d

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

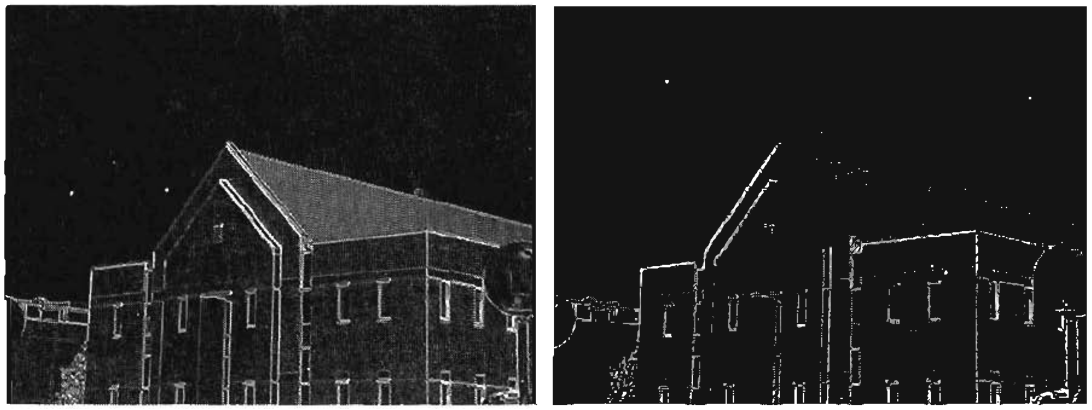

a b

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).

The Laplacian

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

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \qquad (10.1\text{-}13)$$

Digital approximations to the Laplacian were introduced in Section 3.7.2. For a 3 × 3 region, one of the two forms encountered most frequently in practice is

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8) \qquad (10.1\text{-}14)$$

FIGURE 10.13 Laplacian masks used to implement Eqs. (10.1-14) and (10.1-15), respectively.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

where the z's are defined in Fig. 10.8(a). A digital approximation including the diagonal neighbors is given by

$$\nabla^2 f = 8z_5 - (z_1 + z_2 + z_3 + z_4 + z_6 + z_7 + z_8 + z_9). \qquad (10.1\text{-}15)$$

Masks for implementing these two equations are shown in Fig. 10.13. We note from these masks that the implementations of Eqs. (10.1-14) and (10.1-15) are isotropic for rotation increments of 90° and 45°, respectively.

The Laplacian generally is not used in its original form for edge detection for several reasons: As a second-order derivative, the Laplacian typically is unacceptably sensitive to noise (Fig. 10.7). The magnitude of the Laplacian produces double edges (see Figs. 10.6 and 10.7), an undesirable effect because it complicates segmentation. Finally, the Laplacian is unable to detect edge direction. For these reasons, the role of the Laplacian in segmentation consists of (1) using its zero-crossing property for edge location, as mentioned earlier in this section, or (2) using it for the complementary purpose of establishing whether a pixel is on the dark or light side of an edge, as we show in Section 10.3.6.

In the first category, the Laplacian is combined with smoothing as a precursor to finding edges via zero-crossings. Consider the function

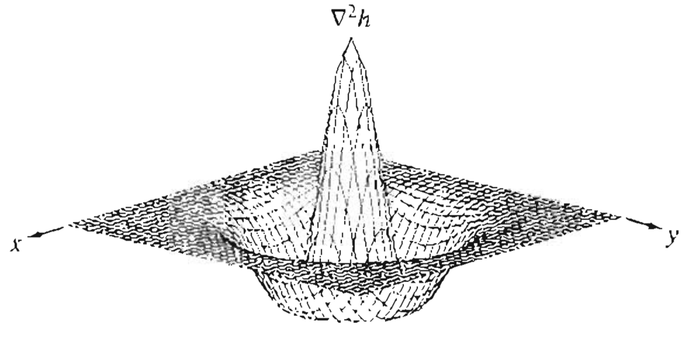
$$h(r) = -e^{-\frac{r^2}{2\sigma^2}} \qquad (10.1\text{-}16)$$

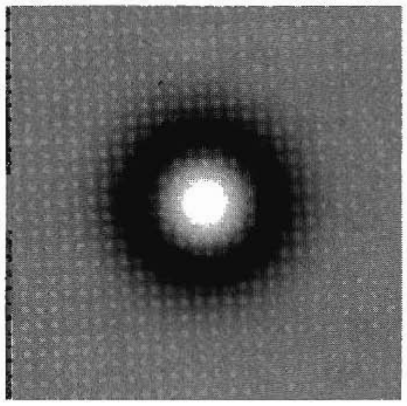
where $r^2 = x^2 + y^2$ and σ is the standard deviation. Convolving this function with an image blurs the image, with the degree of blurring being determined by the value of σ. The Laplacian of h (the second derivative of h with respect to r) is

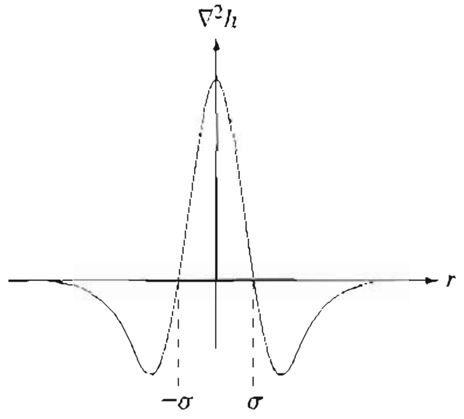
$$\nabla^2 h(r) = -\left[\frac{r^2 - \sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}}. \qquad (10.1\text{-}17)$$

This function is commonly referred to as the *Laplacian of a Gaussian* (LoG) because Eq. (10.1-16) is in the form of a Gaussian function. Figure 10.14 shows a 3-D plot, image, and cross section of the LoG function. Also shown is a 5×5 mask that approximates $\nabla^2 h$. This approximation is not unique. Its purpose is to capture the essential *shape* of $\nabla^2 h$; that is, a positive central term, surrounded by an adjacent negative region that increases in value as a function of distance from the origin, and a zero outer region. The coefficients also must sum to zero, so that the response of the mask is zero in areas of constant gray level. A mask this small is useful only for images that are essentially noise free. Due to its shape, the Laplacian of a Gaussian sometimes is called the *Mexican hat* function.

Because the second derivative is a linear operation, convolving an image with $\nabla^2 h$ is the same as convolving the image with the Gaussian smoothing function of Eq. (10.1-16) first and then computing the Laplacian of the result.







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

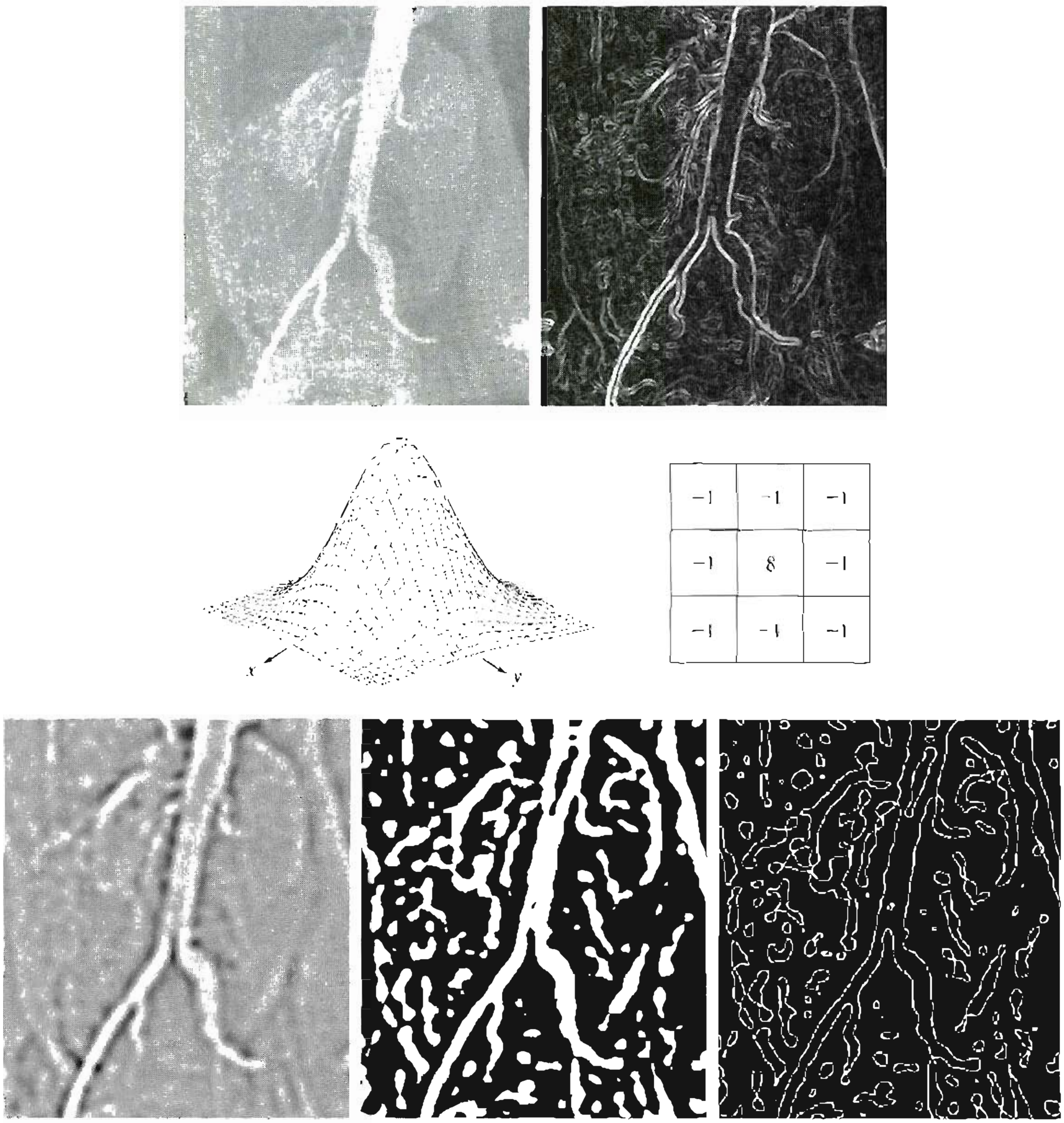
a b
c d

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).

Thus, we see that the purpose of the Gaussian function in the LoG formulation is to smooth the image, and the purpose of the Laplacian operator is to provide an image with zero crossings used to establish the location of edges. Smoothing the image reduces the effect of noise and, in principle, it counters the increased effect of noise caused by the second derivatives of the Laplacian. It is of interest to note that neurophysiological experiments carried out in the early 1980s (Ullman [1981], Marr [1982]) provide evidence that certain aspects of human vision can be modeled mathematically in the basic form of Eq. (10.1-17).

EXAMPLE 10.5: Edge finding by zero crossings.

■ Figure 10.15(a) shows the angiogram image discussed in Section 1.3.2. Figure 10.15(b) shows the Sobel gradient of this image, included here for comparison. Figure 10.15(c) is a spatial Gaussian function (with a standard deviation of five pixels) used to obtain a 27×27 spatial smoothing mask. The mask was obtained by sampling this Gaussian function at equal intervals. Figure 10.15(d) is the spatial mask used to implement Eq. (10.1-15). Figure 10.15(e) is the LoG image obtained by smoothing the original image with the Gaussian smoothing mask, followed by application of the Laplacian mask (this image was cropped to eliminate the border effects produced by the smoothing mask). As noted in the preceding paragraph, $\nabla^2 h$ can be computed by application of (c) followed by (d). Employing this procedure provides more control over the smoothing function, and often results in two masks that are much smaller when compared with a single composite mask that implements Eq. (10.1-17) directly. A composite mask usually is larger because it must incorporate the more complex shape shown in Fig. 10.14(a).



−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.)

The LoG result shown in Fig. 10.15(e) is the image from which zero crossings are computed to find edges. One straightforward approach for approximating zero crossings is to threshold the LoG image by setting all its positive values to, say, white, and all negative values to black. The result is shown in Fig. 10.15(f). The logic behind this approach is that zero crossings occur between positive and negative values of the Laplacian. Finally, Fig. 10.15(g) shows the estimated zero crossings, obtained by scanning the thresholded image and noting the transitions between black and white.

Comparing Figs. 10.15(b) and (g) reveals several interesting and important differences. First, we note that the edges in the zero-crossing image are thinner than the gradient edges. This is a characteristic of zero crossings that makes this approach attractive. On the other hand, we see in Fig. 10.15(g) that the edges determined by zero crossings form numerous closed loops. This so-called spaghetti effect is one of the most serious drawbacks of this method. Another major drawback is the computation of zero crossings, which is the foundation of the method. Although it was reasonably straightforward in this example, the computation of zero crossings presents a challenge in general, and considerably more sophisticated techniques often are required to obtain acceptable results (Huertas and Medione [1986]).

Zero-crossing methods are of interest because of their noise reduction capabilities and potential for rugged performance. However, the limitations just noted present a significant barrier in practical applications. For this reason, edge-finding techniques based on various implementations of the gradient still are used more frequently than zero crossings in the implementation of segmentation algorithms. ■

10.2 Edge Linking and Boundary Detection

Ideally, the methods discussed in the previous section should yield pixels lying only on edges. In practice, this set of pixels seldom characterizes an edge completely because of noise, breaks in the edge from nonuniform illumination, and other effects that introduce spurious intensity discontinuities. Thus edge detection algorithms typically are followed by linking procedures to assemble edge pixels into meaningful edges. Several basic approaches are suited to this purpose.

10.2.1 Local Processing

One of the simplest approaches for linking edge points is to analyze the characteristics of pixels in a small neighborhood (say, 3×3 or 5×5) about every point (x, y) in an image that has been labeled an edge point by one of the techniques discussed in the previous section. All points that are similar according to a set of predefined criteria are linked, forming an edge of pixels that share those criteria.

The two principal properties used for establishing similarity of edge pixels in this kind of analysis are (1) the strength of the response of the gradient operator used to produce the edge pixel; and (2) the direction of the gradient vector. The first property is given by the value of ∇f, as defined in Eq. (10.1-4) or (10.1-12). Thus an edge pixel with coordinates (x_0, y_0) in a predefined neighborhood of

(x, y), is similar in magnitude to the pixel at (x, y) if

$$|\nabla f(x, y) - \nabla f(x_0, y_0)| \leq E \tag{10.2-1}$$

where E is a nonnegative threshold.

The direction (angle) of the gradient vector is given by Eq. (10.1-5). An edge pixel at (x_0, y_0) in the predefined neighborhood of (x, y) has an angle similar to the pixel at (x, y) if

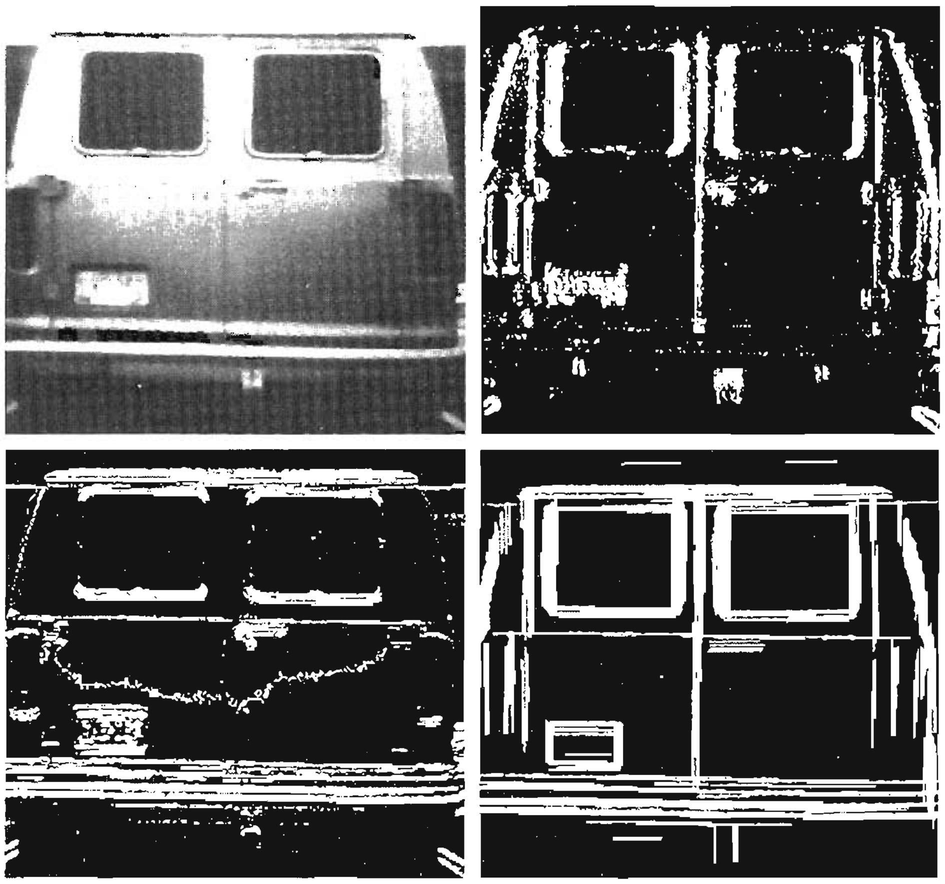
$$|\alpha(x, y) - \alpha(x_0, y_0)| < A \tag{10.2-2}$$

where A is a nonnegative angle threshold. As noted in Eq. (10.1-5), the direction of the edge at (x, y) is *perpendicular* to the direction of the gradient vector at that point.

A point in the predefined neighborhood of (x, y) is linked to the pixel at (x, y) if both magnitude and direction criteria are satisfied. This process is repeated at every location in the image. A record must be kept of linked points as the center of the neighborhood is moved from pixel to pixel. A simple bookkeeping procedure is to assign a different gray level to each set of linked edge pixels.

EXAMPLE 10.6: Edge-point linking based on local processing.

■ To illustrate the foregoing procedure, consider Fig. 10.16(a), which shows an image of the rear of a vehicle. The objective is to find rectangles whose sizes makes them suitable candidates for license plates. The formation of these rectangles can be accomplished by detecting strong horizontal and vertical edges. Figures 10.16(b) and (c) show vertical and horizontal edges obtained by using



a b
c d

FIGURE 10.16 (a) Input image. (b) G_y component of the gradient. (c) G_x component of the gradient. (d) Result of edge linking. (Courtesy of Perceptics Corporation.)

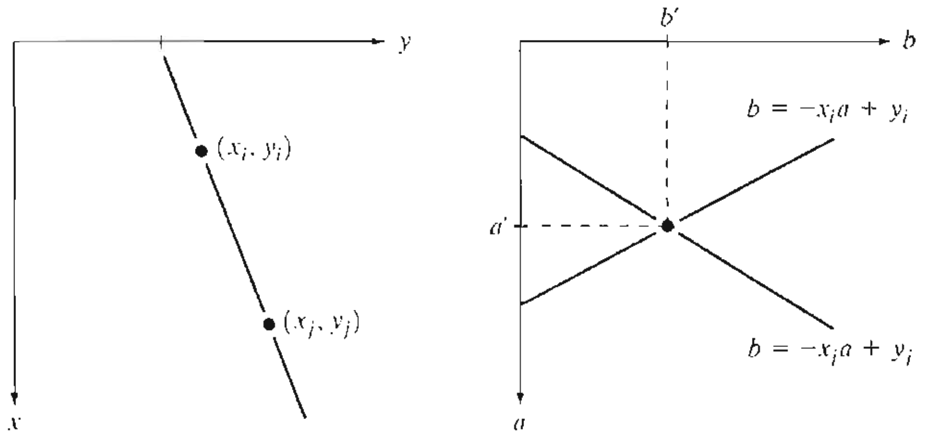
the horizontal and vertical Sobel operators. Figure 10.16(d) shows the result of linking all points that simultaneously had a gradient value greater than 25 and whose gradient directions did not differ by more than 15°. The horizontal lines were formed by sequentially applying these criteria to every row of Fig. 10.16(c). A sequential column scan of Fig. 10.16(b) yielded the vertical lines. Further processing consisted of linking edge segments separated by small breaks and deleting isolated short segments. As Fig. 10.16(d) shows, the rectangle corresponding to the license plate was one of the few rectangles detected in the image. It would be a simple matter to locate the license plate based on these rectangles (the width-to-height ratio of the license plate rectangle has a distinctive 2:1 proportion for U.S. plates).

10.2.2 Global Processing via the Hough Transform

In this section, points are linked by determining first if they lie on a curve of specified shape. Unlike the local analysis method discussed in Section 10.2.1, we now consider global relationships between pixels.

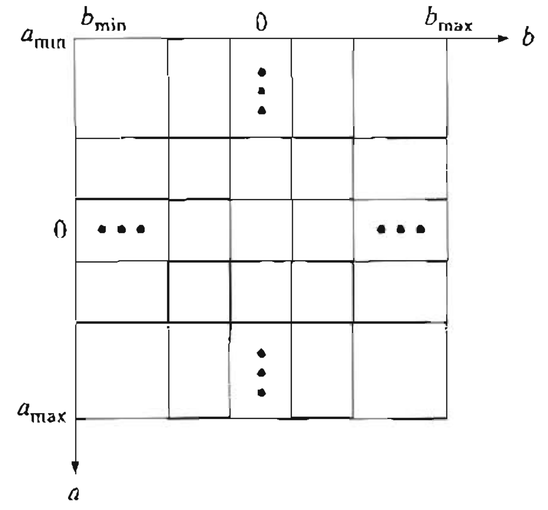
Given n points in an image, suppose that we want to find subsets of these points that lie on straight lines. One possible solution is to first find all lines determined by every pair of points and then find all subsets of points that are close to particular lines. The problem with this procedure is that it involves finding $n(n-1)/2 \sim n^2$ lines and then performing $(n)(n(n-1))/2 \sim n^3$ comparisons of every point to all lines. This approach is computationally prohibitive in all but the most trivial applications.

Hough [1962] proposed an alternative approach, commonly referred to as the *Hough transform*. Consider a point (x_i, y_i) and the general equation of a straight line in slope-intercept form, $y_i = ax_i + b$. Infinitely many lines pass through (x_i, y_i), but they all satisfy the equation $y_i = ax_i + b$ for varying values of a and b. However, writing this equation as $b = -x_i a + y_i$ and considering the ab-plane (also called *parameter space*) yields the equation of a *single* line for a fixed pair (x_i, y_i). Furthermore, a second point (x_j, y_j) also has a line in parameter space associated with it, and this line intersects the line associated with (x_i, y_i) at (a', b'), where a' is the slope and b' the intercept of the line containing both (x_i, y_i) and (x_j, y_j) in the xy-plane. In fact, all points contained on this line have lines in parameter space that intersect at (a', b'). Figure 10.17 illustrates these concepts.



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

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



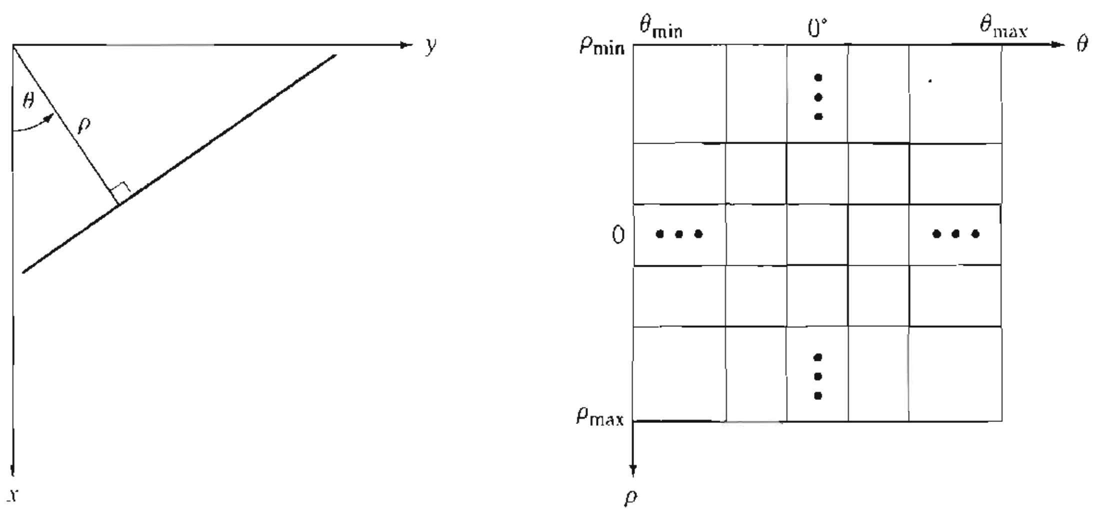
The computational attractiveness of the Hough transform arises from subdividing the parameter space into so-called *accumulator cells*, as illustrated in Fig. 10.18, where $(a_{\max}, a_{\min})$ and $(b_{\max}, b_{\min})$ are the expected ranges of slope and intercept values. The cell at coordinates (i, j), with accumulator value $A(i, j)$, corresponds to the square associated with parameter space coordinates (a_i, b_j). Initially, these cells are set to zero. Then, for every point (x_k, y_k) in the image plane, we let the parameter a equal each of the allowed subdivision values on the a-axis and solve for the corresponding b using the equation $b = -x_k a + y_k$. The resulting b's are then rounded off to the nearest allowed value in the b-axis. If a choice of a_p results in solution b_q, we let $A(p, q) = A(p, q) + 1$. At the end of this procedure, a value of Q in $A(i, j)$ corresponds to Q points in the xy-plane lying on the line $y = a_i x + b_j$. The number of subdivisions in the ab-plane determines the accuracy of the colinearity of these points.

Note that subdividing the a axis into K increments gives, for every point (x_k, y_k), K values of b corresponding to the K possible values of a. With n image points, this method involves nK computations. Thus the procedure just discussed is *linear* in n, and the product nK does not approach the number of computations discussed at the beginning of this section unless K approaches or exceeds n.

A problem with using the equation $y = ax + b$ to represent a line is that the slope approaches infinity as the line approaches the vertical. One way around this difficulty is to use the normal representation of a line:

$$x \cos\theta + y \sin\theta = \rho. \tag{10.2-3}$$

Figure 10.19(a) illustrates the geometrical interpretation of the parameters used in Eq. (10.2-3). The use of this representation in constructing a table of accumulators is identical to the method discussed for the slope-intercept representation. Instead of straight lines, however, the loci are sinusoidal curves in the $\rho\theta$-plane. As before, Q collinear points lying on a line $x \cos\theta_j + y \sin\theta_j = \rho_i$ yield Q sinusoidal curves that intersect at (ρ_i, θ_j) in the parameter space. Incrementing θ and solving for the corresponding ρ gives Q entries in accumulator $A(i, j)$ associated with the cell determined by (ρ_i, θ_j). Figure 10.19(b) illustrates the subdivision of the parameter space.



a b

FIGURE 10.19 (a) Normal representation of a line. (b) Subdivision of the $\rho\theta$-plane into cells.

The range of angle θ is $\pm 90°$, measured with respect to the x-axis. Thus with reference to Fig. 10.19(a), a horizontal line has $\theta = 0°$, with ρ being equal to the positive x-intercept. Similarly, a vertical line has $\theta = 90°$, with ρ being equal to the positive y-intercept, or $\theta = -90°$, with ρ being equal to the negative y-intercept.

EXAMPLE 10.7: Illustration of the Hough transform.

■ Figure 10.20 illustrates the Hough transform based on Eq. (10.2-3). Figure 10.20(a) shows an image with five labeled points. Each of these points is mapped onto the $\rho\theta$-plane, as shown in Fig. 10.20(b). The range of θ values is $\pm 90°$, and the range of the ρ-axis is $\pm\sqrt{2}D$, where D is the distance between corners in the image. Unlike the transform based on using the slope intercept, each of these curves has a different sinusoidal shape. The horizontal line resulting from the mapping of point 1 is a special case of a sinusoid with zero amplitude.

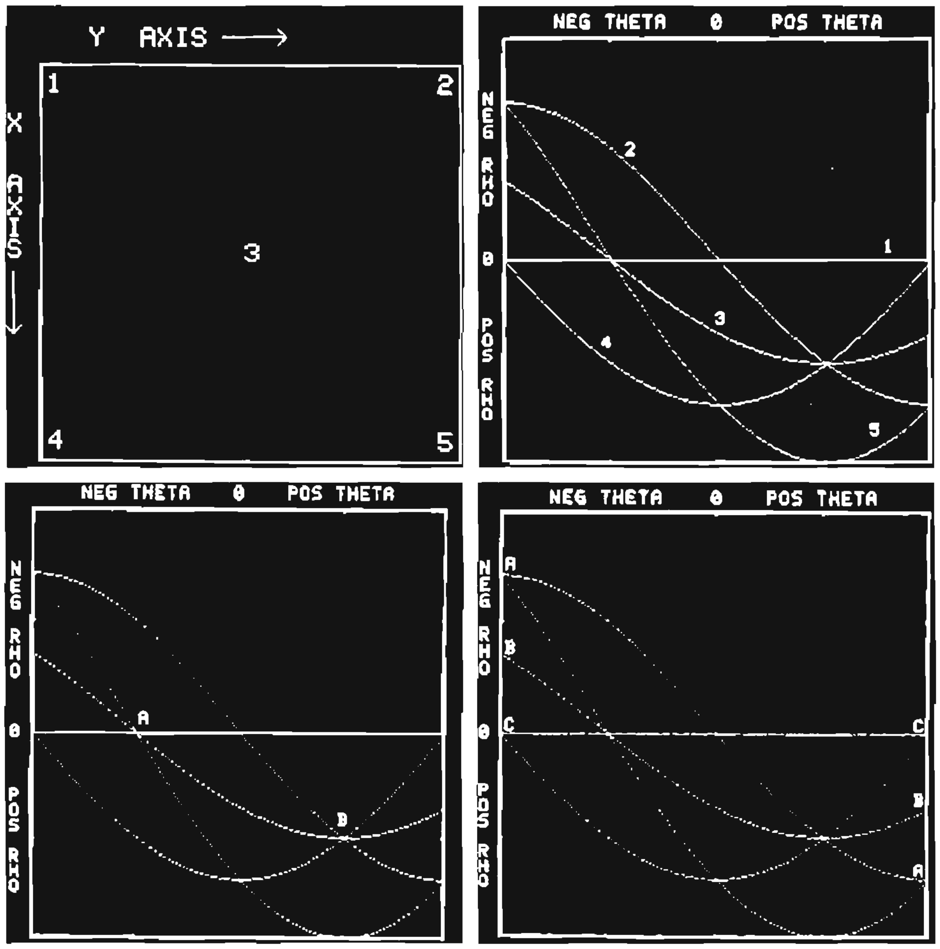
The colinearity detection property of the Hough transform is illustrated in Fig. 10.20(c). Point A (not to be confused with accumulator values) denotes the intersection of the curves corresponding to points 1, 3, and 5 in the xy-image plane. The location of point A indicates that these three points lie on a straight line passing through the origin ($\rho = 0$) and oriented at $-45°$. Similarly, the curves intersecting at point B in the parameter space indicate that points 2, 3, and 4 lie on a straight line oriented at 45° and whose distance from the origin is one-half the diagonal distance from the origin of the image to the opposite corner.

Finally, Fig. 10.20(d) indicates the fact that the Hough transform exhibits a reflective adjacency relationship at the right and left edges of the parameter space. This property, shown by the points marked A, B, and C in Fig. 10.20(d), is the result of the manner in which θ and ρ change sign at the $\pm 90°$ boundaries. ■

Although the focus so far has been on straight lines, the Hough transform is applicable to any function of the form $g(\mathbf{v}, \mathbf{c}) = 0$, where $\mathbf{v}$ is a vector of coordinates and $\mathbf{c}$ is a vector of coefficients. For example, the points lying on the circle

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2 \tag{10.2-4}$$

can be detected by using the approach just discussed. The basic difference is the presence of three parameters (c_1, c_2, and c_3), which results in a 3-D parameter



a b
c d

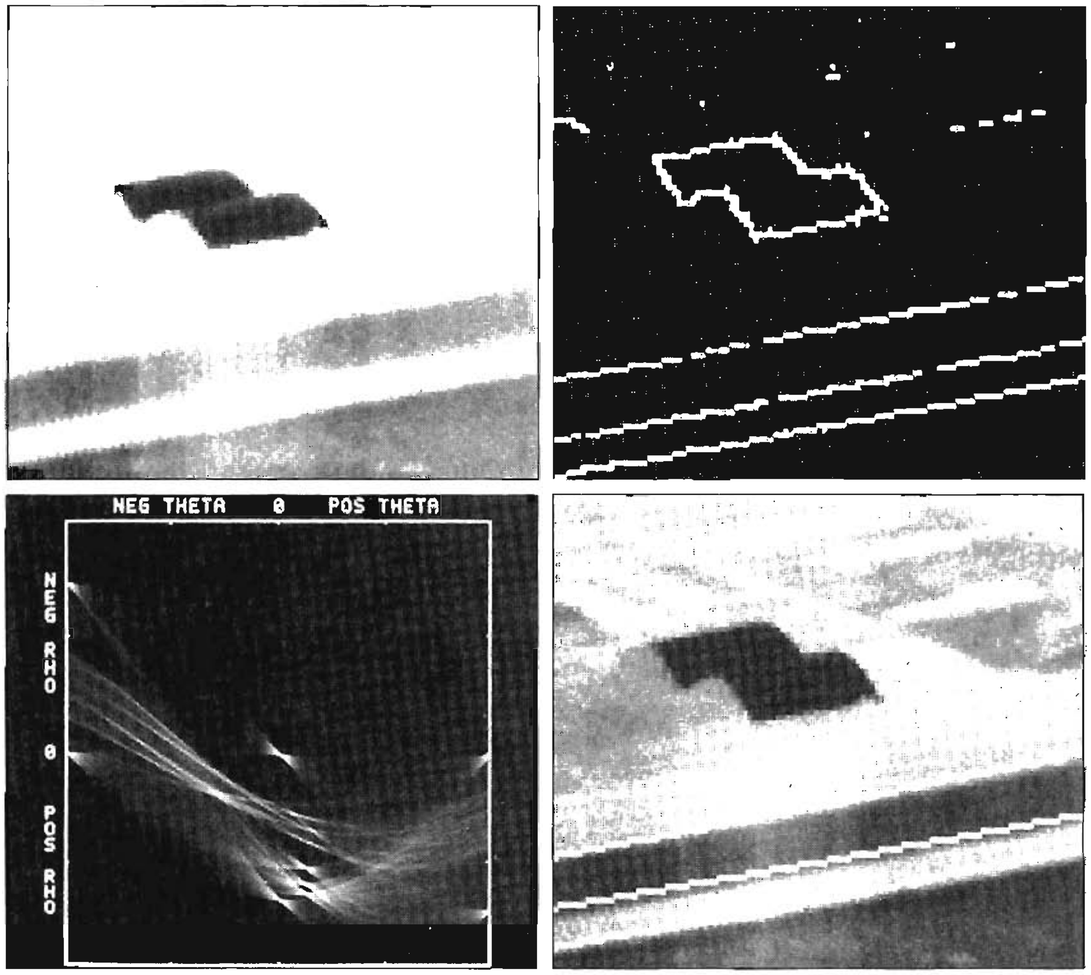
FIGURE 10.20 Illustration of the Hough transform. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

space with cubelike cells and accumulators of the form $A(i, j, k)$. The procedure is to increment c_1 and c_2, solve for the c_3 that satisfies Eq. (10.2-4), and update the accumulator corresponding to the cell associated with the triplet (c_1, c_2, c_3). Clearly, the complexity of the Hough transform is proportional to the number of coordinates and coefficients in a given functional representation. Further generalizations of the Hough transform to detect curves with no simple analytic representations are possible, as is the application of the transform to gray-scale images. Several references dealing with these extensions are included at the end of this chapter.

We now return to the edge-linking problem. An approach based on the Hough transform is as follows:

1. Compute the gradient of an image and threshold it to obtain a binary image.
2. Specify subdivisions in the $\rho\theta$-plane.
3. Examine the counts of the accumulator cells for high pixel concentrations.
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.

The concept of continuity in this case usually is based on computing the distance between disconnected pixels identified during traversal of the set of pixels corresponding to a given accumulator cell. A gap at any point is significant if the distance



a b
c d

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.)

between that point and its closest neighbor exceeds a certain threshold. (See Section 2.5 for a discussion of connectivity, neighborhoods, and distance measures.)

EAMPLE 10.8: Using the Hough transform for edge linking.

■ Figure 10.21(a) shows an aerial infrared image containing two hangars and a runway. Figure 10.21(b) is a thresholded gradient image obtained using the Sobel operators discussed in Section 10.1.3 (note the small gaps in the borders of the runway). Figure 10.21(c) shows the Hough transform of the gradient image. Figure 10.21(d) shows (in white) the set of pixels linked according to the criteria that (1) they belonged to one of the three accumulator cells with the highest count, and (2) no gaps were longer than five pixels. Note the disappearance of the gaps as a result of linking. ■

10.2.3 Global Processing via Graph-Theoretic Techniques

In this section we discuss a global approach for edge detection and linking based on representing edge segments in the form of a graph and searching the graph for low-cost paths that correspond to significant edges. This representation provides a rugged approach that performs well in the presence of noise. As might be expected, the procedure is considerably more complicated and requires more processing time than the methods discussed so far.

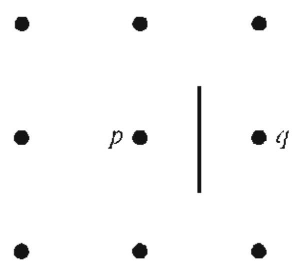


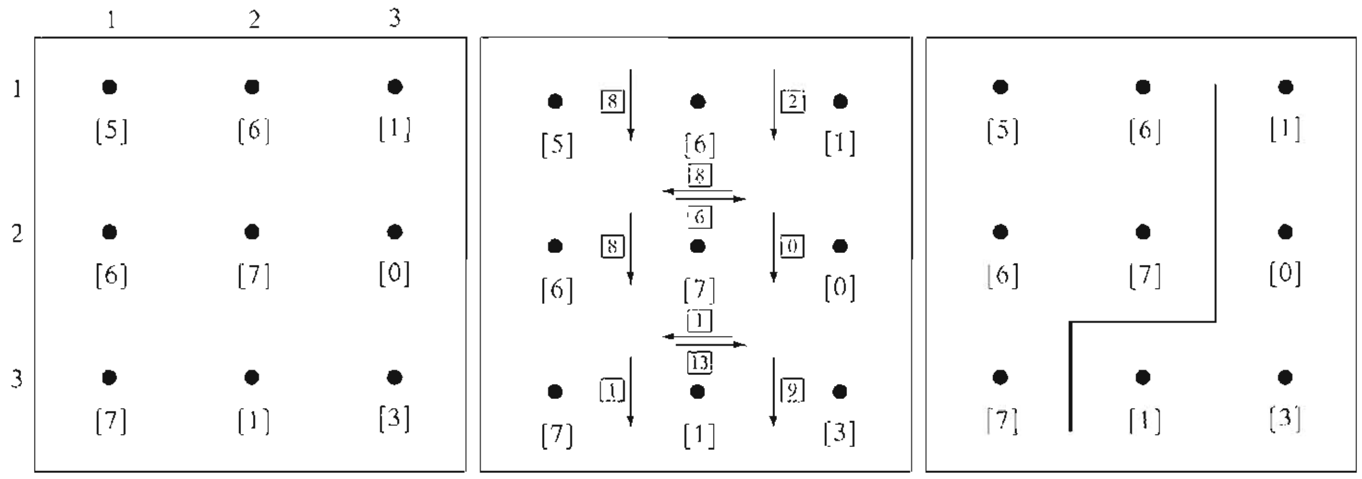
FIGURE 10.22 Edge element between pixels p and q.

We begin the development with some basic definitions. A *graph* $G = (N, U)$ is a finite, nonempty set of nodes N, together with a set U of unordered pairs of distinct elements of N. Each pair (n_i, n_j) of U is called an *arc*. A graph in which the arcs are directed is called a *directed graph*. If an arc is directed from node n_i to node n_j, then n_j is said to be a *successor* of the *parent* node n_i. The process of identifying the successors of a node is called *expansion* of the node. In each graph we define levels, such that level 0 consists of a single node, called the *start* or *root* node, and the nodes in the last level are called *goal* nodes. A *cost* $c(n_i, n_j)$ can be associated with every arc (n_i, n_j). A sequence of nodes $n_1, n_2, \dots, n_k$, with each node n_i being a successor of node n_{i-1}, is called a *path* from n_1 to n_k. The cost of the entire path is

$$c = \sum_{i=2}^{k} c(n_{i-1}, n_i). \qquad (10.2\text{-}5)$$

The following discussion is simplified if we define an *edge element* as the boundary between two pixels p and q, such that p and q are 4-neighbors, as Fig. 10.22 illustrates. Edge elements are identified by the xy-coordinates of points p and q. In other words, the edge element in Fig. 10.22 is defined by the pairs $(x_p, y_p)(x_q, y_q)$. Consistent with the definition given in Section 10.1.3, an *edge* is a sequence of connected edge elements.

We can illustrate how the concepts just discussed apply to edge detection using the 3×3 image shown in Fig. 10.23(a). The outer numbers are pixel



a b c

FIGURE 10.23 (a) A 3×3 image region. (b) Edge segments and their costs. (c) Edge corresponding to the lowest-cost path in the graph shown in Fig. 10.24.

coordinates and the numbers in brackets represent gray-level values. Each edge element, defined by pixels p and q, has an associated cost, defined as

$$c(p,q) = H - [f(p) - f(q)] \tag{10.2-6}$$

where H is the highest gray-level value in the image (7 in this case), and $f(p)$ and $f(q)$ are the gray-level values of p and q, respectively. By convention, the point p is on the right-hand side of the direction of travel along edge elements. For example, the edge segment $(1,2)(2,2)$ is between points $(1,2)$ and $(2,2)$ in Fig. 10.23(b). If the direction of travel is to the right, then p is the point with coordinates $(2,2)$ and q is point with coordinates $(1,2)$; therefore, $c(p,q) = 7 - [7 - 6] = 6$. This cost is shown in the box below the edge segment. If, on the other hand, we are traveling to the *left* between the same two points, then p is point $(1,2)$ and q is $(2,2)$. In this case the cost is 8, as shown above the edge segment in Fig. 10.23(b). To simplify the discussion, we assume that edges start in the top row and terminate in the last row, so that the first element of an edge can be only between points $(1,1)$, $(1,2)$ or $(1,2)$, $(1,3)$. Similarly, the last edge element has to be between points $(3,1)$, $(3,2)$ or $(3,2)$, $(3,3)$. Keep in mind that p and q are 4-neighbors, as noted earlier.

Figure 10.24 shows the graph for this problem. Each node (rectangle) in the graph corresponds to an edge element from Fig. 10.23. An arc exists between two nodes if the two corresponding edge elements taken in succession can be part

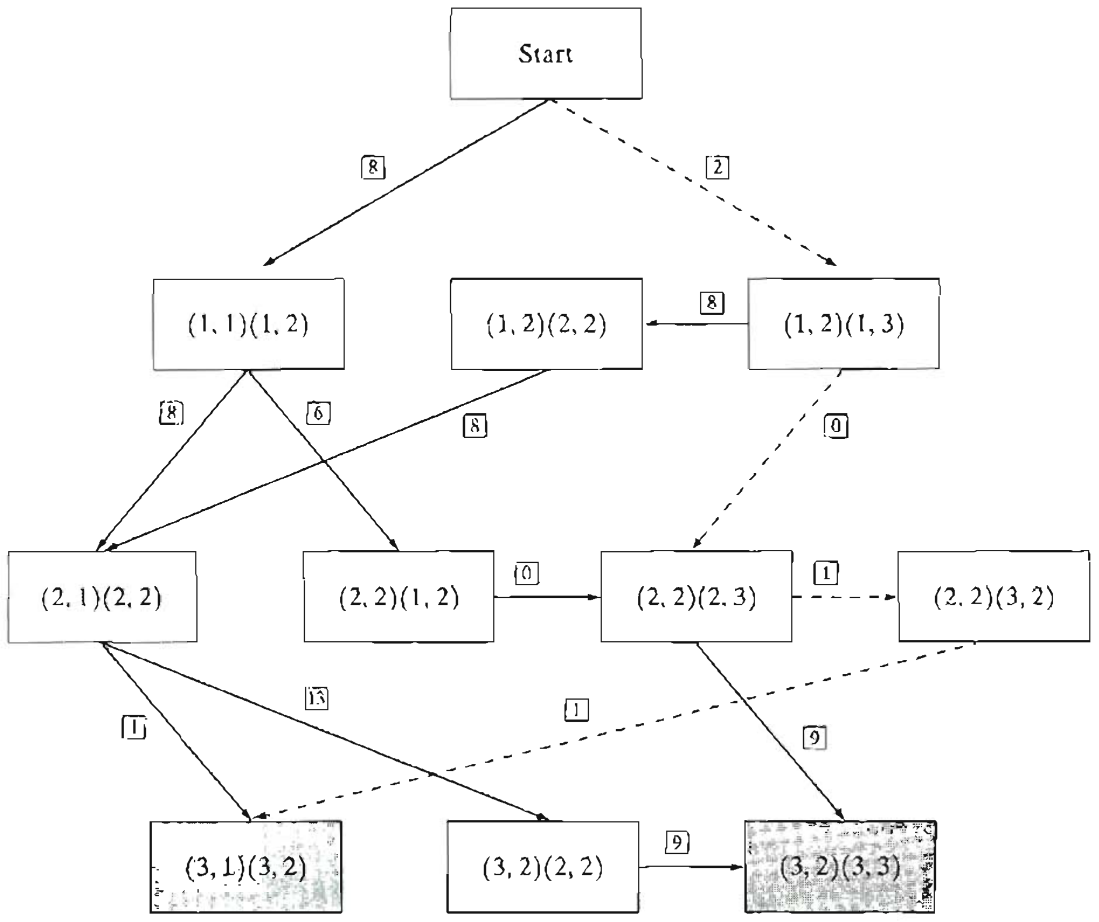


FIGURE 10.24 Graph for the image in Fig. 10.23(a). The lowest-cost path is shown dashed.

of an edge. As in Fig. 10.23(b), the cost of each edge segment, computed using Eq. (10.2-6), is shown in a box on the side of the arc leading into the corresponding node. Goal nodes are shown shaded. The minimum cost path is shown dashed, and the edge corresponding to this path is shown in Fig. 10.23(c).

In general, the problem of finding a minimum-cost path is not trivial in terms of computation. Typically, the approach is to sacrifice optimality for the sake of speed, and the following algorithm represents a class of procedures that use heuristics in order to reduce the search effort. Let $r(n)$ be an estimate of the cost of a minimum-cost path from the start node s to a goal node, where the path is constrained to go through n. This cost can be expressed as the estimate of the cost of a minimum-cost path from s to n plus an estimate of the cost of that path from n to a goal node; that is,

$$r(n) = g(n) + h(n). \tag{10.2-7}$$

Here, $g(n)$ can be chosen as the lowest-cost path from s to n found so far, and $h(n)$ is obtained by using any available heuristic information (such as expanding only certain nodes based on previous costs in getting to that node). An algorithm that uses $r(n)$ as the basis for performing a graph search is as follows:

Step 1: Mark the start node OPEN and set $g(s) = 0$.
Step 2: If no node is OPEN exit with failure; otherwise, continue.
Step 3: Mark CLOSED the OPEN node n whose estimate $r(n)$ computed from Eq. (10.2-7) is smallest. (Ties for minimum r values are resolved arbitrarily, but always in favor of a goal node.)
Step 4: If n is a goal node, exit with the solution path obtained by tracing back through the pointers; otherwise, continue.
Step 5: Expand node n, generating all of its successors. (If there are no successors go to step 2.)
Step 6: If a successor n_i is not marked, set

$$r(n_i) = g(n) + c(n, n_i),$$

mark it OPEN, and direct pointers from it back to n.
Step 7: If a successor n_i is marked CLOSED or OPEN, update its value by letting

$$g'(n_i) = \min[g(n_i), g(n) + c(n, n_i)].$$

Mark OPEN those CLOSED successors whose g' values were thus lowered and redirect to n the pointers from all nodes whose g' values were lowered. Go to step 2.

This algorithm does not guarantee a minimum-cost path; its advantage is speed via the use of heuristics. However, if $h(n)$ is a lower bound on the cost of the minimal-cost path from node n to a goal node, the procedure indeed yields an optimal path to a goal (Hart et al. [1968]). If no heuristic information is available (that is, $h \equiv 0$), the procedure reduces to the *uniform-cost algorithm* of Dijkstra [1959].

EXAMPLE 10.9: Edge finding by graph search.

■ Figure 10.25 shows an image of a noisy chromosome silhouette and an edge found using a heuristic graph search based on the algorithm developed in this

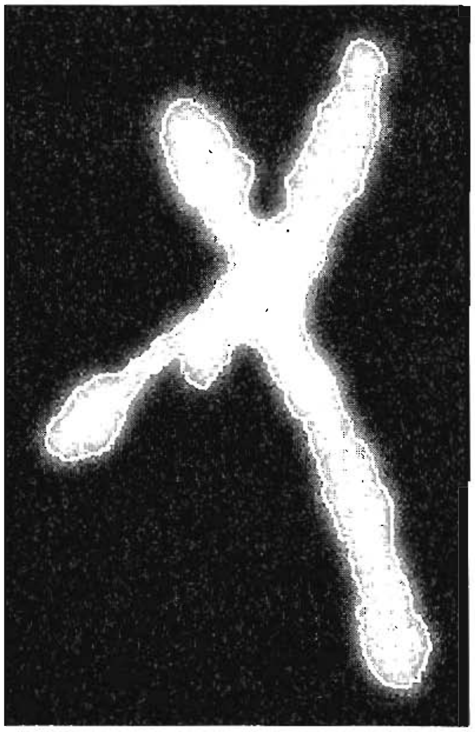


FIGURE 10.25 Image of noisy chromosome silhouette and edge boundary (in white) determined by graph search.

section. The edge is shown in white, superimposed on the original image. Note that in this case the edge and the boundary of the object are approximately the same. The cost was based on Eq. (10.2-6), and the heuristic used at any point on the graph was to determine and use the optimum path for five levels down from that point. Considering the amount of noise present in this image, the graph-search approach yielded a reasonably accurate result. ■

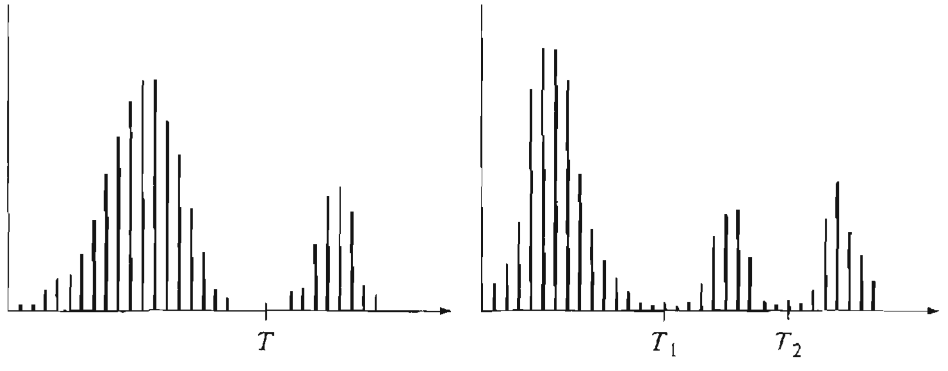
10.3 Thresholding

Because of its intuitive properties and simplicity of implementation, image thresholding enjoys a central position in applications of image segmentation. Simple thresholding was first introduced in Section 3.1, and we have used it in various discussions in the preceding chapters. In this section, we introduce thresholding in a more formal way and extend it to techniques that are considerably more general than what has been presented thus far.

10.3.1 Foundation

Suppose that the gray-level histogram shown in Fig. 10.26(a) corresponds to an image, $f(x, y)$, composed of light objects on a dark background, in such a way that object and background pixels have gray levels grouped into two dominant modes. One obvious way to extract the objects from the background is to select a threshold T that separates these modes. Then any point (x, y) for which $f(x, y) > T$ is called an *object point*; otherwise, the point is called a *background point*. This is the type of thresholding introduced in Section 3.1.

Figure 10.26(b) shows a slightly more general case of this approach, where three dominant modes characterize the image histogram (for example, two types



a b

FIGURE 10.26 (a) Gray-level histograms that can be partitioned by (a) a single threshold, and (b) multiple thresholds.

of light objects on a dark background). Here, *multilevel thresholding* classifies a point (x, y) as belonging to one object class if $T_1 < (x, y) \leq T_2$, to the other object class if $f(x, y) > T_2$, and to the background if $f(x, y) \leq T_1$. In general, segmentation problems requiring multiple thresholds are best solved using region growing methods, such as those discussed in Section 10.4.

Based on the preceding discussion, thresholding may be viewed as an operation that involves tests against a function T of the form

$$T = T[x, y, p(x, y), f(x, y)] \tag{10.3-1}$$

where $f(x, y)$ is the gray level of point (x, y) and $p(x, y)$ denotes some local property of this point—for example, the average gray level of a neighborhood centered on (x, y). A thresholded image $g(x, y)$ is defined as

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T. \end{cases} \tag{10.3-2}$$

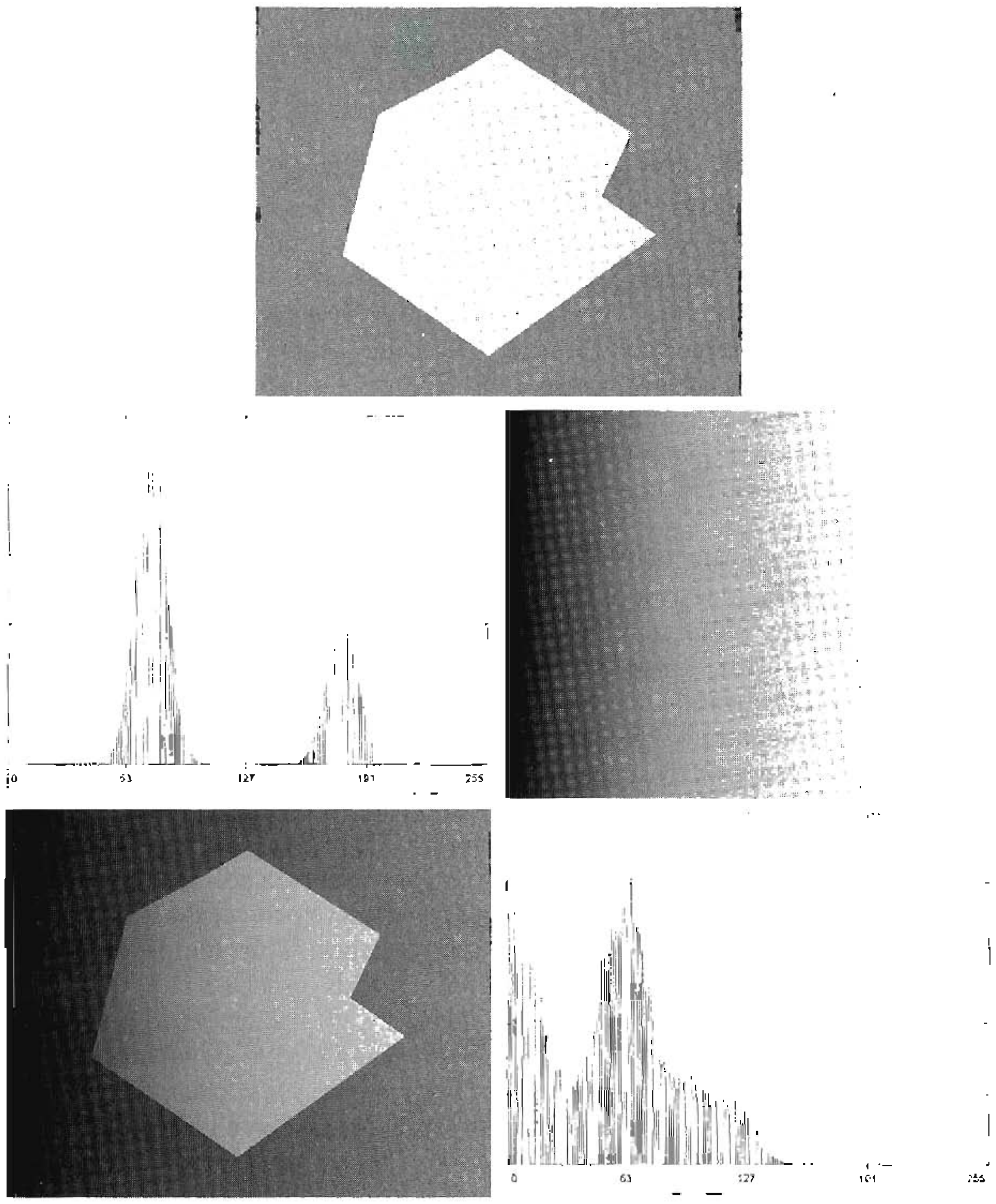
Thus, pixels labeled 1 (or any other convenient gray level) correspond to objects, whereas pixels labeled 0 (or any other gray level not assigned to objects) correspond to the background.

When T depends only on $f(x, y)$ (that is, only on gray-level values) the threshold is called *global*. If T depends on both $f(x, y)$ and $p(x, y)$, the threshold is called *local*. If, in addition, T depends on the spatial coordinates x and y, the threshold is called *dynamic* or *adaptive*.

10.3.2 The Role of Illumination

In Section 2.3.4 we introduced a simple model in which an image $f(x, y)$ is formed as the product of a reflectance component $r(x, y)$ and an illumination component $i(x, y)$. The purpose of this section is to use this model to discuss briefly the effect of illumination on thresholding, especially on global thresholding.

Consider the computer generated reflectance function shown in Fig. 10.27(a). The histogram of this function, shown in Fig. 10.27(b), is clearly bimodal and could be partitioned easily by placing a single global threshold, T, in the histogram



a
b c
d e

FIGURE 10.27 (a) Computer generated reflectance function. (b) Histogram of reflectance function. (c) Computer generated illumination function. (d) Product of (a) and (c). (e) Histogram of product image.

valley. Multiplying the reflectance function in Fig. 10.27(a) by the illumination function shown in Fig. 10.27(c) yields the image shown in Fig. 10.27(d). Figure 10.27(e) shows the histogram of this image. Note that the original valley was virtually eliminated, making segmentation by a single threshold an impossible task. Although we seldom have the reflectance function by itself to work with, this simple illustration shows that the reflective nature of objects and background could be such that they are easily separable. However, the image resulting from poor (in this case nonuniform) illumination could be quite difficult to segment.

The reason why the histogram in Fig. 10.27(e) is so distorted can be explained with aid of the discussion in Section 4.5. From Eq. (4.5-1),

$$f(x, y) = i(x, y)r(x, y). \qquad (10.3\text{-}3)$$

Taking the natural logarithm of this equation yields a sum:

$$\begin{aligned} z(x, y) &= \ln f(x, y) \\ &= \ln i(x, y) + \ln r(x, y) \\ &= i'(x, y) + r'(x, y). \end{aligned} \tag{10.3-4}$$

From probability theory (Papoulis [1991]), if $i'(x, y)$ and $r'(x, y)$ are independent random variables, the histogram of $z(x, y)$ is given by the convolution of the histograms of $i'(x, y)$ and $r'(x, y)$. If $i(x, y)$ were constant, $i'(x, y)$ would be constant also, and its histogram would be a simple spike (like an impulse). The convolution of this impulselike function with the histogram of $r'(x, y)$ would leave the basic shape of this histogram unchanged (recall from the discussion in Section 4.2.4 that convolution of a function with an impulse copies the function at the location of the impulse). But if $i'(x, y)$ had a broader histogram (resulting from nonuniform illumination), the convolution process would smear the histogram of $r'(x, y)$, yielding a histogram for $z(x, y)$ whose shape could be quite different from that of the histogram of $r'(x, y)$. The degree of distortion depends on the broadness of the histogram of $i'(x, y)$, which in turn depends on the nonuniformity of the illumination function.

We have dealt with the logarithm of $f(x, y)$, instead of dealing with the image function directly, but the essence of the problem is clearly explained by using the logarithm to separate the illumination and reflectance components. This approach allows histogram formation to be viewed as a convolution process, thus explaining why a distinct valley in the histogram of the reflectance function could be smeared by improper illumination.

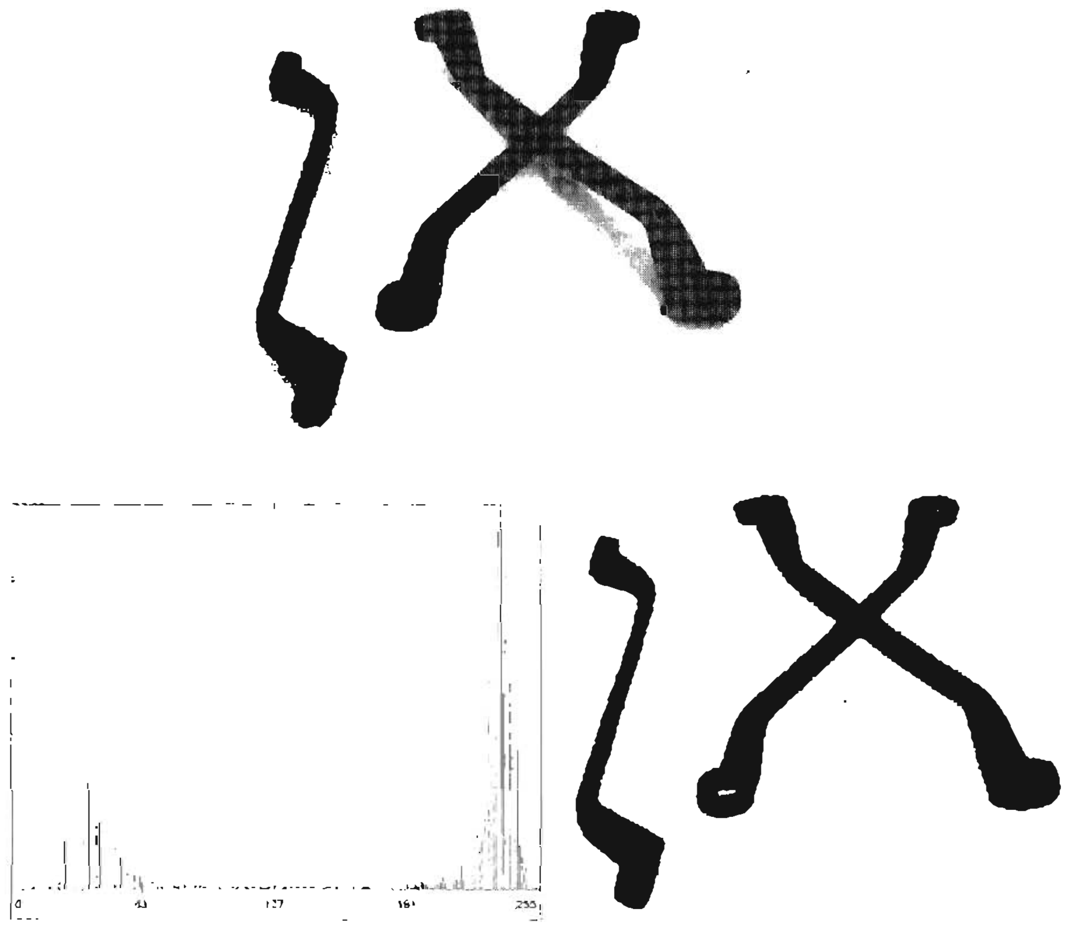
When access to the illumination source is available, a solution frequently used in practice to compensate for nonuniformity is to project the illumination pattern onto a constant, white reflective surface. This yields an image $g(x, y) = ki(x, y)$, where k is a constant that depends on the surface and $i(x, y)$ is the illumination pattern. Then, for any image $f(x, y) = i(x, y)r(x, y)$ obtained with the same illumination function, simply dividing $f(x, y)$ by $g(x, y)$ yields a normalized function $h(x, y) = f(x, y)/g(x, y) = r(x, y)/k$. Thus, if $r(x, y)$ can be segmented by using a single threshold T, then $h(x, y)$ can be segmented by using a single threshold of value T/k.

10.3.3 Basic Global Thresholding

With reference to the discussion in Section 10.3.1, the simplest of all thresholding techniques is to partition the image histogram by using a single global threshold, T, as illustrated in Fig. 10.26(a). Segmentation is then accomplished by scanning the image pixel by pixel and labeling each pixel as object or background, depending on whether the gray level of that pixel is greater or less than the value of T. As indicated earlier, the success of this method depends entirely on how well the histogram can be partitioned.

EXAMPLE 10.10: Global thresholding.

■ Figure 10.28(a) shows a simple image, and Fig. 10.28(b) shows its histogram. Figure 10.28(c) shows the result of segmenting Fig. 10.28(a) by using a threshold T midway between the maximum and minimum gray levels. This threshold



a
b c

FIGURE 10.28 (a) Original image. (b) Image histogram. (c) Result of global thresholding with T midway between the maximum and minimum gray levels.

achieved a "clean" segmentation by eliminating the shadows and leaving only the objects themselves. The objects of interest in this case are darker than the background, so any pixel with a gray level $\leq T$ was labeled black (0), and any pixel with a gray level $>T$ was labeled white (255). The key objective is merely to generate a binary image, so the black–white relationship could be reversed.

The type of global thresholding just described can be expected to be successful in highly controlled environments. One of the areas in which this often is possible is in industrial inspection applications, where control of the illumination usually is feasible. ■

The threshold in the preceding example was specified by using a heuristic approach, based on visual inspection of the histogram. The following algorithm can be used to obtain T automatically:

1. Select an initial estimate for T.
2. Segment the image using T. This will produce two groups of pixels: G_1 consisting of all pixels with gray level values $>T$ and G_2 consisting of pixels with values $\leq T$.
3. Compute the average gray level values μ_1 and μ_2 for the pixels in regions G_1 and G_2.

4. Compute a new threshold value:

$$T = \frac{1}{2}(\mu_1 + \mu_2).$$

5. Repeat steps 2 through 4 until the difference in T in successive iterations is smaller than a predefined parameter T_o.

When there is reason to believe that the background and object occupy comparable areas in the image, a good initial value for T is the average gray level of the image. When objects are small compared to the area occupied by the background (or vice versa), then one group of pixels will dominate the histogram and the average gray level is not as good an initial choice. A more appropriate initial value for T in cases such as this is a value midway between the maximum and minimum gray levels. The parameter T_o is used to stop the algorithm after changes become small in terms of this parameter. This is used when speed of iteration is an important issue.

EXAMPLE 10.11: Image segmentation using an estimated global threshold.

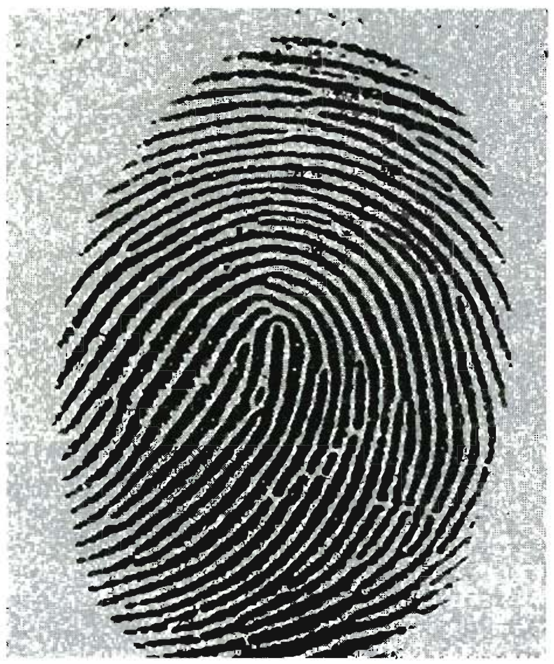
■ Figure 10.29 shows an example of segmentation based on a threshold estimated using the preceding algorithm. Figure 10.29(a) is the original image, and Fig. 10.29(b) is the image histogram. Note the clear valley of the histogram. Application of the iterative algorithm resulted in a value of 125.4 after three iterations starting with the average gray level and $T_o = 0$. The result obtained using $T = 125$ to segment the original image is shown in Fig. 10.29(c). As expected from the clear separation of modes in the histogram, the segmentation between object and background was very effective. ■

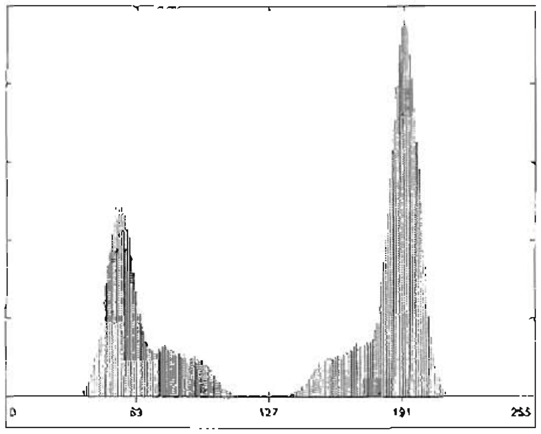
10.3.4 Basic Adaptive Thresholding

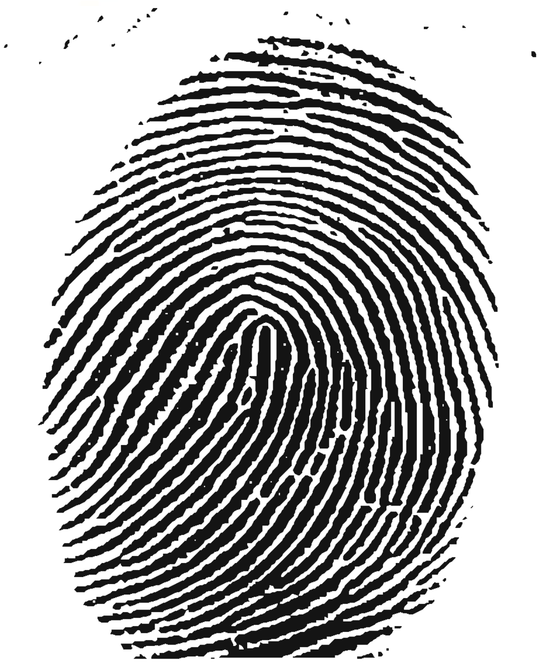
As illustrated in Fig. 10.27, imaging factors such as uneven illumination can transform a perfectly segmentable histogram into a histogram that cannot be partitioned effectively by a single global threshold. An approach for handling such a situation is to divide the original image into subimages and then utilize a different threshold to segment each subimage. The key issues in this approach are how to subdivide the image and how to estimate the threshold for each resulting subimage. Since the threshold used for each pixel depends on the location of the pixel in terms of the subimages, this type of thresholding is adaptive. We illustrate adaptive thresholding with a simple example. A more comprehensive example is given in the next section.

EXAMPLE 10.12: Basic adaptive thresholding.

■ Figure 10.30(a) shows the image from Fig. 10.27(d), which we concluded could not be thresholded effectively with a single global threshold. In fact, Fig. 10.30(b) shows the result of thresholding the image with a global threshold manually placed in the valley of its histogram [see Fig. 10.27(e)]. One approach to reduce the effect of nonuniform illumination is to subdivide the image into smaller subimages, such that the illumination of each subimage is approximately uniform. Figure 10.30(c) shows such a partition, obtained by subdividing the image into four equal parts, and then subdividing each part by four again.





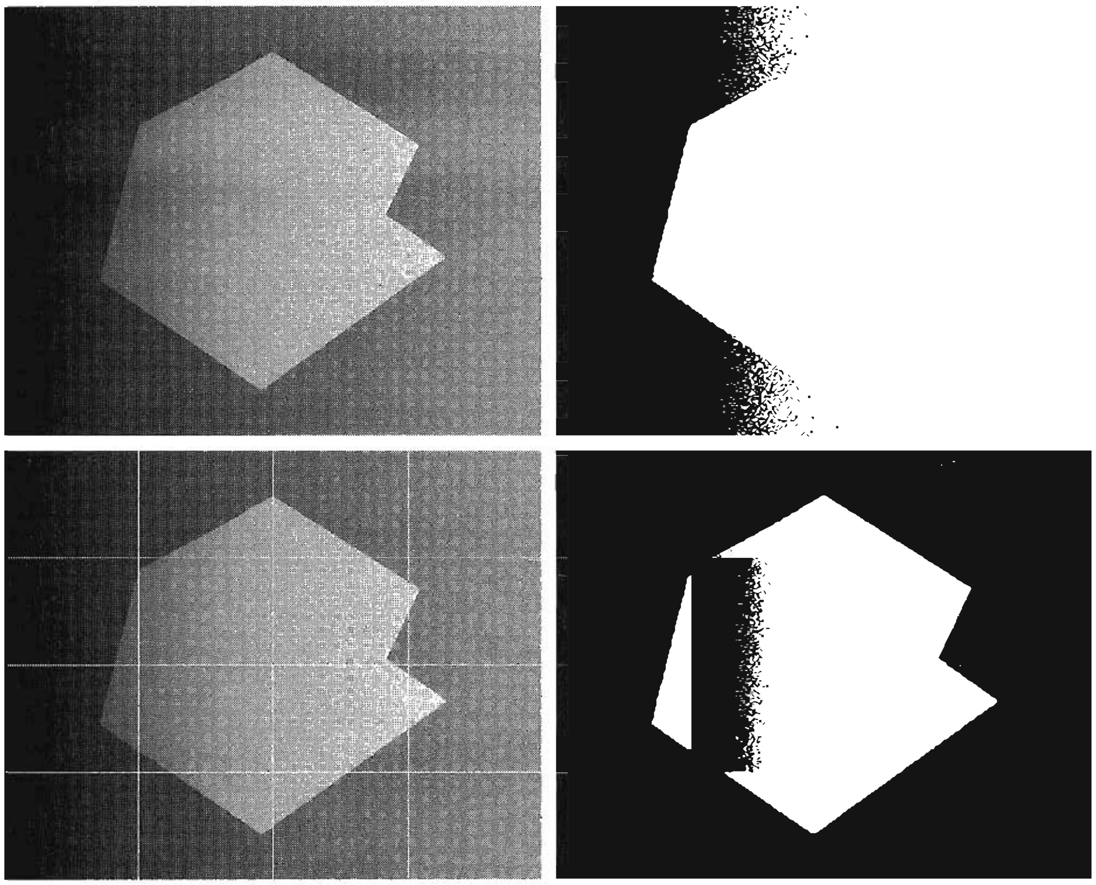


a b
c

FIGURE 10.29 (a) Original image. (b) Image histogram. (c) Result of segmentation with the threshold estimated by iteration. (Original courtesy of the National Institute of Standards and Technology.)

All the subimages that did not contain a boundary between object and background had variances of less than 75. All subimages containing boundaries had variances in excess of 100. Each subimage with variance greater than 100 was segmented with a threshold computed for that subimage using the algorithm discussed in the previous section. The initial value for T in each case was selected as the point midway between the minimum and maximum gray levels in the subimage. All subimages with variance less than 100 were treated as one composite image, which was segmented using a single threshold estimated using the same algorithm.

The result of segmentation using this procedure is shown in Fig. 10.30(d). With the exception of two subimages, the improvement over Fig. 10.30(b) is



a b
c d

FIGURE 10.30 (a) Original image. (b) Result of global thresholding. (c) Image subdivided into individual subimages. (d) Result of adaptive thresholding.

evident. The boundary between object and background in each of the improperly segmented subimages was small and dark, and the resulting histogram was almost unimodal. Figure 10.31(a) shows the top improperly segmented subimage from Fig. 10.30(c) and the subimage directly above it, which was segmented properly. The histogram of the subimage that was properly segmented is clearly bimodal, with well-defined peaks and valley. The other histogram is almost unimodal, with no clear distinction between object and background.

Figure 10.31(d) shows the failed subimage further subdivided into much smaller subimages, and Fig. 10.31(e) shows the histogram of the top, left small subimage. This subimage contains the transition between object and background. This smaller subimage has a clearly bimodal histogram and should be easily segmentable. This, in fact, is the case, as shown in Fig. 10.31(f). This figure also shows the segmentation of all the other small subimages. All these subimages had a nearly unimodal histogram, and their average gray level was closer to the object than to the background, so they were all classified as object. It is left as a project for the reader to show that considerably more accurate segmentation can be achieved by subdividing the entire image in Fig. 10.30(a) into subimages of the size shown in Fig. 10.31(d). ■

10.3.5 Optimal Global and Adaptive Thresholding

In this section we discuss a method for estimating thresholds that produce the minimum average segmentation error. As an illustration, the method is applied

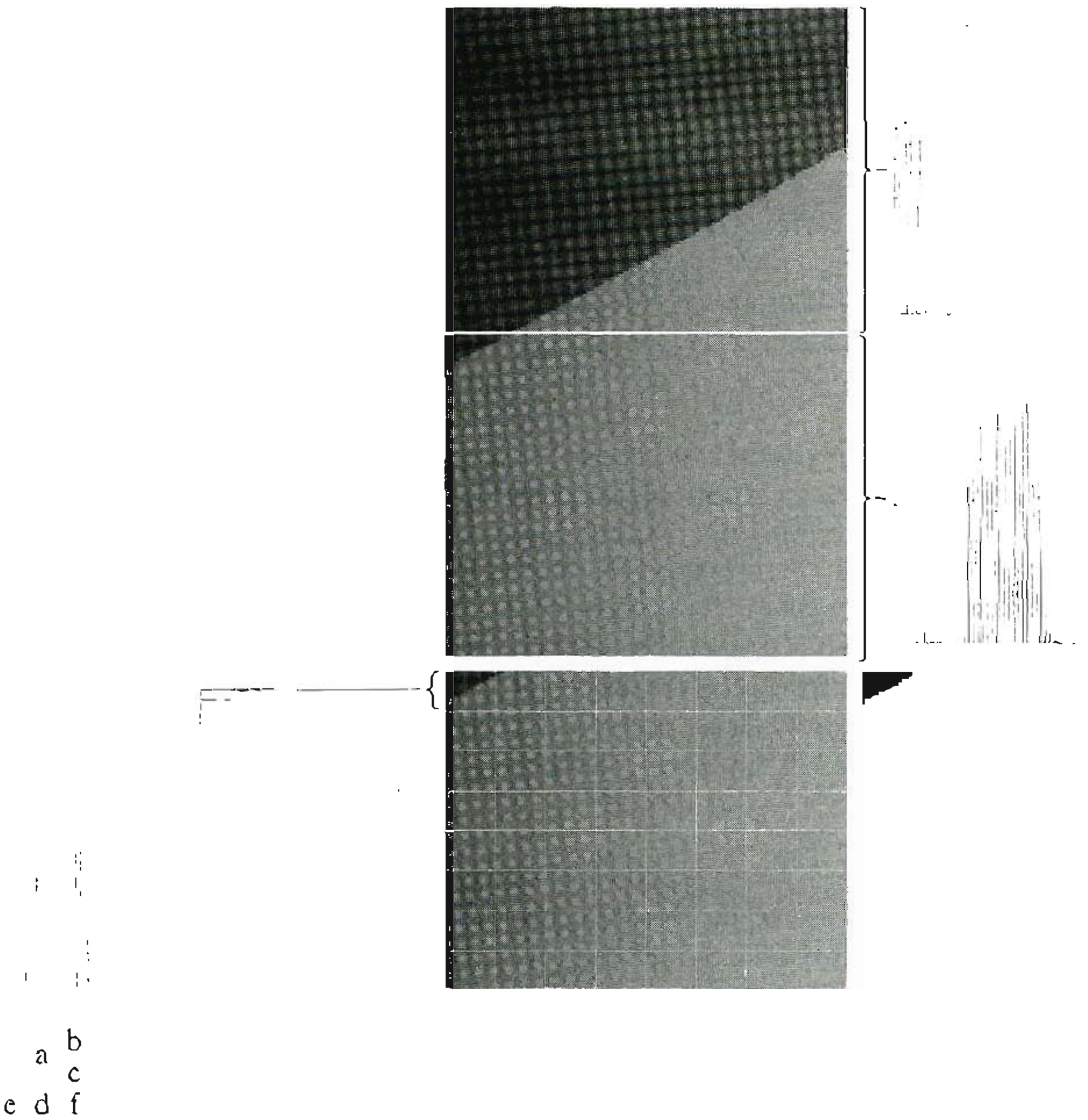


FIGURE 10.31 (a) Properly and improperly segmented subimages from Fig. 10.30. (b)–(c) Corresponding histograms. (d) Further subdivision of the improperly segmented subimage. (e) Histogram of small subimage at top, left. (f) Result of adaptively segmenting (d).

to a problem that requires solution of several important issues found frequently in the practical application of thresholding.

Suppose that an image contains only two principal gray-level regions. Let z denote gray-level values. We can view these values as random quantities, and their histogram may be considered an estimate of their probability density function (PDF), $p(z)$. This overall density function is the sum or mixture of two densities, one for the light and the other for the dark regions in the image. Furthermore, the mixture parameters are proportional to the relative areas of the dark and light regions. If the form of the densities is known or assumed, it is possible to determine an optimal threshold (in terms of minimum error) for segmenting the image into the two distinct regions.

See inside front cover
Consult the book web site for a brief review of probability theory.

Figure 10.32 shows two probability density functions. Assume that the larger of the two PDFs corresponds to the background levels while the smaller one

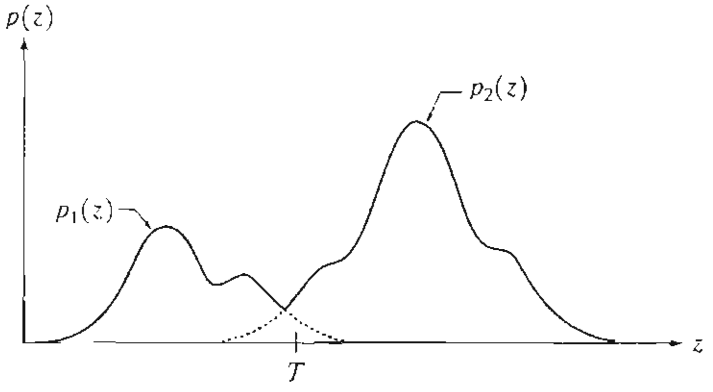


FIGURE 10.32 Gray-level probability density functions of two regions in an image.

describes the gray levels of objects in the image. The mixture probability density function describing the overall gray-level variation in the image is

$$p(z) = P_1 p_1(z) + P_2 p_2(z). \tag{10.3-5}$$

Here, P_1 and P_2 are the probabilities of occurrence of the two classes of pixels; that is, P_1 is the probability (a number) that a random pixel with value z is an object pixel. Similarly, P_2 is the probability that the pixel is a background pixel. We are assuming that any given pixel belongs either to an object or to the background, so that

$$P_1 + P_2 = 1. \tag{10.3-6}$$

An image is segmented by classifying as background all pixels with gray levels greater than a threshold T (see Fig. 10.32). All other pixels are called object pixels. Our main objective is to select the value of T that minimizes the average error in making the decisions that a given pixel belongs to an object or to the background.

Recall that the probability of a random variable having a value in the interval $[a, b]$ is the integral of its probability density function from a to b, which is the area of the PDF curve between these two limits. Thus, the probability of *erroneously* classifying a background point as an object point is

$$E_1(T) = \int_{-\infty}^{T} p_2(z)\, dz. \tag{10.3-7}$$

This is the area under the curve of $p_2(z)$ to the left of the threshold. Similarly, the probability of erroneously classifying an object point as background is

$$E_2(T) = \int_{T}^{\infty} p_1(z)\, dz, \tag{10.3-8}$$

which is the area under the curve of $p_1(z)$ to the right of T. Then the overall probability of error is

$$E(T) = P_2 E_1(T) + P_1 E_2(T). \tag{10.3-9}$$

Note how the quantities E_1 and E_2 are weighted (given importance) by the probability of occurrence of object or background pixels. Note also that the subscripts are opposites. This is simple to explain. Consider, for example, the extreme case in which background points are known never to occur. In this case $P_2 = 0$. The contribution to the overall error (E) of classifying a background point as an object point (E_1) should be zeroed out because background points are known never to occur. This is accomplished by multiplying E_1 by $P_2 = 0$. If background and object points are equally likely to occur, then the weights are $P_1 = P_2 = 0.5$.

To find the threshold value for which this error is minimal requires differentiating $E(T)$ with respect to T (using Leibniz's rule) and equating the result to 0. The result is

$$P_1 p_1(T) = P_2 p_2(T). \tag{10.3-10}$$

This equation is solved for T to find the optimum threshold. Note that if $P_1 = P_2$, then the optimum threshold is where the curves for $p_1(z)$ and $p_2(z)$ intersect (see Fig. 10.32).

Obtaining an analytical expression for T requires that we know the equations for the two PDFs. Estimating these densities in practice is not always feasible, and an approach used often is to employ densities whose parameters are reasonably simple to obtain. One of the principal densities used in this manner is the Gaussian density, which is completely characterized by two parameters: the mean and the variance. In this case,

$$p(z) = \frac{P_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(z-\mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{(z-\mu_2)^2}{2\sigma_2^2}} \tag{10.3-11}$$

where μ_1 and σ_1^2 are the mean and variance of the Gaussian density of one class of pixels (say, objects) and μ_2 and σ_2^2 are the mean and variance of the other class. Using this equation in the general solution of Eq. (10.3-10) results in the following solution for the threshold T:

$$AT^2 + BT + C = 0 \tag{10.3-12}$$

where

$$\begin{aligned} A &= \sigma_1^2 - \sigma_2^2 \\ B &= 2(\mu_1\sigma_2^2 - \mu_2\sigma_1^2) \\ C &= \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2 2\sigma_2^2 \ln(\sigma_2 P_1/\sigma_1 P_2). \end{aligned} \tag{10.3-13}$$

Since a quadratic equation has two possible solutions, two threshold values may be required to obtain the optimal solution.

If the variances are equal, $\sigma^2 = \sigma_1^2 = \sigma_2^2$, a single threshold is sufficient:

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \ln\left(\frac{P_2}{P_1}\right). \tag{10.3-14}$$

If $P_1 = P_2$, the optimal threshold is the average of the means. The same is true if $\sigma = 0$. Determining the optimal threshold may be similarly accomplished for other densities of known form, such as the Raleigh and log-normal densities.

Instead of assuming a functional form for $p(z)$, a minimum mean-square-error approach may be used to estimate a composite gray-level PDF of an image from the image histogram. For example, the mean square error between the (continuos) mixture density $p(z)$ and the (discrete) image histogram $h(z_i)$ is

$$e_{\text{ms}} = \frac{1}{n}\sum_{i=1}^{n}\left[p(z_i) - h(z_i)\right]^2 \tag{10.3-15}$$

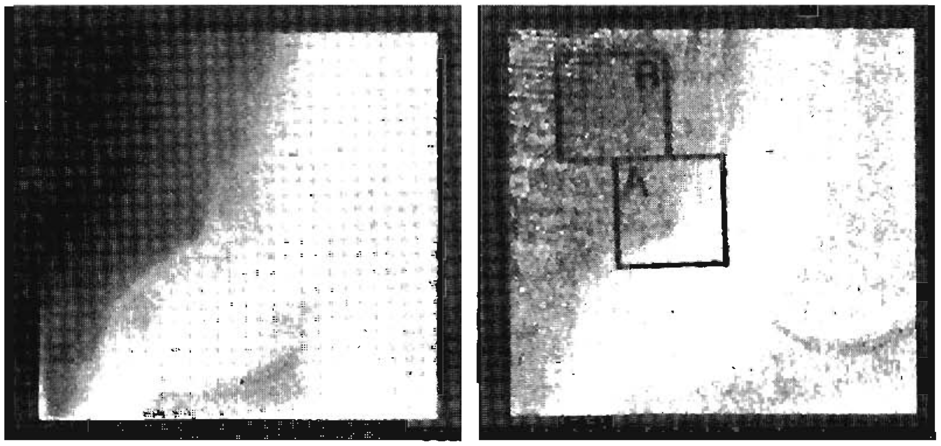
where an n-point histogram is assumed. The principal reason for estimating the complete density is to determine the presence or absence dominant modes in the PDF. For example, two dominant modes typically indicate the presence of edges in the image (or region) over which the PDF is computed.

In general, determining analytically the parameters that minimize this mean square error is not a simple matter. Even for the Gaussian case, the straightforward computation of equating the partial derivatives to 0 leads to a set of simultaneous transcendental equations that usually can be solved only by numerical procedures, such as a conjugate gradients or Newton's method for simultaneous nonlinear equations.

EXAMPLE 10.13: Use of optimum thresholding for image segmentation.

■ The following is one of the earliest (and still one of the most instructive) examples of segmentation by optimum thresholding in image processing. This example is particularly interesting at this junction because it shows how segmentation results can be improved by employing preprocessing techniques based on methods developed in our discussion of image enhancement. In addition, the example also illustrates the use of local histogram estimation and adaptive thresholding. The general problem is to outline automatically the boundaries of heart ventricles in cardioangiograms (X-ray images of a heart that has been injected with a contrast medium). The approach discussed here was developed by Chow and Kaneko [1972] for outlining boundaries of the left ventricle of the heart.

Prior to segmentation, all images were preprocessed as follows: (1) Each pixel was mapped with a log function (see Section 3.2.2) to counter exponential effects caused by radioactive absorption. (2) An image obtained before application of the contrast medium was subtracted from each image captured after the medium was injected in order to remove the spinal column present in both images (see Section 3.4.1). (3) Several angiograms were summed in order to reduce random noise (see Section 3.4.2). Figure 10.33 shows a cardioangiogram



a b

FIGURE 10.33 A cardioangiogram before and after preprocessing. (Chow and Kaneko.)

before and after preprocessing (an explanation of the regions marked A and B is given in the following paragraph).

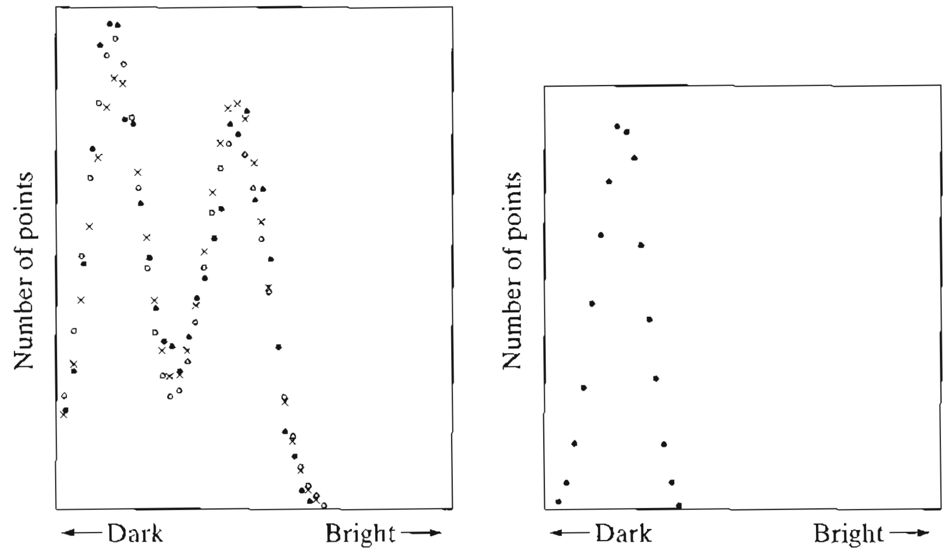
In order to compute the optimal thresholds, each preprocessed image was subdivided into 49 regions by placing a 7×7 grid with 50% overlap over each image (all original images shown in this example are of size 256×256 pixels). Each of the 49 resulting overlapped regions contained 64×64 pixels. Figures 10.34(a) and 10.34(b) are the histograms of the regions marked A and B in Fig. 10.33(b). Note that the histogram for region A clearly is bimodal, indicating the presence of a boundary. The histogram for region B, however, is unimodal, indicating the absence of two markedly distinct regions.

After all 49 histograms were computed, a test of bimodality was performed to reject the unimodal histograms. The remaining histograms were then fitted by bimodal Gaussian density curves [see Eq. (10.3-11)] using a conjugate gradient hill-climbing method to minimize the error function given in Eq. (10.3-15). The X's and O's in Fig. 10.34(a) are two fits to the histogram shown in black dots. The optimum thresholds were then obtained by using Eqs. (10.3-12) and (10.3-13).

At this stage of the process only the regions with bimodal histograms were assigned thresholds. The thresholds for the remaining regions were obtained by interpolating these thresholds. Then a second interpolation was carried out point by point by using neighboring threshold values so that, at the end of the procedure, every point in the image had been assigned a threshold. Finally, a binary decision was carried out for each pixel using the rule

$$f(x, y) = \begin{cases} 1 & \text{if } f(x, y) \geq T_{xy} \\ 0 & \text{otherwise} \end{cases}$$

where T_{xy} was the threshold assigned to location (x, y) in the image [note that these are adaptive thresholds, because they depend on the spatial coordinates (x, y)]. Boundaries were obtained by taking the gradient of the binary picture. Figure 10.35 shows the boundaries superimposed on the original image. Considering the variability and complexity of the images involved, this procedure yielded excellent segmentation results. ■



a b

FIGURE 10.34 Histograms (black dots) of (a) region A, and (b) region B in Fig. 10.33(b). (Chow and Kaneko.)

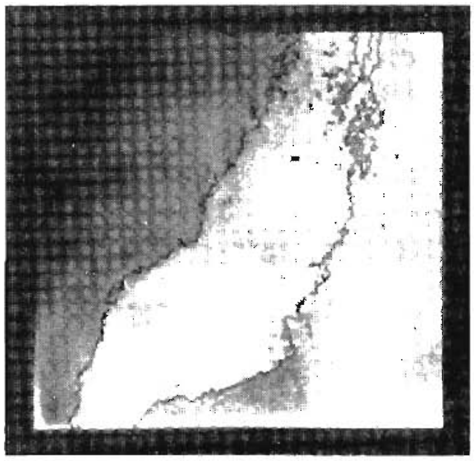


FIGURE 10.35 Cardioangiogram showing superimposed boundaries. (Chow and Kaneko.)

10.3.6 Use of Boundary Characteristics for Histogram Improvement and Local Thresholding

Based on the discussions in the previous five sections, it is intuitively evident that the chances of selecting a "good" threshold are enhanced considerably if the histogram peaks are tall, narrow, symmetric, and separated by deep valleys. One approach for improving the shape of histograms is to consider only those pixels that lie on or near the edges between objects and the background. An immediate and obvious improvement is that histograms would be less dependent on the relative sizes of objects and the background. For instance, the histogram of an image composed of a small object on a large background area (or vice versa) would be dominated by a large peak because of the high concentration of one type of pixels. Figures 10.30 and 10.31 are a good illustration of how segmentation performance is affected by this condition.

If only the pixels on or near the edge between object and the background were used, the resulting histogram would have peaks of approximately the same height. In addition, the probability that any of those given pixels lies on an object would be approximately equal to the probability that it lies on the background, thus improving the symmetry of the histogram peaks. Finally, as indicated in the following paragraph, using pixels that satisfy some simple measures based on gradient and Laplacian operators has a tendency to deepen the valley between histogram peaks.

The principal problem with the approach just discussed is the implicit assumption that the edges between objects and background are known. This information clearly is not available during segmentation, as finding a division between objects and background is precisely what segmentation is all about. However, from the discussion in Section 10.1.3, an indication of whether a pixel is on an edge may be obtained by computing its gradient. In addition, use of the Laplacian can yield information regarding whether a given pixel lies on the dark or light side of an edge. The average value of the Laplacian is 0 at the transition of an edge (see Fig. 10.6), so in practice the valleys of histograms formed from the pixels selected by a gradient/Laplacian criterion can be expected to be sparsely populated. This property produces the highly desirable deep valleys discussed previously.

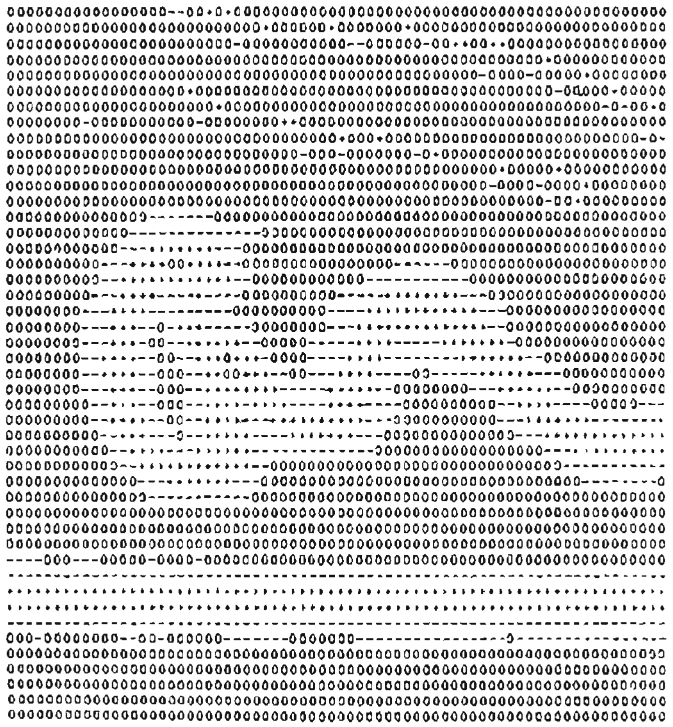


FIGURE 10.36 Image of a handwritten stroke coded by using Eq. (10.3-16). (Courtesy of IBM Corporation.)

The gradient ∇f at any point (x, y) in an image is given by Eq. (10.1-4) or (10.1-12). Similarly, the Laplacian $\nabla^2 f$ is given by Eq. (10.1-14) or (10.1-15). These two quantities may be used to form a three-level image, as follows:

$$s(x, y) = \begin{cases} 0 & \text{if } \nabla f < T \\ + & \text{if } \nabla f \geq T \quad \text{and} \quad \nabla^2 f \geq 0 \\ - & \text{if } \nabla f \geq T \quad \text{and} \quad \nabla^2 f < 0 \end{cases} \tag{10.3-16}$$

where the symbols 0, +, and − represent any three distinct gray levels, T is a threshold, and the gradient and Laplacian are computed at every point (x, y). For a dark object on a light background, and with reference to Fig. 10.6, the use of Eq. (10.3-16) produces an image $s(x, y)$ in which (1) all pixels that are not on an edge (as determined by ∇f being less than T) are labeled 0; (2) all pixels on the dark side of an edge are labeled +; and (3) all pixels on the light side of an edge are labeled −. The symbols + and − in Eq. (10.3-16) are reversed for a light object on a dark background. Figure 10.36 shows the labeling produced by Eq. (10.3-16) for an image of a dark, underlined stroke written on a light background.

The information obtained with this procedure can be used to generate a segmented, binary image in which 1's correspond to objects of interest and 0's correspond to the background. The transition (along a horizontal or vertical scan line) from a light background to a dark object must be characterized by the occurrence of a − followed by a + in $s(x, y)$. The interior of the object is composed of pixels that are labeled either 0 or +. Finally, the transition from the object back to the background is characterized by the occurrence of a + followed by a −. Thus a horizontal or vertical scan line containing a section of an object has the following structure:

$$(\cdots)(-, +)(0 \text{ or } +)(+, -)(\cdots)$$

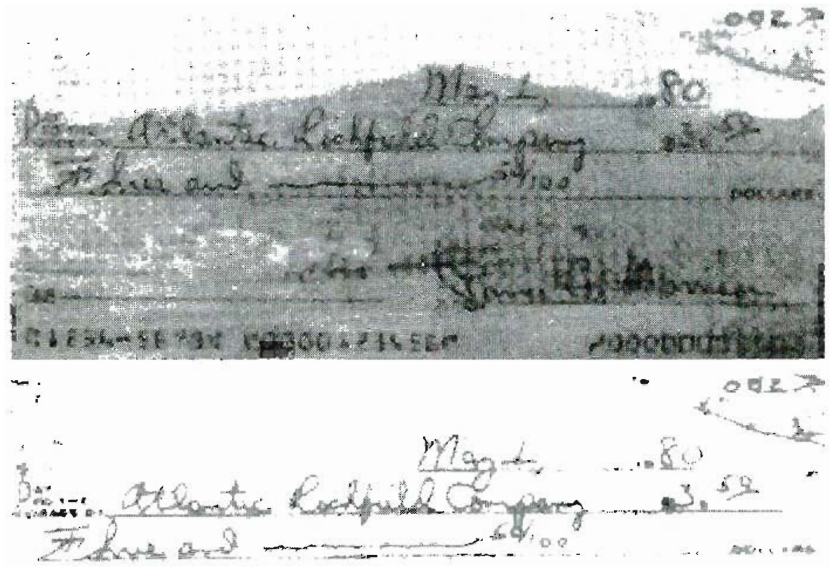

a
b

FIGURE 10.37 (a) Original image. (b) Image segmented by local thresholding. (Courtesy of IBM Corporation.)

where $(\cdots)$ represents any combination of $+$, $-$, and 0. The innermost parentheses contain object points and are labeled 1. All other pixels along the same scan line are labeled 0, with the exception of any other sequence of (0 or $+$) bounded by $(-, +)$ and $(+, -)$.

EXAMPLE 10.14: Image segmentation by local thresholding.

Figure 10.37(a) shows an image of an ordinary scenic bank check. Figure 10.38 shows the histogram as a function of gradient values for pixels with gradients greater than 5. Note that this histogram has two dominant modes that are symmetric, nearly of the same height, and are separated by a distinct valley. Finally, Fig. 10.37(b) shows the segmented image obtained by using Eq. (10.3-16) with T at or near the midpoint of the valley. The result was made binary by using the sequence analysis just discussed. Note that this example is an illustration of local thresholding, as defined in Eq. (10.3-1), because the value of T was determined from a histogram of the gradient and Laplacian, which are local properties.

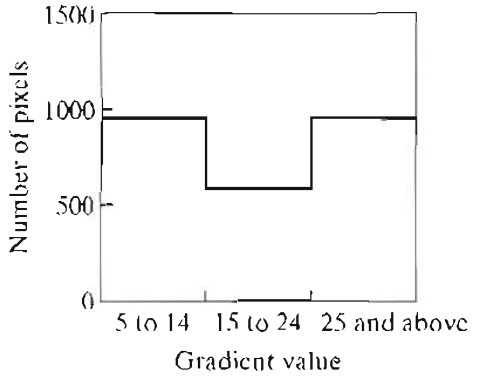


FIGURE 10.38 Histogram of pixels with gradients greater than 5. (Courtesy of IBM Corporation.)

10.3.7 Thresholds Based on Several Variables

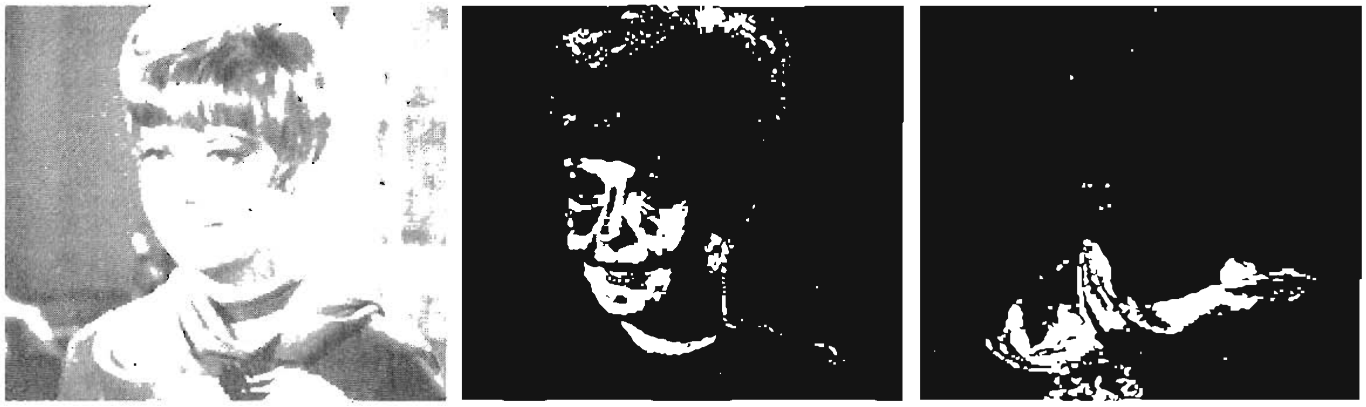
So far we have been concerned with thresholding gray levels. In some cases, a sensor can make available more than one variable to characterize each pixel in an image, and thus allow *multispectral theresholding*. As discussed in some detail in Section 6.7, color imaging is a good example, in which each pixel is characterized by three RGB values. In this case, constructing a 3-D "histogram" becomes possible. The basic procedure is analogous to the method used for one variable. For example, for an image with three variables (RGB components), each having 16 possible levels, a $16 \times 16 \times 16$ grid (cube) is formed. Inserted in each cell of the cube is the number of pixels whose RGB components have values corresponding to the coordinates defining the location of that particular cell. Each entry is then divided by the total number of pixels in the image to form a normalized histogram.

The concept of thresholding now becomes one of finding clusters of points in 3-D space. Suppose, for example, that K significant clusters of points are found in the histogram. The image can be segmented by assigning one arbitrary value (say, white) to pixels whose RGB components are closer to one cluster and another value (say, black) to the other pixels in the image. This concept is easily extendable to more components and certainly to more clusters. The principal difficulty is that cluster seeking becomes an increasingly complex task as the number of variables increases. Cluster-seeking methods can be found, for example, in the books by Duda, Hart, and Stork [2001], and Tou and Gonzalez [1974].

EXAMPLE 10.15: Multispectral thresholding.

The image shown in Fig. 10.39(a) is a monochrome picture of a color photograph. The original color image is composed of three 16-level RGB images. The scarf is a vivid red, and the hair and facial colors are light and different in spectral characteristics from the window and other background features.

Figure 10.39(b) was obtained by thresholding about one of the histogram clusters corresponding to facial tones. Note that the window, which in the monochrome



a b c

FIGURE 10.39 (a) Original color image shown as a monochrome picture. (b) Segmentation of pixels with colors close to facial tones. (c) Segmentation of red components.

picture is close in gray-level value to the hair, does not appear in the segmented image because of the use of multispectral characteristics to separate these two regions. Figure 10.39(c) was obtained by thresholding about a cluster close to the red axis. In this case only the scarf and part of a flower (which is red also) appeared in the segmented result. The threshold used to obtain both results was a distance of one cell. Thus any pixel whose components were outside the cell enclosing the center of the cluster in question was classified as background (black). Pixels whose components placed them inside the cell were coded white. ■

As discussed in Section 6.7, color segmentation can be based on any of the color models introduced in Chapter 6. For instance, hue and saturation are important properties in numerous applications dealing with the use of imaging for automated inspection. These properties are particularly important in attempts to emulate the equivalent function performed by humans, such as in the inspection of fruits for ripeness or in the inspection of manufactured goods. As mentioned in Chapter 6, the Hue, Saturation, Intensity (HSI) model is ideal for these types of applications because it is closely related to the way in which humans describe the perception of color. A segmentation approach using the hue and saturation components of a color signal also is particularly attractive, because it involves 2-D data clusters that are easier to analyze than, say, the 3-D clusters needed for RGB segmentation.

10.4 Region-Based Segmentation

The objective of segmentation is to partition an image into regions. In Sections 10.1 and 10.2 we approached this problem by finding boundaries between regions based on discontinuities in gray levels, whereas in Section 10.3 segmentation was accomplished via thresholds based on the distribution of pixel properties, such as gray-level values or color. In this section we discuss segmentation techniques that are based on finding the regions directly.

10.4.1 Basic Formulation

Let R represent the entire image region. We may view segmentation as a process that partitions R into n subregions, $R_1, R_2, \ldots, R_n$, such that

(a) $\bigcup_{i=1}^{n} R_i = R$.
(b) R_i is a connected region, $i = 1, 2, \ldots, n$.
(c) $R_i \cap R_j = \varnothing$ for all i and j, $i \neq j$.
(d) $P(R_i) = \text{TRUE}$ for $i = 1, 2, \ldots, n$.
(e) $P(R_i \cup R_j) = \text{FALSE}$ for $i \neq j$.

Here, $P(R_i)$ is a logical predicate defined over the points in set R_i and $\varnothing$ is the null set.

Condition (a) indicates that the segmentation must be complete; that is, every pixel must be in a region. Condition (b) requires that points in a region must be connected in some predefined sense (see Section 2.5.2 regarding connectivity).

Condition (c) indicates that the regions must be disjoint. Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region—for example $P(R_i) = \text{TRUE}$ if all pixels in R_i have the same gray level. Finally, condition (e) indicates that regions R_i and R_j are different in the sense of predicate P.

10.4.2 Region Growing

As its name implies, *region growing* is a procedure that groups pixels or subregions into larger regions based on predefined criteria. The basic approach is to start with a set of "seed" points and from these grow regions by appending to each seed those neighboring pixels that have properties similar to the seed (such as specific ranges of gray level or color).

Selecting a set of one or more starting points often can be based on the nature of the problem, as will be shown in Example 10.16. When a priori information is not available, the procedure is to compute at every pixel the same set of properties that ultimately will be used to assign pixels to regions during the growing process. If the result of these computations shows clusters of values, the pixels whose properties place them near the centroid of these clusters can be used as seeds.

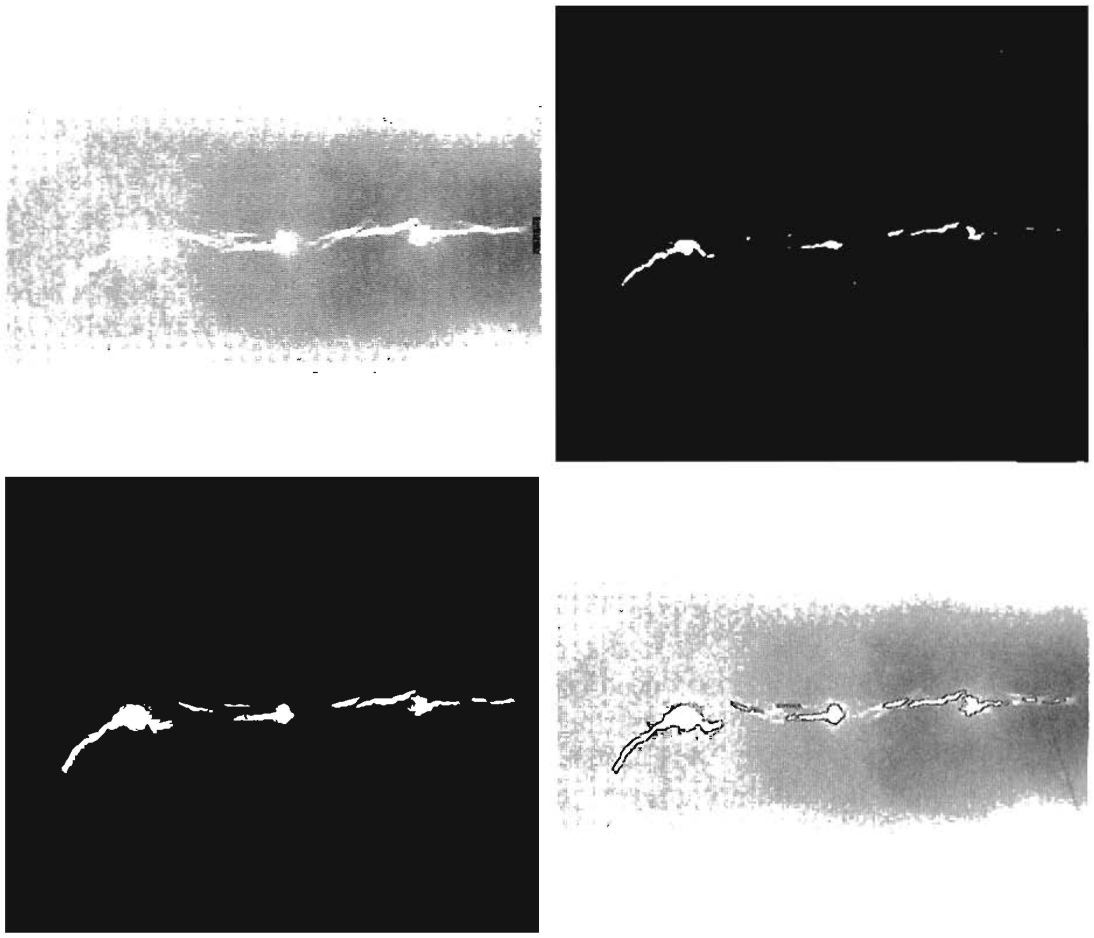
The selection of similarity criteria depends not only on the problem under consideration, but also on the type of image data available. For example, the analysis of land-use satellite imagery depends heavily on the use of color. This problem would be significantly more difficult, or even impossible, to handle without the inherent information available in color images. When the images are monochrome, region analysis must be carried out with a set of descriptors based on gray levels and spatial properties (such as moments or texture). We discuss descriptors useful for region characterization in Chapter 11.

Descriptors alone can yield misleading results if connectivity or adjacency information is not used in the region-growing process. For example, visualize a random arrangement of pixels with only three distinct gray-level values. Grouping pixels with the same gray level to form a "region" without paying attention to connectivity would yield a segmentation result that is meaningless in the context of this discussion.

Another problem in region growing is the formulation of a stopping rule. Basically, growing a region should stop when no more pixels satisfy the criteria for inclusion in that region. Criteria such as gray level, texture, and color, are local in nature and do not take into account the "history" of region growth. Additional criteria that increase the power of a region-growing algorithm utilize the concept of size, likeness between a candidate pixel and the pixels grown so far (such as a comparison of the gray level of a candidate and the average gray level of the grown region), and the shape of the region being grown. The use of these types of descriptors is based on the assumption that a model of expected results is at least partially available.

EXAMPLE 10.16: Application of region growing in weld inspection.

■ Figure 10.40(a) shows an X-ray image of a weld (the horizontal dark region) containing several cracks and porosities (the bright, white streaks running horizontally through the middle of the image). We wish to use region growing to segment the regions of the weld failures. These segmented features could be used



a b
c d

FIGURE 10.40 (a) Image showing defective welds. (b) Seed points. (c) Result of region growing. (d) Boundaries of segmented defective welds (in black). (Original image courtesy of X-TEK Systems, Ltd.).

for inspection, for inclusion in a database of historical studies, for controlling an automated welding system, and for other numerous applications.

The first order of business is to determine the initial seed points. In this application, it is known that pixels of defective welds tend to have the maximum allowable digital value (255 in this case). Based on this information, we selected as starting points all pixels having values of 255. The points thus extracted from the original image are shown in Fig. 10.40(b). Note that many of the points are clustered into *seed regions*.

The next step is to choose criteria for region growing. In this particular example we chose two criteria for a pixel to be annexed to a region: (1) The absolute gray-level difference between any pixel and the seed had to be less than 65. This number is based on the histogram shown in Fig. 10.41 and represents the difference between 255 and the location of the first major valley to the left, which is representative of the highest gray level value in the dark weld region. (2) To be included in one of the regions, the pixel had to be 8-connected to at least one pixel in that region. If a pixel was found to be connected to more than one region, the regions were merged.

Figure 10.40(c) shows the regions that resulted by starting with the seeds in Fig. 10.40(b) and utilizing the criteria defined in the previous paragraph. Superimposing the boundaries of these regions on the original image [Fig. 10.40(d)] reveals that the region-growing procedure did indeed segment the defective

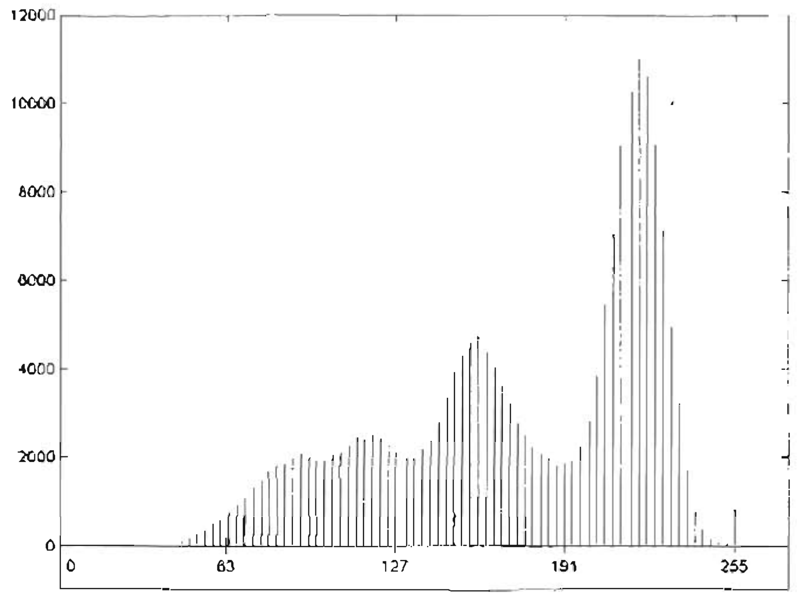


FIGURE 10.41 Histogram of Fig. 10.40(a).

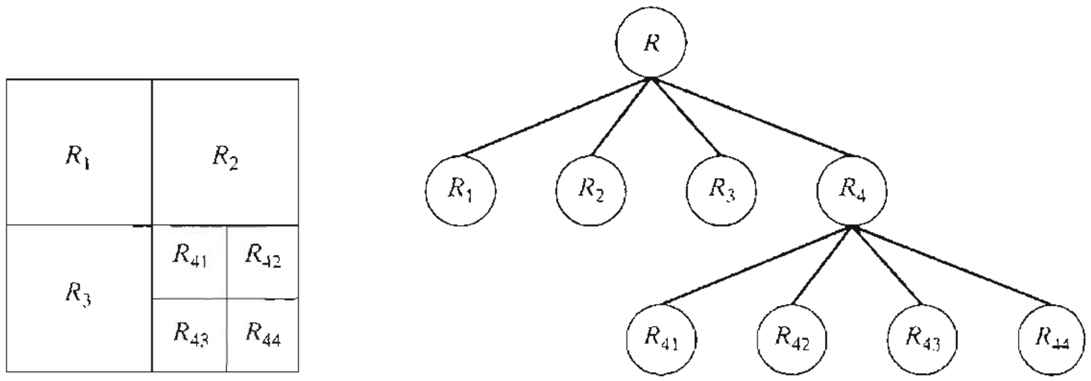
welds with an acceptable degree of accuracy. It is of interest to note that it was not necessary to specify any stopping rules in this case because the criteria for region growing were sufficient to isolate the features of interest. ■

It was mentioned in Section 10.3.1 in connection with Fig. 10.26(b) that problems having multimodal histograms generally are best solved using region-based approaches. The histogram shown in Fig. 10.41 is an excellent example of a "clean" multimodal histogram. This histogram and the results in Example 10.16 confirm the assertion that, even with well-behaved histograms, multilevel thresholding is a difficult proposition. Based on the results of Example 10.16, it should be intuitively obvious that this problem cannot be solved effectively by any reasonably general method of automatic threshold selection based on gray levels alone. The use of *connectivity* was fundamental in solving the problem.

10.4.3 Region Splitting and Merging

The procedure just discussed grows regions from a set of seed points. An alternative is to subdivide an image initially into a set of arbitrary, disjointed regions and then merge and/or split the regions in an attempt to satisfy the conditions stated in Section 10.4.1. A split and merge algorithm that iteratively works toward satisfying these constraints is developed next.

Let R represent the entire image region and select a predicate P. One approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that, for any region R_i, $P(R_i)$ = TRUE. We start with the entire region. If $P(R)$ = FALSE, we divide the image into quadrants. If P is FALSE for any quadrant, we subdivide that quadrant into subquadrants, and so on. This particular splitting technique has a convenient representation in the form of a so-called *quadtree* (that is, a tree in which nodes have exactly four descendants), as illustrated in Fig. 10.42. Note that the root of the tree corresponds to the entire image and that each node corresponds to a subdivision. In this case, only R_4 was subdivided further.



a b

FIGURE 10.42 (a) Partitioned image. (b) Corresponding quadtree.

If only splitting were used, the final partition likely would contain adjacent regions with identical properties. This drawback may be remedied by allowing merging, as well as splitting. Satisfying the constraints of Section 10.4.1 requires merging only adjacent regions whose combined pixels satisfy the predicate P. That is, two adjacent regions R_j and R_k are merged only if $P(R_j \cup R_k) = \text{TRUE}$.

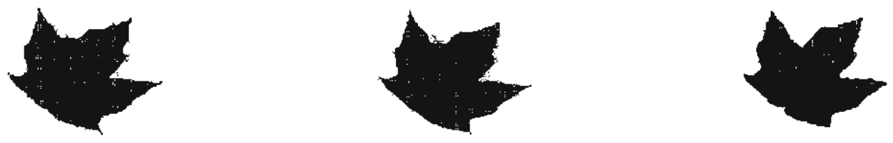
The preceding discussion may be summarized by the following procedure, in which, at any step we

1. Split into four disjoint quadrants any region R_i for which $P(R_i) = \text{FALSE}$.
2. Merge any adjacent regions R_j and R_k for which $P(R_j \cup R_k) = \text{TRUE}$.
3. Stop when no further merging or splitting is possible.

Several variations of the preceding basic theme are possible. For example, one possibility is to split the image initially into a set of blocks. Further splitting is carried out as described previously, but merging is initially limited to groups of four blocks that are descendants in the quadtree representation and that satisfy the predicate P. When no further mergings of this type are possible, the procedure is terminated by one final merging of regions satisfying step 2. At this point, the merged regions may be of different sizes. The principal advantage of this approach is that it uses the same quadtree for splitting and merging, until the final merging step.

EXAMPLE 10.17: Split and merge.

■ Figure 10.43(a) shows a simple image. We define $P(R_i) = \text{TRUE}$ if at least 80% of the pixels in R_i have the property $|z_j - m_i| \leq 2\sigma_i$, where z_j is the gray level of the jth pixel in R_i, m_i is the mean gray level of that region, and σ_i is the



a b c

FIGURE 10.43 (a) Original image. (b) Result of split and merge procedure. (c) Result of thresholding (a).

standard deviation of the gray levels in R_i. If $P(R_i)$ = TRUE under this condition, the values of all the pixels in R_i were set equal to m_i. Splitting and merging was done using the algorithm outlined previously. The result of applying this technique to the image in Fig. 10.43(a) is shown in Fig. 10.43(b). Note that the image was segmented perfectly. The image shown in Fig. 10.43(c) was obtained by thresholding Fig. 10.43(a), with a threshold placed midway between the two principal peaks of the histogram. The shading (and the stem of the leaf) were erroneously eliminated by the thresholding procedure. ■

As used in the preceding example, properties based on the mean and standard deviation of pixels in a region attempt to quantify the *texture* of a region (see Section 11.3.3 for a discussion on texture). The concept of *texture segmentation* is based on using measures of texture for the predicates $P(R_i)$. That is, we can perform texture segmentation by any of the methods discussed in this section by specifying predicates based on texture content.

10.5 Segmentation by Morphological Watersheds

Thus far, we have discussed segmentation based on three principal concepts: (a) detection of discontinuities, (b) thresholding, and (c) region processing. Each of these approaches was found to have advantages (for example, speed in the case of global thresholding) and disadvantages (for example, the need for post-processing, such as edge linking, in methods based on detecting discontinuities in gray levels). In this section we discuss an approach based on the concept of so-called *morphological watersheds*. As will become evident in the following discussion, segmentation by watersheds embodies many of the concepts of the other three approaches and, as such, often produces more stable segmentation results, including continuous segmentation boundaries. This approach also provides a simple framework for incorporating knowledge-based constraints (see Fig. 1.23) in the segmentation process.

10.5.1 Basic Concepts

The concept of watersheds is based on visualizing an image in three dimensions: two spatial coordinates versus gray levels. In such a "topographic" interpretation, we consider three types of points: (a) points belonging to a regional minimum; (b) 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; and (c) points at which water would be equally likely to fall to more than one such minimum. For a particular regional minimum, the set of points satisfying condition (b) is called the *catchment basin* or *watershed* of that minimum. The points satisfying condition (c) form crest lines on the topographic surface and are termed *divide lines* or *watershed lines*.

The principal objective of segmentation algorithms based on these concepts is to find the watershed lines. The basic idea is simple: Suppose that a hole is punched in each regional minimum and that the entire topography is flooded from below by letting water rise through the holes at a uniform rate. When the rising water in distinct catchment basins is about to merge, a dam is built to prevent the merging. The flooding will eventually reach a stage when only the tops of the dams are visible above the water line. These dam boundaries correspond to the divide lines of the watersheds. Therefore, they are the (continuous) boundaries extracted by a watershed segmentation algorithm.

These ideas can be explained further with the aid of Fig. 10.44. Figure 10.44(a) shows a simple gray-scale image and Fig. 10.44(b) is a topographic view, in which the height of the "mountains" is proportional to gray-level values in the input image. For ease of interpretation, the backsides of structures are shaded. This is not to be confused with gray-level values; only the general topography of the three-dimensional representation is of interest. In order to prevent the rising water from spilling out through the edges of the structure, we imagine the perimeter of the entire topography (image) being enclosed by dams of height greater than the highest possible mountain, whose value is determined by the highest possible gray-level value in the input image.

Suppose that a hole is punched in each regional minimum [shown as dark areas in Fig. 10.44(b)] and that the entire topography is flooded from below

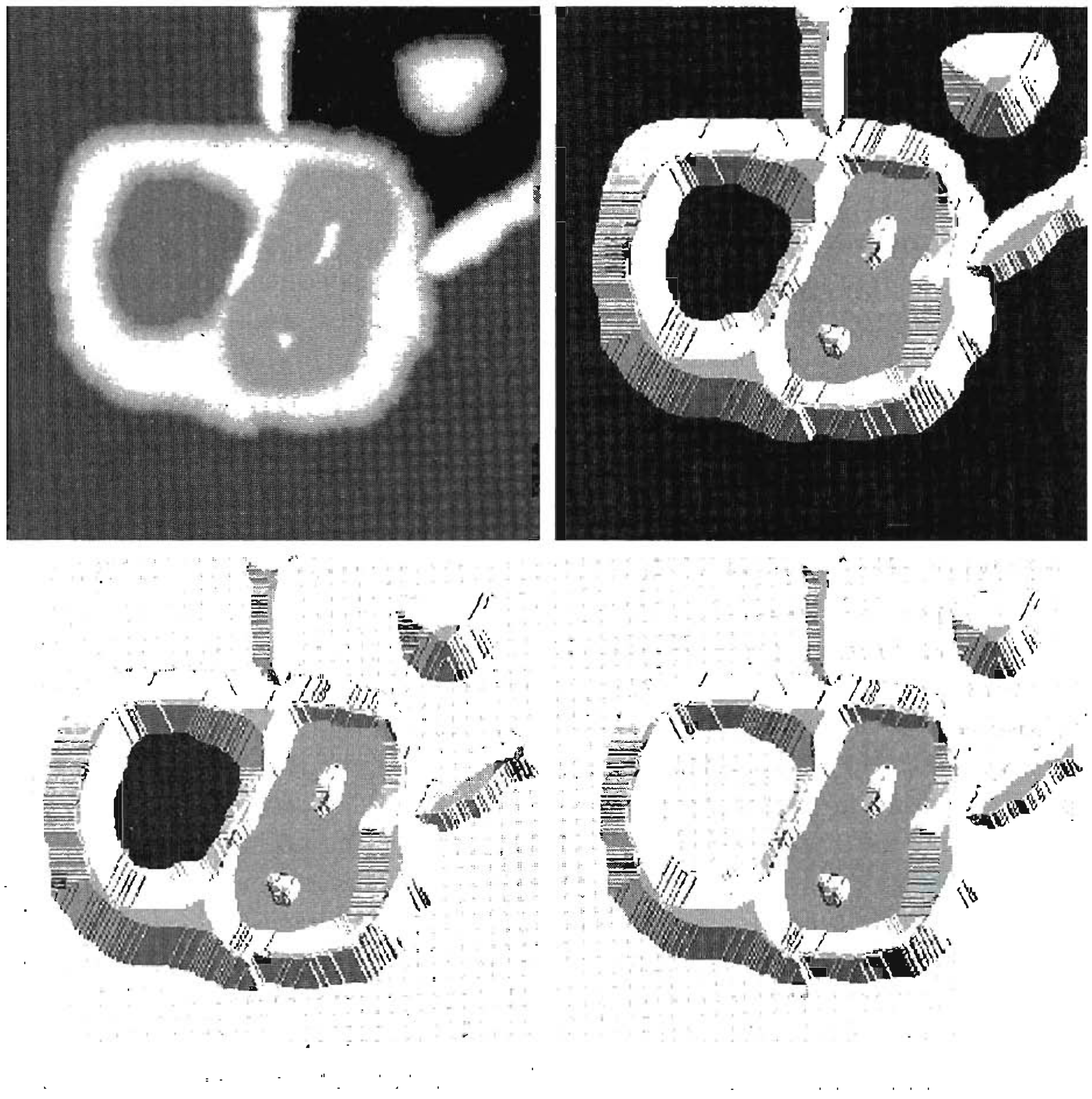
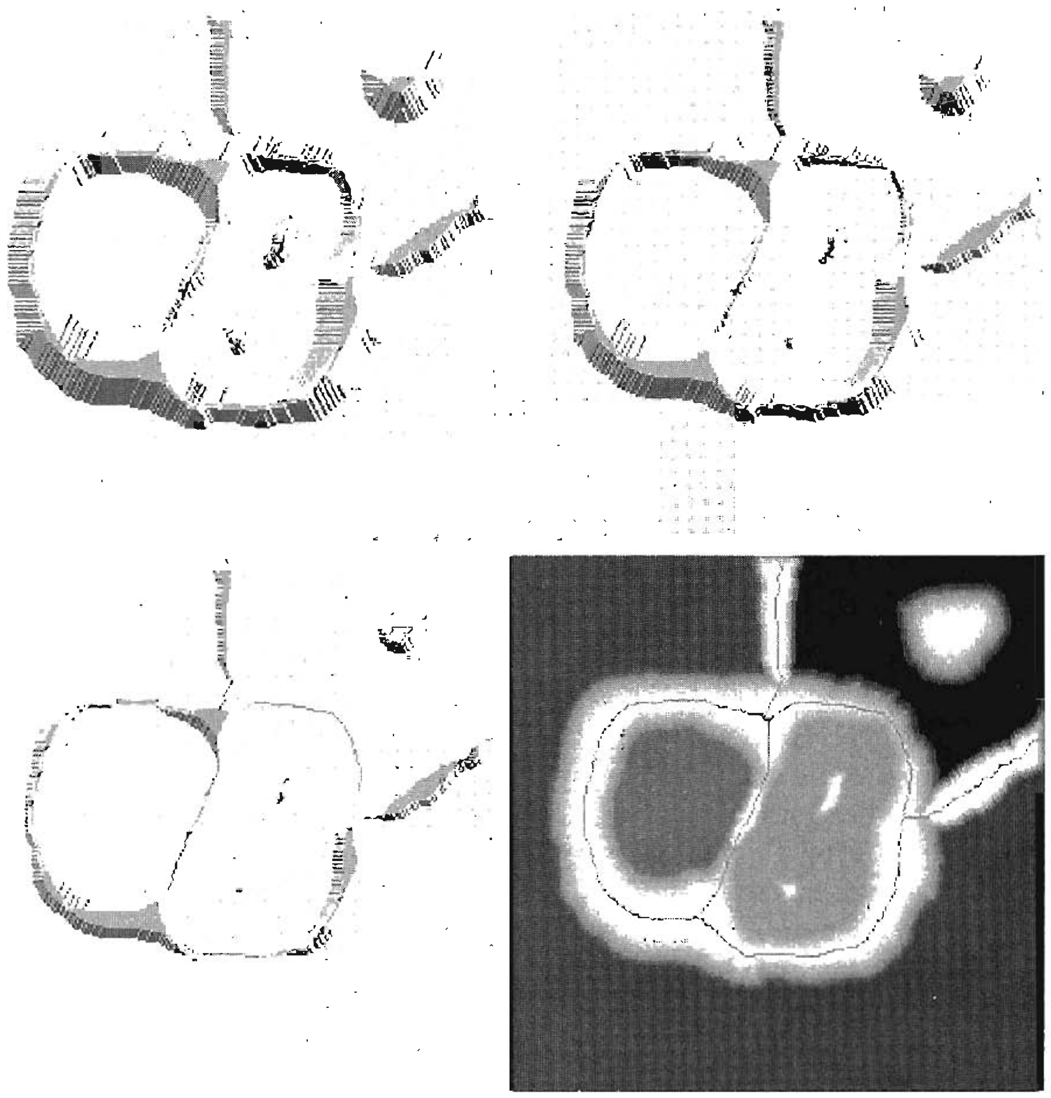

a b
c d

FIGURE 10.44 (a) Original image. (b) Topographic view. (c)–(d) Two stages of flooding.

by letting water rise through the holes at a uniform rate. Figure 10.44(c) shows the first stage of flooding, where the "water," shown in light gray, has covered only areas that correspond to the very dark background in the image. In Figs. 10.44(d) and (e) we see that the water now has risen into the first and second catchment basins, respectively. As the water continues to rise, it will eventually overflow from one catchment basin into another. The first indication of this is shown in 10.44(f). Here, water from the left basin actually overflowed into the basin on the right and a short "dam" (consisting of single pixels) was built to prevent water from merging at that level of flooding (the details of dam building are discussed in the following section). The effect is more pronounced as water continues to rise, as shown in Fig. 10.44(g). This figure shows a longer dam between the two catchment basins and another dam in the top part of the right basin. The latter dam was built to prevent merging of water from that basin with water from areas corresponding to the background. This process is continued until the maximum level of flooding (corresponding to the highest gray-level value in the image) is reached. The final dams correspond to the watershed lines, which are the desired segmentation result. The



e f
g h

FIGURE 10.44 *(Continued)* (e) Result of further flooding. (f) Beginning of merging of water from two catchment basins (a short dam was built between them). (g) Longer dams. (h) Final watershed (segmentation) lines. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

result for this example is shown in Fig. 10.44(h) as a dark, one-pixel-thick path superimposed on the original image. Note the important property that the watershed lines form a connected path, thus giving continuous boundaries between regions.

One of the principal applications of watershed segmentation is in the extraction of nearly uniform (bloblike) objects from the background. Regions characterized by small variations in gray levels have small gradient values. Thus, in practice, we often see watershed segmentation applied to the gradient of an image, rather than to the image itself. In this formulation, the regional minima of catchment basins correlate nicely with the small value of the gradient corresponding to the objects of interest.

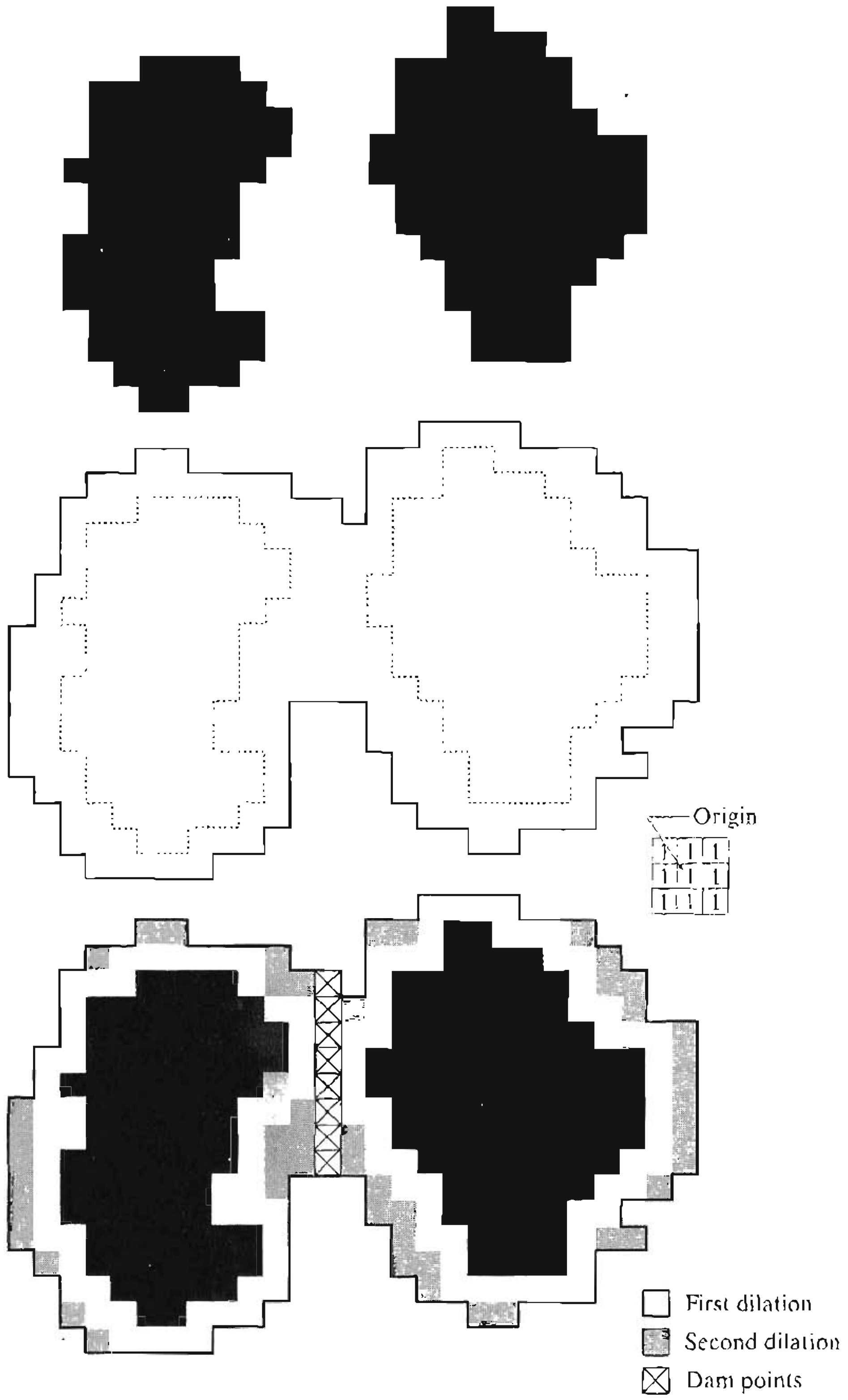
10.5.2 Dam Construction

Before proceeding, let us consider how to construct the dams or watershed lines required by watershed segmentation algorithms. Dam construction is based on binary images, which are members of 2-D integer space Z^2 (see Section 2.4.2). The simplest way to construct dams separating sets of binary points is to use morphological dilation (see Section 9.2.1).

The basics of how to construct dams using dilation are illustrated in Fig. 10.45. Figure 10.45(a) shows portions of two catchment basins at flooding step $n - 1$ and Fig. 10.45(b) shows the result at the next flooding step, n. The water has spilled from one basin to the other and, therefore, a dam must be built to keep this from happening. In order to be consistent with notation to be introduced shortly, let M_1 and M_2 denote the sets of coordinates of points in two regional minima. Then let the set of coordinates of points in the *catchment basin* associated with these two minima at stage $n - 1$ of flooding be denoted by $C_{n-1}(M_1)$ and $C_{n-1}(M_2)$, respectively. These are the two black regions shown in Fig. 10.45(a).

Let the union of these two sets be denoted by $C[n - 1]$. There are two connected components in Fig. 10.45(a) (see Section 2.5.2 regarding connected components) and only one connected component in Fig. 10.45(b). This connected component encompasses the earlier two components, shown dashed. The fact that two connected components have become a *single* component indicates that water between the two catchment basins has merged at flooding step n. Let this connected component be denoted q. Note that the two components from step $n - 1$ can be extracted from q by performing the simple AND operation $q \cap C[n - 1]$. We note also that all points belonging to an individual catchment basin form a single connected component.

Suppose that each of the connected components in Fig. 10.45(a) is dilated by the structuring element shown in Fig. 10.45(c), subject to two conditions: (1) The dilation has to be constrained to q (this means that the center of the structuring element can be located only at points in q during dilation), and (2) the dilation cannot be performed on points that would cause the sets being dilated to merge (become a single connected component). Figure 10.45(d) shows that a first dilation pass (in light gray) expanded the boundary of each original connected component. Note that condition (1) was satisfied by every point



a
b c
d

FIGURE 10.45 (a) Two partially flooded catchment basins at stage $n - 1$ of flooding. (b) Flooding at stage n, showing that water has spilled between basins (for clarity, water is shown in white rather than black). (c) Structuring element used for dilation. (d) Result of dilation and dam construction.

during dilation, and condition (2) did not apply to any point during the dilation process; thus the boundary of each region was expanded uniformly.

In the second dilation (shown in medium gray), several points failed condition (1) while meeting condition (2), resulting in the broken perimeter shown in the figure. It also is evident that the only points in q that satisfy the two conditions under consideration describe the one-pixel-thick connected path shown crossed-hatched in Fig. 10.45(d). This path constitutes the desired separating dam at stage n of flooding. Construction of the dam at this level of flooding is completed by setting all the points in the path just determined to a value greater than the maximum gray-level value of the image. The height of all dams is generally set at 1 plus the maximum allowed value in the image. This will prevent water from crossing over the part of the completed dam as the level of flooding is increased. It is important to note that dams built by this procedure, which are the desired segmentation boundaries, are connected components. In other words, this method eliminates the problems of broken segmentation lines.

Although the procedure just described is based on a simple example, the method used for more complex situations is exactly the same, including the use of the 3×3 symmetric structuring element shown in Fig. 10.45(c).

10.5.3 Watershed Segmentation Algorithm

Let $M_1, M_2, \ldots, M_R$ be sets denoting the *coordinates* of the points in the regional minima of an image $g(x, y)$. As indicated at the end of Section 10.5.1, this typically will be a gradient image. Let $C(M_i)$ be a set denoting the coordinates of the points in the catchment basin associated with regional minimum M_i (recall that the points in any catchment basin form a connected component). The notation min and max will be used to denote the minimum and maximum values of $g(x, y)$. Finally, let $T[n]$ represent the set of coordinates (s, t) for which $g(s, t) < n$. That is,

$$T[n] = \{(s, t) \,|\, g(s, t) < n\}. \tag{10.5-1}$$

Geometrically, $T[n]$ is the set of coordinates of points in $g(x, y)$ lying below the plane $g(x, y) = n$.

The topography will be flooded in *integer* flood increments, from $n = \min + 1$ to $n = \max + 1$. At any step n of the flooding process, the algorithm needs to know the number of points below the flood depth. Conceptually, suppose that the coordinates in $T[n]$ that are below the plane $g(x, y) = n$ are "marked" black, and all other coordinates are marked white. Then when we look "down" on the xy-plane at any increment n of flooding, we will see a binary image in which black points correspond to points in the function that are below the plane $g(x, y) = n$. This interpretation is quite useful in helping understand the following discussion.

Let $C_n(M_i)$ denote the set of coordinates of points in the catchment basin associated with minimum M_i that are flooded at stage n. With reference to the discussion in the previous paragraph, $C_n(M_i)$ may be viewed as a binary image given by

$$C_n(M_i) = C(M_i) \cap T[n]. \tag{10.5-2}$$

In other words, $C_n(M_i) = 1$ at location (x, y) if $(x, y) \in C(M_i)$ AND $(x, y) \in T[n]$; otherwise $C_n(M_i) = 0$. The geometrical interpretation of this result is straightforward. We are simply using the AND operator to isolate at stage n of flooding the portion of the binary image in $T[n]$ that is associated with regional minimum M_i.

Next, we let $C[n]$ denote the union of the flooded catchment basins portions at stage n:

$$C[n] = \bigcup_{i=1}^{R} C_n(M_i). \tag{10.5-3}$$

Then $C[\max + 1]$ is the union of all catchment basins:

$$C[\max + 1] = \bigcup_{i=1}^{R} C(M_i). \tag{10.5-4}$$

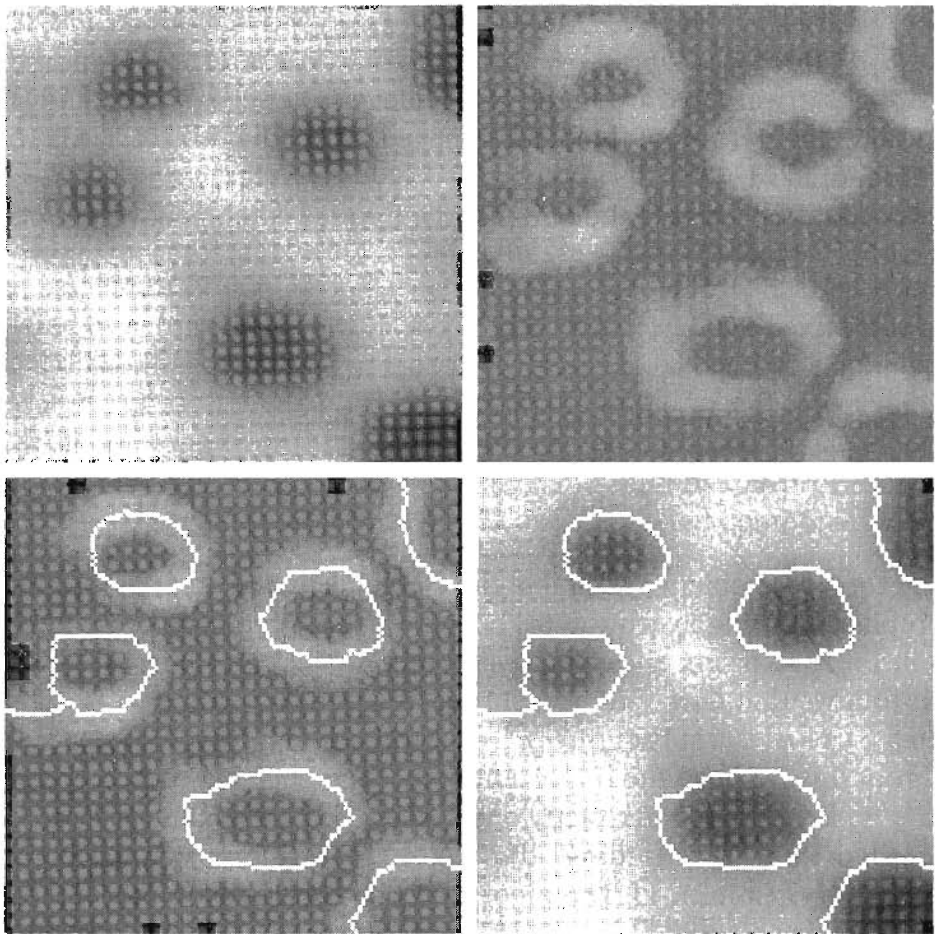
It can be shown (Problem 10.29) that the elements in both $C_n(M_i)$ and $T[n]$ are never replaced during execution of the algorithm, and that the number of elements in these two sets either increases or remains the same as n increases. Thus, it follows that $C[n - 1]$ is a subset of $C[n]$. According to Eqs. (10.5-2) and (10.5-3), $C[n]$ is a subset of $T[n]$, so it follows that $C[n - 1]$ is a subset of $T[n]$. From this we have the important result that each connected component of $C[n - 1]$ is contained in exactly one connected component of $T[n]$.

The algorithm for finding the watershed lines is initialized with $C[\min + 1] = T[\min + 1]$. The algorithm then proceeds recursively, assuming at step n that $C[n - 1]$ has been constructed. A procedure for obtaining $C[n]$ from $C[n - 1]$ is as follows. Let Q denote the set of connected components in $T[n]$. Then, for each connected component $q \in Q[n]$, there are three possibilities:

(a) $q \cap C[n - 1]$ is empty.
(b) $q \cap C[n - 1]$ contains one connected component of $C[n - 1]$.
(c) $q \cap C[n - 1]$ contains more than one connected component of $C[n - 1]$.

Construction of $C[n]$ from $C[n - 1]$ depends on which of these three conditions holds. Condition (a) occurs when a new minimum is encountered, in which case connected component q is incorporated into $C[n - 1]$ to form $C[n]$. Condition (b) occurs when q lies within the catchment basin of some regional minimum, in which case q is incorporated into $C[n - 1]$ to form $C[n]$. Condition (c) occurs when all, or part, of a ridge separating two or more catchment basins is encountered. Further flooding would cause the water level in these catchment basins to merge. Thus a dam (or dams if more than two catchment basins are involved) must be built within q to prevent overflow between the catchment basins. As explained in the previous section, a one-pixel-thick dam can be constructed when needed by dilating $q \cap C[n - 1]$ with a 3×3 structuring element of 1's, and constraining the dilation to q.

Algorithm efficiency is improved by using only values of n that correspond to existing gray-level values in $g(x, y)$; we can determine these values, as well as the values of min and max, from the histogram of $g(x, y)$.



a b
c d

FIGURE 10.46 (a) Image of blobs. (b) Image gradient. (c) Watershed lines. (d) Watershed lines superimposed on original image. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

EXAMPLE 10.18: Illustration of the watershed segmentation algorithm.

■ Consider the image and its gradient, shown in Figs. 10.46(a) and (b), respectively. Application of the watershed algorithm just described yielded the watershed lines (white paths) of the gradient image shown in Fig. 10.46(c). These segmentation boundaries are shown superimposed on the original image in Fig. 10.46(d). As noted at the beginning of this section, the segmentation boundaries have the important property of being connected paths. ■

10.5.4 The Use of Markers

Direct application of the watershed segmentation algorithm in the form discussed in the previous section generally leads to *oversegmentation* due to noise and other local irregularities of the gradient. As shown in Fig. 10.47, oversegmentation can be serious enough to render the result of the algorithm virtually useless. In this case, this means a large number of segmented regions. A practical solution to this problem is to limit the number of allowable regions by incorporating a preprocessing stage designed to bring additional knowledge into the segmentation procedure.

An approach used to control oversegmentation is based on the concept of markers. A *marker* is a connected component belonging to an image. We have *internal* markers, associated with objects of interest, and *external* markers, associated with the background. A procedure for marker selection typically will

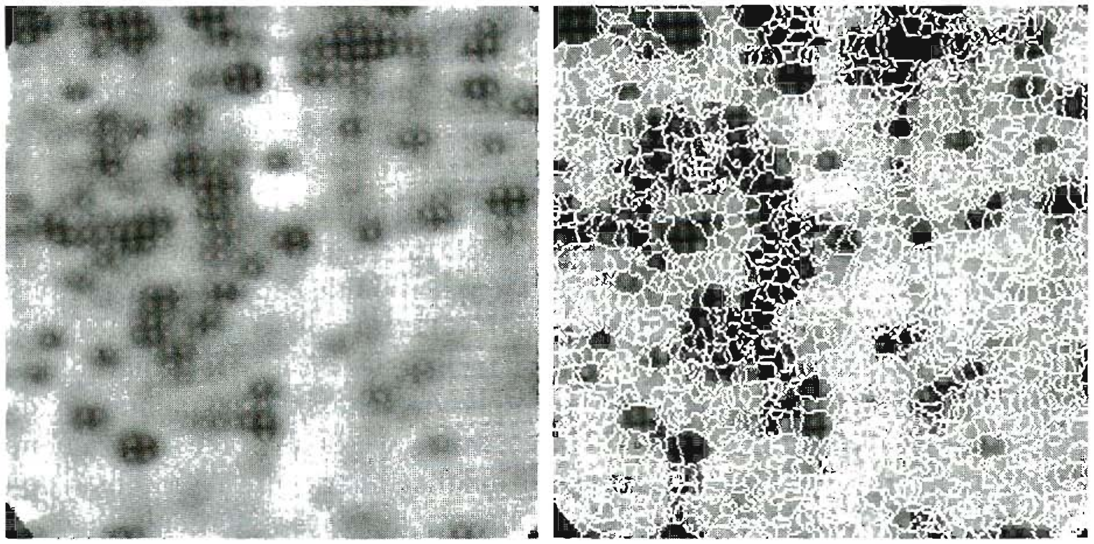

a b

FIGURE 10.47 (a) Electrophoresis image. (b) Result of applying the watershed segmentation algorithm to the gradient image. Oversegmentation is evident. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

consist of two principal steps: (1) preprocessing; and (2) definition of a set of criteria that markers must satisfy. To illustrate, consider Fig. 10.47(a) again. Part of the problem that led to the oversegmented result in Fig. 10.47(b) is the large number of potential minima. Because of their size, many of these minima really are irrelevant detail. As has been pointed out several times in earlier discussions, an effective method for minimizing the effect of small spatial detail is to filter the image with a smoothing filter. This is an appropriate preprocessing scheme in this particular case.

Suppose that we define an internal marker in this case as (1) a region that is surrounded by points of higher "altitude;" (2) such that the points in the region form a connected component; and (3) in which all the points in the connected component have the same gray-level value. After the image was smoothed, the internal markers resulting from this definition are shown as light gray, bloblike regions in Fig. 10.48(a). Next, the watershed algorithm was applied to the smoothed

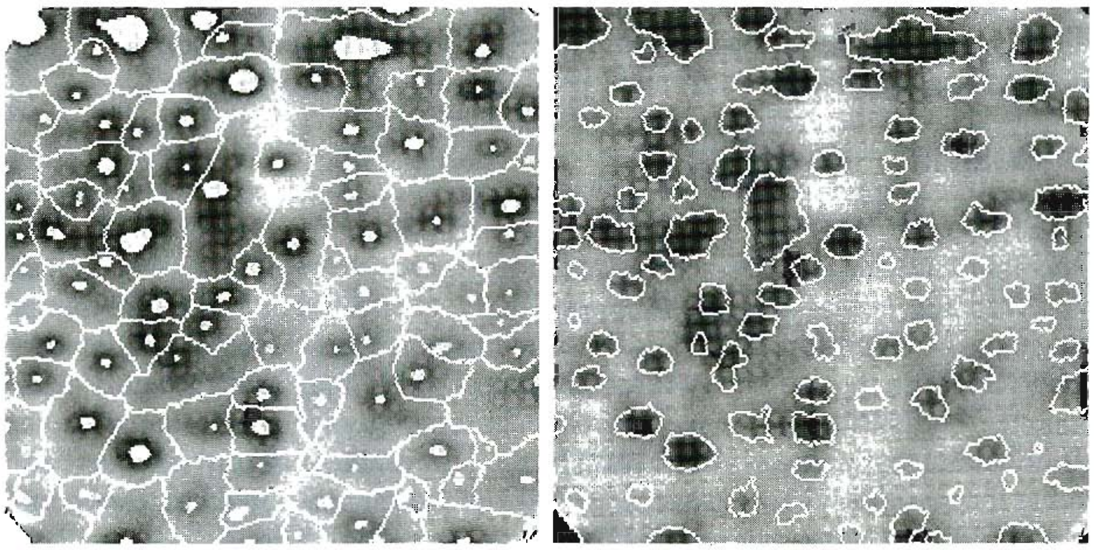

a b

FIGURE 10.48 (a) Image showing internal markers (light gray regions) and external markers (watershed lines). (b) Result of segmentation. Note the improvement over Fig. 10.47(b). (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

image, under the restriction that these internal markers be the *only* allowed regional minima. Figure 10.48(a) shows the resulting watershed lines. These watershed lines are defined as the external markers. Note that the points along the watershed lines are good candidates for the background because they pass along the highest points between neighboring markers.

The external markers shown in Fig. 10.48(a) effectively partition the image into regions, with each region containing a single internal marker and part of the background. The problem is thus reduced to partitioning each of these regions into two: a single object and its background. We can bring to bear on this simplified problem many of the segmentation techniques discussed earlier in this chapter. Another approach is simply to apply the watershed segmentation algorithm to each individual region. In other words, we simply take the gradient of the smoothed image [as in Fig. 10.46(b)] and then restrict the algorithm to operate on a single watershed that contains the marker in that particular region. The result obtained using this approach is shown in 10.48(b). The improvement over the image in 10.47(b) is evident.

Marker selection can range from simple procedures based on gray-level values and connectivity, as was just illustrated, to more complex descriptions involving size, shape, location, relative distances, texture content, and so on (see Chapter 11 regarding descriptors). The point is that using markers brings a priori knowledge to bear on the segmentation problem. The reader is reminded that humans often aid segmentation and higher-level tasks in every-day vision by using a priori knowledge, one of the most familiar being the use of context. Thus, the fact that segmentation by watersheds offers a framework that can make effective use of this type of knowledge is a significant advantage of this method.

10.6 The Use of Motion in Segmentation

Motion is a powerful cue used by humans and many animals to extract objects of interest from a background of irrelevant detail. In imaging applications, motion arises from a relative displacement between the sensing system and the scene being viewed, such as in robotic applications, autonomous navigation, and dynamic scene analysis. In the following sections we consider the use of motion in segmentation both spatially and in the frequency domain.

10.6.1 Spatial Techniques

Basic approach

One of the simplest approaches for detecting changes between two image frames $f(x, y, t_i)$ and $f(x, y, t_j)$ taken at times t_i and t_j, respectively, is to compare the two images pixel by pixel. One procedure for doing this is to form a difference image. Suppose that we have a reference image containing only stationary components. Comparing this image against a subsequent image of the same scene, but including a moving object, results in the difference of the two images canceling the stationary elements, leaving only nonzero entries that correspond to the nonstationary image components.

A difference image between two images taken at times t_i and t_j may be defined as

$$d_{ij}(x, y) = \begin{cases} 1 & \text{if } |f(x, y, t_i) - f(x, y, t_j)| > T \\ 0 & \text{otherwise} \end{cases} \tag{10.6-1}$$

where T is a specified threshold. Note that $d_{ij}(x, y)$ has a value of 1 at spatial coordinates (x, y) only if the gray-level difference between the two images is appreciably different at those coordinates, as determined by the specified threshold T. It is assumed that all images are of the same size. Finally, we note that the values of the coordinates (x, y) in Eq. (10.6-1) span the dimensions of these images, so that the difference image $d_{ij}(x, y)$ also is of same size as the images in the sequence.

In dynamic image processing, all pixels in $d_{ij}(x, y)$ with value 1 are considered the result of object motion. This approach is applicable only if the two images are registered spatially and if the illumination is relatively constant within the bounds established by T. In practice, 1-valued entries in $d_{ij}(x, y)$ often arise as a result of noise. Typically, these entries are isolated points in the difference image, and a simple approach to their removal is to form 4- or 8-connected regions of 1's in $d_{ij}(x, y)$ and then ignore any region that has less than a predetermined number of entries. Although it may result in ignoring small and/or slow-moving objects, this approach improves the chances that the remaining entries in the difference image actually are the result of motion.

Accumulative differences

Isolated entries resulting from noise is not an insignificant problem when trying to extract motion components from a sequence of images. Although the number of these entries can be reduced by a thresholded connectivity analysis, this filtering process can also remove small or slow-moving objects as noted in the previous section. One way to address this problem is by considering changes at a pixel location over several frames, thus introducing a "memory" into the process. The idea is to ignore changes that occur only sporadically over a frame sequence and can therefore be attributed to random noise.

Consider a sequence of image frames $f(x, y, t_1), f(x, y, t_2), \dots, f(x, y, t_n)$ and let $f(x, y, t_1)$ be the reference image. An *accumulative difference image* (ADI) is formed by comparing this reference image with every subsequent image in the sequence. A counter for each pixel location in the accumulative image is incremented every time a difference occurs at that pixel location between the reference and an image in the sequence. Thus when the kth frame is being compared with the reference, the entry in a given pixel of the accumulative image gives the number of times the gray level at that position was different from the corresponding pixel value in the reference image. Differences are established, for example, by using Eq. (10.6-1).

Often useful is consideration of three types of accumulative difference images: *absolute*, *positive*, and *negative* ADIs. Assuming that the gray-level values of the moving objects are larger than the background, these three types of ADIs are defined as follows. Let $R(x, y)$ denote the reference image and, to simplify the notation, let k denote t_k, so that $f(x, y, k) = f(x, y, t_k)$. We assume that $R(x, y) = f(x, y, 1)$. Then, for any $k > 1$, and keeping in mind that the values of the ADIs are *counts*, we define the following for all relevant values of (x, y):

$$A_k(x, y) = \begin{cases} A_{k-1}(x, y) + 1 & \text{if } \left|R(x, y) - f(x, y, k)\right| > T \\ A_{k-1}(x, y) & \text{otherwise} \end{cases} \tag{10.6-2}$$

$$P_k(x, y) = \begin{cases} P_{k-1}(x, y) + 1 & \text{if } \left[R(x, y) - f(x, y, k)\right] > T \\ P_{k-1}(x, y) & \text{otherwise} \end{cases} \tag{10.6-3}$$

and

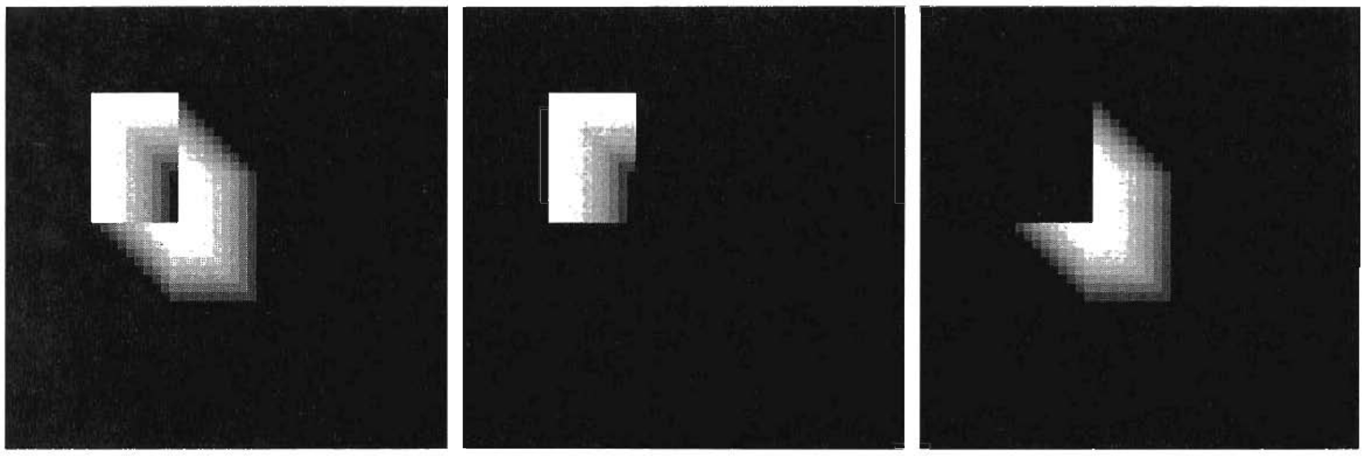
$$N_k(x, y) = \begin{cases} N_{k-1}(x, y) + 1 & \text{if } \left[R(x, y) - f(x, y, k)\right] < -T \\ N_{k-1}(x, y) & \text{otherwise} \end{cases} \tag{10.6-4}$$

where $A_k(x, t)$, $P_k(x, y)$, and $N_k(x, y)$ are the absolute, positive, and negative ADIs, respectively, after the kth image in the sequence is encountered.

It is understood that these ADIs start out with all zero values (counts). Note also that the ADIs are the same size as the images in the sequence. As noted previously, the images in the sequence are all assumed to be of the same size. Finally, we note that the order of the inequalities and signs of the thresholds in Eqs. (10.6-3) and (10.6-4) are reversed if the gray-level values of the background pixels are greater than the levels of the moving objects.

EXAMPLE 10.19: Computation of the absolute, positive, and negative accumulative difference images.

■ Figure 10.49 shows the three ADIs displayed as intensity images for a rectangular object of dimension 75×50 pixels that is moving in a southeasterly direction at a speed of $5\sqrt{2}$ pixels per frame. The images are of size 256×256 pixels. We note the following: (1) The nonzero area of the positive ADI is equal to the size of the moving object. (2) The location of the positive ADI corresponds to the location of the moving object in the reference frame. (3) The number of counts in the positive ADI stops increasing when the moving object is displaced completely with respect to the same object in the reference frame.



a b c

FIGURE 10.49 ADIs of a rectangular object moving in a southeasterly direction. (a) Absolute ADI. (b) Positive ADI. (c) Negative ADI.

(4) The absolute ADI contains the regions of the positive and negative ADI. (5) The direction and speed of the moving object can be determined from the entries in the absolute and negative ADIs.

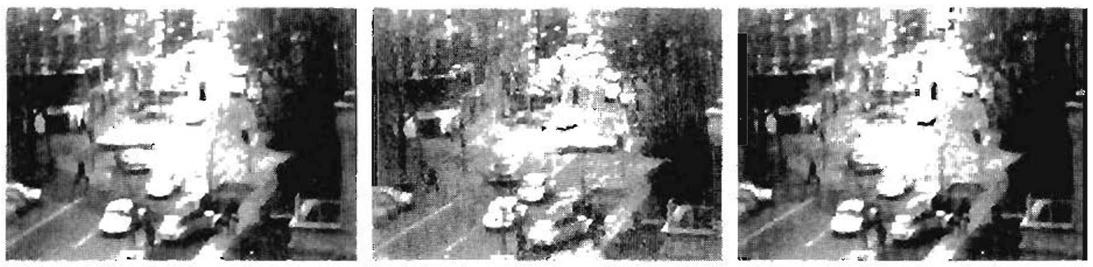
Establishing a reference image

A key to the success of the techniques discussed in the preceding two sections is having a reference image against which subsequent comparisons can be made. As indicated, the difference between two images in a dynamic imaging problem has the tendency to cancel all stationary components, leaving only image elements that correspond to noise and to the moving objects. The noise problem can be handled by the filtering approach mentioned earlier or by forming an accumulative difference image, as discussed in the preceding section.

In practice, obtaining a reference image with only stationary elements is not always possible, and building a reference from a set of images containing one or more moving objects becomes necessary. This necessity applies particularly to situations describing busy scenes or in cases where frequent updating is required. One procedure for generating a reference image is as follows. Consider the first image in a sequence to be the reference image. When a nonstationary component has moved completely out of its position in the reference frame, the corresponding background in the present frame can be duplicated in the location originally occupied by the object in the reference frame. When all moving objects have moved completely out of their original positions, a reference image containing only stationary components will have been created. Object displacement can be established by monitoring the changes in the positive ADI, as indicated in the preceding section.

EXAMPLE 10.20: Building a reference image.

Figures 10.50(a) and (b) show two image frames of a traffic intersection. The first image is considered the reference, and the second depicts the same scene some time later. The objective is to remove the principal moving objects in the reference image in order to create a static image. Although there are other smaller moving objects, the principal moving feature is the automobile at the intersection moving from left to right. For illustrative purposes we focus on this object. By monitoring the changes in the positive ADI, it is possible to determine the initial position of a moving object, as explained previously. Once the area



a b c

FIGURE 10.50 Building a static reference image. (a) and (b) Two frames in a sequence. (c) Eastbound automobile subtracted from (a) and the background restored from the corresponding area in (b). (Jain and Jain.)

occupied by this object is identified, the object can be removed from the image by subtraction. By looking at the frame in the sequence at which the positive ADI stopped changing, we can copy from this image the area previously occupied by the moving object in the initial frame. This area then is pasted onto the image from which the object was cut out, thus restoring the background of that area. If this is done for all moving objects, the result is a reference image with only static components against which we can compare subsequent frames for motion detection, as explained in the previous two sections. The result of removing the east-bound moving vehicle in this case is shown in Fig. 10.50(c). ■

10.6.2 Frequency Domain Techniques

In this section we consider the problem of determining motion estimates via a Fourier transform formulation. Consider a sequence $f(x, y, t)$, $t = 0, 1, \ldots, K - 1$, of K digital image frames of size $M \times N$ generated by a stationary camera. We begin the development by assuming that all frames have a homogeneous background of zero intensity. The exception is a single, 1-pixel object of unit intensity that is moving with constant velocity. Suppose that for frame one ($t = 0$) the image plane is *projected* onto the x-axis; that is, the pixel intensities are summed across the columns. This operation yields a 1-D array with M entries that are 0, except at the location where the object is projected. Multiplying the components of the array by $\exp[j2\pi a_1 x \Delta t]$, $x = 0, 1, 2, \ldots, M - 1$, with the object at coordinates (x', y') at that instant of time, produces a sum equal to $\exp[j2\pi a_1 x' \Delta t]$. In this notation a_1 is a positive integer, and Δt is the time interval between frames.

Suppose that in frame two ($t = 1$) the object has moved to coordinates $(x' + 1, y')$; that is, it has moved 1 pixel parallel to the x-axis. Then repeating the projection procedure discussed in the previous paragraph yields the sum $\exp[j2\pi a_1(x' + 1)\Delta t]$. If the object continues to move 1 pixel location per frame, then, at any integer instant of time, the result is $\exp[j2\pi a_1(x' + t)\Delta t]$, which, using Euler's formula, may be expressed as

$$e^{j2\pi a_1(x'+t)\Delta t} = \cos[2\pi a_1(x' + t)\Delta t] + j\sin[2\pi a_1(x' + t)\Delta t] \quad (10.6\text{-}5)$$

for $t = 0, 1, \ldots, K - 1$. In other words, this procedure yields a complex sinusoid with frequency a_1. If the object were moving v_1 pixels (in the x-direction) between frames, the sinusoid would have frequency $v_1 a_1$. Because t varies between 0 and $K - 1$ in integer increments, restricting a_1 to integer values causes the discrete Fourier transform of the complex sinusoid to have two peaks—one located at frequency $v_1 a_1$ and the other at $K - v_1 a_1$. This latter peak is the result of symmetry in the discrete Fourier transform, as discussed in Section 4.6, and may be ignored. Thus a peak search in the Fourier spectrum yields $v_1 a_1$. Division of this quantity by a_1 yields v_1, which is the velocity component in the x-direction, as the frame rate is assumed to be known. A similar argument would yield v_2, the component of velocity in the y-direction.

A sequence of frames in which no motion takes place produces identical exponential terms, whose Fourier transform would consist of a single peak at a frequency of 0 (a single dc term). Therefore, because the operations discussed so far are linear, the general case involving one or more moving objects in an arbitrary static background would have a Fourier transform with a peak at dc corresponding to static image components and peaks at locations proportional to the velocities of the objects.

These concepts may be summarized as follows. For a sequence of K digital images of size $M \times N$, the sum of the weighted projections onto the x axis at any integer instant of time is

$$g_x(t, a_1) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y, t)e^{j2\pi a_1 x\Delta t} \qquad t = 0, 1, \ldots, K-1. \qquad (10.6\text{-}6)$$

Similarly, the sum of the projections onto the y-axis is

$$g_y(t, a_2) = \sum_{y=0}^{N-1}\sum_{x=0}^{M-1} f(x, y, t)e^{j2\pi a_2 y\Delta t} \qquad t = 0, 1, \ldots, K-1 \qquad (10.6\text{-}7)$$

where, as noted already, a_1 and a_2 are positive integers.

The 1-D Fourier transforms of Eqs. (10.6-6) and (10.6-7), respectively, are

$$G_x(u_1, a_1) = \frac{1}{K}\sum_{t=0}^{K-1} g_x(t, a_1)e^{-j2\pi u_1 t/K} \qquad u_1 = 0, 1, \ldots, K-1 \qquad (10.6\text{-}8)$$

and

$$G_y(u_2, a_2) = \frac{1}{K}\sum_{t=0}^{K-1} g_y(t, a_2)e^{-j2\pi u_2 t/K} \qquad u_2 = 0, 1, \ldots, K-1. \qquad (10.6\text{-}9)$$

In practice, computation of these transforms is carried out using an FFT algorithm, as discussed in Section 4.6.

The frequency-velocity relationship is

$$u_1 = a_1 v_1 \qquad (10.6\text{-}10)$$

and

$$u_2 = a_2 v_2. \qquad (10.6\text{-}11)$$

In this formulation the unit of velocity is in pixels per total frame time. For example, $v_1 = 10$ is interpreted as a motion of 10 pixels in K frames. For frames that are taken uniformly, the actual physical speed depends on the frame rate and the distance between pixels. Thus if $v_1 = 10$, $K = 30$, the frame rate is two images per second, and the distance between pixels is 0.5 m, then the actual physical speed in the x-direction is

$$\begin{aligned} v_1 &= (10 \text{ pixels})(0.5 \text{ m/pixel})(2 \text{ frames/s})/(30 \text{ frames}) \\ &= 1/3 \text{ m/s}. \end{aligned}$$

The sign of the x-component of the velocity is obtained by computing

$$S_{1x} = \left.\frac{d^2\text{Re}\left[g_x(t, a_1)\right]}{dt^2}\right|_{t=n} \qquad (10.6\text{-}12)$$

and

$$S_{2x} = \left. \frac{d^2 \mathrm{Im}\left[g_x(t, a_1)\right]}{dt^2} \right|_{t=n} \qquad (10.6\text{-}13)$$

Because g_x is sinusoidal, it can be shown that S_{1x} and S_{2x} will have the same sign at an arbitrary point in time, n, if the velocity component v_1 is positive. Conversely, opposite signs in S_{1x} and S_{2x} indicate a negative component. If either S_{1x} or S_{2x} is zero, we consider the next closest point in time, $t = n \pm \Delta t$. Similar comments apply to computing the sign of v_2.

EXAMPLE 10.21: Detection of a small moving object via the frequency domain.

■ Figures 10.51 through 10.54 illustrate the effectiveness of the approach just derived. Figure 10.51 shows one of a 32-frame sequence of LANDSAT images generated by adding white noise to a reference image. The sequence contains a superimposed target moving at 0.5 pixel per frame in the x-direction and 1 pixel per frame in the y-direction. The target, shown circled in Fig. 10.52, has a Gaussian intensity distribution spread over a small (9-pixel) area and is not easily discernible by eye. The results of computing Eqs. (10.6-8) and (10.6-9) with $a_1 = 6$ and $a_2 = 4$ are shown in Figs. 10.53 and 10.54, respectively. The peak at $u_1 = 3$ in Fig. 10.53 yields $v_1 = 0.5$ from Eq. (10.6-10). Similarly, the peak at in Fig. 10.54 yields $v_2 = 1.0$ from Eq. (10.6-11). ■

Guidelines for the selection of a_1 and a_2 can be explained with the aid of Figs. 10.53 and 10.54. For instance, suppose that we had used $a_2 = 15$ instead of $a_2 = 4$. In that case the peaks in Fig. 10.54 would now be at $u_2 = 15$ and 17 because $v_2 = 1.0$, which would be a seriously aliased result. As discussed in Section 2.4.4, aliasing is caused by undersampling (too few frames in the present discussion, as the range of u is determined by K). Because $u = av$, one possibility is to select a as the integer closest to $a = u_{\max}/v_{\max}$, where $u_{\max}$ is the aliasing frequency limitation established K and $v_{\max}$ is the maximum expected object velocity.

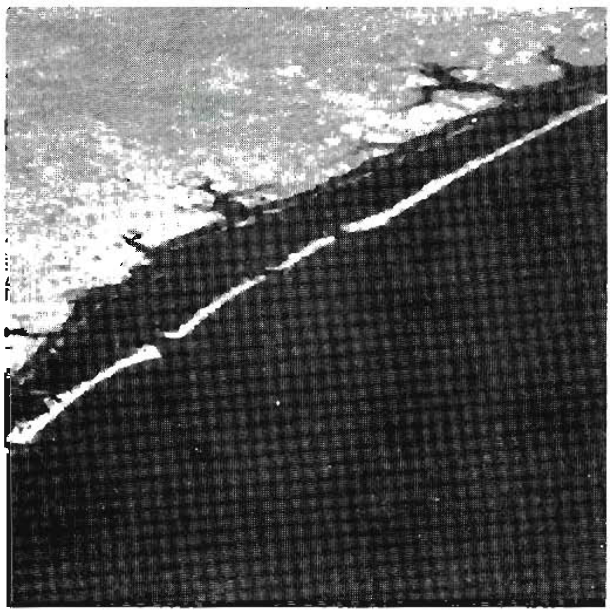


FIGURE 10.51 LANDSAT frame. (Cowart, Snyder, and Ruedger.)

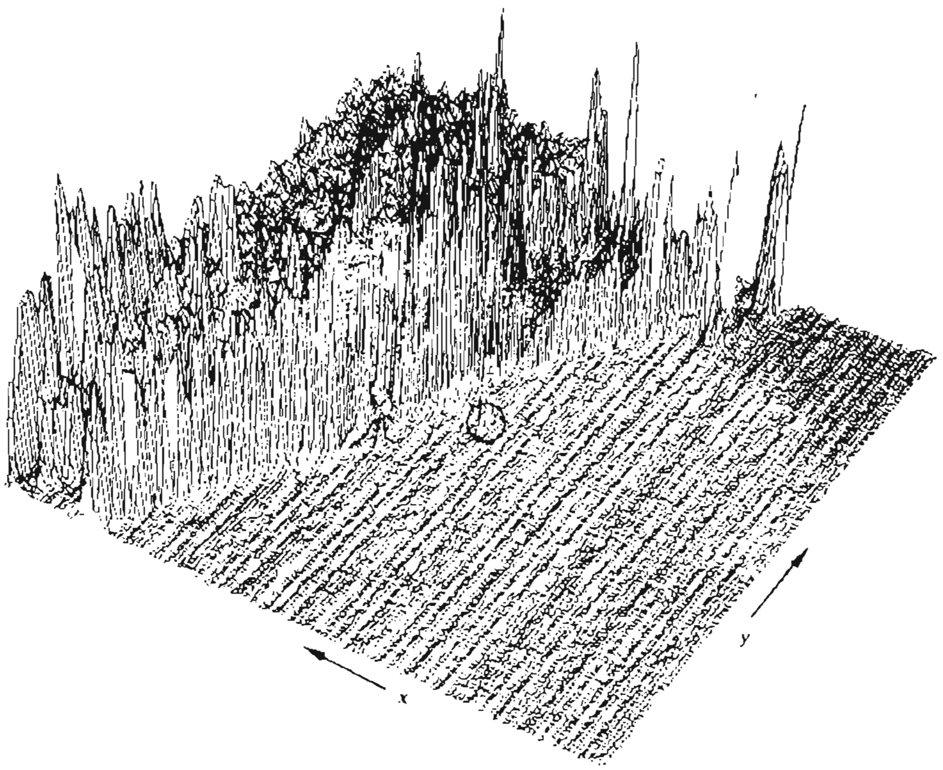


FIGURE 10.52 Intensity plot of the image in Fig. 10.51, with the target circled. (Rajala, Riddle, and Snyder.)

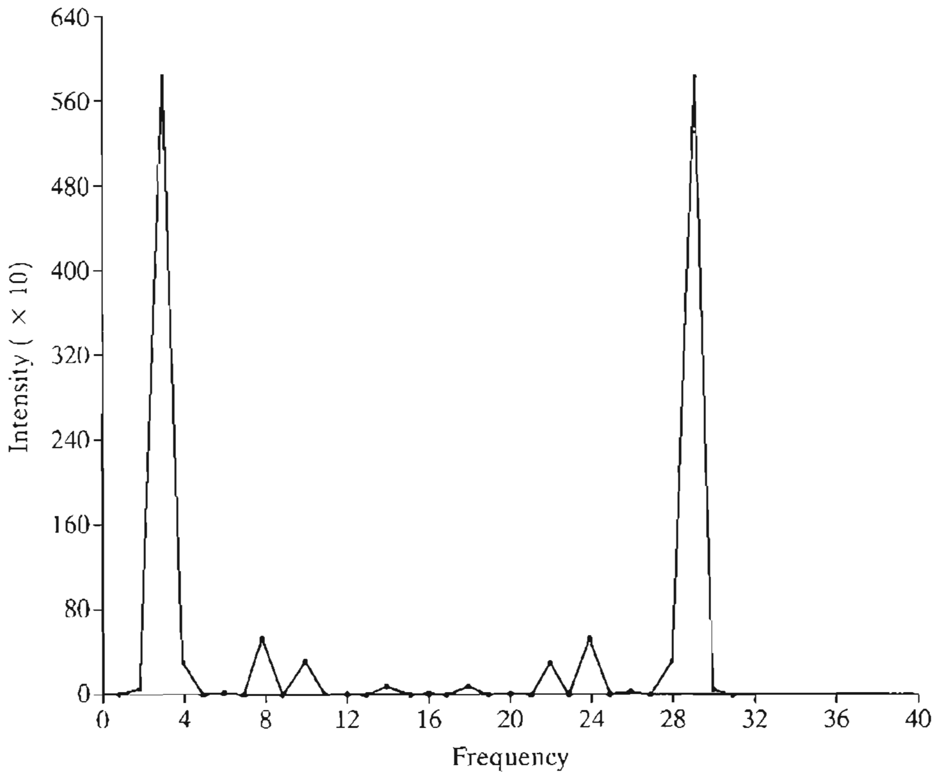


FIGURE 10.53 Spectrum of Eq. (10.6-8) showing a peak at $u_1 = 3$. (Rajala, Riddle, and Snyder.)

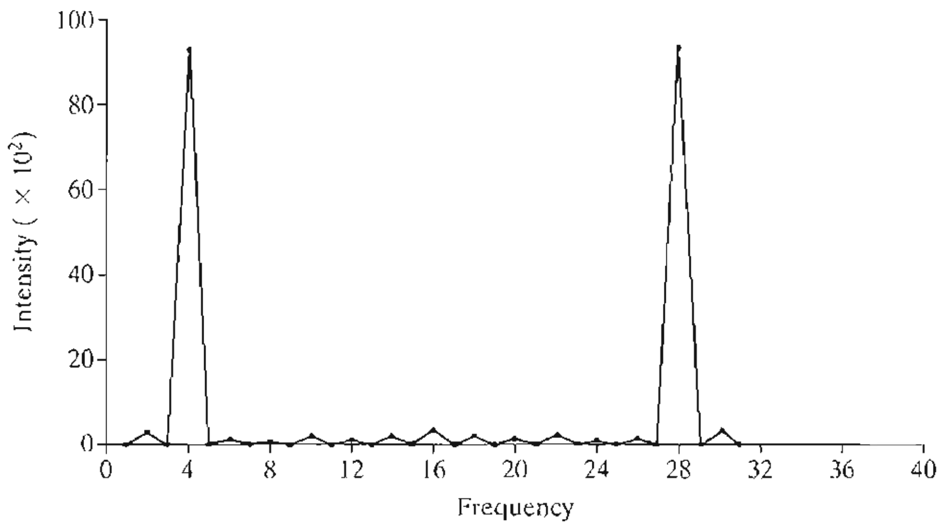


FIGURE 10.54 Spectrum of Eq. (10.6-9) showing a peak at $u_2 = 4$. (Rajala, Riddle, and Snyder.)

Summary

Image segmentation is an essential preliminary step in most automatic pictorial pattern recognition and scene analysis problems. As indicated by the range of examples presented in the previous sections, the choice of one segmentation technique over another is dictated mostly by the peculiar characteristics of the problem being considered. The methods discussed in this chapter, although far from exhaustive, are representative of techniques commonly used in practice. The following references can be used as the basis for further study of this topic.

References and Further Reading

Because of its central role in autonomous image processing, segmentation is a topic covered in most books dealing with image processing, image analysis, and computer vision. The following books provide complementary and/or supplementary reading for our coverage of this topic: Shapiro and Stockman [2001], Sonka et al. [1999], Petrou and Bosdogianni [1999], and Umbaugh [1998].

Work dealing with the use of masks to detect gray-level discontinuities (Section 10.1) has a long history. Numerous masks have been proposed over the years: Roberts [1965], Prewitt [1970], Kirsh [1971], Robinson [1976], Frei and Chen [1977], and Canny [1986]. A review article by Fram and Deutsch [1975] contains numerous masks and an evaluation of their performance. The issue of mask performance, especially for edge detection, still is an area of considerable interest, as exemplified by Qian and Huang [1996], Wang et al. [1996], Heath et al. [1997, 1998], and Ando [2000]. Edge detection on color images has been increasing in popularity for a number of multisensing applications. See, for example, Salinas, Abidi and Gonzalez [1996]; Zugaj and Lattuati [1998]; Mirmehdi and Petrou [2000]; and Plataniotis and Venetsanopoulos [2000]. The interplay between image characteristic and mask performance also is a topic of current interest, as exemplified by Ziou [2001]. Our presentation of the zero-crossing properties of the Laplacian is based on a paper by Marr and Hildredth [1980] and on the book by Marr [1982]. See also a paper by Clark [1989] on authenticating edges produced by zero-crossing algorithms. (Corrections of parts of the Clark paper are given by Piech [1990].) As mentioned in Section 10.1, zero crossing via the Laplacian of a Gaussian is an important approach whose relative performance is still an active topic of research (Gunn [1998, 1999]).

The Hough transform (Hough [1962]) has emerged over the past decade as a method of choice for global pixel linking and curve detection. Numerous generalizations to the basic transform discussed in this chapter have been proposed over the years. For example, Lo and Tsai [1995] discuss an approach for detecting thick lines, Guil et al. [1995, 1997] deal with fast implementations of the Hough transform and detection of primitive curves, Daul at al. [1998] discuss further generalizations for detecting elliptical arcs, and Shapiro [1996] deals with implementation of the Hough transform for gray-scale images. The algorithm presented in Section 10.2.3 is based on Martelli [1972, 1976]. For additional reading on heuristic graph searching, see Nilsson [1980], Umeyama [1988], and Sonka et al. [1999].

As mentioned at the beginning of Section 10.3, thresholding techniques enjoy a significant degree of popularity because they are simple to implement. It is not surprising that there is a considerable body of work reported in the literature on this topic. A good appreciation of the extent of this literature can be gained from the review papers by Sahoo et al. [1988] and by Lee et al. [1990]. We spent a significant level of effort in Section 10.3.2 dealing with the effects of illumination on thresholding. The types of approaches used to deal with this problem are illustrated by the work of Perez and Gonzalez [1987], Parker [1991], Murase and Nayar [1994], Bischsel [1998], and Drew et al. [1999]. For additional reading on the material in Sections 10.3.3 and 10.3.4, see Jain et al. [1995]. The early work of Chow and Kaneko [1972] discussed in Section 10.3.5 is still a standard in terms of illustrating the key aspects of a threshold-based image segmentation solution. Essentially the same can be said for the material presented in Section 10.3.6 (due to White and Rohrer [1983]), which combines thresholding, the gradient, and the Laplacian in the solution of a difficult segmentation problem. It is interesting to compare the fundamental similarities in terms of image segmentation capability between these two articles and work on thresholding done almost twenty years later (Cheriet et al. [1998], Sauvola and Pietikainen [2000]). See also Liang et al. [2000] and Chan et al. [2000] for alternate approaches to the problem of detecting boundaries in images similar in concept to those studied by Chow and Kaneko.

See Fu and Mui [1981] for an early survey on the topic of region-oriented segmentation. The works of Haddon and Boyce [1990] and of Pavlidis and Liow [1990] are among the earliest efforts to integrate region and boundary information for the purpose of segmentation. A newer region-growing approach proposed by Hojjatoleslami and Kittler [1998] also is of interest. For current basic coverage of region-oriented segmentation concepts, see Shapiro and Stockman [2001] and Sonka et al. [1999].

Segmentation by watersheds was shown in Section 10.5 to be a powerful concept. Early references dealing with segmentation by watersheds are Serra [1988], Beucher [1990], and Beucher and Meyer [1992]. The paper by Baccar et al. [1996] discusses segmentation based on data fusion and morphological watersheds. The progress in this field in a little more than one decade is evident in a special issue of *Pattern Recognition* [2000], devoted entirely to this topic. As indicated in our discussion in Section 10.5, one of the key issues with watersheds is the problem of over segmentation. The papers by Najmanand and Schmitt [1996], Haris et al. [1998], and Bleau and Leon [2000] are illustrative of approaches for dealing with this problem. Bieniek and Moga [2000] discuss a watershed segmentation algorithm based on connected components.

The material in Section 10.6.1 is from Jain, R. [1981]. See also Jain, Kasturi, and Schunck [1995]. The material in Section 10.6.2 is from Rajala, Riddle, and Snyder [1983]. See also the papers by Shariat and Price [1990] and by Cumani et al. [1991]. The books by Shapiro and Stockman [2001] and by Sonka et al. [1999] provide additional reading regarding motion estimation.