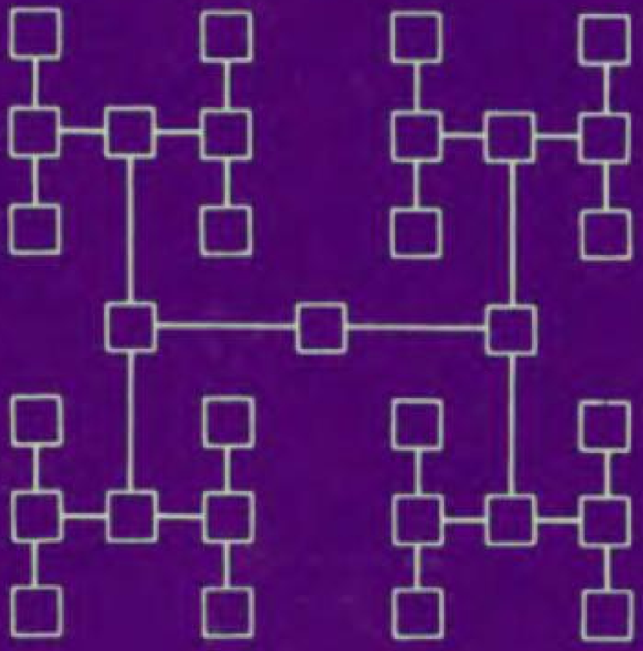
Pattern Recognition and Image Preprocessing

Second Edition, Revised and Expanded



SING-TZE BOW

Signal Processing and Communications Series

Pattern Recognition and Image Preprocessing

Signal Processing and Communications

1. Digital Signal Processing for Multimedia Systems, *edited by Keshab K. Parhi and Takao Nishitani*
2. Multimedia Systems, Standards, and Networks, *edited by Atul Puri and Tsuhan Chen*
3. Embedded Multiprocessors: Scheduling and Synchronization, *Sundararajan Sriram and Shuvra S. Bhattacharyya*
4. Signal Processing for Intelligent Sensor Systems, *David C. Swanson*
5. Compressed Video over Networks, *edited by Ming-Ting Sun and Amy R. Reibman*
6. Modulated Coding for Intersymbol Interference Channels, *Xiang-Gen Xia*
7. Digital Speech Processing, Synthesis, and Recognition: Second Edition, Revised and Expanded, *Sadaoki Furui*
8. Modern Digital Halftoning, *Daniel L. Lau and Gonzalo R. Arce*
9. Blind Equalization and Identification, *Zhi Ding and Ye (Geoffrey) Li*
10. Video Coding for Wireless Communication Systems, *King N. Ngan, Chi W. Yap, and Keng T. Tan*

11. Adaptive Digital Filters: Second Edition, Revised and Expanded, *Maurice G. Bellanger*
12. Design of Digital Video Coding Systems, *Jie Chen, Ut-Va Koc, and K. J. Ray Liu*
13. Programmable Digital Signal Processors: Architecture, Programming, and Applications, *edited by Yu Hen Hu*
14. Pattern Recognition and Image Preprocessing: Second Edition, Revised and Expanded, *Sing-Tze Bow*

Additional Volumes in Preparation

Signal Processing for Magnetic Resonance Imaging and Spectroscopy, *edited by Hong Yan*

Satellite Communication Engineering, *Michael Kolawole*

This Page Intentionally Left Blank

Pattern Recognition and Image Preprocessing

Second Edition, Revised and Expanded

SING-TZE BOW

Northern Illinois University
De Kalb, Illinois

MARCEL DEKKER, INC. NEW YORK • BASEL

ISBN: 0-8247-0659-5

This book is printed on acid-free paper.

Headquarters
Marcel Dekker, Inc.
270 Madison Avenue, New York, NY 10016
tel: 212-696-9000; fax: 212-685-4540

Eastern Hemisphere Distribution
Marcel Dekker AG
Hutgasse 4, Postfach 812, CH-4001 Basel, Switzerland
tel: 41-61-261-8482; fax: 41-61-261-8896

World Wide Web
http://www.dekker.com

The publisher offers discounts on this book when ordered in bulk quantities. For more information, write to Special Sales/Professional Marketing at the headquarters address above.

Current printing (last digit):
10 9 8 7 6 5 4 3 2 1

PRINTED IN THE UNITED STATES OF AMERICA

Series Introduction

Over the past 50 years, digital signal processing has evolved as a major engineering discipline. The fields of signal processing have grown from the origin of fast Fourier transform and digital filter design to statistical spectral analysis and array processing, and image, audio, and multimedia processing, and shaped developments in high-performance VLSI signal processor design. Indeed, there are few fields that enjoy so many applications—signal processing is everywhere in our lives.

When one uses a cellular phone, the voice is compressed, coded, and modulated using signal processing techniques. As a cruise missile winds along hillsides searching for the target, the signal processor is busy processing the images taken along the way. When we are watching a movie in HDTV, millions of audio and video data are being sent to our homes and received with unbelievable fidelity. When scientists compare DNA samples, fast pattern recognition techniques are being used. On and on, one can see the impact of signal processing in almost every engineering and scientific discipline.

Because of the immense importance of signal processing and the fast-growing demands of business and industry, this series on signal processing serves to report up-to-date developments and advances in the field. The topics of interest include but are not limited to the following:

- Signal theory and analysis

- Statistical signal processing
- Speech and audio processing
- Image and video processing
- Multimedia signal processing and technology
- Signal processing for communications
- Signal processing architectures and VLSI design

I hope this series will provide the interested audience with high-quality, state-of-the-art signal processing literature through research monographs, edited books, and rigorously written textbooks by experts in their fields.

K. J. Ray Liu

Preface

This book is based in part on my earlier work, *Pattern Recognition and Image Preprocessing*, which was published in 1992 and reprinted in 1999. At the request of the publisher, in this expanded edition, I am including most of the supplementary materials added to my lectures from year to year since 1992 while I used this book as a text for two courses in pattern recognition and image processing.

Pattern recognition (or pattern classification) can be broadly defined as a process to generate a meaningful description of data and a deeper understanding of a problem through manipulation of a large set of primitive and quantifying data. The set inevitably includes image data—as a matter of fact, some of the data may come directly after the digitization of an actual natural scenic image. Some of that large data set may come from statistics, a document, or graphics, and is eventually expected to be in a visual form. Preprocessing of these data is necessary for error corrections, for image enhancement, and for their understanding and recognition. Preprocessing operations are generally classified as "low-level" operations, while pattern recognition including analysis, description, and understanding of the image (or the large data set), is high-level processing. The strategies and techniques chosen for the low- and high-level processing are interrelated and interdependent. Appropriate acquisition and preprocessing of the original data would alleviate the effort of pattern recognition to some extent. For a specific pattern recognition task, we frequently require a special method for the acquisition of data and its processing. For this reason, I have integrated these two levels of processing into a single book. Together with some exemplary paradigms, this book exposes readers to the whole process in the design of a good pattern recognition system and inspires them to seek applications within their own sphere of influence and personal experience.

Theory and applications are both important topics in the pattern recognition discussion. They are treated on a pragmatic basis in this book. We chose "application" as a vehicle through which to investigate many of the disciplines.

Recently, neural computing has been emerging as a practical technology with successful applications in many fields. The majority of these applications are concerned with problems in pattern recognition. Hence, in this edition we elaborate our discussion of neural networks for pattern recognition, with emphasis on multilayer perceptron, radial basis functions, the Hamming net, the Kohonen self-organizing feature map, and the Hopfield net. These five neural models are presented through simple examples to show the step-by-step procedure for neural computing to help readers start their computer implementation for more complex problems.

The wavelet is a good mathematical tool to extract the local features of variable sizes, variable frequencies, and variable locations in the image; it is very effective in the compression of image data. A new chapter on the wavelet and wavelet transform has been added in this edition. Some work done in our laboratory on wavelet tree-structure-based image compression, wavelet-based morphological processing for image noise reduction, and wavelet-based noise reduction for images with extremely high noise content is presented.

The materials collected for this book are grouped into five parts. Part I emphasizes the principles of decision theoretic pattern recognition. Part II introduces neural networks for pattern recognition. Part III deals with data preprocessing for pictorial pattern recognition. Part IV gives some current examples of applications to inspire readers and interest them in attacking real-world problems in their field with the pattern recognition technique and build their confidence in the capability and feasibility of this technique. Part V discusses some of the practical concerns in image preprocessing and pattern recognition.

Chapter 1 presents the fundamental concept of pattern recognition and its system configuration. Included are brief discussions of selected applications, including weather forecasting, handprinted character recognition, speech recognition, medical analysis, and satellite and aerial-photo interpretation. Chapter 1 also describes and compares the two principal approaches used in pattern recognition, the decision theoretic and syntactic approaches.

The remaining chapters in Part I focus primarily on the decision theoretic approach. Chapter 2 discusses supervised and unsupervised learning in pattern recognition. Chapters 3 and 4 review the principles involved in nonparametric decision theoretic classification and the training of the discriminant functions used in these classifications. Chapter 5 introduces the principles of statistical pattern decision theory in classification.

A great many advances have been made in recent years in the field of clustering (unsupervised learning). Chapter 6 is devoted to the current trends and how to apply these approaches to recognition problems. Chapter 7 discusses dimensionality reduction and the feature selection, which are necessary measures in making machine recognition feasible. In this chapter, attention is given to the following topics: optimal number of features and their ordering, canonical analysis and its application to large data-set problems, principal-component analysis for dimensionality reduction, the optimal classification with Fisher's discriminant, and the nonparametric feature selection method, which is applicable to pattern recognition problems based on mixed features.

Data preprocessing, a very important phase of the pattern recognition system, is the focus of Part III. Emphasis is on the preprocessing of original data for accurate and correct pictorial pattern recognition. Chapters 12, 14, and 15 are devoted primarily to the methodology employed in preprocessing a large data-set problem. Complex problems, such as scenic images, are used for illustration. Processing in spatial domain and transform domain including wavelet is considered in detail. Chapter 13 discusses some prevalent approaches used in pictorial data processing and shape analysis. All these algorithms have already been implemented in our laboratory and evaluated for their effectiveness with real-world problems.

Pattern recognition and image preprocessing can be applied in many different areas to solve existing problems. This is a major reason this discipline has grown so fast. In turn, various requirements posed during the process of resolving practical problems motivate and speed up the development of this discipline. For this reason individual projects are highly recommended to complement course lectures, and readers are highly encouraged to seek applications within their own sphere of influence and personal experience. Although this may cause extra work for the instructors, it is worthwhile to do it for the benefit of the students and of the instructors themselves.

In Part V, we address a problem that is of much concern to pattern recognition and image preprocessing scientists and engineers: The various computer system architectures for the task of image preprocessing and pattern recognition.

A set of sixteen 512 × 512 256 gray-level images is included in Appendix A. These images can be used as large data sets to illustrate many of the pattern recognition and data preprocessing concepts developed in the text. They can be used in their original form or altered to generate a variety of input data sets. Appendices B and C provide some supplementary material on image models and discrete mathematics, respectively, as well as on digital image fundamentals, which can be used as part of lecture material when the digital image preprocessing technique is the main topic of interest in the course.

This book is the outgrowth of two graduate courses—"Principles of Pattern Recognition" and "Digital Image Processing"—which I first developed for the Department of Electrical Engineering at The Pennsylvania State University in 1982 and have updated several times while at the Department of Electrical Engineering at Northern Illinois University since 1987. Much of this material has been used in writing the book, and it is appropriate for both graduate and advanced undergraduate students. This book can be used for a one-semester course on pattern recognition and image preprocessing by omitting some of the material. It can also be used as a two-semester course with the addition of some computer projects similar to those suggested herein. This book will also serve as a reference for engineers and scientists involved with pattern recognition, digital image preprocessing, and artificial intelligence.

I am indebted to Dale M. Grimes, former Head of the Department of Electrical Engineering of The Pennsylvania State University, for his encouragement and support, and to George J. McMurty, Associate Dean of the College of Engineering, The Pennsylvania State University. My thanks also go to Romualdas Kasuba, Dean of the College of Engineering and Engineering Technology, to Darrell E. Newell and Alan Genis, former Chairs of the Department of Electrical Engineering before my term, and to Vincent McGinn, Chair of the Department of Electrical Engineering, all at Northern Illinois University, for their encouragement and support.

I am most grateful to the students who attended my classes, which have been offered twice a year with enrollment of around 20 students in each class since 1987 at Northern Illinois University, and to the students of my off-campus classes in the Chicago area given for high-technology industrial professionals. I thank them for their enthusiastic discussions, both in and out of class, and for writing lengthy programs for performing many experiments. Some of these experiments are included here as end-of-chapter problems, which greatly enrich this book. These programs have been compiled as a software package for student use in the Image Processing Laboratory at Northern Illinois University.

I would also like to express my sincere thanks to Neil Colwell and Keith Lowman of the ArtPhoto Department of Northern Illinois University. Their assistance in putting the images and figures in a very pleasant form is highly appreciated.

Special thanks goes to Rita Lazazzaro and Theresa Dominick, both of Marcel Dekker, Inc., for their enthusiasm in managing this project and excellent, meticulous editing of this book. Without their timely effort this book might still be in preparation.

Hearty appreciation is also extended to Dr. J. L. Koo, the founder of the Shu-ping Memorial Scholarship Foundation, for his kind and constant support in granting me a scholarship for higher education. Without this opportunity, I can hardly imagine how I could have become a professor and scientist, and how I could have published this book.

Finally, I am obliged to Xia-Fang, my dearest, late wife, for her constant encouragement and help during her lifetime. I am very sorry that she is gone, and I miss her. She is always in my heart.

Sing-Tze Bow

This Page Intentionally Left Blank

Contents

Series Introduction *K. J. Ray Liu* *iii*

Preface *v*

PART I. PATTERN RECOGNITION **1**

1. Introduction **3**

1.1 Pattterns and Pattern Recognition 3

1.2 Significance and Potential Function of the Pattern Recognition System 5

1.3 Configuration of the Pattern Recognition System 8

1.4 Representation of Patterns and Approaches to Their Machine Recognition 16

1.5 Paradigm Applications 23

2. Supervised and Unsupervised Learning in Pattern Recognition **29**

3. Nonparametric Decision Theoretic Classification 33

3.1 Decision Surfaces and Discriminant Functions 34
3.2 Linear Discriminant Functions 38
3.3 Piecewise Linear Discriminant Functions 42
3.4 Nonlinear Discriminant Functions 49
3.5 ϕ Machines 52
3.6 Potential Functions as Discriminant Functions 57
Problems 59

4. Nonparametric (Distribution-Free) Training of Discriminant Functions 62

4.1 Weight Space 62
4.2 Error Correction Training Procedures 66
4.3 Gradient Techniques 72
4.4 Training Procedures for the Committee Machine 74
4.5 Practical Considerations Concerning Error Correction Training Methods 76
4.6 Minimum-Squared-Error Procedures 76
Problems 79

5. Statistical Discriminant Functions 82

5.1 Introduction 82
5.2 Problem Formulation by Means of Statistical Design Theory 83
5.3 Optimal Discriminant Functions for Normally Distributed Patterns 93
5.4 Training for Statistical Discriminant Functions 101
5.5 Application to a Large Data-Set Problem: A Practical Example 102
Problems 106

6. Clustering Analysis and Unsupervised Learning 112

6.1 Introduction 112
6.2 Clustering with an Unknown Number of Classes 117
6.3 Clustering with a Known Number of Classes 129
6.4 Evaluation of Clustering Results by Various Algorithms 145
6.5 Graph Theoretical Methods 146
6.6 Mixture Statistics and Unsupervised Learning 161
6.7 Concluding Remarks 164
Problems 164

7. Dimensionality Reduction and Feature Selection **168**

7.1 Optimal Number of Features in Classification of Multivariate Gaussian Data 168
7.2 Feature Ordering by Means of Clustering Transformation 170
7.3 Canonical Analysis and Its Applications to Remote Sensing Problems 172
7.4 Optimum Classification with Fisher's Discriminant 182
7.5 Nonparametric Feature Selection Method Applicable to Mixed Features 188
Problems 190

PART II. NEURAL NETWORKS FOR PATTERN RECOGNITION **197**

8. Multilayer Perceptron **201**

8.1 Some Preliminaries 201
8.2 Pattern Mappings in a Multilayer Perceptron 205
8.3 A Primitive Example 219
Problems 223

9. Radial Basis Function Networks **225**

9.1 Radial Basis Function Networks 225
9.2 RBF Network Training 231
9.3 Formulation of the Radial Basis Functions for Pattern Classification by Means of Statistical Decision Theory 232
9.4 Comparison of RBF Networks with Multilayer Perceptrons 234
Problems 235

10. Hamming Net and Kohonen Self-Organizing Feature Map **236**

10.1 Hamming Net 236
10.2 Kohonen Self-Organizing Feature Map 246
Problems 253

11. The Hopfield Model **256**

11.1 The Hopfield Model 256
11.2 An Illustrative Example for the Explanation of the Hopfield Network Operation 258

11.3 Operation of the Hopfield Network 261
Problems 267

PART III. DATA PREPROCESSING FOR PICTORIAL PATTERN RECOGNITION 269

12. Preprocessing in the Spatial Domain 271

12.1 Deterministic Gray-Level Transformation 271
12.2 Gray-Level Histogram Modification 274
12.3 Smoothing and Noise Elimination 298
12.4 Edge Sharpening 303
12.5 Thinning 333
12.6 Morphological Processing 336
12.7 Boundary Detection and Contour Tracing 343
12.8 Texture and Object Extraction from Textural Background 352
Problems 357

13. Pictorial Data Preprocessing and Shape Analysis 363

13.1 Data Structure and Picture Representation by a Quadtree 363
13.2 Dot-Pattern Processing with Voronoi Approach 365
13.3 Encoding of a Planar Curve by Chain Code 374
13.4 Polygonal Approximation of Curves 374
13.5 Encoding of a Curve with B-Spline 377
13.6 Shape Analysis via Medial Axis Transformation 378
13.7 Shape Discrimination Using Fourier Descriptor 380
13.8 Shape Description via the Use of Critical Points 384
13.9 Shape Description via Concatenated Arcs 385
13.10 Identification of Partially Obscured Objects 392
13.11 Recognizing Partially Occluded Parts by the Concept of Saliency of a Boundary Segment 394
Problems 399

14. Transforms and Image Processing in the Transform Domain 401

14.1 Formulaion of the Image Transform 403
14.2 Functional Properties of the Two-Dimensional Fourier Transform 406
14.3 Sampling 420
14.4 Fast Fourier Transform 437

14.5 Other Image Transforms 454
14.6 Enhancement by Transform Processing 468
Problems 476

15. Wavelets and Wavelet Transform 481

15.1 Introduction 481
15.2 Wavelets and Wavelet Transform 484
15.3 Scaling Function and Wavelet 486
15.4 Filters and Filter Banks 490
15.5 Digital Implementation of DWT 496

PART IV. APPLICATIONS 509

16. Exemplary Applications 511

16.1 Document Image Analysis 513
16.2 Industrial Inspection 529
16.3 Remote Sensing Applications 545
16.4 Vision Used for Control 551

PART V. PRACTICAL CONCERNS OF IMAGE PROCESSING AND PATTERN RECOGNITION 561

17. Computer System Architectures for Image Processing and Pattern Recognition 563

17.1 What We Expect to Achieve from the Point of View of Computer System Architecture 563
17.2 Overview of Specific Logic Processing and Mathematical Computation 564
17.3 Interconnection Networks for SIMD Computers 566
17.4 Systolic Array Architecture 566

Appendix A: Digitized Images 573

Appendix B: Image Model and Discrete Mathematics 579

B.1 Image Model 579
B.2 Simplification of the Continuous Image Model 581
B.3 Two-Dimensional Delta Function 584

B.4 Additive Linear Operators 586
B.5 Convolution 587
B.6 Differential Operators 590
B.7 Preliminaries of Some Methods Used Frequently in Image Preprocessing 591
Problems 595

Appendix C: Digital Image Fundamentals 597

C.1 Sampling and Quantization of an Image 597
C.2 Imaging Geometry 599

Appendix D: Matrix Manipulation 613

D.1 Definition 613
D.2 Matrix Multiplication 614
D.3 Partitioning of Matrices 615
D.4 Computation of the Inverse Matrix 617

Appendix E: Eigenvectors and Eigenvalues of an Operator 621

Appendix F: Notation 625

Bibliography 645
Index 691

Part I

Pattern Recognition

This Page Intentionally Left Blank

1

Introduction

1.1 PATTERNS AND PATTERN RECOGNITION

When we talk about "patterns," very often we refer it to those objects or forms that we can perceive. As a matter of fact, there should be a much broader implication for the word "pattern."

There are good examples to show that a pattern is not necessary confined to be a visible object or form, but a system of data. For example, for the study of the economic situation of a country, we really are talking about the "pattern" of the country's national economy. During the international financial crisis in 1997–1999, some countries suffered very heavy impacts, while some did not. This is because the "patterns" of their national economy are different. Take another example, for the study of weather forecasting, a system of related data are needed. Weather forecasting is based on "patterns" specified on pressure contour maps and radar data over an area. To assure continuous service and economic dispatching of electrical power, bunches of data on various "dispatching patterns" through thorough study on the complicated power system are needed for analysis.

Pattern can then be defined as a quantitative or structural description of an object or some other entity of interest (i.e., not just a visible object, but also a system of data). It follows that a *pattern class* can be defined as a set of patterns that share some properties in common. Since patterns in the same class share some properties in common, we can then easily differentiate buildings of different models. Similarly, we would not have any difficulty to identify alphanumeric characters even when they are of different fonts and with different orientation and size. We can also differentiate men from women; differentiate people who came from west hemisphere from those from east hemisphere; differentiate trucks from cars even with different models. This is because the former ones and the latter ones are defined as different pattern classes for the specific problem.

Pattern recognition is a process of categorizing any sample of measured or observed data as a member of one of the several classes or categories. Due to the fact that pattern recognition is a basic attribute of human beings and other living things, it has been taken for granted for long time. We are now expected to discover the mechanism of their recognition, simulate it, and put it into action with the modern technology to benefit the human beings. This book is dedicated to the design of a system to simulate the recognition of the human being, where the acquisition of information through human sensory organs, processing of this information and making decision through the brain are mainly involved. Pattern recognition is a ramification of artificial intelligence. It is an "interdisciplinary subject." This subject currently challenges scientists and engineers in various disciplines. Electrical and computer scientists and engineers work on this; psychologists, physiologists, biologists, neurophysiologists also work on this. A lot of scientists apply this technology to solve problems in their own field, namely, archaeology, art conservation, astronomy, aviation, chemistry, defense/spy purposes, earth resource management, forensics and criminology, geology, geography, medicine, meteorology, nondestructive testing, oceanography, surveillance, etc. Psychologists, physiologists, biologists, and neurophysiologists devote their effort toward exploring how living things perceive objects. Electrical and computer scientists and engineers, as well as applied mathematicians, devote themselves in the development of the theories and techniques for computer implementation of a given recognition task.

When and where is the *pattern recognition technique* applicable? This technique is useful when (a) normal analysis fails; (b) modeling is inadequate; and (c) simulation is ineffective. Under such situations, pattern recognition technique will be found to be useful and would play an important role.

There are two types of items for recognition:

1. *Recognition of concrete items.* These types of items are visualized and interpreted easier. Among the concrete items are spatial and temporal ones. Examples of spatial items are scenes, pictures, symbols (e.g., traffic symbols), characters (e.g., alphanumeric, Saudi-Arabic character, Chinese characters, etc.), target signatures, road maps, weather maps, speech waveform, ECG, EEG, seismic wave, two-dimensional images, three-dimensional physical objects, etc. Examples of temporal items are real time speech waveform, real time heart beat,

and any other time varying waveforms. Some of those items mentioned above are one-dimensional, e.g., speech waveform, electrocardiogram (ECG), electroencephalogram (EEG), seismic wave, target signature, etc; some of them are two-dimensional, e.g., map, symbol, picture, x-ray images, cytological images, computer tomography images (CT); and some are three-dimensional objects.

2. *Recognition of abstract items* (conceptual recognition). Examples are ideas, arguments, etc. Say, whose idea is the NAFTA (North America Free Trade Agreement)? Many people might recall that this idea was from a person who ran for the U.S. Presidency with Bill Clinton and Bob Dole in 1992. Let us take another example. From the style of writing, can we differentiate a prose from a poem? From the version of a prose, can we identify the Dickens' work from others'? Surely, we can. Since the style of writing is a form of pattern. When we listen to the rhythm, can we differentiate Zhakovski's work from that of Mozart? Surely, we can. The rhythm is a form of pattern. However, recognition of the patterns like those mentioned above (termed conceptual recognition), belongs to another branch of artificial intelligence, and is beyond the scope of this book.

We have to mention here that for the pattern recognition, there is no unifying theory that can be applied to all kinds of pattern recognition problems. Applications tend to be specific and require specific techniques. That is, techniques used are mainly problem oriented. In Part I of this book, basic principles including (1) supervised pattern recognition (with a teacher to train the system), (2) unsupervised pattern recognition (learning by the system itself), and (3) neural network models will be discussed.

1.2 SIGNIFICANCE AND POTENTIAL FUNCTION OF THE PATTERN RECOGNITION SYSTEM

It is not difficult to see that during the twentieth century automation had already liberated human beings from the heavy physical labor in the industry. However, many tasks, which were thought to be light in physical labor, such as parts inspection, including measurements of some important parameters, are still in their primitive human operation stage. As a contrast, they lag behind in efficiency and effectiveness. They even suffer overload to the mass production of products and flooding of graphical documents that need to be handled. Such work involves mainly the acquisition of information through the human sensory organs, especially visual and audio sensing organs; the processing of this information and decision making through the brain. This is really the function of the automation of the pattern recognition.

Application of the pattern recognition is very wide. It can be applied, in theory, to any situation in which the visual and/or audio information is needed in a decision process. Take, as an example, mail sorting. This job does not look heavy in comparison with the steel manufacturing. But the steel-manufacturing plant is highly automated, and mail-sorting work becomes monotonous and boring. If the pattern recognition technique were used to replace human operator to identify the name and address of addressee on the envelope, the efficiency and the effectiveness of the mail sorting would be highly increased.

Automation on the laboratory examination of routine medical images such as (a) chest x-rays for pneumoconiosis and (b) cervical smears and mammograms for the detection of precancer or early stage of cancer is another important application area. It is also possible to screen out those inflammable abnormal cells which look very much like cancerous cells under the microscope.

Aerial and satellite photointerpretation on the ground information is another important application of the pattern recognition. Among the applications in this field are (a) crop forecast and (b) analysis of cloud patterns, etc. Some paradigm applications are given at the end of this chapter and at the end of this book. Aside from these, there are many other applications, especially at a time when we are interested in the global economy.

1.2.1 Modes of Pattern Recognition System

The pattern recognition system that we have so far can be categorized into the following modes.

1. *The system is developed to transform a scene into another which is more suitable for the human to recognize (or understand) the scene.* Various kinds of interference might be introduced during the process of acquiring an image. The interference may come from the outside medium and also from the sensor itself (i.e., the acquiring device). Some techniques need to be developed to improve the image and even to recover the original appearance of the object. This image processing involves a sequence of operations performed upon the numerical representation of objects in order to achieve a "desired" result. In the case of a picture, the image processing changes its form into a modified version so as to make it more desirable for identification purpose. For example, if we want to understand what is in the noisy image shown in Figure 1.1a, we have to first improve the image to the one shown in Figure 1.1b, from which we can then visualize the scene.

2. *The system is developed to enhance a scene for human awareness and also for human reaction if needed.* An example of this application can be found in the identification of a moving tank in the battlefield from the air. Target range and target size must be determined. Some aspects on the target, including its shape and the symbols printed on the target, are useful to distinguish the enemy one from the friendly one. Information such as how fast the target is moving and along which direction is it moving is also needed. In addition, factors influencing

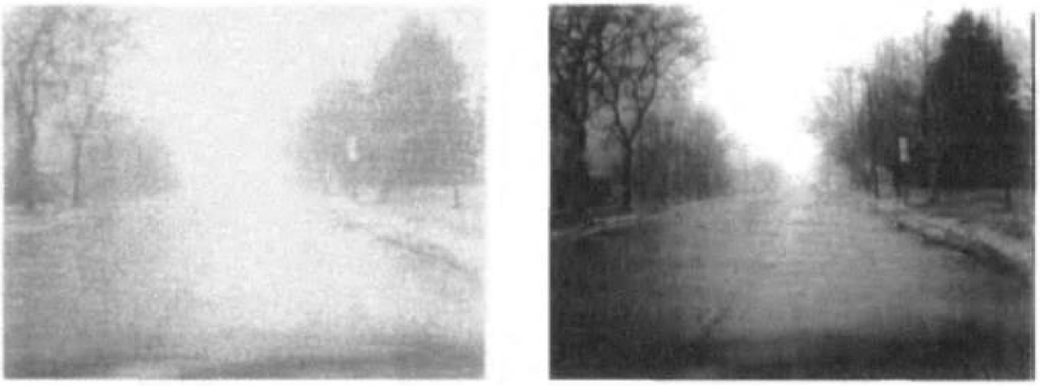


FIGURE 1.1 (a) A scenic image taken during foggy morning. (b) Processed image with an image enhancer.

the correct identification, e.g., background radiance, smoking environment, target/background contrast, stealth, etc. should also be taken into consideration.

3. *The system is developed to complete a task in hand.* To achieve noiseless transmission, the teeth on a pair of gear and pinion should match precisely. Usually this job rests on the human operator with his/her hands, eye, and brain. It is possible, however, to design a computer inspection system with pattern recognition technique to relieve the human inspector in doing this tedious and monotonous work. The pattern recognition system can also be designed for industrial parts structure verification, and for "bin-picking" in industy. Bin-picking uses an artificial vision system to help retrieve components that have been randomly stored in a bin. Another example is the metrological checking and structural verification of hot steel products at a remote distance in a steel mill.

4. *The system is developed for the initiation of subsequent action to optimize the image acquisition or image feature extraction.* Autonomous control of image sensor as described in Chapter 16 (Paradigm Applications) for optimal acquisition of ground information for dynamic analysis is a good example. It is agreed that it is very effective and also very beneficial and favorable to acquire ground information from a satellite for either military or civilian purposes. However, due to the fixed orbit of the satellite and the fixed scanning mode of the multispectral scanner (MSS), the way in which the satellite acquires ground information is in the form of a swath. It is known that two consecutive swaths of information scanned are not contiguously geographically. In addition, two geographically contiguous swaths are scanned at times that differ by several days. It happens that the target area of greatest interest falls either to the left or right outside the current swath. Postflight matching of two or three swaths is thus unavoidable for target interpretation, and therefore on-line processing will not be possible. Off-line processing will be all right (very inefficient, though) when dealing only with a static target. But the situation will become very serious if the

information sought is for the dynamic analysis of strategic military deployment, for example. Even when monitoring a slowly changing flood, information obtained in this way would be of little use.

A desire has thus arisen to enlarge the viewing range of the scanner by means of pattern recognition technique in order to acquire in a single flight all the ground information of interest now located across two or three swaths. This would not only permit on-time acquisition and on-line processing of the relevant time-varying scene information, but would save a lot of postflight ground processing See Chapter 16 for details.

Systems like this can have many applications. It can be designed in the form of an open loop and also a closed loop. If the processed scene is for human reference only, it is an open-loop system. If the processed image is used to help a robot to travel around the room under a seriously hazardous environment, a closed-loop system will be more suitable for the mobile robot.

To summarize, a pattern recognition system can be designed in any one of the above mentioned four modes to suit different applications. A pattern recognition system, in general, consists of image acquisition, image data preprocessing, image segmentation, feature extraction, and object classification. Results may be used for interpretation or for actuation. Image display in spatial and transform domain at intermediate stages is also an important functional process of the system.

1.3 CONFIGURATION OF THE PATTERN RECOGNITION SYSTEM

1.3.1 Three Phases in Pattern Recognition

In pattern recognition we can divide an entire task into three phases: data acquisition, data preprocessing, and decision classification, as shown in Figure 1.2. In the data acquisition phase, analog data from the physical world are gathered through a transducer and converted to digital format suitable for computer processing. In this stage, the physical variables are converted into a set of measured data, indicated in the figure by electrical signal $x(r)$ if the physical variables are sound (or light intensity) and the transducer is a microphone (or photocells). The measured data are then used as the input to the second phase (data preprocessing) and grouped into a set of characteristic features x_N as output. The third phase is actually a classifier that is in the form of a set of decision functions. With this set of features x_N the object may be classified. In Figure 1.2 the set of data at B, C, and D are in the *pattern space*, *feature space*, and *classification space*, respectively.

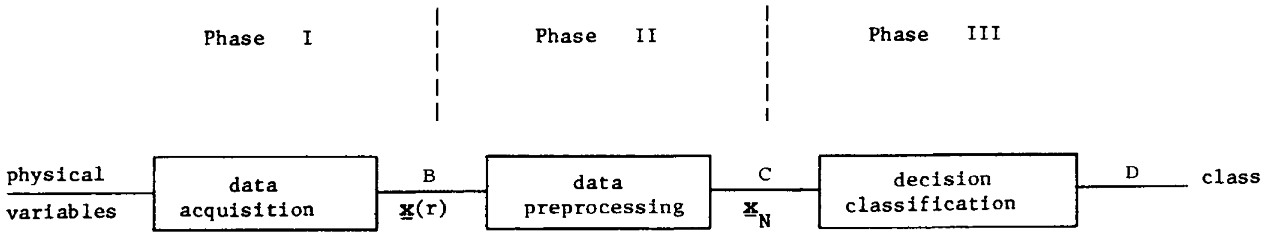


FIGURE 1.2 Conceptual representation of a pattern recognition system.

The data-preprocessing phase includes the process of feature extraction. The reason of including this feature extraction in this phase is simply because the amount of data we have obtained in the data acquisition phase is tremendous and must be reduced to a manageable amount but still carry enough discriminatory information for identification.

1.3.2 Feature Extraction—An Important Component in Pattern Recognition

Necessity of the Data Reduction

To process an image with a computer, we first need to digitize the image in the X direction and also in the Y direction. The finer the digitization, the more vividly close to the original will be the image. This is what we call the spatial resolution. In addition, the larger the number of gray levels used for the quantization of the image function, the more details will be shown in the display. Assume we have an image of size 4×4 in. and would like to have a spatial resolution 500 dpi (dots per inch) and 256 gray levels for image function quantization; we will have $2048 \times 2048 \times 8$ or 33.55 million bits for the representation of a single image. The data amount is very extensive.

The most commonly used and the simplest basic approach for image processing is the convolution of an image with an array $n \times n$ (mask). Let us choose n equal to 3 as an example. There will be 9 multiplication-and-addition operations (or 18 floating-point mathematical operations) for each of the 2048×2048 or 4.19 million pixels, totaling to 75.5×10^6 mathematical operations. Assuming that 6 processes are required for the completion of the specific image processing job, we would need to perform 75.5 million$\times$6 or 453×10^6 operations—very high computational complexity. Say, in average, 20-pulse duration time is needed for each mathematical operation and the Pentium III 500 MHz computer (state of the art technology) is used for the system. Then, $(453 \times 10^6 \times 20)/(500 \times 10^6)$ or 18 s will be needed for the mathematical computation of a single image without taking into consideration the time needed for the data transfer between the CPU and the memory during the processing. This amount of time will, no doubt, be much longer than the CPU time, and may be 20 times as much. In order to speed up the processing of an image, it is therefore necessary to explore a way to accurately represent the image with much less amount of data but without losing any important information for its interpretation.

FeaturesThat Could Best Identify Objects

It is known that when an image is processed through a human vision system, the human vision system does not visualize the image (or an object) pixel by pixel.

Instead, the human vision system extracts some key information that is created through grouping related pixels together to form features. Features are in the form of a list of description called *feature vector*, much less in number but carrying enough discriminating information for identification.

Images containing objects that have been categorized. Proper selection of features is crucial. A set of features may be very effective for one application, but may be of little use for another application. The object (pattern) classification problem is more or less problem oriented. A proper set of features would come out through thorough studies on the object, the preliminary selection of the possible and available features and final sorting out the most effective ones after evaluating each of these features for its effectiveness in the classification. See Chapter 7 (Dimensionality Reduction and Feature Selection) for a detailed discussion on feature ordering.

For objects that have been categorized, *feature* can be referred to as parts of the image with some special properties. Lines, curves, and texture regions are examples. They are *local features*, so called to differentiate it from *global feature* such as average gray level. As a local feature, it should be local, meaningful, and detectable parts of the image. By *meaningful* we mean that the features are associated to interesting scene element via the image formation process. By *detectable* we mean that location algorithms must exist to output a collection of feature descriptors, which specify the position and other essential properties of the features found in the image. See the example given in Section 1.2 which describes the precise matching of gears and pinions. Our concerns focus on whether the pitches between teeth and the profiles of the teeth are the same (or at least within a tolerance) in both the gears and the pinions. Our problem is now to extract these local features for their structural verification.

Figure 1.3 shows a microscopic image of a vaginal smear, where (a) shows the shape of normal cells, while (b) shows that of abnormal cells. A computer image processing system with microscope can be developed to automate the screening of the abnormal cells from the normal ones during general physical examination.

There are many other applications that fall into this category, for instance, the recognition of the alphanumeric characters, bin-picking of manufactured parts by robots, etc.

Scenic images containing objects best represented by their spectral characteristics. Many objects that are not human-made, cannot be well represented by their shapes, especially for those objects that are continuously growing with time. Agricultural products are good examples. For those objects some other features should be extracted for identification. Research shows that different agricultural objects possess different spectral characteristics. Agricultural products such as corn, wheat, and bean respond differently to the various

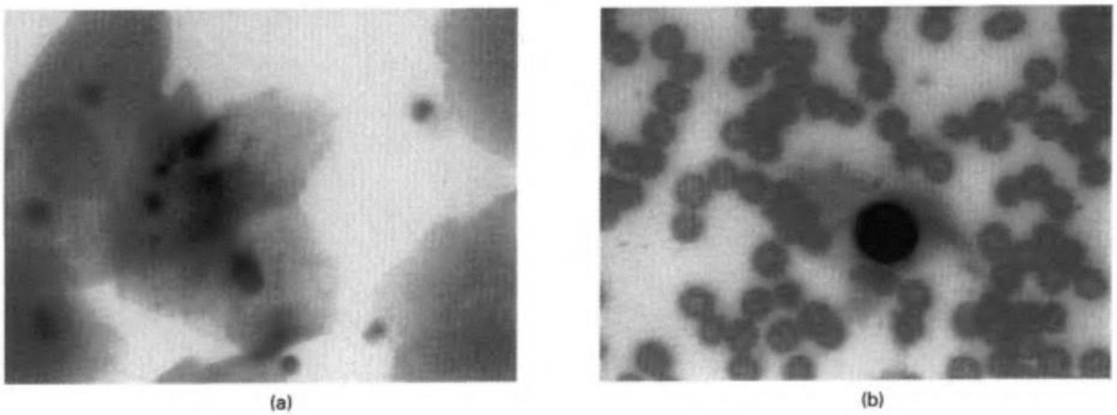


FIGURE 1.3 Cytological images of vaginal smears. (a) Normal vaginal exfoliated cells, pink and light blue colors represent respectively the acidophilic and cosinophilic characteristics of these cells. The original image is a color image. (b) Cancerous cells with red blood cells on the background. The original image is a color one.

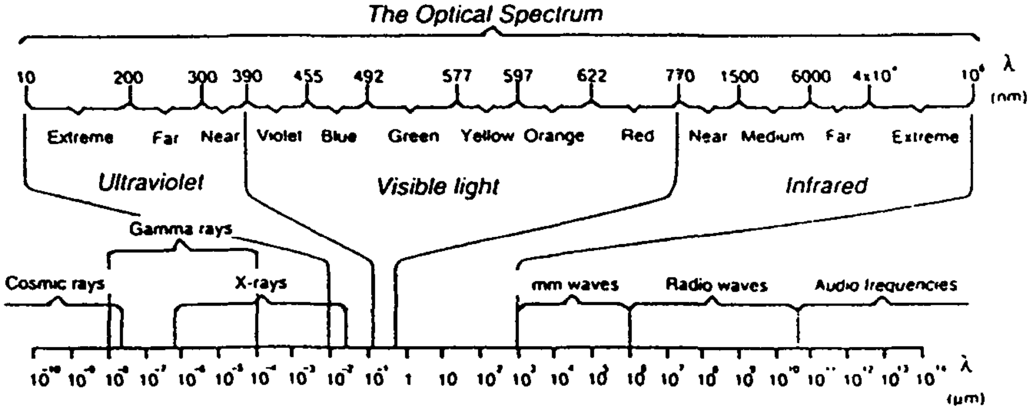


FIGURE 1.4 The optical spectrum ın perspective.

wavebands of the optical electromagnetic wave spectrum (see Figure 1.4). For this reason, strength of responses in some particular wavebands can then be selected as feature(s) for classification.

Remote sensing is concerned with collecting data about the earth and its environment by means of visible and nonvisible electromagnetic radiation. Multispectral data are gathered, with as many as 24 (even more) bands being acquired simultaneously. Information on ultraviolet, visible, infrared, and thermal wavelengths are collected by passive sensors, e.g., multispectral scanners (MSS). Active sensors exploit microwave radiation in the form of synthetic aperture radar (SAR). This can detect objects that are invisible to optical cameras.

Multispectral sensors (satellite or airborne) provide data in the form of several images of the same area on the Earth's surface, through different spectral bands. For a specific application, selection of information from few spectral bands might be sufficient. Effective classification rests on smart choice of the spectral bands, not necessary to be large in number. What is important is to select the most important ones from them for a particular application to reduce the number of features and at the same time retain all or most of the class discriminatory power. Assume that three proper features have already been selected for the above-mentioned crop-type problem. Then, a three-dimensional graph can be plotted in which pixels corresponding to different classes of crop (corn, wheat, bean) will cluster together in the three-dimensional space as three distinct clusters and they will be clearly separated from each other as indicated in Figure 1.5. The classification problem then becomes finding the clusters and the separating boundaries between all these classes. The yearly yields of each of these agricultural products can then be estimated, respectively, from their volumes in the three-dimensional image.

Beyond the estimation of the agricultural crop estimation, there are many fields that can benefit from remote sensing technology. To name a few, this

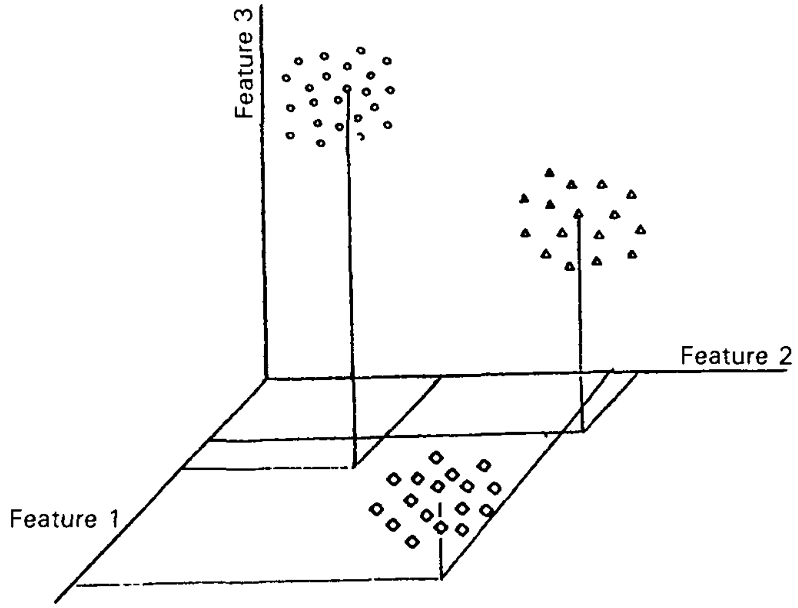


FIGURE 1.5 Predicting the yearly yields of agricultural product via satellite image.

technique has been successfully used to survey large areas of inaccessible and even unmapped land to identify new resources of mineral wealth. This technique has also been used to monitor large or remote tracts of land to determine its existing use or future prospects. Satellite data are very useful for short-term weather forecasting, and important in the study of long-term climate changes such as global warming.

Feature Extraction

By feature extraction we mean to identify the inherent characteristics found within the image acquired. These characteristics (or features, as we usually call them) are used to describe the object, or attributes of the object. Feature extraction operates on a two-dimensional image array and produces a *feature vector*.

Feature directly extracted from pixels. Extraction of features is to convert the image data format from spatially correlated arrays to textual descriptions of structural and/or semantic knowledge. We first bilevel the image, and then group the pixels together with the 8-connectivity convention. Check and see whether it provides some meaningful information. Many of the features of interest are concerned with the shape of the object. Shape of an object or region within an image can be represented by features gleaned from the boundary properties of the shape and/or from the regional properties. For example, structural verification of a pinion could utilize features like diameter of the pinion, number of teeth in the pinion, pitch between the teeth, and the contour shape of the teeth.

Derived features. For some applications, it is more convenient and effective to use computed parameters as features for classification. Such features are called *derived features.* Shape factor (perimeter^2/area) is one of them. It is a dimensionless quantity, invariant of scale, rotation as well as translation, making it a useful and effective feature for identifying various shapes like circle, square, triangle, and ellipse.

Moments are also examples of derived features. Moments can be used to describe the properties of an object in terms of its area, position, and orientation. Let $f(x, y)$ represent the image function or the brightness of the pixel, either 0 (black) or 1 (white); x and y are the pixel coordinates relative to the origin. The zero- and first-order moments can be defined as

$m_{00} = \sum\sum f(x, y)$	Zero-moment, it is the same as the object area for a binary image
$m_{10} = \sum\sum x \cdot f(x, y)$	First-order moment with respect to y axis
$m_{01} = \sum\sum y \cdot f(x, y)$	First-order moment with respect to x axis

Centroid (center of area, or center of mass), a good parameter for specifying the location of an object, can be expressed in terms of moments as

$$x' = \frac{m_{10}}{m_{00}} \quad \text{and} \quad y' = \frac{m_{01}}{m_{00}}$$

where x' and y' are, respectively, the coordinates of the centroid with respect to the origin.

Features obtained from spectral responses. Most real-world images are not monochromatic, but full color. A body will appear white to the observer when it reflects light that is relatively balanced in all visible wavelengths. On the other hand, a body that favors reflectance in a limited range of visible spectrum will exhibit some shades of color. All colors to the human eye are seen as variable combinations of three so-called *primary colors*, red (R), green (G), and blue (B). However, these three R, G, and B sensors in the human eye overlap considerably. For the purpose of image processing, a composite color image can be decomposed into three component images, one in red, one in green, and one in blue. These three component images can be processed separately, and then recombined to form a new image for various applications.

When we scan an image with a 12-channel multispectral scanner, we obtain, for a single picture point, 12 values, each corresponding to a separate spectral response. The pattern $\mathbf{x}$ will be a vector of 12 elements in a 12-

dimensional space. Twelve images will be produced from one scan. Each image corresponds to a separate spectral band.

$$\mathbf{x} = \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_{12} \end{vmatrix}$$

1.4 REPRESENTATION OF PATTERNS AND APPROACHES TO THEIR MACHINE RECOGNITION

1.4.1 Patterns Represented in Multidimensional Vector Form

As discussed in Section 1.3.1, there will be a set of collected, measured data after data acquisition. If the data to be analyzed are physical objects or images, the data acquisition device can be a television camera, a high-resolution camera, a multispectral scanner, or other device. For other types of problems, such as economic problems, the data acquisition system can be a data type.

One function of data preprocessing is to convert a visual pattern into an electrical pattern or to convert a set of discrete data into a mathematical pattern so that those data are more suitable for computer analysis. The output will then be a *pattern vector*, which appears as a point in a *pattern space*.

To clarify this idea, let us make a simple visual image as the system input. If we scan an image with a 12-channel multispectral scanner, we obtain, for a single picture point, 12 values, each corresponding to a separate spectral response. If the image is treated as a color image, three fundamental color-component values can be obtained, each corresponding, respectively, to a red, green, or blue spectrum band.

Each spectrum component value can be considered as a variable in n-dimensional space, known as pattern space, where each spectrum component is assigned to a dimension. Each pattern then appears as a point in the pattern space. It is a vector composed of n component values in the n-dimensional coordinates. A pattern $\mathbf{x}$ can then be represented as

$$\mathbf{x} = \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix}$$

where the subscript n represents the number of dimensions. If $n < 3$, the space can be illustrated graphically. Pattern space $\mathbf{X}$ may be described by a vector of m pattern vectors such that

$$\mathbf{X} = \begin{vmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_m^T \end{vmatrix} = \begin{vmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & & \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{vmatrix}$$

where the superscript T after each vector denotes its transpose, the $\mathbf{x}_i^T = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, 2, \ldots, m$, represent pattern vectors.

The objective of the feature extraction shown in Figure 1.6 is to function as the dimensionality reduction (see Section 1.3.2). It converts the original data to a suitable form (feature vectors) for use as input to the decision processor for classification. Obviously, the feature vectors represented by

$$\mathbf{x}_i^T = (x_{i1}, x_{i2}, \ldots, x_{ir}) \qquad i = 1, 2, \ldots, m$$

are in a smaller dimension (i.e., $r < n$).

The decision processor shown in Figure 1.6 operates on the feature vector and yields a classification decision. As we discussed before, pattern vectors are placed in the pattern space as "points," and patterns belonging to the same class will cluster together. Each cluster represents a distinct class, and clusters of points represent different classes of patterns. The decision classifier implemented with a set of decision function serves to define the class to which a particular pattern belongs.

The inputs to the decision processor are a set of feature data (or feature vectors). The output of the decision processor is in the classification space. It is M-dimensional if the input patterns are to be classified into M classes. For the simplest two-class problem, M equals 2; for aerial-photo interpretation, M can be 10 or more; and for alphabet recognition M equals 26. But for the case of Chinese character recognition, M can be more than 10,000. In such a case, other representations have to be used as supplements.

Both the preprocessor and the decision processor are usually selected by the user or designer. The decision function used may be linear, piecewise linear, nonlinear, or some other kind of functions. The coefficients (or weights) used in the decision processor are either calculated on the basis of complete a priori information of statistics of patterns to be classified, or are adjusted during a training phase. During the training phase, a set of patterns from a training set is presented to the decision processor, and the coefficients are adjusted according to whether the classification of each pattern is correct or not. This may then be called an *adaptive* or *training* decision processor. Note that most of the pattern recognition systems are not adaptive on-line. On-line pattern recognition systems

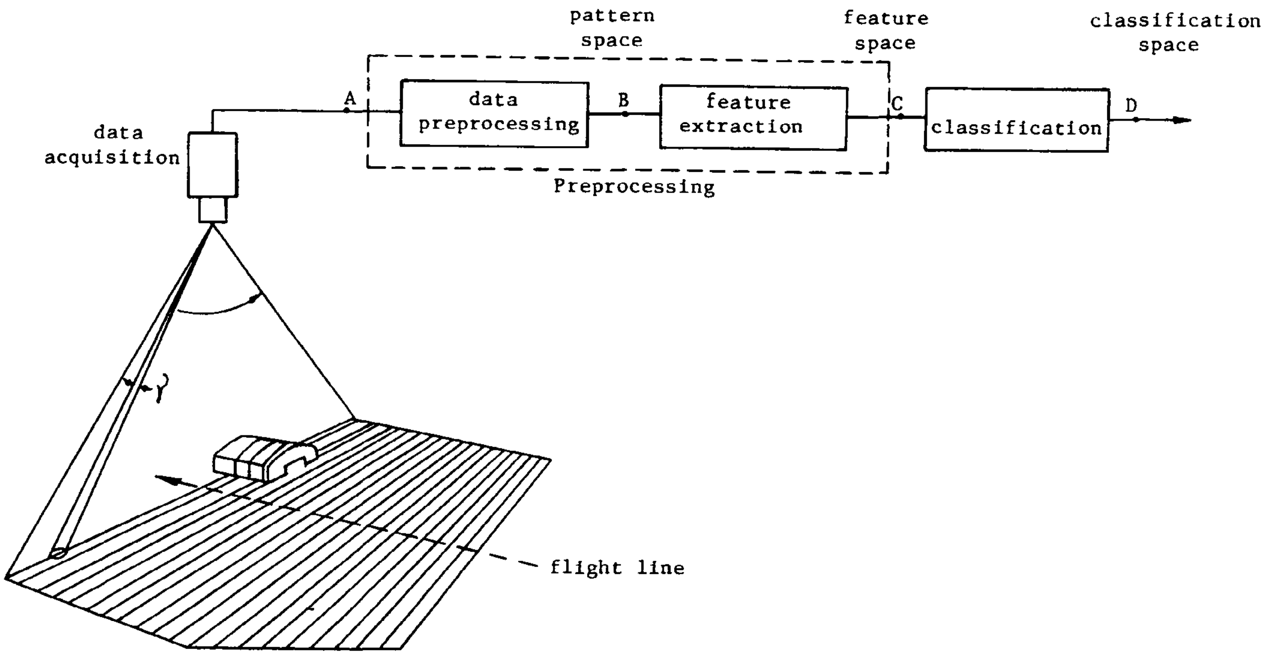


FIGURE 1.6 Multispectral scanner and data analysis system.

are being developed. Note also that the preprocessing and decision algorithms should not be isolated from each other. Frequently, the preprocessing scheme has to be changed to make the decision processing more effective. Some attempts have been made to simulate the human recognition system. Human recognition system has the capabilities, of association, categorization, generalization, classification, feature extraction, and optimization. These capabilities fall into three broad categories, namely, (1) searching, (2) representation, and (3) learning. What we try to do is to design a system that will be as capable as possible.

As discussed previously, a priori knowledge as to correct classification of some data vectors is needed in the training phase of the decision processor. Such data vectors are referred to as prototypes and are denoted as

$$\mathbf{z}_k^m = \begin{vmatrix} z_{k1}^m \\ \vdots \\ z_{ki}^m \\ \vdots \\ z_{kn}^m \end{vmatrix} \qquad \begin{aligned} k &= 1, 2, \dots, M \\ m &= 1, 2, \dots, N_k \end{aligned}$$

where $k = 1, 2, \dots, M$ indexes the particular pattern class; $m = 1, 2, \dots, N_k$ indicates the mth prototype of the class ω_k; and $i = 1, 2, \dots, n$ indexes its component in the n-dimensional pattern vector. M, N_k, and n denote, respectively, the number of pattern classes, the number of prototypes in the kth class ω_k, and the number of dimensions of the pattern vectors.

Prototypes from the same class share the same common properties and thus they cluster in a certain region of the pattern space. Figure 1.7 shows a simple two-dimensional pattern space. Prototypes $\mathbf{z}_1^1$, $\mathbf{z}_1^2, \dots, \mathbf{z}_1^{N1}$ cluster in ω_1; prototypes of another class, $\mathbf{z}_2^1$, $\mathbf{z}_2^2, \dots, \mathbf{z}_2^{N2}$, cluster in another region of the pattern space ω_2. N_1 and N_2 are the number of prototypes in classes ω_1 and ω_2, respectively. The classification problem will simply be to find a separating surface that partitions the known prototypes into correct classes. This separating surface is expected to be able to classify the other unknown patterns if the same criterion is used in the classifier. Since patterns belonging to different classes will cluster into different regions in the pattern space, the distance metric between patterns can be used as a measure of similarity between patterns in the n-dimensional space.

Some conceivable properties between the distance metrics can be enumerated; thus,

$$\begin{aligned} &d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) \\ &d(\mathbf{x}, \mathbf{y}) \le d(\mathbf{y}, \mathbf{z}) + d(\mathbf{z}, \mathbf{x}) \\ &d(\mathbf{x}, \mathbf{z}) \ge 0 \\ &d(\mathbf{x}, \mathbf{y}) = 0 \qquad \text{iff } \mathbf{y} = \mathbf{x} \end{aligned}$$

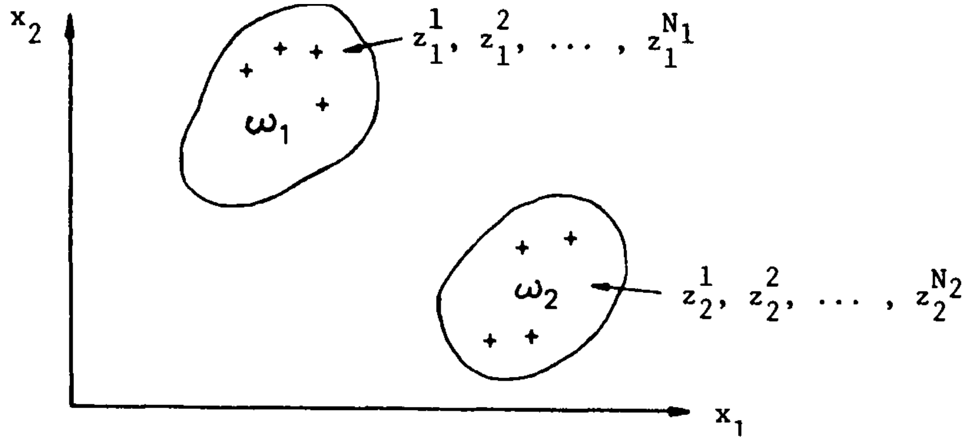

x_2
ω_1
$z_1^1, z_1^2, \ldots, z_1^{N_1}$
ω_2
$z_2^1, z_2^2, \ldots, z_2^{N_2}$
x_1

FIGURE 1.7 Simple two-dimensional pattern space.

where **x**, **y**, and **z** are pattern vectors and $d(.)$ denotes a distance function. Details regarding pattern classification by this approach are presented in subsequent chapters.

1.4.2 Patterns Represented in Linguistically Descriptive Form

We have just discussed representing a pattern by a feature vector. The recognition process of patterns becomes to partition the feature space. This approach is commonly referred to as the decision theoretic approach. This basis of this approach is the meaningful representation of the data set in vector form. There are, on the other hand, patterns whose structural properties are predominant in their descriptions. For such patterns, another approach, called syntactic recognition, will probably be more suitable. The basis of the syntactic approach is to decompose a pattern into subpatterns or primitives. The recognition of a pattern is usually done by parsing the pattern structure according to a set of syntax rules. Figure 1.8a shows a simple pictorial pattern composed of a triangle and a pyramid. Both face F and triangle T are parts of object A. Triangles T_1 and T_2 are parts of object B. The floor and wall together form the background of the scene. Objects A and B together with the background constitute the whole scene, as shown in Figure 1.8a. Figure 1.8b shows its hierarchical representation.

Because of its strong structural regularity, the image of the human chromosome is also a good example of the use of syntactic description. There might be variations in the lengths of arms, but the basic will be the same for certain types of chromosomes, such as submedian or telocentric ones. These variations can easily be recognized visually.

Figure 1.9 shows the structural analysis of a submedian chromosome. Figure 1.9a shows bottom-up parsing on a submedian chromosome. Figure 1.9b

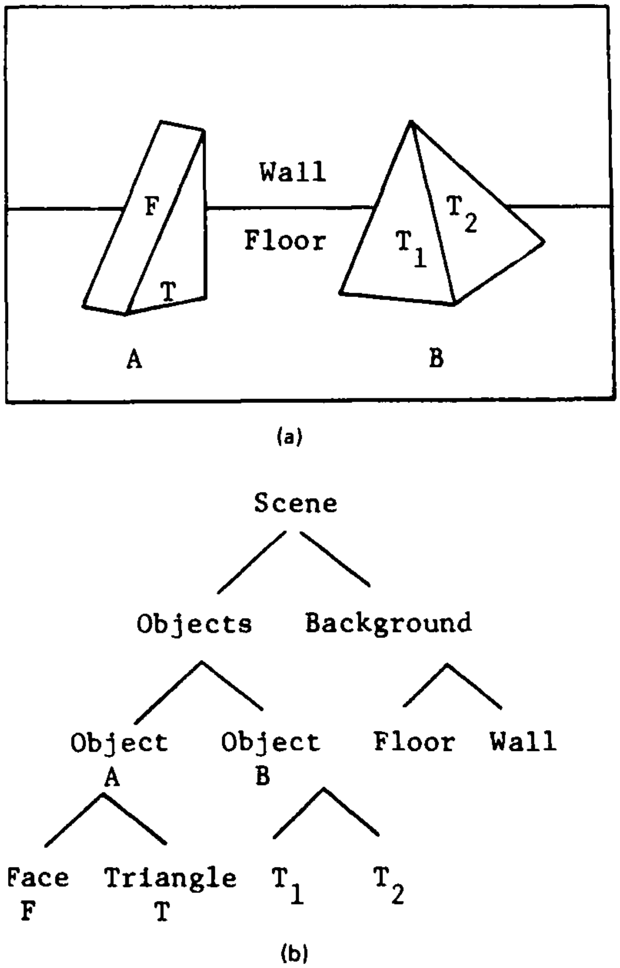
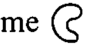


FIGURE 1.8 Hierarchical representation of a simple scene.

shows its structural representation, and Figure 1.9c shows the primitives that we use for shape description.

When the boundary of the chromosome is traced in a clockwise direction, a submedian chromosome can be represented by a string such as *abcbabdbabcbabdb* if the symbols *a*, *b*, *c*, and *d* are suggested for the primitives shown in Figure 1.9c. By the same token, a telocentric chromosome  can be represented with *ebabcbab*. That is, a certain shape will be represented by a certain string of symbols. In the terminology of syntactic pattern recognition, a grammar, or set of rules of syntax, can be established for the generation of sentences for a certain type of chromosome. The sentences generated by two different grammars, say G_1 and G_2, will represent two different shapes; but the sentences generated by the same grammar, say G_1, represent the same category (e.g., submedian chromosomes), with tolerances for minor changes in shape proportion.

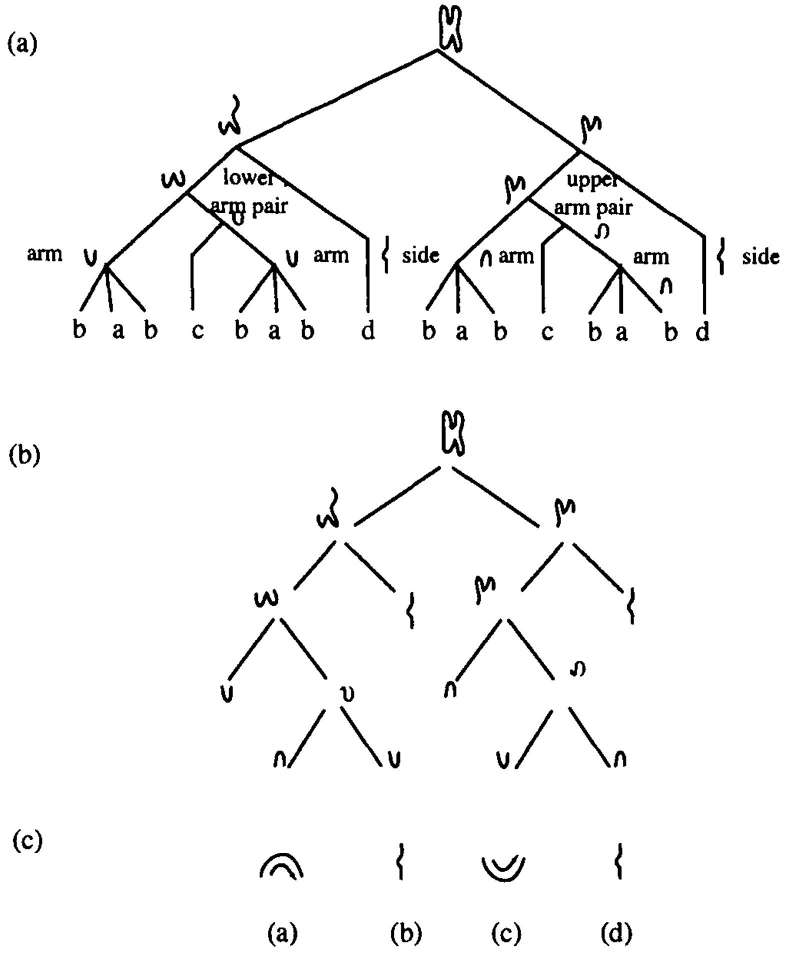


FIGURE 1.9 Structure analysis of a submedian chromosome: (a) bottom-up parsing; (b) structural representation; (c) primitives used for the analysis.

Chinese characters are another good example of the use of syntactic description. They were created and developed according to certain principles, such as pictophonemes and ideographs. They are composed of various primitives and possess strong structural regularities. With these regularities and semantics in mind, thousands of Chinese characters of complex configuration can be segregated and recombined. Thus, the total amount of information will be greatly compressed. Thousands of complex ideographs can then be represented by a few of semantic statements of morphological primitives. It can easily be seen that the total number of fundamental morphological primitives is far much less than 1000, and the complexities of the primitives are also much simpler than the original characters. It is possible, in the meantime, for "heuristics" to play an important role in pattern recognition and grammatical inference on these characters. In addition to structural description of the whole character, the structural approach has been applied to primitive description and extraction for Chinese character recognition.

1.4.3 Approaches to Best Classify Objects for the Above Mentioned Data Categories

Approaches for pattern (or object) classification may be grouped into two categories: (a) the syntactic or structural approach and (b) the decision theoretic approach. For some extreme problems the syntactic or structural approach is most suitable, whereas for some other extreme problems the decision approach is more suitable. The selection of approach depends primarily on the nature of the data set involved in a problem. For those problems where structural information is rich, it might be advantageous to use the syntactic method to show its power for problem description. If the data involved in the problem are better expressed in vector form and at the same time structural information about the patterns is not considered important, the decision theoretic method is recommended for its classification. However, it is not good to be too absolutistic. There are many applications falling half-way between these two extreme cases. In such cases, these two approaches might complement each other. It might be easier or more helpful to use the decision theoretic method to extract some pattern primitives for the syntactic approach, particularly for noisy and distorted patterns. On the other hand, the syntactic method can help to give a structural picture instead of the mathematical results alone obtained through the use of the decision theoretic approach. A comprehensive combination of these two approaches may result in an efficient and practical scheme for pattern recognition. In this book, we will also introduce neural network approach to solve some nonlinear classification problems.

1.5 PARADIGM APPLICATIONS

The pattern recognition technique can be applied to more types of problems than can be enumerated. Readers should not feel restricted to the following applications, which are given for illustration only.

1.5.1 Weather Forecasting

In weather forecasting, the pressure contour map over a certain area (Figure 1.10) constitutes the important data for study. From previous experience and a priori knowledge, several patterns (15 or more, depending on the area) can be specified on the sets of data maps. The weather forecasting problem then becomes to classify the existing pressure contour patterns and to relate them to various weather conditions. Automatic and semiautomatic classifications by computer become necessary when the number of maps builds up.

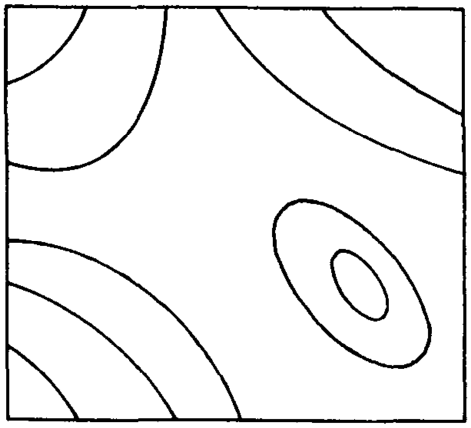


FIGURE 1.10 Example of a pressure contour map over a certain area for weather forecasting studies.

The two methods frequently used for pressure contour map classification are the correlation method and the principal component analysis (Karhunen-Loeve) method. Both of these methods will give global features. Application of the syntactic method for weather forecasting problems, such as the use of string and/or tree representation for pressure contour maps, is also under investigation.

1.5.2 Recognition of Handprinted Characters

Applications of handprinted character recognition are mainly for mail sorting. This problem has been studied for a long time. Due to the wide variations that exist in handwriting (see Figure 1.11 for samples printed by different persons), the correct recognition rate is still not high enough for practical use.

Numerous approaches have been suggested for the recognition of handprinted characters. So far, 121 constrained characters, including 52 uppercase and lowercase alphabetic letters, 10 numerals, and other symbols, are reported to be recognizable.

Machine recognition of more sophisticated characters such as Chinese characters is also under investigation.

1.5.3 Speech Recognition

Speech recognition has numerous applications. One of these is its use to supplement manual handling in mail sorting. When unsorted mail screened from the sorting line is more than manual control operation can handle, speech recognition can be used as a supplementary measure. The essentials of such methods are shown in Figure 1.12.

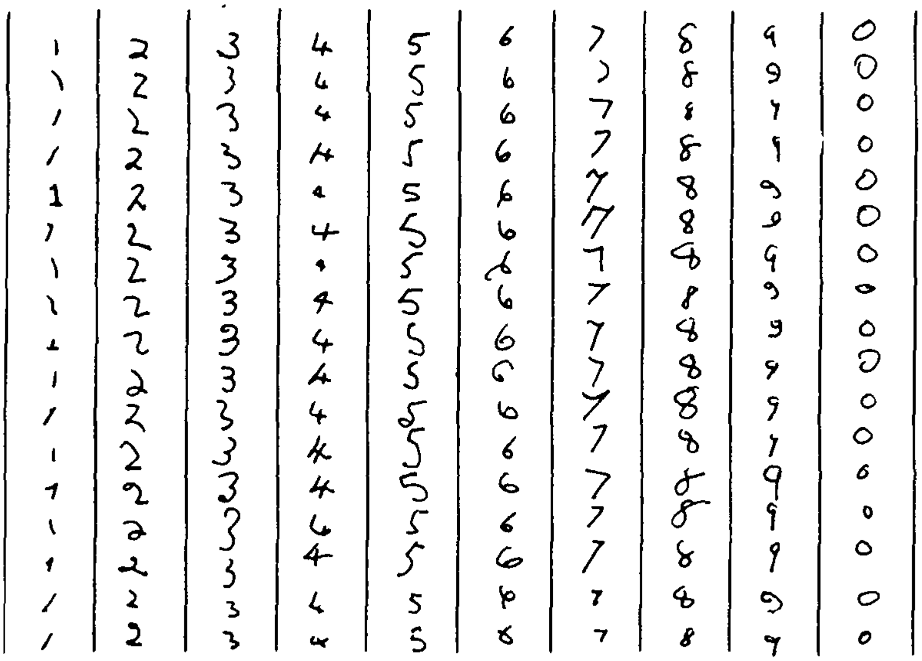


FIGURE 1.11 Samples of handprinted numerals prepared by a variety of people.

Electrical signals converted from spoken words are first filtered and sampled through tuned band-pass filters with center frequencies from 200 to 7500 Hz. Several specific parameters, such as spectral local peaks, speech power, and those representing the gross pattern of spectrum, are extracted for segmentation and phoneme recognition. Errors that have occurred during segmentation and phoneme recognition are corrected by means of preset phoneme correction rules, and then similarity computation is carried out and words of maximum similarity chosen for the solution.

1.5.4 Analysis of ECG to Help Diagnose Heart Activity

Figure 1.13 shows a typical ECG record taken with a cardiograph. Patient's information on his/her heart condition and physician's comments can be easily recorded with the waveforms in a format easily filed for future reference. Figure 1.14 gives the enlarged version of ECG shown in Figure 1.13, as well as the measured ECG parameters. These parameters are very useful for the diagnosis on the patient's heart activity.

1.5.5 Medical Analysis of Chest X-ray

Occupational disease cause workers considerable concern as to job selection. Early cures for such diseases depend on early and accurate diagnosis. An example

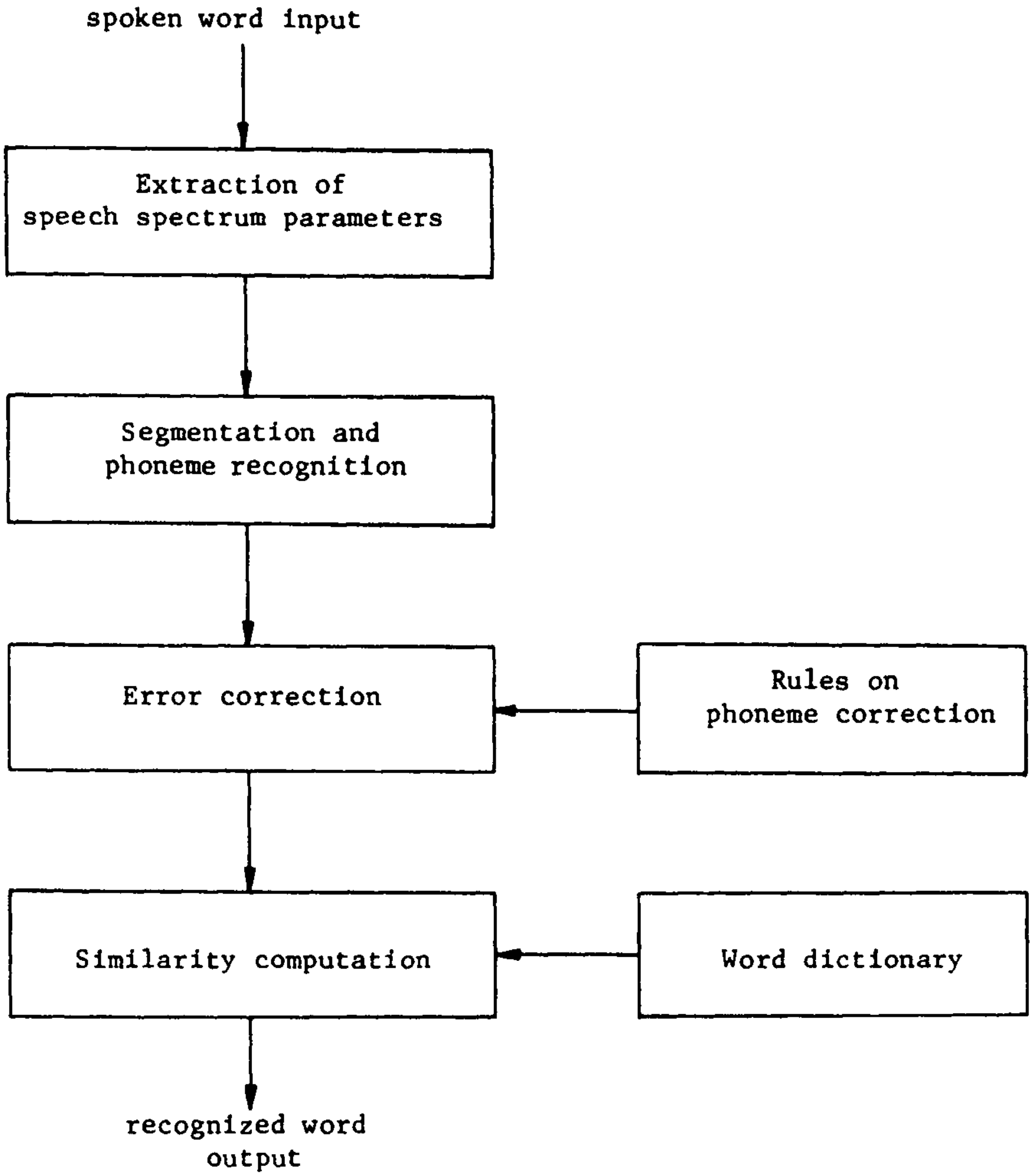

spoken word input
Extraction of speech spectrum parameters
Segmentation and phoneme recognition
Error correction
Rules on phoneme correction
Similarity computation
Word dictionary
recognized word output

FIGURE 1.12 Schematic diagram of a speech recognition system.

is coal miners' pneumoconiosis, a disease of the lungs caused by continual inhalation of irritant mineral or metallic particles. The principal symptom is the descent of the pulmonary arteries. (See Figure 1.15 for an abnormal chest x-ray of a patient.) Accurate diagnosis depends on accurate discrimination of the small opacities of different types from the normal pulmonary vascularity pattern. These opacities appear here and there, sometimes in the interrib space and sometimes in the rib spaces. Those appearing in the rib spaces, overlapped by shadows cast by the major pulmonary arteries, are very hard to recognize. Pattern recognition technique can usefully be applied to this kind of problem.

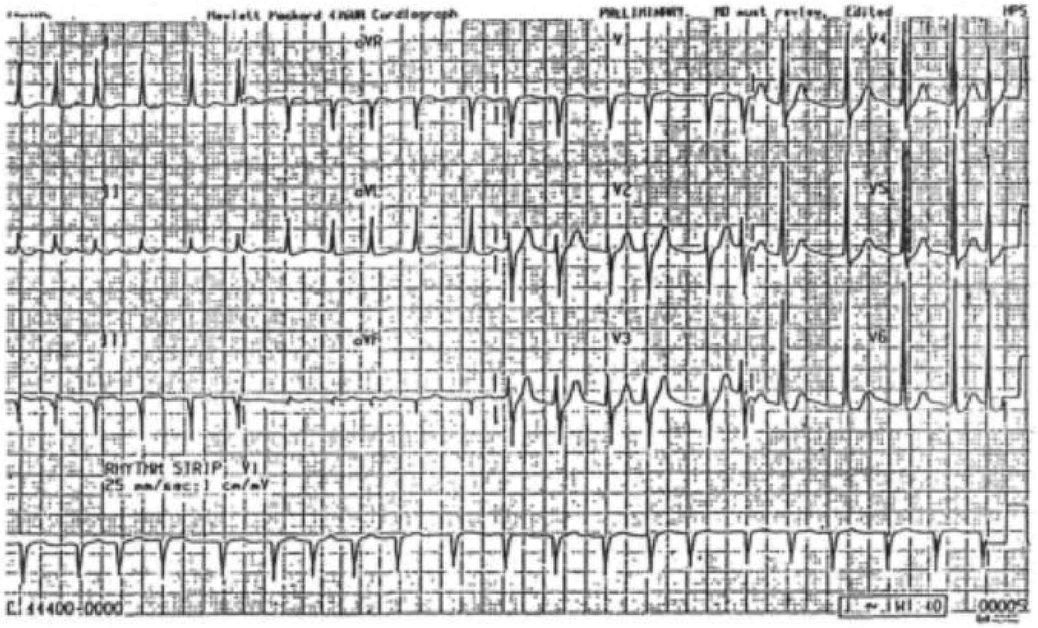


FIGURE 1.13 A typical electrocardiogram on the heart activity.

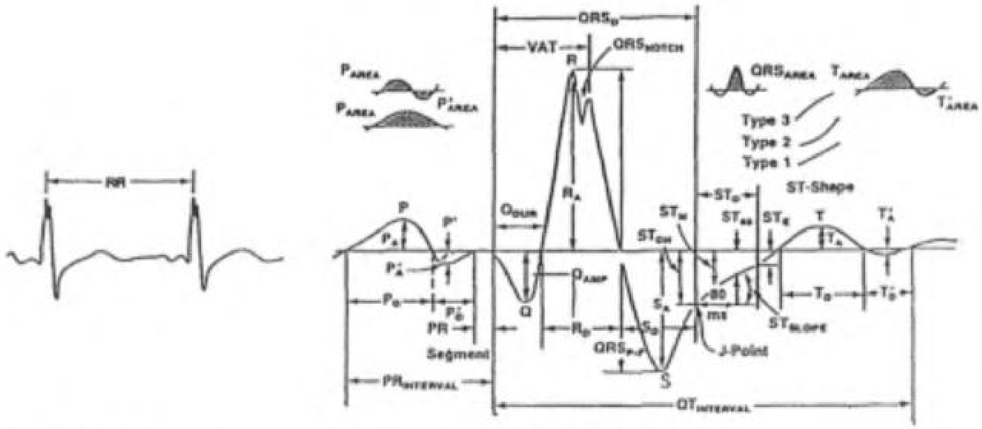


FIGURE 1.14 Measured ECG parameters for ECG shown in Figure 1.13.

To perform this task, the chest x-ray has to be processed to eliminate the major pulmonary arteries, the rib contours, and so on, to provide a frame of reference for the suspicious objects detected. The differences in various texture features are used to classify coal miners' chest x-rays into normal and abnormal classes. Four major categories have been established to indicate the severity of the disease according to the profusion of opacities in the lung region.

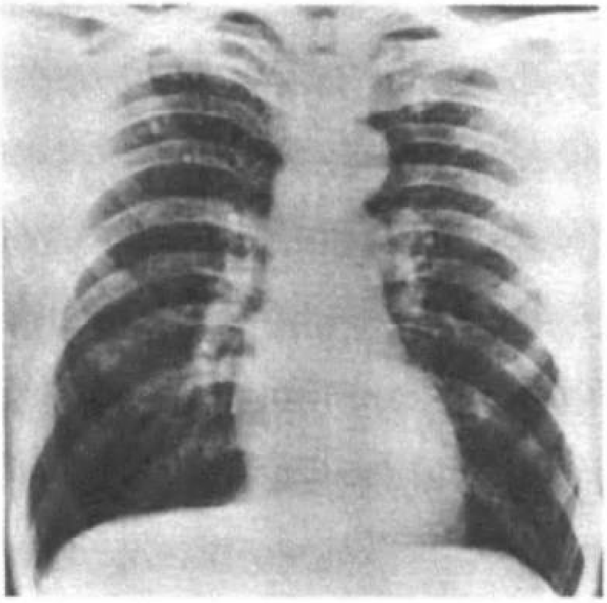


FIGURE 1.15 Chest x-ray of a pneumoconiosis patient. (Courtesy of C.C. Li, Department of Electrical Engineering, University of Pittsburgh.)

1.5.6 Satellite and Aerial-Photo Interpretation

Satellite and/or aerial images are used for both military and civil purposes. Among the civil applications, the remote sensing of earth resources either on or under the surface of the Earth is an important topic for study, especially during the era when we are interested in the global economy. Remote sensing has a wide variety of applications in agriculture, forestry, city planning, geology, geography, and railway line exploitation. The data received from the satellite or from the tape recorded during airplane flight is first restored and enhanced in image form, and then interpreted by a specialist. The principal disadvantage with visual interpretation lies in the extensive training and intensive labor required. In addition, visual interpretation cannot always fully evaluate spectral characteristics. This is because of the limited ability of the eye to discern tonal values on an image and the difficulty an interpreter has in analyzing numerous spectral images simultaneously. In applications where spectral patterns are highly informative, it is therefore preferable to analyze numerical rather than pictorial image data. For these reasons, computer data processing and pattern classification will play an increasingly important role in such applications. Both temporal and spatial patterns are studied to meet different problem requirements. Details of these applications will not be presented here, as a more detailed worked-out problem is given later to illustrate some of the principles discussed in Chapter 5.

2

Supervised and Unsupervised Learning in Pattern Recognition

To classify a pattern into a category is itself a *learning process*. It is expected that the pattern classification (or pattern recognition) system should have the ability to learn and to improve its performance of the classification through learning. The improvement in performance takes place over time in accord with some prescribed measure. A pattern recognition system learns through iterative adjustment of the *synaptic weights* and/or other system parameters. It is hoped that after an iteration of the learning process the system will become a more knowledgeable and effective system, and will produce a higher recognition rate.

To this end the pattern recognition system will first undergo a training process. During the training process, the system is repeatedly presented with a set of *prototypes*, that is, with a set of input patterns along with the category to which each particular pattern belongs. Whenever an error occurs in the system output, an adjustment on the system parameters (i.e., the synaptic weights) will follow. After all the prototypes have been correctly classified, then let the system go free by itself to classify any new pattern that has not been seen before but which, we know, belongs to the same population of patterns used to train the system. If the system is well trained (i.e., when the number of prototypes are properly chosen and all the prototypes are correctly classified), the system should be able to correctly classify this new pattern and many other patterns like this. Pattern classification as described above is called a supervised learning. The advantage of using this *supervised learning* system to perform the pattern classification is that it can construct a linear or a nonlinear decision boundary between different classes in a nonparametric fashion, and therefore offer a practical method for solving highly complex pattern classification problems.

It should be noted that there are many other cases where there exists no a priori knowledge of the categories into which the patterns are to be classified. In such a situation, *unsupervised learning* will play an important role in the pattern classification. In unsupervised learning (also called *clustering*), patterns are associated by themselves into clusters based on some properties in common. These properties are sometimes known as *features*.

Figure 2.1 indicates the main difference between supervised and unsupervised learning processes in pattern recognition. In the supervised pattern recognition there is a *teacher* which provides a desired output vector $\mathbf{d}$ for every prototype vector $\mathbf{z}$ used for training, or a set of ($\mathbf{z}$, $\mathbf{d}$) pairs for the training of the system as $(\mathbf{z}_1, \mathbf{d}_1)$; $(\mathbf{z}_2, \mathbf{d}_2)$; $(\mathbf{z}_3, \mathbf{d}_3)$; ... ; $(\mathbf{z}_N, \mathbf{d}_N)$.

When a prototype $\mathbf{z}$ is presented to an untrained system (or not yet completely trained system), an error will occur. System parameters will then be adjusted as discussed to change the output response $f(\mathbf{z}, \mathbf{w})$ to best approximate the desired response vector $\mathbf{d}$, which is supplied by the teacher in an optimum fashion.

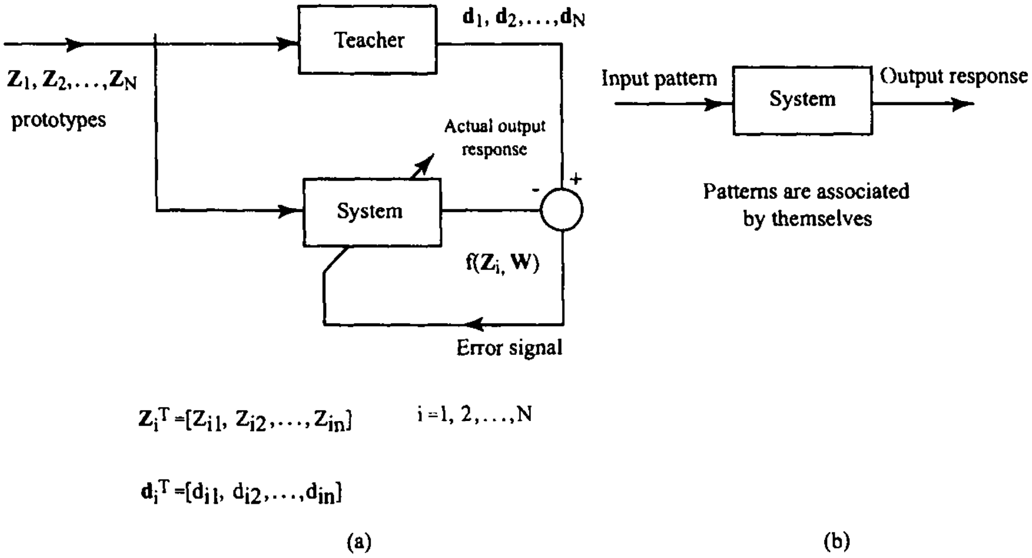


FIGURE 2.1 Block diagrams of supervised and unsupervised learning processes: (a) supervised learning; (b) unsupervised learning.

The iterative adjustment of the system parameters is based on a certain number of properly selected prototypes in the form of a listed pairs, namely, $\{(\mathbf{z}_1, \mathbf{d}_1), (\mathbf{z}_2, \mathbf{d}_2), (\mathbf{z}_3, \mathbf{d}_3), \ldots, (\mathbf{z}_N, \mathbf{d}_N)\}$. The proper number of prototypes needed for the training of a particular system will be discussed in Chapter 4.

When a prototype vector $\mathbf{z}_i$ from the set of $(\mathbf{z}, \mathbf{d})$ pairs is input to the system, an error occurs if the actual output response is a value other than $\mathbf{d}_i$. Let us define the discrepancy between the desired response vector $\mathbf{d}$ ($\mathbf{d}_i$ in this case) and the actual output response $f(\mathbf{z}, \mathbf{w})$ produced by the learning system (say $\mathbf{d}_k$) as the loss function L_{ik}. The conditional average risk function $r(\mathbf{z})$ can then be defined as

$$r_k(\mathbf{z}) = \sum_{i=1}^{N} L_{ik} p(\mathbf{d}_i|\mathbf{z}) \qquad k = 1, 2, \ldots, N \text{ and } i \neq k$$

where $p(\mathbf{d}_i|\mathbf{z})$ is the probability that $\mathbf{z}$ is the same as $\mathbf{d}_i$ from the $(\mathbf{z}, \mathbf{d})$ pairs. This is actually the a posteriori probability. It is the conditional probability distribution of $\mathbf{d}_i$ given $\mathbf{z}$. The goal of the learning process is to minimize the risk function $r(\mathbf{z})$ over the training process. See Chapter 5 for detailed discussions on the formulation of the learning problem with statistical design theory.

In the supervised learning, the system parameters are adjusted whenever there is a discrepancy between the desired response $\mathbf{d}$ and the actual output response. The error signal can be evaluated as the difference between the actual output response of the system $f(\mathbf{z}, \mathbf{w})$ and the desired response $\mathbf{d}$. The adjustment is carried out in an iterative step-by-step fashion to help the system emulate the teacher. When this condition is reached, the system will be left free completely by itself to perform classification on those patterns that are unknown of their belongings, but are known to belong to the same population of patterns used for the system training.

The form of supervised learning that we have just described is an error-correcting learning. Any given operation of the system under the teacher's supervision is represented as a point on the error surface. When the system improves its performance over time via learning from the teacher, the operating point should move down successively toward a minimum point of the error surface. Our job now becomes to design an algorithm to minimize the cost function of interest with an adequate set of input-output pairs for mapping. Chapter 4 of this book will show how to move the operating point toward a minimum point of the error surface.

Unsupervised learning appears under different names in different contexts. *Clustering* is the name most frequently used. We can find names like *numerical taxonomy* in biology and ecology, *typology* in social sciences, and *partition* on graph theory. In unsupervised or self-organized learning there is no class labeling available, nor do we know how many classes there are for the input patterns.

There are no specific examples of the function to be learned by the system. These input patterns mainly associate themselves naturally based on some properties in common. Our major concern now is to discover similarities and dissimilarities among patterns and to "reveal" the organization of patterns into "sensible" clusters (groups). It is expected that patterns belonging to the same cluster should be very close together in the pattern space, while patterns in different clusters should be farther apart from one another. See Chapter 6 (Clustering Analysis and Unsupervised Learning) for detailed discussions on the intraset and interset distances.

In this section we have briefly described what we call supervised and unsupervised pattern recognition. In those cases where there is available a set of training data set (i.e., a set of appropriate input-output pairs), the supervised pattern recognition approach can be adopted. The classifier can be designed with this known information. However, for many situations no such a priori information is available. All we have is only a set of feature vectors. Our job now is to set up some measures and then design an algorithm to search for similarities and dissimilarities among these pattern vectors and then group patterns that possess similar features together to form clusters. That is to say, pattern vectors group themselves by natural association. For such kinds of problems, unsupervised learning (or clustering) will play an important role. A major issue in the unsupervised pattern recognition is to define the "similarity" between two feature vectors and choose an appropriate measure for it. Another issue of importance is to choose an algorithmic scheme that will cluster the vectors on the basis of the adopted similarity measure. Chapter 6 is dedicated to review the currently available algorithms and make some suggestions for various kinds of pattern data sets. Examples include data sets with different density distributions, data sets with a neck or valley between them, data sets in a chain form, etc.

In this book we will first discuss the supervised learning and the various algorithms currently used, and then come to the unsupervised learning (clustering). Use of neural network for pattern recognition will also be discussed.

3

Nonparametric DecisionTheoretic Classification

Supervised pattern recognition (supervised learning) algorithms are often categorized into two approaches: *parametric* and *nonparametric* approaches. For some classification tasks, pattern categories are known a priori to be characterized by a set of parameters. The approach designed for such kinds of tasks is the *parametric approach*. It defines the discriminant function by a class of probability densities defined by a relatively small number of parameters.

There exist many other classifications in which no assumptions can be made about the characterizing parameters. Approaches designed for those tasks are called *nonparametric*. Although some parameterized discriminant functions (e.g., the coefficients of a multivariate polynomial of some degree) are used in nonparametric methods, no conventional form of the distribution is assumed. This makes it (the nonparametric approach) different from the parametric approach. In the parametric approach, pattern classes are usually assumed to arise from a multivariate Gaussian distribution where the parameters are the mean and covariance.

In this chapter emphasis is on the discussion of the nonparametric decision theoretic classification. Based on the nature of the problem, we are going to discuss several related topics in succession. To start, some technical definitions of *decision surfaces* and *discriminant* functions are introduced, and then the discussion is directed to the general form of the discriminant function, its properties, and classifier based on this sort of discriminant function. Cases

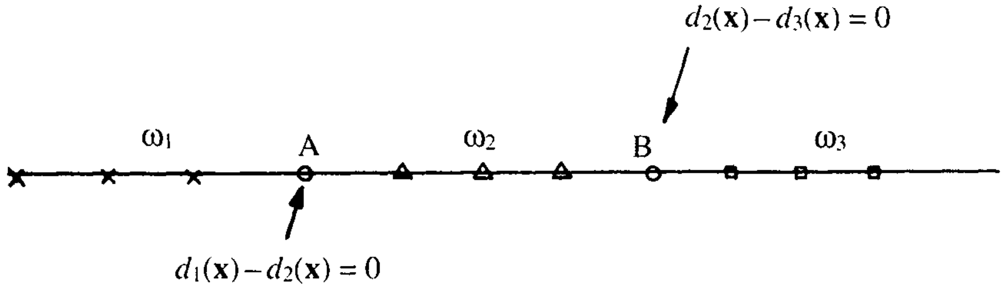

$d_2(\mathbf{x}) - d_3(\mathbf{x}) = 0$
ω1
A
ω2
B
ω3
$d_1(\mathbf{x}) - d_2(\mathbf{x}) = 0$

FIGURE 3.1 One-dimensional pattern space.

dealing with linear discriminant functions, piecewise linear discriminant functions, and nonlinear discriminant functions are discussed separately. A discussion of ϕ machines and their capacity to classify patterns follows to generalize the nonparametric decision theoretic classification method. At the end of this chapter, potential functions used as discriminant functions are included to give a more complete description of the nonparametric decision theoretic classification.

3.1 DECISION SURFACES AND DISCRIMINANT FUNCTIONS

As mentioned in Chapter 1, each pattern appears as a point in the pattern space. Patterns pertaining to different classes will fall into different regions in the pattern space. That is to say, different classes of patterns will cluster in different regions, and can be separated by separating surfaces. Separating surfaces, called *decision surfaces*, can formally be defined as surfaces, which could be found from prototypes (or training samples) to separate these known patterns in the n-dimensional space, and are used to classify unknown patterns. Such decision surfaces are called *hyperplanes*, and are $(n-1)$-dimensional. When $n = 1$, the decision surface is a point. As shown in Figure 3.1, point A is the point that separates classes ω_1 and ω_2, and point B is the separating point between ω_2 and ω_3. When $n = 2$, the decision surface becomes a line.

$$w_1 x_1 + w_2 x_2 + w_3 = 0 \tag{3.1}$$

as shown in Figure 3.2. When $n = 3$, the surface is a plane. When $n = 4$ or higher, the decision surface is a hyperplane represented by

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n + w_{n+1} = 0 \tag{3.2}$$

expressed in matrix form as

$$\mathbf{w} \cdot \mathbf{x} = 0 \tag{3.3}$$

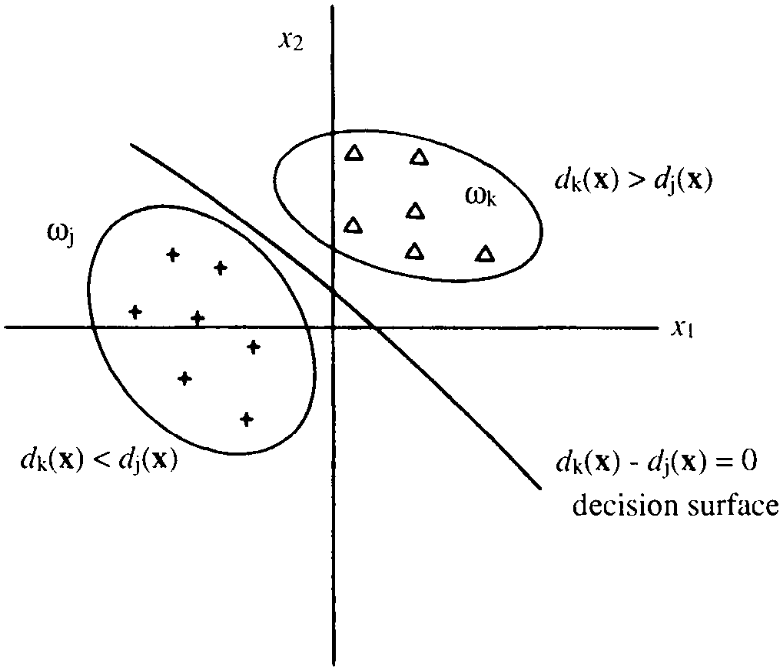

x_2
$d_k(\mathbf{x}) > d_j(\mathbf{x})$
ω_k
ω_j
x_1
$d_k(\mathbf{x}) < d_j(\mathbf{x})$
$d_k(\mathbf{x}) - d_j(\mathbf{x}) = 0$
decision surface

FIGURE 3.2 Two-dimensional pattern space.

where

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w_{n+1} \end{bmatrix} \quad \text{and} \quad \mathbf{x} = \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{vmatrix}$$

w and **x** are, respectively, called the augmented weight vector and the augmented pattern vector. The scalar term w_{n+1} has been added to the weight function for coordinate translation purposes. To make the equation a valid vector multiplication, the input vector **x** has been augmented to become $(n + 1)$-dimensional by adding $x_{n+1} = 1$. This will allow a translation of all linear discriminant functions to pass through the origin of the augmented space when desired.

A discriminant function is a function $d(\mathbf{x})$ which defines the decision surface. As shown in Figure 3.2, $d_k(\mathbf{x})$ and $d_j(\mathbf{x})$ are values of the discriminant function for patterns **x**, respectively, in classes k and j. $d(\mathbf{x}) = d_k(\mathbf{x}) - d_j(\mathbf{x}) = 0$

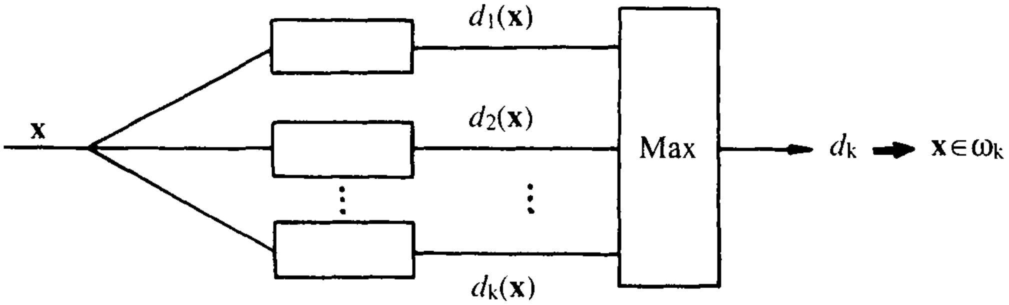


FIGURE 3.3 Schematic diagram of a simple classification system.

will then be the equation defining the surface that separate classes k and j. We can then say that

$$d_k(\mathbf{x}) > d_j(\mathbf{x}) \qquad \begin{array}{l} \forall \mathbf{x} \in \omega_k \\ \forall j \neq k, j = 1, 2, \ldots, M \end{array}$$

A system can then be built to classify pattern $\mathbf{x}$, as shown in Figure 3.3. For a two-class problem

$$d_1(\mathbf{x}) = d_2(\mathbf{x}) \tag{3.5}$$

or

$$d(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x}) = 0 \tag{3.6}$$

will define the separating hyperplane between the two classes.

In general, if we have M different classes, there will be $M(M-1)/2$ separating surfaces. But some of the separating surfaces are redundant: only $M - 1$ are needed to separate the M classes. Figure 3.4 shows the number of separating surfaces as a function of the number of categories to be classified in a two-dimensional pattern space. From Figure 3.4a we can see that the decision surface separating the two categories is a line. On the line, $d(\mathbf{x}) = 0$; below this separating line, $d(\mathbf{x}) > 0$; and above this line, $d(\mathbf{x}) < 0$. Thus the line $d(\mathbf{x}) = 0$ separates two different classes. Similarly, Figure 3.4b is self-explanatory. Note that in the cross-hatched region, where $d_1(\mathbf{x}) < 0$ and $d_2(\mathbf{x}) < 0$, patterns belong neither to ω_1 nor to ω_2. This region may then be classified as ω_3. The same thing happens in Figure 3.4c: Patterns falling in the cross-hatched portion of the plane do not belong to category 1, 2, or 3, and thus form a new category, say ω_4. Portions not mentioned in this pattern space are indeterminate regions.

Example. For a two-class problem ($M = 2$), find a discriminant function to classify the two patterns $\mathbf{x}_1$ and $\mathbf{x}_2$ into two categories.

$$\mathbf{x}_1 = \begin{vmatrix} 1 \\ 4 \end{vmatrix} \quad \text{and} \quad \mathbf{x}_2 = \begin{vmatrix} 4 \\ 2 \end{vmatrix}$$

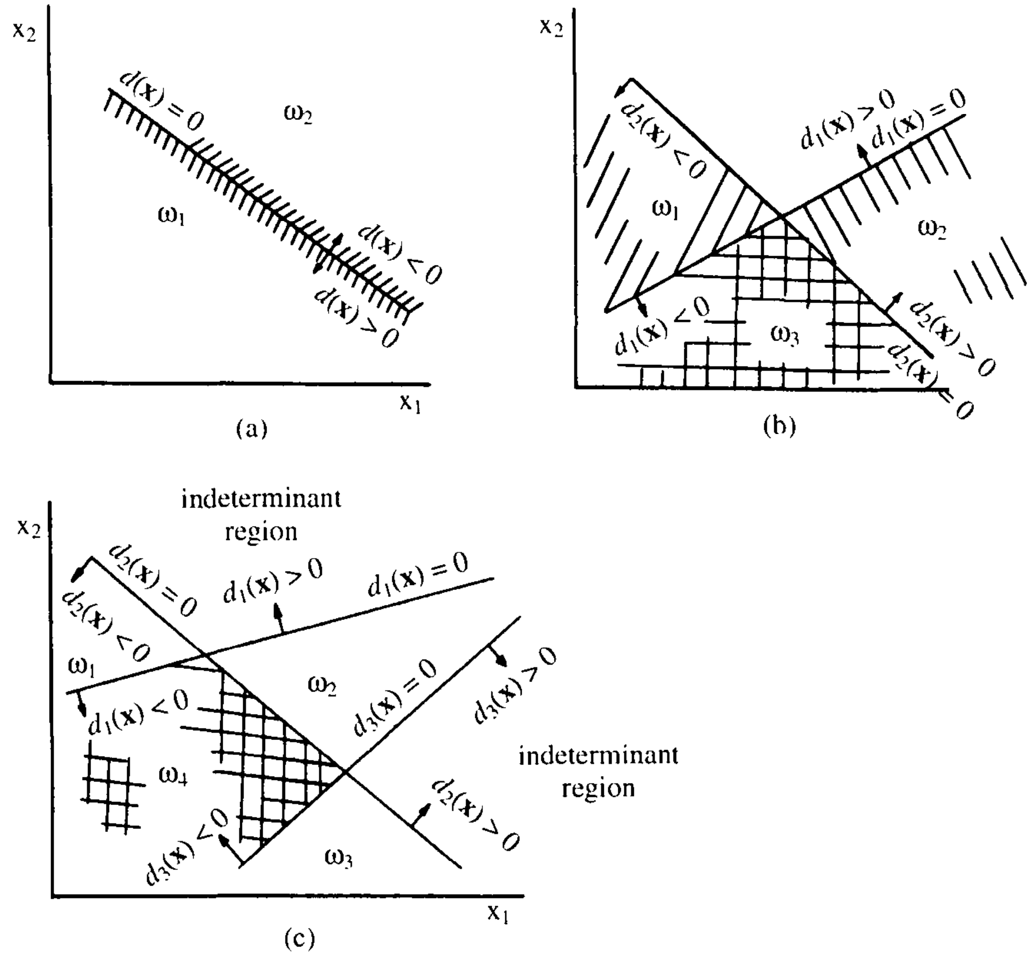


FIGURE 3.4 Separating surfaces in a two-dimensional pattern space.

Let us try $d(\mathbf{x}) = x_1 - \frac{1}{2}x_2 - 2$, and see whether it can be used as the separating line for these two patterns. Substituting the augmented vectors of $\mathbf{x}_1$ and $\mathbf{x}_2$ into $d(\mathbf{x})$, we find that

$$d(\mathbf{x}_1) = \mathbf{w} \cdot \mathbf{x}_1 = (1 \quad -\tfrac{1}{2} \quad -2) \left|\begin{matrix} 1 \\ 4 \\ 1 \end{matrix}\right| = -3 < 0$$

$$d(\mathbf{x}_2) = \mathbf{w} \cdot \mathbf{x}_2 = (1 \quad -\tfrac{1}{2} \quad -2) \left|\begin{matrix} 4 \\ 2 \\ 1 \end{matrix}\right| = +1 > 0$$

Then, the pattern $\mathbf{x}_1$ can be classified in one category, and $\mathbf{x}_2$ in another category according to this discriminant function.

3.2 LINEAR DISCRIMINANT FUNCTIONS

As mentioned in Section 3.1, patterns falling in different regions in the pattern space can be grouped into different categories by means of separating surfaces which are defined by discriminant functions. The discriminant function may be linear or nonlinear according to the nature of the problem.

In this section we start by discussing the general form of the linear discriminant function, and then apply it to the design of the minimum distance classifier. Linear separability will also be discussed.

3.2.1 General Form

The linear discriminant function will be of the following form:

$$d_k(\mathbf{x}) = w_{k1}x_1 + w_{k2}x_2 + \cdots + w_{kn}x_n + w_{k,n+1}x_{n+1} \tag{3.7}$$

Put in matrix form,

$$d_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} \tag{3.8}$$

where

$$\mathbf{w}_k = \begin{vmatrix} w_{k1} \\ w_{k2} \\ \vdots \\ w_{kn} \\ w_{k,n+1} \end{vmatrix} \qquad \mathbf{x} = \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{vmatrix}$$

and $x_{n+1} = 1$ in the augmented $\mathbf{x}$ pattern vector. For a two-class problem where $M = 2$, the decision surface is

$$d(\mathbf{x}) = \mathbf{w}_1^T\mathbf{x}_1 - \mathbf{w}_2^T\mathbf{x}_2 = (\mathbf{w}_1 - \mathbf{w}_2)^T\mathbf{x} = 0 \tag{3.9}$$

It is a hyperplane passing through the origin in an augmented feature space for the reason discussed previously.

3.2.2 Minimum Distance Classifier

Although it is one of the earliest methods suggested, the minimum distance classifier is still an effective tool in solving the pattern classification problem. The decision rule used in this method is

$$\mathbf{x} \in \omega_j \qquad \text{if } D(\mathbf{x}, \mathbf{z}_j) = \min_k D(\mathbf{x}, \mathbf{z}_k);\ k = 1, 2, \ldots, M \tag{3.10}$$

where $D(.)$ is a metric called the Euclidean distance of an unknown pattern $\mathbf{x}$ from $\mathbf{z}_k$, and $\mathbf{z}_k$ is the prototype average (or class center) for class ω_k. Then

$$D(\mathbf{x}, \mathbf{z}_k) = |\mathbf{x} - \mathbf{z}_k| \tag{3.11}$$

Remembering that if

$$D(\mathbf{x}, \mathbf{z}_k) > D(\mathbf{x}, \mathbf{z}_j) \qquad \forall j, k \tag{3.12}$$

then

$$D^2(\mathbf{x}, \mathbf{z}_k) > D^2(\mathbf{x}, \mathbf{z}_j) \tag{3.13}$$

is true for most cases. In other words, we can use D^2 to replace D in the decision rule above. Then we have

$$D^2(\mathbf{x}, \mathbf{z}_k) = |\mathbf{x} - \mathbf{z}_k|^2 \tag{3.14}$$

Put in matrix form, we have

$$\begin{aligned} D^2(\mathbf{x}, \mathbf{z}_k) &= (\mathbf{x} - \mathbf{z}_k)^T(\mathbf{x} - \mathbf{z}_k) \\ &= \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{z}_k + \mathbf{z}_k^T\mathbf{z}_k \end{aligned} \tag{3.15}$$

after expanding. On the right-hand side of the expression above, $\mathbf{x}^T\mathbf{x}$ is constant for all k, and therefore can be eliminated. Thus, to seek the minimum of $D(\mathbf{x}, \mathbf{z}_k)$ is equivalent to seeking

$$\min_k[-2\mathbf{x}^T\mathbf{z}_k + \mathbf{z}_k^T\mathbf{z}_k] \tag{3.16}$$

or alternatively, to seeking

$$\max_k[\mathbf{x}^T\mathbf{z}_k - \tfrac{1}{2}\mathbf{z}_k^T\mathbf{z}_k] \qquad k = 1, 2, \ldots, M \tag{3.17}$$

which is the decision rule for the minimum distance classifier. The discriminant function used in the classifier can then be expressed as

$$d_k(\mathbf{x}) = \mathbf{x}^T\mathbf{z}_k - \tfrac{1}{2}\mathbf{z}_k^T\mathbf{z}_k = \mathbf{x}^T\mathbf{z}_k - \tfrac{1}{2}|\mathbf{z}_k|^2 = \mathbf{x}^T\mathbf{w} \tag{3.18}$$

where

$$\mathbf{w} = \left|\begin{array}{c} z_k^1 \\ z_k^2 \\ z_k^3 \\ \vdots \\ z_k^n \\ -\frac{1}{2}|z_k|^2 \end{array}\right|$$

and $\mathbf{x}$ is an augmented pattern vector. Note that the decision surface between any two classes ω_i and ω_j is formed by the perpendicular bisectors of $\mathbf{z}_i - \mathbf{z}_j$ shown

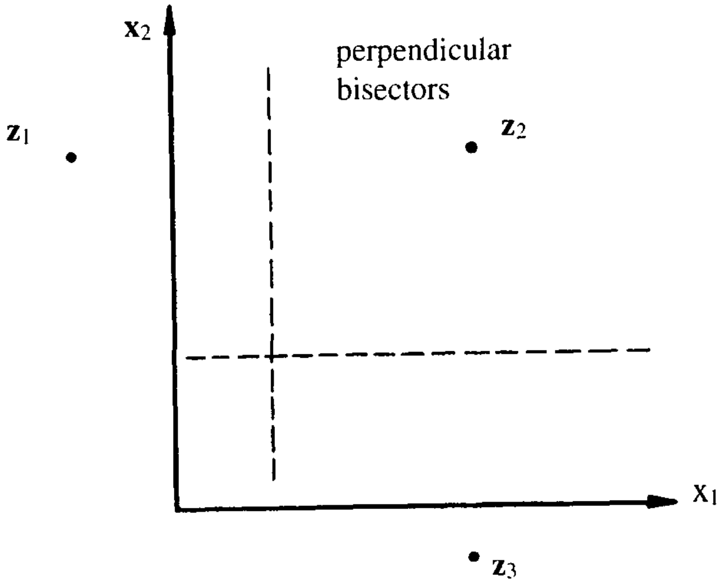


FIGURE 3.5 Geometrical properties of the decision surfaces.

by dashed lines in Figure 3.5. The proof for this is not difficult. From Eq. (3.18), the decision surface between $\mathbf{z}_1$ and $\mathbf{z}_2$ is

$$d(\mathbf{x}) = \mathbf{x}^T(\mathbf{z}_1 - \mathbf{z}_2) - \tfrac{1}{2}(\mathbf{z}_1^T\mathbf{z}_1 - \mathbf{z}_2^T\mathbf{z}_2) = 0 \tag{3.19}$$

Obviously, the midpoint between $\mathbf{z}_1$ and $\mathbf{z}_2$ is on the decision surface. This can be shown by direct substitution of this midpoint into Eq. (3.19), which shows that the equation is satisfied. Similarly, we can find that all other points on the boundary surface (a line in this case) also satisfy Eq. (3.19). It can also be proved that the vector $(\mathbf{z}_1 - \mathbf{z}_2)$ is in the same direction as the unit normal to the hyperplane, which is

$$\frac{\mathbf{z}_1 - \mathbf{z}_2}{|\mathbf{z}_1 - \mathbf{z}_2|} \tag{3.20}$$

Note also that the minimum distance classifier (MDC) uses a single point to represent each class. This representation would be all right if the class were normally distributed with equal variances in all direction, as shown in Figure 3.6a. But if the class is not normally distributed with equal variances in all directions, as shown in Figure 3.6b, misclassification will occur. Even if $D_1 < D_2$, point $\times$ should be classified to ω_2 instead of ω_1. Similarly, in Figure 3.6c, point $\times$ might be classified in class ω_3 by the MDC, but it is really closer to ω_2. That is, single points do not represent classes ω_1, ω_2, and ω_3 very well. This can be remedied by representing a class with multiple prototypes. When each

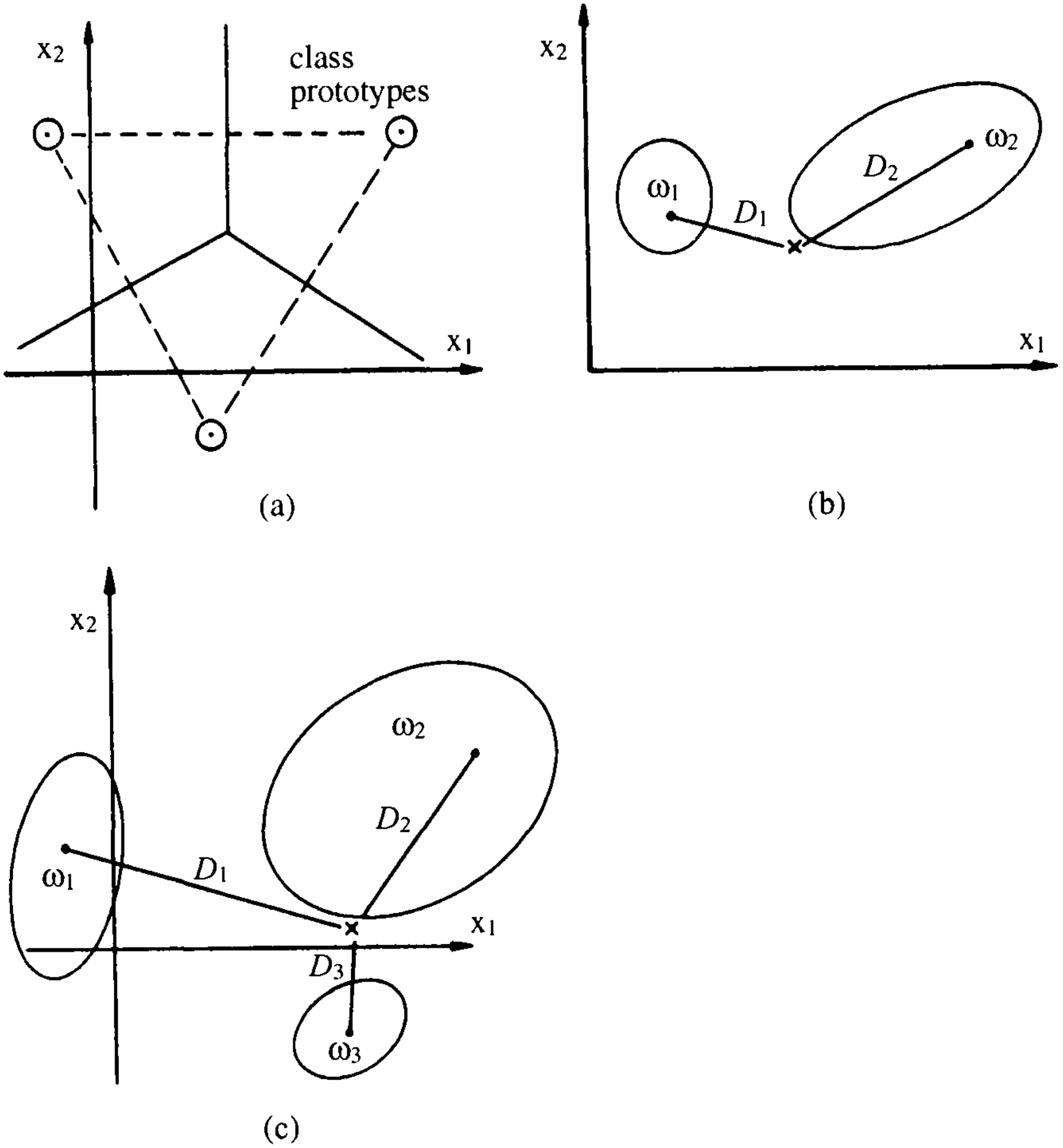


FIGURE 3.6 Possible misclassification by the MDC with a single prototype class representation.

pattern category is represented by multiple prototypes instead of a single prototype, then

$$D(\mathbf{x}, \omega_k) = \min_{m=1,\ldots,N_k} [D(\mathbf{x}, \mathbf{z}_k^m)] \tag{3.21}$$

where k represents the kth category, m represents the mth prototype, and N_k represents the number of prototypes used to represent category k. Equation (3.21) gives the smallest of the distances between $\mathbf{x}$ and each of the prototypes of ω_k. The decision rule then becomes

$$\mathbf{x} \in \omega_j \qquad \text{if } D(\mathbf{x}, \omega_j) = \min_{k=1,\ldots,M} (\mathbf{x}, \omega_k) \tag{3.22}$$

where $D(\mathbf{x}, \omega_k)$ is given by Eq. (3.21). The discriminant function changes correspondingly to the following form:

$$d_k(\mathbf{x}) = \max_{m=1,\ldots,N_k} [\mathbf{x}^T\mathbf{z}_k^m - \tfrac{1}{2}|\mathbf{z}_k^m|^2] \qquad k = 1, 2, \ldots, M \tag{3.23}$$

3.2.3 Linear Separability

Some properties relating to the classification, such as linear separability of patterns, are discussed next. Pattern classes are said to be linearly separable if they are classifiable by any linear function, as shown in Figure 3.7a, whereas the classes in Figure 3.7b and c are not classifiable by any linear function. Such types of problems will be discussed in later chapters.

From Figure 3.7 we can see that the decision surfaces in linearly separable problems are convex. By definition, a function is said to be convex in a given region if a straight line drawn within that region lies entirely in or above the function. The regional function shown in Figure 3.8a is said to be convex, since straight lines *ab* and *ac* are all above the function curve, whereas that shown in Figure 3.8b is not.

3.3 PIECEWISE LINEAR DISCRIMINANT FUNCTIONS

So far, our discussion has focused on linearly separable problems. Linearly separable problems seem relatively simple, but in our real world most problems are linearly nonseparable, and therefore more effective approaches must be sought. One way to treat these linearly nonseparable problems is to use piecewise linear discriminant function. This is the topic of the next several sections.

3.3.1 Definition and Nearest-Neighbor Rule

A piecewise linear function is a function that is linear over subregions of the feature space. These piecewise linear discriminant functions give piecewise linear boundaries between categories as shown in Figure 3.9a. In Figure 3.9b the boundary surface between class ω_1 and class ω_2 is nonconvex. However, it can be

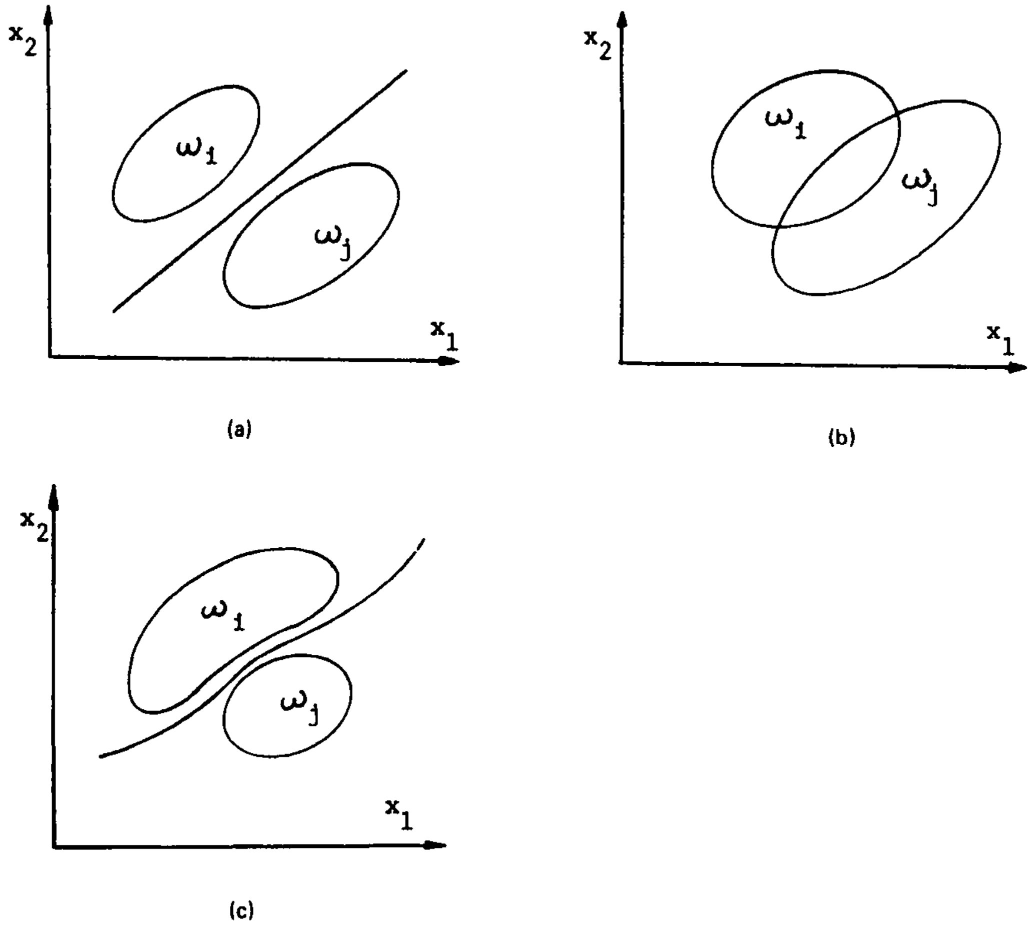


FIGURE 3.7 Linear separability between pattern classes: (a) classes ω_i and ω_j are linearly separable; (b, c) classes ω_i and ω_j are not linearly separable.

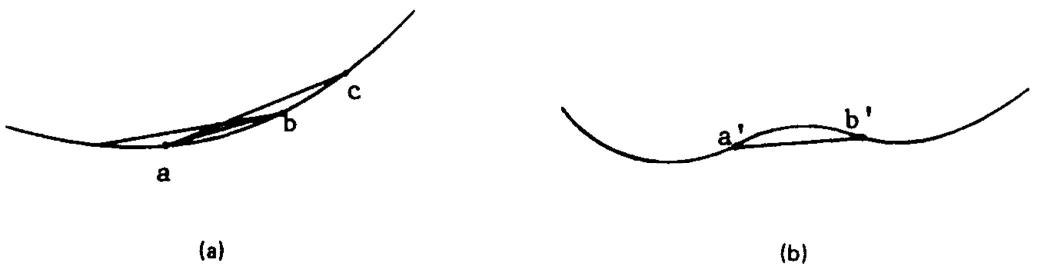


FIGURE 3.8 Convexity property of a function: (a) convex decision surface; (b) not convex.

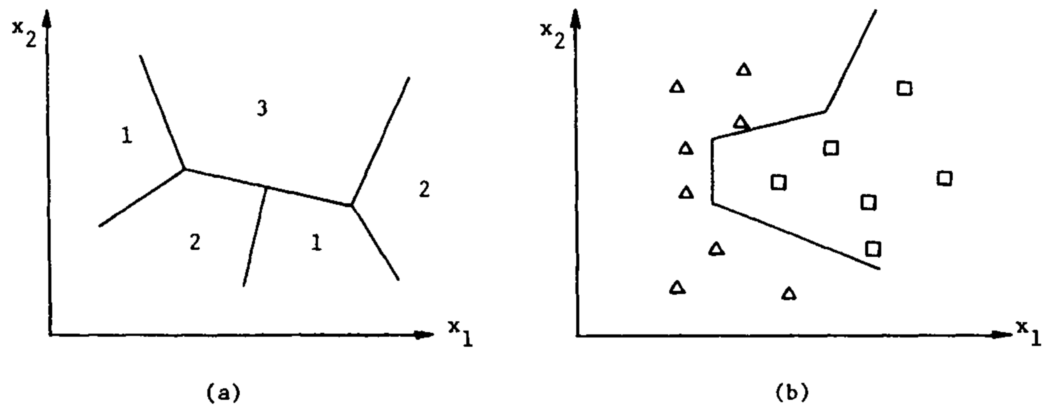


Figure 3.9 Piecewise linear separability among different classes.

broken down into piecewise linear boundaries between these two classes ω_1 and ω_2. The discriminant functions are given by

$$d_k(\mathbf{x}) = \max_{m=1,\ldots,N_k} [d_k^m(\mathbf{x})] \qquad k = 1, \ldots, M \tag{3.24}$$

that is, we find the maximum $d_k^m(\mathbf{x})$ among the prototypes of class k, where N_k is the number of prototypes in class k and

$$\begin{aligned} d_k^m(x) &= w_{k1}^m x_1 + w_{k2}^m x_2 + \cdots + w_{kn}^m x_n + w_{k,n+1}^m \\ &= (\mathbf{w}_k^m)^T \mathbf{x} \end{aligned} \tag{3.25}$$

where

$$\mathbf{w}_k^m = \left| \begin{array}{c} w_{k1}^m \\ w_{k2}^m \\ \vdots \\ w_{kn}^m \\ w_{k,n+1}^m \end{array} \right|$$

and

$$\mathbf{x} = \left| \begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{array} \right|$$

Three different cases of the pattern classification problem can be enumerated:

Case 1. Each pattern class is separable from all other classes by a single decision surface, as shown in Figure 3.10a, where several indeterminate regions can be seen to exist.

Case 2. Each pattern class is pairwise separable from every other class by a distinct decision surface. Indeterminate regions may also exist. In this case, no class is separable from the others by a single decision surface. For example, ω_1 can be separated from ω_2 by the surface $d_{12}(\mathbf{x}) = 0$ and from ω_3 by $d_{13}(\mathbf{x}) = 0$ (see Figure 3.10b). There will be $M(M-1)/2$ decision surfaces which are represented by

$$d_{ij}(\mathbf{x}) = 0 \tag{3.26}$$

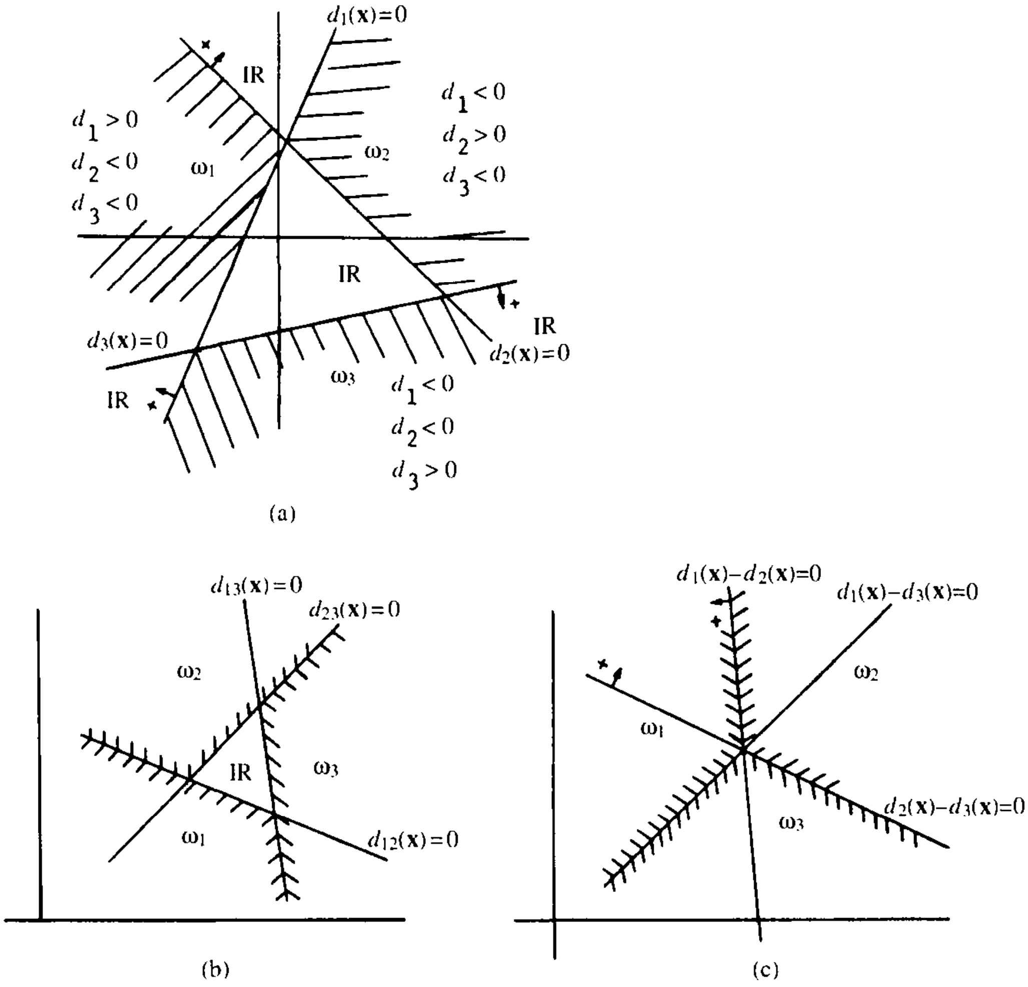


FIGURE 3.10 Three different cases in pattern classification: (a) each class is separable from others by a single decision surface; (b) each class is pairwise separable from others by a distinct decision surface; (c) same as part (b) but with no indeterminate regions.

and

$$\mathbf{x} \in \omega_i \qquad \text{when } d_{ij} > 0, \forall j \neq i \tag{3.27}$$

Case 3. Each pattern class is pairwise separable from every other class by a distinct decision surface, but with no interdeterminate regions (this is a special case of case 2; see Figure 3.10c). In this case, there are M decision functions and

$$d_k(\mathbf{x}) = (\mathbf{w}_k)^T\mathbf{x} \qquad k = 1, 2, \ldots, M \tag{3.28}$$

and

$$\mathbf{x} \in \omega_i \qquad \text{if } d_i(\mathbf{x}) > d_j(\mathbf{x}), \forall j \neq i \tag{3.29}$$

3.3.2 Layered Machine

A two-layered machine is also known as a *committee machine*, so named because it takes a fair vote for each linear discriminant function output to determine the classification. That part to the left of the $\sum$ unit shown in Figure 3.11 is the first layer and that to the right of it is the second layer. $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_R$ are the different n-dimensional weight vectors for each discriminant functions and are, respectively,

$$\begin{aligned} \mathbf{w}_1^T &= (w_{11}, w_{12}, \ldots, w_{1n}) \\ \mathbf{w}_2^T &= (w_{21}, w_{22}, \ldots, w_{2n}) \\ &\vdots \\ \mathbf{w}_R^T &= (w_{R1}, W_{R2}, \ldots, w_{Rn}) \end{aligned} \tag{3.30}$$

The first layer consists of an odd number of linear discriminant surfaces whose outputs are clipped by the threshold logic unit as +1 or −1, depending on the value of $f(\mathbf{x})$ to describe on which half of the feature space the particular pattern input falls. The second layer is a single linear surface with unity weight vector used to determine to which class the particular pattern will finally be assigned.

When an adaptive loop is placed in the threshold logic unit for the training of $\mathbf{w}$, the threshold logic unit is called an *adaptive linear threshold element* (ADALINE). When multiple ADALINEs are used in the machine, it is called a MADALINE—a committee machine.

Let us take a simple two-class problem to interpret the machine geometrically. Suppose that we have three threshold logic units in the first layer; i.e., $R = 3$. $\mathbf{w}_1^T\mathbf{x} = 0$, $\mathbf{w}_2^T\mathbf{x} = 0$, and $\mathbf{w}_3^T\mathbf{x} = 0$ will define three hyperplanes in the feature space, shown in Figure 3.12.

The layered machine will divide the pattern space geometrically into seven regions. In Table 3.1 values are listed on $\mathbf{w}_1^T\mathbf{x}$, $\mathbf{w}_2^T\mathbf{x}$, and $\mathbf{w}_3^T\mathbf{x}$ in each region, with

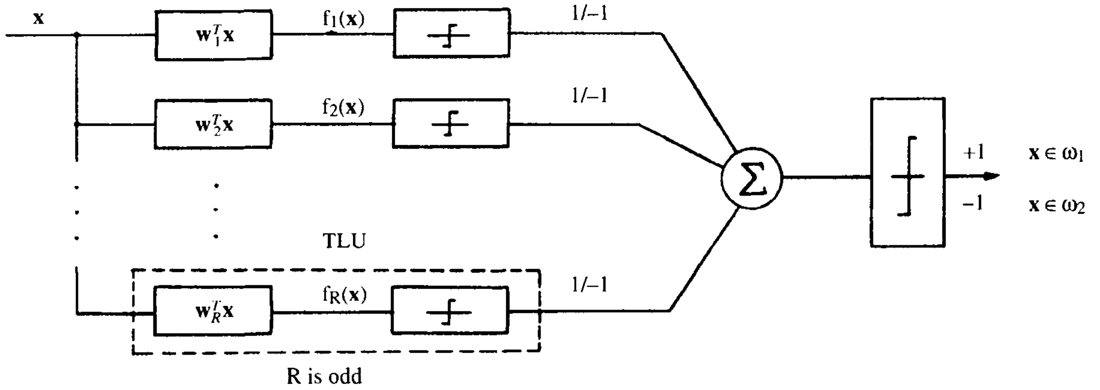


FIGURE 3.11 Two-layered machines.

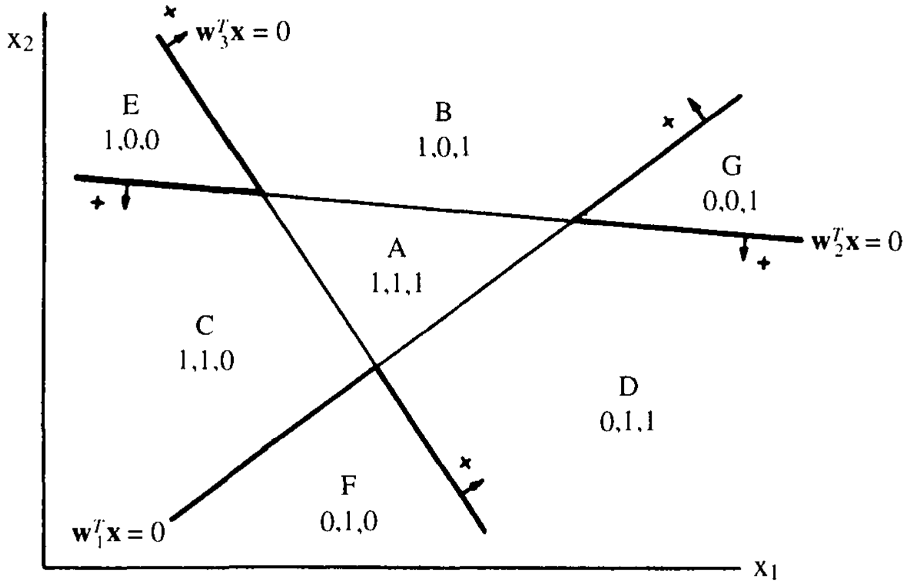


FIGURE 3.12 Geometrical interpretation of a simple two-class classification problem by a layered machine.

TABLE 3.1 Values of w_1^T, x, $w_2^T x$, and $w_3^T x$ in Different Regions of the Pattern Space

	Subregion							
$w^T x$	A	B	C	D	E	F	G	
$w_1^T x$	1	1	1	0	1	0	0	0
$w_2^T x$	1	0	1	1	0	1	0	0
$w_3^T x$	1	1	0	1	0	0	1	0
$x \in$	ω_1	ω_1	ω_1	ω_1	ω_2	ω_2	ω_2	—

1's and 0's denoting greater than and less than zero, respectively. No regions lie on the negative side of all these three hyperplanes, so (0, 0, 0) can never occur.

Four other threshold functions as shown in Figure 3.13 are also used in the processing elements. These threshold functions are: (a) the linear threshold function, (b) the ramp threshold function; (c) the step threshold function; and (d) the sigmoid threshold function. Note that all except (a) are nonlinear functions.

For a committee machine with five discriminant functions (or TLU units), only 15 subregions are available for classification (see Figure 3.14 and Table 3.2).

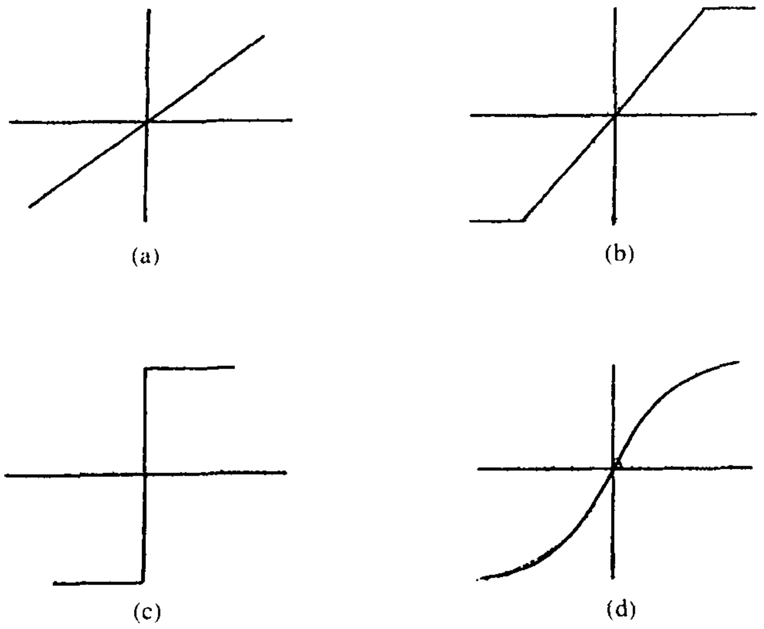


FIGURE 3.13 Four common threshold functions used in processing elements. (a) linear function; (b) ramp function; (c) step function; and (d) sigmoid function.

3.4 NONLINEAR DISCRIMINANT FUNCTIONS

The linear discriminant function is the simplest discriminant function. But in many cases the nonlinear discriminant function should be used. Quadratic discriminant functions have the following form:

$$d(\mathbf{x}) = \sum_{j=1}^{n} w_{jj}x_j^2 + \sum_{j=1}^{n-1}\sum_{k=j+1}^{n} w_{jk}x_jx_k + \sum_{j=1}^{n} w_jx_j + w_{n+1} \tag{3.31}$$

The first set of weights on the right-hand side of Eq. (3.31), $w_{jj}, j = 1, 2, \ldots, n$, consists of n weights; the second set, w_{jk}, $j = 1, 2, \ldots, n-1$, $k = 2, 3, \ldots, n$, consists of $n(n-1)/2$ weights; the third set, $w_j, j = 1, 2, \ldots, n, n$ weights; and the last set, w_{n+1}, only one weight. Hence, the total number of weights on $d(\mathbf{x})$ is $(n+1)(n+2)/2$. When expression (3.31) is put in matrix form,

$$d(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{B} + C \tag{3.32}$$

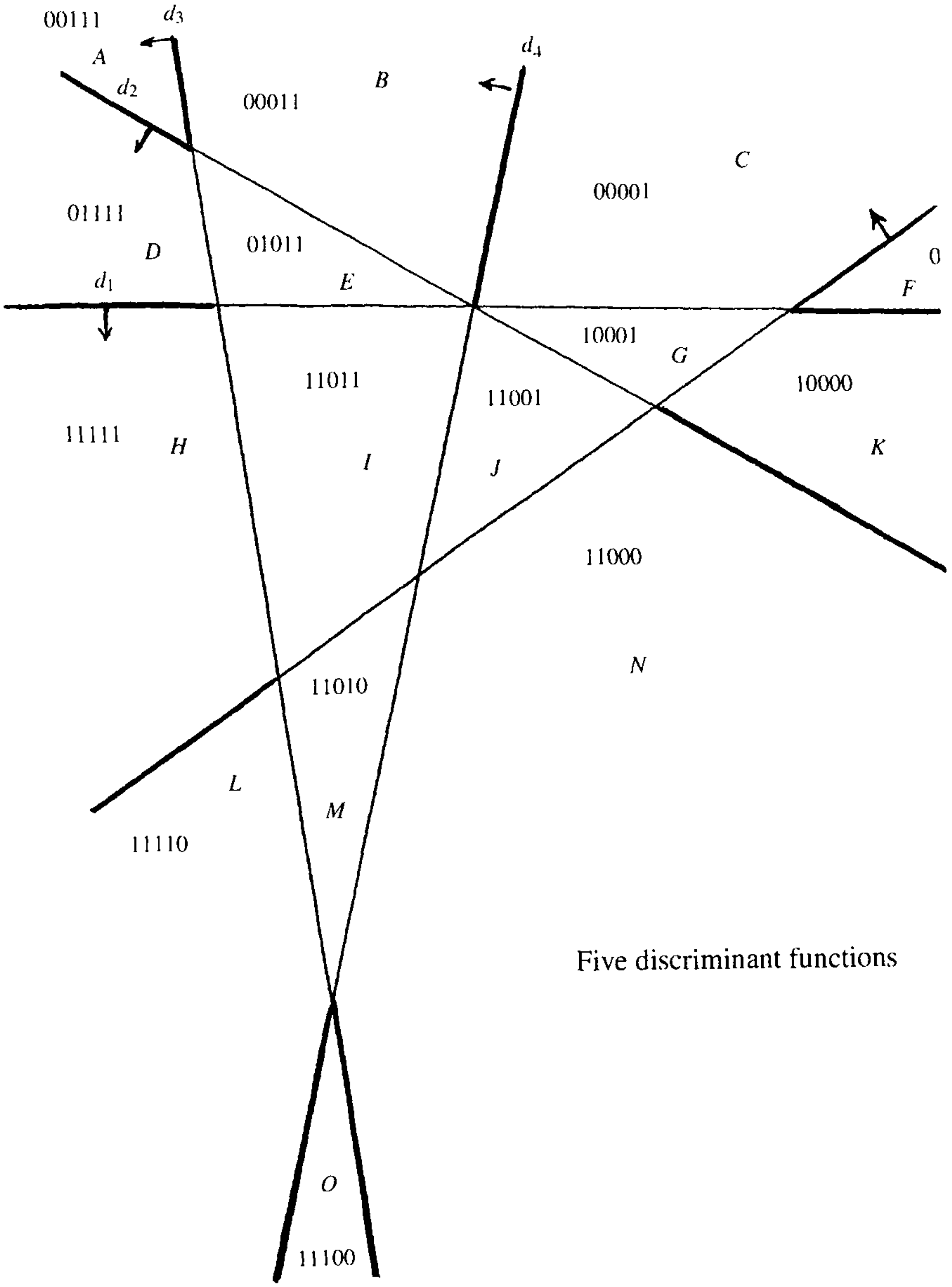


FIGURE 3.14 Geometrical interpretation of a simple two-class classification problem with five TLUs.

TABLE 3.2 Values of $\mathbf{w}_1^T\mathbf{x}$, $\mathbf{w}_2^T\mathbf{x}$, ..., $\mathbf{w}_5^T\mathbf{x}$ in Different Regions of the Pattern Space

	Subregion														
$\mathbf{w}^T\mathbf{x}$	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$\mathbf{w}_1^T\mathbf{x}$	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
$\mathbf{w}_2^T\mathbf{x}$	0	0	0	1	1	0	0	1	1	1	0	1	1	1	1
$\mathbf{w}_3^T\mathbf{x}$	1	0	0	1	0	0	0	1	0	0	0	1	0	0	1
$\mathbf{w}_4^T\mathbf{x}$	1	1	0	1	1	0	0	1	1	0	0	1	1	0	0
$\mathbf{w}_5^T\mathbf{x}$	1	1	1	1	1	0	1	1	1	1	0	0	0	0	0
$\mathbf{x} \in$	ω_1	ω_2	ω_2	ω_1	ω_1	ω_2	ω_2	ω_1	ω_1	ω_1	ω_2	ω_1	ω_1	ω_2	ω_1

where

$$\mathbf{A} = \begin{vmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & & & \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{vmatrix}$$

$$\mathbf{B} = \begin{vmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{vmatrix} \qquad \mathbf{C} = w_{n+1}$$

Note that if all the eigenvalues λ of $\mathbf{A}$ are positive, the quadratic form $\mathbf{x}^T\mathbf{A}\mathbf{x}$ will never be negative, and

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = 0 \qquad \text{iff } \mathbf{x} = 0 \tag{3.33}$$

That means that matrix $\mathbf{A}$ is positive definite and the quadratic form is also positive definite. But if one or more eigenvalues (λ's) equal zero while the others are positive, matrix $\mathbf{A}$ and the quadratic $\mathbf{x}^T\mathbf{A}\mathbf{x}$ are positive semidefinite.

Remember that on the decision surface, $d_i(\mathbf{x}) = d_j(\mathbf{x})$. In other words, the decision surface is defined by $d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0$. For the quadratic case, the quadratic decision surface is given by an equation of the form

$$\mathbf{x}^T[\mathbf{A}_i - \mathbf{A}_j]\mathbf{x} + \mathbf{x}^T[\mathbf{B}_i - \mathbf{B}_j] + [\mathbf{C}_i - \mathbf{C}_j] = 0 \tag{3.34}$$

Varieties of the quadratic surfaces can be defined for different values of A, which is equal to $\mathbf{A}_i - \mathbf{A}_j$.

If $\mathbf{A}$ is positive definitive, the decision surface is a hyperellipsoid with axes in the direction of the eigenvectors of $\mathbf{A}$. If $\mathbf{A} = a\mathbf{I}$, an identity matrix, the decision surface is a hypersphere. If **A** is positive semidefinite, the decision surface is a hyperellipsoidal cylinder whose cross sections are lower-dimensional hyperellipsoidals with axes in the direction of eigenvectors of **A** of nonzero eigenvectors. Otherwise (i.e., when none of the conditions above is fulfilled by **A** or **A** is negative definite), the decision surface is a hyperhyperboloid.

A quadratic discriminant function is much more complicated than a linear function. How to implement such a complicated function? Let us use a ϕ machine to treat this quadratic function as a linear problem.

3.5 ϕ MACHINES

3.5.1 Formulation

ϕ machines are a kind of classification system in which ϕ functions are used for pattern classification. The ϕ function (or generalized decision function) is a discriminant function that can be written in the form

$$d(\mathbf{x}) = \phi(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \cdots + w_M f_M(\mathbf{x}) + w_{M+1} \tag{3.35}$$

where the $f_i(\mathbf{x})$, $i = 1, 2, \ldots, M$, are linearly independent, real, and single-valued functions which are independent of the w_i (weights).

Note that $\phi(\mathbf{x})$ is linear with respect to w_i, but the $f_i(\mathbf{x})$ are not necessarily assumed to be linear. There are $M + 1$ degrees of freedom in this system. The same nonlinear discriminant function problem as used in Section 3.4 is taken for illustration.

$$\begin{aligned} d(\mathbf{x}) &= \sum_{j=1}^{n} w_{jj} x_j^2 + \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} w_{jk} x_j x_k + \sum_{j=1}^{n} w_j x_j + w_{n+1} \\ &= w_{11} x_1^2 + w_{22} x_2^2 + \cdots + w_{12} x_1 x_2 + x_{13} x_1 x_3 \\ &\quad + \cdots + w_{n-1,n} x_{n-1} x_n + w_1 x_1 + w_2 x_2 \\ &\quad + \cdots + w_n x_n + w_{n+1} \end{aligned} \tag{3.36}$$

A schematic diagram of the ϕ machine for this problem is shown in Figure 3.15. The **F** block is a quadratic processor and $\mathbf{F} = (f_1, f_2, f_3, \ldots, f_M)$. The first n component of **F** are $x_1^2, x_2^2, \ldots, x_n^2$; the next $n(n-1)/2$ components are all the pairs $x_1x_2, x_1x_3, \ldots, x_{n-1}x_n$; and the last n components are $x_1, x_2, \ldots, x_n$. The total number of the components is $M = n(n+3)/2$. We have then transformed from an n-dimensional pattern space to an M-dimensional ϕ space. A nonlinear problem is then put in a linear form.

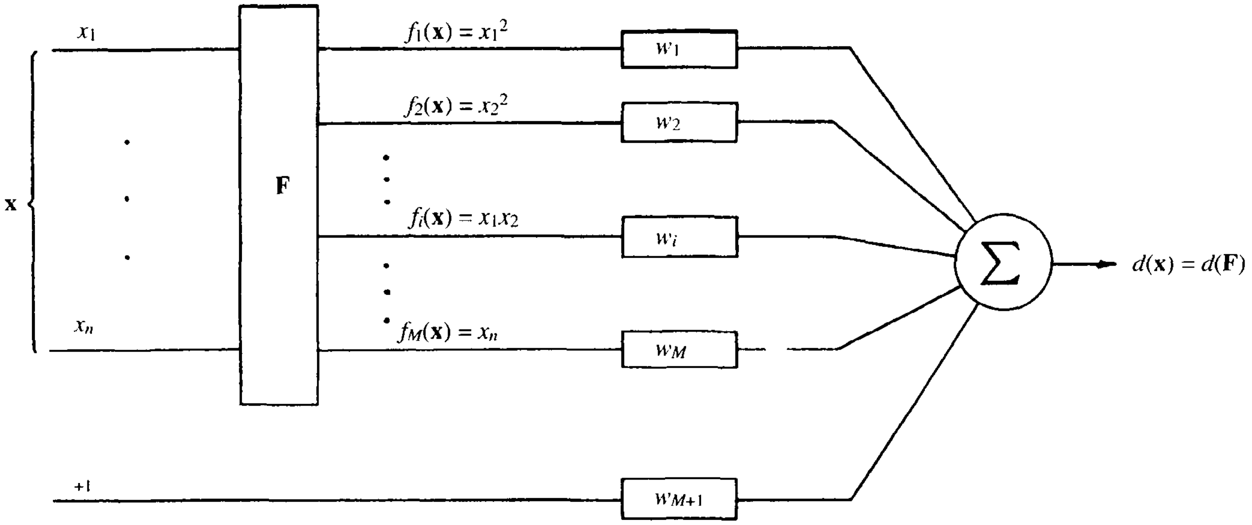

x
x_1
x_n
F
$f_1(\mathbf{x}) = x_1^2$
$f_2(\mathbf{x}) = x_2^2$
$f_i(\mathbf{x}) = x_1x_2$
$f_M(\mathbf{x}) = x_n$
w_1
w_2
w_i
w_M
w_{M+1}
+1
Σ
$d(\mathbf{x}) = d(\mathbf{F})$

FIGURE 3.15 ϕ machines.

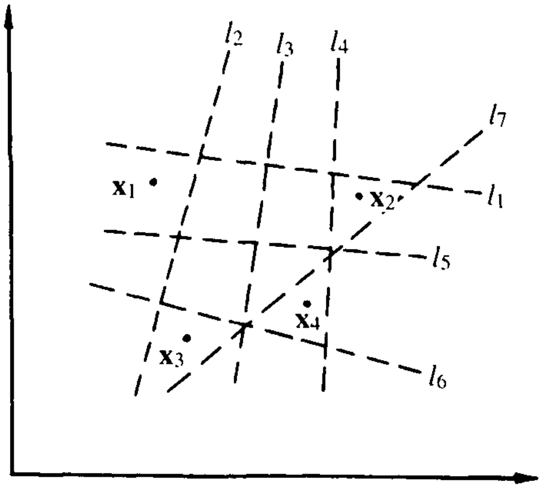


FIGURE 3.16 Linear dichotomization of four patterns.

3.5.2 Capacity of ϕ Machines for Classifying Patterns

Let us compute the number of dichotomies that can be obtained from N patterns. Assume that $M = 2$ and there are N n-dimensional patterns. Since each pattern may fall either in ω_1 or ω_2, there are 2^N distinct ways in which these N patterns could be dichotomized. For $N = 3$, we will have eight dichotomies, and for $N = 4$ we have 16 dichotomies. The total number of dichotomies that a linear discriminant function (ϕ space) can affect is dependent only on n and N, not on how the patterns lie in the pattern space in the form of the ϕ function.

Let $D(N, n)$ be the number of dichotomies that can be affected by a linear machine (linear dichotomies) on N patterns in n-dimensional space. In the four-pattern example given in Figure 3.16, l_2 dichotomizes $\mathbf{x}_1$ from $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$; l_5 dichotomizes $\mathbf{x}_1$ and $\mathbf{x}_2$ from $\mathbf{x}_3$ and $\mathbf{x}_4$; l_7 dichotomizes $\mathbf{x}_4$ from $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$; l_3 dichotomizes $\mathbf{x}_1$ and $\mathbf{x}_3$ from $\mathbf{x}_2$ and $\mathbf{x}_4$; and l_4 dichotomizes $\mathbf{x}_2$ from $\mathbf{x}_1$, $\mathbf{x}_3$, and $\mathbf{x}_4$. That is, in the problem we have here, $N = 4$ and $n = 2$, we have seven linear dichotomies. Each of them can divide the patterns in either one of two ways, such as in the dichotomy by l_3:

$$\mathbf{x}_1, \mathbf{x}_3 \in \omega_1 \qquad \mathbf{x}_2, \mathbf{x}_4 \in \omega_2$$

or

$$\mathbf{x}_1, \mathbf{x}_3 \in \omega_2 \qquad \mathbf{x}_2, \mathbf{x}_4, \in \omega_1$$

The number of linear dichotomies of N points in n-dimensional Euclidean space is equal to twice the number of ways in which the points can be partitioned by an $(n - 1)$-dimensional hyperplane; so

$$D(4, 2) = 2 \cdot 7 = 14$$

Comparing the total number of dichotomoties, $2^N = 16$, we find that two of these are not linearly implementable. It is not difficult to see that $\mathbf{x}_1$ and $\mathbf{x}_4$ cannot be linearly separated from $\mathbf{x}_2$ and $\mathbf{x}_3$.

In general, for a set of N points in an n-dimensional space with the assumption that no subsets of $(n+1)$ points lie on an $(n-1)$-dimensional plane, we can use the recursion relation

$$D(N,n) = D(N-1,n) + D(N-1,n-1) \tag{3.37}$$

to solve for $D(N,n)$. In particular,

$$D(1,n) = 2 \quad \text{and} \quad D(N,1) = 2N \tag{3.38}$$

Then

$$D(N,n) = \begin{cases} 2\sum_{k=0}^{n} \binom{N-1}{k} & N > n+1 \\ 2^N & N \leq n+1 \end{cases} \tag{3.39}$$

where

$$\binom{N-1}{k} = \frac{(N-1)!}{k!(N-1-k)!}$$

Now, let us generalize this problem by finding the probability of the dichotomy that can be implemented. Given a ϕ machine and a set of N patterns in the pattern space, there are 2^N possible dichotomies and any one of the 2^N dichotomies can be picked up with probability

$$p = 2^{-N}$$

For the generalized decision function

$$d(\mathbf{x}) = \phi(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \cdots + w_M f_M(\mathbf{x}) + w_{M+1} \tag{3.40}$$

the probability $p_{N,M}$ that any one dichotomy can be implemented is

$$\begin{aligned} p_{N,M} &= \frac{\text{number of } \phi \text{ dichotomies}}{\text{total possible number of dichotomies}} \\ &= \frac{D(N,M)}{2^N} \end{aligned} \tag{3.41}$$

$$p_{N,M} = 2^{-N} \begin{cases} 2\sum_{k=0}^{M} \binom{N-1}{k} & N > M+1 \\ 2^N & N \leq M+1 \end{cases} \tag{3.42}$$

or

$$p_{N,M} = \begin{cases} 2^{1-N} \sum_{k=0}^{M} \binom{N-1}{k} & N > M+1 \\ 1 & N \leq M+1 \end{cases} . \tag{3.43}$$

Note that $p_{N,M} = 1$ for $N \leq M + 1$, which means that if the number of patterns is less than the number of weights available for adjustment for the generalized decision function, the patterns will always be linearly separable in the M-dimensional pattern space. But when $N > M + 1$, the probability of dichotomization will go down depending on the ratio of N to $M + 1$. Figure 3.17 gives a plot which shows the relation of $p_{N,M}$ with λ, where λ is the ratio of N to $M + 1$. Note that curves with various values of M intersect at a point $p_{N,M} = 0.5$ when $\lambda = 2$. For large values of M, we almost have the ability to totally classify $N = 2(M + 1)$ well-distributed patterns with the generalized decision function of $M + 1$ parameters. On the contrary, if N is greater than $2(M + 1)$, the probability in achieving a dichotomy, $p_{N,M}$, declines sharply for similar values of M. Therefore, the dichotomization capacity C of the generalized decision functions equals $2(M + 1)$.

Although the analysis does not tell us how to choose $d(\mathbf{x})$ or $\phi(\mathbf{x})$, it does tell us something about the ability of the machine to dichotomize patterns. Suppose we have a total of N patterns that properly represent two classes, we are almost sure that we could find a good classifier if M is large. For example, for a

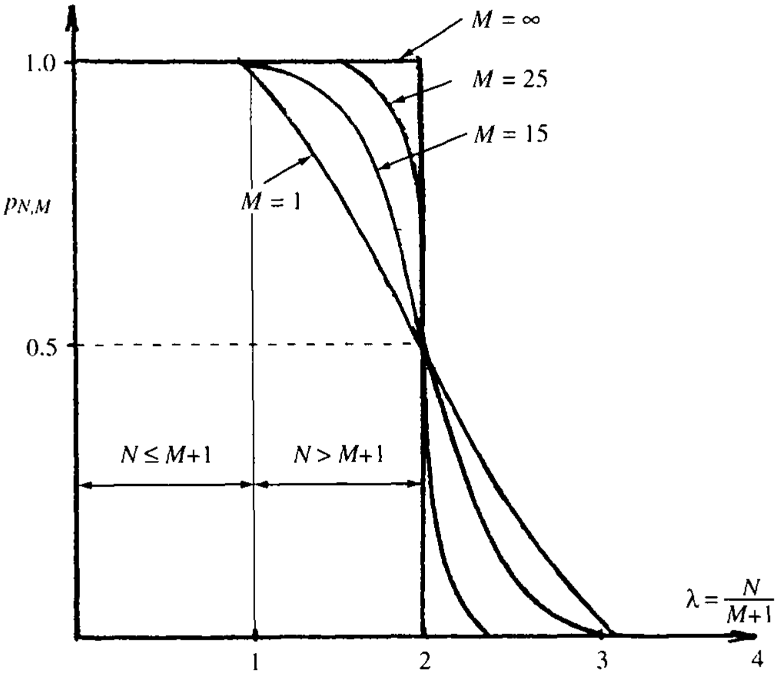


FIGURE 3.17 A $p_{N,M}$ versus λ plot.

two-class three-dimensional pattern classification problem with a quadratic discriminant function $d(\mathbf{x})$, we have

$$M = \frac{n(n+3)}{2} = 9$$

Then, the capacity of dichotomization

$$C = 2(M+1) = 20$$

If $N < 20$, we have a pretty good choice! This example also tells us how many prototypes we need for a good training set without causing endlessly forward and backward adjustment of the weights.

3.6 POTENTIAL FUNCTIONS AS DISCRIMINANT FUNCTIONS

A potential function $\Psi(\mathbf{x}, \mathbf{z}_k^m)$ is known as a *kernel* in the probability density function estimator, or is a function of $\mathbf{x}$ and $\mathbf{z}_k^m$ defined over the pattern space, where $\mathbf{z}_k^m$ is the mth prototype defining class ω_k. The potential function is better illustrated by Figure 3.18 for a one-dimensional pattern space. This potential gives the decreasing relationship between point $\mathbf{z}_k^m$ and point $\mathbf{x}$ as the distance $d(\mathbf{x}, \mathbf{z}_k^m)$ between these two points increases.

Superposition of the individual kernel "potential" functions will be used as a discriminant function

$$d_k(\mathbf{x}) = \frac{\sum_{m=1}^{N_k} \psi(\mathbf{x}, \mathbf{z}_k^m)}{N_k} \tag{3.44}$$

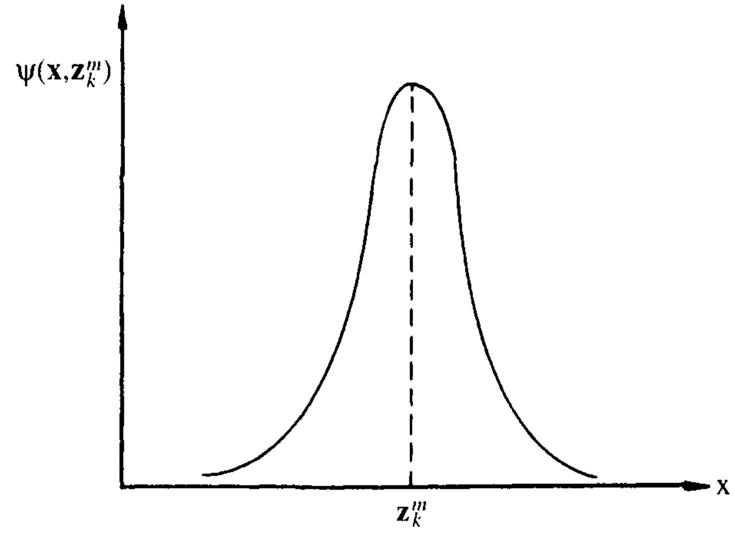


FIGURE 3.18 Potential function of one variable.

which is defined for class k, where N_k is the number of prototypes in class k. Functions ψ may be different between classes or even between prototypes within a class. The average of these potentials of prototypes from a given class indicates a degree of membership of the point $\mathbf{x}$ in the class.

The following characteristics of ψ are desirable:

1. $\psi(\mathbf{x}, \mathbf{z})$ should be maximum for $\mathbf{x} = \mathbf{z}$.
2. $\psi(\mathbf{x}, \mathbf{z})$ should be approximately zero for $\mathbf{x}$ distant from $\mathbf{z}$ in the region of interest.
3. $\psi(\mathbf{x}, \mathbf{z})$ should be a smooth (continuous) function arid tend to decrease in a monotonic fashion with the increase of the distance $d(\mathbf{x}, \mathbf{z})$.
4. If $\psi(\mathbf{x}_1, \mathbf{z}) = \psi(\mathbf{x}_2, \mathbf{z})$, patterns $\mathbf{x}_1$ and $\mathbf{x}_2$ should have approximately the same degree of similarity to $\mathbf{z}$.

If a set of potential functions are found which form a satisfactory discriminant function as

$$d_k(\mathbf{x}) > d_j(\mathbf{x}) \qquad \text{when } \mathbf{x} \in \omega_k, \forall j, k \tag{3.45}$$

then

$$f(\mathbf{x}) + d_k(\mathbf{x}) > f(\mathbf{x}) + d_j(\mathbf{x}) \qquad \forall f(\mathbf{x}) \tag{3.46}$$

and

$$f(\mathbf{x})\, d_k(\mathbf{x}) > f(\mathbf{x})\, d_j(\mathbf{x}) \qquad \forall f(\mathbf{x}) > 0 \tag{3.47}$$

This will help to simplify the computation of Ψ and ultimately the computation of $d(\mathbf{x})$. Since, for example, if

$$\begin{aligned} \Psi_1(\mathbf{x}, \mathbf{z}) &= \exp[-(\mathbf{x} - \mathbf{z})^T(\mathbf{x} - \mathbf{z})] \\ &= \exp[-|\mathbf{x}|^2 - |\mathbf{z}|^2 + 2\mathbf{x}^T\mathbf{z}] \end{aligned} \tag{3.48}$$

after multiplying $\Psi_1(\mathbf{x}, \mathbf{z})$ by $f(\mathbf{x}) = \exp|\mathbf{x}|^2$, we obtain

$$\begin{aligned} \Psi_2 &= f(\mathbf{x})\Psi_1(\mathbf{x}, \mathbf{z}) \\ &= \exp[2\mathbf{x}^T\mathbf{z} - |\mathbf{z}|^2] \end{aligned} \tag{3.49}$$

which will be much simpler than that of $\Psi_1(\mathbf{x}, \mathbf{z})$.

Another form of potential function can also be chosen for sample pattern $\mathbf{z}$:

$$\psi(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{r} \lambda_i^2 \phi_i(\mathbf{x})\phi_i(\mathbf{z}) \tag{3.50}$$

where λ_i, $i = 1, 2, \ldots,$ are constants and ϕ_i, $i = 1, 2, \ldots,$ are orthonormal functions, such that

$$\sum_{i=1}^{\infty} \phi_i(\mathbf{x})\phi_i(\mathbf{z}) = \delta(\mathbf{x} - \mathbf{z}) \tag{3.51}$$

If $[\phi_i]$ is a complete orthonormal set, then for the decision function d_k,

$$\begin{aligned} d_k(\mathbf{x}) &= \frac{1}{N_k}\sum_{m=1}^{N_k} \Psi(\mathbf{x}, \mathbf{z}_k^m) \\ &= \frac{1}{N_k}\sum_{m=1}^{N_k}\sum_{i=1}^{r} \lambda_i^2\phi_i(\mathbf{x})\phi_i(\mathbf{z}_k^m) \\ &= \sum_{i=1}^{r}\phi_i(\mathbf{x})\left(\frac{1}{N_k}\right)\sum_{m=1}^{N_k}\lambda_i^2\phi_i(\mathbf{z}_k^m) \\ &= \sum_{i=1}^{r} C_i^k\phi_i(\mathbf{x}) \end{aligned} \tag{3.52}$$

where

$$C_i^k = \frac{1}{N_k}\sum_{m=1}^{N_k}\lambda_i^2\phi_i(\mathbf{z}_k^m)$$

This procedure is most attractive when either the number of samples N_k is small or the dimensionality of $\mathbf{x}$ is sufficiently small to allow $d(\mathbf{x})$ to be stored as a table for discrete values of $\mathbf{x}$. But if the number of samples is large, computation problems will be severe and storage problems may occur.

PROBLEMS

3.1 Let $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M]$ be the set of all sample points. Find the normalized variables. *Note:* In the derivation of the expression for the normalized variables, N classes and M_N samples for the class N are assumed.

3.2 Discuss whether it would be successful for us to apply the method of successive dichotomies to the problem described by Figure P3.2.

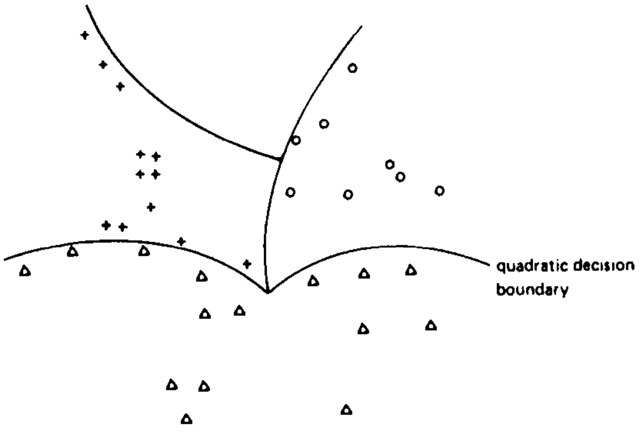


FIGURE P3.2

3.3 You are given the following prototypes in augmented space:

$$[(6, 4, 1), (5, 2, 1), (1, 3, 1), (0, -5, 1)] \in S_1$$
$$[(5, -1, 1), (4, -1, 1), (3, 1, 1)] \in S_2$$

You are also informed that a layered machine (three TLUs in the first layer and one TLU in the second layer) might be a useful tool in dichotomizing the prototypes properly. Suppose that the following weight vectors have been selected for the first-layer TLU:

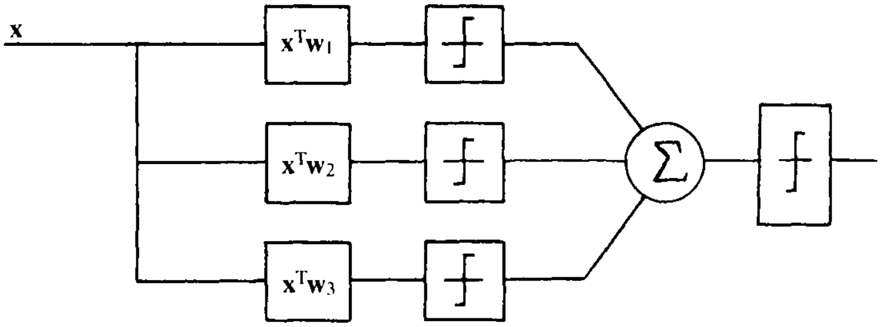


FIGURE P3.3

$$\mathbf{W}_1 = -1, 1, 4$$
$$\mathbf{W}_2 = 1, 1, -1$$
$$\mathbf{W}_3 = -\tfrac{1}{2}, 1, 0$$

(a) Compute and plot the prototypes in the first layer space.

(b) Will a committee machine separate these prototypes with the TLUs, as shown in Figure P3.3

(c) Which prototype must be removed for the first-layer space to be linearly separable?

(d) What happens to the committee machine when the weight vectors are changed to the following?

$$\mathbf{W}_1 = -1, -1, 4$$
$$\mathbf{W}_2 = 1, -1, -1$$
$$\mathbf{W}_3 = -\tfrac{1}{2}, -1, 0$$

3.4 The following three decision functions are given for a three-class problem:

$$d_1(\mathbf{x}) = 10x_1 - x_2 - 10 = 0$$
$$d_2(\mathbf{x}) = x_1 + 2x_2 - 10 = 0$$
$$d_3(\mathbf{x}) = x_1 - 2x_2 - 10 = 0$$

(a) Sketch the decision boundary and regions for each pattern class, assuming that each pattern class is separable from all other classes by a single decision surface.

(b) Assuming that each pattern class is pairwise separable from every other class by a distinct decision surface, and letting $d_{12}(\mathbf{x}) = d_1(\mathbf{x})$, $d_{13}(\mathbf{x}) = d_2(\mathbf{x})$, and $d_{23}(\mathbf{x}) = d_3(\mathbf{x})$ as listed above, sketch the decision boundary and regions for each pattern class.

3.5 (a) Prove that curves in Figure 3.17 with various values of M (M varies from 1 to very large values) will intersect at a point when $\lambda = 2$.

(b) Find the proper number of prototypes for a training set without causing endlessly forward and backward adjustment of the weights (synapses).

3.6 Interpret geometrically a simple two-class classification problem by a layered machine with five discriminant functions.

3.7 Find the number of dichotomies for the five patterns shown in Figure P3.7.

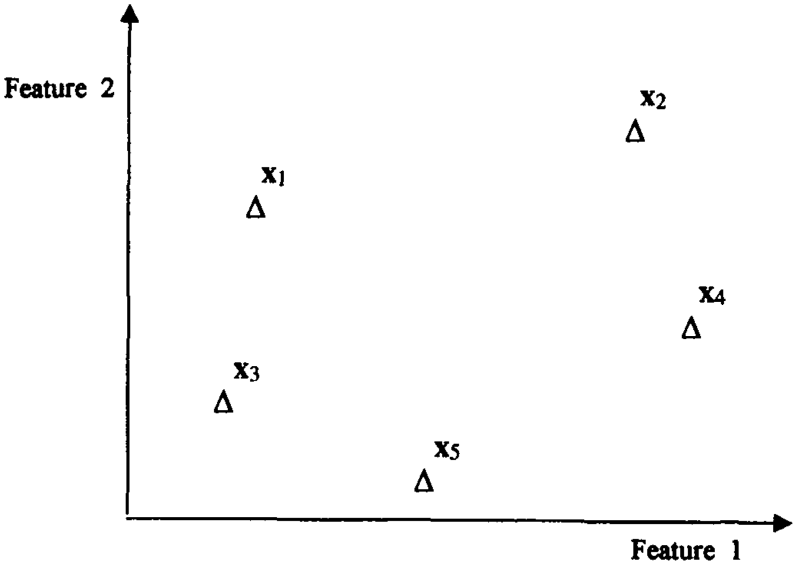


FIGURE P3.7

4

Nonparametric (Distribution-Free) Training of Discriminant Functions

4.1 WEIGHT SPACE

We have already discussed the fact that a pattern vector $\mathbf{x}$ appears as a point in the pattern space, and that a pattern space can be partitioned into subregions for patterns belonging to different categories. The decision surfaces that partition the space may be linear, piecewise linear, or nonlinear, and can be generalized as

$$d(\mathbf{x}) = f(\mathbf{w}, \mathbf{x}) \tag{4.1}$$

where $d(\cdot)$ is called the discriminant function, and

$$\mathbf{x} = (x_1, x_2, \ldots, x_n, 1)^T \qquad \text{and} \qquad \mathbf{w} = (w_1, w_2, \ldots, w_n, w_{n+1})^T$$

represent the augmented pattern and weight vectors, respectively. The problem of training a system is actually to find the weight vector $\mathbf{w}$ shown in Eq. (4.1) with the a priori information obtained from the training samples. It is possible and perhaps much more convenient to investigate the behavior of the training algorithms in a weight space. The weight space is an $(n+1)$-dimensional euclidean space in which the coordinate variables are $w_1, w_2, \ldots, w_n, w_{n+1}$. For each prototype $\mathbf{z}_k^m$, $k = 1, 2, \ldots, M$, $m = 1, 2, \ldots, N_k$ (where M represents the number of categories and N_k represents the number of prototypes belonging to category k), there is in W space (weight space) a hyperplane on which

$$\mathbf{w}^T\mathbf{z}_n^m = 0 \tag{4.2}$$

Any weight vector $\mathbf{w}$ on the positive side of the hyperplane yields $\mathbf{w}^T\mathbf{z} > 0$. That is, if the prototype $\mathbf{z}_k^m$ belongs to category ω_k, any weight vector $\mathbf{w}$ on this side of the hyperplane will probably correctly classify $\mathbf{z}_k^m$ as in ω_k. A similar argument can be made for any weight vector on the other side of this hyperplane, where $\mathbf{w}^T\mathbf{z} < 0$.

Let us take a two-class problem for illustration. Suppose that we have a set of N_1 patterns belonging to ω_1 and a set of N_2 patterns belonging to ω_2, with the total number of patterns $N = N_1 + N_2$. Assume also that ω_1 and ω_2 are two linearly separable classes. Then a vector $\mathbf{w}$ can be found such that

$$\mathbf{w}^T\mathbf{z}_1^m > 0 \qquad \forall\mathbf{z}_1^m \in \omega_1, \qquad m = 1, 2, \ldots, N_1 \tag{4.3}$$

and

$$\mathbf{w}^T\mathbf{z}_2^m < 0 \qquad \forall\mathbf{z}_2^m \in \omega_2, \qquad m = 1, 2, \ldots, N_2 \tag{4.4}$$

where $\mathbf{z}_1^m$ and $\mathbf{z}_2^m$ represent all the prototypes in categories ω_1 and ω_2, respectively. In general, for N patterns there are N pattern hyperplanes in the weight space. The solution region for category ω_1 in W space is that region which lies on the positive side of the N_1 hyperplanes for category ω_1 and on the negative side of the N_2 hyperplanes for category ω_2.

Suppose that we have three prototypes $\mathbf{z}_1^1, \mathbf{z}_1^2$, and $\mathbf{z}_1^3$, and know that all of them belong to category ω_1. Three hyperplanes can then be drawn in the W space, as shown in Figure. 4.1a. The shaded area in Figure 4.1a shows the solution region in this two-class problem. In this region

$$d_1(\mathbf{w}, \mathbf{z}) > 0$$

$$d_2(\mathbf{w}, \mathbf{z}) > 0$$

and

$$d_3(\mathbf{w}, \mathbf{z}) > 0$$

That is, any $\mathbf{w}$ in this region will probably classify the prototypes $\mathbf{z}_1^1, \mathbf{z}_1^2$, and $\mathbf{z}_1^3$ as belonging to category ω_1, while in the cross-hatched area shown in Figure 4.1b,

$$d_1(\mathbf{w}, \mathbf{z}) > 0$$

$$d_2(\mathbf{w}, \mathbf{z}) > 0$$

but

$$d_3(\mathbf{w}, \mathbf{z}) < 0$$

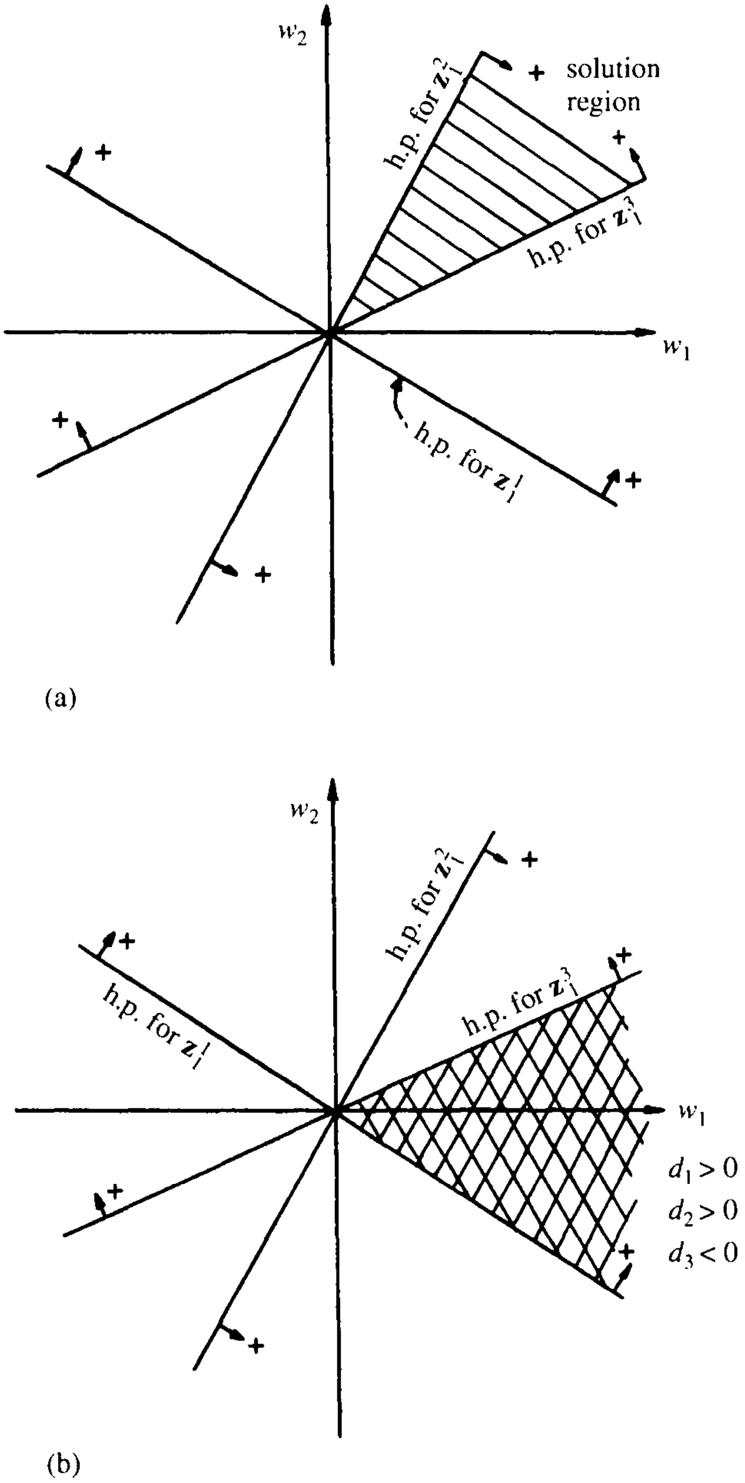

w_2
h.p. for $\mathbf{z}_1^2$
+ solution region
+
h.p. for $\mathbf{z}_1^3$
w_1
h.p. for $\mathbf{z}_1^1$
(a)
w_2
h.p. for $\mathbf{z}_1^2$
h.p. for $\mathbf{z}_1^1$
h.p. for $\mathbf{z}_1^3$
w_1
$d_1 > 0$
$d_2 > 0$
$d_3 < 0$
(b)

FIGURE 4.1 Hyperplanes in W space. $\uparrow^+$ indicates the positive half-plane of the hyperplane.

Any **w** over this region will classify $\mathbf{z}_1^1$ and $\mathbf{z}_1^2$ as being in category ω_1, but classify $\mathbf{z}_1^3$ as being in category ω_2.

As discussed in Chapter 3, the decision surface for a two-class problem is assumed to have the property that $d(\mathbf{w}, \mathbf{x})$ will be greater than zero for all patterns of one class, but less than zero for all patterns belonging to the other class. But if all the $\mathbf{z}_2^m$'s are replaced by their negatives, $-\mathbf{z}_2^m$'s, the solution region can be generalized as that part of the **W** space for which

$$\mathbf{w}^T\mathbf{z} > 0 \qquad \forall \mathbf{z} = \mathbf{z}_1^m, -\mathbf{z}_2^m \tag{4.5}$$

Our problem then simply becomes to find a **w** such that all inequalities are greater than zero.

It might be desirable to have a margin (or threshold) in the discriminant function such that

$$\mathbf{w}^T\mathbf{z} > T \qquad \forall \mathbf{z} = \mathbf{z}_1^m, -\mathbf{z}_2^m \tag{4.6}$$

where $T > 0$ is the margin (or threshold) chosen. Any **w** satisfying inequality (4.6) is a weight solution vector. The solution region is now changed as shown in Figure 4.2.

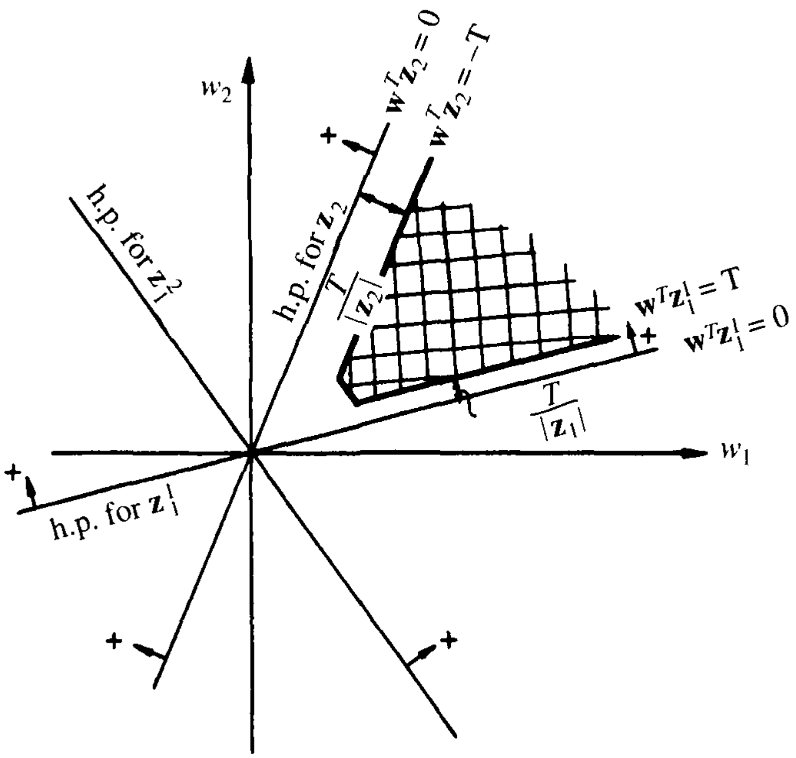

w_2
w_1
$\mathbf{w}^T\mathbf{z}_2 = 0$
$\mathbf{w}^T\mathbf{z}_2 = -T$
h.p. for $\mathbf{z}_1^2$
h.p. for $\mathbf{z}_2$
$\frac{T}{|\mathbf{z}_2|}$
$\mathbf{w}^T\mathbf{z}_1^1 = T$
$\mathbf{w}^T\mathbf{z}_1^1 = 0$
$\frac{T}{|\mathbf{z}_1|}$
h.p. for $\mathbf{z}_1^1$
+
+
+
+
+

FIGURE 4.2 Solution region for a two-class problem with margin set for each decision surface.

In the cross-hatched region, both $\mathbf{w}^T\mathbf{z}_1^1$ and $\mathbf{w}^T\mathbf{z}_1^2$ are positive, while $\mathbf{w}^T\mathbf{z}_2 < 0$. Note that along the original pattern hyperplane

$$\mathbf{w}^T\mathbf{z} = 0 \tag{4.7}$$

and that the vector $\mathbf{z}$ (augmented $\mathbf{z}$) is perpendicular to the hyperplane $\mathbf{w}^T\mathbf{z} = 0$ and heads in its positive direction. Thus the line $\mathbf{w}^T\mathbf{z} = T$ is offset from $\mathbf{w}^T\mathbf{z} = 0$ by a distance $\Delta = T/|z|$. The proof of this is left to the reader as a problem.

4.2 ERROR CORRECTION TRAINING PROCEDURES

It is obvious that for a two-class problem an error would exist if

$$\mathbf{w}^T\mathbf{z}_1^m < 0(T) \tag{4.8}$$

$$\mathbf{w}^T\mathbf{z}_2^m > 0(T) \tag{4.9}$$

Then we need to move the weight vector $\mathbf{w}$ to the positive side of the pattern hyperplane for $\mathbf{z}_i^m$, in other words, move the vector $\mathbf{w}$ to the correct solution region.

The most direct way of doing this is to move $\mathbf{w}$ in a direction perpendicular to the hyperplane (i.e., in a direction of $\mathbf{z}_1^m$ or $-\mathbf{z}_2^m$). In general, the correction of the $\mathbf{w}$ can be formulated as: Replace $\mathbf{w}(k)$ by $\mathbf{w}(k+1)$ such that

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) + c\mathbf{z}_1^m && \text{if } \mathbf{w}^T(k)\mathbf{z}_1^m < 0(\mathrm{T}) \\ \mathbf{w}(k+1) &= \mathbf{w}(k) - c\mathbf{z}_2^m && \text{if } \mathbf{w}^T(k)\mathbf{z}_2^m > 0(-T) \\ \mathbf{w}(k+1) &= \mathbf{w}(k) && \text{if correctly classified} \end{aligned} \tag{4.10}$$

where $\mathbf{w}(k)$ and $\mathbf{w}(k+1)$ are the weight vectors at the kth and $(k+1)$th correction steps, respectively. To add a correction term $c\mathbf{z}_1^m$ implies moving vector $\mathbf{w}$ in the direction of $\mathbf{z}_1^m$. Similarly, subtracting a correction term $c\mathbf{z}_2^m$ implies moving vector $\mathbf{w}$ in the direction of $-\mathbf{z}_2^m$.

During this training period, patterns are presented one at a time through all $N = N_1 + N_2$ prototypes (training patterns). Each complete pass through all the N patterns is called an *iteration*. After one iteration, all patterns are presented again in the same sequence to carry on another iteration. This is repeated until no corrections are made through one complete iteration.

Several rules can be set up in choosing the value of c: the fixed increment rule, absolute correction rule, fractional correction rule, and so on.

4.2.1 Fixed-Increment Rule

In this algorithm, c is chosen to be a positive fixed constant. This algorithm begins with any $\mathbf{w}(0)$, and Eq. (4.10) is applied to the training sequence P, where $P = [\mathbf{z}_1^1, \mathbf{z}_2^1, \ldots, \mathbf{z}_1^{N_1}, \mathbf{z}_2^{N_2}]$. The whole weight-adjustment process will terminate in some finite steps, k.

The choise of c for this process is actually not important. If the theorem holds for $c = 1$, it holds for any $c \neq 1$, since this, in effect, just scales all patterns by some amount without changing their separability.

4.2.2 Absolute Correction Rule

In this algorithm c is chosen to be the smallest integer that will make $\mathbf{w}(k+1)$ cross the pattern hyperplane into the solution region of W space each time a classification error is made. Let $\mathbf{z}_i'$ be the average of the sample vectors that do not satisfy the inequality $\mathbf{w}^T \cdot \mathbf{z} \geq T$. The constant c is chosen such that

$$\mathbf{w}^T(k+1)\mathbf{z}_i' = [\mathbf{w}(k) + c\mathbf{z}_i']^T \mathbf{z}_i' > T \tag{4.11}$$

or

$$c\mathbf{z}_t'^T \mathbf{z}_i' > T - \mathbf{w}^T(k)\mathbf{z}_i' > 0 \tag{4.12}$$

and therefore

$$c > \frac{T - \mathbf{w}^T(k)\mathbf{z}_i'}{\mathbf{z}_i'^T \mathbf{z}_i'} = \frac{T - \mathbf{w}^T(k)\mathbf{z}_i'}{|\mathbf{z}_i'|^2} \tag{4.13}$$

Note that if $T = 0$, $-\mathbf{w}^T(k)\mathbf{z}_i'$ must be greater than zero, or $\mathbf{w}^T(k)\mathbf{z}_i' < 0$. Taking its absolute value into consideration, Eq. (4.13) becomes

$$c > \frac{|\mathbf{w}^T(k)\mathbf{z}_i'|}{|\mathbf{z}_i'|^2} \tag{4.14}$$

The absolute correction rule will also yield a solution weight vector in a finite number of steps.

4.2.3 Fractional Correction Rule

In W space, the augmented pattern vector $\mathbf{z}$ is perpendicular to the hyperplane $\mathbf{w}^T\mathbf{z} = 0$ and heads in the positive direction, as shown in Figure 4.3. The distance from $\mathbf{w}(k)$ to the desired hyperplane is

$$\begin{aligned} D = \Delta + \delta &= \frac{T}{|\mathbf{z}_i'|} + \left|\mathbf{w}(k) \cdot \frac{\mathbf{z}_i'}{|\mathbf{z}_i'|}\right| \\ &= \frac{T}{|\mathbf{z}_i'|} + \frac{|\mathbf{w}^T(k)\mathbf{z}_i'|}{|\mathbf{z}_i'|} \end{aligned} \tag{4.15}$$

When $\mathbf{w}(k)$ is on the other side on the hyperplane,

$$D = \frac{T - |\mathbf{w}^T(k)\mathbf{z}_i'|}{|\mathbf{z}_i'|} \tag{4.16}$$

In the fractional correction algorithm c is chosen so that $\mathbf{w}$ is moved by a fraction of the distance in a direction normal to the desired hyperplane. That is,

$$c = \lambda\frac{D}{|\mathbf{z}_i'|} \qquad \lambda > 0 \tag{4.17}$$

and

$$\mathbf{w}(k+1) - \mathbf{w}(k) = \lambda D\frac{\mathbf{z}_i'}{|\mathbf{z}_i'|} \tag{4.18}$$

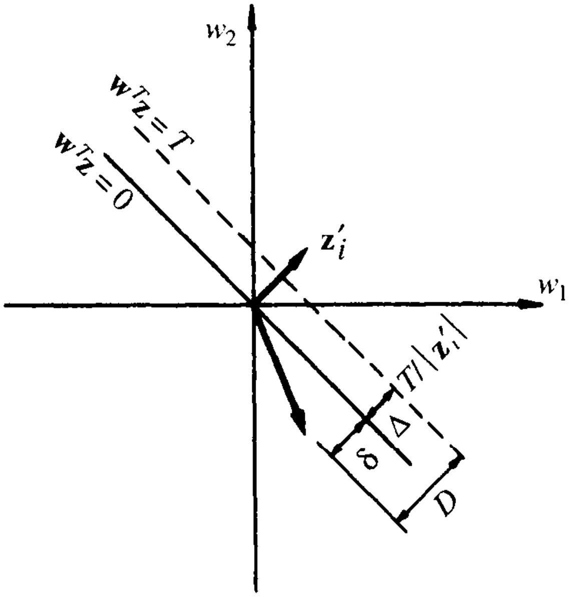


FIGURE 4.3 Augmented pattern vector in W space.

If the threshold is set at 0, then

$$
\begin{aligned}
c &= \lambda \frac{|\mathbf{w}^T(k)\mathbf{z}'_i|}{|\mathbf{z}'_i|^2}, \qquad \lambda > 0 \\
D &= \frac{T - |w^T(k)\mathbf{z}_i|}{|\mathbf{z}'_i|}
\end{aligned}
\tag{4.19}
$$

It can be seen that when $\lambda = 1$, the correction is to the hyperplane (absolute correction rule); when $\lambda < 1$, the correction is short of the hyperplane (underrelaxation); and when $\lambda > 1$, the correction is over the hyperplane (overrelaxation). For $0 < \lambda < 2$, the fractional correction rule will either terminate on a solution weight vector in a finite number of steps or else converge to a point on the boundary of the solution weight space.

The training procedure can then be generalized for any of the foregoing three algorithms as follows:

1. Take each $\mathbf{z}$ from the training set and test $d(\mathbf{z})$ for its category belonging. $M = 2$ is assumed here.
2. If a correct answer is obtained, go to the next $\mathbf{z}$.
3. If misclassification occurs, correct $\mathbf{w}(k)$ with $\mathbf{w}(k+1)$.
4. After all $\mathbf{z}$'s from the training set have been examined, repeat the entire process in the same (or different) sequential order. If the $\mathbf{z}$'s are linearly separable, all three of these algorithms will converge to a correct $\mathbf{w}$.

Figure 4.4 shows an example of training a two-category classification problem with two sets of prototypes, namely,

$$\mathbf{z}_1^1, \mathbf{z}_1^2 \in \omega_1$$

and

$$\mathbf{z}_2^1, \mathbf{z}_2^2 \in \omega_2$$

Absolute correction rule is used in this example. Hyperplanes for the prototypes $\mathbf{z}_1^1, \mathbf{z}_1^2$, $\mathbf{z}_2^1, \mathbf{z}_2^2$ can be drawn on the two-dimensional W space, when these four prototype vectors $\mathbf{z}_i^m, i = 1, 2, m = 1, 2$ are given. The initial weight vector $\mathbf{W}$ was chosen randomly. Assume that it was chosen at position a on Figure 4.4. When $\mathbf{z}_1^1$ is presented to the system, $d(\mathbf{z}_1^1)$ should be greater than zero; i.e., the $\mathbf{W}$ vector should be on the positive side of $\mathbf{z}_1^1$ hyperplane. But it is not with the present position of the weight vector, and therefore the weight vector $\mathbf{W}$ should be adjusted to position b. At this time when $\mathbf{z}_2^1$ is presented to the system, this weight vector $\mathbf{W}$ now lies on the positive side of the $\mathbf{z}_2^1$ hyperplanes. This is also not correct (see the figure), so the $\mathbf{W}$ vector should again be adjusted to a right position c relative to the said hyperplane.

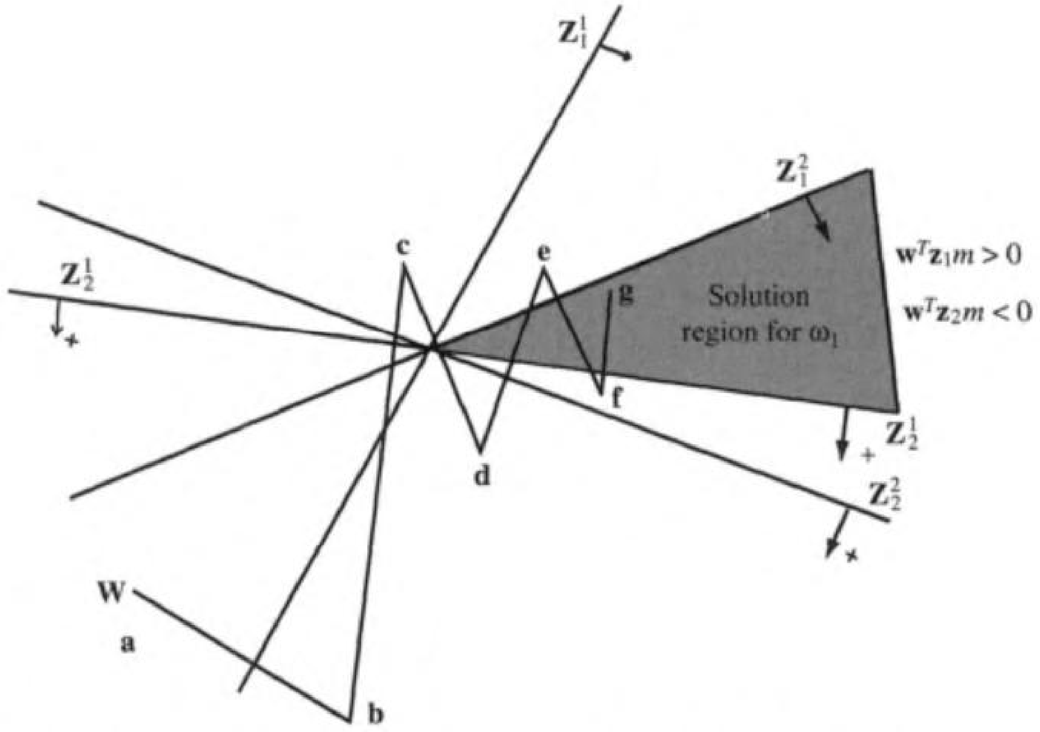


FIGURE 4.4 Training of the two-category classification system with prototypes.

Let us repeatedly present the prototypes to the system in a random order as shown in Table 4.1, and the weight vector is adjusted accordingly. Table 4.1 and Figure 4.4 show the sequence of the weight adjustments. The weight vector **W** eventually stops at the solution region where no more adjustment is needed whenever and in any order a prototype is presented to the system. The system is then said trained.

Figure 4.5 shows the correction steps for the above three different procedures. Absolute correction terminates in three steps, whereas fractional correction terminates in four steps.

TABLE 4.1 Adjustment of the Weight Vector During the Training Period

	Order of presentation															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Prototypes $\mathbf{z}_i^m$	$\mathbf{z}_1^1$	$\mathbf{z}_2^1$	$\mathbf{z}_1^2$	$\mathbf{z}_2^2$	$\mathbf{z}_1^1$	$\mathbf{z}_2^1$	$\mathbf{z}_1^2$	$\mathbf{z}_2^2$	$\mathbf{z}_1^1$	$\mathbf{z}_2^1$	$\mathbf{z}_1^2$	$\mathbf{z}_2^2$	$\mathbf{z}_1^1$	$\mathbf{z}_2^1$	$\mathbf{z}_1^2$	$\mathbf{z}_2^1$
$d(\mathbf{z}_i^m) = \mathbf{W}^T\mathbf{z}_i^m$ evaluated on the $\mathbf{z}_i^m$-hyperplane	−	+	−	+	+	−	−	−	+	+	+	−	+	−	+	−
$d(\mathbf{z}_i^m) = \mathbf{W}^T\mathbf{z}_i^m$ should be	+	−	+	−	+	−	+	−	+	−	+	−	+	−	+	−
W adjustment needed	Y	Y	Y	Y	N	N	Y	N	N	Y	N	N	N	N	N	N
W moves to position	b	c	d	e	e	e	f	f	f	g	g	g	g	g	g	g

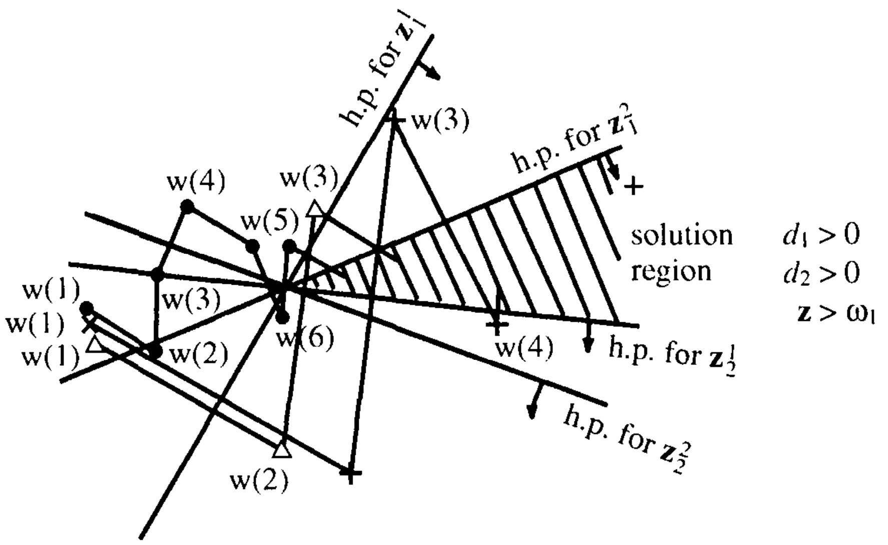


FIGURE 4.5 Weight vector adjustment procedures. ●, Fixed increment; △, absolute correction; +, fractional correction.

For classes greater than 2 ($M > 2$), similar procedures can be followed. Assume that we have training sets available for all pattern classes $\omega_i, i = 1, 2, \ldots, M$. Compute the discriminant function $d_i(\mathbf{z}) = \mathbf{w}_i^T\mathbf{z}, i = 1, 2, \ldots, M$. Obviously, we desire

$$d_i(\mathbf{z}) > d_j(\mathbf{z}) \qquad \text{if } \mathbf{z} \in \omega_i, \forall j \neq i \tag{4.20}$$

If so, the weight vectors are not adjusted. But if

$$d_j(\mathbf{z}) > d_i(\mathbf{z}) \qquad \text{if } \mathbf{z} \in \omega_i, \forall j \neq i \tag{4.21}$$

misclassification occurs, and weight adjustment will be needed. Under these circumstances the following adjustment can be made for the fixed-increment correction rule:

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + c\mathbf{z} \tag{4.22}$$

$$\mathbf{w}_j(k+1) = \mathbf{w}_j(k) - c\mathbf{z} \tag{4.23}$$

$$\mathbf{w}_l(k+1) = \mathbf{w}_l(k) \tag{4.24}$$

where k and $k+1$ denote the kth and $(k+1)$th iteration steps. Equation (4.23) is for those $\mathbf{z}_j$'s that make $d_j(\mathbf{z}) > d_i(\mathbf{z})$, and Eq. (4.24) is for those $\mathbf{z}_l$'s that are neither i nor those making an incorrect classification. If the classes are separable, this algorithm will converge in a finite number of iterations. Similar adjustments can be derived for the absolute and fractional correction algorithms.

4.3 GRADIENT TECHNIQUES

4.3.1 General Gradient Descent Technique

The gradient descent technique is another approach to train the system. A gradient vector possesses the important property of pointing in the direction of the maximum rate of increase of the function when the argument increases. The weight-adjustment procedure can then be formulated as

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \rho_k \nabla J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}(k)} \tag{4.25}$$

where $J(\mathbf{w})$ is an index of performance or a criterion function that is to be minimized by adjusting $\mathbf{w}$. Minimum $J(\mathbf{w})$ can be approached by moving $\mathbf{w}$ in the direction of the negative of the gradient. The procedure can be summarized as follows:

1. Start with some arbitrarily chosen weight vector $\mathbf{w}(1)$ and compute the gradient vector $\nabla J[\mathbf{w}(1)]$.
2. Obtain the next value $\mathbf{w}(2)$ by moving some distance from $\mathbf{w}(1)$ in the direction of steepest descent.

ρ_k in Eq. (4.25) is a positive scale factor that sets the step size. For its optimum choice, let us assume that $J(\mathbf{w})$ can be approximated by

$$\begin{aligned} J[\mathbf{w}(k+1)] \simeq J[\mathbf{w}(k)] &+ [\mathbf{w}(k+1) - \mathbf{w}(k)]^T \nabla J[\mathbf{w}(k)] \\ &+ \tfrac{1}{2}[\mathbf{w}(k+1) - \mathbf{w}(k)]^T D[\mathbf{w}(k+1) - \mathbf{w}(k)] \end{aligned} \tag{4.26}$$

where

$$D = \left.\frac{\partial^2 J}{\partial \mathbf{w}_i \partial \mathbf{w}_j}\right|_{\mathbf{w}=\mathbf{w}(k)}$$

Substitution of Eq. (4.25) into Eq. (4.26) yields

$$J[\mathbf{w}(k+1)] \simeq J[\mathbf{w}(k)] - \rho_k |\nabla J[\mathbf{w}(k)]|^2 + \tfrac{1}{2}\rho_k^2 \nabla J^T D \nabla J \tag{4.27}$$

Setting $\partial J[\mathbf{w}(k+1)]/\partial \rho_k = 0$ for minimum $J[\mathbf{w}(k+1)]$, we obtain

$$|\nabla J|^2 = \rho_k \nabla J^T D \nabla J \tag{4.28}$$

or

$$\rho_k \left.\frac{|\nabla J|^2}{\nabla J^T D \nabla J}\right|_{\mathbf{w}=\mathbf{w}(k)} \tag{4.29}$$

which is equivalent to Newton's algorithm for optimum descent, in which

$$\rho_k = D^{-1} \tag{4.30}$$

Some problems may exist with this optimum ρ_k: D^{-1} in Eq. (4.29) may not exist; the matrix operations involved are time consuming and expensive; and the assumption of the second-order surface may be incorrect. For those reasons, setting ρ_k equal to a constant may do just as well.

4.3.2 Perceptron Criterion Function

Let the criterion function be

$$J_p(\mathbf{w}) = \sum_{\mathbf{z}\in P} (-\mathbf{w}^T\mathbf{z}) \tag{4.31}$$

where the summation is over the incorrectly classified pattern samples. Geometrically, $J_p(\mathbf{w})$ is proportional to the sum of the distances of the misclassified patterns to the hyperplane. Taking the derivative of $J_p(\mathbf{w})$ with respect to $\mathbf{w}(k)$ yields

$$\nabla J_p[\mathbf{w}(k)] = \sum_{\mathbf{z}\in P} \mathbf{z} \tag{4.32}$$

where $\mathbf{w}(k)$ denotes the value of $\mathbf{w}$ at the kth iteration step. The perceptron training algorithm can then be formulated as

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \rho_k \nabla J_p[\mathbf{w}(k)] \tag{4.33}$$

or

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \rho_k \sum_{\mathbf{z}\in P} \mathbf{z} \tag{4.34}$$

where P is the set of samples misclassified by $\mathbf{w}(k)$. Equation (4.34) can be thus interpreted to mean that the $(k+1)$th weight vector can be obtained by adding some multiple of the sum of the misclassified samples to the kth weight vector. This is a "many-at-a-time" procedure, since we determine $\mathbf{w}^T\mathbf{z}$ for all $\mathbf{z}$ and adjust only after all patterns were classified.

If we make an adjustment after each incorrectly classified pattern (we call it "one-at-a-time" procedure), the criterion function becomes

$$J(\mathbf{w}) = -\mathbf{w}^T\mathbf{z} \tag{4.35}$$

and

$$\nabla J(\mathbf{w}) = -\mathbf{z} \tag{4.36}$$

The training algorithm is to make

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \rho_k \mathbf{z} \tag{4.37}$$

This is the fixed-increment rule if $\rho_k = c$ (a constant).

4.3.3 Relaxation Criterion Function

The criterion function used in this algorithm is chosen to be

$$J_r(\mathbf{w}) = \tfrac{1}{2}\sum_{\mathbf{z}\in P}\frac{(-\mathbf{w}^T\mathbf{z}+b)^2}{|\mathbf{z}|^2} \tag{4.38}$$

Again, P is the set of samples misclassified by $\mathbf{w}$. That is, P consists of those $\mathbf{z}$'s for which $-\mathbf{w}^T\mathbf{z}+b > 0$ or $\mathbf{w}^T\mathbf{z} < b$. The gradient of $J_r(\mathbf{w})$ with respect to $\mathbf{w}(k)$ yields

$$\nabla J_r(\mathbf{w}) = -\sum_{\mathbf{z}\in P}\frac{-\mathbf{w}^T\mathbf{z}+b}{|\mathbf{z}|^2}\mathbf{z} \tag{4.39}$$

The basic relaxation training algorithm is then formulated as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \rho_k\sum_{\mathbf{z}\in P}\frac{-\mathbf{w}^T(k)\mathbf{z}+b}{|\mathbf{z}|^2}\mathbf{z} \tag{4.40}$$

This is also a many-at-a-time algorithm. Its corresponding one-at-a-time algorithm is

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \rho_k\frac{-\mathbf{w}^T(k)\mathbf{z}+b}{|\mathbf{z}|^2}\mathbf{z} \tag{4.41}$$

which becomes the fractional correction algorithm with $\lambda = \rho_k$.

4.4 TRAINING PROCEDURES FOR THE COMMITTEE MACHINE

In general, no convergence theorems exist for the training procedures of committee (or other piecewise linear) machines. One procedure that frequently is satisfactory is given here. Assume that $M = 2$ and that there are R discriminant functions, where R is odd. Then

$$d_i(\mathbf{z}) = \mathbf{w}_i^T(k)\mathbf{z} \qquad i = 1, 2, \ldots, R \tag{4.42}$$

The classification of the committee machines will then be made according to

$$d(\mathbf{z}) = \sum_{i=1}^{R}\operatorname{sgn} d_i(\mathbf{z}) \tag{4.43}$$

such that

$$\begin{aligned}&\mathbf{z} \text{ is assigned to } \omega_1 \text{ when } d(\mathbf{z}) > 0\\ &\mathbf{z} \text{ is assigned to } \omega_2 \text{ when } d(\mathbf{z}) < 0\end{aligned} \tag{4.44}$$

where

$$\operatorname{sgn} d_i(\mathbf{z}) = \begin{cases} +1 & \text{if } d_i(\mathbf{z}) \geq 0 \\ -1 & \text{if } d_i(\mathbf{z}) < 0 \end{cases} \tag{4.45}$$

Note that since R is odd, $d(\mathbf{z})$ cannot be zero and will also be odd. Thus $d(\mathbf{z})$ is equal to the difference between the number of $d_i(\mathbf{z}) > 0$ and that of $d_i(\mathbf{z}) < 0$ for a weight vector $\mathbf{w}_i(k)$ at the kth iteration step. In this regard, we always desire $d(\mathbf{z}) > 0$. In other words, we want to have more weight vectors that yield $d_i(\mathbf{z}) > 0$ than those which yield $d_i(\mathbf{z}) < 0$.

When $d(\mathbf{z}) < 0$, incorrect classification results. It will be obvious that in this case there will be $[R + d(\mathbf{z})]/2$ weight vectors among the $\mathbf{w}_i(k)$, $i = 1, \ldots, R$, which yield negative responses $[d_i(\mathbf{z}) < 0]$ and $[R - d(\mathbf{z})]/2$ weight vectors which yield positive responses $[d_i(\mathbf{z}) > 0]$. To obtain correct classification, we then need to change at least n responses of the $\mathbf{w}_i(k)$ from -1 to $+1$, where n can be found by setting up the equation

$$\left[\frac{R - d(\mathbf{z})}{2} + n\right] - \left[\frac{R + d(\mathbf{z})}{2} - n\right] = 1 \tag{4.46}$$

The first set of brackets represents the number of d_i that are presently greater than zero; the set of brackets after the minus sign represents the number of d_i that are presently less than zero. The minimum value of n is then

$$n_{\min} = \frac{d(\mathbf{z}) + 1}{2} \tag{4.47}$$

which is the minimum number of weight vectors needed to be adjusted. The procedure for the weight vector adjustment will then be as follows:

1. Pick out the least negative $d_i(\mathbf{z})$'s among those negative $d_i(\mathbf{z})$'s.
2. Adjust the $[d(\mathbf{z}) + 1]/2$ weight vectors that have the least negative $d_i(\mathbf{z})$'s by the following rule:

   $$\mathbf{w}_i(k + 1) = \mathbf{w}_i(k) + c\mathbf{z} \tag{4.48}$$

   so that their resulting $d_i(\mathbf{z})$'s become positive. All the other weight vectors are left unaltered at this stage.
3. If at the kth stage, the machine incorrectly classifies a pattern that should belong to ω_2, give the correction increment c a negative value, such as

   $$\mathbf{w}_i(k + 1) = \mathbf{w}_i(k) - c\mathbf{z} \tag{4.49}$$

4.5 PRACTICAL CONSIDERATIONS CONCERNING ERROR CORRECTION TRAINING METHODS

Since error-correcting rules never allow an error in pattern classification without adjusting the discriminant function, some oscillations may result. For example, for the case of two normally distributed classes with overlap, an error will always occur even if the optimum hyperplane is found. The error correction rule will cause the hyperplane continually to be adjusted and never stabilize at the optimum location.

For the case that classes have more than one "cluster" or "grouping" in the pattern space, the error correction training method will again encounter problems. The remedy is to add a stopping rule. But this stopping rule must be employed appropriately; otherwise, the system may terminate on a poor **w**. Another way of solving such problems is to go to a training procedure that is not error correcting, such as clustering (determining only the modes of a multimodal problem), stochastic approximation, potential functions, or the minimum squared error procedure.

4.6 MINIMUM-SQUARED-ERROR PROCEDURES

4.6.1 Minimum Squared Error and the Pseudoinverse Method

Consider that we wish to have the equalities

$$\mathbf{Zw} = \mathbf{b} \tag{4.50}$$

instead of the inequalities $\mathbf{zw} > 0$. Then we are required to solve $N = \sum_{i=1}^{M} N_i$ linear equations, where N is the total number of prototypes for all classes, N_i is that for class ω_i, and M is the total number of classes. **z** and **b** can then be defined, respectively, as

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_n^T \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \vdots & & & \vdots \\ z_{N1} & z_{N2} & \cdots & z_{Nn} \end{bmatrix} \tag{4.51}$$

and

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \tag{4.52}$$

If $\mathbf{Z}$ were square and nonsingular, we could set

$$\mathbf{w} = \mathbf{Z}^{-1}\mathbf{b} \tag{4.53}$$

and solve for $\mathbf{w}$. But, in general, $\mathbf{Z}$ is rectagnular (i.e., more rows than columns) and many solutions to $\mathbf{Zw} = \mathbf{b}$ exist. Let us define an error vector as

$$\mathbf{e} = \mathbf{Zw} - \mathbf{b} \tag{4.54}$$

and a sum-of-squared-error criterion function as

$$J_s(\mathbf{w}) = \tfrac{1}{2}|\mathbf{e}|^2 = \tfrac{1}{2}|\mathbf{Zw} - \mathbf{b}|^2 = \tfrac{1}{2}\sum_{i=1}^{N}(\mathbf{z}_i\mathbf{w} - b_i)^2 \tag{4.55}$$

Taking the partial derivative of J_s with respect to $\mathbf{w}$, we obtain

$$\nabla J_s(\mathbf{w}) = \sum_{i=1}^{N}(\mathbf{z}_i\mathbf{w} - b_i)\mathbf{z}_i \tag{4.56}$$

or in matrix form,

$$\nabla J_s(\mathbf{w}) = \mathbf{Z}^T(\mathbf{Zw} - \mathbf{b}) \tag{4.57}$$

To obtain minimum square error, set $\nabla J_s(\mathbf{w}) = 0$. We then have

$$\mathbf{Z}^T\mathbf{w}\mathbf{Z} = \mathbf{Z}^T\mathbf{b} \tag{4.58}$$

or

$$\mathbf{w} = \mathbf{Z}^{\#}\mathbf{b} \tag{4.59}$$

where $\mathbf{Z}^{\#} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T$ is called the *pseudoinverse* or generalized inverse of $\mathbf{Z}$. $\mathbf{Z}^{\#}$ has the following properties: (1) $\mathbf{Z}^{\#}\mathbf{Z} = I$, but in general, $\mathbf{ZZ}^{\#} \neq I$; and (2) $\mathbf{Z}^{\#} = \mathbf{Z}^{-1}$ if $\mathbf{Z}$ is square and nonsingular. The value of $\mathbf{b}$ in Eq. (4.52) may be set arbitrarily except that $b_i > 0, \forall i$. If no other information is available, a good choice is

$$\mathbf{b} = [1 \quad 1 \quad 1 \quad \cdots \quad 1] = \mathbf{u}^T$$

In fact, if $\mathbf{b} = \mathbf{u}^T$, the minimum-squared-error solution approaches a minimum-mean-squared-error approximation to the Bayes discriminant function. Note that this method is not error correcting, since it does not compute new $\mathbf{w}$ for every $\mathbf{z}$ In fact, all $\mathbf{z}$ are considered together and only one solution is needed; therefore, the training time is very short.

4.6.2 Ho-Kashyap Method

When the criterion function $J(\mathbf{w}))$ is to be minimum not only with respect to $\mathbf{w}$, but also with respect to $\mathbf{b}$ (i.e., assume that $\mathbf{b}$ is not a constant), the training algorithm is known as the Ho-Kashyap method. The same criterion function J as that shown in Eq. (4.55) is to be used and repeated here:

$$J(\mathbf{w}) = \tfrac{1}{2}|\mathbf{Zw} - \mathbf{b}|^2 \tag{4.60}$$

Partial derivatives of $J(\mathbf{w})$ with respect to $\mathbf{w}$ and $\mathbf{b}$ are, respectively,

$$\frac{\partial J(\mathbf{w})}{\partial w} = \mathbf{Z}^T(\mathbf{Zw} - \mathbf{b}) \tag{4.61}$$

and

$$\frac{\partial J(\mathbf{w})}{\partial b} = -(\mathbf{Zw} - \mathbf{b}) \tag{4.62}$$

Setting $\partial J/\partial w = 0$ yields

$$\mathbf{w} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{b} = \mathbf{Z}^{\#}\mathbf{b} \tag{4.63}$$

Since all components of $\mathbf{b}$ are constrained to be positive, adjustments on $\mathbf{b}$ can be made such that

$$\mathbf{b}(k+1) = \mathbf{b}(k) + \delta\mathbf{b}(k) \tag{4.64}$$

where

$$\delta b_i(k) = \begin{cases} 2c[\mathbf{e}(k)] & \text{where } \mathbf{e}(k) > 0 \\ 0 & \text{when } \mathbf{e}(k) \leq 0 \end{cases} \tag{4.65}$$

where k, i, and c represent the iteration index, the component index of the vector, and the positive correction increment, respectively. From Eq. (4.63) we have

$$\mathbf{w}(k+1) = \mathbf{Z}^{\#}\mathbf{b}(k+1) \tag{4.66}$$

Combining Eqs. (4.63), (4.64), and (4.66), we obtain

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{Z}^{\#}\delta\mathbf{b}(k) \tag{4.67}$$

Remembering that the components of $\mathbf{b} = (b_1, b_2, \ldots, b_N)^T$ are all positive, that is,

$$\mathbf{w}(1) = \mathbf{Z}^{\#}\mathbf{b}(1) \quad \mathbf{b}(1) > 0 \tag{4.68}$$

$$\mathbf{e}(k) = \mathbf{Zw}(k) - \mathbf{b}(k) \tag{4.69}$$

the algorithm for the weight and b adjustments can be put in the following form:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + c\mathbf{Z}^{\#}[\mathbf{e}(k) + |\mathbf{e}(k)|] \tag{4.70}$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + c[\mathbf{e}(k) + |\mathbf{e}(k)|] \tag{4.71}$$

4.6.3 Widrow-Hoff Rule

If either $\mathbf{Z}^T\mathbf{Z}$ is singular or the matrix operations in finding $\mathbf{Z}^{\#}$ are unwieldy, we can minimize $J_s(\mathbf{w})$ by a gradient descent procedure:

Step 1. Choose $\mathbf{w}(0)$ arbitrarily.
Step 2. Adjust the weight vector such that

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \rho_k \nabla J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}(k)}$$

or

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \rho_k \mathbf{Z}^T[\mathbf{w}(k)\mathbf{Z} - \mathbf{b}]$$

If ρ_k is chosen to be ρ_1/k, it can be shown that this converges to a limiting weight vector $\mathbf{w}$ satisfying

$$\nabla J_s(\mathbf{w}) = \mathbf{Z}^T(\mathbf{w}\mathbf{Z} - \mathbf{b}) = 0$$

In this algorithm matrix operations are still required, but the storage requirements are usually less here than with the $\mathbf{Z}^{\#}$ above.

PROBLEMS

4.1 Given the sample vectors

$$\mathbf{z}_1 = (0, 0)$$
$$\mathbf{z}_2 = (-1, -1)$$
$$\mathbf{z}_3 = (2, 2)$$
$$\mathbf{4}_4 = (4, 0)$$
$$\mathbf{z}_5 = (4, 1)$$

where $[\mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5] \in \omega_2$ and $[\mathbf{z}_1, \mathbf{z}_2] \in \omega_1$. If they are presented in numerical order repeatedly, give the sequence of weight vectors and the solution generated by using fixed increment correction rule. Start with $\mathbf{W}^T(1) = 0$.

4.2 Repeat Problem 4.1 with the following sample vectors:

$$\mathbf{z}_1 = (0, 0) \in \omega_1$$
$$\mathbf{z}_2 = (3, 3) \in \omega_2$$
$$\mathbf{z}_3 = (-3, 3) \in \omega_3$$

Start with $\mathbf{W}_1(1) = \mathbf{W}_2(1) = \mathbf{W}_3(1) = (0, 0, 0)^T$.

4.3 Given the following set of data:

$$\begin{array}{ll} \mathbf{z}_1 = (0, 0, 0)^T & \mathbf{z}_2 = (1, 0, 0)^T \\ \mathbf{z}_3 = (1, 0, 1)^T & \mathbf{z}_4 = (1, 1, 0)^T \in \omega_1 \\ \mathbf{z}_5 = (0, 0, 1)^T & \mathbf{z}_6 = (0, 1, 1)^T \\ \mathbf{z}_7 = (0, 1, 0)^T & \mathbf{z}_8 = (1, 1, 1)^T \in \omega_2 \end{array}$$

Find a solution weight vector using the perceptron algorithm. Start with $\mathbf{W}(1) = (-1, -2, -2, 0)^T$.

4.4 It is much more convenient to train the classification system in the weight space than in the pattern space. Why? Explain in detail.

4.5 To train a classification system, $\mathbf{z}_1^1, \mathbf{z}_1^2, \mathbf{z}_2^1, \mathbf{z}_2^2$, and $\mathbf{z}_2^3$ are used as the prototypes for the training, where

$$\mathbf{z}_1^1 = (2 \quad 4)^T \qquad \mathbf{z}_1^2 = (4, \quad 3)^T \in \omega_1$$

and

$$\mathbf{z}_2^1 = (-2 \quad 2)^T \qquad \mathbf{z}_2^2 = (-3, \quad 1)^T \qquad \mathbf{z}_2^3 = (-1 \quad 5) \in \omega_2$$

(a) Draw the hyperplanes respectively for these training pattern samples.

(b) If these pattern samples are presented to the system in the following order:

$$\mathbf{z}_1^1, \quad \mathbf{z}_2^1, \quad \mathbf{z}_1^2, \quad \mathbf{z}_2^2 \text{ and } \mathbf{z}_2^3$$

Show the weight vector adjustment procedure with the *absolute correction rule*. Start with $W^T(1) = (6 \quad 0)$.

4.6 Write a program to find the decision surface for the following known data:

$$\mathbf{z}_1^1 = (3 \quad 4),\ \mathbf{z}_1^2 = (2 \quad 6),\ \mathbf{z}_1^3 = (4 \quad 5);\ \mathbf{z}_1^4 = (3 \quad 5),$$
$$\mathbf{z}_1^5 = (2 \quad 4) \quad \text{in class } \omega_1$$
$$\mathbf{z}_2^1 = (-1 \quad 2),\ \mathbf{z}_2^2 = (-2 \quad 2),\ \mathbf{z}_2^3 = (-3 \quad 1);\ \mathbf{z}_2^4 = (-2 \quad -1);$$
$$\mathbf{z}_2^5 = (-3 \quad -3) \quad \text{in class } \omega_2$$

To start, the $\mathbf{w} = (w_1 \quad w_2 \quad w_3)$ can be selected as any values.

4.7 (a) Draw a three-dimensional diagram to show the *solution region* of Problem 4.1.

(b) Draw a three-dimensional diagram showing the step-by-step change of $\mathbf{W}$ for the following three cases:

(1) Order of presentation: $\mathbf{z}_1^1$, $\mathbf{z}_2^1$, $\mathbf{z}_1^2$, $\mathbf{z}_2^2$, $\mathbf{z}_1^3$, and repeat until all the prototypes are correctly classified.

(2) Order of presentation: $\mathbf{z}_1^1$, $\mathbf{z}_1^2$, $\mathbf{z}_2^1$, $\mathbf{z}_2^2$, $\mathbf{z}_2^3$, and repeat until all the prototypes are correctly classified.

(3) Order of presentation: Choose a random order of presentation by yourself.

5

Statistical Discriminant Functions

In this chapter we discuss primarily statistical discriminant functions used to deal with those sorts of classification in which pattern categories are known a priori to be characterizable by a set of parameters. First, formulation of the classification problem by means of statistical decision theory is introduced, and loss functions, Bayes' discriminant function, maximum likelihood decision, and so on, are discussed. Some analysis of the probability error is given.

The optimal discriminant function for normally distributed patterns is then discussed in more detail, followed by a discussion of how to determine the probability density function when it is unknown. At the end of the chapter, a large-data-set aerial-photo interpretation problem is taken as an example to link the theory we have discussed with the real-world problem we actually have.

5.1 INTRODUCTION

The use of statistical discriminant functions for classification is advantageous because (1) considerable knowledge already exists in areas such as statistical communication, detection theory, decision theory, and so on, and this knowledge is directly applicable to pattern recognition; and (2) statistical formulation is particularly suitable for the pattern recognition problem, since many pattern recognition processes are modeled statistically. In pattern recognition it is desirable to use all the a priori information available and the performance of the system is also often evaluated statistically.

In the training of a statistical classification system, an underlying distribution density function such as gaussian distribution or some other distribution function is assumed; however, no known distribution is assumed in nonparametric training, as we discussed in Chapters 3 and 4.

5.2 PROBLEM FORMULATION BY MEANS OF STATISTICAL DESIGN THEORY

5.2.1 Loss Function

Before we establish the loss functions, it will be helpful to make the following assumptions:

1. $p(\omega_i)$ is known or can be estimated.
2. $p(\mathbf{x}|\omega_i)$ is known or can be estimated directly from the training set.
3. $p(\omega_i|\mathbf{x})$ is generally not known.

Here $p(\omega_i)$ is the a priori probability of class ω_i, and $p(\mathbf{x}|\omega_i)$ is the likelihood function of class ω_i, or the state conditional probability density function of $\mathbf{x}$. More explicitly, it is the probability density function for $\mathbf{x}$ given that the state of nature is ω_i and $p(\omega_i|\mathbf{x})$ is the probability that $\mathbf{x}$ comes from ω_i. This is actually the a posteriori probability.

A loss function L_{ij} may be defined as the loss, cost, or penalty due to deciding that $\mathbf{x} \in \omega_j$ when, in fact, $\mathbf{x} \in \omega_i$. Thus we seek to minimize the average loss. Similarly, the conditional average loss or conditional average risk $r_k(\mathbf{x})$ may be defined as

$$r_k(\mathbf{x}) = \sum_{i=1}^{M} L_{ik} p(\omega_i|\mathbf{x}) \tag{5.1}$$

that is, the average or expected loss of misclassifying $\mathbf{x}$ as in ω_k; but in fact, it should be in some other classes ω_i, $i = 1, 2, \ldots, M$ and $i \neq k$.

The job of the classifier is then to find an optimal decision that will minimize the average risk or cost. The decision rule will then consist of the following steps:

1. Compute the expected losses, $r_i(\mathbf{x})$ of deciding that $\mathbf{x} \in \omega_i$, $\forall i, i = 1, 2, \ldots, M$.
2. Decide that $\mathbf{x} \in \omega_k$ if $r_k(\mathbf{x}) \leq r_i(\mathbf{x})$, $\quad \forall i, i \neq k$.

The corresponding discriminant function is then

$$d_k(\mathbf{x}) = -r_k(\mathbf{x}) \tag{5.2}$$

The negative sign in front of $r_k(\mathbf{x})$ is chosen so as to make $d_k(\mathbf{x})$ represent the most likely class. The smaller $r_k(\mathbf{x})$, the more likely it is that $\mathbf{x} \in \omega_k$.

A loss matrix can then be set up as

$$\mathbf{L} = \begin{bmatrix} 0 & & 1 \\ & \ddots & \\ 1 & & 0 \end{bmatrix} \tag{5.3}$$

where $L_{ii} = 0, i = 1, \ldots, M$, since no misclassifications occur in such cases; while for $L_{ik} = 1$, there is a penalty in misclassifying $\mathbf{x} \in \omega_k$, but actually $\mathbf{x} \in \omega_i, i = 1, \ldots, M, i \neq k$. This is a symmetric loss function since

$$L_{ik} = 1 - \delta(k - i) \tag{5.4}$$

where $\delta(k - i)$ is the Kronecker delta function and

$$\delta(k - i) = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

If the value of L_{ik} is such that

$$L_{ik} = \begin{cases} -h_i & i = k \\ 0 & i \neq k \end{cases} \tag{5.6}$$

The loss matrix becomes a negative loss function matrix:

$$\mathbf{L}_{\text{neg}} = \begin{bmatrix} -h_1 & & & & \\ & -h_2 & & 0 & \\ & & -h_3 & & \\ & & & \ddots & \\ 0 & & & & \\ & & & & -h_M \end{bmatrix} \tag{5.7}$$

The significance of this negative loss function matrix is that a negative loss (i.e., a positive gain) is assigned to a correct decision and no loss to an erroneous decision. In other words, the loss assigned to a decision is greater for an erroneous decision than for a correct one.

Note that the h_i in the matrix do not have to be equal. They may be different to indicate the relative importance of guessing correctly on one class rather than the other. Similarly, the L_{ik} and L_{ki} in the loss matrix do not have to be equal.

For a two-class problem, $L_{ik} = L_{21}$, where $i = 2, k = 1$. This means that $\mathbf{x}$ should be in ω_2 but is misclassified as being in ω_1. $L_{ik} = L_{12}$ when $i = 1, k = 2$, meaning that $\mathbf{x}$ should be in ω_1 but is misclassified as being in ω_2. $L_{ik} = 0$ when $i = k$. Thus we have

$$\mathbf{L}\begin{bmatrix} 0 & L_{21} \\ L_{12} & 0 \end{bmatrix} \tag{5.8}$$

Suppose that ω_1 is the class of friendly aircraft and ω_2 is the class of enemy aircraft; then undoubtedly

$$L_{21} > L_{12}$$

since L_{12} is just a false alarm, but L_{21} would mean disaster.

However, in another example, such as a fire sprinkling system in a laboratory with expensive equipment, a false alarm should have a large L, because when the sprinkler goes off and there is no fire, a lot of equipment could be ruined. Thus we may end up with $L_{12} = L_{21}$, which is then a symmetric loss function.

5.2.2 Bayes' Discriminant Function

By Bayes' rule, we can write

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{p(\mathbf{x})} \tag{5.9}$$

where $p(\mathbf{x}) = \sum_i p(\mathbf{x}|\omega_i)p(\omega_i)$, $i = 1, 2, \ldots, M$, is the probability that $\mathbf{x}$ occurs without regard to the category in which it belongs. $p(\omega_i)$ is the a priori probability of class ω_i, and $p(\mathbf{x}|\omega_i)$ is the likelihood function of class ω_i with respect to $\mathbf{x}$; it is the probability density function for $\mathbf{x}$ given that the state of nature is ω_i (i.e., it is a pattern belonging to class ω_i).

Substituting Eq. (5.9) into (5.1) for $r_k(\mathbf{x})$, we have

$$r_k(\mathbf{x}) = \frac{1}{p(\mathbf{x})}\sum_{i=1}^{M} L_{ik}p(\mathbf{x}|\omega_i)p(\omega_i) \tag{5.10}$$

Since $p(\mathbf{x})$ in Eq. (5.10) is common to all $r_j(\mathbf{x}), j = 1, \ldots, M$, we can eliminate it from the conditional average risk equation and seek only the following minimum:

$$\min_k r_k(\mathbf{x}) = \min_k \sum_{i=1}^{M} L_{ik}p(\mathbf{x}|\omega_i)p(\omega_i) \tag{5.11}$$

to obtain the best one among all the possible decisions, or alternatively, we can just say that

$$d_k(\mathbf{x}) = -r_k(\mathbf{x}) \tag{5.12}$$

which is the Bayes discriminant function. The classifier basing on this minimization is called Bayes' classifier, which gives the optimum performance from the statistical point of view.

5.2.3 Maximum Likelihood Decision

As defined in Section 5.2.2, $p(\mathbf{x}|\omega_i)$ is called the likelihood function of ω_i. The expression for average or expected loss of deciding $\mathbf{x} \in \omega_k$ is

$$r_k(\mathbf{x}) = \sum_{i=1}^{M} L_{ik} p(\mathbf{x}|\omega_i)p(\omega_i) \tag{5.13}$$

which can then be used for minimization to get the maximum likelihood for $\mathbf{x} \in \omega_k$. For a two-class problem, the average or expected loss of deciding $\mathbf{x} \in \omega_1$ will be

$$r_1(\mathbf{x}) = L_{11}p(\mathbf{x}|\omega_1)p(\omega_1) + L_{21}p(\mathbf{x}|\omega_2)p(\omega_2) \tag{5.14}$$

Similarly, the loss of deciding $\mathbf{x} \in \omega_2$ will be

$$r_2(\mathbf{x}) = L_{12}p(\mathbf{x}|\omega_1)p(\omega_1) + L_{22}p(\mathbf{x}|\omega_2)p(\omega_2) \tag{5.15}$$

In matrix form,

$$\mathbf{r} = \mathbf{Lp} \tag{5.16}$$

or

$$\begin{vmatrix} r_1 \\ r_2 \end{vmatrix} = \begin{vmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{vmatrix} \begin{vmatrix} p(\mathbf{x}|\omega_1)p(\omega_1) \\ p(\mathbf{x}|\omega_2)p(\omega_2) \end{vmatrix} \tag{5.17}$$

The decision that $\mathbf{x} \in \omega_1$ will be made if

$$L_{11}p(\mathbf{x}|\omega_1)p(\omega_1) + L_{21}p(\mathbf{x}|\omega_2)p(\omega_2) < L_{12}p(\mathbf{x}|\omega_1)p(\omega_1) + L_{22}p(\mathbf{x}|\omega_2)p(\omega_2) \tag{5.18}$$

or

$$(L_{21} - L_{22})p(\mathbf{x}|\omega_2)p(\omega_2) < (L_{12} - L_{11})p(\mathbf{x}|\omega_1)p(\omega_1) \tag{5.19}$$

The inequality above can be put in another form; that is, we assign $\mathbf{x} \in \omega_1$ if

$$\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} > \frac{(L_{21} - L_{22})p(\omega_2)}{(L_{12} - L_{11})p(\omega_1)} \tag{5.20}$$

Using the notation $l_{12}(\mathbf{x})$ for $p(\mathbf{x}|\omega_1)/p(\mathbf{x}|\omega_2)$ as the likelihood ratio and θ_{12} for $(L_{21} - L_{22})p(\omega_2)/(L_{12} - L_{11})p(\omega_1)$ as the threshold value, the criterion for the decision becomes

$$\mathbf{x} \in \omega_1 \quad \text{if } l_{12}(\mathbf{x}) > \theta_{12} \tag{5.21}$$

The derivation above can easily be generalized to a multiclass problem (i.e., when $M > 2$). The generalized likelihood ratio and the generalized threshold value will then become, respectively,

$$l_{ki} = \frac{p(\mathbf{x}|\omega_k)}{p(\mathbf{x}|\omega_i)} \tag{5.22}$$

and

$$\theta_{ki} = \frac{(L_{ik} - L_{ii})p(\omega_i)}{(L_{ki} - L_{kk})p(\omega_k)} \tag{5.23}$$

Then the criterion for the decision can be stated:

$$\text{Assign } \mathbf{x} \in \omega_k \qquad \text{if } l_{ki} > \theta_{ki}, \forall i \tag{5.24}$$

This is what we call the *maximum likelihood rule*. Basing on these mathematical relations, it would not be difficult to implement it as a classifier.

Now let us consider the case that L is a symmetric loss function. The problem becomes to assign $\mathbf{x} \in \omega_k$ if $l_{ki} > \theta_{ki} \forall i, i = 1, \ldots, M$. The maximum likelihood rule for this symmetric loss function becomes

$$\frac{p(\mathbf{x})|\omega_k)}{p(\mathbf{x}|\omega_i)} > \frac{p(\omega_i)}{p(\omega_k)} \tag{5.25}$$

since $L_{ik} = 1$ and $L_{ii} = 0 \; \forall i, k$ and $i \neq k$; $i, k = 1, \ldots, M$. If $p(\omega_i) = p(\omega_k) \; \forall i, k$, the maximum likelihood rule becomes:

$$\text{Assign } \mathbf{x} \in \omega_k \qquad \text{if } l_{ki} > 1 \tag{5.26}$$

Note that a different loss function yields a different maximum likelihood rule.

Now let us go back to the more general case that $p(\omega_i) \neq p(\omega_k)$, and let us formulate a discriminant function for the case of the symmetric loss function. Since we have

$$l_{ki} = \frac{p(\mathbf{x}|\omega_k)}{p(\mathbf{x}|\omega_i)} > \frac{p(\omega_i)}{p(\omega_k)} \tag{5.27}$$

then

$$p(\mathbf{x}|\omega_k)p(\omega_k) > p(\mathbf{x}|\omega_i)p(\omega_i) \tag{5.28}$$

In other words, we can assign $\mathbf{x} \in \omega_k$ if

$$p(\mathbf{x}|\omega_k)p(\omega_k) > p(\mathbf{x}|\omega_i)P(\omega_i) \qquad \forall i \tag{5.29}$$

Thus the discriminant function is now

$$d_k(\mathbf{x}) = p(\mathbf{x}|\omega_k)p(\omega_k) \tag{5.30}$$

An alternative form of this discriment function is

$$d_k'(\mathbf{x}) = \log p(\mathbf{x}|\omega_k) + \log p(\omega_k) \tag{5.31}$$

Extending this to a more general case, the average loss of deciding that $\mathbf{x} \in \omega_k$ is

$$r_k(\mathbf{x}) = \sum_{i=1}^{M} L_{ik} p(\mathbf{x}|\omega_i) p(\omega_i) \tag{5.32}$$

or

$$\mathbf{r} = \mathbf{L}^T \mathbf{p} \tag{5.33}$$

The maximum likelihood rule is then

$$\mathbf{x} \in \omega_i \qquad \text{if } r_i(\mathbf{x}) < r_j(\mathbf{x}) \tag{5.34}$$

or

$$\sum_{k=1}^{M} L_{ki} p(\mathbf{x}|\omega_k) p(\omega_k) < \sum_{q=1}^{M} L_{qj} p(\mathbf{x}|\omega_q) p(\omega_q) \qquad \forall j, j \neq i; j = 1, \ldots, M \tag{5.35}$$

The summation of the terms on the left-hand side of (5.35) represent the average loss of deciding $\mathbf{x} \in \omega_i$, while that on the right-hand side represents the loss of deciding that $\mathbf{x} \in \omega_j, j = 1, \ldots, M$ and $j \neq i$.

5.2.4 Binary Example

Let us take, for illustration, a binary example, in which each pattern $\mathbf{x}$ has independent binary components as follows:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \qquad x_i = 1 \text{ or } 0, i = 1, 2, \ldots, n \tag{5.36}$$

For a two-class problem ($M = 2$), the discriminant function $d(\mathbf{x})$ is

$$d(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x}) \tag{5.37}$$

where $d_1(\mathbf{x}) = \log[p(\mathbf{x}|\omega_1)p(\omega_1)]$ and $d_2(\mathbf{x}) = \log[p(\mathbf{x}|\omega_2)p(\omega_2)]$. Then

$$d(\mathbf{x}) = \log\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} + \log\frac{p(\omega_1)}{p(\omega_2)} \tag{5.38}$$

For a two-class problem, $p(\omega_1) + p(\omega_2) = 1$; hence

$$d(\mathbf{x}) = \log\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} + \log\frac{p(\omega_1)}{1 - p(\omega_1)} \tag{5.39}$$

Since the components x_i are independent,

$$p(\mathbf{x}|\omega_j) = p(x_1|\omega_j)p(x_2|\omega_j)\cdots p(x_n|\omega_j) = \prod_{i=1}^{n} p(x_i|\omega_j) \tag{5.40}$$

and

$$d(\mathbf{x}) = \sum_{i=1}^{n} \log\frac{p(x_i|\omega_1)}{p(x_i|\omega_2)} + \log\frac{p(\omega_1)}{1 - p(\omega_1)} \tag{5.41}$$

Since the pattern elements of $\mathbf{x}$ are binary for the problem we discussed here, for simplicity we can let

$$p(x_i = 1|\omega_1) = p_i \tag{5.42}$$

Then

$$p(x_i = 0|\omega_1) = 1 - p_i \tag{5.43}$$

Similarly, let

$$p(x_i = 1|\omega_2) = q_i \tag{5.44}$$

Then

$$p(x_i = 0|\omega_2) = 1 - q_i \tag{5.45}$$

We can then claim that

$$\log\frac{p(x_i|\omega_1)}{P(x_i|\omega_2)} = x_i \log\frac{p_i}{q_i} + (1 - x_i)\log\frac{1 - p_i}{1 - q_i} = \gamma \tag{5.46}$$

The validity of (5.46) can easily be checked by setting either $x_i = 1$ or 0 in this expression. Rewriting expression (5.46) gives

$$\gamma = \log\frac{p(x_i|\omega_1)}{p(x_i|\omega)_2)} = x_i \log\frac{p_i(1 - q_i)}{q_i(1 - p_i)} + \log\frac{1 - p_i}{1 - q_i} \tag{5.47}$$

Substituting back in Eq. (5.41) yields

$$d(\mathbf{x}) = \sum_{i=1}^{n} x_i \log\frac{p_i(1 - q_i)}{q_i(1 - p_i)} + \sum_{i=1}^{n} \log\frac{1 - p_i}{1 - q_i} + \log\frac{p(\omega_1)}{1 - p(\omega_1)} \tag{5.48}$$

where $\log[p_i(1 - q_i)/q_i(1 - p_i)]$ can be represented by w_i and

$$\sum_{i=1}^{n} \log\frac{1 - p_i}{1 - q_i} + \log\frac{p(\omega_1)}{1 - p(\omega_1)}$$

can be represented by w_{n+1}. Then we have

$$d(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + w_{n+1} \tag{5.49}$$

from which we can see that the optimum discriminant function is linear in $\mathbf{x}_i$.

5.2.5 Probability of Error

The probability of error that would be introduced in the scheme discussed in Section 5.2.4 is a problem of much concern. Let us again take the two-class problem for illustration. The classifier will divide the space into two regions, R_1 and R_2. The decision that $\mathbf{x} \in \omega_1$ will be made when the pattern $\mathbf{x}$ falls into the region R_1; and $\mathbf{x} \in \omega_2$, when $\mathbf{x}$ falls into R_2. Under such circumstances, there will be two possible types of errors:

1. $\mathbf{x}$ falls in region R_1, but actually $\mathbf{x} \in \omega_2$. This gives the probability of error E_1, which may be denoted by Prob($\mathbf{x} \in R_1, \omega_2$).
2. $\mathbf{x}$ falls in region R_2, but actually $\mathbf{x} \in \omega_1$. This gives the probability of error E_2, or Prob($\mathbf{x} \in R_2, \omega_1$). Thus the total probability of error is

$$\begin{aligned} \mathrm{P}_{\text{error}} &= \operatorname{Prob}(\mathbf{x} \in R_1|\omega_2)p(\omega_2) + \operatorname{Prob}(x \in R_2|\omega_1)p(\omega_1) \\ &= \int_{R_1} p(\mathbf{x}|\omega_2)p(\omega_2)\, d\mathbf{x} + \int_{R_2} p(\mathbf{x}|\omega_1)p(\omega_1)\, d\mathbf{x} \end{aligned} \tag{5.50}$$

This is the performance criterion that we try to minimize to give a good classification. The two integrands in expression (5.50) are plotted in Figure 5.1.

Areas under the curves shown by the hatched lines represent E_1 and E_2, where $E_1 = \int_{R_1} p(\mathbf{x}|\omega_2)p(\omega_2)d\mathbf{x}$ and $E_2 = \int_{R_2} p(\mathbf{x}|\omega_1)p(\omega_1)d\mathbf{x}$. It is not difficult to see that with an arbitrary decision boundary E_2 represents both the right slash-hatched and the cross-hatched areas. If the decision boundary is moved to the right to the optimum position, which is the vertical line passing through the intersection of the two probability curves, the double-hatched area is eliminated

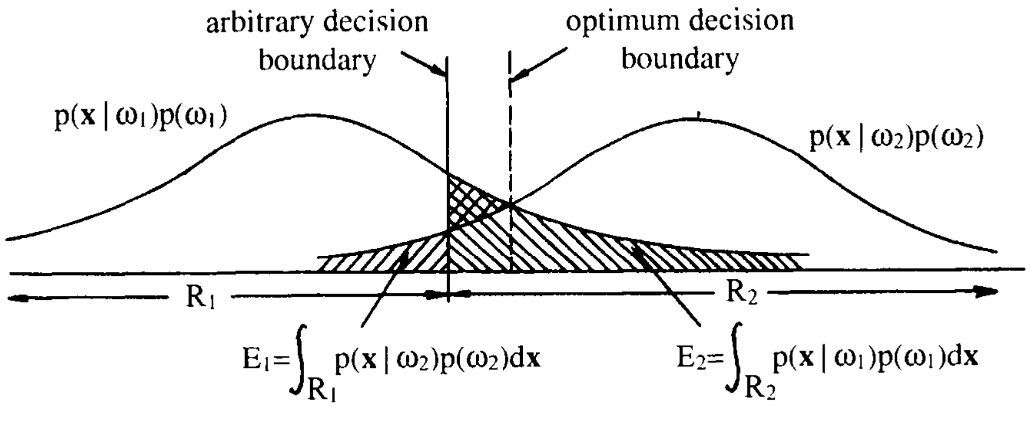

arbitrary decision boundary
optimum decision boundary
p(x | ω₁)p(ω₁)
p(x | ω₂)p(ω₂)
R₁
R₂
E₁=∫_{R₁} p(x | ω₂)p(ω₂)dx
E₂=∫_{R₂} p(x | ω₁)p(ω₁)dx

FIGURE 5.1 Probability of error in a two-class problem.

from the total area and a minimum error would occur. This optimum decision boundary occurs when $\mathbf{x}$ satisfies the following equation:

$$d_1(\mathbf{x}) = d_2(\mathbf{x}) \tag{5.51}$$

or

$$p(\mathbf{x}|\omega_1)p(\omega_1) = p(\mathbf{x}|\omega_2)p(\omega_2) \tag{5.52}$$

and hence the maximum likelihood rule (or Bayes' decision rule with symmetric loss function) is the optimum classifier from a minimum probability of error viewpoint.

To give an analytical expression for the probability of error, let us assume multivariate normal density functions for the pattern vectors with $\mathbf{C}_1 = \mathbf{C}_2 = \mathbf{C}$; thus

$$p(\mathbf{x}|\omega_1) = \frac{1}{(2\pi)^{n/2}|\mathbf{C}|^{1/2}}\exp[-\tfrac{1}{2}(\mathbf{x}-\mathbf{m}_1)^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m}_1)] \tag{5.53}$$

and

$$p(\mathbf{x}|\omega_2) = \frac{1}{(2\pi)^{n/2}|\mathbf{C}|^{1/2}}\exp[-\tfrac{1}{2}(\mathbf{x}-\mathbf{m}_2)^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m}_2)] \tag{5.54}$$

Then, according to Eqs. (5.20) and (5.21),

$$\mathbf{x} \in \omega_1 \qquad \text{if } l_{12} > \theta_{12} \tag{5.55}$$

or

$$\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} > \frac{(L_{21}-L_{22})p(\omega_2)}{(L_{12}-L_{11})p(\omega_1)} \tag{5.56}$$

For the case where the loss functions are symmetric, we have

$$\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} > \frac{p(\omega_2)}{p(\omega_1)} = \theta'_{12} \tag{5.57}$$

Similarly, $\mathbf{x} \in \omega_2$ if $l_{21} > \theta_{21}$, that is,

$$\mathbf{x} \in \omega_2 \quad \text{if } \frac{p(\mathbf{x}|\omega_2)}{p(\mathbf{x}|\omega_1)} > \frac{p(\omega_1)}{p(\omega)_2)} = \theta'_{21} \tag{5.58}$$

Substituting the normal density functions for $p(\mathbf{x}|\omega_1)$ and $p(\mathbf{x}|\omega_2)$, respectively, we obtain

$$\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} = \frac{\exp[-\frac{1}{2}(\mathbf{x}-\mathbf{m}_1)^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m}_1)]}{\exp[-\frac{1}{2}(\mathbf{x}-\mathbf{m}_2)^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m}_2)]} \tag{5.59}$$

Taking the logarithm of the ratio $p(\mathbf{x}|\omega_1)/p(\mathbf{x}|\omega_2)$ and denoting it by p_{12}, then

$$\begin{aligned} p_{12} &= -\tfrac{1}{2}(\mathbf{x}-\mathbf{m}_1)^T\mathbf{C}^{-1}(\mathbf{x}-m_1)+\tfrac{1}{2}(\mathbf{x}-\mathbf{m}_2)^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m}_2) \\ &= \mathbf{x}^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2)-\tfrac{1}{2}(\mathbf{m}_1+\mathbf{m}_2)^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2) \end{aligned} \tag{5.60}$$

Then

$$\mathrm{Prob}(\mathbf{x}\in R_1,\omega_2)=p[p_{12}>\log\theta'_{12}|\omega_2] \tag{5.61}$$

and

$$\mathrm{Prob}(\mathbf{x}\in R_2,\omega_1)=p[p_{12}<\log\theta'_{12}|\omega_1] \tag{5.62}$$

The expected value of p_{12} for class 1 can then be found as

$$\begin{aligned} E_1[p_{12}] &= \mathbf{m}_1^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2)-\tfrac{1}{2}(\mathbf{m}_1+\mathbf{m}_2)^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2) \\ &= \tfrac{1}{2}[(\mathbf{m}_1-\mathbf{m}_2)^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2)] \end{aligned} \tag{5.63}$$

The variance of p_{12} for class 1 is defined by

$$\mathrm{var}_1[p_{12}]=E_1[(p_{12}-\overline{p_{12}})^2] \tag{5.64}$$

and equals

$$\begin{aligned} \mathrm{var}_1[p_{12}] = E_1[&\mathbf{x}^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2)-\tfrac{1}{2}(\mathbf{m}_1+\mathbf{m}_2)^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2) \\ &-\mathbf{m}_1^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2)+\tfrac{1}{2}(\mathbf{m}_1+\mathbf{m}_2)^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2)]^2 \\ &= E_1[(\mathbf{x}-\mathbf{m}_1)^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2)]^2 \\ &= E_1[(\mathbf{m}_1-\mathbf{m}_2)^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m}_1)(\mathbf{x}-\mathbf{m}_1)^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2)] \\ &= (\mathbf{m}_1-\mathbf{m}_2)^T\mathbf{C}^{-1}E_1[(\mathbf{x}-\mathbf{m}_1)(\mathbf{x}-\mathbf{m}_1)^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2)] \\ &= (\mathbf{m}_1-\mathbf{m}_2)^T\mathbf{C}^{-1}\mathbf{C}\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2) \end{aligned} \tag{5.65}$$

since $E_1[(\mathbf{x}-\mathbf{m}_1)(\mathbf{x}-\mathbf{m}_1)^T]=\mathbf{C}$ by definition. Therefore,

$$\begin{aligned} \mathrm{var}_1[p_{12}] &= (\mathbf{m}_1-\mathbf{m}_2)^T\mathbf{C}^{-1}(\mathbf{m}_1-\mathbf{m}_2) \\ &= r_{12} \end{aligned} \tag{5.66}$$

Substituting back in Eq. (5.63), we obtain

$$E_1[p_{12}]=\tfrac{1}{2}r_{12} \tag{5.67}$$

where r_{12} equals the Mahalanobis distance between $p(\mathbf{x}|\omega_1)$ and $p(\mathbf{x}|\omega_2)$. Thus, for $\mathbf{x}\in\omega_1$, the ratio $p(\mathbf{x}|\omega_1)/p(\mathbf{x}|\omega_2)$ is distributed with a mean equal to $\frac{1}{2}r_{12}$ and a variance equal to r_{12}; while for $\mathbf{x}\in\omega_2$, that ratio will be distributed with a mean equal to $-\frac{1}{2}r_{12}$ and a variance equal to r_{12}. Therefore, the probability of misclassifying a pattern when $\mathbf{x} \in \omega_2$ is

$$p(p_{12} > \log \theta'_{12} | \omega_2) = \int_{\log \theta'_{12}}^{\infty} \frac{1}{\sqrt{2\pi r_{12}}} \cdot \exp\left[-\frac{(p_{12} + \frac{1}{2}r_{12})^2}{2r_{12}} \right] dp_{12} \tag{5.68}$$

and the probability of misclassifying a pattern when it comes from ω_1 is

$$p(p_{12} < \log \theta'_{12} | \omega_1) = \int_{-\infty}^{\log \theta'_{12}} \frac{1}{\sqrt{2\pi r_{12}}} \cdot \exp\left[-\frac{(p_{12} - \frac{1}{2}r_{12})^2}{2r_{12}} \right] dp_{12} \tag{5.69}$$

The total probability of error P_{error} is then

$$P_{\text{error}} = E_1 + E_2 = p(p_{12} > \log \theta'_{12} | \omega_2) p(\omega_2) + p(p_{12} < \log \theta'_{12} | \omega_1) p(\omega_1) \tag{5.70}$$

This analysis can easily be extended to a multiclass case. In the multiclass cases, there are more ways to be wrong than to be right. So it is simpler to compute the probability of being correct.

Let us denote the probability of being correct as

$$P_{\text{correct}} = \sum_{i=1}^{M} \text{Prob}(\mathbf{x} \in R_i, \omega_i) = \sum_{i=1}^{M} \int_{R_i} p(\mathbf{x}|\omega_i) p(\omega_i)\, d\mathbf{x} \tag{5.71}$$

where $\text{Prob}(\mathbf{x} \in R_i, \omega_i)$ denotes the probability that $\mathbf{x}$ falls in R_i, while the true state of nature is also that $\mathbf{x} \in \omega_i$. Summation of $\text{Prob}(\mathbf{x} \in R_i, \omega_i)$, $i = 1, 2, \ldots, M$, gives the total classification probability of being correct. The Bayes classifier with symmetric loss function maximizes P_{correct} by choosing the regions R_i so that the integrands are maximum. Analysis of the multivariate normal density function for the pattern vectors can be worked out similarly without too much difficulty.

5.3 OPTIMAL DISCRIMINANT FUNCTIONS FOR NORMALLY DISTRIBUTED PATTERNS

5.3.1 Normal Distribution

The multivariate normal density function for M pattern classes can be represented by

$$\begin{aligned} p(\mathbf{x}|\omega_k) &= \frac{1}{(2\pi)^{n/2} |\mathbf{C}_k|^{1/2}} \exp\left[-\tfrac{1}{2} (\mathbf{x} - \mathbf{m}_k)^T \mathbf{C}_k^{-1} (\mathbf{x} - \mathbf{m}_k) \right] \\ &= \mathcal{N}(\mathbf{m}_k, \mathbf{C}_k) \qquad k = 1, 2, \ldots, M; \\ n &= \text{dimensionality of the pattern vector} \end{aligned} \tag{5.72}$$

where $\mathcal{N}$ is the normal density function, $\mathbf{m}_k$ is the mean vector, and $\mathbf{C}_k$ is the covariance matrix for class k, defined respectively by their expected values over the patterns belonging to the class k. Thus

$$\mathbf{m}_k = E_k[\mathbf{x}] \tag{5.73}$$

and

$$\mathbf{C}_k = E_k[(\mathbf{x} - \mathbf{m}_k)(\mathbf{x} - \mathbf{m}_k)^T] \tag{5.74}$$

Pattern samples drawn from a normal population in the pattern space form a single cluster, the center of which is determined by the mean vector obtained from the samples and the shape of the cluster is determined by the covariance matrix. Figure 5.2 shows three different clusters with different shapes. For the cluster in part (a), $\mathbf{m} = (0 \quad 0)^T$ and $\mathbf{C} = I$ (an identity matrix). Because of its symmetry, $C_{ji} = C_{ij} = 0$. $C_{ii} = 1$. For the cluster in part (b),

$$\mathbf{m} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} \mathbf{C}_{11} & 0 \\ 0 & C_{22} \end{bmatrix}$$

$C_{22} > C_{11}$; while for the cluster in (c) (still in the same figure),

$$\mathbf{m} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

The principal axes of the hyperellipsoids (contours of equal probability density) are given by the eigenvectors of $\mathbf{C}$ with the eigenvalues determining the relative lengths of these axes.

A useful measure for similarity, known as *Mahalanobis distance* (r) from pattern $\mathbf{x}$ to mean $\mathbf{m}$, can be defined as

$$r = (\mathbf{x} - \mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x} - \mathbf{m}) \tag{5.75}$$

The Mahalanobis distance between two classes can similarly be expressed as

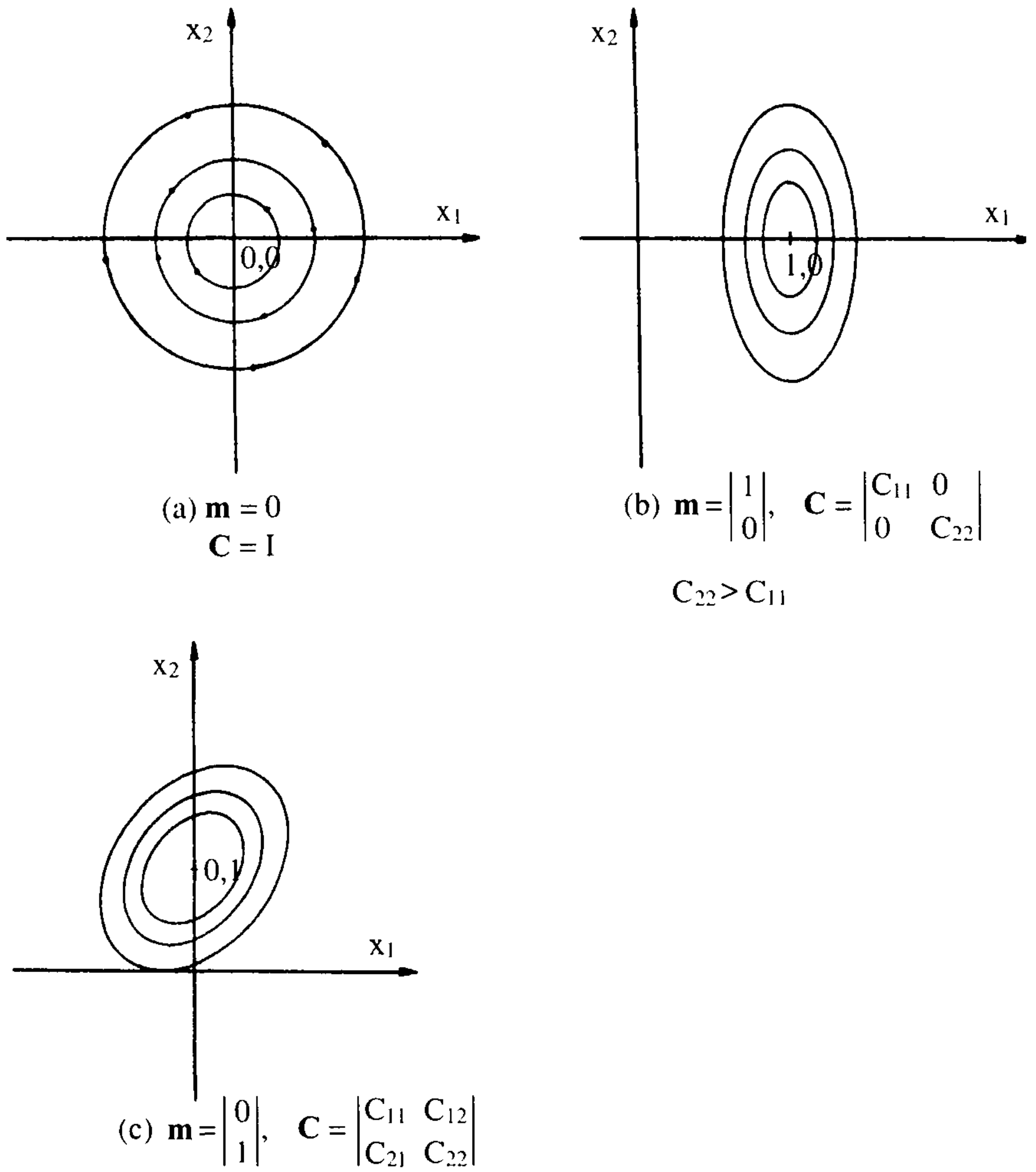
$$r_{ij} = (\mathbf{m}_i - \mathbf{m}_j)^T\mathbf{C}^{-1}(\mathbf{m}_i - \mathbf{m}_j) \tag{5.76}$$

Recall that for $n = 1$, approximately 95% of the samples $\mathbf{x}$ fall in the region $|\mathbf{x} - \mathbf{m}| < 2\sigma$, where σ is the standard deviation and is equal to $\mathbf{C}^{1/2}$.

5.3.2 Optimal Discriminant Functions

From Eq. (5.31), the discriminant function for $\mathbf{x} \in \omega_k$ can be put in the following form:

$$d'_k(\mathbf{x}) = \log p(\mathbf{x}|\omega_k) + \log p(\omega_k) \tag{5.77}$$



(a) $\mathbf{m} = 0$
$\mathbf{C} = \mathbf{I}$

(b) $\mathbf{m} = \begin{vmatrix} 1 \\ 0 \end{vmatrix}$, $\mathbf{C} = \begin{vmatrix} C_{11} & 0 \\ 0 & C_{22} \end{vmatrix}$

$C_{22} > C_{11}$

(c) $\mathbf{m} = \begin{vmatrix} 0 \\ 1 \end{vmatrix}$, $\mathbf{C} = \begin{vmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{vmatrix}$

FIGURE 5.2 Three clusters with different shapes.

When this discriminant function is applied to the multivariate normal density for an M-pattern class with

$$p(\mathbf{x}|\omega_k) = \frac{1}{(2\pi)^{n/2}|\mathbf{C}_k|^{1/2}} \exp\left[-\tfrac{1}{2}(\mathbf{x} - \mathbf{m}_k)^T \mathbf{C}_k^{-1}(\mathbf{x} - \mathbf{m}_k)\right]$$

$$k = 1, 2, \ldots, M \qquad (5.78)$$

the discriminant function $d'_k(\mathbf{x})$ becomes

$$d''_k(\mathbf{x}) = -\frac{n}{2}\log(2\pi) - \tfrac{1}{2}\log|\mathbf{C}_k| - \tfrac{1}{2}(\mathbf{x}-\mathbf{m}_k)^T\mathbf{C}_k^{-1}(\mathbf{x}-\mathbf{m}_k) + \log p(\omega_k) \tag{5.79}$$

It is clear that if the first term on the right-hand side is the same for all k, it can be eliminated. Then the discriminant function reduces to

$$d'''_k(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x}-\mathbf{m}_k)^T\mathbf{C}_k^{-1}(\mathbf{x}-\mathbf{m}_k) + [\log p(\omega_k) - \tfrac{1}{2}\log|\mathbf{C}_k|] \tag{5.80}$$

This is a quadratic discriminant function, and can be put in more compact form as

$$d_k^{(\text{iv})}(\mathbf{x}) = -\tfrac{1}{2}r + f(k) \quad \text{for } \mathbf{x} \in \omega_k \tag{5.81}$$

where $r = (\mathbf{x}-\mathbf{m}_k)^T\mathbf{C}_k^{-1}(\mathbf{x}-\mathbf{m}_k)$ is the Mahalanobis distance defined by Eq. (5.75) and $f(k) = \log p(\omega_k) - \frac{1}{2}\log|\mathbf{C}_k|$. Let us discuss this discriminant function in more detail for two different cases.

Case 1. When the covariance matrices are equal for different classes ($\mathbf{C}_i = \mathbf{C}_j = \mathbf{C}_k = \mathbf{C}$). The physical significance of this is that the separate classes (or clusters in our special terminology) are of equal size and of similar shape, but the clusters are centered about different means. Expanding the general equation for $d_k(\mathbf{x})$ [Eq. (5.80)], we get

$$d_k(\mathbf{x}) = -\tfrac{1}{2}\mathbf{x}^T\mathbf{C}^{-1}\mathbf{x} - \tfrac{1}{2}\mathbf{m}_k^T\mathbf{C}^{-1}\mathbf{m}_k + \mathbf{x}^T\mathbf{C}^{-1}\mathbf{m}_k + \log p(\omega_k) - \tfrac{1}{2}\log|\mathbf{C}| \tag{5.82}$$

The first and last terms on the right-hand side of Eq. (5.82) are the same for all classes (i.e., for all k). Then this discriminant function can be put in an even more compact form as follows:

$$d_k(\mathbf{x}) = \mathbf{x}^T\mathbf{C}^{-1}\mathbf{m}_k + [\log p(\omega_k) - \tfrac{1}{2}\mathbf{m}_k^T\mathbf{C}^{-1}\mathbf{m}_k] \qquad k = 1, 2, \ldots, M \tag{5.83}$$

Obviously, this is a linear discriminant function if we treat $\mathbf{C}^{-1}\mathbf{m}_k$ as $\mathbf{w}_k$ and treat the two terms inside the brackets as an augmented term, $w_{k,n+1}$. For a two-class problem ($M = 2$),

$$\begin{aligned} d(\mathbf{x}) &= d_1(\mathbf{x}) - d_2(\mathbf{x}) \\ &= \mathbf{x}^T\mathbf{C}^{-1}(\mathbf{m}_1 - \mathbf{m}_2) + \log\frac{p(\omega_1)}{p(\omega_2)} - \tfrac{1}{2}(\mathbf{m}_1^T\mathbf{C}^{-1}\mathbf{m}_1 - \mathbf{m}_2^T\mathbf{C}^{-1}\mathbf{m}_2) \end{aligned} \tag{5.84}$$

or

$$d(\mathbf{x}) = \mathbf{x}^T\mathbf{C}^{-1}(\mathbf{m}_1 - \mathbf{m}_2) + \log\frac{p(\omega_1)}{p(\omega_2)} - \tfrac{1}{2}[(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{C}^{-1}\mathbf{m}_1 + \mathbf{m}_2)] \tag{5.85}$$

Case 2. When the covariance matrix $\mathbf{C}_k$ is of diagonal form $\sigma_k^2 I$, where $\sigma_k^2 = |\mathbf{C}_k|$. The physical significance of this is that the cluster has equal components along all the principal axes, and the distribution is of spherical shape. Then substitution of $\sigma_k^2 I$ for $\mathbf{C}_k$ in Eq. (5.80) gives

$$d_k(\mathbf{x}) = -\frac{1}{2}\frac{(\mathbf{x} - \mathbf{m}_k)^T(\mathbf{x} - \mathbf{m}_k)}{\sigma_k^2} + \left[\log p(\omega_k) - \tfrac{1}{2}\log\sigma_k^2\right] \tag{5.86}$$

because $\mathbf{C}_k^{-1} = (1/\sigma_k^2)I$. When the features are statistically independent, and when each feature has the same variance, σ^2, then $\sigma_k = \sigma_j = \sigma, \forall j, k$, that is,

$$\mathbf{C}_k = \mathbf{C}_j = \sigma^2 I \tag{5.87}$$

and

$$d_k(\mathbf{x}) = -\frac{1}{2}\frac{\mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{m}_k + \mathbf{m}_k^T\mathbf{m}_k}{\sigma^2} + \log p(\omega_k) - \tfrac{1}{2}\log\sigma^2 \tag{5.88}$$

Again, $\mathbf{x}^T\mathbf{x}$ and $\frac{1}{2}\log\sigma^2$ are the same for all k. We can neglect these two terms in $d_k(\mathbf{x})$ and get a new expression:

$$d_k(\mathbf{x}) = \frac{1}{\sigma^2}\mathbf{x}^T\mathbf{m}_k + \left[\log p(\omega_k) - \frac{1}{2\sigma^2}\mathbf{m}_k^T\mathbf{m}_k\right] \tag{5.89}$$

which can then also be treated as a linear discriminant function.

If in addition to the assumption that $\mathbf{C}_k = \mathbf{C}_j = \sigma^2 I$, the assumption is made to let $p(\omega_k) = 1/K\ \forall k$, where K is a constant, the term "$\log p(\omega_k)$" can also be dropped from the expression for $d_k(\mathbf{x})$. Then $d_k(\mathbf{x})$ will be further simplified as

$$d_k(\mathbf{x}) = \mathbf{x}^T\mathbf{m}_k - \tfrac{1}{2}|\mathbf{m}_k|^2 \tag{5.90}$$

which is obviously a linear equation.

From the analyses we have done so far, the quadratic discriminant function as obtained for the multivariate normal density for M-pattern classes can be simplified into a form that can be implemented by a linear machine, thus making the problem much simpler.

Equation (5.86) can be simplified into another form, with which we are familiar. Since it is assumed that $\mathbf{C}_k = \mathbf{C}_j = \mathbf{C} = \sigma^2 I$ and $p(\omega_k) =$

$p(\omega_j) = \cdots = 1/K$ = constant, after dropping the unnecessary terms, Eq. (5.86) becomes

$$d_k(\mathbf{x}) = -\frac{1}{2}\frac{(\mathbf{x}-\mathbf{m}_k)^T(\mathbf{x}-\mathbf{m}_k)}{\sigma^2} \tag{5.91}$$

or simply

$$d_k(\mathbf{x}) = -(\mathbf{x}-\mathbf{m}_k)^T(\mathbf{x}-\mathbf{m}_k) = -|\mathbf{x}-\mathbf{m}_k|^2 \tag{5.92}$$

which is the same as the *minimum distance classifier.*

To conclude this section, we would like to add that the multivariate normal density function mentioned in Sec. 5.3.1 is only one of the probability density functions available to represent the distribution of random variables. If K_n, $|\mathbf{W}|^{1/2}$, and $f[(\mathbf{x}-\mathbf{m})^T\mathbf{W}(\mathbf{x}-\mathbf{m})]$ replace $(2\pi)^{-n/2}$, $|\mathbf{C}|^{-1/2}$, and $\exp[-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})]$, respectively, the multivariate normal density function

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\mathbf{C}|^{1/2}}\exp[-\tfrac{1}{2}(\mathbf{x}-\mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})] \tag{5.93}$$

can be generalized as

$$p(\mathbf{x}) = K_n|\mathbf{W}|^{1/2}f[(\mathbf{x}-\mathbf{m})^T\mathbf{W}(\mathbf{x}-\mathbf{m})] \tag{5.94}$$

with K_n as the normalizing constant and $\mathbf{W}$ as the weight matrix. When different values and functions are given to K_n, $\mathbf{W}$, and f, different types of density function will be obtained. Examples of these are Pearson type II and type VII functions.

A very simple example is used to illustrate the computation of the mean, covariance, and the discriminant function by the statistical decision method. A practical example using computer computation with a large data set is given at the end of the chapter.

Example. Given pattern points $(1, 2)^T, (2, 2)^T, (3, 1)^T, (3, 2)^T$, and $(2, 3)^T$, are known to be in class ω_1. Another set of points, $(7, 9)^T$, $(8, 9)^T, (9, 8)^T, (9, 9)^T$, and $(8, 10)^T$, are known to be in class ω_2. It is required to find a Bayes decision boundary to separate them.

Solution:

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{j=1}^{N_1} x_{1j} = \frac{1}{5}\begin{bmatrix} 11 \\ 10 \end{bmatrix}$$

$$\mathbf{m}_2 = \frac{1}{N_2}\sum_{j=1}^{N_2} x_{2j} = \frac{1}{5}\begin{bmatrix} 41 \\ 45 \end{bmatrix}$$

By definition,

$$\begin{aligned}\mathbf{C} &= E[(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T] \\ &= E[\mathbf{x}\mathbf{x}^T] - \mathbf{m}\mathbf{m}^T\end{aligned}$$

When it is put in discrete form,

$$\mathbf{C}_1 = \frac{1}{N_1}\sum_{j=1}^{N_1} x_{1j}x_{1j}^T - \mathbf{m}_1\mathbf{m}_1^T$$

Therefore,

$$\begin{aligned}\mathbf{C}_1 &= \frac{1}{5}\left[\begin{pmatrix}1\\2\end{pmatrix}(1 \quad 2) + \begin{pmatrix}2\\2\end{pmatrix}(2 \quad 2) + \begin{pmatrix}3\\1\end{pmatrix}(3 \quad 1)\right. \\ &\quad \left. + \begin{pmatrix}3\\2\end{pmatrix}(3 \quad 2) + \begin{pmatrix}2\\3\end{pmatrix}(2 \quad 3)\right] - \frac{1}{25}\begin{pmatrix}11\\10\end{pmatrix}(11 \quad 10) \\ &= \frac{1}{25}\begin{pmatrix}14 & -5\\ -5 & 10\end{pmatrix}\end{aligned}$$

Similarly,

$$\begin{aligned}\mathbf{C}_2 &= \frac{1}{N_2}\sum_{j=1}^{N_2} x_{2j}x_{2j}^T - \mathbf{m}_2\mathbf{m}_2^T \\ &= \frac{1}{25}\begin{pmatrix}14 & -5\\ -5 & 10\end{pmatrix}\end{aligned}$$

We have

$$\mathbf{C}_1 = \mathbf{C}_2 = \mathbf{C} = \frac{1}{25}\begin{pmatrix}14 & -5\\ -5 & 10\end{pmatrix}$$

The determinant and adjoint of $\mathbf{C}$ can be computed as

$$|\mathbf{C}| = \frac{1}{25}\begin{vmatrix}14 & -5\\ -5 & 10\end{vmatrix} = \frac{23}{5} \qquad \text{adj}\,\mathbf{C} = \begin{pmatrix}\frac{10}{25} & \frac{5}{25}\\ \frac{5}{25} & \frac{14}{25}\end{pmatrix}$$

The inverse of $\mathbf{C}$, $\mathbf{C}^{-1}\mathbf{m}_1$, and $\mathbf{m}_1^T\mathbf{C}^{-1}\mathbf{m}_1$ are then, respectively, as follows:

$$\mathbf{C}^{-1} = \frac{1}{|\mathbf{C}|}\text{adj}\,\mathbf{C} = \frac{1}{115}\begin{pmatrix}10 & 5\\ 5 & 14\end{pmatrix}$$

$$\mathbf{C}^{-1}m_1 = \frac{1}{115}\begin{pmatrix}32\\39\end{pmatrix}$$

$$\mathbf{m}_1^T\mathbf{C}^{-1}\mathbf{m}_1 = \frac{742}{5 \times 115}$$

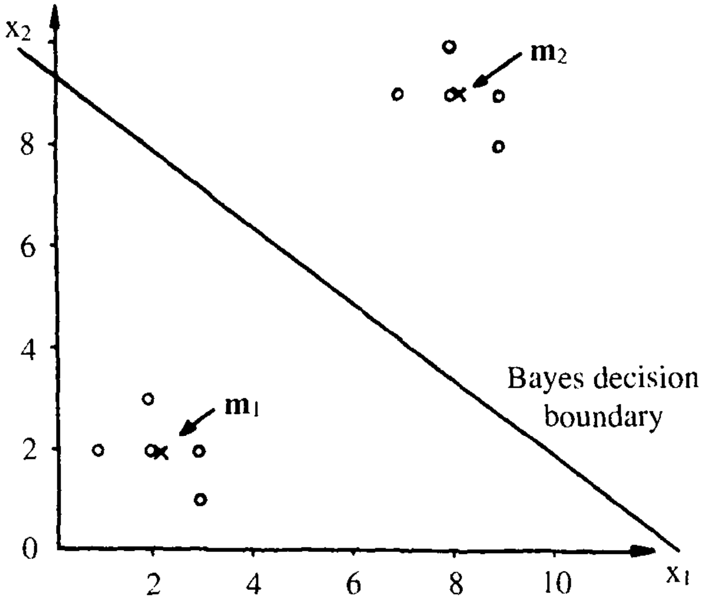


FIGURE 5.3 Illustrative example.

The discriminant function for class 1 is

$$d_1(\mathbf{x}) = \mathbf{x}^T\mathbf{C}^{-1}\mathbf{m}_1 - \tfrac{1}{2}\mathbf{m}_1^T\mathbf{C}^{-1}\mathbf{m}_1$$
$$= \frac{32}{115}x_1 + \frac{39}{115}x_2 - 0.65$$

Similarly, we obtain

$$\mathbf{C}^{-1}\mathbf{m}_2 = \frac{1}{115}\begin{pmatrix}127\\167\end{pmatrix}$$
$$\mathbf{m}_2^T\mathbf{C}^{-1}\mathbf{m}_2 = 22$$

The discriminant function for class 2 is

$$d_2(\mathbf{x}) = \mathbf{x}^T\mathbf{C}^{-1}\mathbf{m}_2 - \tfrac{1}{2}\mathbf{m}_2^T\mathbf{C}^{-1}\mathbf{m}_2$$
$$= \tfrac{127}{115}x_1 + \tfrac{167}{115}x_2 - 11$$

The decision surface is then given by

$$d(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x}) = 0$$

or

$$d(\mathbf{x}) = -0.826x_1 - 1.11x_2 + 10.35 = 0$$

which is shown in Figure 5.3.

5.4 TRAINING FOR STATISTICAL DISCRIMINANT FUNCTIONS

So far the formulation of the statistical classification problem and the optimum discriminant function for a normally distributed pattern have been discussed. The next problem that might interest us will be how to determine the unknown probability density function. One of the ways of doing this is by functional approximation. Assume that we wish to approximate $p(\mathbf{x}|\omega_i)$ by a set of functions

$$\hat{p}(\mathbf{x}|\omega_i) = \sum_{k=1}^{K} c_{ik}\phi_k(\mathbf{x}) \tag{5.95}$$

where the caret sign over $p(\cdot)$ represents the estimated value. The $\phi_k(\mathbf{x})$ are arbitrary functions and can be a set of some basic functions, which may be Hermite polynomials or others. The problem that we have now becomes to seek the coefficients c_{ik} so that the mean squared error

$$Q = \int_{\mathbf{x}} [p(\mathbf{x}|\omega_i) - \hat{p}(\mathbf{x}|\omega_i)]^2 \, d\mathbf{x} \tag{5.96}$$

over all $\mathbf{x}$ for class ω_i can be minimized. After substitution of Eq. (5.95) in Eq. (5.96), we have

$$Q = \int_{\mathbf{x}} [p(\mathbf{x}|\omega_i) - \sum_{k=1}^{K} c_{ik}\phi_k(\mathbf{x})]^2 \, d\mathbf{x} \tag{5.97}$$

A necessary condition for minimum Q is

$$\frac{\partial Q}{\partial c_{ik}} = 0 \qquad k = 1, \ldots, K \tag{5.98}$$

or

$$\frac{\partial Q}{\partial c_{ik}} = 2\int_{\mathbf{x}} \left[p(\mathbf{x}|\omega_i) - \sum_{k=1}^{K} c_{ik}\phi_k(x) \right] \phi_k(\mathbf{x}) \, d\mathbf{x} = 0 \tag{5.99}$$

from which we get

$$\int_{\mathbf{x}} \phi_k(\mathbf{x}) \left[\sum_{k=1}^{K} c_{ik}\phi_k(\mathbf{x}) \right] d\mathbf{x} = \int_{\mathbf{x}} \phi_k(\mathbf{x}) p(\mathbf{x}|\omega_i) \, d\mathbf{x} \tag{5.100}$$

Since, by definition, $\int_{\mathbf{x}} \phi_k(\mathbf{x})p(\mathbf{x}|\omega_i) \, d\mathbf{x}$ is the expected value $E_i[\phi_k(\mathbf{x})]$, then

$$\sum_{k=1}^{K} c_{ik} \int_{\mathbf{x}} \phi_k(\mathbf{x})\phi_k(\mathbf{x})d\mathbf{x} = E_i[\phi_k(\mathbf{x})] \qquad k = 1, \ldots, K \tag{5.101}$$

A set of K linear equations in $c_{ik}(k = 1, \ldots K)$ for a certain i can be obtained to solve for c_{ik}, but knowledge of $p(\mathbf{x}|\omega_i)$ is required.

Knowing that $E_i[\phi_k(\mathbf{x})] = \int_x \phi_k(\mathbf{x})p(\mathbf{x}|\omega_i)\, d\mathbf{x}$ can be approximated by

$$E_i[\phi_k(\mathbf{x})] \simeq \frac{1}{\mathrm{N}_i}\sum_{j=1}^{N_i} \phi_k(\mathbf{x}_j) \tag{5.102}$$

where N_i is the number of pattern samples in class i, then

$$\sum_{k=1}^{K} c_{ik} \int_{\mathbf{x}} \phi_k(\mathbf{x})\phi_k(\mathbf{x})\, d\mathbf{x} = \frac{1}{N_i}\sum_{j=1}^{N_i} \phi_k(\mathbf{x}_j) \qquad k = 1, \ldots, K \tag{5.103}$$

This is a set of K linear equations and can be solved for the K $\phi_k(\mathbf{x})$'s. If, in particular, orthonormal functions are chosen for the $\phi_k(\mathbf{x})$'s, that is,

$$\int_{\mathbf{x}} \phi_i(\mathbf{x})\phi_j(\mathbf{x})\, d\mathbf{x} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{5.104}$$

then

$$c_{ik} = \frac{1}{N_i}\sum_{j=1}^{N_i} \phi_k(\mathbf{x}_j) \qquad \begin{array}{l} k = 1, \ldots, K \\ i = 1, \ldots, M \end{array} \tag{5.105}$$

Once the coefficients c_{ik} have been determined, the density function $\hat{p}(\mathbf{x}|\omega_i)$ is formed. Note that the $\mathbf{x}_j$ do not have to be stored but can be presented in sequential order. The c_{ik} can then be obtained iteratively from the following relation:

$$c_{ik}(N_i + 1) = \frac{1}{N_i + 1}[N_i c_{ik}(N_i) + \phi_k(\mathbf{x}_{N_i+1})] \tag{5.106}$$

where $c_{ik}(N_i)$ and $c_{ik}(N_i + 1)$ represent, respectively, the coefficients obtained with N_i and $N_i + 1$ pattern samples. For more detailed discussion on this topic, see Tou and Gonzales (1974).

5.5 APPLICATION TO A LARGE DATA-SET PROBLEM: A PRACTICAL EXAMPLE

Problems with large data sets are very common in our daily life. Many such problems can be found in agriculture, industry, and commerce as well as defense. An example consisting of three basic color bands (red, green, and blue), each with 254 × 606 digitizing picture elements from an aerial photograph of a water treatment plant area, is used for illustration. See Figure 5.6 or the computer-

TABLE 5.1 Means and Standard Deviations for Red, Green, and Blue Spectral Data Sets of the Image

	Mean	Standard deviation
Red	33.80	17.84
Green	38.11	26.51
Blue	25.12	10.17

processed image. Every pixel (picture element) in the image corresponds to a pattern in the pattern space:

$$\mathbf{x}_{ij} = \begin{vmatrix} x_{ij1} \\ x_{ij2} \\ x_{ij3} \end{vmatrix}$$

The norm of the vector representing this pattern point is $\sqrt{\sum_{k=1}^{3} x_{ijk}^2}$, or

$$|\mathbf{x}_{ij}| = (\mathbf{x}_{ij}^T \mathbf{x}_{ij})^{1/2} \qquad i = 1, 2, \dots, 254; j = 1, 2, \dots, 606$$

$\mathbf{x}_{ij}$ are computed and normalized to 256 gray levels for each of the three channels above.

The aerial image is mainly the photoreflection of the ground objects. From that image a fairly good idea can be obtained about what is in the image if a set of spectral responses, one for each spectral band, is chosen as the basis for analysis (see Table 5.1).

A histogram of the data set gives us a rough idea of the gray-level distribution among the pixels. From the information given in the histogram, we can then set the appropriate gray-level limits to separate the pattern points into categories. A small set of known data (or *ground truth*, as we usually call them) can be used to train the system, so that the pattern points in the whole image (254 × 606 pattern points altogether in our data set) can be classified. Figure 5.4a, b, and c show, respectively, the histograms for the red, green, and blue bands of the aerial photograph.

A portion of the symbol-coding map reproduced by ordinary line printer is shown in Figure 5.5, in which x's represent object points with gray levels below 20.49; +'s represent those below 26.94 but above 20.49; −'s represent those below 35.48 but above 26.94; and blanks represent those above 35.48. All values are converted to a percentage of the whole gray-level range. A digitized image of the aerial photograph is shown in Figure 5.6 to check the effectiveness of the use of the line printer. This image was plotted with the norm values of the pixels.

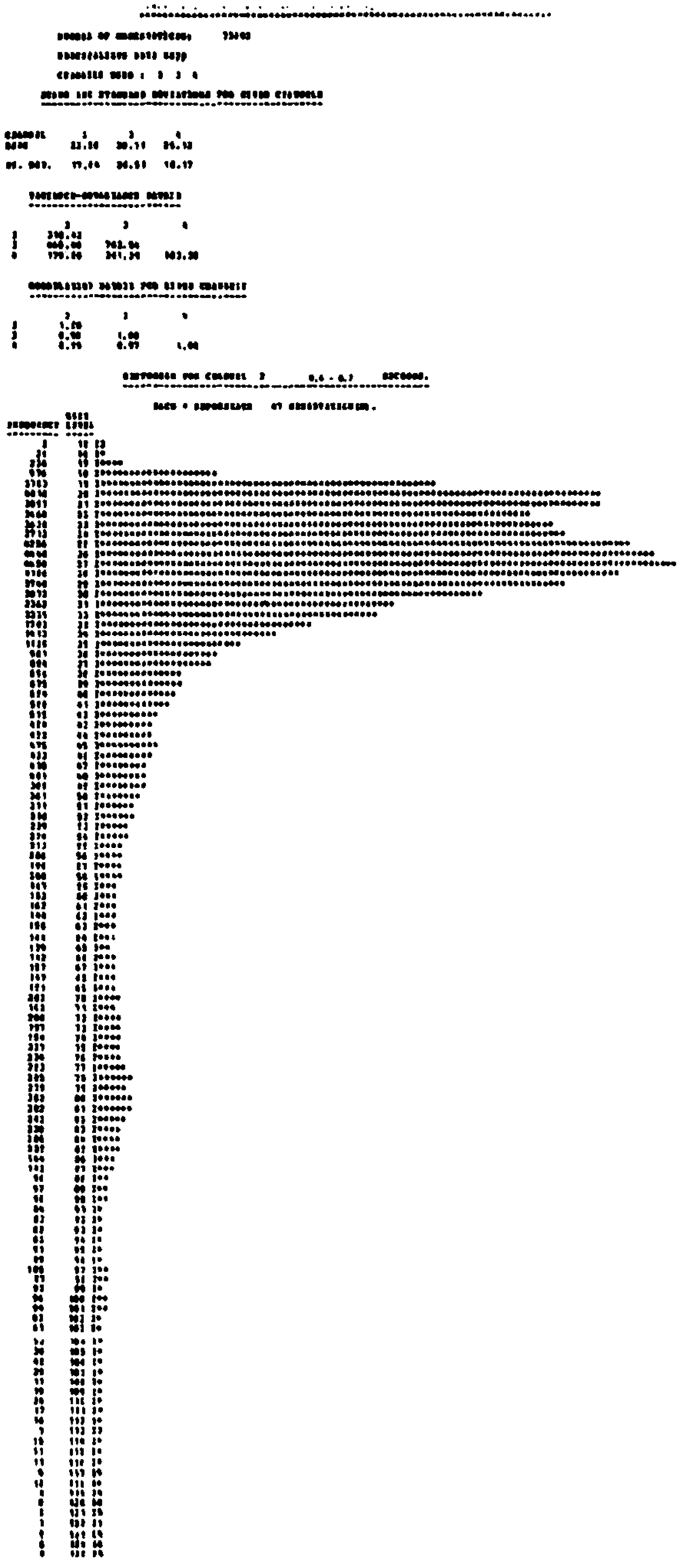


FIGURE 5.4a Histogram of the data set in the spectral range 0.6–0.7 μm (red).

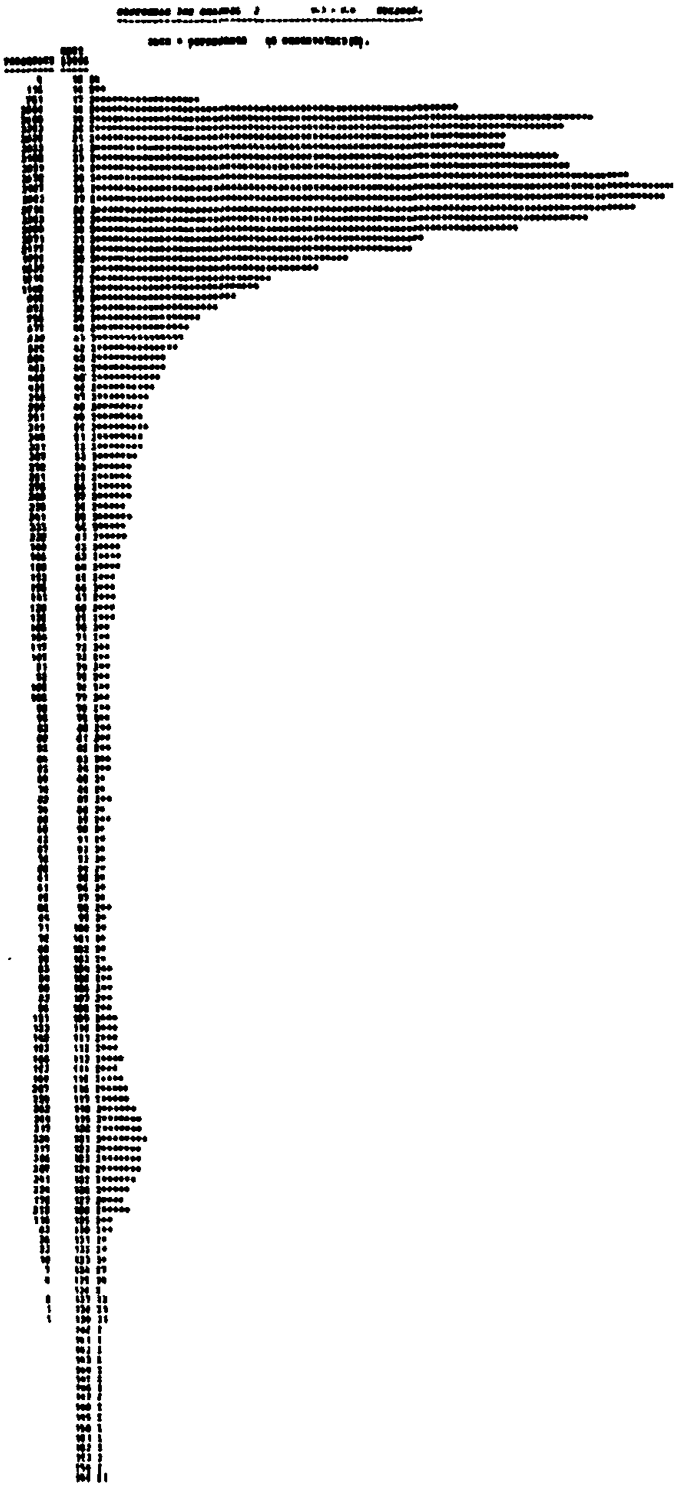


FIGURE 5.4b Histogram of the data set in the spectral range 0.5–0.6 μm (green).

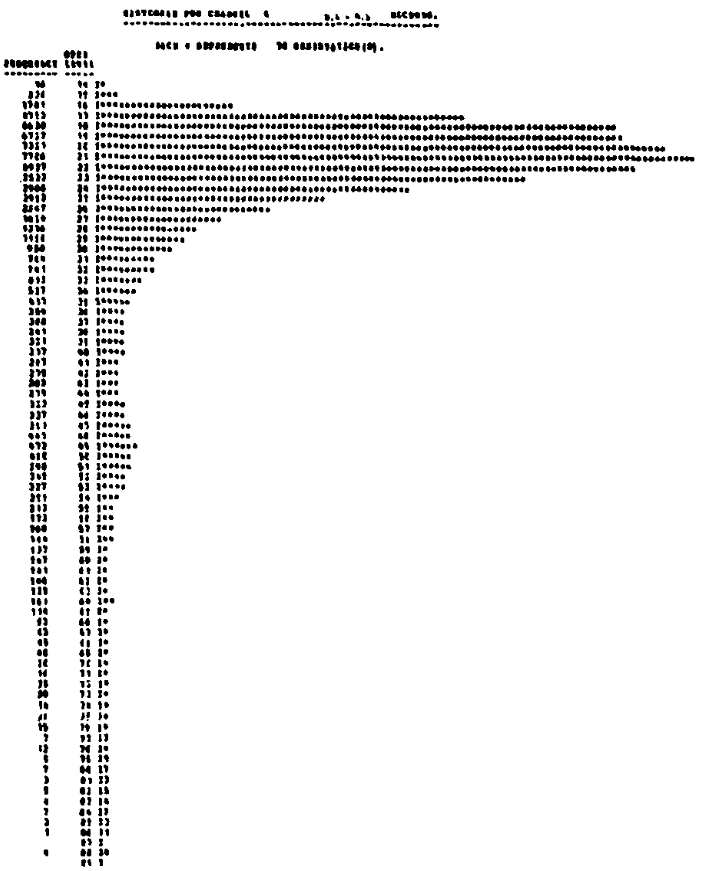


FIGURE 5.4c Histogram of the data set in the spectral range 0.4–0.5 μm (blue).

PROBLEMS

5.1 We are given the following patterns and their class belongings:

$$(1,0),(3,0),(2,1),(1,2),(3,2) \in \omega_1$$
$$(4,3),(4,5),(5,4),(6,3),(6,5) \in \omega_2$$

Obtain the equation of the Bayes decision boundary between the two classes by assuming that $p(\omega_1) = p(\omega_2) = \frac{1}{2}$.

5.2 Consider the following patterns:

$$\left.\begin{matrix}(0,1,0),(0,1,1),(1,0,1),(1,1,1)\\(0,0,0),(0,0,1),(1,0,0),(1,1,0)\end{matrix}\right\} \in \omega_1$$

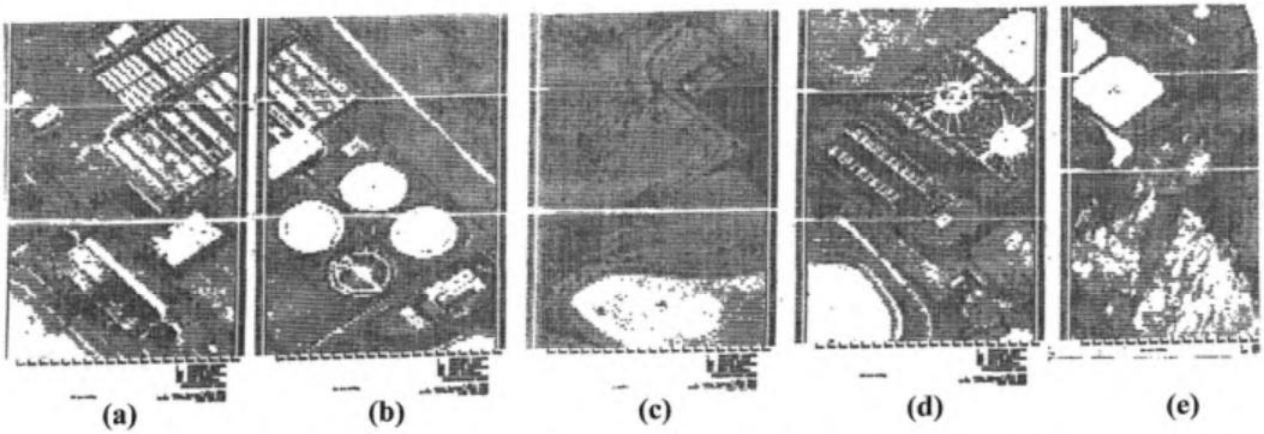


FIGURE 5.5 Computer symbol-coding map obtained from the data set given: (a) from columns 200 to 299; (b) from columns 300 to 399; (c) from columns 400 to 499; (d) from columns 500 to 599; (e) from columns 600 to 699.

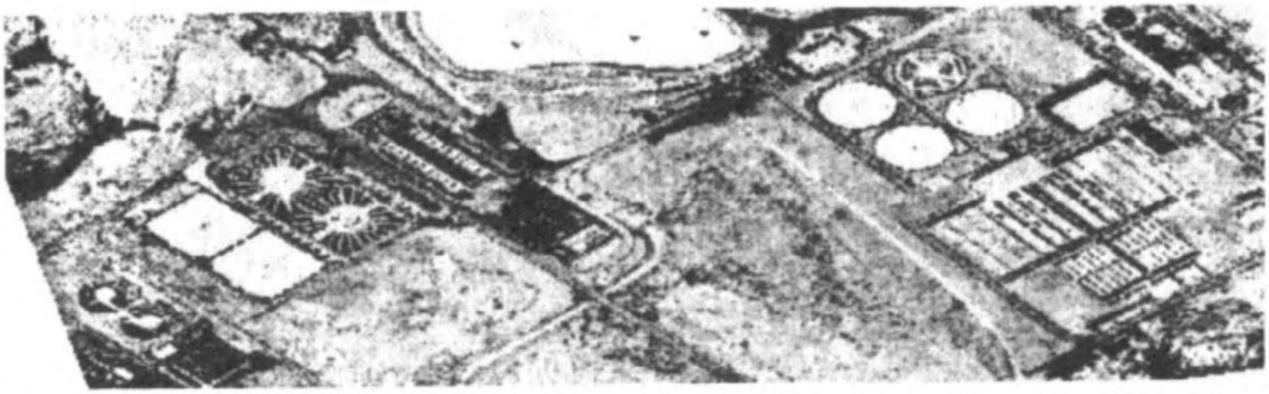


FIGURE 5.6 Computer image plot from the data set given. This image is plotted with norm values. (Data courtesy of Frederick Luce, ORSER, Pennsylvania State University.)

and

$$\left.\begin{array}{l}(5,5,5),(5,6,6),(6,5,5),(6,6,6)\\(5,6,5),(5,5,6),(6,5,6),(6,6,5)\end{array}\right\}\in\omega_2$$

Find the Bayes decision surface between the two classes of patterns, assuming that they have normal probability density functions and that $p(\omega_1) = p(\omega_2) = \frac{1}{2}$.

5.3 Derive a Bayes discriminant function for a negative loss function, such as

$$L_{ik} = \begin{cases} -h_1 & \text{if } k = i \\ 0 & \text{otherwise } (k \neq i) \end{cases}$$

5.4 Derive the Bayes discriminant function for patterns with independent binary components, and see whether the discriminant function is linear.

5.5 Show that for $\mathbf{x} \in \omega_2$, the ratio $p(\mathbf{x}|\omega_1)/p(\mathbf{x}|\omega_2)$ will be distributed with a mean equal to $-\frac{1}{2}\mathbf{r}_{12}$, and a variance equal to $\mathbf{r}_{12}$.

5.6 From the mean and covariance of the normally distributed multivariate problems, we can easily estimate respectively the center and the shape of the cluster, and vice versa. Given the relative values of $\mathbf{m}, C_{11}, C_{12}, C_{21}, C_{22}$, draw the center and shape of the clusters for following different cases:

Case I:

$$\mathbf{m}^T = [0 \quad 0]$$

$$C_{11} = C_{22} = 1$$

$$C_{12} = C_{21} = 0$$

Case II:

$$\mathbf{m}^T = [2 \quad 0]$$

$$C_{11} > C_{22}$$

$$C_{12} = C_{21} = 0$$

Case III:

$$\mathbf{m} = [1 \quad 0]$$

$$C_{11} < C_{22}$$

$$C_{12} = C_{21} = 0$$

Case IV:

$\mathbf{m} = [0 \quad 3]$

$C_{22} > C_{11}$

$C_{12} = C_{21} = 0$

Case V:

$\mathbf{m} = [0 \quad 2]$

$C_{11} > C_{22}$

$C_{12} \neq C_{21} \neq 0$

Case VI:

$\mathbf{m} = [2 \quad 2]$

$C_{11} < C_{22}$

$C_{12} \neq C_{21} \neq 0$

5.7 According to the maximum likelihood rule, the discriminant function $d_k(\mathbf{x})$ can be simplified as

$$d_k(\mathbf{x}) = \mathbf{x}^T C_k^{-1} \mathbf{m}_k - \tfrac{1}{2} \mathbf{m}_k C_k^{-1} \mathbf{m}_k$$

Find the Bayes decision boundary to separate the following two classes:

Patterns belonging to class 1 (ω_1):

$\mathbf{x}_{11} = (2, \ 3)^T$

$\mathbf{x}_{12} = (4, \ 1)^T$

$\mathbf{x}_{13} = (4, \ 3)^T$

$\mathbf{x}_{14} = (4, \ 5)^T$

$\mathbf{x}_{15} = (6, \ 3)^T$

Patterns belonging to class 2 (ω_2):

$\mathbf{x}_{21} = (-3, \ -1)^T$

$\mathbf{x}_{22} = (-3, \ -3)^T$

$\mathbf{x}_{23} = (-3, \ -5)^T$

$\mathbf{x}_{24} = (-1, \ -3)^T$

$\mathbf{x}_{25} = (-5, \ -3)^T$

6

Clustering Analysis and Unsupervised Learning

6.1 INTRODUCTION

6.1.1 Definition of Clustering

What we have discussed so far has been supervised learning; that is, there is a supervisor to teach the system how to classify a known set of patterns first, and then let the system go ahead freely to classify other patterns. In such systems we usually need a priori information (information on syntax, semantics, or pragmatics) to form the basis of teaching.

In this chapter we discuss nonsupervised learning, in which the classification process will not depend on a priori information. As a matter of fact, it happens quite frequently that there does not exist much a priori knowledge about the patterns; neither can the proper training pattern sets be obtained.

Clustering is the nonsupervised classification of objects. It is the process of generating classes without any a priori knowledge of prototype classification.

When we are given M patterns, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$, contained in the pattern space S, the process of clustering can be formally stated as: to seek the regions $S_1, S_2, \ldots, S_K$ such that every $\mathbf{x}_i$, $i = 1, 2, \ldots, M$, falls into one of these regions and no $\mathbf{x}_i$ falls in two regions; that is,

$$\begin{aligned} &S_1 \cup S_2 \cup S_3 \cup \cdots \cup S_K = S \\ &S_i \cap S_j = \emptyset \qquad \forall i \neq j \end{aligned} \tag{6.1}$$

where $\cup$ and $\cap$ stand for union and intersection, respectively.

Algorithms derived for clustering classify objects into clusters by natural association according to some similarity measures. It is expected that the degree of natural association is high among members belonging to the same category and low among members of different categories.

6.1.2 Similarity Measure

From the definition of clustering, we are to cluster, or form into each class, those patterns $\mathbf{x}_i$ that are as much alike as possible, and hence we need some kind of similarity measure (or dissimilarity measure). If ζ denotes the dissimilarity measure between two patterns, it is obvious that

$$\zeta(\mathbf{x}_i, \mathbf{x}_i) = 0$$

but

$$\zeta(\mathbf{x}_i, \mathbf{x}_j) \neq 0 \qquad \forall j \neq i \tag{6.2}$$

The similarity measure (or dissimilarity measure) is usually given in numerical form to indicate the degree of natural association or degree of resemblance between patterns in a group, between a pattern and a group of patterns, or between pattern groups.

Many different functions, such as the inertia function and the fuzzy membership function, have also been suggested as the similarity measure, but the most common ones are described next.

Euclidean Distance

Euclidean distance is the simplest and most frequently used measure and is represented by

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j|^2 \tag{6.3}$$

in multidimensional euclidean space. It may be all right to use this distance as a similarity measure if the relative size of the dimension has significance. If not, we should consider weighted euclidean distance, which is

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{n} \alpha_k (x_{ki} - x_{kj})^2 \tag{6.4}$$

where $\mathbf{x}_i = [x_{1i}, x_{2i}, \dots, x_{ni}]^T$; x_{ki} and x_{kj} are the kth components of $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively; and α_k is the weighting coefficient. In particular, let us let $\mathbf{m}_m = [m_{1m}, m_{2m}, \dots, m_{nm}]^T$ be the mean of the mth cluster (we still presume the class is unknown), and let

$$\alpha_k = \frac{1}{\sigma_{km}^2} \tag{6.5}$$

where $\boldsymbol{\sigma}_m = [\sigma_{1m}, \sigma_{2m}, \ldots, \sigma_{nm}]$ and σ_{km}^2 is the variance of the mth cluster in the kth direction. Then the weighted euclidean distance from $\mathbf{x}_i$ to the mth cluster is

$$d_m^2(\mathbf{x}_i, \mathbf{m}_m) = \sum_{k=1}^{n} \frac{(x_{ki} - m_{km})^2}{\sigma_{km}^2} \tag{6.6}$$

The cluster shapes obtained by using this measure have loci of equal d_m^2, which are hyperellipsoids aligned with the axes of the n-dimensional pattern space.

Mahalanobis Distance

The squared Mahalanobis distance from $\mathbf{x}_i$ to $\mathbf{x}_j$ is in the form

$$r(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T C^{-1} (\mathbf{x}_i - \mathbf{x}_j) \tag{6.7}$$

where C^{-1} is the inverse of the covariance matrix.

Tanimoto Coefficient

Tanimoto suggested a similarity ratio known as the Tanimoto coefficient:

$$d_t(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\mathbf{x}_i^T \mathbf{x}_i + \mathbf{x}_j^T \mathbf{x}_j - \mathbf{x}_i^T \mathbf{x}_j} \tag{6.8}$$

where $\mathbf{x}_i^T \mathbf{x}_j$ denotes the number of common attributes between $\mathbf{x}_i$, and $\mathbf{x}_j$, $\mathbf{x}_i^T \mathbf{x}_i$ denotes the number of attributes possessed by $\mathbf{x}_i$, and $\mathbf{x}_j^T \mathbf{x}_j$ denotes the number of attributes possessed by $\mathbf{x}_j$. The denominator then gives the number of attributes that are in $\mathbf{x}_i$ or $\mathbf{x}_j$ but not in both. The entire expression will therefore represent the ratio of the number of common attributes between $\mathbf{x}_i$ and $\mathbf{x}_j$ to the number of attributes that are in either one of the vectors $\mathbf{x}_i$, $\mathbf{x}_j$ but not in both.

6.1.3 Types of Clustering Algorithms

Classification of Clustering Algorithms

Lots of clustering algorithms have been suggested. They can be grouped into direct (constructive) or indirect (optimization) algorithms according to whether or not a criterion function is used in the clustering process. For a direct approach, sometimes called the *heuristic approach*, it is simply to isolate pattern classes without the necessity of using a criterion function, whereas for an indirect approach we do use a criterion function to optimize the classification.

Very frequently, clustering algorithms can be classified as an agglomerative or a divisive approach according to the clustering process being worked along the "bottom-up" or the "top-down" direction. A clustering algorithm is said to be *agglomerative* if it starts from isolated patterns and coalesces the nearest patterns or groups according to a threshold from the bottom up to form hierarchies.

A clustering algorithm is said to be *divisive* if it starts from a set of patterns and divides along the top-down direction on minimizing or maximizing some estimating function into optimum clusters.

Many programs have been written in each of these algorithms, but quite a lot have tried to take advantage of both and include both divisive and agglomerative approaches in one program. This leads to another classification based on whether the number of classes is known or unknown beforehand. This is the method we use in this book.

Intraset and Interset Distances: OneType of Criterion*

We mentioned earlier that the degree of natural association is expected to be high among members belonging to the same category, and low among members of different categories. In other words, the intraset distance should be small, whereas the interset distance should be large.

Mathematically, the interset distance between two separate sets is

$$D_{12} = \overline{D^2([\mathbf{x}_1^i], [\mathbf{x}_2^j])} \qquad i = 1, 2, \ldots, N_1; j = 1, 2, \ldots, N_2 \tag{6.9}$$

or

$$D_{12} = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} D^2(\mathbf{x}_1^i, \mathbf{x}_2^j) \tag{6.10}$$

which is the average squared distance between points of separate classes. The subscripts 1 and 2 in the pattern sets $[\mathbf{x}_1^i]$ and $[\mathbf{x}_2^j]$ represent classes ω_1 and ω_2, respectively, and N_1 and N_2 are the number of pattern samples in classes ω_1 and ω_2, respectively.

The intraset distance for a set of N patterns (all patterns belonging to the same class) can be derived similarly. Since

$$D^2(\mathbf{x}^i, \mathbf{x}^j) = \sum_{k=1}^{n} (x_k^i - x_k^j)^2 \tag{6.11}$$

the mean squared distance from a specific $\mathbf{x}^i$ to $N - 1$ other patterns in the same set is

$$\overline{D^2(\mathbf{x}^i, [\mathbf{x}^j])} = \frac{1}{N-1} \sum_{j=1}^{N} \sum_{k=1}^{n} (x_k^i - x_k^j)^2 \tag{6.12}$$

*See Tou and Gonzalez (1974) and Babu (1973) for supplementary reading.

The intraset distance or the average over all N patterns in the set is then

$$D_{ii} = \overline{D^2([\mathbf{x}^i], [\mathbf{x}^j])} = \frac{1}{N}\sum_{j=1}^{N}\left[\frac{1}{N-1}\sum_{j=1}^{N}\sum_{k=1}^{n}(x_k^i - x_k^j)^2\right] \tag{6.13}$$

or

$$D_{ii} = \frac{1}{N-1}\sum_{k=1}^{n}\left[\frac{1}{N^2}\sum_{j=1}^{N}\sum_{i=1}^{N}(x_k^i - x_k^j)^2\right] \tag{6.14}$$

Expanding the terms inside the brackets yields

$$\begin{aligned} D_{ii} &= \overline{D^2([\mathbf{x}^i], [\mathbf{x}^j])} \\ &= \frac{N}{N-1}\sum_{k=1}^{N}\left[\frac{1}{N}\sum_{j=1}^{N}\frac{1}{N}\sum_{i=1}^{N}(x_k^i)^2 - 2\frac{1}{N}\sum_{i=1}^{N}x_k^i\frac{1}{N}\sum_{j=1}^{N}x_k^j + \frac{1}{N}\sum_{i=1}^{N}\frac{1}{N}\sum_{j=1}^{N}(x_k^j)^2\right] \\ &= \frac{N}{N-1}\sum_{k=1}^{n}\left[\frac{1}{N}\sum_{j=1}^{N}\overline{(x_k^i)^2} - 2\overline{x_k^i}\,\overline{x_k^j} + \frac{1}{N}\sum_{i=1}^{N}\overline{(x_k^j)^2}\right] \end{aligned} \tag{6.15}$$

Since we are working on the same pattern set,

$$\overline{(x_k^j)^2} = \overline{(x_k^i)^2} \tag{6.16}$$

we have

$$D_{ii} = \overline{D^2([\mathbf{x}^i], [\mathbf{x}^j])} = \frac{2N}{N-1}\sum_{k=1}^{n}[\overline{(x_k^i)^2} - (\overline{x_k^i})^2] \tag{6.17}$$

Note that, by definition, the variance of the kth component of N patterns is given by

$$\begin{aligned} (\sigma_k)^2 &= \frac{1}{N}\sum_{i=1}^{N}(x_k^i - \overline{x_k^i})^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}(x_k^i)^2 - \frac{2}{N}\sum_{i=1}^{N}x_k^i\overline{x_k^i} + \frac{1}{N}\sum_{i=1}^{N}(\overline{x_k^i})^2 \\ &= \overline{(x_k^i)^2} - (\overline{x_k^i})^2 \end{aligned} \tag{6.18}$$

after simplification. Therefore, the intraset distance is now

$$D_{ii} = \overline{D^2([\mathbf{x}^i], [\mathbf{x}^j])} = \frac{2N}{N-1}\sum_{k=1}^{n}\sigma_k^2 \tag{6.19}$$

6.1.4 General Remarks

Most clustering algorithms are heuristic. Most papers on clustering present experimental evidence of results to illustrate the effectiveness of their clustering processes. But to our knowledge, no objective quantitative measure of clustering performance is yet available, although a lot of effort has been expended toward that end. We are not quite certain, at least at the moment, how data dependent the results are.

In clustering applications we generally try to locate the modes, that is, to obtain the local maximum of the probability density if the number M of the class is known. When the number of classes is unknown, we usually try to obtain an estimate of the number and location of modes, that is, to find the natural grouping of patterns. Thus we "learn" something about the statistics. For example, the mean and covariance of the data to be analyzed are useful for data preprocessing and training of the minimum distance classifier for multiple classes as well as for on-line adaptive classification in a nonstationary environment. This is because more significant features from the measurement vectors are extracted to realize more efficient and more accurate pattern classification.

6.2 CLUSTERING WITH AN UNKNOWN NUMBER OF CLASSES

6.2.1 Adaptive Sample Set Construction (Heuristic Method)

When the number of classes is unknown, classification by clustering is actually to construct the probability densities from pattern samples. Adaptive sample set construction is one of the approaches commonly used.

The essential point of this algorithm is to build up clusters by using distance measure. The first cluster can be chosen arbitrarily. Once the cluster is chosen, try to assign pattern samples to it if the distance from a sample to this cluster center is less than a threshold, If not, form a new cluster. When a pattern sample falls in a cluster, the mean and variance of that cluster will be adjusted. Repeat the process until all the pattern samples are assigned. The whole procedure consists of the following steps:

Step 1. Take the first sample as representative of the first cluster:

$$\mathbf{z}_1 = \mathbf{x}_1$$

where $\mathbf{z}_1$ is the first cluster center.

Step 2. Take the next sample and compute its distance (similarity measure) to all the existing clusters (when starting, there is only one cluster).

a. Assign $\mathbf{x}$ to $\mathbf{z}_i$ (the ith cluster) if

$$d_i(\mathbf{x}, \mathbf{z}_i) \leq \theta\tau \qquad 0 \leq \theta \leq 1 \tag{6.20}$$

where τ is the membership boundary for a specified cluster. Its value is properly set by the designer.

b. Do not assign $\mathbf{x}$ to $\mathbf{z}_i$ if

$$d_i(\mathbf{x}, \mathbf{z}_i) > \tau \tag{6.21}$$

c. No decision will be made on $\mathbf{x}$ if $\mathbf{x}$ falls in the "intermediate region" for $\mathbf{z}_i$, as shown in Figure 6.1.

Step 3. a. Each time a new $\mathbf{x}$ is assigned to $\mathbf{z}_i$, compute $\mathbf{z}_i(n+1)$ and $\mathbf{C}(n+1)$ according to the following expressions:

$$\left.\begin{aligned} \mathbf{z}_i(n+1) &= \frac{1}{n+1}[n\mathbf{z}_i(n) + \mathbf{x}] \qquad (6.22) \\ \mathbf{C}(n+1) &= \frac{1}{n+1}[n\mathbf{C}(n) + (\mathbf{x} - \mathbf{z}_i(n+1))^2] \qquad (6.23) \end{aligned}\right\} \text{for} = 1, 2, \ldots, M$$

Where n is the number of pattern samples already assigned to $\mathbf{z}_i$ and $\mathbf{x}$ is the $(n+1)$st such sample. $\mathbf{z}_i(n)$ and $\mathbf{C}(n)$, the variance, were already computed from the n samples.

b. Form a new cluster $\mathbf{z}_j$ if

$$d(\mathbf{x}, \mathbf{z}_i) > \tau \qquad \forall i \tag{6.24}$$

Step 4. Repeat steps 2 and 3 until all pattern samples have been assigned. There would be some reassignment of $\mathbf{x}$ when all $\mathbf{x}$ are again passed through in order. This is because the means and variances have been adjusted with each $\mathbf{x}$ assigned to $\mathbf{z}_i$.

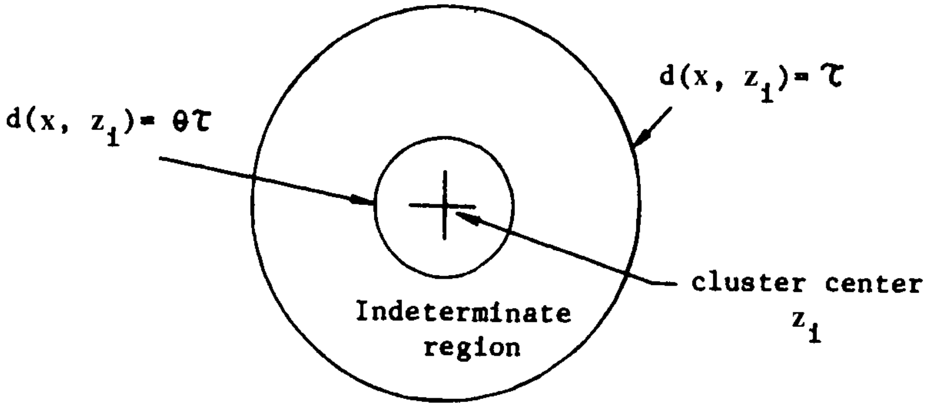


FIGURE 6.1 Clustering based on a distance measure.

Step 5. After the training is considered complete (that means that $\mathbf{x}$ no longer changes class belongings, or some number of $\mathbf{x}$ are unassigned each time), we can let the system go freely to do the clustering on a large number of pattern samples. No indeterminate region will exist this time. All $\mathbf{x}$'*s* falling on the indeterminate region may be assigned to the nearest class according to the minimum distance rule. All those $\mathbf{x}$'s could be considered unclassified if their distances to all cluster centers are greater than τ.

This algorithm is simple and efficient. Other than these, it possesses the following advantages: Minimum computations are required; pattern samples are processed sequentially without the necessity of being stored; and there is no need to have the number of classes specified.

On the other hand., there are some drawbacks to the use of this algorithm. First, strong assumptions are required, such as that clusters themselves should be tight and also that clusters should be widely separated from one another. Second, clustering results are dependent on the order of presentation of $\mathbf{x}$'s and also on the first $\mathbf{x}$ being used as the initial cluster center. If, for example, cluster center $\mathbf{z}_i$ (and also $\mathbf{C}$) changes, or $\mathbf{x}(n)$ is presented at a later order $n + m$, that pattern sample might be classified differently. Also, different results of clustering might have resulted during training. Third, clustering results also depend heavily on the value of $\tau\theta$ chosen.

6.2.2 Batchelor and Wilkins' Algorithm

Batchelor and Wilkins suggested another simple heuristic procedure for clustering, sometimes known as the maximum distance algorithm. An artificially simple example consisting of 10 two-dimensional patterns as shown in Figure 6.2a and b is used to illustrate the procedure of this algorithm.

Step 1. Arbitrarily, let $\mathbf{x}_1$ be the first cluster center, designated by $\mathbf{z}_1$.

Step 2. Determine the pattern sample farthest from $\mathbf{x}_1$, which is $\mathbf{x}_6$. Call it cluster center $\mathbf{z}_2$.

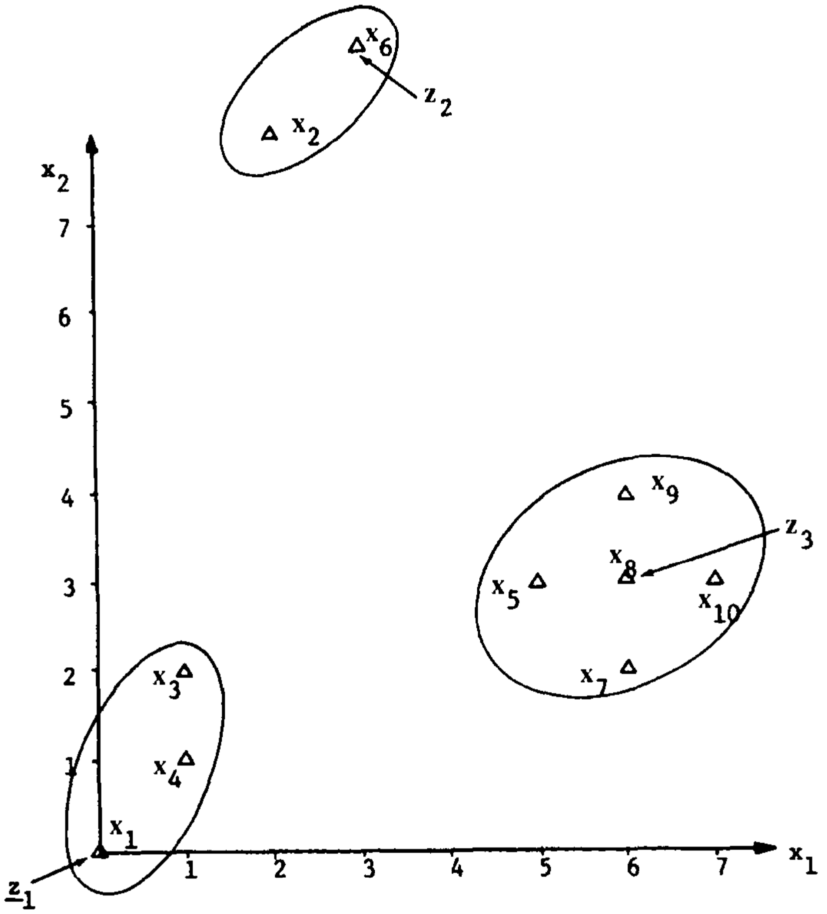
Step 3. Compute the distance from each remaining pattern sample to $\mathbf{z}_1$ and $\mathbf{z}_2$.

Step 4. Save the minimum distance for each pair of these computations.

Step 5. Select the maximum of these minimum distances.

Step 6. If this distance is appreciably greater than a fraction of the distance $d(\mathbf{z}_1, \mathbf{z}_2)$, call the corresponding sample cluster center $\mathbf{z}_3$. Otherwise, the algorithm is terminated.

Step 7. Compute the distance from each of the three established cluster centers to the remaining samples and save the minimum of every group of three distances. Again, select the maximum of these



(a)

FIGURE 6.2a Illustrative example for Batchelor and Wilkins' algorithm. 10 two-dimensional patterns.

minimum distances. If this distance is an appreciable fraction of the "typical" previous maximum distances, the corresponding sample becomes cluster center z_4. Otherwise, the algorithm is terminated.

Step 8. Repeat until the new maximum distance at a particular step fails to satisfy the condition for the creation of a new cluster center.

Step 9. Assign each sample to its nearest cluster center.

Figure 6.2b tabulates the intermediate clustering results. x_1, x_6, and x_8 are the three cluster centers. $[x_1, x_3, x_4]$, $[x_2, x_6]$, and $[x_5, x_7, x_8, x_9, x_{10}]$ are the three cluster domains.

1. $\mathbf{z}_1 = \mathbf{x}_1$
2. $|\mathbf{x}_6 - \mathbf{z}_1| > |\mathbf{x}_2 - \mathbf{z}_1| > |\mathbf{x}_{10} - \mathbf{z}_1| > |\mathbf{x}_9 - \mathbf{z}_1| > |\mathbf{x}_8 - \mathbf{z}_1| > |\mathbf{x}_7 - \mathbf{x}_1|$
   $> |\mathbf{x}_5 - \mathbf{z}_1| > |\mathbf{x}_3 - \mathbf{z}_1| > |\mathbf{x}_4 - \mathbf{z}_1|$
   Let $\mathbf{z}_2 = \mathbf{x}_6$
3. $|\mathbf{x}_4 - \mathbf{z}_1| < |\mathbf{x}_4 - \mathbf{z}_2|$
   $|\mathbf{x}_3 - \mathbf{z}_1| < |\mathbf{x}_3 - \mathbf{z}_2|$
   $|\mathbf{x}_5 - \mathbf{z}_1| < |\mathbf{x}_5 - \mathbf{z}_2|$
   $|\mathbf{x}_8 - \mathbf{z}_2| < |\mathbf{x}_8 - \mathbf{z}_1|$
   $|\mathbf{x}_7 - \mathbf{z}_1| < |\mathbf{x}_7 - \mathbf{z}_2|$
   $|\mathbf{x}_9 - \mathbf{z}_2| < |\mathbf{x}_9 - \mathbf{z}_1|$
   $|\mathbf{x}_{10} - \mathbf{z}_2| < |\mathbf{x}_{10} - \mathbf{z}_1|$
   $|\mathbf{x}_2 - \mathbf{z}_2| < |\mathbf{x}_2 - \mathbf{z}_1|$
4. Save all distances on left in step 3.
5. The maximum of distances saved in step 4 is $|\mathbf{x}_8 - \mathbf{z}_2|$.
6. Since $|\mathbf{x}_8 - \mathbf{z}_2| > \frac{1}{2}|\mathbf{z}_2 - \mathbf{z}_1|$, let $\mathbf{z}_3 = \mathbf{x}_8$.
7. $|\mathbf{x}_4 - \mathbf{z}_1| < |\mathbf{x}_4 - \mathbf{z}_3| < |\mathbf{x}_4 - \mathbf{z}_2|$
   $|\mathbf{x}_3 - \mathbf{z}_1| < |\mathbf{x}_3 - \mathbf{z}_3| < |\mathbf{x}_3 - \mathbf{z}_2|$
   $|\mathbf{x}_5 - \mathbf{z}_3| < |\mathbf{x}_5 - \mathbf{z}_1| < |\mathbf{x}_5 - \mathbf{z}_2|$
   $|\mathbf{x}_7 - \mathbf{z}_3| < |\mathbf{x}_7 - \mathbf{z}_1| < |\mathbf{x}_7 - \mathbf{z}_2|$
   $|\mathbf{x}_9 - \mathbf{z}_3| < |\mathbf{x}_9 - \mathbf{z}_2| < |\mathbf{x}_9 - \mathbf{z}_1|$
   $|\mathbf{x}_{10} - \mathbf{z}_3| < |\mathbf{x}_{10} - \mathbf{z}_2| < |\mathbf{x}_{10} - \mathbf{z}_1|$
   $|\mathbf{x}_2 - \mathbf{z}_2| < |\mathbf{x}_2 - \mathbf{z}_3| < |\mathbf{x}_2 - \mathbf{z}_1|$
8. Save all distances on left.
9. The maximum of these minimum distances is $|\mathbf{x}_3 - \mathbf{z}_1|$.
10. Since $|\mathbf{x}_3 - \mathbf{z}_1| < \frac{1}{2}$ Avg$[|\mathbf{z}_2 - \mathbf{z}_1|, |\mathbf{z}_3 - \mathbf{z}_2|]$ and since the condition for creation of a new cluster is not satisfied, the algorithm terminates.

FIGURE 6.2b Intermediate clustering results.

6.2.3 Hierarchical Clustering Algorithm Based on *k*-Nearest Neighbors

Natural association by minimum distance (closeness measure) works very well for many cases that can be distinctly separated. However, it does not work well for sets of data where there are no clearly cut boundary surfaces among them, nor for

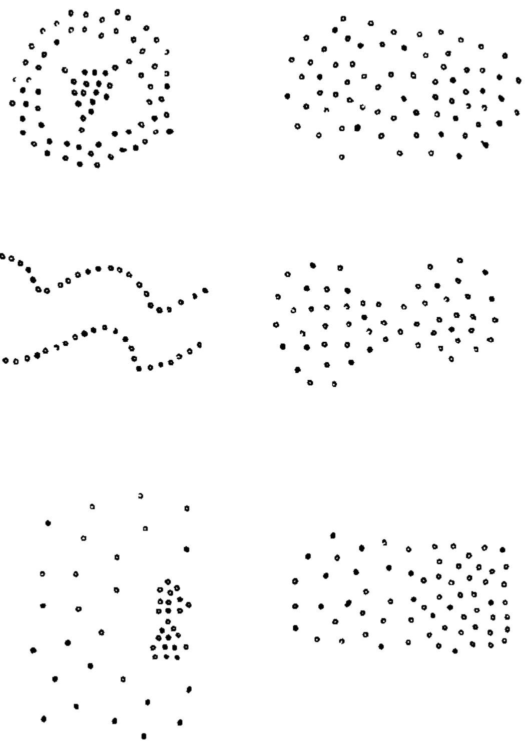


FIGURE 6.3 Examples showing pattern sets with no clearly cut boundaries or patterns belonging to different classes are interweaved together.

those patterns which belong to different classes but are being interweaved together, as shown in Figures 6.3 and 6.4. For such kinds of problem an approach called *nearest neighbor classification* might be useful for their solution. The nearest neighbor classification is a process to assign a pattern point to a class to which its nearest neighbor belongs. If membership is decided by a majority vote of the *k*-nearest neighbors, the procedure will be called a *k*-nearest neighbor

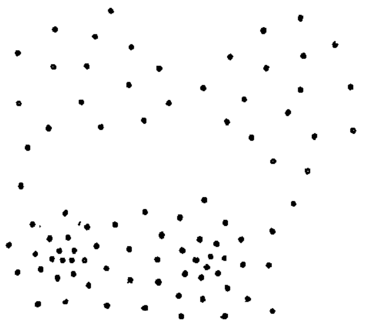


FIGURE 6.4 Examples showing pattern sets with no clearly cut boundaries or pattern belonging to different classes are interweaved together.

decision rule. The clustering algorithm discussed in this section follows the concept suggested by Mizoguchi and Kakusho (1978). The procedure consists of two stages, In the first stage, pregrouping of data is made to obtain subclusters (steps 1 to 4). In the second stage (i.e., the remaining part of the algorithm), the subclusters are merged hierarchically by using a similarity measure.

The algorithm can be generalized as follows.

1. Determine k appropriately.
2. Compute $\Omega_k(\mathbf{x}_i)$, $P_k(\mathbf{x}_i)$, and $\xi_k(\mathbf{x}_i)$ for every pattern sample, where $\Omega_k(\mathbf{x}_i)$ is a set of k-nearest neighbors of the sample pattern point $\mathbf{x}_i$, $i = 1, 2, \ldots, N$, based on a euclidean distance measure. $P_k(\mathbf{x}_i)$ is the potential of the pattern sample point $\mathbf{x}_i$ and is defined as

$$P_k(\mathbf{x}_i) = \frac{1}{k} \sum_{\mathbf{x}_j \in \Omega_k(\mathbf{x}_i)} d(\mathbf{x}_i, \mathbf{x}_j)$$

$d(\mathbf{x}_i, \mathbf{x}_j)$ is the dissimilarity measure between sample points $\mathbf{x}_i$ and $\mathbf{x}_j$. Obviously, $d(\mathbf{x}_i, \mathbf{x}_i) = 0$. The smaller the value of $P_k(\mathbf{x}_i)$, the larger the potential of $\mathbf{x}_i$ to be a cluster center. For any pattern point, we can always find its k-nearest neighbors, but the distance measure (length) may not be the same. $\xi_k(\mathbf{x}_i)$ is a set of sample points k-adjacent to the sample point $\mathbf{x}_i$. Figure 6.5 illustrates geometrically the definitions of $\Omega_k(\mathbf{x}_i)$ and $\xi_k(\mathbf{x}_i)$ for a set of points $[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_6;\ \mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c]$. Points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$ and $\mathbf{x}_6$ are k-nearest neighbors (kNN) of $\mathbf{x}_i$, or $\mathbf{x}_i$ is k-adjacent to these six pattern points. If $P_k(\mathbf{x}_i)$ has the highest potential among the six pattern points, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_6$, then these six pattern points $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3, \ldots$, $\mathbf{x}_6$ will cluster to $\mathbf{x}_i$ to form a cluster as shown. So we have

$\Omega_k(\mathbf{x}_i) = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_6]$	kNN of $\mathbf{x}_i$.
$\Omega_k(\mathbf{x}_a) = [\mathbf{x}_i, \mathbf{x}_b, \ldots.]$	kNN of $\mathbf{x}_a$
$\Omega_k(\mathbf{x}_b) = [\mathbf{x}_i, \mathbf{x}_a, \ldots.]$	kNN of $\mathbf{x}_b$
$\xi_k(\mathbf{x}_i) = [\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \ldots]$	a set of pattern points which are k-adjacent to $\mathbf{x}_i$

$\xi_k(\mathbf{x}_i) = [\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \ldots]$ implies that $\mathbf{x}_i$ is a k-nearest neighbor of $\mathbf{x}_a$; it is also a k-nearest neighbor of $\mathbf{x}_b$, $\mathbf{x}_c$, etc. It also implies that there are possibilities to assign $\mathbf{x}_i$ either to $\mathbf{x}_a$ or to $\mathbf{x}_b$, $\mathbf{x}_c, \ldots$ depending on which one among $P_k(\mathbf{x}_a)$, $P_k(\mathbf{x}_b)$, $P_k(\mathbf{x}_c), \ldots$ has the highest potential to be the subcluster [i.e., which one among $P_k(\mathbf{x}_a)$, $P_k(\mathbf{x}_b)$, $P_k(\mathbf{x}_c)$ has the smallest value]. Note that $\mathbf{x}_a$, $\mathbf{x}_b, \ldots$ may not be the k-nearest neighbor of $\mathbf{x}_i$. In other words, $\mathbf{x}_i$ may not be k-adjacent to $\mathbf{x}_a$. This means that there is no possibility to assign $\mathbf{x}_a$ to $\mathbf{x}_i$. So, we have to compute $\Omega_k(.)$ and $\xi_k(.)$ for every pattern point to determine its

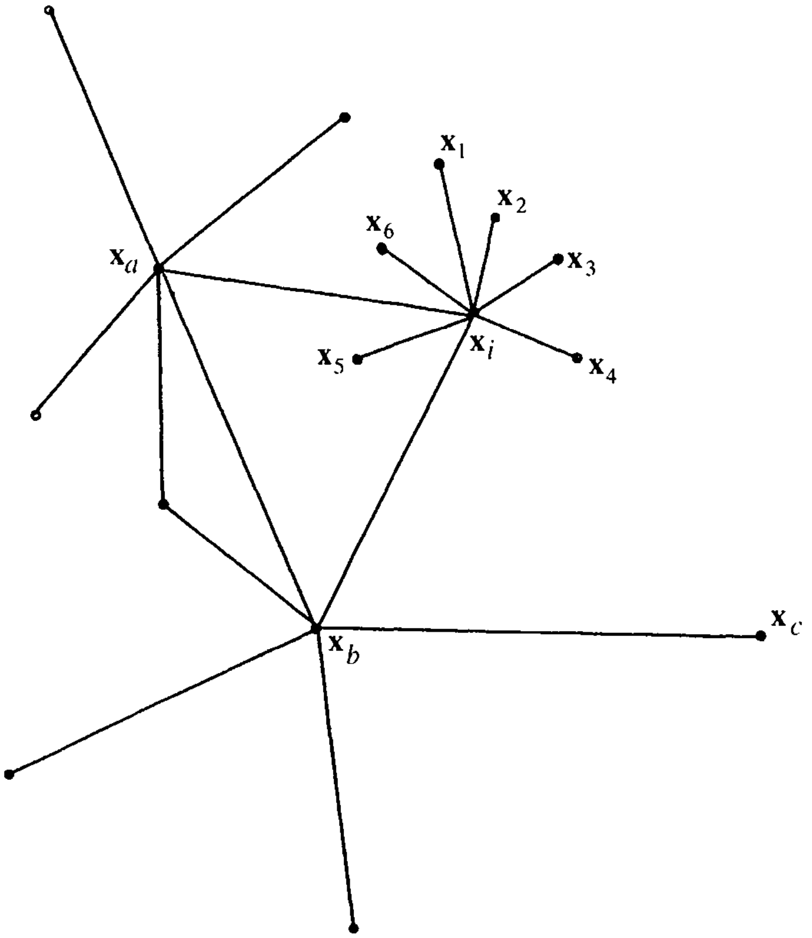


FIGURE 6.5 Definitions of $\Omega_k(\mathbf{x}_i)$ and $\xi_k(\mathbf{x}_i)$.

subordination; i.e., to determine whether it is to be subordinated to some other pattern points, or other pattern points subordinated to it to form a subcluster. Through such a sequence operation of subordination, all the pattern points will be agglomerated to form a subcluster or a certain number of subclusters.

3. Subordinate every point $\mathbf{x}_i$ to the point $\mathbf{x}_j$ such that

$$P_k(\mathbf{x}_j) = \min_{\mathbf{x}_m \in \xi_k(\mathbf{x}_i)} \mathrm{P}_k(\mathbf{x}_m)$$

That is, subordinate every sample point $\mathbf{x}_i$ to $\mathbf{x}_j$ that is in the set $\xi_k(x_i)$ and has the smallest value of $P_k(\mathbf{x}_j)$, or has the highest potential to be a subcluster. If $P_k(\mathbf{x}_i) = P_k(\mathbf{x}_j)$, then this pattern point $\mathbf{x}_i$ will subordinate to no point.

4. Detect and count the subclustering points and assign every point to its nearest subclusters. These four steps completes the data pregrouping task in the first stage of the algorithm.
5. Merge the subclusters according to whether or not there are k-boundary points between the two neighboring subclusters.
   a. If there exist k-boundary point sets between pairs of neighboring subclusters, merge the two most similar subclusters among those unordered pairs of subclusters by a similarity measure, $\mathrm{SIM}(m, n)$, which is

$$\mathrm{SIM}_1(m, n) * \mathrm{SIM}_2(m, n)$$

where m and n denote the subclusters. $\mathrm{SIM}_1(m, n)$ represents the difference in density between the cluster and the boundary, and can be used to detect the valley in the pattern dispersion. $\mathrm{SIM}_2(m, n)$ represents the relative size of the boundary to that of the cluster, and can be used to detect the neck between two clusters. Mathematically,

$$\mathrm{SIM}_1(m, n) = \frac{\min[P_k^{\mathrm{sc}}(m), P_k^{\mathrm{sc}}(n)]}{\max[\mathrm{BP}_k^{m,n}, \mathrm{BP}_k^{n,m}]}$$

and

$$\mathrm{SIM}_2(m, n) = \frac{N(Y_k^{m,n}) + N(Y_k^{n,m})]}{2 \min[N(W_m), N(W_n)]}$$

Let us elaborate a little bit in details about the terms: $P_k^{\mathrm{sc}}(m)$, $P_k^{\mathrm{sc}}(n)$, $\mathrm{BP}_k^{m,n}$, $\mathrm{BP}_k^{n,m}$, $Y_k^{m,n}$, $Y_k^{n,m}$, $N(W_m)$, and $N(W_n)$. Figure 6.6 shows the pattern points that are in subcluster m and are also the k-nearest neighbors of $\mathbf{c}_m$ which is the center of subcluster m. It also shows in the same figure those pattern points that are in subcluster n and are also the k-nearest neighbors of $\mathbf{c}_n$, the center of the subcluster n. Figure 6.7 shows the pattern points that are in subcluster m and are also the k-nearest neighbors of some pattern points belonging to subcluster n.

Define $P_k^{\mathrm{sc}}(m)$ as the average of $P_k(\mathbf{x}_i)$ over all the points that are in subcluster m and are also the k-nearest neighbors of the cluster center $\mathbf{c}_m$ (the central point of the subcluster m). Similar definition can be made to $P_k^{\mathrm{sc}}(n)$. Mathematically, $P_k^{\mathrm{sc}}(m)$ can be represented as

$$P_k^{\mathrm{sc}}(m) = \frac{1}{N[(\Omega_k(\mathbf{c}_m) \cap W_m]} \sum_{\mathbf{x}_i \in [\Omega_k(\mathbf{c}_m) \cap W_m]} P_k(\mathbf{x}_i)$$

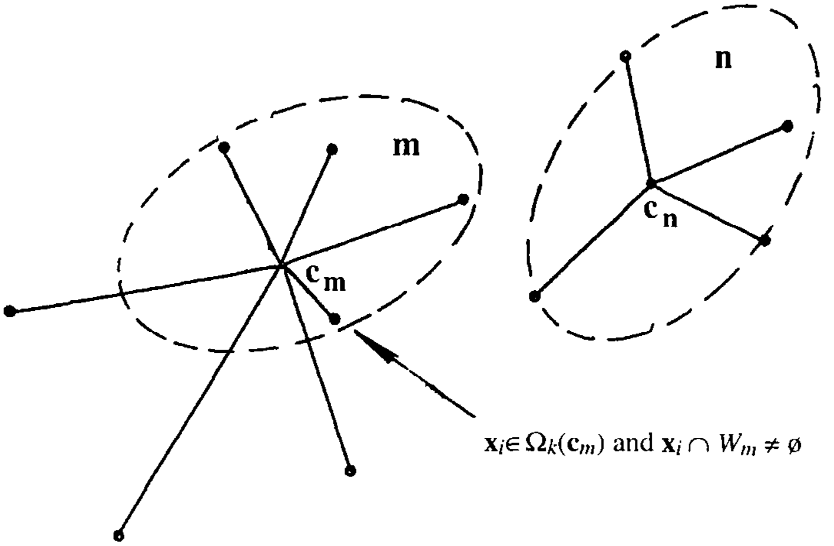


FIGURE 6.6 Pattern points that are in subcluster m and are also the k-nearest neighbors of $\mathbf{c}_m$, which is the center of subcluster m.

where W_m is a set of points contained in subcluster m, and $N(.)$ denotes the set of elements of the set in parentheses.

We can also define the k-boundary point set of subcluster m to subcluster n as $Y_k^{m,n}$, which is the set of points that are in subcluster m, but their respective k-adjacent points are in subcluster n. Or

$$Y_k^{m,n} = [\mathbf{x}_i | \mathbf{x}_i \in W_m \text{ and } \xi_k(\mathbf{x}_i) \cap W_n \neq 0]$$

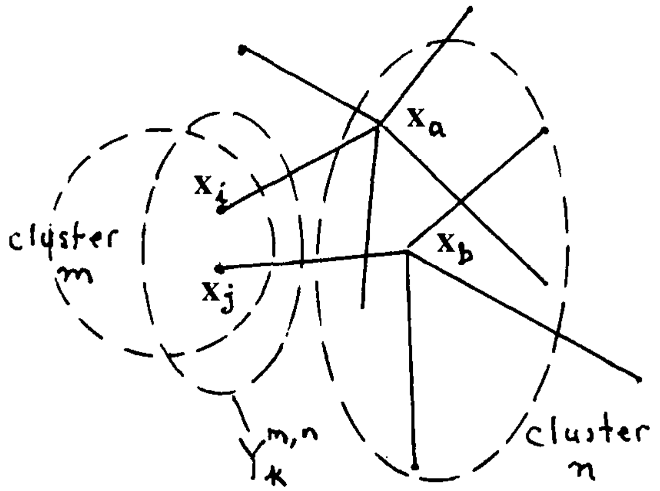


FIGURE 6.7 Pattern points that are in subcluster m and are also the k-nearest neighbors of some pattern points belonging to subcluster n.

With this $Y_k^{m,n}$, we can define $\mathrm{BP}_k^{m,n}$, the average of $P_k(\mathbf{x}_i)$ over all $\mathbf{x}_i$ in $Y_k^{m,n}$ as

$$\mathrm{BP}_k^{m,n} = \frac{1}{N(Y_k^{m,n})} \sum_{\mathbf{x}_i \in Y_k^{m,n}} P_k(\mathbf{x}_i)$$

This $\mathrm{BP}_k^{m,n}$ provides information of the density of the boundary points. The ratio of the density of the boundary points over the density of the cluster is a good measure for us to merge any two neighboring subclusters. However, it should be remembered that in the above expression, $\mathrm{BP}_k^{m,n}$ is computed in terms of distance, and therefore the smaller the computed value of the $\mathrm{BP}_k^{m,n}$, the denser will the boundary points distribution. This is also true in the computation of the density of the cluster. So, to find the ratio of the density of $\mathrm{BP}_k^{m,n}$ to that of the cluster, we should find the ratio of the numerical values of $P_k^{\mathrm{sc}}(m)$ [or $P_k^{\mathrm{sc}}(n)$] to $\mathrm{BP}_k^{m,n}$. To play safe, let us choose the smaller boundary density value from subclusters m and n, and the larger subcluster density value from subclusters m and n. Thus, we obtain the SIM_1 measure as follows:

$$\mathrm{SIM}_1(m, n) = \frac{\min[P_k^{\mathrm{sc}}(m), P_k^{\mathrm{sc}}(n)]}{\max[\mathrm{BP}_k^{m,n}, \mathrm{BP}_k^{n,m}]}$$

In brief, $\mathrm{SIM}_1(m, n)$ is a similarity measure. It is expected that this SIM_1 measure is high for the two neighboring subclusters to be merged, otherwise leave them as two separate clusters. SIM_1 measure is therefore useful to detect the valley in the pattern dispersion (see Figure 6.8).

It would be nice to take the relative size of the boundary and that of the subclusters into consideration, when we design an algorithm for the merging of two neighboring subclusters. We then have $\mathrm{SIM}_2(m, n)$, which is

$$\mathrm{SIM}_2(m, n) = \frac{N(Y_k^{m,n}) + N(Y_k^{n,m})}{2 \min[N(W_m), N(W_n)]}$$

where the numerator $N(Y_k^{m,n}) + N(Y_k^{n,m})$ represent the sum of two numbers, namely, (1) the number of those points that are in subcluster m and their "k-adjacent to them" points are in subcluster n; and (2) the number of those points that are in subcluster n and their "k-adjacent to them" points are in subcluster m. The denominator, $2 \min[N(W_m), N(W_n)]$, represents the total number of pattern points in these two neighboring subclusters. These two subclusters m and n are to be merged when SIM_2 is large. Otherwise, leave

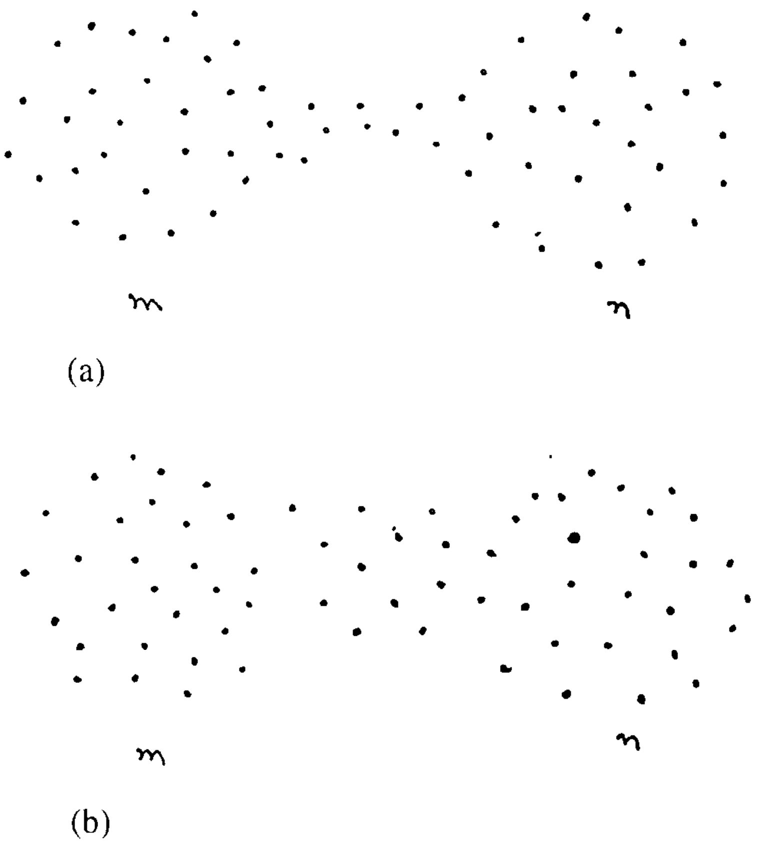


FIGURE 6.8 Set of patterns with valley and neck distribution. (a) Patterns with neck distribution. (b) Patterns with valley distribution.

them as two separate clusters. This means SIM_2 measure is useful to detect the necks between two clusters in the pattern dispersion (see Figure 6.8). Taking both factors as mentioned above into consideration, we therefore have the following $\mathrm{SIM}(m, n)$ measure:

$$\mathrm{SIM}(m, n) = \mathrm{SIM}_1(m, n) * \mathrm{SIM}_2(m, n)$$

for the merging of pairs of neighboring subclusters.

b. If no k-boundary point set exists between any pair of subclusters, use distance measure to merge subclusters p and q such that

$$D(p, q) = \min_{(m,n)\in\varphi} d(m, n)$$

where ϕ denotes a set of unordered pairs of subclusters.

6.3 CLUSTERING WITH A KNOWN NUMBER OF CLASSES

In this section we assume that the number of classes in the image is known or that at least a rough idea of the number and locations of clusters is available. Several algorithms will be introduced.

6.3.1 Minimization of Sum of Squared Distance

This algorithm is based on the minimization of the sum of squared distances from all points in a cluster domain to the cluster center, that is,

$$\min \sum_{\mathbf{x} \in S_j(k)} (\mathbf{x} - \mathbf{z}_j)^2 \tag{6.34}$$

where $S_j(k)$ is the cluster domain for cluster center $\mathbf{z}_j$ at the kth iteration. The clustering procedure of this algorithm can be illustrated by means of an example as shown on Figure 6.9. For clarity 20 two-dimensional pattern samples are considered in this example.

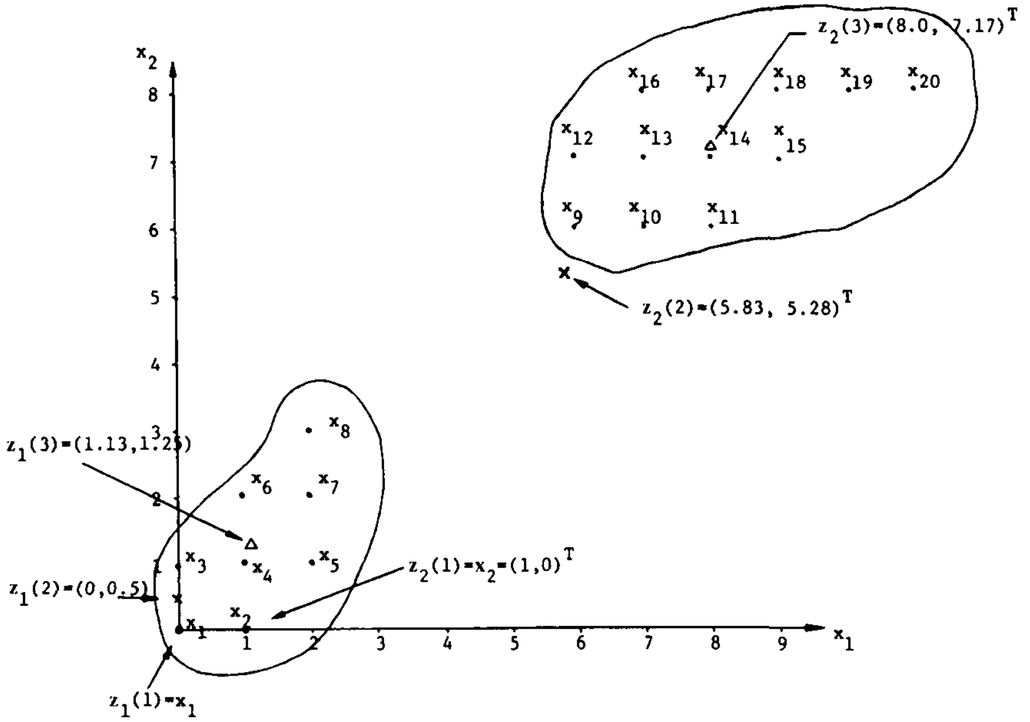


FIGURE 6.9 Illustrative example for the K-means algorithm.

1. Arbitrarily choose two samples as the initial cluster centers.

$$\begin{aligned}\mathbf{z}_1(1) &= \mathbf{x}_1 = (0, 0)^T \\ \mathbf{z}_2(1) &= \mathbf{x}_2 = (1, 0)^T\end{aligned} \tag{6.35}$$

The number inside the brackets indexes the iteration order.

2. Distribute the pattern samples $\mathbf{x}$ among the chosen cluster domains according to the following rule:

$$\mathbf{x}_1 \in S_1(1) \qquad \text{since } |\mathbf{x}_1 - \mathbf{z}_1(1)| < |\mathbf{x}_1 - \mathbf{z}_i(1)| \quad \forall i, i = 1, 2, \ldots, K, i \neq 1 \tag{6.36}$$

$$\mathbf{x}_4 \in S_2(1) \qquad \text{since } |\mathbf{x}_4 - \mathbf{z}_2(1)| < |\mathbf{x}_4 - \mathbf{z}_i(1)| \quad \forall i, i = 1, 2, \ldots, K, i \neq 2 \tag{6.37}$$

where $K = 2$ in this case. Therefore,

$$\begin{aligned}S_1(1) &= [\mathbf{x}_1, \mathbf{x}_3] \\ S_2(1) &= [\mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_5, \ldots, x_{20}]\end{aligned}$$

3. Update the cluster centers with Eq. (6.38):

$$\mathbf{z}_j(k+1) = \frac{1}{N_j} \sum_{\mathbf{x} \in S_j(k)} \mathbf{x} \qquad j = 1, 2, \ldots, K \tag{6.38}$$

or

$$\mathbf{z}_1(2) = \frac{1}{N_1} \sum_{\mathbf{x} \in S_1(1)} \mathbf{x} = \tfrac{1}{2}(\mathbf{x}_1 + \mathbf{x}_3) = \begin{pmatrix} 0.0 \\ 0.5 \end{pmatrix}$$

and

$$\mathbf{z}_2(2) = \frac{1}{N_2} \sum_{\mathbf{x} \in S_2(1)} \mathbf{x} = \tfrac{1}{18}(\mathbf{x}_2 + \mathbf{x}_4 + \cdots + \mathbf{x}_{20}) = \begin{pmatrix} 5.83 \\ 5.28 \end{pmatrix}$$

Note that these adjusted cluster centers, which are the means of all the pattern samples in their respective cluster domains, will minimize the sum of squared distances from all points in $S_j(k)$ to the new cluster centers. In this case, $j = 2$, $k = 2$. N_1 and N_2 are respectively the number of samples in $S_1(1)$ and $S_2(1)$.

4. Since $\mathbf{z}_j(2) \neq \mathbf{z}_j(1)$, $j = 1, 2$, the algorithm has not converged. We return to step 2 and repeat the process. Otherwise, the procedure is terminated.

5. With the new cluster centers, we obtain

$$|\mathbf{x}_l - \mathbf{z}_1(2)| < |\mathbf{x}_l - \mathbf{z}_2(2)| \qquad \text{for } l = 1, 2, \ldots, 8$$

and

$$|\mathbf{x}_l - \mathbf{z}_2(2)| < |\mathbf{x}_l - \mathbf{z}_1(2)| \qquad \text{for } l = 9, 10, \ldots, 20$$

Cluster domain $S_1(2)$ and $S_2(2)$ are, respectively,

$$S_1(2) = [\mathbf{x}_1, \mathbf{x}_2, \ldots, x_8]$$
$$S_2(2) = [\mathbf{x}_9, \mathbf{x}_{10}, \ldots, \mathbf{x}_{20}]$$

6. Update the cluster centers:

$$\mathbf{z}_1(3) = \frac{1}{N_1} \sum_{\mathbf{x} \in S_1(2)} \mathbf{x} = \tfrac{1}{8}(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_8) = \begin{pmatrix} 1.13 \\ 1.25 \end{pmatrix}$$
$$\mathbf{z}_2(3) = \frac{1}{N_2} \sum_{\mathbf{x} \in S_2(2)} \mathbf{x} = \tfrac{1}{12}(\mathbf{x}_9 + \mathbf{x}_{10} + \cdots + \mathbf{x}_{20}) = \begin{pmatrix} 8.00 \\ 7.17 \end{pmatrix}$$

7. Return to step 2, since

$$\mathbf{z}_j(3) \neq \mathbf{z}_j(2) \qquad j = 1, 2$$

8. Yields the same results as in the previous iteration:

$$S_1(3) = S_1(2) \qquad \text{and} \qquad S_2(3) = S_2(2)$$

9. Since $\mathbf{z}_j(4) = \mathbf{z}_j(3), j = 1, 2$, the algorithm has converged. The cluster centers we finally obtain are

$$\mathbf{z}_1 = \begin{pmatrix} 1.13 \\ 1.25 \end{pmatrix} \qquad \mathbf{z}_2 = \begin{pmatrix} 8.00 \\ 7.17 \end{pmatrix}$$

From the procedure listed above it is not difficult to see that the cluster centers are sequentially updated. This is why the term *K-means* is sometimes used for this algorithm. It is also not difficult to see that the performance of this K-means algorithm is influenced by the number of cluster centers initially chosen and also by the order in which pattern samples are passed through to the system. It is also influenced by the geometrical properties of the data to be analyzed.

6.3.2 ISODATA Algorithm

ISODATA is an acronym for *I*terative *S*elf-*O*rganizing *D*ata *A*nalysis *T*echniques *A* (the *A* being added to make the word pronounciable). In this algorithm, several process parameters are to be specified:

M = number of clusters desired
η = minimum number of samples desired in a cluster
σ_s = maximum standard deviation allowed in our problem
δ = minimum distance required between clusters

L = maximum number of pairs of cluster centers that can be lumped
I = number of iterations allowed

The procedure of this algorithm can be generalized as follows:

1. Choose some initial cluster centers.
2. Assign patterns to their nearest cluster centers.
3. Recompute the cluster centers (take the average of the samples in their domains as their new cluster centers).
4. Check and see if any cluster does not have enough members. If so, discard that cluster.
5. Compute the standard deviation for each cluster domain and see if it is greater than the maximum value allowed. If so, and if it is also found that the average distance of the samples in cluster domain S_j from their corresponding cluster center is greater than the overall average distance of the samples from their respective cluster centers, then split that cluster into two.
6. Compute the pairwise distances among all cluster centers. If some of them are smaller than the minimum distance allowed, combine that pair of clusters into one according to some suggested rule.

The whole procedure can be depicted with a flow diagram as shown in Figure 6.10. Explanations for the terms used in the figure are as follows:

$\mathbf{x}$ = pattern samples
S_i = cluster domain
$\mathbf{z}_j$ = cluster center
N_i = number of samples in S_i
N_c = arbitrarily chosen initial number of cluster centers
N = total number of samples
$\bar{D}_i$ = average distance of samples in cluster domain S_i from $\mathbf{z}_i$
$\bar{D}$ = overall average distance of samples from their respective cluster centers
$\mathbf{z}_{il}, \mathbf{z}_{jl}$ = cluster centers to be lumped
N_{il}, N_{jl} = number of samples in clusters $\mathbf{z}_{il}, \mathbf{z}_{jl}$

Example. Apply the ISODATA algorithm to the problem of 20 two-dimensional pattern samples discussed in Section 6.3.1.

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
(0, 0)	(1, 0)	(0, 1)	(1, 1)	(2, 1)	(1, 2)	(2, 2)	(2, 3)	(6, 6)	(7, 6)
$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$	$\mathbf{x}_{19}$	$\mathbf{x}_{20}$
(8, 6)	(6, 7)	(7, 7)	(8, 7)	(9, 7)	(7, 8)	(8, 8)	(9, 8)	(10, 8)	(11, 8)

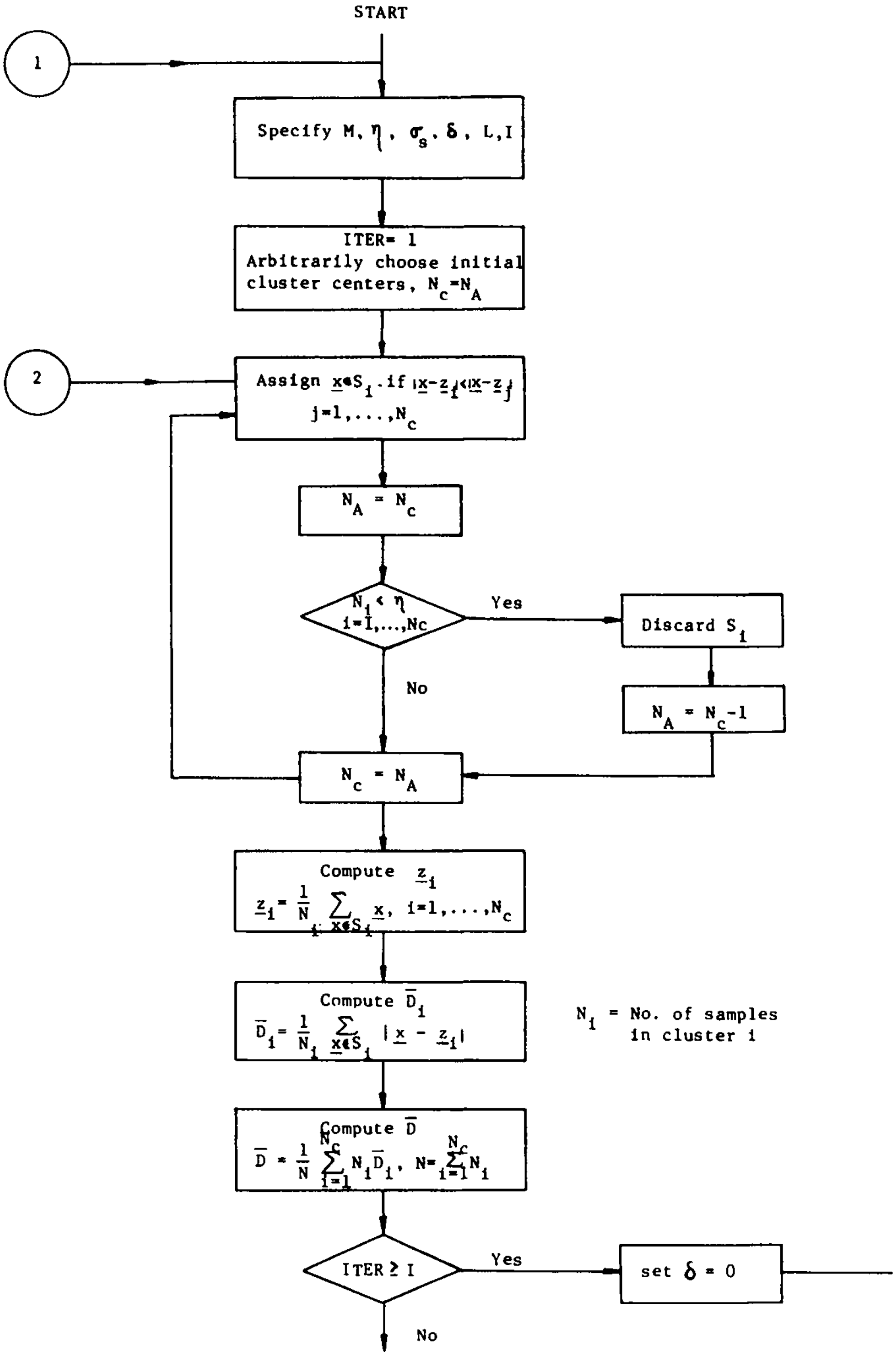

START
1
Specify $M, \eta, \sigma_s, \delta, L, I$
ITER= 1
Arbitrarily choose initial cluster centers, $N_c = N_A$
2
Assign $\underline{x} \in S_i$, if $|\underline{x}-\underline{z}_i| < |\underline{x}-\underline{z}_j|$ $j=1,\ldots,N_c$
$N_A = N_c$
$N_i < \eta$ $i=1,\ldots,N_c$
Yes
Discard S_i
$N_A = N_c - 1$
No
$N_c = N_A$
Compute $\underline{z}_i$
$\underline{z}_i = \frac{1}{N_i} \sum_{\underline{x} \in S_i} \underline{x}, \quad i=1,\ldots,N_c$
Compute $\overline{D}_i$
$\overline{D}_i = \frac{1}{N_i} \sum_{\underline{x} \in S_i} |\underline{x} - \underline{z}_i|$
N_i = No. of samples in cluster i
Compute $\overline{D}$
$\overline{D} = \frac{1}{N} \sum_{i=1}^{N_c} N_i \overline{D}_i, \quad N = \sum_{i=1}^{N_c} N_i$
ITER ≥ I
Yes
set $\delta = 0$
No

FIGURE 6.10 Flow diagram for the ISODATA algorithm.

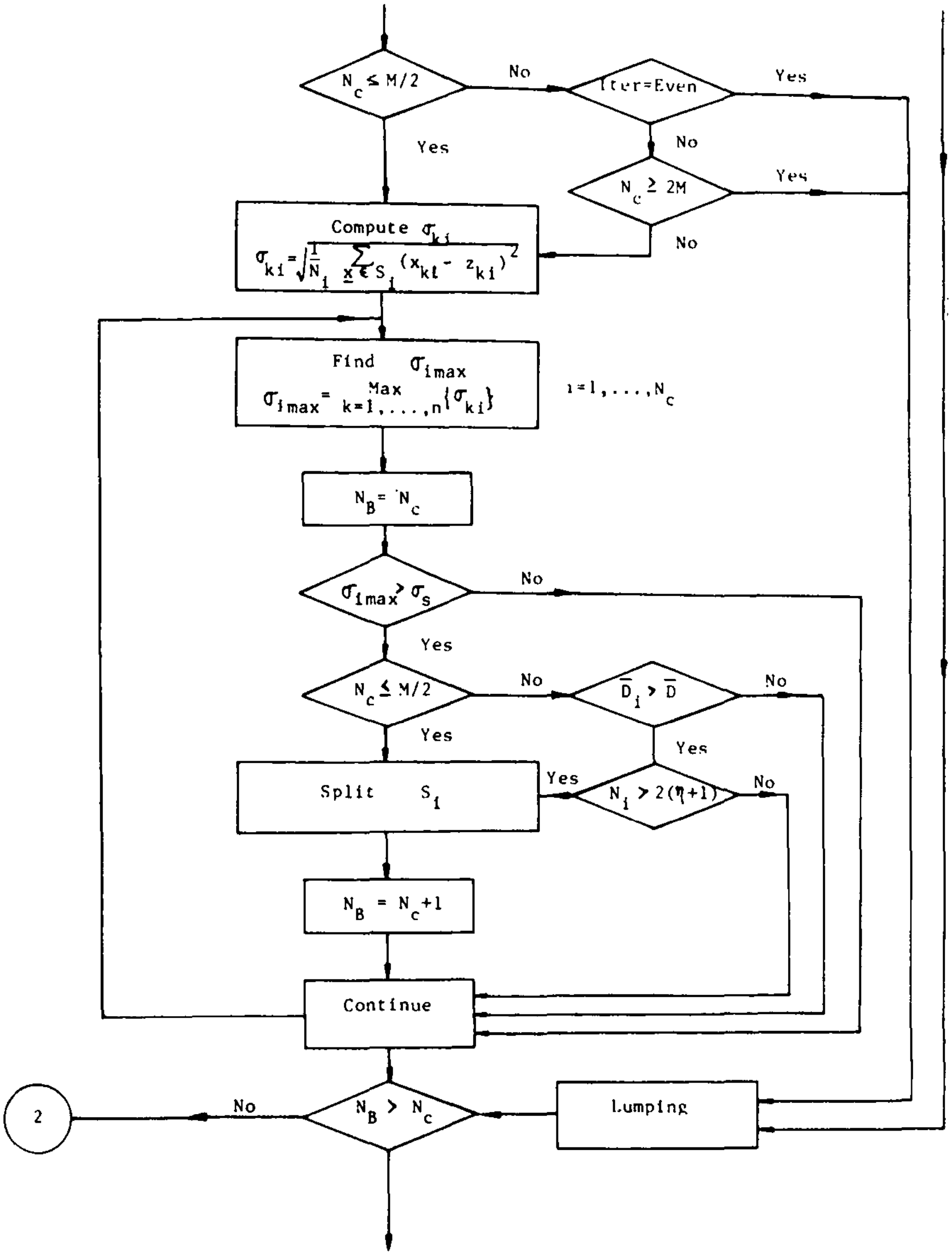


FIGURE 6.10 *Continued.*

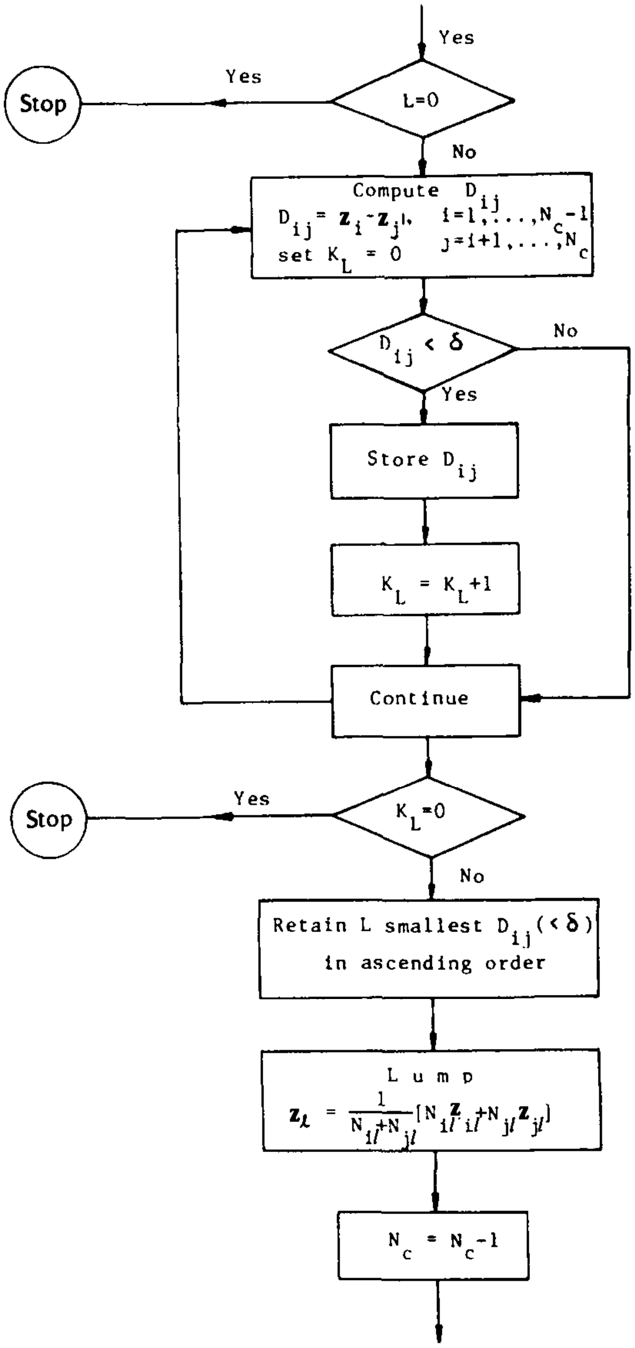

Yes
Yes
Stop
L=0
No
Compute D_{ij}
$D_{ij} = |\mathbf{z}_i - \mathbf{z}_j|$, $i=1,\dots,N_c-1$
set $K_L = 0$ $\quad j=i+1,\dots,N_c$
$D_{ij} < \delta$
No
Yes
Store D_{ij}
$K_L = K_L+1$
Continue
Stop
Yes
$K_L=0$
No
Retain L smallest $D_{ij}(<\delta)$ in ascending order
Lump
$\mathbf{z}_l = \frac{1}{N_{il}+N_{jl}}[N_{il}\mathbf{z}_{il}+N_{jl}\mathbf{z}_{jl}]$
$N_c = N_c - 1$

FIGURE 6.10 *Continued.*

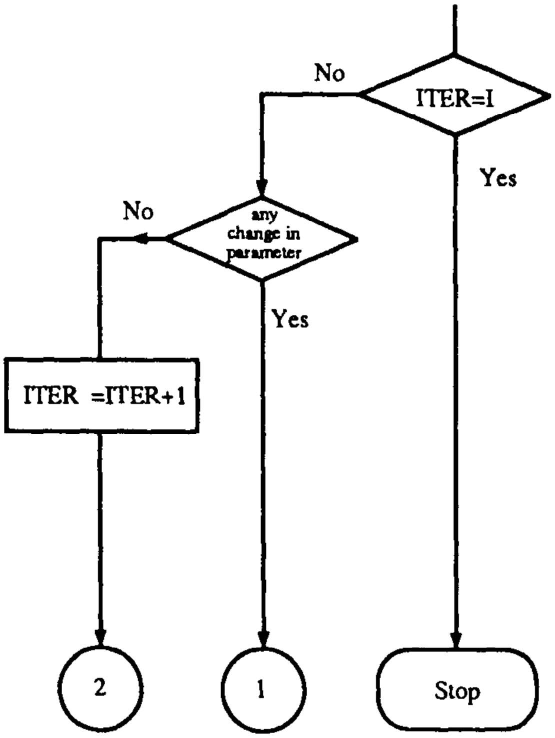


FIGURE 6.10 *Continued.*

Specify the following specified process parameters:

$$M = 2 \qquad \delta = 4$$
$$\eta = 1 \qquad L = 0$$
$$\sigma_s = 1.5 \qquad I = 4$$

In this case, $N = 20$ and $n = 2$. To start with, let $N_c = 1$, with the initial cluster center being $\mathbf{z}_1 = (0, 0)^T$. Since there is only one cluster center,

$$S_1 = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{20}]$$

and $N_1 = 20$. Since $N_1 > \eta$, no subsets are discarded.

Update the cluster centers:

$$\mathbf{z}'_1 = \frac{1}{20} \sum_{\mathbf{x} \in S_1} \mathbf{x} = (5.25, 4.8)^T$$

Compute $\bar{D}_i$:

$$\bar{D}_1 = \frac{1}{N_1} \sum_{\mathbf{x} \in S_1} |\mathbf{x} - \mathbf{z}'_1| = 4.42$$

Compute $\bar{D}$. In this case,

$$\bar{D} = \bar{D}_1 = 4.42$$

Since this is not the last iteration and $N = M/2$, find the standard deviation vector:

$$\sigma_1 = (\sigma_{11}, \sigma_{21})^T$$

$$\sigma_{11} = \left[\frac{1}{N_1}\sum_{\mathbf{x}\in S_1}(x_{1l} - z_{11})^{1/2}\right] = 3.59$$

$$\sigma_{21} = \left[\frac{1}{N_1}\sum_{\mathbf{x}\in S_1}(x_{2l} - z_{21})^2\right]^{1/2} = 3.02$$

$$\sigma_1 = (3.59, 3.02)^T \qquad \sigma_{1\max} = 3.59$$

Since $\sigma_{1\max} > \sigma_s$ and $N_c = M/2$, split $\mathbf{z}_1$. *Since* $\sigma_{1\max} = \sigma_{11}$, split along the first component of $\mathbf{z}_1$ (let $\gamma = 0.6$, $\gamma\sigma_{1\max} = 2.15$).

$$\mathbf{z}_1'^{+} = ((5.25 + 2.15), 4.8)^T = (7.40, 4.8)^T$$

$$\mathbf{z}_1'^{-} = ((5.25 - 2.15), 4.8)^T = (3.1, 4.8)^T$$

$$N_c = 2$$

Computing the distance from each sample to the two cluster centers yields the following sample sets:

$$S_1 = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8]$$

$$S_2 = [\mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{15}, \mathbf{x}_{16}, \mathbf{x}_{17}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}]$$

$$N_1 = 8$$

$$N_2 = 12$$

Since both N_1 and N_2 are greater than η, no subsets are discarded.

Update the cluster centers:

$$\mathbf{z}_1 = \frac{1}{N_1}\sum_{\mathbf{x}\in S_1}\mathbf{x} = (1.125, 1.25)^T$$

$$\mathbf{z}_2 = \frac{1}{N_2}\sum_{\mathbf{x}\in S_2}\mathbf{x} = (8.00, 7.17)^T$$

Compute $\bar{D}_1$ and $\bar{D}_2$:

$$\bar{D}_1 = \frac{1}{N_1}\sum_{\mathbf{x}\in S_1}|\mathbf{x} - \mathbf{z}_1| = 1.14$$

$$\bar{D}_2 = \frac{1}{N_2}\sum_{\mathbf{x}\in S_2}|\mathbf{x} - \mathbf{z}_2| = 1.49$$

Compute $\bar{D}$:

$$\bar{D} = \frac{1}{N}\sum_{j=1}^{N_c} N_j \bar{D}_j = \frac{1}{20}\sum_{j=1}^{2} N_j \bar{D}_j = 1.35$$

This is an even-numbered iteration. Compute D_{12}:

$$D_{12} = |\mathbf{z}_1 - \mathbf{z}_2| = 9.07$$

Since $L = 0$, no action is taken. Since the requested number of clusters is satisfied, and the distance between the clusters are large relative to the minimum distance required, there is no need to change parameters.

Distribute the samples again to the two-cluster center according to distance measure. The same results are obtained.

$$\boldsymbol{\sigma}_1 = (\sigma_{11}, \sigma_{21})^T$$

$$\sigma_{11} = \left[\frac{1}{N_1}\sum_{\mathbf{x}\in S_1} (\mathbf{x}_{1l} - z_{11})^2\right]^{1/2} = 0.78$$

$$\sigma_{21} = 0.96$$

$$\boldsymbol{\sigma}_1 = (0.78, 0.96)^T$$

$$\sigma_{12} = \left[\frac{1}{N_2}\sum_{\mathbf{x}\in S_2} (x_{1l} - z_{12})^2\right]^{1/2}$$

$$\sigma_{22} = \left[\frac{1}{N_2}\sum_{\mathbf{x}\in S_2} (x_{2l} - z_{22})^2\right]^{1/2} = 0.8$$

$$\boldsymbol{\sigma}_2 = (1.47, 0.8)^T$$

$$\sigma_{1\max} = 0.96 < \sigma_s$$

$$\sigma_{2\max} = 1.47 < \sigma_s$$

$N_c \geq M/2$; therefore, no splitting takes place.

The final results as shown in Figure 6.11 are

$$\mathbf{z}_1 = (1.125, 1.25)^T$$

$$\mathbf{z}_2 = (8.00, 7.17)^T$$

$$S_1 = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8]$$

$$S_2 = [\mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{15}, \mathbf{x}_{16}, \mathbf{x}_{17}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}]$$

Note that these results check with those obtained by the minimization-of-sum-of-squared-distances method discussed in Section 6.3.1. But if the σ_s is set at a small value and M is set at higher value, say 3, then $\mathbf{z}_2$ can be further split into two clusters.

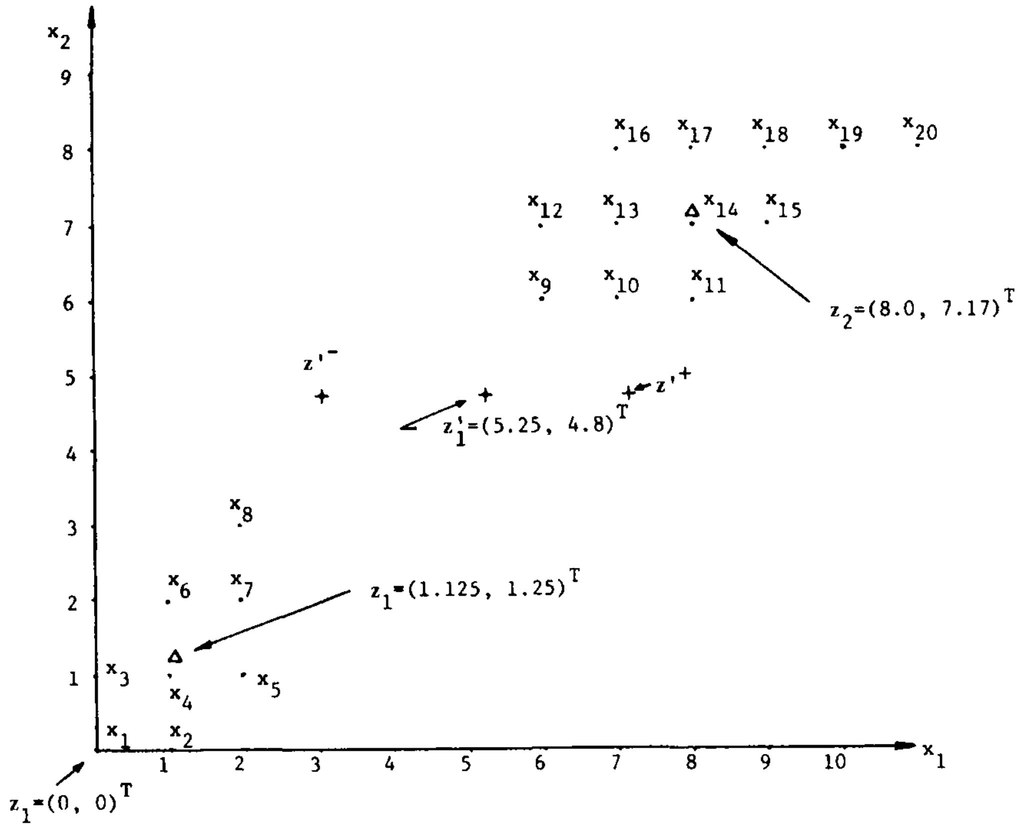


FIGURE 6.11 Example of the ISODATA method.

6.3.3 Modification of the ISODATA Algorithm (Without Human Intervention in Specifying Certain Process Parameters)

As can be seen from Section 6.3.2, in using the ISODATA algorithm certain process parameters have to be specified, such as the number of clusters desired, the minimum acceptable standard deviation, and the minimum acceptable distance between clusters. Knowledge of those parameters presumes that previous studies have been done on the data. In addition, the performance of the algorithm is highly dependent on the various parameters preset by the user. The "proper" setting usually can be determined only by a trial-and-error method.

Davies and Bouldin (1979) suggested a clustering parameter which is to be minimized to obtain natural partitions of the data sets, i.e, to obtain an optimum number of cluster. The parameter they used is

$$R_{ij} = \frac{D_{ii} + D_{jj}}{D_{ij}} \tag{6.39}$$

where D_{ii} and D_{jj} are defined as the dispersions for clusters i and j, respectively, and D_{ij} is the distance between clusters i and j. It is obvious from the definitions that if $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \in S_p$, the cluster domain, then

$$D_{ii}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \geq 0 \tag{6.40}$$

$$D_{ii}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = 0 \qquad \text{iff } \mathbf{x}_i = \mathbf{x}_j, \forall \mathbf{x}_i, \mathbf{x}_j \in S_p \tag{6.41}$$

Some limitations on R to make it meaningful are:

$$\begin{aligned}
&(1) \quad R_{ij}(D_{ii}, D_{jj}, D_{ij}) \geq 0\\
&(2) \quad R_{ij}(D_{ii}, D_{jj}, D_{ij}) = R_{ji}(D_{jj}, D_{ii}, D_{ji})\\
&(3) \quad R_{ij}(D_{ii}, D_{jj}, D_{ij}) = 0 \qquad \text{iff } D_{ii} = D_{jj} = 0\\
&(4) \quad \text{If } D_{jj} = D_{kk} \text{ and } D_{ij} < D_{ik}, \text{ then } R_{ij}(D_{ii}, D_{jj}, D_{ij}) > R_{ik}(D_{ii}, D_{kk}, D_{ik})\\
&(5) \quad \text{If } D_{ij} = D_{ik} \text{ and } D_{jj} \geq D_{kk}, \text{ then } R_{ij}(D_{ii}, D_{jj}, D_{ij}) > R_{ik}(D_{ii}, D_{kk}, D_{ik})
\end{aligned} \tag{6.42}$$

The first and second expressions indicate that the similarity function R is nonnegative and possesses the property of symmetry. The third expression implies that the similarity between clusters is zero if and only if their dispersions equal zero.

The fourth and fifth expressions imply that if the interset distance between clusters increases while their dispersions remain constant, the similarity of the cluster decreases. On the contrary, if the interest distance remains constant, the similarity of clusters increases when the dispersions increase.

To start with, the initial number of clusters can be chosen as a large number; it can even be chosen as large as the number of patterns in a given data set. By following the ISODATA algorithm as discussed in Section 6.3.2, we can obtain the location of clusters as well as the pattern samples allocated to each cluster for the number of clusters chosen. D_{ii} and D_{jj} can be computed according to the definition as

$$D_{ii} = \left[\frac{1}{N_i} \sum_{j=1}^{N_i} |\mathbf{x}_j - \mathbf{z}_i| \right]^{1/2}$$

and

$$D_{ij} = \left[\sum_{k=1}^{n} |z_{ki} - z_{kj}|^2 \right]^{1/2}$$

where z_{ki} is the kth component of cluster i. R_{ij} can be computed according to

$$R_{ij} = \frac{D_{ii} + D_{jj}}{D_{ij}}$$

Let N_c be the number of clusters chosen. For a specific value of i, with $j = 1, 2, \ldots, N_c, j \neq i$, there will be $(N_c - 1)$ R_{ij}'s, or N_c R_{ij}'s (one of which is R_{ii}). Denote R_i as the maximum one among these R_{ij}, i.e.,

$$R_i = \max_{\substack{j=1,2,\ldots,N_c \\ j \neq i}} R_{ij}$$

and $\bar{R}$ as the average of the similarity measures of each cluster with its most similar cluster. $\bar{R}$ can then be computed as

$$\bar{R} = \frac{1}{N_c} \sum_{i=1}^{N_c} R_i$$

Different values of $\bar{R}$ will result for different value of N_c (the number of clusters chosen in the course of clustering computations). The number of clusters N_c corresponding to the smallest values of $\bar{R}$ seems to be the most appropriate number of clusters. Unfortunately, there is no easy way to find the most appropriate number of clusters analytically. Nevertheless, we can plot $\bar{R}$ for different number of clusters chosen in the course of clustering computation versus the number of clusters N_c, and look for the appropriate number of clusters N_c to give the minimum value of $\bar{R}$.

Figure 6.12a and c show, respectively, a data set of 225 points for test, and the performance of $\bar{R}$ for the smallest 20 values of N_c chosen. $\bar{R}$ is minimal when $N_c = 8$ and about 5% greater than the minimum when $N_c = 9$.

Figure 6.12b shows that the data points are grouped into seven clusters as shown by the partitioning, and into eight as indicated by the additional dashed line.

6.3.4 Dynamic Optimal Cluster Seeking Technique (DYNOC)

The DYNOC is an algorithm suggested by Tou to circumvent the shortcomings mentioned earlier. The main point of this algorithm is to introduce a performance index

$$\lambda(N_c) = \frac{\min\{D_{ij}\}}{\max\{D_{jj}\}} \tag{6.47}$$

to determine the optimal clusters. Optimal clusters occur when the performance index $\lambda(N_c)$ reaches a peak, and N_c is optimal if $\lambda(N_c)$ is a global maximum. The maximization of this performance index can be inserted into any of the algorithms, such as the maximum distance algorithm (the K-means algorithm) and the ISODATA algorithm, immediately before splitting of a king size cluster and/or merging of small clusters take place.

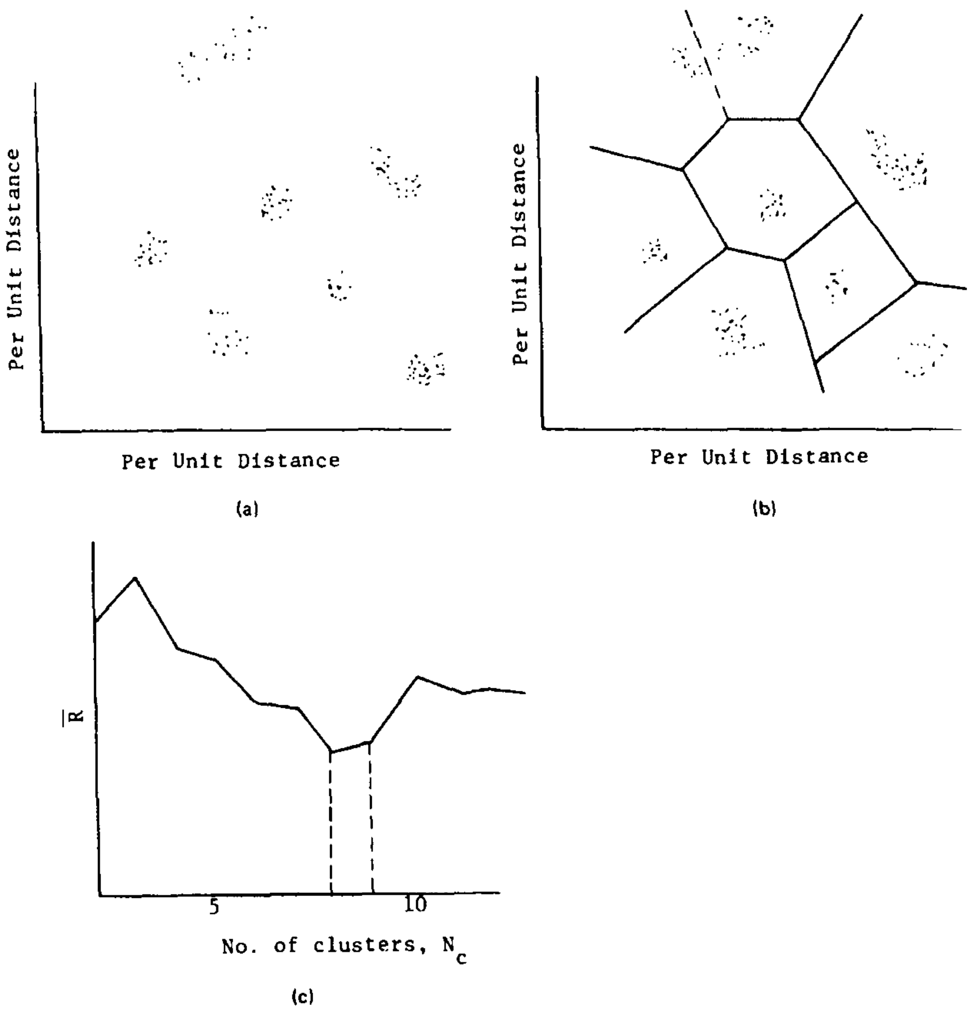


FIGURE 6.12 Illustrative example for the modification of the ISODATA algorithm: (a) a data set of 225 points; (b) clustering of the 225 points into seven or eight groups; (c) $\bar{R}$ versus N_c plot.

6.3.5 Dynamic Clusters Method in Nonhierarchical Clustering

In the methods discussed previously, the cluster center was represented by a simple representative point. In this section we introduce another method, called by Diday the dynamic clusters method, in which a cluster is represented by several representative points called *multicenters* or *sampling*. A Q function is used for determination of the centering. This algorithm can be stated briefly as follows. Given M, the number of clusters, and N_i, the number of pattern points in E_i, $i = 1, 2, \ldots, M$, with $S = (S_1, \ldots, S_M)$ as the M-cluster domains of E and

$E = (E_1, \ldots, E_M)$ as the M sampling of S. Clustering problem is then to find the pair (E, S) that minimizes

$$\Delta(E, S) = \sum_i D(E_i, S_i) = \sum_i \sum_{\mathbf{x} \in S_i} \sum_{\mathbf{z} \in E_i} d(\mathbf{x}, \mathbf{z}) \tag{6.48}$$

where $E_i \subset E$, $i = 1, 2, \ldots, M$, are called sampling or multiple centers or cores; S_i, $i = 1, 2, \ldots M$, are cluster domains with the property $S_i \cap S_j = \emptyset$; $D(E_i, S_i)$ is the "degree of similarity" of E_i to S_i; $d(\mathbf{x}, \mathbf{z})$, called the *intraset distance*, applies not to a single cluster center, but to a multiple center to core as shown in Figure 6.13. $S = (S_1, S_2, \ldots, S_M)$ is achieved such that S_i are formed from the set of elements $\mathbf{x}$ such that $D(\mathbf{x}, E_i^{(0)}) \leq D(\mathbf{x}, E_j^{(0)})$. New samplings $E_i^{(1)}$ can be defined by the N_i elements of E which are closest to S_i in the sense of a certain function Q; that is, the N_i elements of E are chosen such that the following function Q is minimized:

$$Q = \frac{D(\mathbf{x}, E_i)}{\sum_j D(\mathbf{x}, E_j)} \tag{6.49}$$

The choice of the Q function is important in this algorithm. With a good choice of Q, the convergence is generally achieved in about five iterations.

The advantage of this algorithm is that by using N_i multiple centers instead of only the center of gravity, the real form (may be the elongated form) would have been obtainable. If only the center of gravity is used, the recognized form would have been "rounded up."

Use the same example as that used in previous sections to illustrate this algorithm (see Figure 6.14).

1. Find $S_1^{(0)}$ and $S_2^{(0)}$ by distance measure.

$$S_1^{(0)} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8]$$
$$S_2^{(0)} = [\mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{15}, \mathbf{x}_{16}, \mathbf{x}_{17}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}]$$

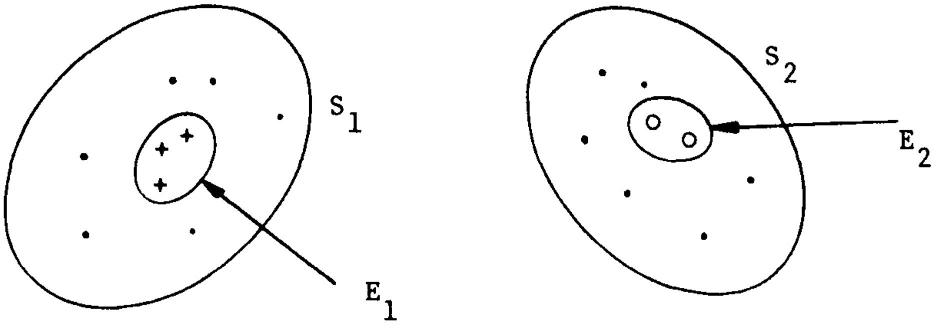


FIGURE 6.13 Multicenter representation of a cluster.

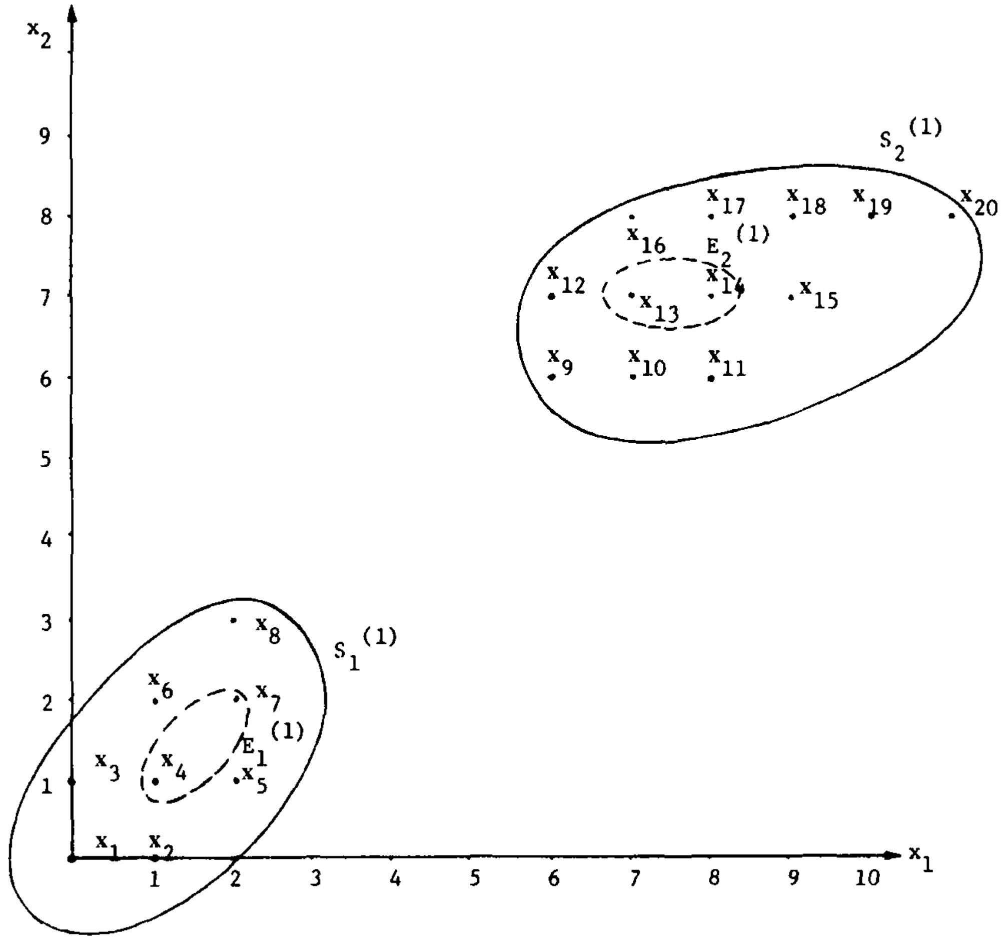


FIGURE 6.14 Same example as in Figures 6.9 and 6.10 but with multicenter representation of a cluster.

2. From x's in $S_1^{(0)}$ and x's in $S_2^{(0)}$, find $E_1^{(1)}$ and $E_2^{(1)}$ to minimize Q

$$E_1^{(1)} = [\mathbf{x}_4, \mathbf{x}_7]$$
$$E_2^{(1)} = [\mathbf{x}_{13}, \mathbf{x}_{14}]$$

3. Reassign the pattern points to get $S_1^{(1)}$ and $S_2^{(1)}$

$$S_1^{(1)} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8]$$
$$S_2^{(1)} = [\mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{15}, \mathbf{x}_{16}, \mathbf{x}_{17}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}]$$

4. Get the correct representation of sampling E_1 and E_2.

6.4 EVALUATION OF CLUSTERING RESULTS BY VARIOUS ALGORITHMS

The principal difficulty in evaluating the results of the various clustering algorithms is the inability to visualize the geometrical properties of a high-dimensional space. However, there are some measures, such as distance between cluster centers, that can be used as a tool for the valuation of the clustering results.

From the numbers shown on Table 6.1, it can be seen that $\mathbf{z}_8$ is significantly remote from the other seven cluster centers. Clusters $\mathbf{z}_1$, $\mathbf{z}_3$, and $\mathbf{z}_4$ are close together, as are the clusters $\mathbf{z}_2$ and $\mathbf{z}_6$, and $\mathbf{z}_5$ and $\mathbf{z}_7$. The number of pattern samples falling into the domain of each cluster is also an aid in interpreting the results. For the example above, if the number of samples associated with the cluster $\mathbf{z}_8$ is numerous, we will certainly accept it as a cluster center. But when the number of samples is small, cluster $\mathbf{z}_8$ can be discarded without causing too many discrepancies from the original data.

Another bit of useful information that can be used in the evaluation of clustering is the variance of each cluster domain about its mean. Variances are useful to infer the relative distribution of the samples in the domains. From the component values of a variance along the coordinate axes, we can estimate the tightness of the pattern points around the cluster as well as the shape of the cluster domain. For the cluster $\mathbf{z}_1$ shown in the variance table (Table 6.2) we can say that it has a hyperspherical shape for its domain, since σ_i^2, $i = 1, 2, 3$, are almost the same for each component. But for cluster $\mathbf{z}_4$, the shape of its cluster domain will be somewhat elongated about the third coordinate axis.

The ratio of interset distance to intraset distance is another criterion for the evaluation of clustering results. We definitely prefer a high performance value for this ratio.

Other quantitative measures of the clustering properties can be the closest and most distant points from the cluster center in each domain and the covariance

TABLE 6.1 Distances Between Cluster Centers

Cluster center	$\mathbf{z}_1$	$\mathbf{z}_2$	$\mathbf{z}_3$	$\mathbf{z}_4$	$\mathbf{z}_5$	$\mathbf{z}_6$	$\mathbf{z}_7$	$\mathbf{z}_8$
$\mathbf{z}_1$	0.0	14.8	3.5	2.1	18.0	10.0	21.0	55.6
$\mathbf{z}_2$		0.0	15.4	16.1	23.0	3.5	28.0	54.3
$\mathbf{z}_3$			0.0	5.6	21.0	12.0	19.0	52.8
$\mathbf{z}_4$				0.0	15.0	14.0	19.0	50.0
$\mathbf{z}_5$					0.0	25.0	4.0	49.3
$\mathbf{z}_6$						0.0	23.0	55.8
$\mathbf{z}_7$							0.0	48.2
$\mathbf{z}_8$								0.0

TABLE 6.2 Variances of Various Cluster Domains

	Variance		
Cluster domain	σ_1^2	σ_2^2	σ_3^2
$\mathbf{z}_1$	1.1	0.8	0.7
$\mathbf{z}_2$	1.8	1.5	1.0
$\mathbf{z}_3$	2.5	3.6	5.7
$\mathbf{z}_4$	2.5	3.8	19.5
$\mathbf{z}_5$	4.1	4.7	5.4
$\mathbf{z}_6$	3.7	3.9	5.5
$\mathbf{z}_7$	4.2	8.6	4.8

matrix of each sample set. Computational complexity and computer time are other measures for comparison.

6.5 GRAPH THEORETICAL METHODS

The disadvantage of the approaches discussed so far is that the clustering results are dependent on the presentation ordering of the pattern samples. One could argue that the clusters might be determined more accurately if all the samples were considered simultaneously. Graph theoretic approaches are suggested to meet such requirements, but at a possible increase in computational time and also at a substantial cost in rapid-access storage.

6.5.1 Similarity Matrix

The similarity matrix is such a matrix used to show the degree of similarity between a variety of pattern points. Consider that matrix as an $N \times N$ symmetric matrix whose elements are

$$s_{ij} = \begin{cases} 1 & d(\mathbf{x}_i, \mathbf{x}_j) \le \theta \\ 0 & d(\mathbf{x}_i, \mathbf{x}_j) > \theta \end{cases} \qquad i,j = 1, 2, \ldots, N \tag{6.50}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the intersample distance between pattern points $\mathbf{x}_i$ and $\mathbf{x}_j$. S is the threshold distance used to denote the similarity between the two pattern points. In other words, s_{ij} tells whether the pair of samples are closer than a distance θ. s_{ij} are binary numbers chosen such that only one bit of storage is required by $d(\mathbf{x}_i, \mathbf{x}_j)$.

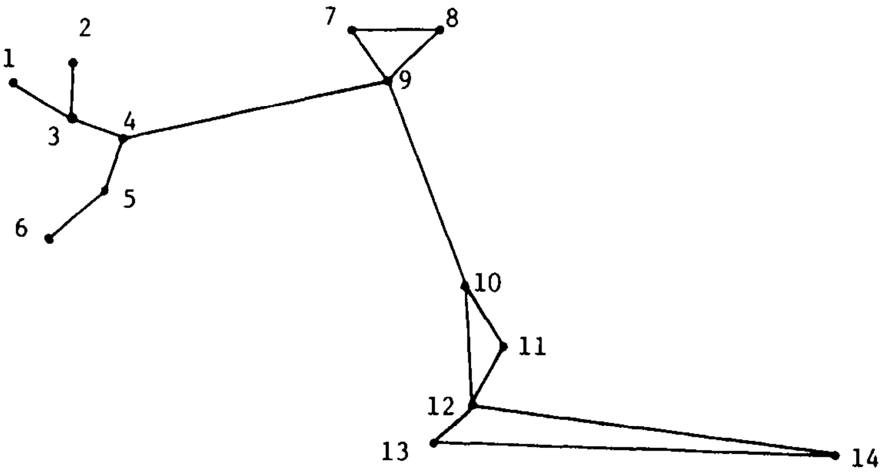


FIGURE 6.15 Illustrative example of 14 samples for similarity matrix studies.

Figure 6.15 shows a two-dimensional plot for a set of 14 samples. Its corresponding similarity matrix drawn from the two-dimensional samples above is shown below

$$S = \begin{array}{c|cccccccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array} \quad \theta = 2$$

This similarity matrix can be used for clustering. The procedure is as follows:

1. Choose the row of S with the most l's (the choice is arbitrary if there is more than one such row), say row i.
2. Form a cluster of $\mathbf{x}_i$ and all $\mathbf{x}_j$ corresponding to the l's in row i.

3. Add $\mathbf{x}_k$ to the cluster if $s_{jk} = 1$; that is, if $\mathbf{x}_j$ is already in the cluster and $d(\mathbf{x}_j, \mathbf{x}_k) \leq \theta$ (or $s_{jk} = 1$), then $\mathbf{x}_k$ should also be in the cluster even if $s_{ik} = 0$ [i.e., $d(\mathbf{x}_i, \mathbf{x}_k) > \theta$].
4. Repeat step 1 until no new $\mathbf{x}$'s can be added to the cluster.
5. Remove all columns and rows corresponding to $\mathbf{x}$'s in the cluster to form a reduced matrix.
6. Repeat steps 1 through 5 for the reduced matrices until no further reductions are possible (i.e., no more clusters can be formed).

For our example, choose $\theta = 2$.

1. Choose row 1 with three 1's in it.
2. $[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3]$ form a cluster $[s_{11} = s_{12} = s_{13} = 1]$.

3–4. Samples $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ are drawn into the cluster. Row 3 has a 1 in column 4; therefore, $\mathbf{x}_4$ is added to the cluster, resulting in the cluster $[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4]$. By the same token, $\mathbf{x}_5$ and $\mathbf{x}_6$ are also added to the cluster, resulting in a cluster consisting of $[\mathbf{x}_i]$, $i = 1, 2, \ldots, 6$.

5. The reduced matrix after removal of all columns and rows corresponding to $\mathbf{x}$'s in the cluster:

$$S' = \begin{array}{c|cccccccc|}
 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
7 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
8 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
9 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
11 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
12 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
13 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}$$

1. Choose row 7 with three 1's in that row.
2. $[\mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9]$ form a cluster.

3–4. Since there are no $s_{jk} = 1$ for $j = 9$, no $\mathbf{x}_k$ is added to this cluster.

5. The reduced matrix after removal of all columns and rows corresponding to $\mathbf{x}$'s in the cluster:

$$S'' = \begin{array}{c|ccccc|}
 & 10 & 11 & 12 & 13 & 14 \\
10 & 1 & 1 & 0 & 0 & 0 \\
11 & 1 & 1 & 1 & 0 & 0 \\
12 & 0 & 1 & 1 & 1 & 0 \\
13 & 0 & 0 & 1 & 1 & 0 \\
14 & 0 & 0 & 0 & 0 & 1
\end{array}$$

1. Choose row 11 with three 1's in that row.
2. $[\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}]$ form a cluster.
3–4. Since $s_{12,13} = 1$, $\mathbf{x}_{13}$ is added to the cluster according to procedure 3 as stated above. The cluster then consists of $[\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}]$.
5. The reduced matrix after removal of all columns and rows corresponding to $\mathbf{x}$'s in the cluster:

$$S''' = \begin{array}{c|c|} & 14 \\ 14 & 1 \end{array}$$

leaving $[\mathbf{x}_{14}]$ as the final cluster.

When θ is chosen, clusters are defined by disjoint connected subgraphs of the graph so defined. Obviously, the choice of the value of θ is critical. Assume for the example above that $\theta = 4$; then three clusters will be formed from the same data set. When the θ chosen becomes larger, say $\theta = 8$, there will be only one cluster (see Figure 6.16c). If $\theta = 1$, there will be 14 clusters with one pattern point in each cluster,

Note that in this method there are a total of N^2 elements in S and $N(N - 1)/2$ nonredundant elements (distances) in S. This may impose a severe limitation on the number of pattern points that can be examined. If we have 1000 samples, these will generate about 500,000 interpoint distances.

6.5.2 SpanningTree Methods

Minimal SpanningTree Method

The minimal spanning tree method results from a graph analysis of arbitrary point sets of data. Before our discussion on this method, let us introduce some terms that should prove useful. Given a set G of points $\mathbf{x}_i$, $i = 1, 2, \ldots, N$:

1. An *edge* is a connection between two points.
2. A *path* is a sequence of edges connecting two points.
3. A loop is a closed path.
4. A *connected graph* has one or more paths between any pair of points.
5. A *tree* is a connected graph with no closed loops.
6. A *spanning tree* is a tree that contains every point in G.
7. The *weight* of a tree is the sum of weights assigned to each edge in the tree; for example, the weight equals the distance between two points at the ends of the edge.
8. The *minimal spanning tree* (MST) is that spanning tree of minimal weight (among all possible spanning trees of G).
9. The *main diameter* is that path of the MST containing the largest number of points (formed by removing the branch points from the minimal spanning tree).

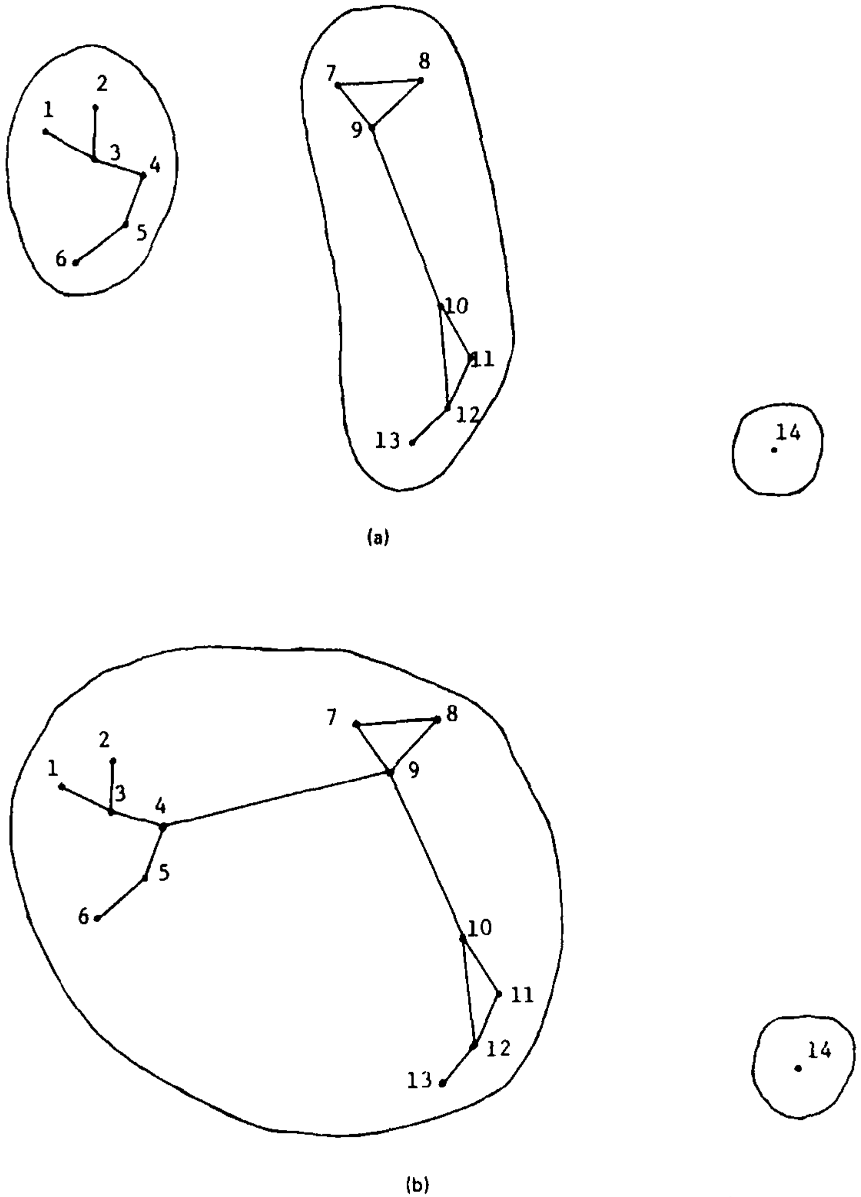
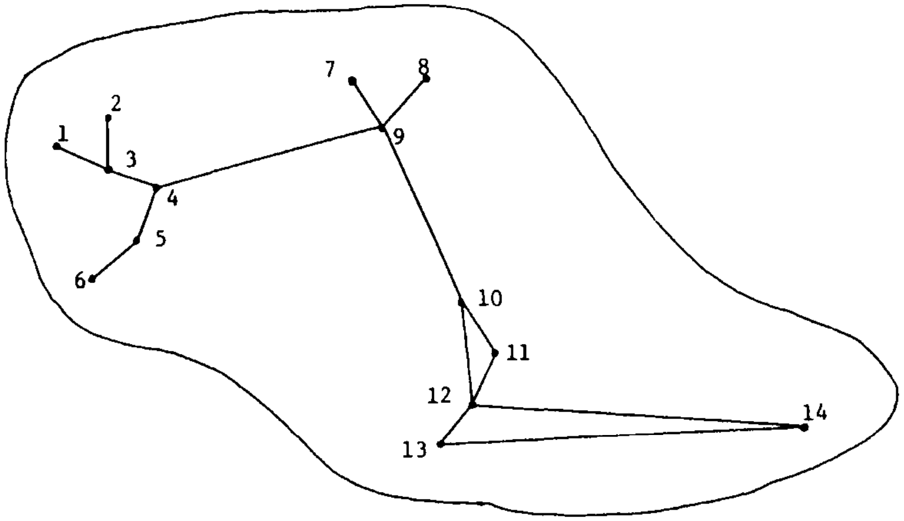


FIGURE 6.16 Effect of the choice of θ on clustering: (a) $\theta = 4$, three clusters; (b) $\theta = 5$, two clusters; (c) $\theta = 8$, one cluster.

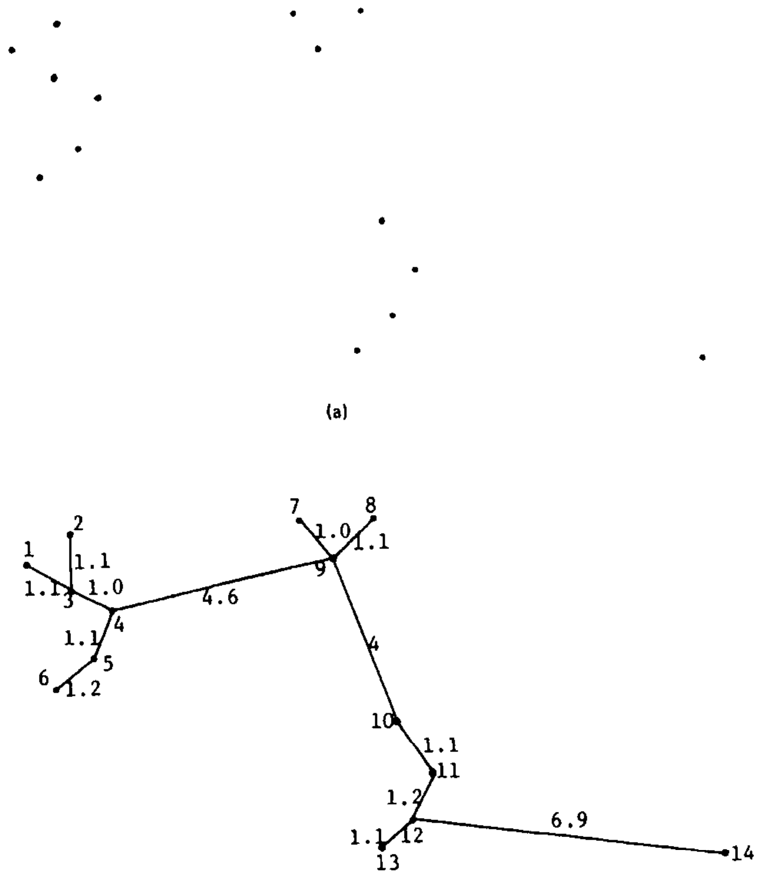
Figure 6.17 shows the minimal spanning tree and its original data. Because the minimum spanning tree is unique to a set of points in terms of a minimum total weight, it is possible to use the tree as a basis for cluster detection by combining both distance properties and density properties.

Several main diameters can be drawn. Two of them are shown in Figures 6.18 and 6.19 with their edge weight plots. From the minimal spanning tree shown in Figure 6.17 clusterings can be found by the nearest neighbor algorithm.



(c)

FIGURE 6.16 *Continued.*



(a)

(b)

FIGURE 6.17 Minimal spanning tree: (a) original data; (b) minimal spanning tree.

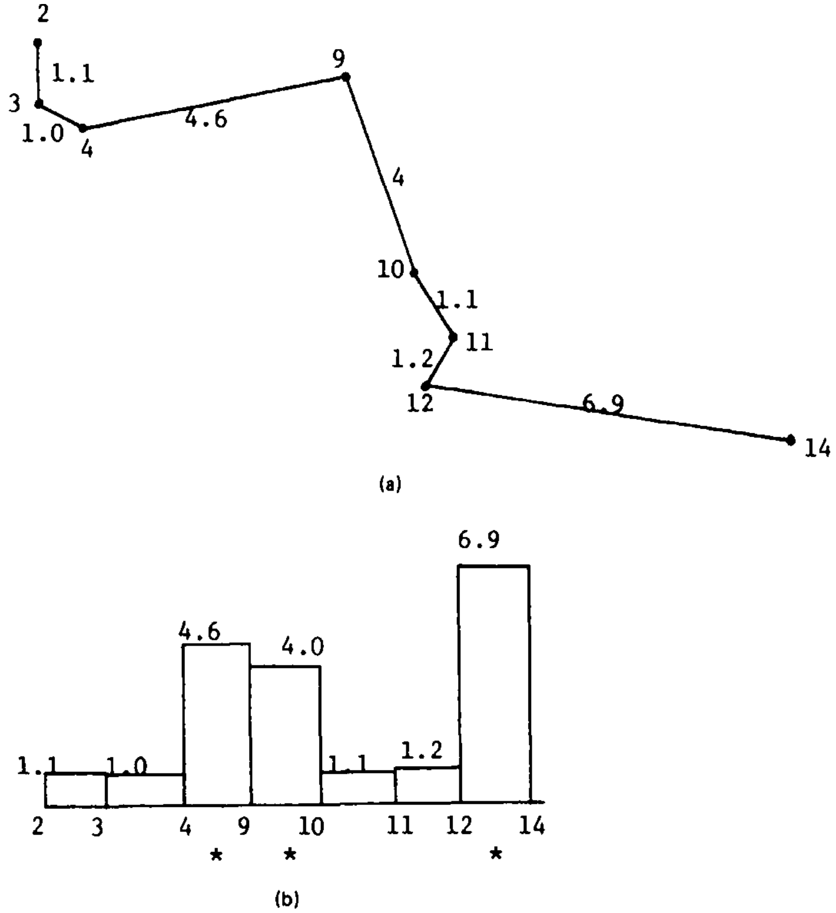


FIGURE 6.18 One of the main diameters of the MST shown in Figure 6.17. (a) Main diameter; (b) edge weight plot.

Removal of the longest edge, 12–14, produces a two-cluster grouping; further removal of the next longest edge, 4–9, produces a three-cluster grouping; and removal of all three long edges produces a four-cluster grouping. These correspond to choosing breaks where maximum weights occur in the main-diameter histogram.

Shared Near Neighbor Maximal SpanningTree for Clustering

The method by Jarvis (1974) that we describe next is one that associates the shared near neighbor rules with the spanning tree in a graph theoretic framework. In this method the influence of other points in the set is taken into consideration quantitatively on the relative similarity of each pair of pattern points. The idea behind this shared near neighbor maximal spanning tree concept is to transform

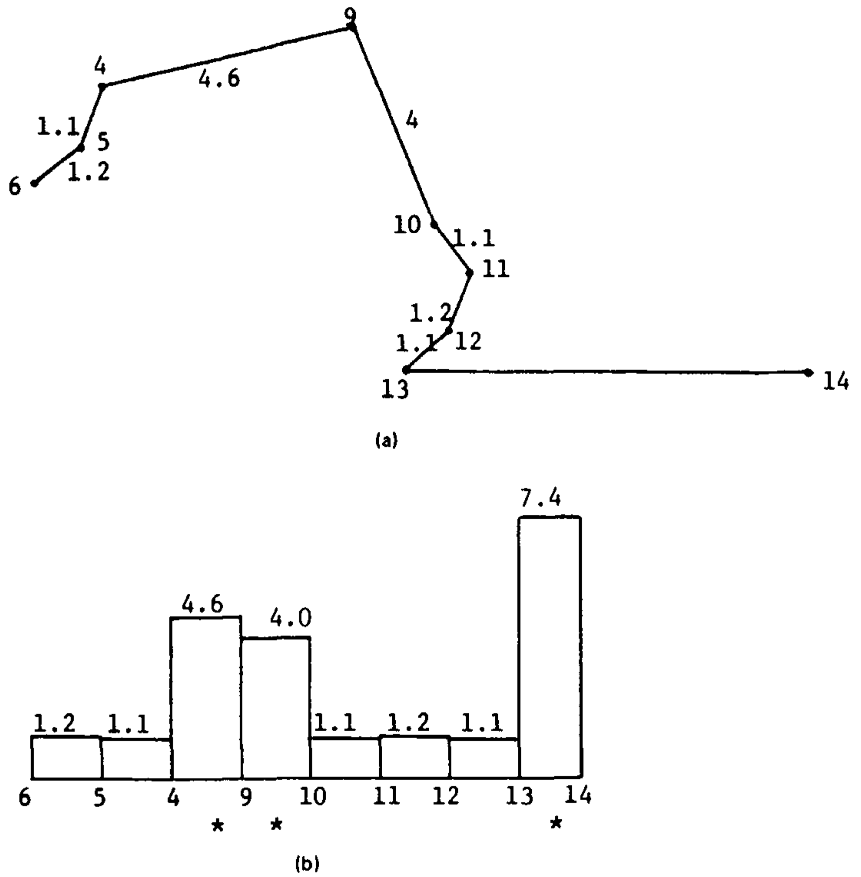


FIGURE 6.19 One of the main diameters of the MST shown in Figure 6.17. (a) Main diameter; (b) edge weight plot.

context-insensitive measures into ones that reflect an interaction of point placement relationships in the relative vicinity of the candidate pair. In this method it is presumed that pairs of points in the set are similar to the extent that they share the same near neighbors provided that each is in the defined near neighborhood of the other.

The procedure can be stated as follows:

1. List the k-nearest neighbors for each pattern point $\mathbf{x}_i$, $i = 1, 2, \ldots, N$, shown in Figure 6.20 in order of closeness, as shown in Table 6.3. The simplest euclidean distance measure can be used for this purpose. The k-nearest neighbor $N \times (k + 1)$ matrix generated is. to be used in subsequent processing.
2. Test for occurrence of the first entry of each row in the other rows of the matrix to find pairs of rows for later processing (usually not more than k rows can be found).

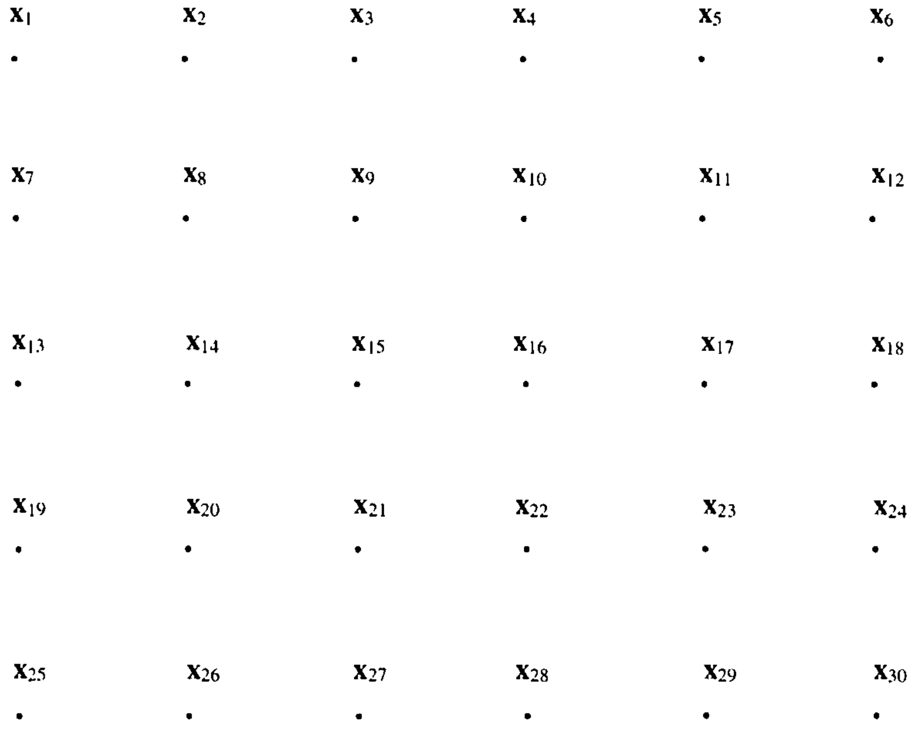


FIGURE 6.20 Simple data set used for illustration of the shared near neighbor maximal spanning tree method.

TABLE 6.3 $N \times (k + 1)$ Nearest Neighbor Integer Matrix for the Simple N-Point Data Set Shown in Figure 6.20

Point	0	1	2	3	4	5	6	7	8
x_8	x_8	x_2	x_9	x_{14}	x_7	x_3	x_{15}	x_{13}	x_1
x_9	x_9	x_3	x_{10}	x_{15}	x_8	x_4	x_{16}	x_{14}	x_2
x_{10}	x_{10}	x_4	x_{11}	x_{16}	x_9	x_5	x_{17}	x_{15}	x_3
x_{11}	x_{11}	x_5	x_{12}	x_{17}	x_{10}	x_6	x_{18}	x_{16}	x_4
x_{14}	x_{14}	x_8	x_{15}	x_{20}	x_{13}	x_9	x_{21}	x_{19}	x_7
x_{15}	x_{15}	x_9	x_{16}	x_{21}	x_{14}	x_{10}	x_{22}	x_{20}	x_8
x_{16}	x_{16}	x_{10}	x_{17}	x_{22}	x_{15}	x_{11}	x_{23}	x_{21}	x_9
x_{17}	x_{17}	x_{11}	x_{18}	x_{23}	x_{16}	x_{12}	x_{24}	x_{22}	x_{10}
x_{20}	x_{20}	x_{14}	x_{21}	x_{26}	x_{19}	x_{15}	x_{27}	x_{25}	x_{13}
x_{21}	x_{21}	x_{15}	x_{22}	x_{27}	x_{20}	x_{16}	x_{28}	x_{26}	x_{14}
x_{22}	x_{22}	x_{16}	x_{23}	x_{28}	x_{21}	x_{17}	x_{29}	x_{27}	x_{15}
x_{23}	x_{23}	x_{17}	x_{24}	x_{29}	x_{22}	x_{18}	x_{30}	x_{28}	x_{16}

3. Count the number of index matches between the two rows. If the number of matches exceeds k_t (a threshold number to be set), the two points indexed in the first column of the two rows are said to be in the same cluster.

Use the match count as a similarity index (see Table 6.4) to develop single-link and MST-like structures against orderings of this new measure. As the near-neighbor sharing number in this case is a similarity measure (not a distance measure), the structure is a maximum spanning tree. Use absolute thresholds to cut edges in the maximal spanning tree and define the resulting cluster properties in the single linkage context.

Figure 6.21 shows another example, consisting of a point set and its corresponding euclidean metric minimal spanning tree and the shared near neighbor maximal spanning tree for $k_t = 10$. The links with the smallest sharing number are marked "I," those with the next smallest are marked "II," and so on up to "IV." These markings indicate how a hierarchy of clusters would form.

GraphTheoretic Clustering Based on Limited Neighborhood Sets

Most of the approaches discussed previously were based on distance measure, which is effective in many applications. But difficulties would occur if this simple distance measure were employed for the clustering of certain types of data sets, such as those with a change in point density, those with a neck between subclusters, and those with chained clusters within the set, as shown in Figure 6.22a, b, and c, respectively, where it is not difficult to identify the clusters visually.

TABLE 6.4 Index Match Count Between Rows in Table 6.3[a]

Point	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{20}$	$\mathbf{x}_{21}$	$\mathbf{x}_{22}$	$\mathbf{x}_{23}$
$\mathbf{x}_8$	8											
$\mathbf{x}_9$	6	8										
$\mathbf{x}_{10}$	3	6	8									
$\mathbf{x}_{11}$	0	3	6	8								
$\mathbf{x}_{14}$	6	4	2	0	8							
$\mathbf{x}_{15}$	4	6	4	2	6	8						
$\mathbf{x}_{16}$	2	4	6	4	3	6	8					
$\mathbf{x}_{17}$	0	2	4	6	0	3	6	8				
$\mathbf{x}_{20}$	3	2	1	0	6	4	2	0	8			
$\mathbf{x}_{21}$	2	3	2	1	4	6	4	2	6	8		
$\mathbf{x}_{22}$	1	2	3	2	2	4	6	4	3	6	8	
$\mathbf{x}_{23}$	0	1	2	3	0	2	4	6	0	3	6	8

[a] Boundary points are not included.

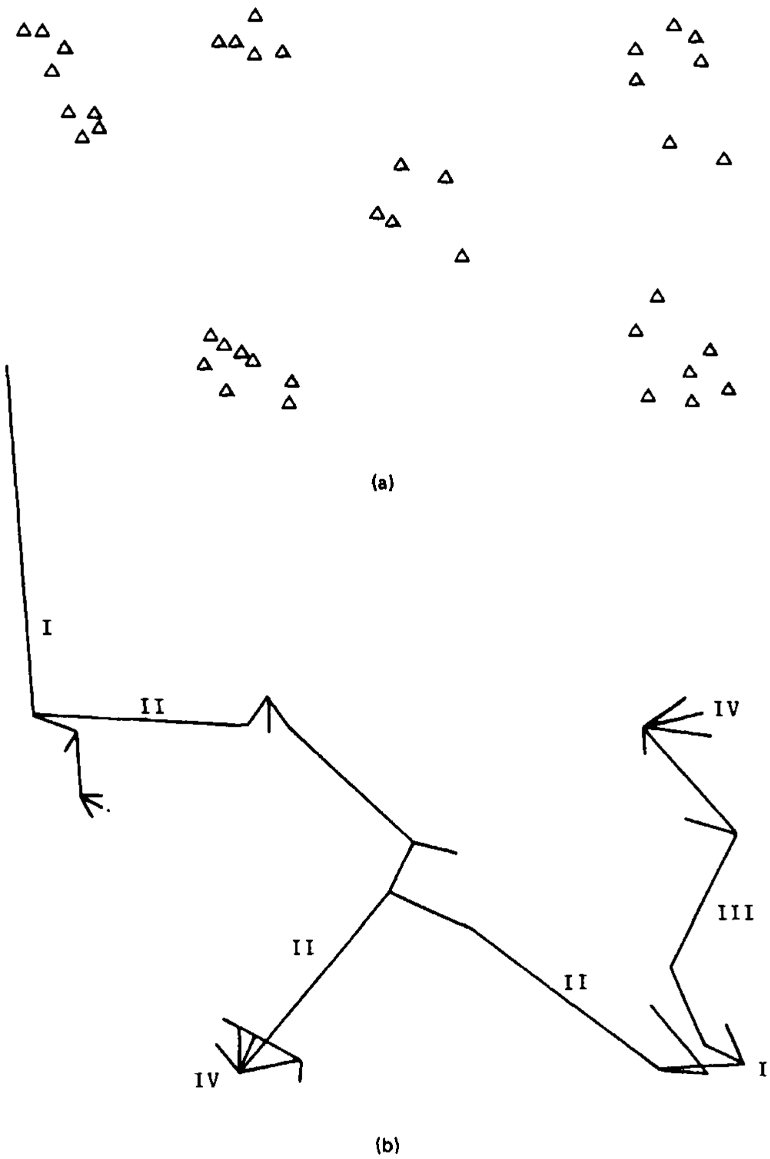


FIGURE 6.21 Another example: (a) a point set and (b) its shared near neighbor maximal spanning tree for $k_t = 10$.

The method discussed in this section is designated primarily for such problems. It is based on the limited neighborhood concept, which originated from the visual perceptual model of clusters.

Several definitions are useful for illustrating this method. Let

$$\mathcal{S} = [S_1, S_2, \ldots, S_M] \qquad \text{and} \qquad \mathcal{R} = [R_1, R_2, \ldots, R_M]$$

where S_l and R_l, $l = 1, 2, \ldots M$, represent, respectively, the graphs and the regions of influence; (p_i, p_j) represents a graph edge joining points p_i and p_j.

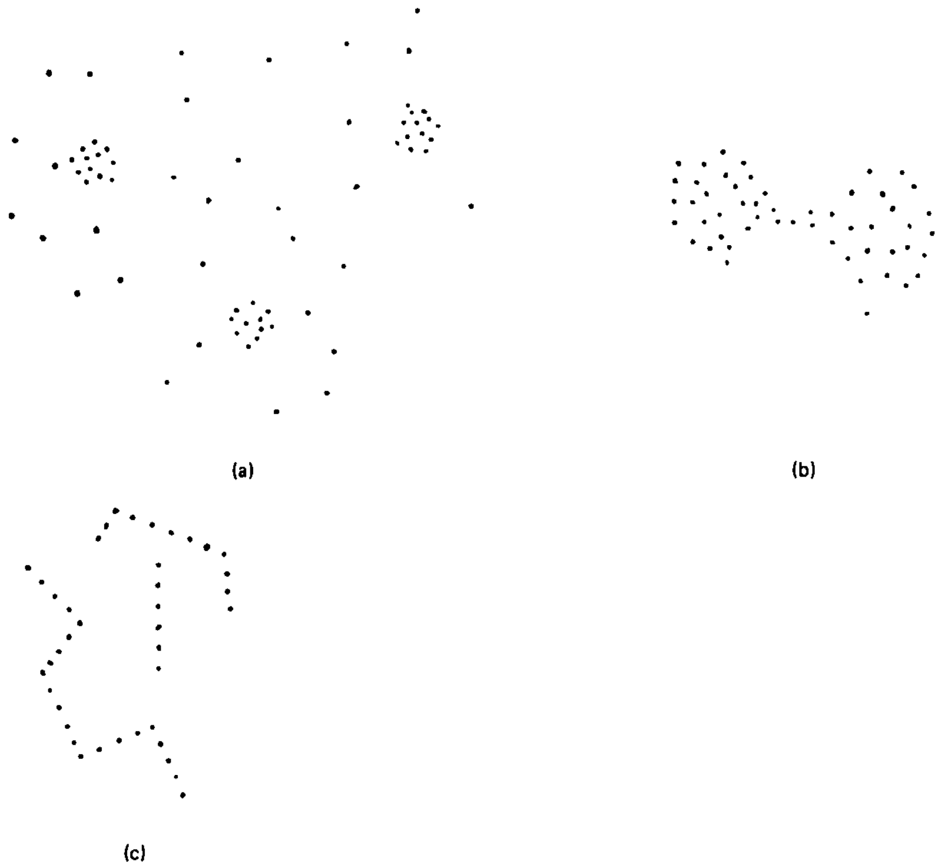


FIGURE 6.22 Examples of some special types of data sets: (a) with change in point density; (b) with a neck between subclusters; (c) with chain clusters.

To illustrate the region of influence, two graphs should be defined: the Gabriel graph and the relative neighborhood graph.

The *Gabriel graph* (GG) is defined in terms of circular regions. Line segment (p_i, p_j) is included as an edge of the GG if no other point p_k lies within or on the boundary of the circle with (p_i, p_j) as the diameter, as shown in Figure 6.23a.

Similarly, the *relative neighborhood graph* (RNG) is defined in terms of a lune region. Line segment (p_i, p_j) is included as an edge of the RNG if no other point p_k lies within or on the boundary of the line, with p_i and p_j as the two points on the circular arcs of the Lunes, S_l and R_l can then be defined as

$$(p_i, p_j) \in S_l \qquad \text{iff } p_k \notin R_l(p_i, p_j)$$

$$\forall k = 1, 2, \ldots, n;\ k \neq i \neq j \tag{6.51}$$

$$R_l(p_i, p_j) = \{\mathbf{x} \colon f[d(\mathbf{x}, p_i), d(\mathbf{x}, p_i)] < d(p_i, p_j);\ i \neq j\} \tag{6.52}$$

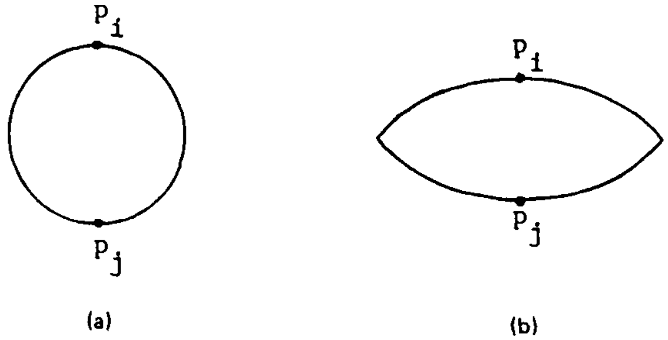


FIGURE 6.23 Definitions for a Gabriel graph and relative neighborhood graph. (a) Circular region defined by GG; (b) lune region defined by RNG.

From the two definitions above it can be seen that S_l defines a limited neighborhood set. If $\max[d(\mathbf{x},p_i), d(\mathbf{x},p_j)]$ is chosen for the function $f[d(\mathbf{x},p_i), d(\mathbf{x},p_j)]$ in Eq. (6.52) [i.e., we find the maximum between $d(\mathbf{x},p_i)$ and $d(\mathbf{x},p_j)$ and use it for the f function], we obtain

$$R_{\text{RNG}}(p_i,p_j) = \{\mathbf{x} : \max[d(\mathbf{x},p_i), d(\mathbf{x},p_j)] < d(p_i, p_j), i \neq j\} \tag{6.53}$$

where $R_{\text{RNG}}(p_i,p_j)$ represents the RNG region of influence. When

$$d^2(\mathbf{x},p_i) + d^2(\mathbf{x},p_j)$$

is used for

$$f[d(\mathbf{x},p_i), d(\mathbf{x},p_j)]$$

we have

$$R_{\text{GG}}(p_i,p_j) = \{\mathbf{x}: d^2(\mathbf{x},p_i) + d^2(\mathbf{x},p_j) \leq d^2(p_i,p_j), i \neq j\} \tag{6.54}$$

where $R_{\text{GG}}(p_i,p_j)$ represents the GG region of influence.

The definition of R_l will determine the property of S_l. If $R_l \subseteq R_{GG}$, the edges of S_l will be nonintersecting. But if $R_l \supset R_{GG}$, intersecting edges are allowed. Take an example to illustrate this. Assume that we have regions of influence such as the following:

$$R_1(p_i,p_j,\beta) = R_{\text{GG}}(p_i,p_j) \cup \{\mathbf{x}: \beta \min[d(\mathbf{x},p_i), d(\mathbf{x},p_j)] < d(p_i,p_j), i \neq j\}$$
$$R_2(p_i,p_j,\beta) = R_{\text{RNG}}(p_i,p_j) \cup \{\mathbf{x}: \beta \min[d(\mathbf{x},p_i), d(\mathbf{x},p_j)] < d(p_i,p_j), i \neq j\}$$

where $0 < \beta < 1$ is a factor of relative edge consistency. Thus $S_1(\beta)$ is obtained from the GG by removing edges (p_i, p_j) if

$$\frac{d(p_i, p_j)}{\min[d(p_i, p_a), d(p_j, p_b)]} > \beta$$

where $p_a(\neq p_j)$ denotes the nearest Gabriel neighbor to p_i and p_b $(\neq p_i)$ denotes the nearest Gabriel neighbor to p_j.

It is then clear that varying β would control the fragmentation of the data set and hence would give a sequence of nested clusterings. Increasing β would break the data set into a greater number of smaller clusters. The examples of two-

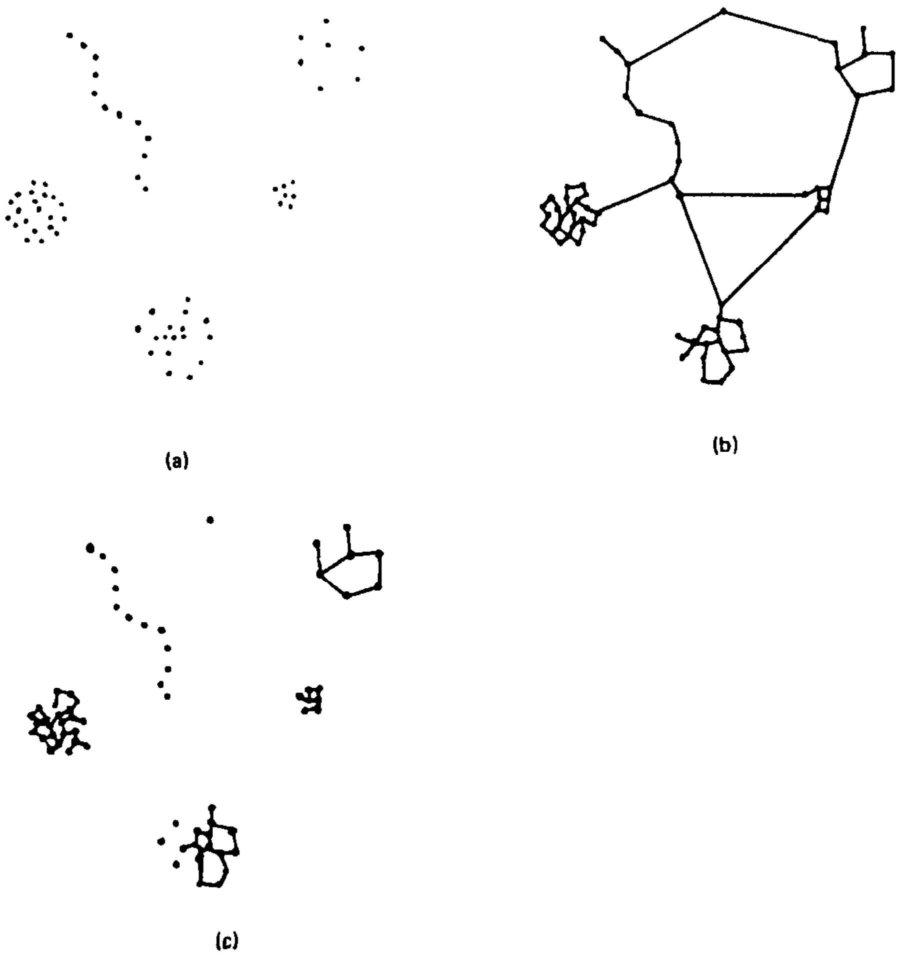


FIGURE 6.24 Set of two-dimensional dot patterns to illustrate a graph theoretic clustering algorithm based on limited neighborhood sets. (a) Data set; (b) one cluster; (c) six clusters.

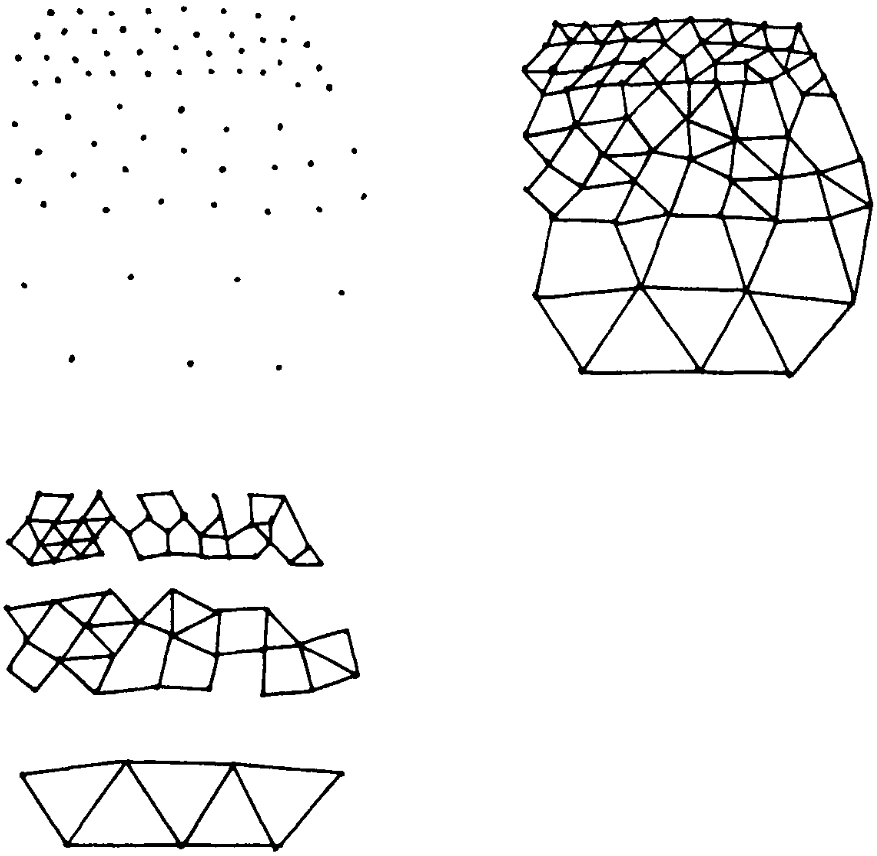


FIGURE 6.25 Set of two-dimensional dot patterns to illustrate a graph theoretic clustering algorithm based on limited neighborhood sets.

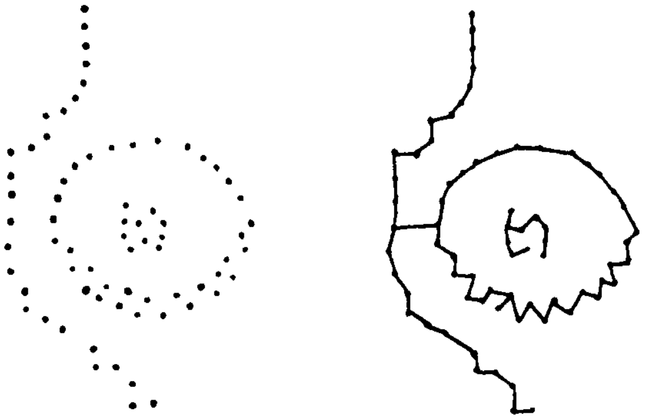


FIGURE 6.26 Set of two-dimensional dot patterns to illustrate a graph theoretic clustering algorithm based on limited neighborhood sets.

dimensional dot patterns shown in Figures 6.24 to 6.26 demonstrate the effectiveness of this clustering method. See Urquhart (1982) for supplementary reading.

6.6 MIXTURE STATISTICS AND UNSUPERVISED LEARNING

Consider a probability density function that is a mixture of other probability functions. This probability function can then be expressed as

$$p(\mathbf{x}) = \sum_{i=1}^{M} p(\omega_i)p(\mathbf{x}|\omega_i) \tag{6.55}$$

$$\sum_{i=1}^{M} p(\omega_i) = 1 \tag{6.56}$$

$$p(\omega_i) \geq 0 \qquad \forall i, i = 1, 2, \ldots, M \tag{6.57}$$

where $p(\mathbf{x}|\omega_i)$ is the likelihood function of ω_i as defined previously and can be interpreted as the probability of $\mathbf{x}$ given that the state of nature is ω_i. $p(\omega_i)$ denotes the a priori probability of $\mathbf{x}$ falling in subset S_i, which is later classified as ω_i. $P(\mathbf{x})$ can then be interpreted as the probability of the unknown pattern samples expressed in terms of the statistics of the natural clusters of those samples. If

$$p(\mathbf{x}|\omega_i)p(\mathbf{x}|\omega_j) \simeq 0 \qquad \forall \mathbf{x}, j \neq i \tag{6.58}$$

that is, little overlap (or nearly no overlap) exists between clusters, then Eq. (6.55) becomes approximately

$$p(\mathbf{x}) = p(\omega_i)p(\mathbf{x}|\omega_i) \qquad \text{if } \mathbf{x} \in S_i \tag{6.59}$$

where

$$S_i = \{\mathbf{x}|\, p(\omega_i)p(\mathbf{x}|\omega_i) \geq p(\omega_j)p(\mathbf{x}|\omega_j), \forall j\}$$

If the overlap is not too great among the component density functions of the mixture, there will exist a one-to-one correspondence between the modes (or local maximums) of the mixture and the individual component class density functions, as shown in Figure 6.27.

The clustering problem is now to locate the modes (or local maximums) of the probability density $p(\mathbf{x})$, and further to represent each mode by $p(\mathbf{x}|\omega_i)$, $i = 1, 2, \ldots$; that is, we are to learn the distribution of $\mathbf{x}$ without supervision. Locations of these modes in $p(\mathbf{x})$ might be used for the center-distance definition of clusters.

Several procedures can be used in the multimodal search. Random search is one procedure, gradient search is another. But if the statistics of each class and the

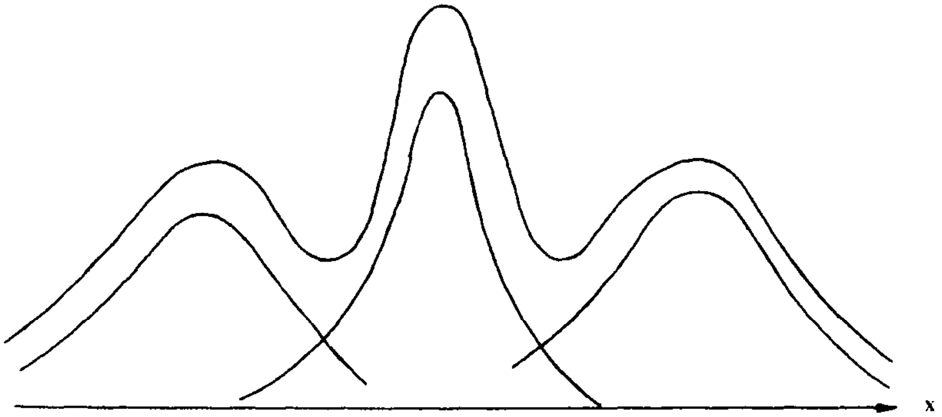


FIGURE 6.27 Composite probability density function.

number of classes are not all known, we must search for some other procedure to use for this problem.

For simplicity, the normal distribution of n dimensions with zero mean is assumed for $p(\mathbf{x})$, or

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|C|^{1/2}} \exp\left(-\tfrac{1}{2}\mathbf{x}^T C^{-1}\mathbf{x}\right) \tag{6.60}$$

Since C^{-1} is real and symmetric, Eq. (6.60) can be diagonalized by an orthogonal transformation by choosing eigenvectors of C^{-1} as the new basis vector ζ so that

$$\mathbf{x} = \zeta\mathbf{y} \qquad \text{and} \qquad \mathbf{x}^T C^{-1}\mathbf{x} = \mathbf{y}^T\Lambda\mathbf{y} \tag{6.61}$$

where

$$\Lambda = \begin{vmatrix} \lambda_1 & & & & 0 \\ & \lambda_2 & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \cdot \\ 0 & & & & \lambda_n \end{vmatrix}$$

and λ_i, $i = 1, 2, \ldots, n$, are the eigenvalues of C^{-1}. The distribution becomes

$$p(\mathbf{y}) = \frac{1}{(2\pi)^{n/2}|C|^{1/2}} \exp\left(-\tfrac{1}{2}\mathbf{y}^T\Lambda\mathbf{y}\right) \tag{6.62}$$

The probability for a pattern to fall within the characteristic domain D of this density function is then given by

$$p(\mathbf{y} \in D) = \frac{1}{(2\pi)^{n/2}|C|^{1/2}} \int_D \exp\left(-\tfrac{1}{2}\mathbf{y}^T \Lambda \mathbf{y}\right) d\mathbf{y} \tag{6.63}$$

where D is the interior of the quadric surface defined by

$$\mathbf{y}^T \Lambda \mathbf{y} = 1 \tag{6.64}$$

whose center is at the origin and whose principal axes have the same directions as the eigenvectors of C^{-1}. Let

$$\mathbf{z} = \Lambda^{1/2}\mathbf{y} = \begin{vmatrix} \sqrt{\lambda_1} & & & & 0 \\ & \ddots & & & \\ & & \sqrt{\lambda_i} & & \\ & & & \ddots & \\ 0 & & & & \sqrt{\lambda_n} \end{vmatrix} \mathbf{y}$$

Equation (6.63) can be further reduced to

$$p(\mathbf{y} \in D) = \frac{\prod_{i=1}^{n} \sqrt{\lambda_i}}{(2\pi)^{n/2}|C|^{1/2}} \int_\phi \exp\left(-\tfrac{1}{2}\mathbf{z}^T \mathbf{z}\right) dz = \text{const.} = \alpha \tag{6.65}$$

where ϕ, the region over which the integral operates, is the circle of unit radius defined by

$$\mathbf{z}^T \mathbf{z} = 1 \qquad \text{and} \qquad |C|^{1/2} = \prod_{i=1}^{n} \sqrt{\lambda_i} \tag{6.66}$$

Equation (6.59) becomes

$$p(\mathbf{x}) = \alpha p(\omega_i) \tag{6.67}$$

Then we have

$$\frac{p(\omega_1)}{p(\mathbf{x} \in D_1)} = \frac{p(\omega_2)}{p(\mathbf{x} \in D_2)} = \cdots = \frac{p(\omega_M)}{p(\mathbf{x} \in D_M)} \tag{6.68}$$

and

$$\sum_{i=1}^{M} p(\omega_i) = 1 \tag{6.69}$$

If N_i, $i = 1, 2, \ldots$, represents the number of patterns falling in the modal domain D_i, then the probability $p(\mathbf{x} \in D_i)$ can be estimated from N_i/N, where N

is the number of patterns used for the test and Eq. (6.68) becomes

$$\frac{p(\omega_1)}{N_1} = \frac{p(\omega_2)}{N_2} = \cdots = \frac{p(\omega_M)}{N_M}$$

$$\sum_{i=1}^{M} p(\omega_i) = 1$$

The problem remaining to be solved is how to locate the modes. The geometrical properties of the modal domain with multivariate gaussian density function have been studied. Analysis of the variations of the mean value of the function $p(\mathbf{x})$ within a suitable domain can be used to determine the local convexity of the function $p(\mathbf{x})$, at point $\mathbf{x}$. See Postaire and Vasseur (1981) for more detailed description of this technique.

6.7 CONCLUDING REMARKS

Clustering, which is a very powerful tool in data classification, is an unsupervised approach. Compared to the supervised approach, this approach is less restricted subjectively by prior knowledge. Approximate application of this natural clustering sometimes produces unexpected inspiration and innovation.

Clustering can be used for training, classification, and mode location as well as for learning. The importance of this approach to classification is considerable. Much has been achieved recently using this line of approach.

It is worthwhile to mention here that the contrast and distinctness of an image can be greatly improved through the use of clustering. More details on this method are given in our discussion of image enhancement by clustering in Chapter 12.

PROBLEMS

6.1 Consider the following samples:

$\mathbf{x}_1 = (6, 0)$	$\mathbf{x}_9 = (-7, 6)$	$\mathbf{x}_{17} = (9, 0)$
$\mathbf{x}_2 = (-4, -5)$	$\mathbf{x}_{10} = (7, 0)$	$\mathbf{x}_{18} = (-7, -5)$
$\mathbf{x}_3 = (-4, 6)$	$\mathbf{x}_{11} = (-5, 8)$	$\mathbf{x}_{19} = (-8, 7)$
$\mathbf{x}_4 = (-6, 6)$	$\mathbf{x}_{12} = (-5, -6)$	$\mathbf{x}_{20} = (7, -1)$
$\mathbf{x}_5 = (-7, -4)$	$\mathbf{x}_{13} = (-6, 9)$	$\mathbf{x}_{21} = (-6, -6)$
$\mathbf{x}_6 = (7, 1)$	$\mathbf{x}_{14} = (-6, -7)$	$\mathbf{x}_{22} = (-7, 8)$
$\mathbf{x}_7 = (-5, -4)$	$\mathbf{x}_{15} = (8, -1)$	$\mathbf{x}_{23} = (-5, 7)$
$\mathbf{x}_8 = (9, 1)$	$\mathbf{x}_{16} = (-7, -6)$	$\mathbf{x}_{24} = (8, 2)$

Determine the cluster centers by means of minimization-of-sum-of-squared-distances algorithm. Arbitrarily choose $z_1(0) = (8, 8)$, $z_2(0) = (-8, -8)$, and $z_3(0) = (-8, 8)$.

6.2 Repeat Problem 6.1 using the following pattern samples:

$\mathbf{x}_1 = (0, 0)$	$\mathbf{x}_8 = (2, 2)$	$\mathbf{x}_{15} = (7, 7)$
$\mathbf{x}_2 = (1, 0)$	$\mathbf{x}_9 = (8.5)$	$\mathbf{x}_{16} = (8, 7)$
$\mathbf{x}_3 = (0, 1)$	$\mathbf{x}_{10} = (6, 6)$	$\mathbf{x}_{17} = (7, 8)$
$\mathbf{x}_4 = (1, 1)$	$\mathbf{x}_{11} = (7, 6)$	$\mathbf{x}_{18} = (8, 8)$
$\mathbf{x}_5 = (2, 1)$	$\mathbf{x}_{12} = (8, 6)$	$\mathbf{x}_{19} = (9, 8)$
$\mathbf{x}_6 = (3, 1)$	$\mathbf{x}_{13} = (9, 6)$	$\mathbf{x}_{20} = (8, 9)$
$\mathbf{x}_7 = (0, 2)$	$\mathbf{x}_{14} = (6, 7)$	

Two classes are assumed and the cluster centers can be chosen arbitrarily.

6.3 Repeat Problem 6.2 using the ISODATA algorithm. Start the procedure with one cluster center.

6.4 Repeat Problem 6.1 with the pattern samples shown in Figure P6.4.

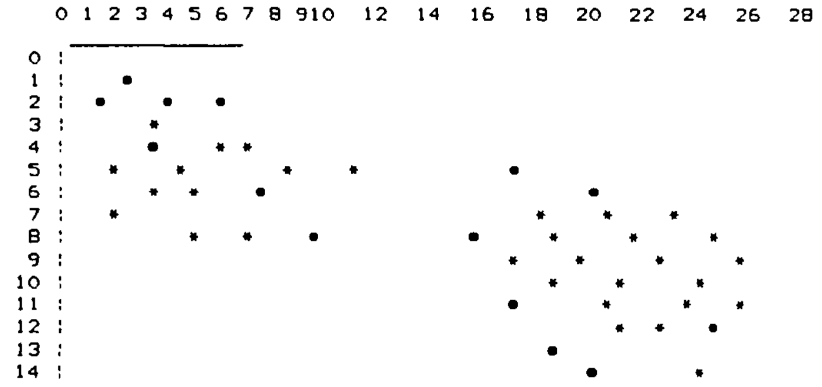


Figure P6.4

6.5 Repeat Problem 6.4 using the ISODATA algorithm. Start the process with one cluster center.

6.6 Use k-nearest neighbor approach to cluster the following set of data:

(3, 4); (5, 5); (7, 7); (3, 7); (6, 10); (3, 11); (6, 12); (8, 9); (9, 5); (9, 12); (10, 14); (11, 8); (12, 11); (13, 5); (14, 9); (15, 7); (17, 7); (19, 5); (19, 10); (20, 7); (21, 4); (22, 12); (22, 9); (22, 2); (24, 5); (25, 7); (25, 12); (25, 10); (25. 1); (26, 3); (28,10); (29, 7); (28, 4); (1, −4); (3, −2); (6, −2); (12, −2); (10, −3); (10, −5); (10, −7); (11, −9); (13, −.6); (13, −4); (8, −6); (8, −9); (7, −3); (8, −4); (7, −5); (7, −7); (2, −6); (4, −3); (3, −8); (5, −8); (5, −5); (6, −5); (5, −4); (6, −4); (3, −4); (3, −6); (4, −5); (6, −3); (5, −6); (15, −2); (15, −5); (15, −7); (14, −8); (14, −10); (18, −10); (16, −9); (16, −7); (16, −6); 16, −4); (17, −1); (18, −3); (18, −5); (18, −7); (18, −6); (19, −6); (18, −8); (20, −9); (21, −10); (21, −7); (19, −4); (22, −4); (24, −2).

Choose $k = 5$, and plot your result to show the clusters. *Suggestion*: Write a program in C or use Matlab to work out this problem.

6.7 Apply K mean method together with DYNOC (Dynamic Optimum Cluster seeking technique) to cluster the following set of data into an optimum number of clusters:

(0, 0), (1, 0), (3, 7); (3, 12); (4, 14); (4, 16); (5, 6); (5, 9); (6, 12);
(6, 17); (7, 5); (7, 14); (8, 10); (8, 12); (8, 17); (9, 7); (10, 13);
(10, 16); (11, 5); (12, 8); (12, 15); (13, 12); (13, 19); (15, 7); (15, 13);
(19, 7); (19,11); (19, 16); (19, 18); (20, 5); (20, 11); (21, 6); (21, 8);
(21, 10); (21, 17); (22, 6); (22, 16); (23, 19); (24, 17); (25, 16);
(25, 19); (26, 5); (26, 7); (26, 18); (27, 12); (27, 17); (27, 19); (28, 3);
(28, 7); (28, 9); (28, 11); (28, 18); (30, 6); (30, 10); (30, 12); (30, 13);
(31, 3); (31, 8); (32, 5); (32,11); (32, 13); (34, 6); ((34, 11).

(a) Write a program to group these data into clusters.

(b) Try K (or N_c) equal to 3, 4, 5, and 6.

(c) Plot $\lambda(N_c)$ versus N_c.

$$\lambda(N_c) = \frac{\min[D_{ij}]}{\max[D_{jj}]}$$

(d) Find the optimum number of clusters and plot them.

Suggestion: Write a program or use Matlab to work out this problem.

6.8 Repeat the previous problem by applying ISODATA together with DYNOC.

6.9 Use the minimum spanning tree approach to cluster the following data:

$\mathbf{x}_1 = (1, 6)$
$\mathbf{x}_2 = (7.6)$
$\mathbf{x}_3 = (10, 7)$
$\mathbf{x}_4 = (11, 6)$
$\mathbf{x}_5 = (12, 6)$
$\mathbf{x}_6 = (13, 6)$
$\mathbf{x}_7 = (14, 6)$
$\mathbf{x}_8 = (1, 5)$
$\mathbf{x}_9 = (3, 5)$
$\mathbf{x}_{10} = (4, 5)$
$\mathbf{x}_{11} = (6, 5)$
$\mathbf{x}_{12} = (10, 5)$
$\mathbf{x}_{13} = (11, 5)$
$\mathbf{x}_{14} = (12, 5)$
$\mathbf{x}_{15} = (13, 5)$
$\mathbf{x}_{16} = (14, 5)$
$\mathbf{x}_{17} = (15, 5)$

$\mathbf{x}_{18} = (2, 4)$
$\mathbf{x}_{19} = (3, 4)$
$\mathbf{x}_{20} = (5, 4)$
$\mathbf{x}_{21} = (6, 4)$
$\mathbf{x}_{22} = (7, 4)$
$\mathbf{x}_{23} = (10, 4)$
$\mathbf{x}_{24} = (11, 4)$
$\mathbf{x}_{25} = (12, 4)$
$\mathbf{x}_{26} = (14, 4)$
$\mathbf{x}_{27} = (15, 4)$
$\mathbf{x}_{28} = (16, 4)$
$\mathbf{x}_{29} = (1, 3)$
$\mathbf{x}_{30} = (2, 3)$
$\mathbf{x}_{31} = (4, 3)$
$\mathbf{x}_{32} = (6, 3)$
$\mathbf{x}_{33} = (8, 3)$
$\mathbf{x}_{34} = (9, 3)$

$\mathbf{x}_{35} = (11.3)$
$\mathbf{x}_{36} = (12, 3)$
$\mathbf{x}_{37} = (13, 3)$
$\mathbf{x}_{38} = (14, 3)$
$\mathbf{x}_{39} = (0, 2)$
$\mathbf{x}_{40} = (3, 2)$
$\mathbf{x}_{41} = (5, 2)$
$\mathbf{x}_{42} = (7, 2)$
$\mathbf{x}_{43} = (10, 2)$
$\mathbf{x}_{44} = (11, 2)$
$\mathbf{x}_{45} = (14, 2)$
$\mathbf{x}_{46} = (4, 1)$
$\mathbf{x}_{47} = (6, 1)$
$\mathbf{x}_{48} = (11, 1)$
$\mathbf{x}_{49} = (12, 1)$
$\mathbf{x}_{50} = (13, 0)$

(a) How many clusters result?

(b) List the connections that define the main diameter.

6.10 A set of two-dimensional dot patterns is shown in Figure P6.10. Use the Gabriel graph (GG) method to do the clustering.

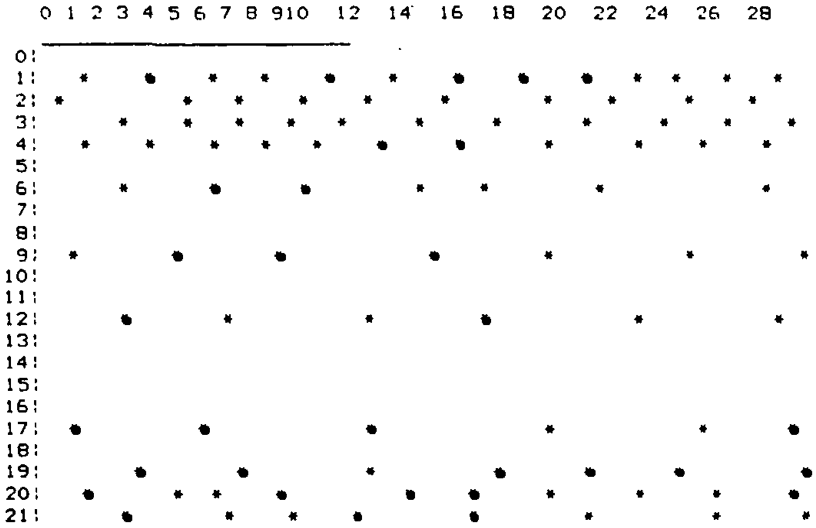


Figure P6.10

6.11 Use the Gabriel graph method to cluster the dot patterns given in Figure P6.11.

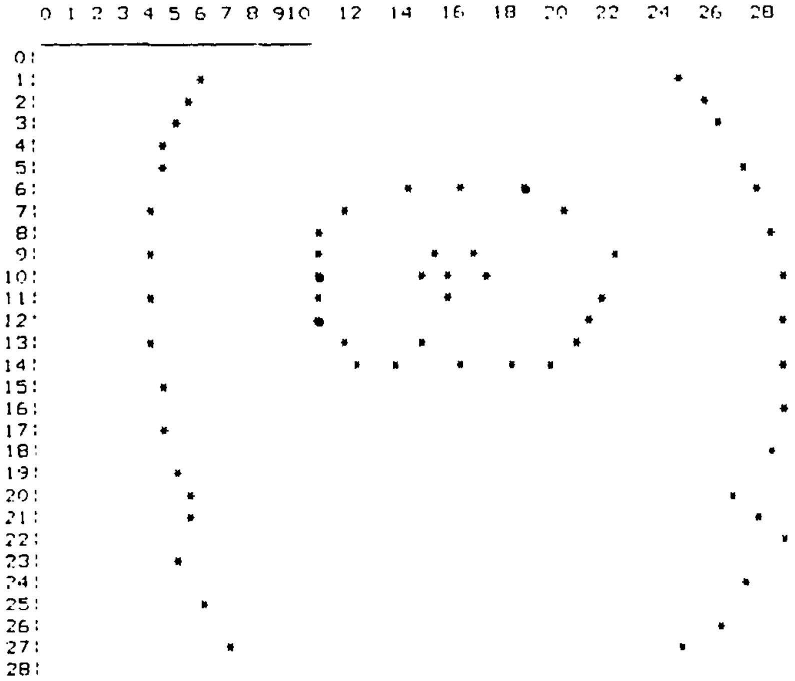


Figure P6.11

7

Dimensionality Reduction and Feature Selection

7.1 OPTIMAL NUMBER OF FEATURES IN CLASSIFICATION OF MULTIVARIATE GAUSSIAN DATA

The pattern space is usually of high dimensionality. The objective of the feature selection is to reduce the dimensionality of the measurement space to a space suitable for the application of pattern classification algorithms. During this process of feature selection, only the salient features necessary for the recognition process are retained so that classification can be implemented on a vastly reduced feature space. Much work has been done in finding the dependences of the probability of misclassification on the dimensionality of the feature vector, the number of training samples, and the true parameters of the class-conditional densities.

As discussed in Chapter 5, for a two-class problem with $p(\omega_1) = p(\omega_2)$, $\mathbf{C}_1 = \mathbf{C}_2 = \mathbf{C}$, and all parameters, including $\mathbf{m}_1$ and $\mathbf{m}_2$, being known, the minimum classification error occurs when a pattern sample $\mathbf{x}$ is classified such that

$$\begin{array}{ll} \mathbf{x} \in \omega_1 & \text{when } \mathbf{x}^T\mathbf{C}^{-1}(\mathbf{m}_1 - \mathbf{m}_2) - \frac{1}{2}(\mathbf{m}_1 + \mathbf{m}_2)^T\mathbf{C}^{-1}(\mathbf{m}_1 - \mathbf{m}_2) > 0 \\ \mathbf{x} \in \omega_2 & \text{otherwise} \end{array} \tag{7.1}$$

But in the case when the parameters are not known, the mean vectors $\mathbf{m}_1$ and $\mathbf{m}_2$ and the estimate of the covariance matrix have to be computed from the training samples N_i, from class $\omega_i, i = 1, 2$.

Unfortunately, evaluation of these quantities is not easy. Various approximations to this statistic have been developed, but they are still very complex in their mathematical forms. Nevertheless, the dependences of the average probability of misclassification on the Mahalanobis distance r^2, the number of samples per class N, and the number of features p are obtained. In general, for a given p, an increase in the values of r^2 and/or N decreases the average error rate. Jain and Walter (1978) have derived an expression for the minimal increase in the Mahalanobis distance needed to keep the misclassification rate unchanged when a new feature is added to the original set of p features:

$$r_{p+1}^2 - r_p^2 = \delta r_p^2 = \frac{r_p^2}{2N - 3 - p} \tag{7.2}$$

where r_p^2 and r_{p+1}^2 represent, respectively, the Mahalanobis distances produced by the p and $p + 1$ features. Equation (7.2) shows that the minimal increase is a fraction of r_p^2 and that this fraction increases with p and decreases with N, the sample size. The problem of determining the optimal number of features for a given sample size now becomes to find when the contribution of an additional feature to the accumulated Mahalanobis distance is below a threshold. Let the contribution of the ith feature to the Mahalanobis distance be d_i^2; then we have

$$r_p^2 = \sum_{i=1}^{p} d_i^2 \tag{7.3}$$

Assume that the features do not have the same power of discrimination and that the contribution of each feature is a fixed fraction of that of the previous feature, such that

$$d_i = \xi d_{i-1} \qquad i = 2, \ldots, p \tag{7.4}$$

where ξ is a positive arbitrary constant and less than I. Substitution of Eq. (6.4) in Eq. (6.3) yields

$$\begin{aligned} r_p^2 &= d_1^2 + \xi^2 d_1^2 + \cdots + \xi^{2p} d_1^2 \\ &= d_1^2 \frac{1 - \xi^{2p}}{1 - \xi^2} \end{aligned} \tag{7.5}$$

where d_1^2 is the Mahalanobis distance computed with only the first feature. Since $\xi < 1$, it implies that the features are arranged in a best-to-worst order. We can then determine the smallest set of features by comparing it with the threshold to maximize the classifier's performance.

7.2 FEATURE ORDERING BY MEANS OF CLUSTERING TRANSFORMATION

As discussed in Section 7.1, features chosen for classification are usually not of the same significance. Decreasing weights assigned to measurements with decreasing significance can be realized through a linear transformation (Tou and Gonzalez, 1974). Let the transformation matrix used for this purpose be a diagonal matrix, or

$$\mathbf{w} = (w_{jk}) \qquad w_{jk} = \begin{cases} 0 & \text{when } j \neq k \\ w_{jj} & \text{when } j = k \end{cases} \tag{7.6}$$

where w_{jj}, $j = 1, \ldots, n$, represents the feature-weighting coefficients. Our problem now is to determine the coefficients w_{jj} so that a good clustering can be obtained. Under such circumstances, the intraset distance between pattern points in a set is minimized. The intraset distance $\bar{D}^2$ for pattern points after transformation has already been derived, as shown by Eq. (6.19), which is repeated here with the weighting coefficients added:

$$\bar{D}^2 = 2\sum_{j=1}^{n}(w_{jj}\sigma_j)^2 \tag{7.7}$$

where σ_j^2 is the sample variance of the components along the x_j coordinate direction. The Lagrange multiplier can be used for minimization of the intraset distance. Two different constraints can be considered.

Constraint 1. When the constraint is $\sum_{j=1}^{n} w_{jj} = 1$, the minimization of $\bar{D}^2$ is equivalent to the minimization of

$$S = 2\sum_{j=1}^{n}(w_{jj}\sigma_j)^2 - \rho_1\left(\sum_{j=1}^{n} w_{jj} - 1\right) \tag{7.8}$$

Take the partial derivative of S with respect to w_{jj} and equating it to zero, we have

$$w_{jj} = \frac{\rho_1}{4\sigma_j^2} \tag{7.9}$$

Similarly, taking the partial derivative of S with respect to the Lagrange multiplier ρ_1 and equating it to zero yields

$$\sum_{j=1}^{n} w_{jj} = 1 \tag{7.10}$$

Combining Eqs. (7.9) and (7.10) yields

$$\sum_{j=1}^{n} \frac{\rho_1}{4\sigma_j^2} = 1 \tag{7.11}$$

or

$$\rho_1 = \frac{4}{\sum_{j=1}^{n} \sigma_j^{-2}} \tag{7.12}$$

Substitution of Eq. (7.12) back into Eq. (7.9) gives the feature weighting coefficients

$$w_{jj} = \frac{1}{\sigma_j^2 \sum_{j=1}^{n} \sigma_j^{-2}} \tag{7.13}$$

From Eq. (7.13) it can be seen that the value under the summation sign in the denominator is the same for all w_{jj}, $j = 1, \ldots, n$, and therefore, w_{jj} varies inversely with σ_j^2.

Constraint 2. When the constraint is $\prod_{j=1}^{n} w_{jj} = 1$, the minimization of $\bar{D}^2$ is equivalent to the minimization of

$$S = 2\sum_{j=1}^{n}(w_{jj}\sigma_j)^2 - \rho_2\left(\prod_{j=1}^{n} w_{jj} - 1\right) \tag{7.14}$$

Taking the partial derivative of S with respect to w_{jj} and equating it to zero yields

$$4w_{jj}\sigma_j^2 = \rho_2 \prod_{\substack{k=1\\k\neq j}}^{n} w_{kk} = 0 \tag{7.15}$$

Multiplying both sides by w_{jj}, we have

$$4w_{jj}^2\sigma_j^2 - \rho_2 \prod_{k=1}^{n} w_{kk} = 0 \tag{7.16}$$

Substitution of $\prod_{j=1}^{n} w_{jj} = 1$ into Eq. (7.16) gives

$$4w_{jj}^2\sigma_j^2 - \rho_2 = 0 \tag{7.17}$$

or

$$w_{jj} = \frac{\sqrt{\rho_2}}{2\sigma_j} \tag{7.18}$$

Similarly,

$$\frac{\partial S}{\partial \rho_2} = \prod_{j=1}^{n} w_{jj} - 1 = 0 \tag{7.19a}$$

or

$$\prod_{j=1}^{n} w_{jj} = 1 \tag{7.19b}$$

which satisfies the given constraint. Substituting Eq. (7.18) into Eq. (7.19b) yields

$$\prod_{j=1}^{n} \frac{\sqrt{\rho_2}}{2\sigma_j} = 1 \tag{7.20}$$

or

$$\frac{(\rho_2)^{n/2}}{2^n \prod_{j=1}^{n} \sigma_j} = 1 \tag{7.21}$$

After simplification, we obtain

$$\rho_2 = 4\left(\prod_{j=1}^{n} \sigma_j\right)^{2/n} \tag{7.22}$$

Combining Eqs. (7.18) and (7.22) yields

$$w_{jj} = \frac{1}{\sigma_j}\left(\sum_{j=1}^{n} \sigma_j\right)^{1/n} \tag{7.23}$$

Note that the continual product inside the parentheses is the same for all w_{jj}, $j = 1, \ldots, n$, and therefore, w_{jj} varies inversely with σ_j.

Although the results obtained are somewhat different for different constraints on w_{jj}, the guides for choosing the feature-weighting coefficients are the same for both cases. That is, a small weight is to be assigned to a feature of large variation, whereas a feature with a small standard deviation σ_j will be weighted heavily. A feature with a small standard deviation σ_j implies that it is more reliable. It is desirable that the more reliable features be more heavily weighted.

7.3 CANONICAL ANALYSIS AND ITS APPLICATIONS TO REMOTE SENSING PROBLEMS

7.3.1 Principal Component Analysis for Dimensionality Reduction

In previous sections we have discussed attempts to perform object classification in high-dimensional spaces. We have also mentioned that improvements can be achieved by mapping the data in pattern space into a feature space. Feature space is of a much lower dimensionality, yet it still preserves most of the intrinsic information. On this basis let us go to the technique of canonical analysis (or the principal component analysis).

The objective of the canonical analysis (or principal component analysis) is to derive a linear transformation that will emphasize the difference among the pattern samples belonging to different categories. In other words, the principal component analysis is to define new coordinate axes in directions of high information content useful for classification purposes.

For multispectral remote sensing applications, each observation will be represented as an n-component vector,

$$\mathbf{x}_{ij} = \begin{vmatrix} x_{ij1} \\ \vdots \\ x_{ijk} \\ \vdots \\ x_{ijn} \end{vmatrix} \tag{7.24}$$

where n represents the dimensionality of the observation vector $\mathbf{x}_{ij}$, or the number of channels used for the observation. x_{ijk} represents the observation (or the intensity of picture element) in the kth channel for picture element j in scan line i. Let $\hat{\mathbf{m}}_l$ and $\hat{\mathbf{C}}_l$ denote, respectively, the sample mean vector and the covariance matrix for the lth category $(l = 1, 2, \ldots, M)$. These two quantities can be obtained from the training set.

Our problem now is to find a transformation matrix,, By means of this transformation, two results are expected. First, the n-dimensional observation vector $\mathbf{x}$ will be transformed into a new vector $\mathbf{y}$ with a dimensionality p which is less than n; or

$$\mathbf{y}_{ij} = \mathbf{A}\mathbf{x}_{ij} \tag{7.25}$$

where $\mathbf{A}$ is a $p \times n$ transformation matrix and $\mathbf{y}_{ij}$ will be represented as

$$\mathbf{y}_{ij} = \begin{vmatrix} y_{ij1} \\ \vdots \\ y_{ijk} \\ \vdots \\ y_{ijp} \end{vmatrix} \tag{7.26}$$

Second, the transformation matrix $\mathbf{A}$ is chosen such that $\mathbf{ACA}^T$ will be a diagonal matrix whose diagonal elements are the variances of the transformed variables and are arranged in descending order. The transformed variables with the largest values can be assumed to have the greatest discriminatory power, since they show the greatest spread (variance) among categories.

The matrix A mentioned above is the covariance of the means of the categories (referred to as the *among-categories* covariance matrix) and represents the spread in n-dimensional space among the categories. It can be defined as

$$\mathbf{C} = \mathbf{XNX}^T - \frac{1}{\sum_{l=1}^{M} n_l}(\mathbf{Xn})(\mathbf{Xn})^T \tag{7.27}$$

where n_l, is the number of observations for category l; M is the number of categories, and $\mathbf{X}$ is defined as an $n \times M$ matrix of all the category means composed of all means vectors $\hat{\mathbf{m}}_k$, $k = 1, 2, \ldots, M$ as

$$\mathbf{X} = [\hat{\mathbf{m}}_1, \hat{\mathbf{m}}_2, \ldots, \hat{\mathbf{m}}_M] = \begin{array}{c|cccc|} & \text{cat. 1} & \text{cat. 2} & \cdots & \text{cat. } M \\ \text{ch. 1} & m_{11} & m_{12} & \cdots & m_{1M} \\ \text{ch. 2} & m_{21} & m_{22} & \cdots & m_{2M} \\ \vdots & \vdots & \vdots & \vdots & \\ \text{ch. } n & m_{n1} & m_{n2} & \cdots & m_{nM} \end{array} \tag{7.28}$$

The $\mathbf{N}$ and $\mathbf{n}$ in Eq. (7.27) are, respectively, an $M \times M$ matrix and an $M \times 1$ vector of the number of observations in the categories as

$$\mathbf{N} = \begin{vmatrix} n_1 & & & 0 \\ & n_2 & & \\ & & \ddots & \\ 0 & & & n_M \end{vmatrix} \tag{7.29}$$

and

$$\mathbf{n} = \begin{vmatrix} n_1 \\ n_2 \\ \vdots \\ n_M \end{vmatrix} \tag{7.30}$$

Let W be the combined covariance matrix for all the categories (referred to as the *within-categories* covariance matrix), which can be computed as

$$W = \frac{\sum_{l=1}^{M}(n_l - 1)\hat{C}_l}{\sum_{l=1}^{M} n_l - M} \tag{7.31}$$

where $\hat{C}_l$ is the covariance matrix for category l, n_l is the number of observations for category l, and M is the number of categories.

The matrix C can be made to be unique only if the following constraint is placed on it:

$$AWA^T = I \tag{7.32}$$

where I is a $p \times p$ identity matrix. Let a new matrix $W^{1/2}$ be such defined that

$$W^{1/2}(W^{1/2})^T = W \tag{7.33}$$

and

$$W^{-1/2} = (W^{1/2})^{-1} \tag{7.34}$$

where $(W^{1/2})^{-1}$ is the inverse of $W^{1/2}$. Then

$$\begin{aligned} ACA^T &= AW^{1/2}W^{-1/2}C(W^{-1/2})^T(W^{1/2})^TA^T \\ &= (AW^{1/2})W^{-1/2}C(W^{-1/2})^T(AW^{1/2})^T \end{aligned} \tag{7.35}$$

and Eq. (7.32) becomes

$$AWA^T = I = (AW^{1/2})(W^{1/2})^TA^T = (AW^{1/2})(AW^{1/2})^T$$

or

$$FF^T = I \tag{7.36}$$

if $CW^{1/2}$ is replaced by F. Equation (7.35) then becomes

$$ACA^T = FVF^T = \Lambda \tag{7.37}$$

where V is used to substitute for $W^{-1/2}C(W^{-1/2})^T$ in Eq. (7.35). Our problem now becomes one of finding F to diagonalize matrix V subject to the constraint $FF^T = I$. Before we can do this, we must first find $W^{1/2}$. Let us construct matrices D and E such that

$$W = EDE^T \tag{7.38}$$

and

$$EE^T = I \tag{7.39}$$

where D is a diagonal matrix whose diagonal elements are the eigenvalues of W, and E is the matrix whose columns are the normalized eigenvectors of W. Then we have

$$W^{1/2} = ED^{1/2}E^T \tag{7.40}$$

and

$$W^{-1/2} = ED^{-1/2}E^T$$

where $D^{1/2}$ is defined as a diagonal matrix whose diagonal elements are the square roots of the corresponding elements in D, and $D^{-1/2}$ is similarly defined.

Once $W^{-1/2}$ has been computed, $V = W^{-1/2}C(W^{-1/2})T$ may be determined. Then the problem becomes one of finding Λ and F from Eq. (7.37) subject to the constraint of Eq. (7.36). That is, Λ is the diagonal matrix of eigenvalues of V, and F is the matrix whose rows are the corresponding normalized eigenvectors. The matrices are then as follows:

$$\Lambda = \left|\begin{array}{cccc|ccc} \lambda_1 & & & 0 & & & \\ & \lambda_2 & & & & 0 & \\ & & \ddots & & & & \\ 0 & & & \lambda_p & & & \\ \hline & & & & \ddots & & \\ & 0 & & & & & \lambda_n \end{array}\right| \tag{7.41a}$$

and

$$F = [f_1, f_2, \ldots, f_n] \tag{7.41b}$$

Only the $p \times p$ submatrix Λ^* of Λ contains the distinguishable eigenvalues such that

$$\Lambda^* = \left|\begin{array}{ccc} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_p \end{array}\right| \tag{7.42}$$

with $\lambda_{p+1} = \cdots = \lambda_n = 0$. This will be used as a discriminant space. In a similar manner, F is partitioned as

$$F^* = [f_1, f_2, \ldots, f_p] \tag{7.43}$$

As a result of this partitioning, the transformation matrix A becomes A^*, such that

$$A^* = F^* W^{-1/2} \tag{7.44}$$

which is now an $p \times n$ matrix.

The mathematics derived in previous paragraphs tells us that n-dimensional observations $\mathbf{x}$ ($\mathbf{x} = [x_1, x_2, \ldots, x_n]$) can be transformed into a new vector $\mathbf{y}$ ($\mathbf{y} = [y_1, y_2, \ldots, y_p]$) with a dimensionality p which is less than n. It also tells us that a diagonal matrix Λ could be found whose elements are eigenvalues of V, and a matrix F whose rows are the corresponding normalized eigenvectors. The relative importance of these eigenvectors (i.e., the ones with the higher discriminating power) could be determined from the relative values of their eigenvalues. Put in other words, we can find a coordinate axis that contains the highest amount of information. All these will help us in the design of a good classification system.

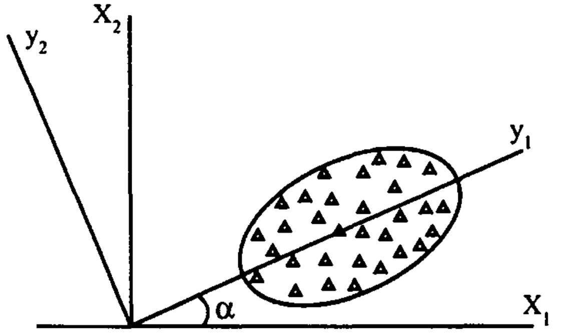


FIGURE 7.1 A two-dimensional pattern space with principal component axes.

Figure 7.1 shows data plotted in a two-dimensional pattern space. This data comes from NASA agricultural application. Although x_1 and x_2 are the two features best selected to represent the objects for this particular application, most of the information is still not on the x_1 axis, nor on the x_2 axis, but on a line y_1 with inclination α with the x_1 axis. This line with the maximum information content is the so-called first *principal component axis*, while the line y_2, which is perpendicular to y_1, containing the least amount of information is called the *least principal component axis.*

Figure 7.2 shows an example of a two-class problem (ω_1 and ω_2). From the distribution of the pattern samples shown in the figure, neither the x_1 axis nor the x_2 axis can effectively discriminate these pattern points from one another. But when we project the distributions onto y_2 as shown, the error probability will be reduced. It is much smaller than that when the data are projected either on the x_1 or the x_2 axis. According to the discussion given above, y_2 will be ranked as the first component axis by its ability to distinguish class ω_1 from ω_2. Data analysis through projection of the pattern data on the first and second principal component axes is very effective in dealing with highs dimensional data sets. Therefore, approach of principal component axes is highly preferred for the optimum design of a linear classifier.

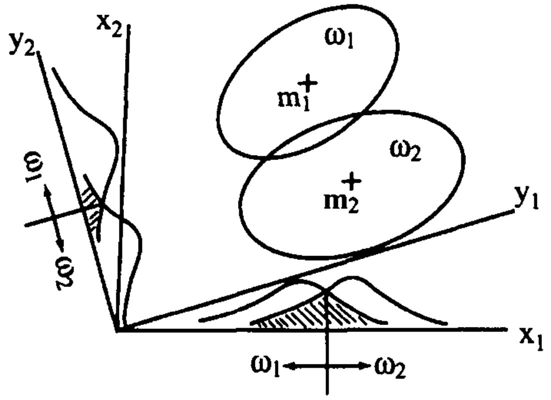


FIGURE 7.2 Selection of the first principal component axis classifier design.

7.3.2 Procedure for Finding the Principal Component Axes

How many principal components are sufficient for a classification problem depends on the problem in discussion. Sometimes the number of components is fixed a priori, as in the case of situations requiring visualization of feature space with limitations imposed due to two- or three-dimensional space requirements. In some other situations, we can set up a threshold value to drop out those principal components when their associated eigenvalues λ are less than the threshold.

Figure 7.3 shows a coordinate system (x_1, x_2). Choose a basis vector such that these vectors point in the direction of maximum variance of the data, say (y_1, y_2). $P(x_1, x_2)$ is a data point in the (x_1, x_2) coordinate system, and can also be expressed as $P(y_1, y_2)$. We then have

$$\begin{aligned} y_1 &= x_1 \cos\theta + x_2 \sin\theta \\ y_2 &= -x_1 \sin\theta + x_2 \cos\theta \end{aligned} \tag{7.45}$$

or

$$\begin{vmatrix} y_1 \\ y_2 \end{vmatrix} = \begin{vmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \end{vmatrix} \tag{7.46}$$

in matrix form:

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{7.47}$$

where

$$\mathbf{A} = \begin{vmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{vmatrix} \tag{7.48}$$

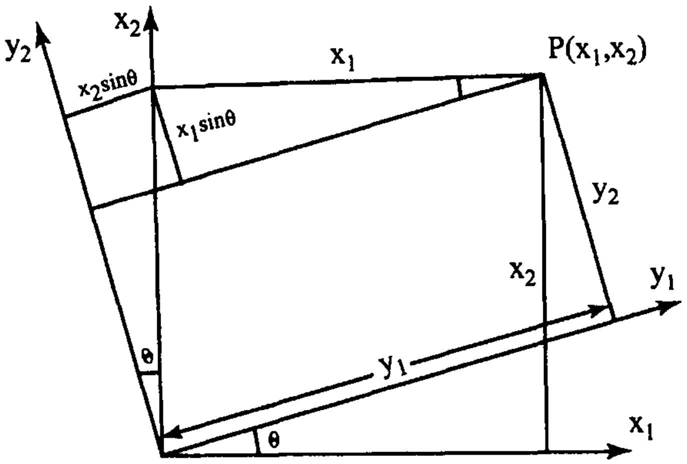


FIGURE 7.3 Example showing the relation between two coordinate systems.

Given $\mathbf{x}$, we can compute $\mathbf{C}_x$. With $\mathbf{A}$ and $\mathbf{C}_x$ we can then compute $\mathbf{C}_y$ such that

$$\mathbf{C}_y = \mathbf{A}\mathbf{C}_x\mathbf{A}^T \tag{7.49}$$

When rows of $\mathbf{A}$ are eigenvectors of $\mathbf{C}_x$, then $\mathbf{C}_y$ should be in diagonal matrix form. Thus, we can find the line that will point in the direction of maximum variance of the data. Let us transpose the matrix $\mathbf{A}$

$$\mathbf{A}^T = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix} = \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{vmatrix} \tag{7.50}$$

Since $\mathbf{e}_1$ and $\mathbf{e}_2$ are eigenvectors of $\mathbf{C}_x$, we then have

$$\mathbf{C}_x\mathbf{e}_1 = \lambda_1\mathbf{e}_1 \tag{7.51}$$

$$\mathbf{C}_x\mathbf{e}_2 = \lambda_2\mathbf{e}_2 \tag{7.52}$$

where λ_1 and λ_2 are eigenvalues. Then

$$\mathbf{C}_x\mathbf{A}^T = \begin{vmatrix} \mathbf{C}_x\mathbf{e}_1 & \mathbf{C}_x\mathbf{e}_2 \end{vmatrix} = \begin{vmatrix} \lambda_2\mathbf{e}_1 & \lambda_2\mathbf{e}_2 \end{vmatrix} \tag{7.53}$$

and

$$\begin{aligned} \mathbf{C}_y = \mathbf{A}(\mathbf{C}_x\mathbf{A}^T) &= \begin{vmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{vmatrix} \begin{vmatrix} \lambda_1\cos\theta & -\lambda_2\sin\theta \\ \lambda_1\sin\theta & \lambda_2\cos\theta \end{vmatrix} \\ &= \begin{vmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{vmatrix} \end{aligned} \tag{7.54}$$

Both $[\cos\theta \quad \sin\theta]^T$ and $[-\sin\theta \quad \cos\theta]^T$ are eigenvectors of $\mathbf{C}_x$. Which one will be accepted as a good eigenvector (or a good principal component) will depend on the value of θ (or depend on the distribution of the pattern points). This seems clear, because θ is determined by $\mathbf{C}_y$, which, in turn, is determined by $\mathbf{C}_x$, and $\mathbf{C}_x$ is computed from the set of pattern points. A threshold value can then be set up to select those principal components with their associated eigenvalues (λ's) greater than the threshold.

Example. Let us use a numerical example to close this section. Given that the following pattern points belong to class ω_1:

$$\begin{array}{llll} x_1^1 = (1 \quad 1)^T & x_1^2 = (2 \quad 10)^T & x_1^3 = (6 \quad 15) & x_1^4 = (10 \quad 18)^T \\ x_1^5 = (18 \quad 19)^T & x_1^6 = (18 \quad 11)^T & x_1^7 = (15 \quad 6) & x_1^8 = (10 \quad 2)^T \end{array}$$

and that the following pattern points belong to class ω_2:

$$\begin{array}{lll} x_2^1 = (-17 \quad -17)^T & x_2^2 = (-16 \quad -7)^T & x_2^3 = (-13 \quad -3) \\ x_2^4 = (-7 \quad -1)^T & x_2^5 = (-1 \quad -1)^T & x_2^6 = (0 \quad -8)^T \\ x_2^7 = (-3 \quad -13) & x_2^8 = (-7 \quad -16)^T & \end{array}$$

our problem is to find the new coordinate axis so that it is along the direction of the highest information content. $\mathbf{m}_1$ and $\mathbf{m}_2$, mean vectors of patterns belonging to ω_1 and ω_2, are, respectively,

$$\begin{aligned}\mathbf{m}_1 &= \tfrac{1}{8}[(1 \quad 1)^T + (2 \quad 10)^T + (6 \quad 15)^T + (10 \quad 18)^T + (18 \quad 19)^T \\ &\quad + (18 \quad 11)^T + (15 \quad 6)^T + (10 \quad 2)^T] = (10 \quad 10.25)^T \\ \mathbf{m}_2 &= \tfrac{1}{8}[(-17 \quad -17)^T + (-16 \quad -7)^T + (-13 \quad -3)^T \\ &\quad + (-7 \quad -1)^T + (-1 \quad -1)^T + (0 \quad -8)^T + (-3 \quad -13)^T \\ &\quad + (-7 \quad -16)^T] = (-8 \quad -8.25)^T\end{aligned}$$

The covariance matrix $\mathbf{C}_{1x}$ for ω_1 is

$$\begin{aligned}\mathbf{C}_{1x} &= \mathbf{E}[(\mathbf{x}-\mathbf{m}_1)(\mathbf{x}-\mathbf{m}_1)^T] = \mathbf{E}[\mathbf{x}\mathbf{x}^T] - \mathbf{m}\mathbf{m}^T \\ &= \frac{1}{8}\begin{vmatrix} 314 & 171 \\ 171 & 338.49 \end{vmatrix}\end{aligned}$$

From matrix $\mathbf{A}$, we have its transpose as

$$\mathbf{A}^T = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}$$

Then

$$\mathbf{C}_{1x}\mathbf{A}^T = \frac{1}{8}\begin{vmatrix} 314 & 171 \\ 171 & 338.49 \end{vmatrix}\begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}$$

or

$$\mathbf{C}_{1x}\mathbf{A}^T = \frac{1}{8}\begin{vmatrix} 314\cos\theta + 171\sin\theta & -314\sin\theta + 171\cos\theta \\ 171\cos\theta + 338.49\sin\theta & -171\sin\theta + 338.49\cos\theta \end{vmatrix}$$

and

$$\mathbf{C}_{1y} = \mathbf{A}(\mathbf{C}_{1x}\mathbf{A}^T)$$

or

$$\mathbf{C}_{1y} = \frac{1}{8}\begin{vmatrix} 314 + 24.49\sin^2\theta & 171(\cos^2\theta - \sin^2\theta) \\ \quad + 171\sin 2\theta & \quad + 24.49\sin\theta\cos\theta \\ & \\ 171(\cos^2\theta - \sin^2\theta) & 314 + 24.49\cos^2\theta \\ \quad + 24.49\sin\theta\cos\theta & \quad - 171\sin 2\theta \end{vmatrix}$$

Set all terms except the diagonal ones equal to zero, we have

$$171(\cos^2\theta - \sin^2\theta) + 24.49\sin\theta\cos\theta = 0$$

θ is evaluated as $-43°$, and

$$C_{1y} = \frac{1}{8}\begin{vmatrix} 154.51 & 0 \\ 0 & 497.98 \end{vmatrix} = \begin{vmatrix} 19.3 & 0 \\ 0 & 62.3 \end{vmatrix}$$

Substituting value of θ into the expression of **A** gives

$$\mathbf{A} = \begin{vmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{vmatrix} = \begin{vmatrix} \cos(-43^\circ) & \sin(-43^\circ) \\ -\sin(-43^\circ) & \cos(-43^\circ) \end{vmatrix}$$

The eigenvectors ϕ_1 and ϕ_2 for ω_1 are then

$$\phi_1 = \begin{vmatrix} \cos\theta \\ \sin\theta \end{vmatrix} = \begin{vmatrix} \cos -43^\circ \\ \sin -43^\circ \end{vmatrix} = \begin{vmatrix} 0.73 \\ -0.68 \end{vmatrix}$$

and

$$\phi_2 = \begin{vmatrix} -\sin\theta \\ \cos\theta \end{vmatrix} = \begin{vmatrix} -\sin(-43^\circ) \\ \cos(-43^\circ) \end{vmatrix} = \begin{vmatrix} 0.68 \\ 0.73 \end{vmatrix}$$

The two eigenvalues are, respectively,

$$\lambda_1 = 314 + 24.49\sin^2\theta + 171\sin 2\theta = 154.51$$
$$\lambda_2 = 314 + 24.49\cos^2\theta - 171\sin 2\theta = 497.98$$

Since $\lambda_2 > \lambda_1$, ϕ_2 will be chosen as the first principal component axis. Similar computations can be worked out for the set of pattern points for ω_2

$$\mathbf{C}_{2y} = \mathbf{A}(\mathbf{C}_{2x}\mathbf{A}^T) = \frac{1}{8}\begin{vmatrix} 370 + 2\sin 2\theta & 71\cos 2\theta \\ 71\cos 2\theta & 370 - 71\sin 2\theta \end{vmatrix}$$

Set the off-diagonal term 71 $\cos 2\theta$ equal to 0, we obtain $\theta = 45^\circ$. Substituting the value of θ into the expression for $\mathbf{C}_{2y}$ gives

$$\mathbf{C}_{2y} = \frac{1}{8}\begin{vmatrix} 46.5 & 0 \\ 0 & 37.38 \end{vmatrix}$$

The eigenvectors ϕ_1 and ϕ_2 for ω_2 are then

$$\phi_1 = \begin{vmatrix} \cos\theta \\ \sin\theta \end{vmatrix} = \begin{vmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{vmatrix}$$

and

$$\phi_2 = \begin{vmatrix} -\sin\theta \\ \cos\theta \end{vmatrix} = \begin{vmatrix} -1\sqrt{2} \\ 1/\sqrt{2} \end{vmatrix}$$

The two eigenvalues are, respectively, $\lambda_1 = 46.5$ and $\lambda_2 = 37.38$. Since $\lambda_1 > \lambda_2$, ϕ_1 is then chosen as the first principal component axis. Figure 7.4 shows the results of the numerical example.

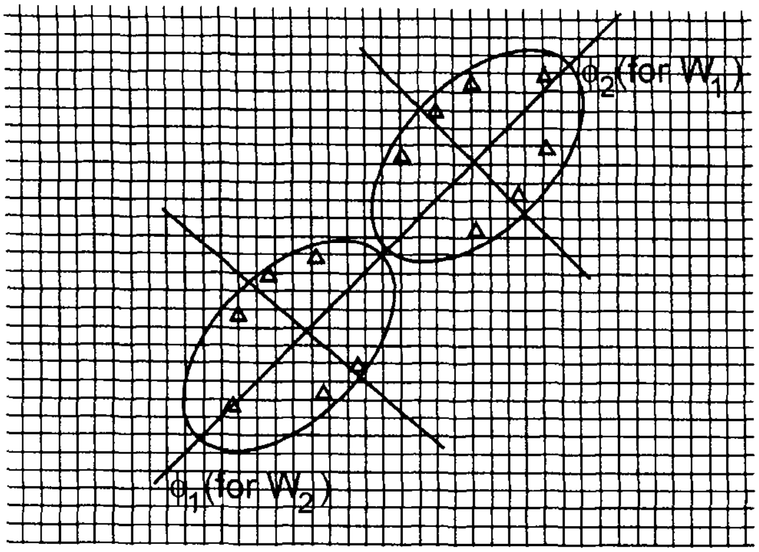


FIGURE 7.4 Results for the numerical examples.

7.4 OPTIMUM CLASSIFICATION WITH FISHER'S DISCRIMINANT

Multidimensional space, no doubt, provides us with more information for classification. However, the extremely large amount of data involved makes it more difficult for us to find the most appropriate hyperplane for classification. The conventional way to handle this problem is to preprocess the data so as to reduce its dimensionality before applying a classification algorithm. Fisher's linear discriminant uses a linear projection of the n-dimensional data onto a one-dimensional space (i.e., a line). It is our hope that their projections onto a line will be well separated by class. In so doing, our classification problem becomes choosing a line that is so oriented to maximize this class separation without data intermingled.

Consider an input vector $\mathbf{x}$ is projected on a line as a scalar value y, and is given by

$$y = \mathbf{w}^T\mathbf{x} \tag{7.55}$$

where $\mathbf{w}$ is a vector of adjustable weight parameter. By adjusting the components of the weight vector $\mathbf{w}$, we can select a projection, which maximizes the class separation.

Consider a two-class problem. There are N_1 pattern samples in class ω_1, and N_2 samples in ω_2. The mean vectors of these two classes, $\mathbf{m}_1$ and $\mathbf{m}_2$, are then, respectively,

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{i=1}^{N_1} \mathbf{x}_1^i \tag{7.56}$$

$$\mathbf{m}_2 = \frac{1}{N_2}\sum_{i=1}^{N_2} \mathbf{x}_2^i \tag{7.57}$$

where the subscripts denote the classes, while the superscripts denote the pattern samples in the class. Note that the projection mean m_1 for class ω_1 is a scalar and is given by

$$\begin{aligned} m_1 &= \frac{1}{N_1}\sum_{i=1}^{N_1} y_1^i = \frac{1}{N_1}\sum_{i=1}^{N_1} \mathbf{w}^T\mathbf{x}_1^i \\ &= \mathbf{w}^T \frac{1}{N_1}\sum_{i=1}^{N_1} \mathbf{x}_1^i = \mathbf{w}^T\mathbf{m}_1 \end{aligned} \tag{7.58}$$

Similarly, the projection mean of class ω_2 is also a scalar and is given by

$$\begin{aligned} m_2 &= \frac{1}{N_2}\sum_{i=1}^{N_2} y_2^i = \frac{1}{N_2}\sum_{i=1}^{N_2} \mathbf{w}^T\mathbf{x}_2^i \\ &= \mathbf{w}^T \frac{1}{N_2}\sum_{i=1}^{N_2} \mathbf{x}_2^i = \mathbf{w}^T\mathbf{m}_2 \end{aligned} \tag{7.59}$$

The difference between the means of the projected data is therefore

$$m_2 = m_1 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1) \tag{7.60}$$

It seems that when data are projected onto $\mathbf{w}$, separation of the classes looks the same as the separation of the projected class means, and we might then simply choose an appropriate $\mathbf{w}$ to maximize $(m_2 - m_1)$. However, some cases, as shown in Figure 7.5, would remain to be solved. Projection of $(\mathbf{m}_2 - \mathbf{m}_1)$ on the x_1 axis is larger than that on the x_2 axis, resulting in larger separation on x_1 axis but with larger class overlap. We, therefore, have to take into account the within-class spreads (scatters) of the data points (or the covariance of the class).

Let us define the within-class scatter of the projection data as

$$\mathbf{C}_{y1} = \sum_{y \in Y_1} (y - m_1)^2 \tag{7.61}$$

and

$$\mathbf{C}_{y2} = \sum_{y \in Y_2} (y - m_2)^2 \tag{7.62}$$

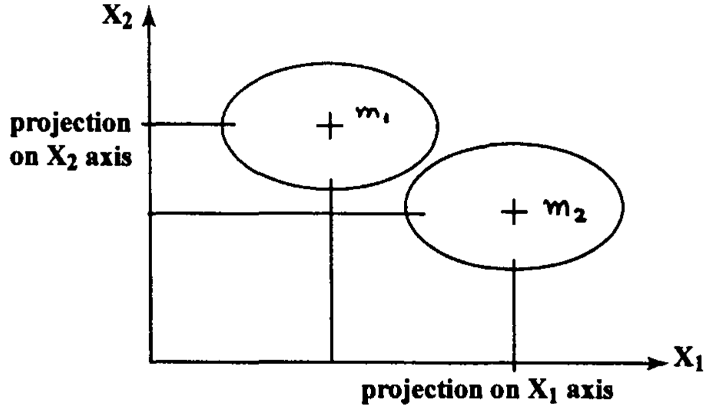


FIGURE 7.5 Illustration on the necessity to take into account the within-class covariance when constructing Fisher's discriminant.

where Y_1 and Y_2 are, respectively, the projections of the pattern points from ω_1 and ω_2. Fisher's criterion is then

$$J(\mathbf{w}) = \frac{\text{squared distance of the } Y \text{ means}}{\text{variance of } Y} \tag{7.63}$$

or

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{\mathbf{C}_{y1} + \mathbf{C}_{y2}} \tag{7.64}$$

where $\mathbf{C}_{y1}$ and $\mathbf{C}_{y2}$ are, respectively, measures of *within-class scatters* of the projected data. The sum of $\mathbf{C}_{y1}$ and $\mathbf{C}_{y2}$ gives the total within-class covariance for the whole data set. The numerator of $J(\mathbf{w})$, $(m_2 - m_1)^2$, can be rewritten as sample means as

$$(m_2 - m_1)^2 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T\mathbf{w} = \mathbf{w}^T\mathbf{S}_B\mathbf{w} \tag{7.65}$$

with

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \tag{7.66}$$

where $\mathbf{S}_B$ is the *between-class* covariance matrix. Let $\mathbf{S}_w$ be the total within-class covariance matrix:

$$\mathbf{S}_w = \sum_{\substack{i=1 \\ \mathbf{x}^i \in \omega_1}}^{N_1} (\mathbf{x}^i - \mathbf{m}_1)(\mathbf{x}^i - \mathbf{m}_1)^T + \sum_{\substack{i=1 \\ \mathbf{x}^i \in \omega_2}}^{N_2} (\mathbf{x}^i - \mathbf{m}_2)(\mathbf{x}^i - \mathbf{m}_2)^T = \mathbf{C}_{x1} + \mathbf{C}_{x2} \tag{7.67}$$

Then we have

$$\mathbf{C}_{y1} + \mathbf{C}_{y2} = \mathbf{w}^T \mathbf{S}_w \mathbf{w} \tag{7.68}$$

$J(\mathbf{w})$ can be expressed as

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}} \tag{7.69}$$

$J(\mathbf{w})$ is Fisher's index of performance (or Fisher's criterion function) to be maximized. To maximize $J(\mathbf{w})$, let us differentiate it with respect to $\mathbf{w}$, and set the result equal to zero, we obtain

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w})\mathbf{S}_w \mathbf{w} = (\mathbf{w}^T \mathbf{S}_w \mathbf{w})\mathbf{S}_B \mathbf{w} \tag{7.70}$$

Note that both $(\mathbf{w}^T \mathbf{S}_B \mathbf{w})$ and $(\mathbf{w}^T \mathbf{S}_w \mathbf{w})$ are scalar. Rearranging gives

$$\mathbf{S}_w \mathbf{w}(\mathbf{w}^T \mathbf{S}_B \mathbf{w})(\mathbf{w}^T \mathbf{S}_w \mathbf{w})^{-1} = \mathbf{S}_B \mathbf{w} \tag{7.71}$$

Multiplying both sides by $\mathbf{S}_w^{-1}$ and replacing $(\mathbf{w}^T \mathbf{S}_B \mathbf{w})(\mathbf{w}^T \mathbf{S}_w \mathbf{w})^{-1}$ with λ gives

$$(\mathbf{S}_w^{-1} \mathbf{S}_B)\mathbf{w} = \lambda \mathbf{w} \tag{7.72}$$

Remembering that $(\mathbf{S}_w^{-1} \mathbf{S}_B)$ is a matrix, the above expression turns out to be an eigenvector problem. A solution for $\mathbf{w}$ may be found by solving for an eigenvector of the matrix $(\mathbf{S}_w^{-1} \mathbf{S}_B)$.

An alternative solution can be obtained basing on the fact that $\mathbf{S}_B \mathbf{w}$ has the direction of $(\mathbf{m}_2 - \mathbf{m}_1)$. From

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \tag{7.73}$$

we can see that $\mathbf{S}_B \mathbf{w}$ is always in the direction of $(\mathbf{m}_2 - \mathbf{m}_1)$. In some cases we do not even care about the magnitude of $\mathbf{w}$, but only in its direction. Thus, we can drop out the scalar factors and have the following proportionality relationship:

$$\mathbf{w} \propto \mathbf{S}_w^{-1}(\mathbf{m}_2 - \mathbf{m}_1) \tag{7.74}$$

A Numerical Example

Given that the following pattern points belong to class ω_1:

$$\begin{array}{lll}
x_1^1 = (8 \quad 12)^T & x_1^2 = (8 \quad 14)^T & x_1^3 = (9 \quad 16)^T \\
x_1^4 = (10 \quad 14)^T & x_1^5 = (12 \quad 13)^T & x_1^6 = (12 \quad 14)^T \\
x_1^7 = (11 \quad 17)^T & x_1^8 = (11 \quad 19)^T & x_1^9 = (14 \quad 24)^T \\
x_1^{10} = (14 \quad 22)^T & x_1^{11} = (16 \quad 21)^T & x_1^{12} = (14 \quad 16)^T \\
x_1^{13} = (16 \quad 17)^T & x_1^{14} = (18 \quad 18)^T & x_1^{15} = (18 \quad 20)^T \\
x_1^{16} = (17 \quad 24)^T & x_1^{17} = (19 \quad 22)^T & x_1^{18} = (19 \quad 24)^T \\
x_1^{19} = (21 \quad 24)^T & x_1^{20} = (22 \quad 22)^T &
\end{array}$$

and that the following pattern points belong to class ω_2:

$$\begin{array}{lll}
x_2^1 = (21 \quad 11)^T & x_2^2 = (19 \quad 14)^T & x_2^3 = (21 \quad 17)^T \\
x_2^4 = (23 \quad 14)^T & x_2^5 = (25 \quad 13)^T & x_2^6 = (24 \quad 20)^T \\
x_2^7 = (27 \quad 17)^T & x_2^8 = (28 \quad 15)^T & x_2^9 = (29 \quad 18)^T \\
x_2^{10} = (27 \quad 23)^T & x_2^{11} = (32 \quad 22)^T & x_2^{12} = (30 \quad 36)^T \\
x_2^{13} = (32 \quad 19)^T & x_2^{14} = (33 \quad 25)^T & x_2^{15} = (33 \quad 29)^T \\
x_2^{16} = (35 \quad 22)^T & x_2^{17} = (36 \quad 27)^T & x_2^{18} = (37 \quad 25)^T \\
x_2^{19} = (38 \quad 29)^T & x_2^{20} = (37 \quad 31)^T &
\end{array}$$

find the line on which projection of the two-dimensional data as listed above can be well separated by class.

Let $\mathbf{m}_l$ and $\mathbf{C}_l$, denote, respectively, the sample mean vector and the covariance matrix for the lth category ($l = 1, 2$). These two sets of quantities can be obtained from the data sets listed above and are found as:

$$\mathbf{m}_1 = \begin{vmatrix} 14.45 \\ 18.65 \end{vmatrix} \qquad \mathbf{m}_2 = \begin{vmatrix} 29.35 \\ 21.35 \end{vmatrix}$$

and

$$\mathbf{C}_{x1} = \begin{vmatrix} 17.35 & 12.61 \\ 12.61 & 15.83 \end{vmatrix} \qquad \mathbf{C}_{x2} = \begin{vmatrix} 32.83 & 28.03 \\ 28.03 & 43.63 \end{vmatrix}$$

The total within-class covariance matrix is then

$$\mathbf{S}_w = \mathrm{C}_{x1} + \mathbf{C}_{x2} = \begin{vmatrix} 50.18 & 40.64 \\ 40.64 & 59.26 \end{vmatrix}$$

From $\mathbf{S}_w$, we can obtain its inverse as

$$\mathbf{S}_w^{-1} = \begin{vmatrix} 0.0453 & -0.0310 \\ -0.0310 & 0.0382 \end{vmatrix}$$

The between class covariance matrix, $\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$, can be evaluated as

$$\mathbf{S}_B = \begin{vmatrix} 29.35 - 14.45 \\ 21.35 - 18.65 \end{vmatrix} |29.35 - 14.45 \qquad 21.35 - 18.65|$$

$$= \begin{vmatrix} 222 & 40.23 \\ 40.23 & 7.29 \end{vmatrix}$$

From the two matrices $(\mathbf{S}_w^{-1})$ and $(\mathbf{S}_B)$ we get

$$(\mathbf{S}_w^{-1})(\mathbf{S}_B) = \begin{vmatrix} 8.81 & 1.59 \\ -5.35 & -0.97 \end{vmatrix}$$

Referring to the eigenvector equation (7.72) as derived above, and reproduced here:

$$(\mathbf{S}_w^{-1})(\mathbf{S}_B)\mathbf{w} = \lambda\mathbf{w}$$

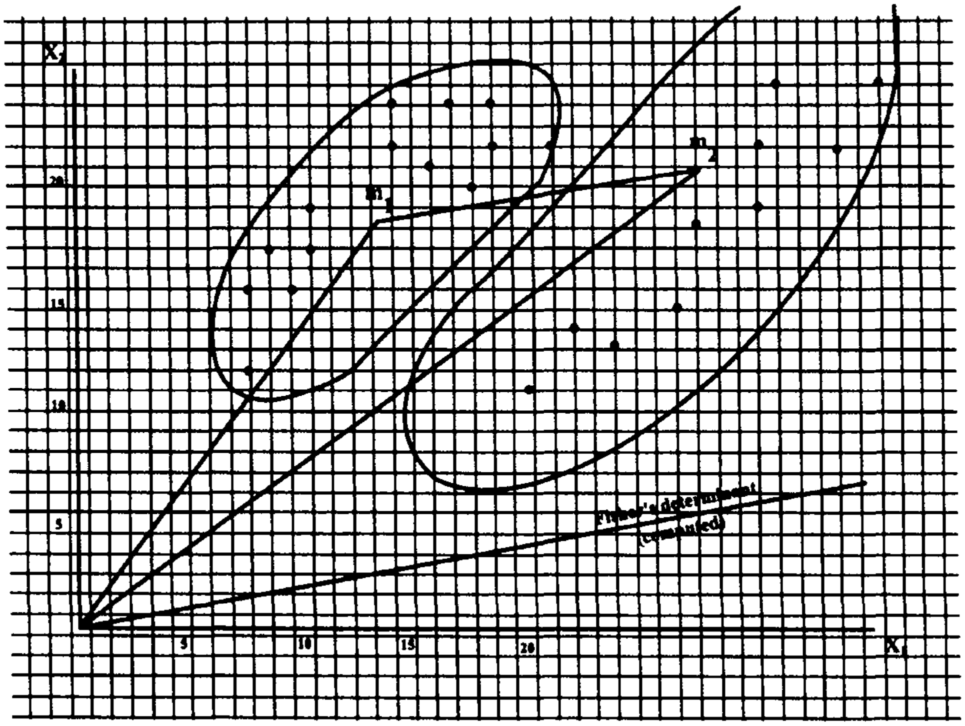


FIGURE 7.6 A two-dimensional plot with Fisher's discriminant drawn for the numerical example.

we can find the eigenvectors

$$\phi_1 = \begin{vmatrix} 8.81 \\ 1.59 \end{vmatrix}$$

and

$$\phi_2 = \begin{vmatrix} -5.35 \\ -0.97 \end{vmatrix}$$

Figure 7.6 shows the two-dimensional plot with Fisher's discriminant drawn [a line with an inclination of arctangent (1.59/8.81) or 10.23° with the x_1 axis]. Note the derivation in direction of Fisher's discriminant from $(\mathbf{m}_2 - \mathbf{m}_1)$. This is because spreads of the two classes have been taken into account.

7.5 NONPARAMETRIC FEATURE SELECTION METHOD APPLICABLE TO MIXED FEATURES

The feature selection methods discussed previously are for those features that are usual quantitative variables. In this section we discuss a nonparametric feature selection method that can be applied to pattern recognition problems based on mixed features. That is, some of the features are quantitative, whereas others are qualitative.

Feature selection for this purpose based on local interclass structure was suggested by Ichino (1981). This method consists of the following three steps:

1. Divide the pattern space into a set of subregions, or, in basic event generation (BEG) terminology, generate the basic events $\mathbf{E}_{ik}$, $k = 1, \ldots, N_{e_i}$ for class ω_i by means of the BEG algorithm.
2. Use the set theoretic feature selection method to find a subset of features, F_{ik}, for each subregion (i.e., for each basic event $\mathbf{E}_{ik}$).
3. Construct the desired feature subset by taking the union of feature subsets obtained in step 2 as $\bigcup_{k=1}^{N_{e_i}} F_{ik}$.

The basic event generation algorithm used in step 1 is essentially a merging process. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a pattern vector in the pattern space $\mathbf{R}^n = R \times R \times \cdots \times R$. Then if $\mathbf{E}_1$ and $\mathbf{E}_2$ are two events in R^n, it is obvious that "merge" of these two events is also in R^n, thus

$$M(\mathbf{E}_1, \mathbf{E}_2) = E \subset R^n \tag{7.75}$$

where $M(.,.)$ represents the merging function. Suppose that we have training samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_i}$ from class ω_i, and training samples $\mathbf{y}, \mathbf{y}_2, \ldots, \mathbf{y}_{N_j}$ from classes other than class ω_i. The events generated by the BEG algorithm for class ω_i will be $\mathbf{E}_{ik}$, $k = 1, 2, \ldots, N_{e_i}(N_{e_i} \leq N_i)$. Then we have

$$\mathbf{x}_l \in \bigcup_{k=1}^{N_{e_i}} \mathbf{E}_{ik} \qquad l = 1, 2, \ldots, N_i \tag{7.76}$$

and

$$\text{dist}(\mathbf{y}_l|\mathbf{E}_{lk}) \geq T_A \qquad l = 1, 2, \ldots, N_j;\; k = 1, 2, \ldots, N_{e_i} \tag{7.77}$$

where dist$(\mathbf{y}_l|\mathbf{E}_{ik})$, the distance between $\mathbf{y}_l$ and $\mathbf{E}_{ik}$, must be greater than or equal to T_A, a certain positive number that is usually chosen to be 1. If $\mathbf{E}_{ik}$ and $\mathbf{y}_l$ are expressed, respectively, as

$$\mathbf{E}_{ik} = E_{ik}^1 \times E_{ik}^2 \times \cdots \times E_{ik}^n \tag{7.78}$$

and

$$\mathbf{y}_l = (y_{li}, y_{l2}, \ldots, y_{ln}) \tag{7.79}$$

then dist$(\mathbf{y}_l|\mathbf{E}_{ik})$ can be defined as

$$\text{dist}(\mathbf{y}_l|\mathbf{E}_{ik}) = \sum_{p=1}^{n} \phi(y_{lp}|E_{ik}^p) \tag{7.80}$$

where

$$\phi(y_{lp}|E_{ik}^p) = \begin{cases} 1 & \text{if } y_{lp} \notin E_{ik}^p \\ 0 & \text{otherwise} \end{cases} \tag{7.81}$$

After the basic events $\mathbf{E}_{ik}$, $k = 1, 2, \ldots, N_{e_i}$, are generated by the BEG algorithm for class ω_i, the next step will be to select a minimum number of features F_{ik}, by which the basic events $\mathbf{E}_{ik}$ for class ω_i can be separated from the training samples drawn from classes other than ω_i. F_{ik} is said to be the minimum feature subset for the event $\mathbf{E}_{ik}$ if the following two conditions are satisfied:

$$(1) \quad \text{dist}(\mathbf{y}_l|\mathbf{E}_{ik})_{F_{ik}} \geq T_A \qquad l = 1, 2, \ldots, N_j; k = 1, 2, \ldots, N_{e_i} \tag{7.82}$$

$$(2) \quad \text{dist }(\mathbf{y}_l|E_{ik})_{F_{ik}-[p]} < T_A \qquad \text{for some values of } l \tag{7.83}$$

where p is a feature in F_{ik}. Equations (7.82) and (7.83) are evaluated, respectively, with feature subsets F_{ik} and $F_{ik} - [p]$. Conditions 1 and 2 are equivalent, respectively, to

$$|F_{ik} \geq F_{ik}^l| \geq T_A \qquad l = 1, 2, \ldots, N_j;\; k = 1, 2, \ldots, N_{e_i} \tag{7.84}$$

and

$$|(F_{ik} - [\,p]) \cap F_{ik}^l| < T_A \qquad \text{for some values of } l \tag{7.85}$$

where the F_{ik}^l represent the sets of effective features for l training samples not in class ω_i. A sequential search procedure for this feature selection can be summarized as follows:

1. Let F_{ik} be the original feature set F_0.
2. For a feature $p \in F_{ik}$, if $|F_{ik} - [p]) \cap F^l_{ik}| \geq T_A$, $l = 1, 2, \ldots, N_j$, discard the feature and replace F_{ik} by $F_{ik} - [p]$; otherwise, try for other features in F_{ik}.
3. Iterate step 2 until no feature in F_{ik} can be removed.

The resultant F_{ik} obtained by this procedure is a minimum feature subset. For example, the original feature set is chosen to be $F_0 = [a, b, c, d]$. Let $F^1_{ik} = [b, c]$, $F^2_{ik} = [a]$, $F^3_{ik} = [a]$, and $F^4_{ik} = [b, d]$ be the sets of effective features for the given four training samples, $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$, and $\mathbf{y}_4$, respectively. According to the sequential search procedure, either features c and d or feature b can be discarded, but not both. The minimum feature subset is then $[a, b]$ or $[a, c, d]$.

PROBLEMS

7.1 Given that the following pattern points belong to class ω_1:

$$\begin{array}{lll}
x_1^1 = (8 \quad 12)^T & x_1^2 = (8 \quad 14)^T & x_1^3 = (9 \quad 16)^T \\
x_1^4 = (10 \quad 14)^T & x_1^5 = (12 \quad 13)^T & x_1^6 = (12 \quad 14)^T \\
x_1^7 = (11 \quad 17)^T & x_1^8 = (11 \quad 19)^T & x_1^9 = (14 \quad 24)^T \\
x_1^{10} = (14 \quad 22)^T & x_1^{11} = (16 \quad 21)^T & x_1^{12} = (14 \quad 16)^T \\
x_1^{13} = (16 \quad 17)^T & x_1^{14} = (18 \quad 18)^T & x_1^{15} = (18 \quad 20)^T \\
x_1^{16} = (17 \quad 24)^T & x_1^{17} = (19 \quad 22)^T & x_1^{18} = (19 \quad 24)^T \\
x_1^{19} = (21 \quad 24)^T & x_1^{20} = (22 \quad 22)^T &
\end{array}$$

and that the following pattern points belong to class ω_2:

$$\begin{array}{lll}
x_2^1 = (21 \quad 11)^T & x_2^2 = (19 \quad 14)^T & x_2^3 = (21 \quad 17)^T \\
x_2^4 = (23 \quad 14)^T & x_2^5 = (25 \quad 13)^T & x_2^6 = (24 \quad 20)^T \\
x_2^7 = (27 \quad 17)^T & x_2^8 = (28 \quad 15)^T & x_2^9 = (29 \quad 18)^T \\
x_2^{10} = (27 \quad 23)^T & x_2^{11} = (32 \quad 22)^T & x_2^{12} = (30 \quad 36)^T \\
x_2^{13} = (32 \quad 19)^T & x_2^{14} = (33 \quad 25)^T & x_2^{15} = (33 \quad 29)^T \\
x_2^{16} = (35 \quad 22)^T & x_2^{17} = (36 \quad 27)^T & x_2^{18} = (37 \quad 25)^T \\
x_2^{19} = (38 \quad 29)^T & x_2^{20} = (37 \quad 31)^T &
\end{array}$$

find the line on which projection of the two-dimensional data as listed can be well separated by class.

7.2 (a) Under what condition it is accurate enough to choose $(\mathbf{m}_2 - \mathbf{m}_1)$ as the direction of Fisher's discriminant for projection of the data down to one dimension?

(b) Shown in which one of the cases shown in Figure P7.2 we can use $(\mathbf{m}_2 - \mathbf{m}_1)$ as the direction of the line for data projection

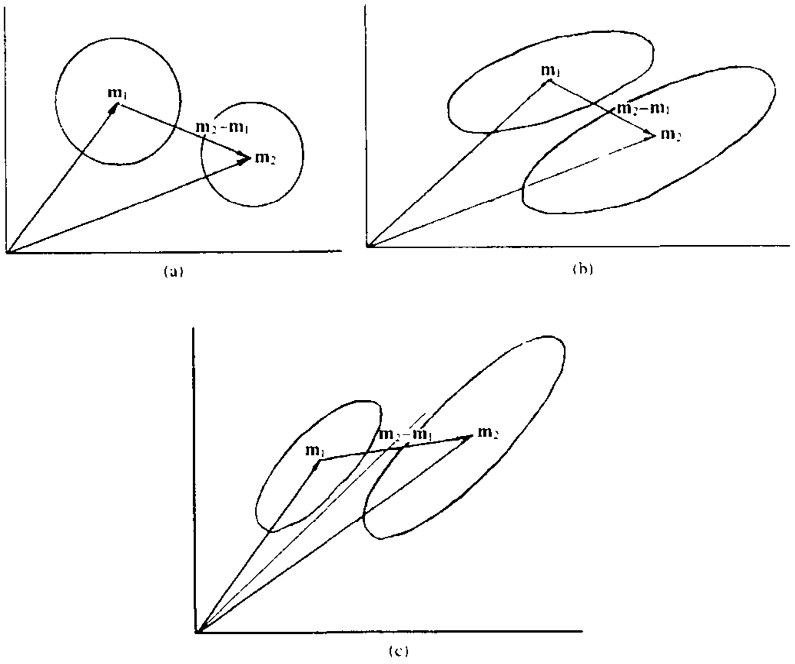


Figure P7.2

7.3 Given that the following pattern points belong to class ω_1:

$$x_1^1 = (1 \quad 1)^T \qquad x_1^2 = (2 \quad 10)^T \qquad x_1^3 = (6 \quad 15)^T \qquad x_1^4 = (10 \quad 18)^T$$
$$x_1^5 = (18 \quad 19)^T \qquad x_1^6 = (18 \quad 11)^T \qquad x_1^7 = (15 \quad 6)^T \qquad x_1^8 = (10 \quad 2)^T$$

and that the following pattern points belong to class ω_2:

$$x_2^1 = (-17 \quad -17)^T \qquad x_2^2 = (-16 \quad -7)^T \qquad x_2^3 = (-13 \quad -3)^T$$
$$x_2^4 = (-7 \quad -1)^T \qquad x_2^5 = (-1 \quad -1)^T \qquad x_2^6 = (0 \quad -8)^T$$
$$x_2^7 = (-3 \quad -13)^T \qquad x_2^8 = (-7 \quad -16)^T$$

find the new coordinate axis, which is in the direction of the highest information content and is useful for the classification of the two classes.

7.4 Given that the following pattern points belong to class ω_1:

$$x_1^1 = (0 \quad 0)^T \qquad x_1^2 = (5 \quad 5)^T \qquad x_1^3 = (14 \quad 8)^T$$
$$x_1^4 = (17 \quad 1)^T \qquad x_1^5 = (15 \quad -7)^T \qquad x_1^6 = (11 \quad -13)^T$$
$$x_1^7 = (3 \quad -15)^T \qquad x_1^8 = (-1 \quad -7)^T$$

and that the following pattern points belong to class ω_2:

$x_2^1 = (-16 \quad 2)^T$ $\quad x_2^2 = (-15 \quad 7)^T$ $\quad x_2^3 = (-11 \quad 11)^T$

$x_2^4 = (-2 \quad 13)^T$ $\quad x_2^5 = (0 \quad 5)^T$ $\quad x_2^6 = (-1 \quad -1)^T$

$x_2^7 = (-5 \quad -4)^T$ $\quad x_2^8 = (-13 \quad -5)^T$

find the new coordinate axis, which is in the direction of the highest information content and is useful for the classification of the two classes.

7.5 Write a program to perform the linear transformation that will emphasize the difference among the pattern samples belonging to two different categories:

Class 1:

$\mathbf{x}_1^1 = (11.3, 11.2)$ $\quad \mathbf{x}_1^2 = (11.3, 12.2)$ $\quad \mathbf{x}_1^3 = (13.3, 16.2)$

$\mathbf{x}_1^4 = (14.3, 14.2)$ $\quad \mathbf{x}_1^5 = (15.3, 16.2)$ $\quad \mathbf{x}_1^6 = (13.3, 15.2)$

$\mathbf{x}_1^7 = (16.3, 14.2)$ $\quad \mathbf{x}_1^8 = (16.3, 14.2)$ $\quad \mathbf{x}_1^9 = (17.3, 18.2)$

$\mathbf{x}_1^{10} = (16.3, 16.2)$ $\quad \mathbf{x}_1^{11} = (17.3, 17.2)$ $\quad \mathbf{x}_1^{12} = (15.3, 15.2)$

$\mathbf{x}_1^{13} = (18.3, 16.2)$ $\quad \mathbf{x}_1^{14} = (15.3, 14.2)$ $\quad \mathbf{x}_1^{15} = (15.3, 15.2)$

$\mathbf{x}_1^{16} = (12.3, 16.2)$ $\quad \mathbf{x}_1^{17} = (13.3, 12.2)$ $\quad \mathbf{x}_1^{18} = (14.3, 12.2)$

$\mathbf{x}_1^{19} = (13.3, 13.2)$ $\quad \mathbf{x}_1^{20} = (15.3, 12.2)$ $\quad \mathbf{x}_1^{21} = (14.3, 14.2)$

$\mathbf{x}_1^{22} = (10.3, 11.2)$ $\quad \mathbf{x}_1^{23} = (15.3, 12.2)$ $\quad \mathbf{x}_1^{24} = (12.3, 13.2)$

$\mathbf{x}_1^{25} = (13.3, 13.2)$ $\quad \mathbf{x}_1^{26} = (13.3, 12.2)$ $\quad \mathbf{x}_1^{27} = (14.3, 14.2)$

$\mathbf{x}_1^{28} = (14.3, 14.2)$ $\quad \mathbf{x}_1^{29} = (15.3, 16.2)$ $\quad \mathbf{x}_1^{30} = (15.3, 16.2)$

$\mathbf{x}_1^{31} = (13.3, 14.2)$ $\quad \mathbf{x}_1^{32} = (13.3, 15.2)$ $\quad \mathbf{x}_1^{33} = (15.3, 16.2)$

$\mathbf{x}_1^{34} = (16.3, 14.2)$ $\quad \mathbf{x}_1^{35} = (13.3, 13.2)$ $\quad \mathbf{x}_1^{36} = (13.3, 13.2)$

$\mathbf{x}_1^{37} = (15.3, 15.2)$ $\quad \mathbf{x}_1^{38} = (13.3, 12.2)$ $\quad \mathbf{x}_1^{39} = (13.3, 14.2)$

$\mathbf{x}_1^{40} = (12.3, 12.2)$ $\quad \mathbf{x}_1^{41} = (14.3, 14.2)$ $\quad \mathbf{x}_1^{42} = (11.3, 12.2)$

$\mathbf{x}_1^{43} = (15.3, 13.2)$ $\quad \mathbf{x}_1^{44} = (14.3, 16.2)$ $\quad \mathbf{x}_1^{45} = (13.3, 14.2)$

$\mathbf{x}_1^{46} = (14.3, 15.2)$ $\quad \mathbf{x}_1^{47} = (14.3, 17.2)$ $\quad \mathbf{x}_1^{48} = (15.3, 13.2)$

$\mathbf{x}_1^{49} = (16.3, 12.2)$ $\quad \mathbf{x}_1^{50} = (13.3, 13.2)$ $\quad \mathbf{x}_1^{51} = (14.3, 13.2)$

$\mathbf{x}_1^{52} = (17.3, 17.2)$ $\quad \mathbf{x}_1^{53} = (17.3, 15.2)$ $\quad \mathbf{x}_1^{54} = (18.3, 15.2)$

$\mathbf{x}_1^{55} = (15.3, 17.2)$ $\quad \mathbf{x}_1^{56} = (16.3, 17.2)$ $\quad \mathbf{x}_1^{57} = (15.3, 16.2)$

$\mathbf{x}_1^{58} = (16.3, 16.2)$ $\quad \mathbf{x}_1^{59} = (17.3, 15.2)$ $\quad \mathbf{x}_1^{60} = (15.3, 15.2)$

$\mathbf{x}_1^{61} = (16.3, 17.2)$ $\quad \mathbf{x}_1^{62} = (15.3, 16.2)$

Class 3:

$x_3^1 = (-5.7, -0.8)$	$x_3^2 = (-11.7, -7.8)$	$x_3^3 = (-5.7, -4.8)$
$x_3^4 = (-3.7, -1.8)$	$x_3^5 = (-0.7, -0.8)$	$x_3^6 = (-2.7, -2.8)$
$x_3^7 = (-5.7, -3.8)$	$x_3^8 = (-4.7, -4.8)$	$x_3^9 = (-4.7, -1.8)$
$x_3^{10} = (-3.7, -4.8)$	$x_3^{11} = (-3.7, -2.8)$	$x_3^{12} = (-9.7, -9.8)$
$x_3^{13} = (-8.7, -7.8)$	$x_3^{14} = (-10.7, -7.8)$	$x_3^{15} = (-4.7, -2.8)$
$x_3^{16} = (-9.7, -8.8)$	$x_3^{17} = (-5.7, -7.8)$	$x_3^{18} = (-6.7, -5.8)$
$x_3^{19} = (-9.7, -9.8)$	$x_3^{20} = (-3.7, 1.2)$	$x_3^{21} = (-9.7, -5.8)$
$x_3^{22} = (-3.7, -2.8)$	$x_3^{23} = (-6.7, -6.8)$	$x_3^{24} = (-7.7, -7.8)$
$x_3^{25} = (-7.7, -5.8)$	$x_3^{26} = (-0.7, 1.2)$	$x_3^{27} = (-8.7, -3.8)$
$x_3^{28} = (1.3, 1.2)$	$x_3^{29} = (2.3, 1.2)$	$x_3^{30} = (-1.7, 3.2)$
$x_3^{31} = (1.3, 1.2)$	$x_3^{32} = (2.3, 3.2)$	$x_3^{33} = (1.3, 1.2)$
$x_3^{34} = (1.3, 3.2)$	$x_3^{35} = (-0.7, -0.8)$	$x_3^{36} = (3, 3, 3.2)$
$x_3^{37} = (2.3, 2.2)$	$x_3^{38} = (2.3, -0.8)$	$x_3^{39} = (-0.7, -1.8)$
$x_3^{40} = (1.3, 0.2)$	$x_3^{41} = (-0.7, -4.8)$	$x_3^{42} = (2.3, 4.2)$
$x_3^{43} = (5.3, 1.2)$	$x_3^{44} = (2.3, -0.8)$	$x_3^{45} = (-3.7, -3.8)$
$x_3^{46} = (4.3, 4.2)$	$x_3^{47} = (2.3, 3.2)$	$x_3^{48} = (-2.7, -3.8)$
$x_3^{49} = (-6.7, -7.8)$	$x_3^{50} = (4.3, 4.2)$	$x_3^{51} = (-2.7, -2.8)$
$x_3^{52} = (-1, 7 - 1.8)$	$x_3^{53} = (3.3, -0.8)$	$x_3^{54} = (6.3, 6.2)$
$x_3^{55} = (1.3, 3.2)$	$x_3^{56} = (8.3, 2.2)$	$x_3^{57} = (1.3, 1.2)$
$x_3^{58} = (3.3, 5.2)$	$x_3^{59} = (3.3, 4.2)$	$x_3^{60} = (3.3, 4.2)$
$x_3^{61} = (2.3, 2.2)$	$x_3^{62} = (11.3, 7.2)$	$x_3^{63} = (5.3, 7.2)$
$x_3^{64} = (4.3, 3.2)$		

7.6 Write a program to perform the linear transformation that will emphasize the difference among the pattern samples belonging to two different categories:

Class 2:

$x_2^1 = (-17.7, -12.8)$	$x_2^2 = (-18.7, -14.8)$	$x_2^3 = (-13.7, -11.8)$
$x_2^4 = (-21.7, -16.8)$	$x_2^5 = (-20.7, -15.8)$	$x_2^6 = (-18.7, -13.8)$
$x_2^7 = (-20.7, -15.8)$	$x_2^8 = (-16.7, -12.8)$	$x_2^9 = (-21.7, -14.8)$

$x_2^{10} = (-21.7, -17.8)$	$x_2^{11} = (-20.7, -16.8)$	$x_2^{12} = (-20.7, -17.8)$
$x_2^{13} = (-16.7, -15.8)$	$x_2^{14} = (-16.7, -17.8)$	$x_2^{15} = (-26.7, -24.8)$
$x_2^{16} = (-23.7, -17.8)$	$x_2^{17} = (-18.7, -19.8)$	$x_2^{18} = (-25.7, -23.8)$
$x_2^{19} = (-19.7, -17.8)$	$x_2^{20} = (-22.7, -22.8)$	$x_2^{21} = (-26.7, -24.8)$
$x_2^{22} = (-20.7, -16.8)$	$x_2^{23} = (-19.7, -19.8)$	$x_2^{24} = (-20.7, -19.8)$
$x_2^{25} = (-17.7, -15.8)$	$x_2^{26} = (-23.7, -21.8)$	$x_2^{27} = (-17.7, -16.8)$
$x_2^{28} = (-21.7, -18.8)$	$x_2^{29} = (-16.7, -14.8)$	$x_2^{30} = (-21.7, -17.8)$
$x_2^{31} = (-17.7, -13.8)$	$x_2^{32} = (-15.7, -11.8)$	$x_2^{33} = (-16.7, -11.8)$
$x_2^{34} = (-18.7, -15.8)$	$x_2^{35} = (-15.7, -13.8)$	$x_2^{36} = (-19.7, -18.8)$
$x_2^{37} = (-19.7, -16.8)$	$x_2^{38} = (-16.7, -11.8)$	$x_2^{39} = (-21.7, -18.8)$
$x_2^{40} = (-18.7, -15.8)$	$x_2^{41} = (-13.7, -11.8)$	$x_2^{42} = (-17.7, -16.8)$
$x_2^{43} = (-16.7, -10.8)$	$x_2^{44} = (-13.7, -12.8)$	$x_2^{45} = (-18.7, -15.8)$
$x_2^{46} = (-16.7, -13.8)$	$x_2^{47} = (-13.7, -13.8)$	$x_2^{48} = (-22.7, -16.8)$
$x_2^{49} = (-14.7, -13.8)$	$x_2^{50} = (-16.7, -11.8)$	$x_2^{51} = (-18.7, -16.8)$
$x_2^{52} = (-14.7, -12.8)$	$x_2^{53} = (-22.7, -19.8)$	$x_2^{54} = (-16.7, -14.8)$
$x_2^{55} = (-15.7, -13.8)$	$x_2^{56} = (-19.7, -18.8)$	$x_2^{57} = (-10.7, -7.8)$
$x_2^{58} = (-14.7, -11.8)$	$x_2^{59} = (-18.7, -16.8)$	$x_2^{60} = (-11.7, -13.8)$
$x_2^{61} = (-18.7, -14.8)$		

Class 3:

$x_3^{1} = (-5.7, -0.8)$	$x_3^{2} = (-11.7, -7.8)$	$x_3^{3} = (-5.7, -4.8)$
$x_3^{4} = (-3.7, -1.8)$	$x_3^{5} = (-0.7, -0.8)$	$x_3^{6} = (-2.7, -2.8)$
$x_3^{7} = (-5.7, -3.8)$	$x_3^{8} = (-4.7, -4.8)$	$x_3^{9} = (-4.7, -1.8)$
$x_3^{10} = (-3.7, -4.8)$	$x_3^{11} = (-3.7, -2.8)$	$x_3^{12} = (-9.7, -9.8)$
$x_3^{13} = (-8.7, -7.8)$	$x_3^{14} = (-10.7, -7.8)$	$x_3^{15} = (-4.7, -3.8)$
$x_3^{16} = (-9.7, -8.8)$	$x_3^{17} = (-5.7, -7.8)$	$x_3^{18} = (-6.7, -5.8)$
$x_3^{19} = (-9.7, -9.8)$	$x_3^{20} = (-3.7, 1.2)$	$x_3^{21} = (-9.7, -5.8)$
$x_3^{22} = (-3.7, -2.8)$	$x_3^{23} = (-6.7, -6.8)$	$x_3^{24} = (-7.7, -7.8)$
$x_3^{25} = (-7.7, -5.8)$	$x_3^{26} = (-0.7, 1.2)$	$x_3^{27} = (-8.7, -3.8)$
$x_3^{28} = (1.3, 1.2)$	$x_3^{29} = (2.3, 1.2)$	$x_3^{30} = (-1.7, 3.2)$
$x_3^{31} = (1.3, 1.2)$	$x_3^{32} = (2.3, 3.2)$	$x_3^{33} = (1.3, 1.2)$

$\mathbf{x}_3^{34} = (1.3, 3.2)$ $\quad \mathbf{x}_3^{35} = (-0.7, -0.8)$ $\quad \mathbf{x}_3^{36} = (3.3, 3.2)$

$\mathbf{x}_3^{37} = (2.3, 2.2)$ $\quad \mathbf{x}_3^{38} = (2.3, -0.8)$ $\quad \mathbf{x}_3^{39} = (-0.7, -1.8)$

$\mathbf{x}_3^{40} = (1.3, 0.2)$ $\quad \mathbf{x}_3^{41} = (-0.7, -4.8)$ $\quad \mathbf{x}_3^{42} = (2.3, 4.2)$

$\mathbf{x}_3^{43} = (5.3, 1.2)$ $\quad \mathbf{x}_3^{44} = (2.3, -0.8)$ $\quad \mathbf{x}_3^{45} = (-3.7, -3.8)$

$\mathbf{x}_3^{46} = (4.3, 4.2)$ $\quad \mathbf{x}_3^{47} = (2.3, 3.2)$ $\quad \mathbf{x}_3^{48} = (-2.7, -3.8)$

$\mathbf{x}_3^{49} = (-6.7, -7.8)$ $\quad \mathbf{x}_3^{50} = (4.3, 4.2)$ $\quad \mathbf{x}_3^{51} = (-2.7, -2.8)$

$\mathbf{x}_3^{52} = (-1.7, -1.8)$ $\quad \mathbf{x}_3^{53} = (3.3, -0.8)$ $\quad \mathbf{x}_3^{54} = (6.3, 6.2)$

$\mathbf{x}_3^{55} = (1.3, 3.2)$ $\quad \mathbf{x}_3^{56} = (8.3, 2.2)$ $\quad \mathbf{x}_3^{57} = (1.3, 1.2)$

$\mathbf{x}_3^{58} = (3.3, 5.2)$ $\quad \mathbf{x}_3^{59} = (3.3, 4.2)$ $\quad \mathbf{x}_3^{60} = (3.3, 4.2)$

$\mathbf{x}_3^{61} = (2.3, 2.2)$ $\quad \mathbf{x}_3^{62} = (11.3, 7.2)$ $\quad \mathbf{x}_3^{63} = (5.3, 7.2)$

$\mathbf{x}_3^{64} = (4.3, 3.2)$

This Page Intentionally Left Blank

Part II

Neural Networks for Pattern Recognition

Neural networks are networks of living nerve cells. Such a network of neurons (nerve cells) possesses the capability of thinking, feeling, learning, and remembering. Artificial neural networks and neurocomputers are models inspired by these brain functions, and are defined as mathematical models of theorized mind and brain activity. Artificial neural networks go by many names, such as connectionist models, parallel distributed processing models, neuromorphic systems, adaptive systems, and self-organizing systems. Whatever the name, the objective in studying these models is to understand how the brain provides human beings with such abilities as perceptual interpretations, reasoning, and learning, that is, how such "computations" are organized and carried out in the brain.

Artificial neural networks (ANNs) are being developed as a technological discipline that can automatically develop operational capabilities to adaptively respond to an information environment. The evolution of the artificial neural networks has not been smooth. Research in this field has been under way since the 1950s. The period 1950–1960 was the golden era of artificial neural networks. By the mid-1960s, the first success of neurocomputing drew to a close for some reasons.

During its quiet years (from 1967 to 1982) little explicit neurocomputing research was carried out in the United States. A great deal of neural network research went on under the headings of pattern recognition, biological modeling, and adaptive signal processing. In 1983 enthusiasm for neural networks returned. The first breakthrough was through the DARPA (Defense Advance Research Projects Agency). They bravely diverged from tradition. Now the research on neural networks has become a hot subject, and there are quite a few new applications

Why did it die and resurrect again? This is due to the restriction inherent in the technology in the early days of this research. Inputs passed serially to the classifier and computations were performed sequentially. In addition, system parameters were typically estimated from the training data set and then held constant. However, in neural network classifier, the internal functional computations would naturally be carried out in parallel with output feedback. The parameters or weights used in the net would not be held constant but adapted or trained continuously during use.

Thanks to advances in technology such as new net topologies and algorithms, and new analog VLSI implementation techniques, as well as a deeper understanding of how the human brain works, a great deal of interest has been promoted during recent years. This has been motivated primarily by the desire to build a powerful computer to resolve a variety of problems that are still very difficult to handle using conventional digital computers. Examples of such problems are pattern recognition under real-world environments, fuzzy pattern matching, and nonlinear discrimination, where two functions are most important: (1) the associative property, the ability to recall; and (2) self-organizing, the ability to learn through organizing and reorganizing in response to external stimuli. Such humanlike performance will require an enormous amount of processing. To obtain the required processing capability, an effective approach should be developed for the dense interconnection of a large number of simple processing elements, an effective scheme for achieving high computational rates is required, and many hypotheses would need to be pursued simultaneously.

There is no doubt that vector quantization or grouping of inputs into clusters should be implemented in an artificial neural system for data compression without losing important information. As suggested by Lippman (1987), a taxonomy of five neural nets that can be used as pattern classifiers is shown below:

$$\begin{array}{l}\text{Neural network}\\\text{classifiers}\end{array}\left\{\begin{array}{l}\left.\begin{array}{l}\text{Hopfield net}\\\text{Hamming net}\\\text{Carpenter-Grossberg classifier}\end{array}\right\}\text{for binary inputs}\\\left.\begin{array}{l}\text{Multlayer perceptron}\\\text{Kohonen self-organizing}\\\quad\text{feature maps}\end{array}\right\}\begin{array}{l}\text{for continuous-}\\\quad\text{valued inputs}\end{array}\end{array}\right.$$

The Hopfield net, Hamming net, and Carpenter-Grossberg classifier have been proposed for binary inputs, while the perceptron and Kohonen self-organizing feature maps have been developed for continuous-value inputs. Some of these net models are discussed below. An artificial neural network can be taxonomized as feedforward and feedback recall based on their recall qualities. It can also be taxonomized as a supervised or an unsupervised learning based on their encoding qualities. Both supervised and unsupervised learning are part of the successful research results.

Applications currently on the neural networks can be found in:

1. Adaptive noise canceling in telecommunication
2. Mortgage risk evaluator
3. Bomb sniffer in airport
4. Process monitor on the production line tracking such things as variations in heat, pressure, and the chemicals used to make a product (e.g., bulbs)
5. Word recognizer
6. Blower motor noise checker
7. Airline marketing tactician working on optimizing airline seating and fee schedule
8. Etc.

It could be expected that lots of applications would pop up like spring flowers after rain. To name a few, in the area of environmental science and technology, ANNs are being applied to weather forecasting and trend analysis. In the financial area, ANNs are being used in credit risk assessment, forgeries identification, and handwritten forms interpretation. In manufacturing, use of ANN for process control, quality assessment, and parts selection on an assembly line are being investigated. Research is also underway in medical diagnosis and prescribing treatments from symptoms, in monitoring surgery, and in monitoring epileptic seizures. Aside from these, ANN will definitely play art important role in the military applications, for example, in recognizing and tracking targets, in reconnaissance, and in classifying radar signals and creating smart weapons.

It will not be difficult to see that most of potential applications have something to do with pattern recognition. Briefly speaking, in these applications we are searching for patterns from the environment and then classifying them, or we are reconstructing correct patterns from distorted ones and then reinterpreting them. It is also obvious that the applications mentioned above have something to do with perception and sensory datalike visual, auditory, infrared, and other signals. They exhibit behaviors characteristic of people rather than of conventional computers. What follows in this book will focus on the design of an artificial neural network for pattern recognition.

This Page Intentionally Left Blank

8

Multilayer Perceptron

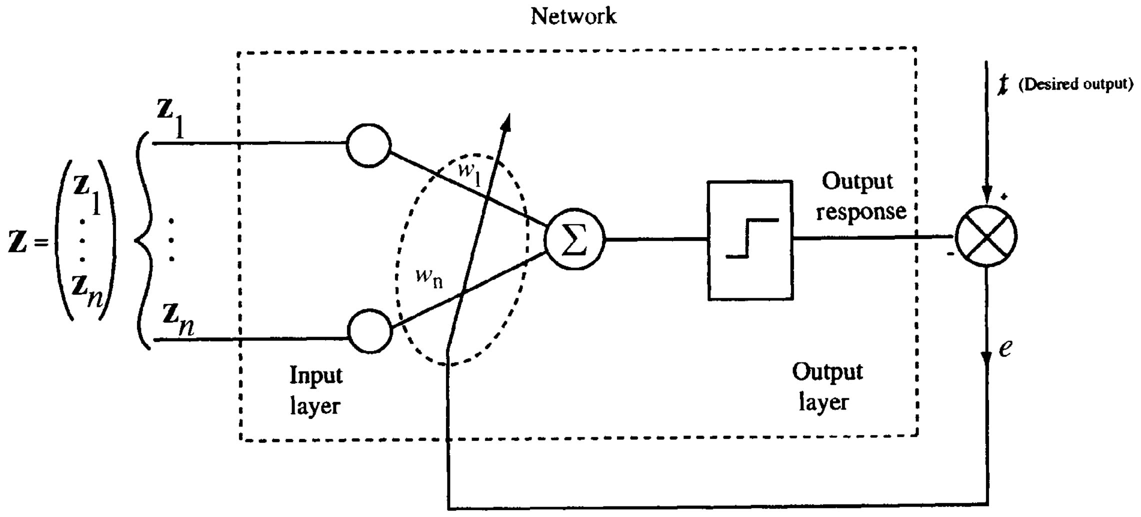
8.1 SOME PRELIMINARIES

Neural network is a very popular subject now. Many different models have been proposed. In this book the author does not intend to review all these models, but to select only some of them to illustrate the concepts of the neural networks and how to proceed from concepts to the solution of our real-world pattern recognition problems.

Perceptron is the earliest of the neural network paradigms. ADALINE (ADAptive LINear Elements) and MADALINE (Multiple ADALINEs in parallel) as discussed in Chapter 3 are examples of perceptron. They use the least-mean-square (LMS) algorithm for their system operations. Their implementation is comparatively simple and can be used for classification of patterns that are linearly separable.

In Chapter 4 we depicted a solution region for a two-class problem, and also formulated an error correction training procedure by the method of steepest descent, that is, moving the weight vector **W** in a direction perpendicular to the hyperplane (i.e., in the direction of $\mathbf{z}_1$ or $-\mathbf{z}_2$) where $\mathbf{z}_1 \in \omega_1$ and $\mathbf{z}_2 \in \omega_2$. Figure 8.1 shows a schematic diagram for the system that is self-explanatory.

Suppose that the system shown in Figure 8.1 has already been well trained. When $\mathbf{z}_1 \in \omega_1$ is presented to the system, output response of the system should be greater than zero; it is 1 after it is thresholded by a hard limiter. When $\mathbf{z}_2 \in \omega_2$



$\mathbf{w}^T = (w_1 w_2 \cdots w_n^T)$ $d(\mathbf{x}) = \mathbf{W}^T\mathbf{x}$

FIGURE 8.1 A linear two-class pattern classification system.

is presented, the output response of the system should be less than zero, and is thresholded to 0 according to the discriminant function described below:

$$d_{12}(\mathbf{x}) = \mathbf{W}^T\mathbf{z}$$

Otherwise, the system is still not well trained, and the weight vector $\mathbf{W} = (w_1 \quad w_2 \quad \cdots \quad w_n)^T$ should be adjusted according to Eqs. (8.1) to (8.3) to make $\mathbf{W}^T\mathbf{z}$ a proper value as expected

$$\mathbf{W}(k+1) = \mathbf{W}(k) + c\mathbf{z}_1 \qquad \text{if } W^T\mathbf{z}_1 < 0 \tag{8.1}$$

$$\mathbf{W}(k+1) = \mathbf{W}(k) - c\mathbf{z}_2 \qquad \text{if } W^T\mathbf{z}_2 > 0 \tag{8.2}$$

$$\mathbf{W}(k+1) = \mathbf{W}(k) \qquad \text{if correctly classified} \tag{8.3}$$

where $W(k)$ and $W(k+1)$ are, respectively, the weight vectors at the kth and $(k+1)$th correction steps. To add a correction term $c\mathbf{z}_1$ implies moving the weight vector in the direction of $\mathbf{z}_1$. Similarly, subtracting a correction term $c\mathbf{z}_2$ implies moving the weight vector $\mathbf{W}$ in the direction of $-\mathbf{z}_2$, where $\mathbf{z}_1$ and $\mathbf{z}_2$, respectively, belonging to ω_1 and ω_2.

For the case that the number of classes is greater than 2 (i.e., $M > 2$), similar procedures can be followed. However, it would be more convenient to use separate discriminant functions:

$$d_i(\mathbf{x}) = \mathbf{W}_i^T\mathbf{x} \qquad i = 1, 2, \ldots, M \tag{8.4}$$

We desire that

$$d_i(\mathbf{z}) > d_j(\mathbf{z}) \qquad \text{when } \mathbf{z} \in \omega_i, \forall j \neq i \tag{8.5}$$

If so, the weight vectors remains unchanged. But if $d_i(\mathbf{z}) < d_j(\mathbf{z})$ and $\mathbf{z}$ is known to be in ω_i for all $j \neq i$, misclassification occurs, and weight adjustments will be needed. Under these circumstances, the following adjustment can be made for the fixed increment rule:

$$\mathbf{W}_i(k+1) = \mathbf{W}_i(k) + c\mathbf{z} \qquad \text{if } d_i(\mathbf{z}) < d_j(\mathbf{z}) \tag{8.6}$$

$$\mathbf{W}_j(k+1) = \mathbf{W}_j(k) - c\mathbf{z} \qquad \text{if } d_j(\mathbf{z}) > d_i(\mathbf{z}) \tag{8.7}$$

$$\mathbf{W}_l(k+1) = \mathbf{W}_l(k) \qquad \text{if correctly classified} \tag{8.8}$$

Adjustment according to Eq. (8.6) is to increase $d_i(\mathbf{z})$ to make $d_i(\mathbf{z}) > d_j(\mathbf{z})$. Adjustment according to Eq. (8.7) is to decrease $d_j(\mathbf{z})$ so that $d_j(\mathbf{z})$ will be less than $d_i(\mathbf{z})$. Equation (8.8) implies that no adjustment on W_l is needed for those that did not make any incorrect classification. Figure 8.2 describes the scheme for

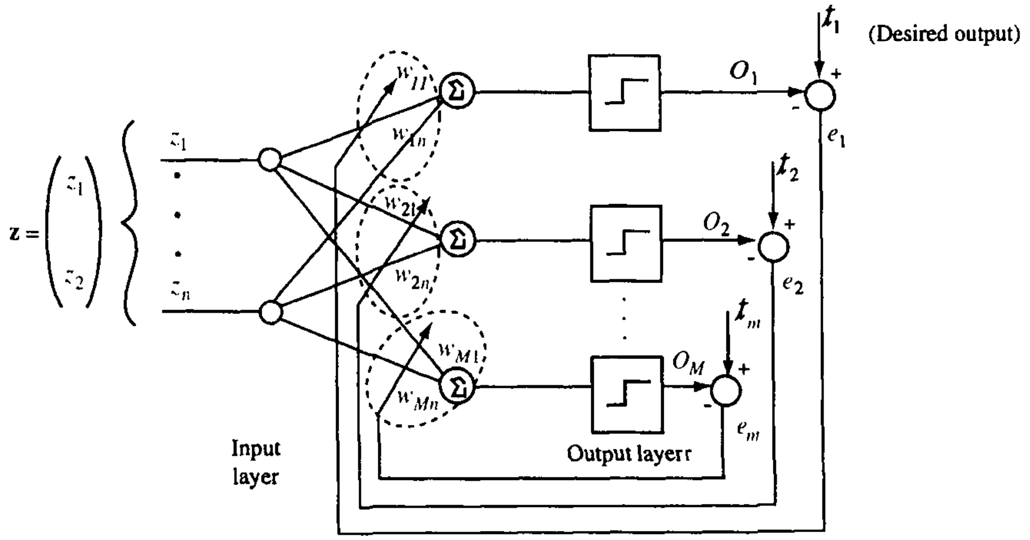


FIGURE 8.2 A linear pattern classification system for more than two classes.

$$\begin{array}{ll}
\mathbf{W}_1 = (w_{11} \quad w_{12} \quad \cdots \quad w_{1n})^T & d_1(\mathbf{z}) = \mathbf{W}_1^T\mathbf{z} \\
\mathbf{W}_2 = (w_{21} \quad w_{22} \quad \cdots \quad w_{2n})^T & d_2(\mathbf{z}) = \mathbf{W}_2^T\mathbf{z} \\
\vdots & \vdots \\
\mathbf{W}_M = (w_{M1} \quad w_{M2} \quad \cdots \quad w_{Mn})^T & d_M(\mathbf{z}) \quad \mathbf{W}_M^T\mathbf{z}
\end{array}$$

the above weight adjustments. We desire that outputs of the neurons in the output layer come out as

$$\left.\begin{array}{lll}
O_1 = \mathbf{W}_1^T\mathbf{z} > 0 & \text{or thresholded to} & 1 \\
O_2 = \mathbf{W}_2^T\mathbf{z} < 0 & \text{or thresholded to} & 0 \\
\vdots & & \vdots \\
O_M = \mathbf{W}_M^T\mathbf{z} < 0 & \text{or thresholded to} & 0
\end{array}\right\} \quad \text{if } \mathbf{z} \in \omega_1$$

$$\left.\begin{array}{lll}
O_1 = \mathbf{W}_1^T\mathbf{z} < 0 & \text{or thresholded to} & 0 \\
O_2 = \mathbf{W}_2^T\mathbf{z} > 0 & \text{or thresholded to} & 1 \\
\vdots & & \vdots \\
O_M = \mathbf{W}_M^T\mathbf{z} < 0 & \text{or thresholded to} & 0
\end{array}\right\} \quad \text{if } \mathbf{z} \in \omega_2$$

and

$$\left.\begin{array}{lll}
O_1 = \mathbf{W}_1^T\mathbf{z} < 0 & \text{or thresholded to} & 0 \\
O_2 = \mathbf{W}_2^T\mathbf{z} < 0 & \text{or thresholded to} & 0 \\
\vdots & & \vdots \\
O_M = \mathbf{W}_M^T\mathbf{z} > 0 & \text{or thresholded to} & 1
\end{array}\right\} \quad \text{if } \mathbf{z} \in \omega_M$$

where $\mathbf{W}_i^T = (w_{i1} \quad w_{i2} \cdots w_{in})^T$, $i = 1, 2, \ldots, M$. Note that Figures 8.1 and 8.2 are already in a neural network form, but with only input and output layers in the network. The network described above is called perceptron of its simplest form. It is used for linear classification problems only. To solve some more complicated and diverse problems, we have to go to perceptrons with one or multiple hidden layers in between the input and output of the system. This is known as multilayer perceptron. The powerful capability of the multilayer perceptron comes from the characteristics of its network arrangement. They are (1) one or more layers of hidden neurons used that are not part of the input or output of the network; (2) a smooth nonlinearity, e.g., sigmoidal nonlinearity, employed at the output end of each neuron; and (3) a high degree of connectivity in the network. These three distinct characteristics enable the multilayer perceptron to learn complex tasks by extracting more meaningful features from the input patterns. The reasons are obvious. A perceptron with only one input and output layer forms only half-plane decision regions. It then has the difficulty in differentiating classes within nested regions. Additional layers containing hidden units or nodes introduced will give smooth close contour-bound input distribution for two different classes. With a hidden layer in between the input and output layers of the perceptron, any convex region in the space can be formed. A perceptron with two hidden layers can then form arbitrary complex decision regions, and can therefore separate the meshed classes (see the workout example at the end of this chapter). In practice, no more than two hidden layers are required in a perceptron net.

8.2 PATTERN MAPPINGS IN A MULTILAYER PERCEPTRON

Figure 8.3 shows a three-layer perceptron with N continuous-valued inputs, and M outputs, which represent M output classes. Between the inputs and outputs are two hidden layers. y_l, $l = 1, 2, \ldots, M$, are the outputs of the multilayer perceptron, and x'_j and x''_k are the outputs of the nodes in the first and second hidden layers. θ'_j and θ''_k are the initial offsets (biases). W_{ji}, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, N_1$, are the weights from neurons in the input layer to those of the first hidden layer. They are to be adjusted during training. Similarly, W_{kj}, $j = 1, 2, \ldots, N_1$, $k = 1, 2, \ldots, N_2$ and W_{lk}, $k = 1, 2, \ldots, N_2$, $l = 1, 2, \ldots, M$, are, respectively, the weights connecting neurons in the first and those in the second hidden layers, and the weights connecting neurons in the second hidden layer and those of the output layer.

The outputs of the first hidden layer are computed according to

$$x'_j = f\left(\sum_{i=1}^{N} w_{ji}x_i - \theta_j\right) \qquad j = 1, 2, \ldots, N_1 \tag{8.9}$$

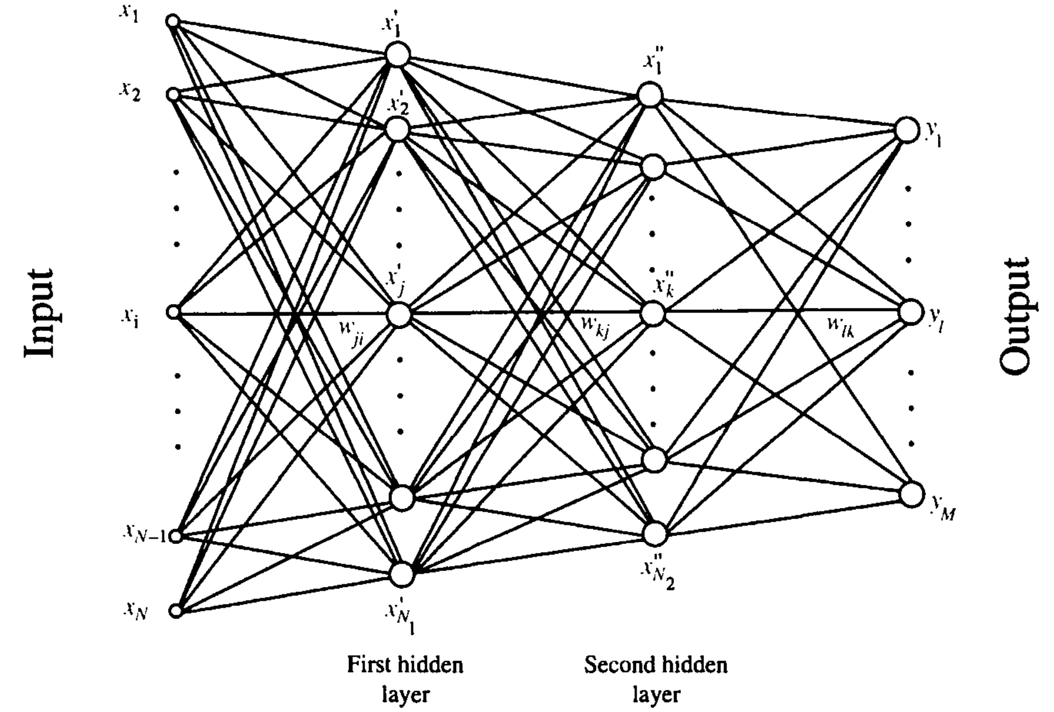


FIGURE 8.3 A multiplayer perception with two hidden layers.

Those of the second layer (also a hidden layer) are computed as:

$$x''_k = f\left(\sum_{j=1}^{N_1} w'_{kj} x'_j - \theta'_k\right) \qquad k = 1, 2, \ldots, N_2 \tag{8.10}$$

and the outputs of neurons in the output layer are

$$y_l = f\left(\sum_{k=1}^{N_2} w'_{lk} x''_k - \theta''_l\right) \qquad l = 1, 2, \ldots, M \tag{8.11}$$

The decision rule is to select that class which corresponds to the output node with the largest value. The f's in the computations above can be either a hard limiter, threshold logic, or sigmoid logistic nonlinearity, which is $1/(1 + e^{-(\alpha-\theta)})$.

Without doubt, to train such a perceptron with multilayers is much more complicated than that for a simple perceptron (i.e., a perceptron with only input and output layers). This is because when there exists an output error, it is hard to know how much error comes from this input node, how much come from others, and how to adjust the synapses (weights) according to their respective contributions to the output error. To solve such a problem, we have to find out the effects of all the synapses (weights) in the network. This is a backward process. The back-propagation algorithm is actually a generalization of the least-mean-square

(LMS) algorithm. It uses an iterative gradient technique to minimize the mean-square error between the desired output and the actual output of a multilayer feedforward perceptron. The training procedure initializes by selecting small random weights and internal thresholds. Training samples are repeatedly presented to the net. Weights are adjusted until they stabilize. At that time the cost function (or the mean-square error as mentioned above) is reduced to an acceptable value. In brief, the whole sequence involves two phases: a forward phase and a backward phase. In the forward phase the error is estimated, while in the backward phase weights are modified to decrease the error.

8.2.1 Weight Adjustment in Backward Direction Based on Proper Share of each Processing Element—A Hypothetical Example for Explanation

Let us take a very simple hypothetical example to explain how can we adjust the weights in the backward direction based on the system output error and the proper share of each processing element (PE) on the total error. Figure 8.4 shows the hypothetical network for this purpose.

In the present example, there are three layers. The input layer, or the first layer of this network, serves as the holding site for values applied to the network. It holds the input values and distributes these values to all units in the next layer.

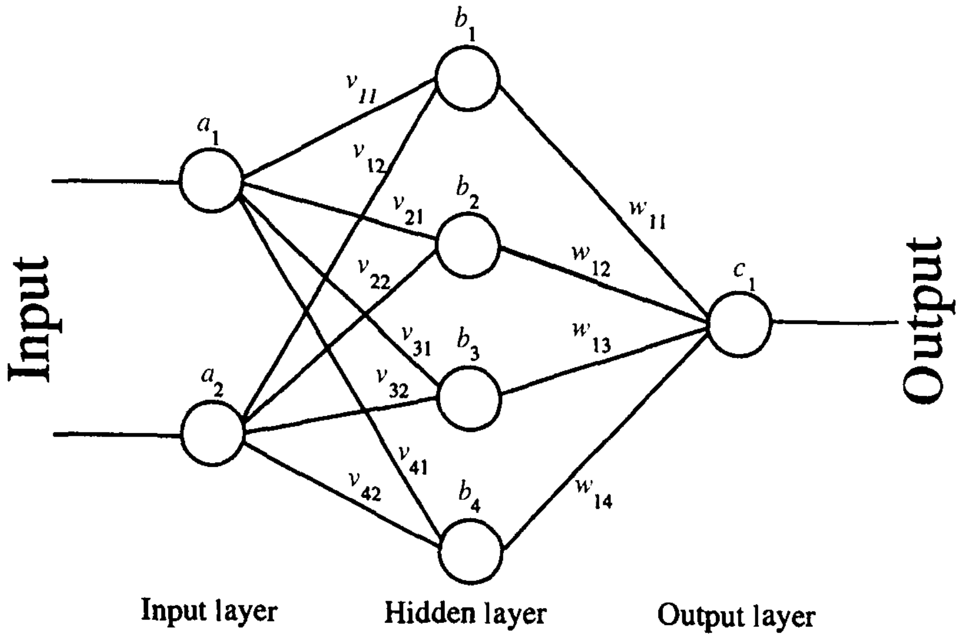


FIGURE 8.4 A simple hypothetical perceptron network.

The output layer, or the last layer of the system, is the point at which the final state of the network is read. The layer between the input and the output layers is the hidden unit (layer).

In this figure, a_1 and a_2 are the processing elements (PEs) in the input layer; b_1, b_2, b_3, and b_4 are the PEs in the hidden layer; and c is the PE in the output layer. v_{11}, v_{12}; v_{21}, v_{22}, v_{31}, v_{32}; and v_{41}, v_{42} are the weights (synapses) connecting PEs in the input layer and those in the hidden layer. w_{11}, w_{12}, w_{13}, and w_{14} are the weights connecting the PEs of the hidden layer and that of the output layer. For brevity, let us just use a_1 and a_2 as the outputs of PEs of the input layer; b_1, b_2, b_3, and b_4 as the outputs of the PBs of the hidden layer; and c as the output of the PE of the output layer.

This network is supposed to function as a pattern associator through training. It should have the ability to learn pattern mappings. Training is accomplished by presenting the pattern to be classified to the perceptron network and determining its output. The actual output of the network is compared with a desired output (or called "target" output) and an error message is calculated. The error measure is then propagated backward through the network and used to determine weight changes within the network. The purpose of the network weight adjustment is to minimize the system output error, and the minimization of this error is done at each stage through the weight adjustment. This process is repeated until the network reaches a desired state of response.

To start with, let us randomly choose values for the weights as listed below and proceed to adjust these weights backward from the output layer

$$
\begin{aligned}
w_{11} &= 1.25 \\
w_{12} &= 1.50 \\
w_{13} &= 4.00 \\
w_{14} &= 3.50 \\
v_{11} &= 1.00 \\
v_{12} &= -1.50 \\
v_{21} &= 3.50 \\
v_{22} &= -4.50 \\
v_{31} &= 2.10 \\
v_{32} &= 2.50 \\
v_{41} &= 1.00 \\
v_{42} &= -1.00
\end{aligned}
$$

Assume that we are given the following two prototypes; one belonging to class 1, and the other belonging to class 2:

$$(a_1, a_2) = (0, 1) \in \text{class } 1$$

and

$$(a_1, a_2) = (1, 0) \in \text{class } 2$$

For this two-class problem, the output of the system should be 1, if a pattern belonging to class 1 (i.e., the first pattern) is presented to the system. When a pattern belonging to class 2 (the second pattern) is presented to the system, the output response of the system should be 0. Let us now present the pattern $(a_1, a_2) = (0, 1)$ to the system. The desired output t of the system in this case should be 1, because this pattern is known belonging to class 1. Let us proceed to compute the actual output of the system with this known information. Using the notation as suggested in the previous paragraph, the output of the processing element a_1 is a_1, and that of the processing element a_2 is a_2. With the same notation, the output of the PEs, b_1, b_2, b_3, and b_4, are, respectively:

$$\begin{aligned}
b_1 &= v_{11} \times a_1 + v_{12} \times a_2 = 1.00 \times 0 + (-1.50) \times 1 = -1.50 \\
b_2 &= v_{21} \times a_1 + v_{22} \times a_2 = 3.50 \times 0 + (-4.50) \times 1 = -4.50 \\
b_3 &= v_{31} \times a_1 + v_{32} \times a_2 = 2.10 \times 0 + 2.50 \times 1 = 2.50 \\
b_4 &= v_{41} \times a_1 + v_{42} \times a_2 = 1.00 \times 0 + (-1.00) \times 1 = -1.00
\end{aligned}$$

The actual output of the system c is

$$\begin{aligned}
c &= w_{11}b_1 + w_{12}b_2 + w_{13}b_3 + w_{14}b_4 \\
&= 1.25 \times (-1.50) + 1.50 \times (-4.50) + 4.00 \times 2.50 + 3.50 \times (-1.00) \\
&= -1.88 - 6.75 + 10 - 3.50 = -2.13
\end{aligned}$$

The error for the network output e is equal to the difference between the desired output t and the actual output c, or

$$e = t - c = 1 - (-2.13) = 3.13$$

This error should be properly shared by these processing elements b_1, b_2, b_3, and b_4 of the hidden layer. The shares for those processing elements, namely b_1, b_2, b_3, and b_4, are, respectively,

$$\begin{aligned}
e_{b1} &= w_{11}e = 1.25 \times 3.13 = 3.91 \\
e_{b2} &= w_{12}e = 1.50 \times 3.13 = 4.70 \\
e_{b3} &= w_{13}e = 4.00 \times 3.13 = 12.52 \\
e_{b4} &= w_{14}e = 3.50 \times 3.13 = 10.96
\end{aligned}$$

From these values of shared error we can compute the new values for each of the 12 weights. Thus we obtain

$$
\begin{aligned}
w_{11}(1) &= w_{11}(0) + (t - c) \times c = 1.25 + 3.13 \times (-2.13) = -5.42 \\
w_{12}(1) &= w_{12}(0) + (t - c) \times c = 1.50 + 3.13 \times (-2.13) = -5.17 \\
w_{13}(1) &= w_{13}(0) + (t - c) \times c = 4.00 + 3.13 \times (-2.13) = -2.67 \\
w_{14}(1) &= w_{14}(0) + (t - c) \times c = 3.50 + 3.13 \times (-2.13) = -3.17
\end{aligned}
$$

where $(t - c)$ is the system error, and c is the output of the system

$$
\begin{aligned}
v_{11}(1) &= v_{11}(0) + e_{b1}b_1 = 1.00 + 3.91 \times (-1.50) = -4.87 \\
v_{12}(1) &= v_{12}(0) + e_{b1}b_1 = -1.50 + 3.91 \times (-1.50) = -7.37 \\
v_{21}(1) &= v_{21}(0) + e_{b2}b_2 = 3.50 + 4.70 \times (-4.50) = -17.65 \\
v_{22}(1) &= v_{22}(0) + e_{b2}b_2 = -4.50 + 4.70 \times (-4.50) = -25.65 \\
v_{31}(1) &= v_{31}(0) + e_{b3}b_3 = 2.10 + 12.52 \times (2.50) = 33.40 \\
v_{32}(1) &= v_{32}(0) + e_{b3}b_3 = 2.50 + 12.52 \times (2.50) = 33.80 \\
v_{41}(1) &= v_{41}(0) + e_{b4}b_4 = 1.00 + 10.96 \times (-1.00) = -9.96 \\
v_{42}(1) &= v_{42}(0) + e_{b4}b_4 = -1.00 + 10.96 \times (-1.00) = -11.96
\end{aligned}
$$

where e_{bi}, $i = 1, 2, 3$ and 4, are the errors shared by the four neurons in the hidden layer, namely, b_1, b_2, b_3, and b_4; and b_1, b_2, b_3, and b_4, as mentioned previously, represent respectively outputs of the neurons b_1, b_2, b_3, and b_4. The weights just computed for the first iteration (i.e., through one feed forward and one backward pass) compared with their original values resulting from random selection are tabulated in the following chart:

Processing elements	Weight	Original value	Adjusted value
b_1	v_{11}	1.00	−4.87
b_1	v_{12}	−1.50	−7.37
b_2	v_{21}	3.50	−17.65
b_2	v_{22}	−4.50	−25.65
b_3	v_{31}	2.10	33.40
b_3	v_{32}	2.50	33.80
b_4	v_{41}	1.00	−9.96
b_4	v_{42}	−1.00	−11.96
c_1	w_{11}	1.25	−5.42
c_1	w_{12}	1.50	−5.17
c_1	w_{13}	4.00	−2.67
c_1	w_{14}	3.50	−3.17

There may be numerous iterations needed to satisfactorily train the network. Each iteration requires all the calculations as shown above.

After this pattern has been correctly classified as belonging to class 1, then the second pattern $(a_1, a_2) = (1, 0)$ is presented to the system. The desired output of the system now should be 0, since this pattern is known belonging to class 2. The same comutation procedure will follow. As before, there may be many iterations needed again to satisfactorily train the system to give output response 0. When the system makes no misclassification on both these two prototypes, the system can then be said trained.

Even for this very simple problem, lots of computations are needed. For a real-world problem the complexity of computation is obvious. Manual computation will definitely not be a possible solution and help from modern computer to perform the tedious computation is needed. What follows will be the development of an algorithm for the backpropagation training for the system.

8.2.2 Derivation of the Back-Propagation Algorithm

The development of the back-propagation learning algorithm provides a computational efficient method for the training of a multilayer perceptrons. The term backpropagation, which is based on the error correction learning rule, appears to have evolved after 1985. The back-propagation algorithm derives its name from the fact that partial derivatives of the performance measure with respect to the synaptic weights and biases of the network are determined by back-propagating the error signals through the network layer by layer.

Figure 8.3 shows a multilayer perceptron network with two hidden layers. The error back-propagation learning consists of two passes: a forward pass and a backward pass. In the forward pass, the input signal propagates through the network on a layer-by-layer basis, and a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights are all kept fixed. An error signal is then obtained from the difference between the actual output response of the network and the desired (or target) response. This error signal is then propagated backward through the network, against the direction of the synaptic connections, and the synaptic weights are so adjusted to minimize the output error (i.e., to make the actual output response close to the desired value). This is what we called the backward pass. To perform this minimization, weights are updated on a pattern-by-pattern basis. The adjustments to the weights are made in accordance with the respective errors computed for each pattern presented to the network.

As before, we use the gradient descent approach for the error back propagation for the multilayer perceptron. This involves two processes:

1. Based on the mapping error E, compute the gradient of E with respect to w_{ji}, $\delta E/\delta w_{ji}$, where w_{ji} refers to the synapse (or weight) connecting the processing element j of this layer (say the Lth layer) to the

processing element i of the nearest previous layer (say the $(L-1)$st layer).

2. Adjust the weights with an increment which is proportional to $-\delta E/\delta w_{ji}$, i.e.,

$$\Delta w_{ji} \propto -\frac{\delta E}{\delta w_{ji}}$$

The negative sign in front of $\delta E/\delta w_{ji}$ signifies that the change in each weight, Δw_{ji}, will be along the negative gradient descent which leads to a steepest descent along the local error surface. Adjusting the weight with this Δw_{ji} will result in a step in the weight space toward lower error.

Assume that we are given a training set H:

$$H = \{(\mathbf{i}^k, \mathbf{t}^k)\} \qquad k = 1, 2, \ldots, n \tag{8.12}$$

where $\mathbf{i}^k$ is the kth input pattern vector presented to the system. $\mathbf{t}^k$ is the desired (or the target) vector, which corresponds to the kth input vector $\mathbf{i}^k$. ($\mathbf{i}^k$, $\mathbf{t}^k$) forms the kth pair of input/output vectors. Let i_i^k be the ith component of the kth input pattern vector $\mathbf{i}^k$, or

$$i_i^k \in \mathbf{i}^k \tag{8.13}$$

and let O_j^k be the jth component of the kth actual (or computed) output vector $\mathbf{O}^k$, or

$$O_j^k \in \mathbf{O}^k \tag{8.14}$$

Similarly, t_j^k is the jth component of the corresponding kth target vector $\mathbf{t}^k$.

Let us now proceed to do the individual weight correction with a prespecified input training pattern $\mathbf{i}^k$. Starting with the output layer, we have

$$\mathbf{e}^k = \mathbf{t}^k - \mathbf{O}^k \tag{8.15}$$

where $\mathbf{e}^k$ is the output vector when the kth input vector $\mathbf{i}^k$ is presented. Chosse mean-squared error as the performance criterion function $\mathbf{J}$,

$$\mathbf{J} = \tfrac{1}{2}(\mathbf{e}^k)^T\mathbf{e}^k = \tfrac{1}{2}|\mathbf{e}^k|^2 \tag{8.16}$$

Then we can write

$$\mathbf{E}^k = \tfrac{1}{2}|\mathbf{t}^k - \mathbf{O}^k|^2 \tag{8.17}$$

or

$$\mathbf{E}^k = \tfrac{1}{2}\sum_{j=1}^{M}(t_j^k - O_j^k)^2 \tag{8.18}$$

where $|\mathbf{t}^k - \mathbf{O}^k|^2$ is the measure of the "distance" between the desired and actual outputs of the network, and $\frac{1}{2}\sum_{j=1}^{M}(t_j^k - O_j^k)^2$ is the sum of the squared errors. The constant "$\frac{1}{2}$" added is to simplify the derivation which follows. Note that each item in the sum is the error contribution of a single output neuron.

The basic idea of backpropagation is to propagate the error backward through the network. Each neuron in the output layer adjusts its weights, which, respectively, connect it to the neurons in the nearest previous hidden layer, in proportion to the error contribution of the individual neuron in the previous layer. This applies to all the neurons in the output layer. After these are done, each of the neurons in the nearest previous hidden layer will follow the same way to propagate the share of the allocated error to each of the weights which connect it to each of the neurons in the next previous hidden layer and adjust them. So on and so forth until the previous layer is the input layer. This constitutes an iteration. By completing this iteration, all the weights in network will be adjusted according to their proper shares to the output error. There will be many iterations in the training process. It can therefore be easily seen that there are lots of adjustments needed to be done on the weights (synapses). So, let us analyze the network in more details.

As in a single layer perceptron, each neuron in a multilayer perceptron is also modeled as $f_j[\sum_i(w_{ji}O_i + \text{bias})]$ or $f_j(\text{net}_j)$, except that the activation function is a sigmoidal function instead of a hard limiter. Note that O_i here refers to the output of the neuron in the previous layer. It is also the input to this neuron. For the jth neuron of the output layer, the activation function is of the following form:

$$O_j^k = f_j(\text{net}_j^k) = f_j\left(\sum_i w_{ji}O_i^k + \theta_j\right) \tag{8.19}$$

where O_j^k is the output of the jth neuron unit in the output layer, O_i^k is the input to the particular jth neuron in the output layer from the ith neuron of the previous hidden layer, and θ_j is the bias. $f(.)$ is an activation function, which is kept unchanged for the presentation of the various patterns. Let us choose

$$f(\text{net}_j) = \frac{1}{1 + \exp(-\text{net}_j)} \tag{8.20}$$

and

$$0 \leq f(\text{net}_j) \leq 1$$

as the activation function. It is a nondecreasing and differentiable function called sigmoidal activation function. Remember that when pattern k is presented to the multiplayer perceptron system,

$$E^k = \frac{1}{2}\sum_{j=1}^{M}(t_j^k - O_j^k)^2 \tag{8.21}$$

As mentioned before, the gradient descent approach can be used to minimize E_k through the weight adjustment, i.e.,

$$\Delta w_{ji} \propto -\frac{\partial E^k}{\partial w_{ji}} \tag{8.22}$$

Figure 8.5 shows the signal flow graph highlighting the details of output neuron j. To minimize the error E_k, take the partial derivative of E_k with respect to w_{ji},

$$\frac{\partial E^k}{\partial w_{ji}} = \frac{\partial E^k}{\partial O_j^k} \cdot \frac{\partial O_j^k}{\partial w_{ji}} \tag{8.23}$$

where $\partial E_k / \partial O_j^k$ represents the effect on E^k due to the jth neuron of the output layer and $\partial O_j^k / \partial w_{ji}$ measures the change on O_j^k as a function of w_{ji}. Recall that the output O_j^k is directly a function of net_j, or

$$O_j^k = f(\text{net}_j) \tag{8.24}$$

and

$$\frac{\partial O_j^k}{\partial \text{net}_j} = f'(\text{net}_j) \tag{8.25}$$

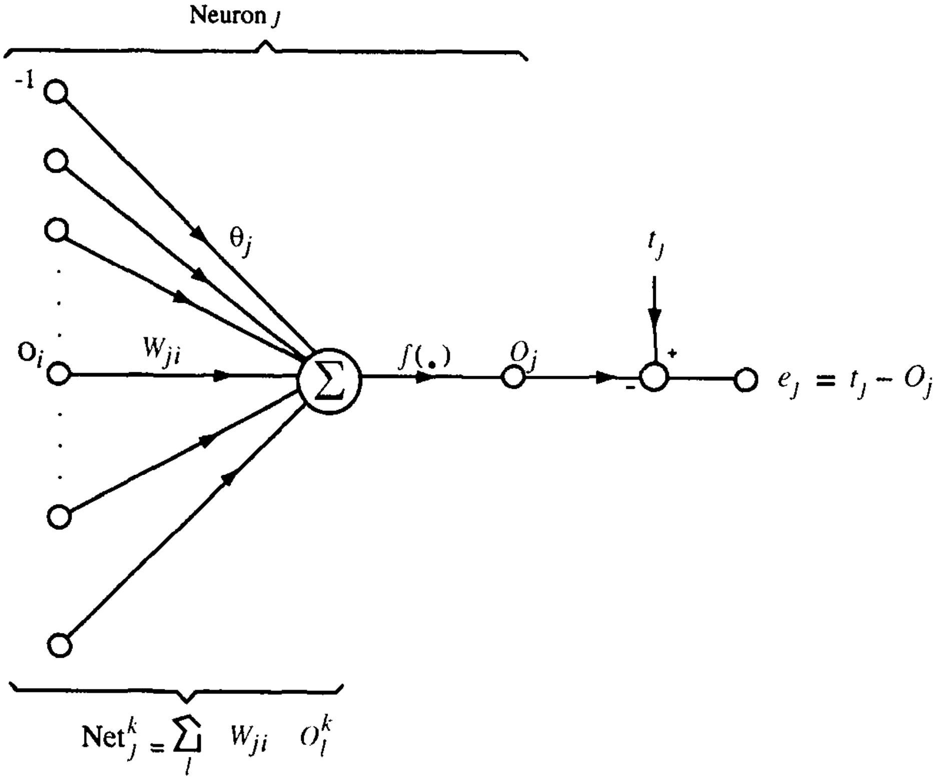


FIGURE 8.5 Highlight on the output neuron j in a perceptron pattern classification system.

$\partial E_k / \partial w_{ji}$ then becomes

$$\frac{\partial E^k}{\partial w_{ji}} = \frac{\partial E^k}{\partial O_j^k} \cdot \frac{\partial O_j^k}{\partial \text{ net}_j} \cdot \frac{\partial \text{ net}_j}{\partial w_{ji}} \tag{8.26}$$

In general, net_j can be put as:

$$\text{net}_j = \sum_l w_{jl} O_l^k \tag{8.27}$$

and O_l^k is the output of a neuron in a previous nearest layer (i.e., the second hidden layer) and it is the input to this neuron (the neuron j). This is true for the output layer and is also true for the hidden layer. If this input is a direct input to the network (i.e., from the input layer), it then becomes $O_l^k = i_l^k$. Taking partial derivative of net_j with respect to w_{ji} gives

$$\frac{\partial \text{ net}_j}{\partial w_{ji}} = \frac{\partial(\sum_l w_{jl} O_l^k)}{\partial w_{ji}} \tag{8.28}$$

or
$$= \frac{\partial(\sum_{l' \neq i} w_{jl'} O_{l'}^k + w_{ji} O_i)}{\partial w_{ji}} \tag{8.29}$$

or
$$= O_i^k \tag{8.30}$$

Substitution of Eq. (8.30) into (8.26) gives

$$\frac{\partial E^k}{\partial w_{ji}} = \frac{\partial E^k}{\partial O_j^k} \cdot \frac{\partial O_j^k}{\partial \text{ net}_j} \cdot O_i^k \tag{8.31}$$

or
$$= \frac{\partial E^k}{\partial \text{ net}_j} \cdot O_i^k$$

where $\partial E^k / \partial \text{ net}_j = (\partial E^k / \partial O_j^k) \cdot (\partial O_j^k / \partial \text{ net}_j)$ is the sensitivity of the pattern error on the activation of the jth unit. Let us define the error signal σ_j^k as

$$\sigma_j^k = -\frac{\partial E^k}{\partial \text{ net}_j} \tag{8.33}$$

Substitution of σ_j^k for $-\partial E^k / \partial \text{ net}_j$ in Eq. (8.32) gives

$$\frac{\partial E^k}{\partial w_{ji}} = -(\sigma_j^k) O_i^k \tag{8.34}$$

We can then apply the gradient descent approach to adjust w_{ji}. The weight adjustment $\Delta^k w_{ji}$ is then

$$\Delta^k w_{ji} = -\eta \frac{\partial E^k}{\partial w_{ji}} = \eta \sigma_j^k O_i^k \tag{8.35}$$

η in (8.35) is a positive constant used to control the learning rate.

In the context, we may identify two distinct cases depending on where in the network neuron j is located. In case 1, neuron j is an output node, while in case 2, neuron j is a hidden node.

Case 1: When neuron j *is an output node.* Since net_j^k remains unchanged for all input patterns, we can leave the superscript k out from net_j. Express σ_j^k by chain rule as

$$\sigma_j^k = -\frac{\partial E^k}{\partial\,\mathrm{net}_j} = -\frac{\partial E^k}{\partial O_j^k}\cdot\frac{\partial O_j^k}{\partial\,\mathrm{net}_j} \tag{8.36}$$

Since $O_j^k = f(\mathrm{net}_j)$, we then have

$$\frac{\partial O_j^k}{\partial\,\mathrm{net}_j} = f'(\mathrm{net}_j) \tag{8.37}$$

and

$$\begin{aligned} f'(\mathrm{net}_j) &= \frac{\partial}{\partial\,\mathrm{net}_j}\left(\frac{1}{1+\exp(-\mathrm{net}_j)}\right) \\ &= \frac{1}{[1+\exp(-\mathrm{net}_j)]^2}[\exp(-\mathrm{net}_j)] \\ &= O_j^k(1-O_j^k) \end{aligned} \tag{8.38}$$

Subscription of the $f'(\mathrm{net}_j)$ into the expression for σ_j^k [Eq. (8.36)] gives

$$\begin{aligned} \sigma_j^k &= -\frac{\partial E^k}{\partial O_j^k}\cdot\frac{\partial O_j^k}{\partial\,\mathrm{net}_j} \\ &= -\frac{\partial E^k}{\partial O_j^k}O_j^k(1-O_j^k) \end{aligned} \tag{8.39}$$

Remember that the neuron j in this case is an output node,

$$E^k = \tfrac{1}{2}\sum_{j=1}^{M}(t_j^k - O_j^k)^2 \tag{8.40}$$

and

$$\frac{\partial E^k}{\partial O_j^k} = -(t_j^k - O_j^k) \tag{8.41}$$

By substituting Eq. (8.41) into Eq. (8.39), we obtain an expression to compute the σ_j^k for the neuron j of the output layer:

$$\sigma_j^k = (t_j^k - O_j^k)O_j^k(1 - O_j^k) \tag{8.42}$$

The increment adjustment of the weight $\Delta^k w_{ji}$, which connects the neuron j of the output layer to the neuron i of the nearest previous hidden layer (i.e., the second hidden layer), for the kth presented pattern can be computed by

$$\Delta^k w_{ji} = \eta\sigma_j^k O_i^k \tag{8.43}$$

or

$$\Delta^k w_{ji} = \eta(t_j^k - O_j^k)O_j^k(1 - O_j^k)O_i^k \tag{8.44}$$

Case 2: When neuron j *is a hidden node* Even though hidden neurons are not directly accessible, they share responsibility for any error made at the output of the network. We should penalize or reward hidden neurons for their share of the responsibility. Figure 8.6 shows the signal flow graph highlighting the details of output neuron m connected to hidden neuron j. From this figure it is obvious that we cannot differentiate the error function directly with respect to the output of the jth neuron O_j^k. We have to apply chain rule again. From Figure 8.6 we can see that the output at net_m is

$$\text{net}_m = \sum_l w_{ml}O_l \tag{8.45}$$

When we propagate the error backward, we are to find the effect of E^k due to O_j^k. Take the partial derivative of E^k with respect to O_j^k.

$$\frac{\partial E^k}{\partial O_j} = \sum_m \frac{\partial E^k}{\partial\, \text{net}_m} \cdot \frac{\partial\, \text{net}_m}{\partial O_j} \tag{8.46}$$

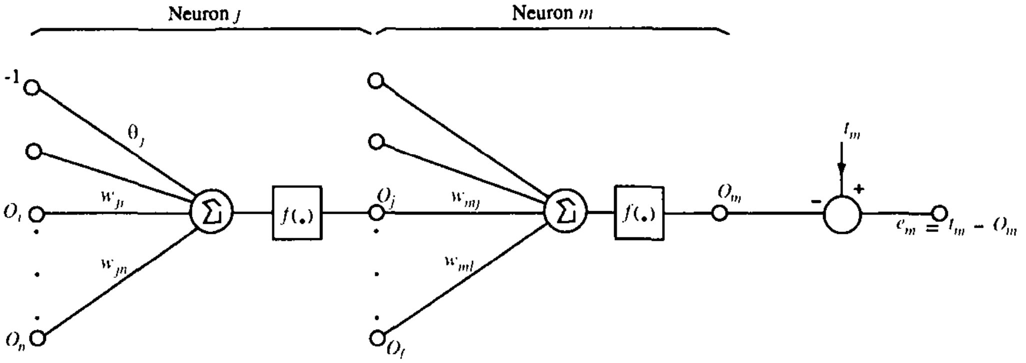


FIGURE 8.6 Highlight on the output neuron m connected to hidden neuron j in a multilayer perceptron pattern classification system.

where

$$\frac{\partial \text{ net}_m}{\partial O_j^k} = \frac{\partial[\sum_l w_m O_l^k]}{\partial O_j^k} \qquad l = 1, 2, \ldots$$
$$= w_{mj} \tag{8.47}$$

This is because all the other products $w_{ml}O_l$ equals zero when $l \neq j$. Therefore,

$$\frac{\partial E_k}{\partial O_j^k} = \sum_m \frac{\partial E^k}{\partial \text{ net}_m} \cdot w_{mj} \tag{8.48}$$

Using the similar notation as that for the σ_j^k [see Eq. (8.33)], we can define

$$\sigma_m^k = -\frac{\partial E^k}{\partial \text{ net}_m} \tag{8.49}$$

So, we obtain

$$\frac{\partial E^k}{\partial O_j^k} = -\sum_m \sigma_m^k \cdot w_{mj} \tag{8.50}$$

Combining Eqs. (8.36), (8.37), and (8.50) and with net_j replaced by net_m, we obtain

$$\sigma_j^k = -\frac{\partial E^k}{\partial O_j^k} \cdot \frac{\partial O_j^k}{\partial \text{ net}_m}$$
$$= \left(\sum_m \sigma_m^k \cdot w_{mj}\right) \cdot f'(\text{net}_m) \tag{8.51}$$

As mentioned previously, $f'(\text{net}_m)$ is the same as $f'(\text{net}_j)$. It has been derived before as $O_j^k(1 - O_j^k)$ [see Eq. (8.38)]. So, the expression for the error signal σ_j^k for a neuron in a hidden layer is

$$\sigma_j^k = O_j^k(1 - O_j^k)\left(\sum_m \sigma_m^k \cdot w_{mj}\right) \tag{8.52}$$

So far, we have analyzed these two possible cases. (1) When the neuron j is a neuron in the output layer, we have a direct access to the error E^k. We therefore can find the error signal σ_j^k, the sensitivity of the error on the activation of the jth unit:

$$\sigma_j^k = (t_j^k - O_j^k)O_j^k(1 - O_j^k) \tag{8.53}$$

(2) However, when the neuron is a neuron in a hidden layer, the error E^k is not a direct function O_j^k. And so we cannot directly differentiate the error function with respect to O_j^k and we have to apply the chain rule as shown in the above derivation [see Eqs. (8.45) to (8.52)], and σ_j^k is in the form of

$$\sigma_j^k = O_j^k(1 - O_j^k)\left(\sum_m \sigma_m^k \cdot w_{mj}\right) \tag{8.54}$$

What we have discussed can be summarized as: The increment adjustment of the weight $\Delta^k w_{ji}$, which connects the neuron j of the second hidden layer to the neuron i of the first hidden layer for the kth presented pattern, can be computed by

$$\Delta^k w_{ji} = \eta \sigma_j^k O_i^k \tag{8.55}$$

or

$$\Delta^k w_{ji} = \eta O_j^k(1 - O_j^k)\left(\sum_m \sigma_m^k w_{mj}\right) O_i^k \tag{8.56}$$

A flowchart for the back-propagation training learning of a multilayer perceptron is shown in Fig. 8.7 and is self-explanatory.

8.3 A PRIMITIVE EXAMPLE

The flowchart shown in Figure 8.7 is used to illustrate the whole process of error back-propagation learning, including both forward and backward passes. Initially, weights are set randomly with small values. When a pattern is presented to the system, the forward pass will compute a set of values at the output of neurons in the output layer. Errors will then be evaluated by comparing these output values with their desired values from the training set. Error back propagation then follows to establish a new set of weights. These two passes constitute an iteration. There may be numerous iterations for one pattern presentation, and the output error should decrease over the course of such iterations. The same process will repeat for all the prototypes in the training set in any order. When the prototypes in the training set have all been presented, and the weights approach values such that the output error falls below a preset value, the network is said to be trained.

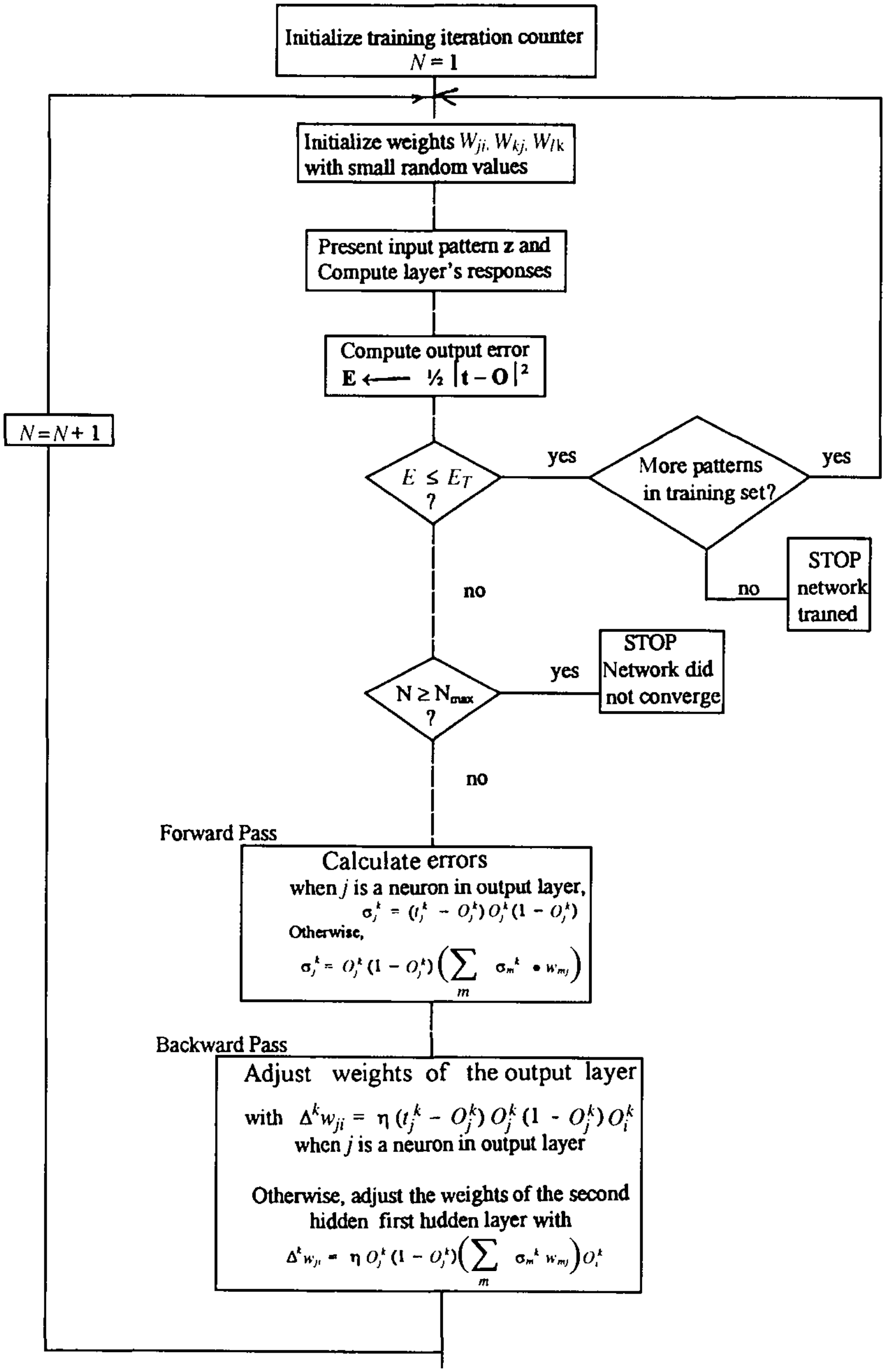

Initialize training iteration counter $N = 1$
Initialize weights W_{ji}, W_{kj}, W_{lk} with small random values
Present input pattern z and Compute layer's responses
Compute output error $\mathbf{E} \longleftarrow \frac{1}{2} |\mathbf{t} - \mathbf{O}|^2$
$N = N + 1$
$E \leq E_T$?
yes
More patterns in training set?
yes
no
STOP network trained
no
STOP Network did not converge
yes
$\mathrm{N} \geq \mathrm{N}_{max}$?
no
Forward Pass
Calculate errors
when j is a neuron in output layer,
$\sigma_j^k = (t_j^k - O_j^k) O_j^k (1 - O_j^k)$
Otherwise,
$\sigma_j^k = O_j^k (1 - O_j^k) \left(\sum_m \sigma_m^k \bullet w_{mj} \right)$
Backward Pass
Adjust weights of the output layer
with $\Delta^k w_{ji} = \eta (t_j^k - O_j^k) O_j^k (1 - O_j^k) O_i^k$
when j is a neuron in output layer
Otherwise, adjust the weights of the second hidden first hidden layer with
$\Delta^k w_{ji} = \eta O_j^k (1 - O_j^k) \left(\sum_m \sigma_m^k w_{mj} \right) O_i^k$

FIGURE 8.7 A flowchart for the back-propagation learning of a multiplayer perceptron.

Example. Given that the following 50 pattern sample points belong to class 1 (ω_1):

(2, 12); (3, 10); (3, 15); (4, 13); (4, 17); (5, 9); (5, 12); (5, 16); (6, 10);
(6, 15); (6, 19); (7, 9); (7, 11); (7, 17); (7, 20); (8, 13); (8, 15); (8, 21);
(9, 11); (9, 16); (9, 19); (10, 10), (10, 14); (10, 20); (11, 12); (11, 17);
(11, 19); (11, 22); (12, 11); (12, 13); (12, 15); (12, 21); (13, 18);
(14, 13); (14, 16); (14, 19); (14, 21); (15, 20); (15, 23); (16, 14);
(16, 16); (16, 18); (16, 22); (17, 21); (17, 15); (18, 18); (18, 20);
(18, 23); (19, 19); (20, 22)

and the following 52 pattern sample points belong to class 2 (ω_2):

(9, 3); (9, 5); (10, 7); (11, 4); (12, 5); (12, 8); (13, 3); (13, 9); (14, 7);
(15, 2); (15, 5); (15, 10); (16, 7); (17, 4); (17, 9); (17, 12); (18, 6);
(19, 8); (19, 11); (19, 14); (20, 4); (20, 12); (21, 7); (21, 10); (21, 16);
(22, 6); (22, 14); (22, 18); (23, 9); (23, 12); (23, 17); (24, 7); (24 15);
(24, 19); (25, 11); (25, 13); (25, 18); (26, 9); (26, 16); (26, 22);
(27, 14); (27, 19); (28, 12); (28, 17); (28, 21); (29, 15); (29, 19);
(29, 22), (30, 18), (30, 21), (31, 17); (31, 20)

use multiplayer perceptron with one hidden layer to separate the above two sets of data. Write a program in C or C + + for this problem.

Solution. The perceptron network with one hidden layer is shown in Figure 8.4. The initial values of w_{ji}, $j = 1$, $i = 1, 2, 3, 4$, were chosen as

$$w_{11} = 1.0; \; w_{12} = 0.45; \; w_{13} = 0.76; \; w_{14} = 0.98$$

The initial values of v_{11}, v_{12}, v_{21}, v_{22}, v_{31}, v_{32}, v_{41}, and v_{42} were chosen as:

$$\begin{array}{llll} v_{11} = 0.1; & v_{12} = 0.3; & v_{21} = 0.23; & v_{22} = 0.8; \\ v_{31} = 0.56; & v_{32} = 0.43; & v_{41} = 0.32; & v_{42} = 0.76 \end{array}$$

Since the multiplayer perceptrons is in the category of supervised learning, we need to have some a priori information to train the system. Let us select some of the pattern points from the data set as the input/output pairs. Note that the selection of these pattern points as a training set is very crucial. It would make a lot of difference in the results. The training set must be both large enough and diverse enough to adequately represent the problem domain. Within each class sufficient samples must be present to reflect real-world variations within the class. To take care of this nonlinear problem, more pattern samples near the nonlinear boundary of these two classes will be selected.

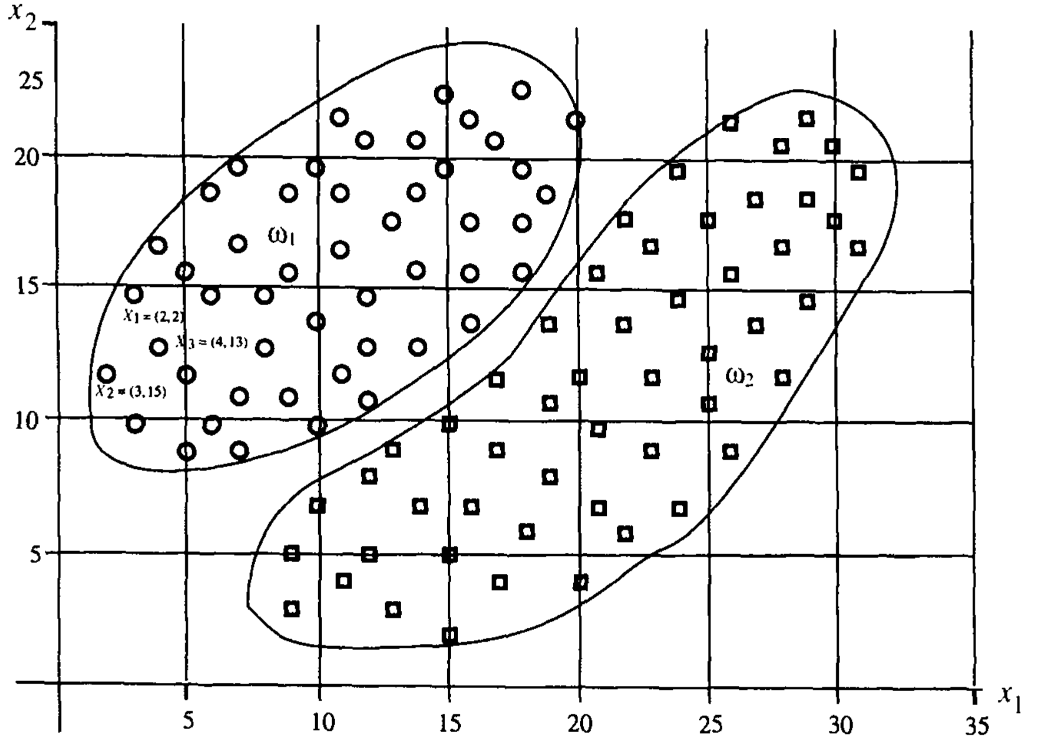


FIGURE 8.8 One hidden layer structure perceptron.

The samples selected for the training set are

$$p_1^1 = (7,9);\ p_1^2 = (10,10);\ p_1^3 = (14,13);\ p_1^4 = (17,15);$$
$$p_1^5 = (20,22);\ p_1^6 = (5,16);\ p_1^7 = (6,19);\ p_1^8(8,21);$$
$$p_1^9 = (12,21);\ p_1^{10} = (20,22) \qquad \text{from class 1}$$

and

$$p_2^1 = (13,9);\ p_2^2 = (17,12);\ p_2^3 = (9,5);\ p_2^4 = (15,10);\ p_2^5 = (19,14);$$
$$p_2^6 = (21,16); p_2^7 = (15,5);\ p_2^8 = (20,4); p_2^9 = (22,6);\ p_2^{10} = (24,7)$$
$$\text{from class 2}$$

Run the program by presenting the samples in the training set (usually called prototypes) as the input vectors in random order. Eventually the synapses (or weights) stabilize to the values shown below:

$$w_{11} = 0.0269;\ w_{12} = -1.4230;\ w_{13} = -1.7330;\ w_{14} = 0.0069$$

and

$$v_{11} = -0.2205;\ v_{12} = 0.6205;\ v_{21} = 0.5794;\ v_{22} = -0.4506;$$

$$v_{31} = 0.0731;\ v_{32} = 0.0569;\ v_{41} = -1.4761;\ v_{42} = -1.9160.$$

Let us assume that the system has been trained. With these weight parameters fixed for the system, present one by one all the patterns from the two data sets listed above, and see how many of these patterns are correctly classified by the system.

Results showed that two pattern points in ω_2, namely, (9, 5) and (10, 7), were misclassified as ω_1; and four pattern points in ω_1, namely (17, 21), (18, 20), (18, 23) and (20, 22), were misclassified as ω_2. Note that all these misclassified pattern points in ω_1 and ω_2 are in the two extremity portions of circled regions. It seems that we need to select one or two more prototypes over those regions to improve the classification rate.

When another hidden layer was added to the network, the processing results turned out nicely even with the same prototypes. All pattern points in ω_1 and ω_2 were correctly classified. The two-dimensional plot shown in Figure 8.8 shows the data distribution and the nonlinear boundary of these two classes. It is expected that the processing described above could work very well for a problem with two very large data sets in ω_1 and ω_2, so long as these two sets of data fall, respectively, into their own category regions as depicted by the two curved contours.

PROBLEMS

8.1 Prove that $f'(\text{net}_j) = O_j^k(1 - O_j^k)$ in Eq. (8.38).

8.2 Work out couple of more iterations for the problem in Section 8.2.1 and see what changes happen in the weights.

8.3 Explain why we can use $\frac{1}{2}|\mathbf{e}^k|^2$ as the mean-square criterion function in Eq. (8.16).

8.4 Consider a multilayer network with N inputs, K hidden units, and M output units. Write down an expression for the total number of weights and biases in the network.

8.5 Given that the following two-dimensional pattern points belong to class 1:

Data set 1 (pattern points belonging to class ω_1):

(7, 14), (8, 10), (8, 11), (8, 13), (8, 15), (8, 16), (9, 9), 9, 14),
(9, 17), (10, 9), (10, 12), (10, 16), (10, 18), (11, 14), (12, 9),
(12, 11), (12, 13), (12, 17), (12, 20), (13, 12), (13, 16),
(13, 18), (14, 9), (14, 11), (14, 14), (15, 10), (15, 12), (15, 16),
(15, 19), (15, 21), (16, 9), (16, 13), (16, 15), (17, 10), (17, 18),
(17, 20), (18, 15), (19, 10), (19, 13), (19, 17), (19, 21),
(20, 12), (20, 17), (20, 19), (21, 11), (21, 16), (21, 21),
(22, 12), (22, 20), (23, 12), (23, 14), (23, 17), (23, 22),
(24, 13), (24, 15), (25, 13), (25, 18), (25, 22), (26, 15),
(26, 17), (27, 16), (27, 19), (27, 21), (28, 18), (28, 20).

and that the following two-dimensional pattern points belonging to class 2:

Data set 2 (pattern points belonging to class ω_2):

(8, 5), (9, 3), (9, 7), (10, 2), (10, 4), (10, 6), (11, 8), (12, 2),
(12, 5), (13, 3), (13, 8), (14, 6), (15, 2), (15, 4), (15, 8), (16, 5),
(17, 3), (17, 6), (17, 8), (18, 2), (19, 5), (19, 7), (19, 9), (20, 2),
(20, 6), (21, 3), (21, 7), (21, 9), (22, 5), (22, 10), (23, 3),
(23, 8), (24, 6), (24, 11), (25, 5), (25, 8), (25, 10), (25, 12),
(26, 4), (26, 7), (26, 13), (27, 11), (27, 14), (28, 6), (28, 9),
(28, 15), (28, 16), (29, 7), (29, 11), (29, 13), (29, 17), (30, 8),
(30, 15), (30, 17), (31, 10), (31, 16), (32, 13), (32, 15).

(a) Write a program in C or $C++$ for a multilayer perceptron with one hidden layer to separate these two sets of data. Select 15 pattern points from each of these two data sets as prototypes to train the system and then test all the remaining pattern points.

(b) Same as (a) but with two hidden layers.

(c) Discuss your results obtained with respect to the choice of the training set.

(d) Develop a suitable training set to correctly separate all these two sets of data.

9

Radial Basis Function Networks

9.1 RADIAL BASIS FUNCTION NETWORKS

Use of multilayer perceptron to solve nonlinear classification problem is very effective. Nevertheless, a multilayer perceptron often has many layers of weights and a complex pattern of connectivity. The interference and cross-coupling among the hidden units results in a highly nonlinear network training with nearly flat regions in the error function which arises from near cancellations in the effects of different weights. This can lead to very slow convergence of the training procedure. It therefore arouses people's interest to explore other better way to overcome these deficiencies without losing its major features in approximating arbitrary nonlinear functional mappings between multidimensional spaces. At the same time there appears a new viewpoint in the interpretation of the function of pattern classification to view *pattern classification as a data interpolation problem in a hyperspace, where learning amounts to finding a hypersurface that will best fit the training data*. Cover (1965) stated in his theorem on the separability of patterns that a complex pattern classification problem cast in high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space. From there it can then be inferred that once these patterns have been transformed into their counterparts that can be linearly separable, the classification problem would be relatively easy to solve. This motivates the method of radial basis functions (RBF), which could be substantially faster than the methods used to train multilayer perceptron networks.

The method of radial basis functions originates from the technique in performing the exact interpolation of a set of data points in a multidimensional space. In this radial basis function method, we are not computing a nonlinear function of the scalar product of the input vector and a weight vector in the hidden unit. Instead, we determine the activation of a hidden unit by the distance between the input vector and a prototype vector.

As mentioned, the RBF method is developed from the exact interpolation approach, but with modifications, to provide a smooth interpolating function. The construction of a radial basis function network involves three different layers, namely, an input layer, a hidden layer, and an output layer. The input layer is primarily made up of source nodes (or sensory units) to hold the input data for processing. For an RBF in its basic form, there is only one hidden layer. This hidden layer is of high enough dimensions. It provides a nonlinear transformation from the input space. The output layer, which gives the network response to an activation pattern applied to the input layer, provides a linear transformation from the hidden unit space to the output space. Figure 9.1 shows the transformations imposed on the input vector by each layer of the RBF network. It can be noted that a nonlinear mapping is used to transform a nonlinearly separable classification problem into a linearly separable one.

As shown in Figure 9.1, the training procedure can then be split into two stages. The first stage is a nonlinear transformation. In this stage a nonlinear mapping function $\phi(\mathbf{x})$ of high enough dimensions is to be found such that we will have linear separability in the ϕ space. This is similar to what we discussed on the ϕ machine in Chapter 3, where a non-linear quadratic discriminant function is transformed into a linear function of $f_i(\mathbf{x})$, $i = 1, 2, \ldots, M$, representing, respectively, $x_1^2, x_2^2, \ldots, x_n^2, x_1x_2, x_3, \ldots, x_{n-1}x_n$, and $x_1, x_2 \ldots, x_n$; and $f_i(\mathbf{x})$ are linearly independent, real- and single-value functions, which are independent of w_i (weight). Note that in this case the discriminant function $d(\mathbf{x})$ is linear with respective to w_i, but $f_i(\mathbf{x})$ are not necessary assumed to be linear [see Eqs. (3.35) and (3.36)].

Let us come back to the radial basis function problem. The basis functions used in this scheme are all localized functions. The parameters governing the basis functions (corresponding to hidden units) can be determined by using relatively fast, unsupervised methods. Some of these methods were discussed in

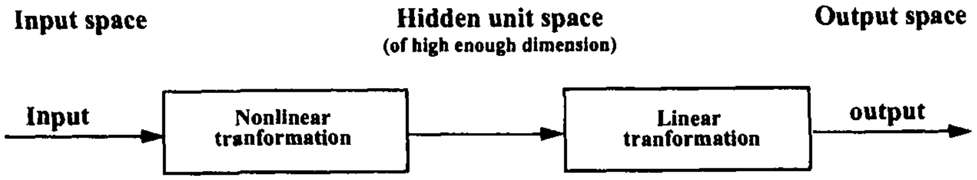


FIGURE 9.1 Radial basis function network.

Chapter 6. In these methods, only the input data are used. The second stage of the RBF network (i.e., from the hidden unit space to the output space) is a linear transformation that determines the weights for the final layer. It is a linear problem and is therefore fast also.

As mentioned in the previous paragraph, a radial basis function network is a modification to the exact interpolation approach, in which the number of basis functions is determined by the complexity of the mapping to be represented rather than by the size of the data set. That is, the number of the basis functions is much less than that of the pattern data points ($M \ll N$, where M is the number of basis functions and N represents the number of pattern data points). With such an argument, the centers of the basis functions will not be constrained to the input data vectors. Suitable centers will be determined during the training process.

Let $\mathbf{x}$ be a d-dimensional input vector, $\mathbf{t}$ a target vector which is one-dimensional, N the number of input vectors $\mathbf{x}^n$, $n = 1, 2, \ldots N$, and M the number of basis functions in the hidden unit. A set of basis functions can be chosen with the following general forms:

$$\phi(|\mathbf{x} - \mathbf{c}_i|) \tag{9.1}$$

The argument of the function is the euclidean distance of the input vector $\mathbf{x}$ from a center $\mathbf{c}_i$. This justifies the name radial basis function. Several forms for the basis functions ϕ have been considered, namely,

$$\phi(\mathbf{x}) = \exp\left(-\frac{1}{2\sigma_i^2}|\mathbf{x} - \mathbf{c}_i|^2\right) \tag{9.2}$$

$$\phi(\mathbf{x}) = \frac{\sigma^2}{\sigma^2 + |\mathbf{x} - \mathbf{c}_i|^2} \tag{9.3}$$

$$\phi(\mathbf{x}) = (x^2 + \sigma^2)^{-\alpha} \qquad \alpha > 0 \tag{9.4}$$

$$\phi(\mathbf{x}) = x^2 \ln \mathbf{x} \qquad \text{(thin plate spline function)} \tag{9.5}$$

The gaussian form is the most commonly used one. It is a localized basis function. As $|\mathbf{x}| \to 0$, ϕ will approach zero as limit. The mapping of the radial basis function network is

$$y_k(\mathbf{x}) = \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x}) + w_{k0} \tag{9.6}$$

or

$$y_k(\mathbf{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathbf{x}) \qquad \text{with } \phi_0 = 1 \tag{9.7}$$

where $y_k(\mathbf{x})$ is the kth component of $\mathbf{y}(\mathbf{x})$, the function to map the input space to the one-dimension target space. $y_k(\mathbf{x})$ are obtained by linear superposition of the

M basis functions. $\phi_j(\mathbf{x}), j = 1, 2, \ldots, M$ are linearly independent, real and local basis functions which are independent of the w_k (weights). Note that the $y(\mathbf{x})$ is linear with respect to w_i, but $\phi_j(\mathbf{x})$ are not necessarily assumed to be linear. For the case of gaussian basis functions we have

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2\sigma_j^2}|\mathbf{x} - \mathbf{c}_j|^2\right) \tag{9.8}$$

where $\mathbf{c}_j$ is the center chosen for the jth basis function, $\phi_j(\mathbf{x})$. Note that centers chosen need not be the same for the radial basis functions. In contrast, we set the centers to different appropriate values determined through training. With values so chosen for the individual basis functions, the processing speed will be increased. This expression can be generalized as

$$\phi_j(\mathbf{x}) = \exp\left[-\tfrac{1}{2}(\mathbf{x} - \mathbf{c}_j)^T \sum_j{}^{-1}(\mathbf{x} - \mathbf{c}_j)\right] \tag{9.9}$$

where $\mathbf{x}$ represents, respectively, the d-dimensional input vector with elements x_i, and $\mathbf{c}_j$ is the vector determining the center of basis function ϕ_j and has elements c_{ji}. Figure 9.2 shows the architecture of the radial basis function network. Hart shows in his simulation (Hart 90) that a RBF should be of much high order. This

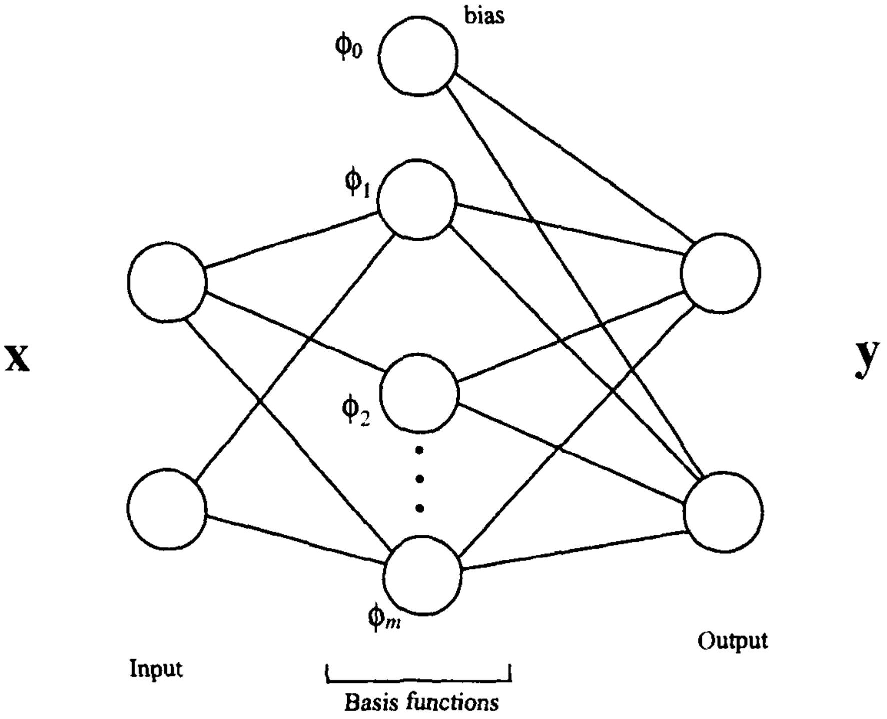


FIGURE 9.2 Architecture of the radial function network.

is due to the locality of the RBF activations which makes it necessary to use a large number of centers to fill in the space in which $g(\mathbf{x})$ is defined.

Let us take a simple example to show how does the method of RBF perform the mapping of a nonlinear problem to a linearly separable class problem. Figure 9.3 shows the positions of the pattern points $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_5$, $\mathbf{x}_6$, $\mathbf{x}_7$, $\mathbf{x}_8$ and $\mathbf{x}_9$ in space and the class each of them belongs to. $\mathbf{x}_1$, $\mathbf{x}_2$ belong to class A, while the rest of them, $\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_5$, $\mathbf{x}_6$, $\mathbf{x}_6$, $\mathbf{x}_7$, $\mathbf{x}_8$ and $\mathbf{x}_9$, belong to class B. It is apparent from this figure that no single straight line exists that can separate the two classes. That is, these two classes are not linearly separable. Properly choose the centers, $\mathbf{c}_1 = [1.1]^T$ and $\mathbf{c}_2 = [0, 0]^T$, and the basis function $\phi_i(\mathbf{x}) = \exp(-|\mathbf{x} - \mathbf{c}_i|^2)$, $i = 1, 2$, assuming that σ^2 is the same for all the basis functions and are left out in the following expressions. The corresponding ϕ resulting from the mapping is

$$\phi(\mathbf{x}) = \begin{vmatrix} \exp(-|\mathbf{x} - \mathbf{c}_1|^2) \\ \exp(-|\mathbf{x} - \mathbf{c}_2|^2) \end{vmatrix} \tag{9.10}$$

For the pattern point $\mathbf{x}_1(0, 0)$,

$$\begin{aligned}\phi_1 &= \exp[-|\mathbf{x} - \mathbf{c}_1|^2] = \exp[-((x_{11} - c_{11})^2 + (x_{12} - c_{12})^2)] \\ &= \exp[-2] = 0.135 \\ \phi_2 &= \exp[-|\mathbf{x} - \mathbf{c}_2|^2] = \exp[-((x_{11} - c_{21})^2 + (x_{12} - c_{22})^2)] \\ &= \exp[-0] = 1\end{aligned}$$

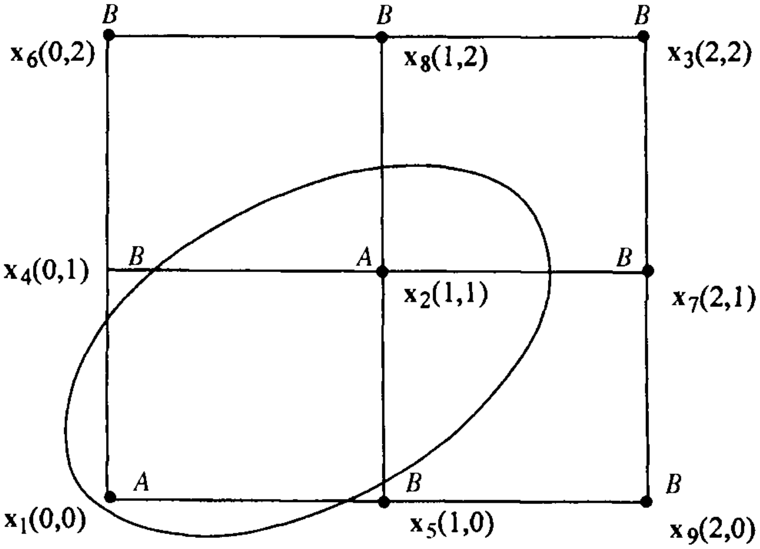


FIGURE 9.3 A nonlinear two-class problem and the decision surface drawn in the original pattern space. $\mathbf{x}_1 = (0, 0)$ and $\mathbf{x}_2 = (1, 1)$ in class A. $\mathbf{x}_3 = (2, 2)$, $\mathbf{x}_4 = (0, 1)$, $\mathbf{x}_5(1, 0)$, $\mathbf{x}_6 = (0, 2)$, $\mathbf{x}_7 = (2, 1)$; $\mathbf{x}_8 = (1, 2)$ and $\mathbf{x}_9 = (2, 0)$ in class B.

or

$$\phi^1 = [0.135, 1]^T \tag{9.11}$$

where ϕ^1 represents the pattern point $\mathbf{x}_1$ in the transformed space. Similar notations (ϕ^1, $\phi^2, \ldots, \phi^9$) are for patterns $\mathbf{x}_1 = (0, 0)$, $\mathbf{x}_2 = (1, 1)$, $\mathbf{x}_3 = (2, 2)$, $\mathbf{x}_4 = (0, 1)$, $\mathbf{x}_5 = (1, 0)$, $\mathbf{x}_6 = (0, 2)$, $\mathbf{x}_7 = (2, 1)$, $x_8 = (1, 2)$, and $\mathbf{x}_9 = (2, 0)$ in the transformed space. For pattern point $\mathbf{x}_2 = (1, 1)$,

$$\begin{aligned}\phi_1 &= \exp[-|\mathbf{x}_2 - \mathbf{c}_1|^2] = \exp[-((x_{21} - c_{11})^2 + (x_{22} - c_{12})^2)]\\ &= \exp[-0] = 1;\\ \phi_2 &= \exp[-|\mathbf{x}_2 - \mathbf{c}_2|^2] = \exp[-((x_{21} - c_{21})^2 + (x_{22} - c_{22})^2)]\\ &= \exp[-2] = 0.135\end{aligned}$$

or

$$\phi^2 = [1, 0.135]^T \tag{9.12}$$

Similarly, we can compute all the ϕ_1's and ϕ_2's, respectively, for $\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_5$, $\mathbf{x}_6$, $\mathbf{x}_7$, $\mathbf{x}_8$, and $\mathbf{x}_9$ as

$$\phi^3 = [0.135, 0.067]^T \tag{9.13}$$

$$\phi^4 = [0.368, 0.368]^T \tag{9.14}$$

$$\phi^5 = [0.368, 0.368]^T \tag{9.15}$$

$$\phi^6 = [0.135, 0.183]^T \tag{9.16}$$

$$\phi^7 = [0.135, 0.183]^T \tag{9.17}$$

$$\phi^8 = [0.368, 0.0067]^T \tag{9.18}$$

$$\phi_9 = [0.368, 0.0067]^T \tag{9.19}$$

Figure 9.4 shows the resulting pattern points' positions in the transformed space. Obviously, the two classes are now linearly separable. Any line drawn with ϕ_1, ϕ_2 located on one side and rest of the ϕ's on the other side is a possible solution. The simplest line

$$\phi_1 + \phi_2 - 1 = \mathbf{0} \tag{9.20}$$

which is drawn with dashes as shown in Figure 9.4, is also a possible solution. The corresponding decision surface in the input vector space is then represented by

$$\exp[-|\mathbf{x} - \mathbf{c}_1|^2] + \exp[-|\mathbf{x} - \mathbf{c}_2|^2] - 1 = 0 \tag{9.21}$$

if the gaussian form of the basis function is adopted. Note that in this example the centers were carefully selected (i.e., the RBF network was assumed to have

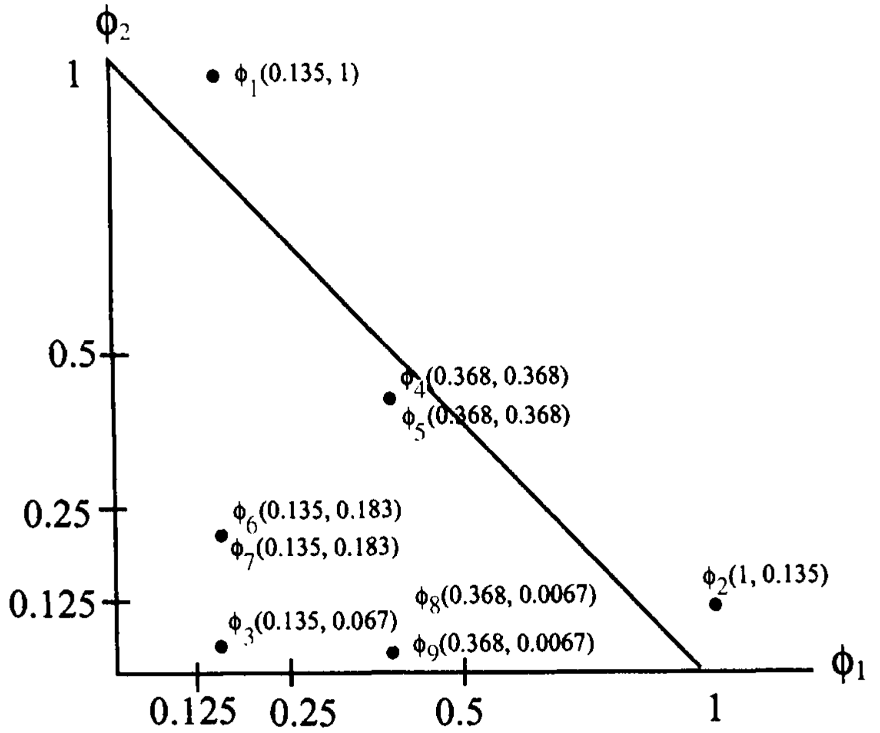


FIGURE 9.4 Decision surface built by an RBF generalized linear in the transformed space of the problem shown in Figure 9.3.

already been well trained) to separate $\mathbf{x}_1$ and $\mathbf{x}_2$ as a class, and the rest of the data points belonging to another class. But when patterns $\mathbf{x}_1 = (0, 0)$ and $\mathbf{x}_9 = (2, 0)$ belong to one class, while the rest of the pattern points belong to another class in another problem, the centers should be chosen as $\mathbf{c}_1 = \mathbf{x}_1 = (0, 0)$ and $\mathbf{c}_2 = \mathbf{c}_9 = (2, 0)$. In so doing, $\mathbf{x}_1$ and $\mathbf{x}_9$ will be linearly separated from the rest in the ϕ space. The reader can check it as a homework problem.

9.2 RBF NETWORK TRAINING

A two-stage training procedure for the radial basis function networks is adopted. In the first stage, the unsupervised learning (or clustering) method on the input data set $[\mathbf{x}]$ can be used to determine the parameters of the basis functions, for example, $\mathbf{c}_j$ and σ_j for the spherical gaussian basis functions considered above. We then keep the basis functions fixed and find the weights during the second phase of training. Suppose we have selected M centers for the RBF functions. The problem then becomes a typical linear one in the M-dimensional space of ϕ

$$\phi = \begin{vmatrix} \exp[-|\mathbf{x} - \mathbf{c}_1|^2] \\ \vdots \\ \exp[-|\mathbf{x} - \mathbf{c}_M|^2] \end{vmatrix} \tag{9.22}$$

The output of the network can be computed as:

$$y_k(\mathbf{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathbf{x}) + w_{k0} \tag{9.23}$$

in matrix form

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}^T\phi + w_0 \tag{9.24}$$

where $\mathbf{W} = (w_{kj})$ and $\phi = (\phi_j)$. All methods described in Chapters 2 to 4 can now be recalled to estimate w_0 and $\mathbf{W}$. Note that if the centers are not preselected, they have to be estimated during the training phase.

In addition to adopting the clustering method for the first stage in the training procedure as mentioned previously, the error minimization method can also be used. Choose an appropriate cost function $\mathcal{J}$:

$$\mathcal{J} = \sum_{j=1}^{N} f(e(j)) \qquad \text{and} \qquad e(j) = d(j) - y(j) \tag{9.25}$$

where $y(j)$ and $d(j)$ are respectively the computed and desired output of the network; $f(.)$ is a differentiable function (e.g., the square of its argument) of the error; and N is the number of input/desired output training pairs $[x(j), d(j), j = 1, 2, \ldots, N]$. We can then follow the procedure of the gradient descent methods as described in Chapters 2 to 4 for the training of the W_t, $\mathbf{c}_t$, and σ_t:

$$W_i(k+1) = w_i(k) - \beta_1 \frac{\partial\mathcal{J}}{\partial w_i}|_t \qquad i = 0, 1, \ldots, M \tag{9.26}$$

$$c_i(k+1) = c_i(k) - \beta_2 \frac{\partial\mathcal{J}}{\partial c_t}|_t \qquad i = 0, 1, \ldots, M \tag{9.27}$$

$$\sigma_i(k+1) = \sigma_i(k) - \beta_3 \frac{\partial\mathcal{J}}{\partial \sigma_t}|_t \qquad i = 0, 1, \ldots, M \tag{9.28}$$

where k denotes the current iteration step. An alternative scheme such as the pseudoinverse of ϕ can also be used.

9.3 FORMULATION OF THE RADIAL BASIS FUNCTIONS FOR PATTERN CLASSIFICATION BY MEANS OF STATISTICAL DECISION THEORY

As discussed in Chapter 5 in the formulation of a pattern classification by means of statistical decision theory, our goal is to model the a posteriori probabilities $p(\omega_i|\mathbf{x})$ for each of the classes. By Bayes' rule, we can write:

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{p(\mathbf{x})} \tag{9.29}$$

where $p(\mathbf{x}) = \sum_i p(\mathbf{x}|\omega_i)p(\omega_i)$, $i = 1, 2, \ldots, M$ is the probability that $\mathbf{x}$ occurs without regard to the category in which it belongs. $p(\omega_i)$ is the a priori probability of class ω_i, and $p(\mathbf{x}|\omega_i)$ is the likelihood function of class ω_i with respect to $\mathbf{x}$. It is the probability density function for $\mathbf{x}$ given that the state of nature is ω_i (i.e., it is a pattern belonging to class ω_i). $p(\omega_i|\mathbf{x})$ is the probability that $\mathbf{x}$ comes from ω_i. This is a posteriori probability. By substituting $\sum_i p(\mathbf{x}|\omega_i)p(\omega_i)$ for $p(\mathbf{x})$, Eq. (9.29) can be rewritten as

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)}{\sum_i p(\mathbf{x}|\omega_i)p(\omega_i)} \cdot p(w_i) \tag{9.30}$$

Representing $p(\mathbf{x}|\omega_i)/ \sum_i p(\mathbf{x}|\omega_i)p(\omega_i)$ as $\phi_k(\mathbf{x})$ gives

$$p(\omega_i|\mathbf{x}) = p(\omega_i) \cdot \phi_k(\mathbf{x}) \tag{9.31}$$

Equation (9.31) is in a single form of basis function network with $p(\omega_i)$ as the weight from each hidden unit going to the corresponding output unit. The outputs of this network represents approximations to the a posteriori probabilities.

Suppose a common pool of M basis functions can be used to represent all of the class-conditioned densities $p(\mathbf{x}|\omega_i)$. $p(\mathbf{x}|\omega_i)$ can then be written as:

$$p(\mathbf{x}|\omega_i) = \sum_{j=1}^{M} p(\mathbf{x}|j)p(j|\omega_i) \tag{9.32}$$

and $p(\mathbf{x})$, which is $\sum_i p(\mathbf{x}|\omega_i)p(\omega_i)$, can be written in terms of $p(\mathbf{x}|j)$ and $p(j|\omega_i)$ as follows:

$$p(\mathbf{x}) = \sum_i \sum_{j=1}^{M} p(\mathbf{x}|j)p(j|\omega_i)p(\omega_i) \tag{9.33}$$

$$= \sum_{j=1}^{M} p(\mathbf{x}|j) \sum_i p(j|\omega_i)p(\omega_i) \tag{9.34}$$

$$= \sum_{j=1}^{M} p(\mathbf{x}|j)p(j) \tag{9.35}$$

Substituting expressions (9.32) and (9.35) respectively for $p(\mathbf{x}|\omega_i)$ and $p(\mathbf{x})$ into Eq. (9.29), we have

$$p(\omega_i|\mathbf{x}) = \frac{\sum_{j=1}^{M} p(\mathbf{x}|j)p(j|\omega_i)p(\omega_i)}{\sum_{j=1}^{M} p(\mathbf{x}|j)p(j)} \tag{9.36}$$

Multiplying both the numerator and denominator of the above equation by $p(j)$ and rearranging, Eq. (9.36) becomes

$$p(\omega_i|\mathbf{x}) = \sum_{j=1}^{M}\left[\frac{p(j|\omega_i)p(\omega_i)}{p(j)}\right]\cdot\left[\frac{p(\mathbf{x}|j)p(j)}{\sum_{j=1}^{M}p(\mathbf{x}|j)p(j)}\right] \tag{9.37}$$

or

$$p(\omega_i|\mathbf{x}) = \sum_{j=1}^{M} w_{kj}\cdot\phi_j(\mathbf{x}) \tag{9.38}$$

In other words, solution of the a posteriori probabilities $p(\omega_i|\mathbf{x})$ can be represented in a radial basis function network form, and the output of the network will give the a posteriori probabilities $p(\omega_i|\mathbf{x})$ for each of the classes. The normalized basis function and the weights are, respectively,

$$\phi_j(\mathbf{x}) = \frac{p(\mathbf{x}|j)p(j)}{\sum_{j=1}^{M}p(\mathbf{x}|j)p(j)} \tag{9.39}$$

$$= p(j|\mathbf{x}) \tag{9.40}$$

and

$$w_{kj} = \frac{p(j|\omega_i)p(\omega_i)}{p(j)} \tag{9.41}$$

$$= p(\omega_i|j) \tag{9.42}$$

For more elaborated discussion on this subject see Lowe (1995) and Bishop (1995).

9.4 COMPARISON OF RBF NETWORKS WITH MULTILAYER PERCEPTRONS

Both the radial basis function (RBF) networks and multilayer perceptrons (MLP) provide techniques for approximating nonlinear functional mappings between multidimensional spaces. However, the structures of these two networks are quite different from each other. Some of the important differences between the RBF and MLP networks are outlined below.

1. An RBF network (in its most basic form) has one hidden layer, whereas a MLP may have one or more hidden layers and a complex pattern of connectivity. The interference and cross-coupling between the hidden units in a multilayer perceptron can lead to slow convergence of the training procedure.

2. In multilayer perceptron networks, the activation responses of the nodes are of a global nature, and the output is the same for all points on a hyperplane, whereas the activation responses of the nodes are of a local nature in the RBF networks in the sense that output of each RBF node $f(.)$ is the same for all points having the same euclidean distance from the respective center $\mathbf{c}_i$ and decreases exponentially with the distance.
3. In a multilayer perceptron all parameters are usually determined at the same time as part of a single global training strategy involving supervised training, whereas an RBF network using exponentially decaying localized nonlinearities to construct local approximations to nonlinear mapping for the hidden layer achieves fast learning and makes the system less sensitive to the order of presentation of the training data.

PROBLEMS

9.1 Check the results shown in Figure 9.4.

9.2 Refer to the example given in the text. Choose the centers as (0, 0) and (2, 0). Which pattern points will be separated from the others? What conclusion you can draw from the example and this exercise?

10

Hamming Net and Kohonen Self-Organizing Feature Map

Among the real-world pattern classification problems there exists such a case that we have more possible correct responses of the network than we are able to incorporate. More specifically, when a network was trained to classify the input signal into one of the several output categories, the response from the network sometimes would assign the signal to two or more than two of the given classes. To overcome such a case, additional structure should be added to help decide as to which unit to respond so that only one neuron in the group will be enforced to have a positive output while the rest of them turn to zero. This is commonly called neural networks on competitive basis.

The Hamming net, Maxnet (Lippmann, 1987), and the Kohonen self-organizing feature maps (Kohonen, 1987) are the networks to achieve this goal. Hamming nets and Maxnet are for bipolar inputs, whereas the Kohonen self-organizing feature map is for continuous inputs. Let us discuss the Hamming net first. After the discussion of the Hamming net, we will include Maxnet as a second layer for the Hamming net to achieve the goal of winner-take-all.

10.1 HAMMING NET

The Hamming net is a maximum likelihood classifier or a minimum Hamming distance classifier, which selects one of the stored classes that are at a minimum

Hamming distance to the n-tuple pattern vector presented at the input. The Hamming distance between two vectors is the number of components in which these two vectors differ. Suppose that two binary bipolar n-tuple vectors have **a** components agree and **d** components differ from each other; then **d** gives the Hamming distance (HD), and $\mathbf{a} = n - \text{HD}$ gives the number of components in which the vectors agree, where n is the total number of components in the vector. When one of the vectors is the input vector **x** and the other one is the encoded class prototype vector (say $\mathbf{c}_i$), the dot product of these two bipolar binary n-tuple vectors is $\mathbf{a} - \mathbf{d}$, or

$$\mathbf{x}^T \cdot \mathbf{c}_i = (n - \text{HD}) - \text{HD} \tag{10.1}$$

or

$$\tfrac{1}{2}\mathbf{x}^T \cdot \mathbf{c}_i = \frac{n}{2} - \text{HD}(\mathbf{x}^T, \mathbf{c}_i) \tag{10.2}$$

where $\text{HD}(\mathbf{x}^T, \mathbf{c}_i)$ is the Hamming distance between the input vector **x** and the encoded class prototype vector $\mathbf{c}_i$. Recalling that $\text{HD} = n - \mathbf{a}$, Eq. (10.2) can be put in the following form:

$$\mathbf{a} = \tfrac{1}{2}\mathbf{x}^T \cdot \mathbf{c}_i + \frac{n}{2} \tag{10.3}$$

Let the network be a pattern classifier for M classes We then have M outputs, one for each class. When the input **x** is known to belong to class ω_i with prototype vector $\mathbf{c}_i$, then only the ith output of the network is 1. The rest of the network outputs are zero.

Refer to the architecture of Hamming net as shown in Figure 10.1; outputs of the neurons in the Hamming net are

$$\text{net}_j = \mathbf{x}^T \cdot \mathbf{w}_j + b_j \qquad j = 1, 2, \ldots, M \tag{10.4}$$

where $\mathbf{w}_j = [w_{j1} \quad w_{j2} \quad \cdots \quad w_{jn}]^T$. Combining Eqs. (10.3) with (10.4), we can then set $b_j = n/2$ and $\mathbf{w}_j = \frac{1}{2}\mathbf{c}_j, j = 1, 2, \ldots, M$, or

$$\mathbf{W} = \frac{1}{2}\begin{vmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & & & \\ c_{M1} & c_{M2} & \cdots & c_{Mn} \end{vmatrix} \tag{10.5}$$

and $w_{ji} = \frac{1}{2}c_{ji}$, $i = 1, 2, \ldots, n; j = 1, 2, \ldots, M$.

The function of the second layer of the network as shown in Figure 10.1 is used to enforce the initial dominant response of a node that has the largest network excitation. When this Maxnet is initialised with the input vector $\mathbf{y}(0)$, the network starts processing it by adding positive self-feedback and negative cross-feedback. As a result of the Maxnet recurrent processing, only the node which has

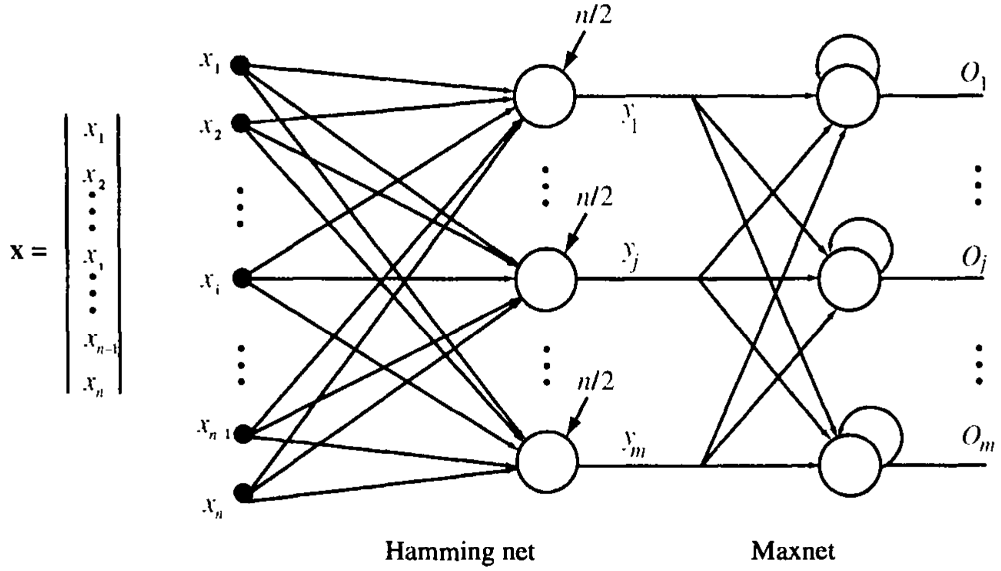


FIGURE 10.1 Hamming net with Maxnet as the second layer.

the largest initializing entry will be unsuppressed, and has the nonzero output response, while all the remaining nodes responses decay to zero. Denoting the output of the Maxnet as **O**, we have

$$\mathbf{O} = \mathbf{w}_M^T \mathbf{y}$$

where $\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_M]^T$ is the input to the Maxnet, $\mathbf{O} = [O_1 \quad O_2 \quad \cdots \quad O_M]^T$ is the output of the Maxnet in Figure 10.1, and $\mathbf{w}_M$ is the weight matrix for the maxnet. A symmetric weight matrix for $\mathbf{w}_M$ with diagonal entries as 1's and all the off-diagonal entries as $-\varepsilon$ was proposed by Pao (1989) to simulate the lateral interaction:

$$\mathbf{w}_M = \begin{vmatrix} 1 & -\varepsilon & \cdots & -\varepsilon \\ -\varepsilon & & & -\varepsilon \\ \vdots & & \ddots & \vdots \\ -\varepsilon & -\varepsilon & \cdots & 1 \end{vmatrix} \tag{10.6}$$

In this matrix ε is chosen in the range of $0 < \varepsilon < 1/M$. Let us choose $\varepsilon = 0.2$ for the case when $M = 3$:

$$\mathbf{w}_M = \begin{vmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{vmatrix} \tag{10.7}$$

An algorithm can be designed for the minimum Hamming distance classifier.

Step 1. Initialize the weight matrix $\mathbf{w}_j$ and the biases:

$$w_{ji} = \frac{c_{ji}}{2}$$
$$b_j = \frac{n}{2} \qquad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, M \tag{10.8}$$

Step 2. For each input vector $\mathbf{x}$, do steps 3 to 5.

Step 3. Compute the $\text{net}_j, j = 1, 2, \ldots, M$:

$$\text{net}_j = b_j + \sum_i x_i w_{ji} \qquad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, M. \tag{10.9}$$

Step 4. Initialize the activation y_j for the Maxnet, the second layer of the network:

$$y_j = \text{net}_j$$

which is the output of neurons in Hamming net, and also inputs to the Maxnet. It represents the Hamming similarity.

Step 5. Maxnet compares the outputs of the net_j, $j = 1, 2, \ldots, M$. It enforces the largest one as the best match prototype, while suppressing the rest of them to zero.

Step 6. Recurrent processing of the Maxnet:

$$\mathbf{O} = \mathbf{w}_M \mathbf{y} = \begin{vmatrix} 1 & -0.2 & \cdots & -0.2 \\ -0.2 & & & -0.2 \\ \vdots & & \ddots & \vdots \\ -0.2 & -0.2 & \cdots & 1 \end{vmatrix} \begin{vmatrix} y_1^k \\ y_2^k \\ \vdots \\ y_M^k \end{vmatrix}$$
$$= \mathbf{w}_M \mathbf{net}^k$$

where

$$\mathbf{net}^k = \begin{vmatrix} \text{net}_1^k \\ \vdots \\ \text{net}_j^k \\ \vdots \\ \text{net}_M^k \end{vmatrix}$$

and

$$\mathbf{O} = \begin{vmatrix} f(\text{net}_1^k) \\ \vdots \\ f(\text{net}_j^k) \\ \vdots \\ f(\text{net}_M)^k \end{vmatrix} \quad f(\text{net}_j) = \begin{cases} 0 & \text{when net}_j < 0 \\ \text{net}_j & \text{when net}_j \geq 0 \end{cases} \tag{10.12}$$

Example. Given that stored in a Hamming net are three prototype vectors, respectively, representing the encoding of three characters C, H, and L with a simple 3 × 3 matrix (see Figure 10.2):

$$\begin{array}{lrrrrrrrrrl} \mathbf{c}_1 = (1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1) & & \text{Character C} \\ \mathbf{c}_2 = (1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 & 1) & & \text{Character H} \\ \mathbf{c}_3 = (1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & 1) & & \text{Character L} \end{array}$$

we are to find the prototype vectors that are closest, respectively, to each of the following input bipolar patterns:

$$\begin{array}{lrrrrrrrrr} \mathbf{x}_1\!: & (1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1) \\ \mathbf{x}_2\!: & (1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1) \\ \mathbf{x}_3\!: & (1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1) \end{array}$$

Step 1. Use the prototype vectors $\mathbf{c}_i$, $i = 1, 2,$ and 3, and set up the weight matrix $\mathbf{W}$ of the Hamming net as:

$$\mathbf{w} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 \end{vmatrix}$$

and set

$$b_1 = b_2 = b_3 = \frac{n}{2} = \frac{9}{2}$$

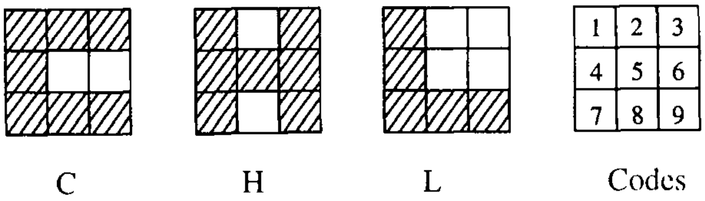


FIGURE 10.2 Characters C, H, and L and their encoding with simple 3 × 3 matrix.

Step 2. For the input vector $\mathbf{x}_1 = (1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad 1 \quad 1 \quad 1)^T$, do steps 3 to 9.

Step 3. Compute $\text{net}_j, j = 1, 2$, and 3:

$$\begin{aligned}
\text{net}_1 &= b_1 + \sum_i x_i w_{1i} \qquad i = 1, 2, \ldots, 9 \\
&= \tfrac{9}{2} + 1(0.5) + 1(0.5) + 1(0.5) + 1(0.5) + 1(-0.5) \\
&\quad + (-1)(-0.5) + 1(0.5) + 1(0.5) + 1(0.5) \\
&= 8 \\
\text{net}_2 &= b_2 + \sum_i x_i w_{2i} \\
&= \tfrac{9}{2} + 1(0.5) + 1(-0.5) + 1(0.5) + 1(0.5) + 1(0.5) \\
&\quad + (-1)(0.5) + 1(0.5) + 1(-0.5) + 1(0.5) \\
&= 6 \\
\text{net}_3 &= b_3 + \sum_i x_i w_{3i} \\
&= \tfrac{9}{2} + 1(0.5) + 1(-0.5) + 1(-0.5) + 1(0.5) \\
&\quad + 1(-0.5) + (-1)(-0.5) + 1(0.5) + 1(-0.5) \\
&\quad + 1(0.5) \\
&= 6
\end{aligned}$$

Step 4. Initialize the outputs (i.e., the inputs to the Maxnet):

$$\begin{aligned}
y_1(0) &= 8 \\
y_2(0) &= 6 \\
y_3(0) &= 6
\end{aligned}$$

Step 5. Maxnet compares the outputs of $\text{net}_j, j = 1, 2$, and 3, and picks up the largest one. Based on the computation results obtained in steps 2 and 3, i.e., $y_1(0) > y_2(0)$ and $y_3(0)$, it appears that unit y_1 of the Hamming net gives the best match prototype vector $\mathbf{c}_1$ for the input vector:

$$\mathbf{x} = (1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad 1 \quad 1 \quad 1 \quad)^T$$

Maxnet then starts to enforce this largest one as the best match prototype and suppress the other two is zero.

Step 6. Recurrent processing of the Maxnet follows; $k = 0$.

$$\mathbf{net}^0 = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix} \begin{bmatrix} 8 \\ 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 5.6 \\ 3.2 \\ 3.2 \end{bmatrix}$$

$$\mathbf{y}^1 = f(\mathbf{net}^0) = \begin{bmatrix} 5.6 \\ 3.2 \\ 3.2 \end{bmatrix}$$

Step 7. $k = 1$.

$$\mathbf{net}^1 = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix} \begin{bmatrix} 5.6 \\ 3.2 \\ 3.2 \end{bmatrix} = \begin{bmatrix} 4.32 \\ 1.44 \\ 1.44 \end{bmatrix}$$

$$\mathbf{y}^2 = f(\mathbf{net}^1) = \begin{bmatrix} 4.32 \\ 1.44 \\ 1.44 \end{bmatrix}$$

Step 8. $k = 2$.

$$\mathbf{net}^2 = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix} \begin{bmatrix} 4.32 \\ 1.44 \\ 1.44 \end{bmatrix} = \begin{bmatrix} 3.744 \\ 0.288 \\ 0.288 \end{bmatrix}$$

$$\mathbf{y}^3 = f(\mathbf{net}^2) = \begin{bmatrix} 3.744 \\ 0.288 \\ 0.288 \end{bmatrix}$$

Step 9. $k = 3$.

$$\mathbf{net}^3 = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix} \begin{bmatrix} 3.744 \\ 0.288 \\ 0.288 \end{bmatrix} = \begin{bmatrix} 1.592 \\ -0.5184 \\ -0.4684 \end{bmatrix}$$

$$\mathbf{y}^4 = f(\mathbf{net}^3) = \begin{bmatrix} 1.592 \\ 0 \\ 0 \end{bmatrix}$$

The above result shows that the Hamming net with Maxnet has eventually and successfully located the best-matched prototype vector $\mathbf{c}_1$ for the input vector, and identifies the unknown input pattern as character C.

Let us repeat the above steps for the vector

$$\mathbf{x}_3 = (1 \quad -1 \quad -1 \quad 1 \quad -1, \quad 1 \quad 1 \quad 1 \quad 1)^T,$$

do steps 3 to 9.

Step 3. Compute $\mathrm{net}_j, j = 1, 2,$ and 3:

$$\begin{aligned}
\mathrm{net}_1 &= b_1 + \sum_i x_i w_{1i} \qquad i = 1, 2, \ldots, 9 \\
&= \tfrac{9}{2} + 3 \times 0.5 = 6 \\
\mathrm{net}_2 &= b_2 + \sum_i x_i w_{2i} \\
&= \tfrac{9}{2} + 3 \times 0.5 = 6 \\
\mathrm{net}_3 &= b_3 + \sum_i x_i W_{3i} \\
&= \tfrac{9}{2} + 7 \times 0.5 = 8
\end{aligned}$$

Step 4. Initialize the outputs:

$$\begin{aligned}
y_1(0) &= 6 \\
y_2(0) &= 6 \\
y_3(0) &= 8
\end{aligned}$$

Step 5. Maxnet compares the outputs of net_j and enforces the largest one. Since $y_3(0) > y_1(0)$ and $y_2(0)$, the Hamming and Maxnet will find that unit y_3 has the best match prototype vector $\mathbf{c}_3$ for the input vector:

$$\mathbf{x} = (1 \quad -1 \quad -1 \quad 1 \quad -1 \quad -1 \quad 1 \quad 1 \quad 1 \quad)$$

Step 6. Recurrent processing of the Maxnet follows; $k = 0$.

$$\mathbf{net}^0 = \begin{vmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{vmatrix} \begin{vmatrix} 6 \\ 6 \\ 8 \end{vmatrix} = \begin{vmatrix} 3.2 \\ 3.2 \\ 5.6 \end{vmatrix}$$

$$\mathbf{y}^1 = f(\mathbf{net}^0) = \begin{vmatrix} 3.2 \\ 3.2 \\ 5.6 \end{vmatrix}$$

Step 7. $k = 1.$

$$\mathbf{net}^1 = \begin{vmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{vmatrix} \begin{vmatrix} 3.2 \\ 3.2 \\ 5.6 \end{vmatrix} = \begin{vmatrix} 1.44 \\ 1.44 \\ 4.32 \end{vmatrix}$$

$$\mathbf{y}^2 = f(\mathbf{net}^1) = \begin{vmatrix} 1.44 \\ 1.44 \\ 4.32 \end{vmatrix}$$

Step 8. $k = 2.$

$$\mathbf{net}^2 = \begin{vmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{vmatrix} \begin{vmatrix} 1.44 \\ 1.44 \\ 4.32 \end{vmatrix} = \begin{vmatrix} 0.18 \\ 0.18 \\ 4.28 \end{vmatrix}$$

$$\mathbf{y}^3 = f(\mathbf{net}^2) = \begin{vmatrix} 0.18 \\ 0.18 \\ 4.28 \end{vmatrix}$$

Step 9. $k = 3.$

$$\mathbf{net}^3 = \begin{vmatrix} 1 & -0.2 & -0,2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{vmatrix} \begin{vmatrix} 0.18 \\ 0.18 \\ 4.28 \end{vmatrix} = \begin{vmatrix} -0.71 \\ -0.71 \\ 4.21 \end{vmatrix}$$

$$\mathbf{y}^4 = f(\mathbf{net}^3) = \begin{vmatrix} 0 \\ 0 \\ 4.21 \end{vmatrix}$$

The result shows that the best-matched prototype character vector is L, and the unknown input pattern is identified as character L.

Let us repeat the computations for the vector

$$\mathbf{x}_2 = (1 \quad -1 \quad -1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad 1)^T,$$

do steps 3 to 10.

Step 3. Compute net_j for $j = 1, 2$, and 3:

$$\text{net}_1 = b_1 + \sum_i x_i W_{1i} \qquad i = 1, 2, \ldots, 9$$

$$= \tfrac{9}{2} - 0.5 = 4$$

$$\text{net}_2 = b_2 + \sum_i x_i W_{2i}$$

$$= \tfrac{9}{2} + 5 \times 0.5 = 7$$

$$\text{net}_3 = b_3 + \sum_i x_i W_{3i}$$

$$= \tfrac{9}{2} + 0.5 = 5.$$

Step 4. Initialize the outputs:

$$y_1(0) = 4$$
$$y_2(0) = 7$$
$$y_3(0) = 5$$

Step 5. Maxnet compares the outputs of net_j and picks up the largest one. Since $y_2(0) > y_1(0)$ and $y_3(0)$, the Hamming and Maxnet will find that unit y_2 has the best match prototype vector $\mathbf{c}_2$ for the input vector:

$$\mathbf{x} = (1 \quad -1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad 1)^T$$

Step 6. Recurrent processing of the Maxnet follows; $k = 0$.

$$\mathbf{net}^0 = \begin{vmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{vmatrix} \begin{vmatrix} 4 \\ 7 \\ 5 \end{vmatrix} = \begin{vmatrix} 1.6 \\ 5.2 \\ 2.8 \end{vmatrix}$$

$$\mathbf{y}^1 = f(\mathbf{net})^0 = \begin{vmatrix} 1.6 \\ 5.6 \\ 2.8 \end{vmatrix}$$

Step 7. $k = 1$.

$$\mathbf{net}^1 = \begin{vmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{vmatrix} \begin{vmatrix} 1.6 \\ 5.2 \\ 2.8 \end{vmatrix} = \begin{vmatrix} 0 \\ 4.32 \\ 1.44 \end{vmatrix}$$

$$\mathbf{y}^2 = f(\mathbf{net}^1) = \begin{vmatrix} 0 \\ 4.32 \\ 1.44 \end{vmatrix}$$

Step 8. $k = 2$.

$$\mathbf{net}^2 = \begin{vmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{vmatrix} \begin{vmatrix} 0 \\ 4.32 \\ 1.44 \end{vmatrix} = \begin{vmatrix} -1.15 \\ 4.03 \\ 0.58 \end{vmatrix}$$

$$\mathbf{y}^3 = f(\mathbf{net}^2) = \begin{vmatrix} -1.15 \\ 4.03 \\ 0.58 \end{vmatrix}$$

Step 9. $k = 3$.

$$\mathbf{net}^3 = \begin{vmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{vmatrix} \begin{vmatrix} -1.15 \\ 4.03 \\ 0.58 \end{vmatrix} = \begin{vmatrix} -2.07 \\ 4.14 \\ 0 \end{vmatrix}$$

$$\mathbf{y}^4 = f(\mathbf{net}^3) = \begin{vmatrix} -2.07 \\ 4.14 \\ 0 \end{vmatrix}$$

Step 10. $k = 4$.

$$\mathbf{net}^4 = \begin{vmatrix} 1 & 0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{vmatrix} \begin{vmatrix} -2.07 \\ 4.14 \\ 0 \end{vmatrix} = \begin{vmatrix} -2.9 \\ 4.55 \\ -0.41 \end{vmatrix}$$

$$\mathbf{y}^4 = f(\mathbf{net}^4) = \begin{vmatrix} 0 \\ 4.55 \\ 0 \end{vmatrix}$$

The best-matched prototype character vector $\mathbf{c}_2$ is found. This helps identify the unknown input pattern as the character H. Note that the number of steps involved in the recurrent processing of the Maxnet for each input pattern may not be the same.

10.2 KOHONEN SELF-ORGANIZING FEATURE MAP [KOHONEN, 1987]

What we have discussed so far assumed that the synapses $\mathbf{w}_i$ are not interconnected. As a matter of fact, this assumption is not really valid according to the research from the physiologists and psychologists. Many biological neural networks in the brain are found to be essentially two-dimensional layers of processing neurons densely interconnected. Every input neuron is connected to every output neuron through a variable connecting weight, and these output neurons are extensively interconnected with many local (lateral) connections. The strengths of the lateral connection depend on the distance between the neuron and its neighboring neurons. The self-feedback produced by each biological neuron connecting to itself is positive, while the neighboring neurons would produce positive (excitatory) or negative (inhibitory) action depending on the distance from the activation neuron.

These networks assume a topological structure among the cluster units. They are known as the self-organizing feature maps (SOFM), or topology preserving maps from the fact that the weights (from input node i to output node j) will be organized such that topologically close nodes are sensitive to inputs that are physically similar. Output nodes will thus be ordered in a natural way. The low-level organization in this feature map is generally predetermined while some of the organization at higher levels is created during learning by algorithms that promote self-organization. In this sense Kohonen (1982) claims that this self-organizing feature map is similar to those that occur in the brain.

The Kohonen self-organizing feature map (SOFM) is based on competitive learning. The output neurons of the network compete among themselves to be activated or fired. Through training of the network, the single excitations of points would successfully map into single peaks of neuron responses at positions directly above the excitation, thus causing the feature array to self-organize. That is, when a neuron wins on the current input vector $\mathbf{x}$, *all* the synapses in its neighborhood will be updated resulting that only one output neuron is on at any one time.

The self-organizing feature map (SOFM) algorithm developed by Kohonen (1987) serves such a purpose to transform an incoming signal pattern of arbitrary dimension into a one- or two-dimensional discrete map and to perform this transformation adaptively in a topological order fashion. Figure 10.3 shows the architecture of a Kohonen self-organizing feature map.

The way of inducing a winner-takes-all competition among the output neurons is to use lateral inhibitory connections between them. Figure 10.4 shows a one-dimension lattice of neurons with both feedforward and lateral feedback connections. $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ in Figure 10.4 is the external input excitation signal applied to the network, where n is the number of input terminals and $w_{j1}, w_{j2}, \ldots, w_{jn}$ are the corresponding weights of neuron j. The lateral feedback is usually described by a Mexican hat function, as shown in Figure 10.5. From these two figures, we can see that some feedback are positive when the neurons are within the range as indicated by (1) in Figure 10.5, while some feedback are negative when they are farther away from the region as indicated by (2) in the figure. In the area indicated by (3), weak excitation is usually ignored.

To consider the lateral interaction, let us use $\gamma_{j,-K}, \ldots, \gamma_{j,-1}, \gamma_{j,0}, \gamma_{j1}, \gamma_{j2}, \ldots, \gamma_{jK}$ to denote the lateral feedback weights connected to neuron j, and y_1,

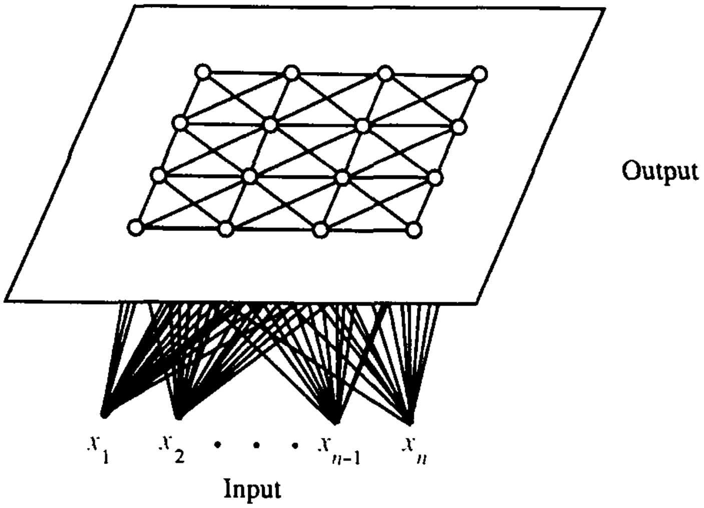


FIGURE 10.3 Kohonen self-organızing feature map.

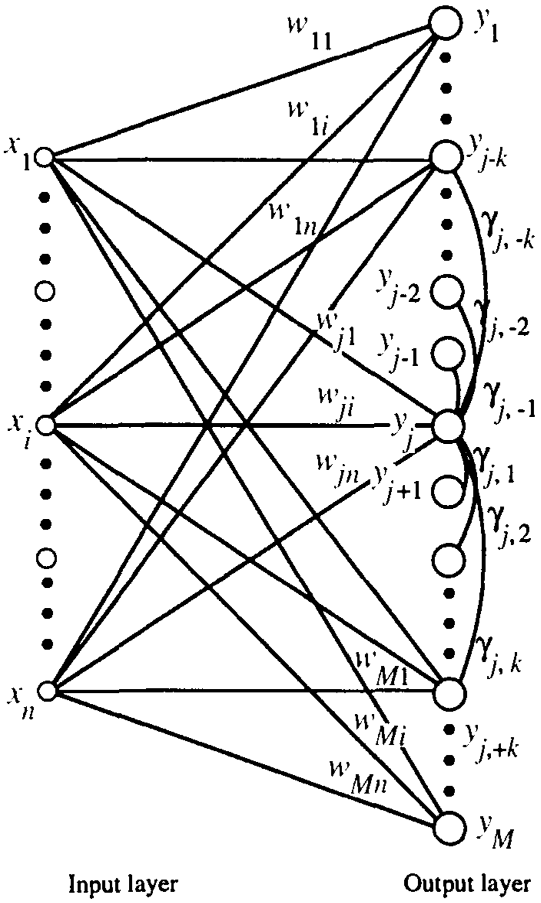


FIGURE 10.4 A one-dimensıonal lattice of neutrons with feed forward and lateral feedback connections.

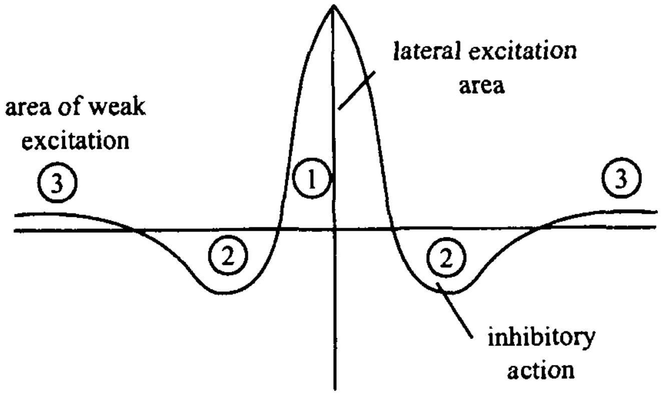


FIGURE 10.5 The Mexican hat function of lateral- and self-feedback strength.

$y_2, \ldots, y_m$ to denote the output signals of the network, where the subscript m is the number of neurons in the network. The output response y_j of neuron j in the output layer is then composed of two parts; one part is from the input signal (the external stimulus signal ϑ_j) on the neuron j of the network, that is,

$$\vartheta_j = \sum_{i=1}^{n} w_{ji} x_i \qquad i = 1, 2, \ldots, n \qquad (10.13)$$

and the other part is from the lateral interaction of neurons $(\rho_j = \sum_{k=-K}^{K} \gamma_{jk} y_{j+k})$ in the output layer. Some of them are excitatory, while some are inhibitory depending on which part of the Mexican hat area they are located. The output response of neuron j of the network y_j is then

$$y_j = f\left(\vartheta_j + \sum_{k=-K}^{K} \gamma_{jk} y_{j+K}\right) \qquad j = 1, 2, \ldots, M \qquad (10.14)$$

or

$$y_j = f\left(\sum_{i=1}^{n} w_{ji} x_i + \sum_{k=-K}^{K} \gamma_{jk} y_{j+K}\right) \qquad j = 1, 2, \ldots, M \qquad (10.15)$$

where $f(.)$ is a nonlinear limiting function. The iteration process will be carried out to find a solution for this nonlinear equation. The output response of neuron j at $(t + 1)$st iteration is

$$y_j(t + 1) = f[\vartheta_j(t + 1) + \alpha \sum_{k=-K}^{K} \gamma_{jk} y_{j+K}(t)] \qquad j = 1, 2, \ldots, M \qquad (10.16)$$

where t is the discrete time and α is a feedback factor to control the rate of convergence of the relaxation process. γ_{jk} in the above equation is the feedback coefficient as a function of interneuronal distance.

Referring to Figure 10.4, there are m cluster units arranged in a linear array structure. This network is to cluster N pattern vectors: $\mathbf{x}_k = (x_{k1}, x_{k2}, \ldots, x_{kn})^T$, $k = 1, 2, \ldots, N$ into these clusters. $\mathbf{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jn})^T$, $j = 1, 2, \ldots, M$, where $\mathbf{w}_j$ is the synaptic weight vector of neuron j in the output layer.

To find the neuron in the output layer that will best match the input vector $\mathbf{x}$, we simply compare the inner products $\mathbf{w}_j^T\mathbf{x}$ for $j = 1, 2, \ldots, M$ and select the largest one.

The above best-matching criterion is equivalent to the minimum euclidean distance between vectors, or

$$|\mathbf{x} - \mathbf{w}_j| = \min_i |\mathbf{x} - \mathbf{w}_i| \qquad i = 1, 2, \ldots, M \tag{10.17}$$

when neuron j is the neuron that best matches the input vector $\mathbf{x}$, where $|\mathbf{x} - \mathbf{w}_j|$ is the euclidean norm of the vector. This particular neuron j is called the best matching or winning neuron for the input vector $\mathbf{x}$. By using this approach, a continuous input space is mapped onto a discrete set of neurons.

Searching for the minimum among the M distances $|\mathbf{x} - \mathbf{w}_i|$, $i = 1, 2, \ldots, M$ is equivalent to finding the maximum among the scalar $\mathbf{w}_i^T\mathbf{x}$, $i = 1, 2 \ldots, M$, or

$$\mathbf{w}_j^T\mathbf{x} = \max_i(\mathbf{w}_i^T\mathbf{x}) \qquad i = 1, 2, \ldots, M \tag{10.18}$$

$\mathbf{w}_j^T\mathbf{x}$ will then be the activation value of the "winning" neuron. After the winning neuron has been found, the synaptic weight vector $\mathbf{w}_j$ of neuron j should be modified in relation to the input vector $\mathbf{x}$ so that it becomes more similar (or closer) to the current input vector. That is, the weight of the winning neuron should be changed by an increment of weight in the negative gradient direction. Using the discrete time formulation, $\Delta\mathbf{w}_j(t)$ is therefore:

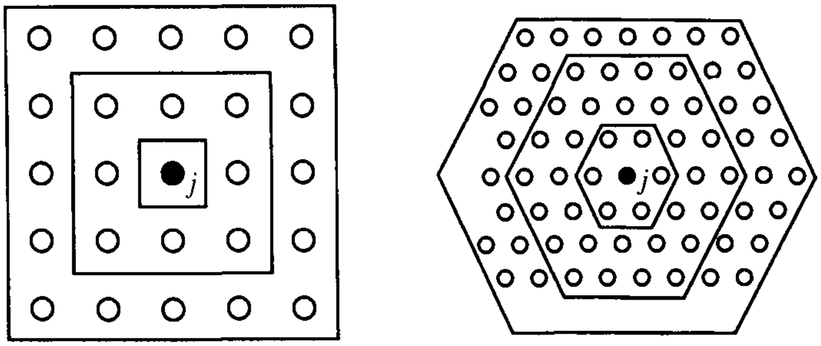
$$\Delta\mathbf{w}_j(t) = \beta[(\mathbf{x}(t) - \mathbf{w}_j(t)] \tag{10.19}$$

and the weight $\mathbf{w}_j$ can be updated by following the following rule:

$$\begin{cases} \mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \beta(t)[\mathbf{x} - \mathbf{w}_j(t)] & \text{when } j \text{ is the winning neuron} \\ \mathbf{w}_i(t+1) = \mathbf{w}_i(t) & \text{when } i \neq j \end{cases} \tag{10.20}$$

where $\mathbf{w}_j(t)$ is the synaptic weight vector $\mathbf{w}_j$ of neuron j at discrete time t; $\mathbf{w}_j(t+1)$ is the updated value at time $t+1$; and $\beta(t)$ is the learning rate control parameter and is varied dynamically during learning for best results.

Figure 10.6 shows two planar arrays of neurons with rectangular and hexagonal neighborhoods. Input $\mathbf{x}$ is applied simultaneously to all nodes. The spatial neighborhood Λ_j is used here as a measure of similarity between $\mathbf{x}$ and $\mathbf{w}_j$. Following the argument of the winner-take-all, the weights affecting the currently



Winning neuron is j

(a) (b)

FIGURE 10.6 Neighborhoods around a winning neuron j. (a) Rectangular grid; (b) hexagonal grid.

winning neighbourhood Λ_j undergo adaptation at the current learning step; other weights remain unchanged.

The radius of Λ_j should be decreasing as the training progresses, $\Lambda_j(t_1) > \Lambda_j(t_2) > \Lambda_j(t_3) \cdots$, where $t_1 < t_2 < t_3 < \cdots$. The radius can be very large as learning starts, since it may be needed for initial global ordering of weights. Toward the end of the training, the neighborhood may involve no cells other than the central winning one.

Kohonen Self-Organizing Feature Map (SOFM) Algorithm

Step 1. *Initialization:* Initialize weights w_{ji}. To start with, randomly set these weights from input (n in number) to outputs (M in number) as small values. Set topological neighborhood parameters $\Lambda_M(t)$. Set the learning rate parameter $\beta(t)$ in the range from 0 to 1. While stopping condition is false, do steps 2 to 8.

Step 2. Similarity matching: Find the best matching neuron (winning neuron j) of the output layer at time t for each input vector $\mathbf{x}$. Do steps 3 and 4.

Step 3. For each neuron $j, j = 1, 2, \ldots, M$ of the output layer, compute the euclidean distance d_j between the inputs and the output neuron j with

$$d(j) = \sum_{i=1}^{n}[w_{ji}(t) - x_i(t)]^2 \qquad j = 1, 2, \ldots, M$$

where $x_i(t)$, $i = 1, 2, \ldots, n$, is the input to node i at time t, and $w_{ji}(t)$ is the weight from input node i to output node j at time t.

Step 4. Use this distance measure to select the output node with minimum d_j as the winning neuron.

Step 5. Update the synaptic weight vector of all neurons within a specified neighborhood of the winning neuron through the following iteration [see Eq. (10.20)]:

$$\mathbf{w}_j(t+1) = \begin{cases} \mathbf{w}_j(t) + \beta(t)[\mathbf{x} - \mathbf{w}_j(t)] & j \in \Lambda_{\mathrm{wn}}(t) \\ \mathbf{w}_j(t) & \text{otherwise} \end{cases} \tag{10.21}$$

or

$$w_{ji}(t+1) = w_{ji}(t) + \beta(t)[x_i(t) - w_{ji}(t)] \qquad i = 1, 2, \ldots, n \tag{10.22}$$

where $\Lambda_{\mathrm{wn}}(t)$ is the neighborhood function centered around the winning neuron; $\beta(t)$ is a learning rate control factor, and decreases with time. It ranges from 0 to 1.

Step 6. Update learning rate $\beta(t)$.

Step 7. Reduce the radius of $\Lambda_{\mathrm{wn}}(t)$. j is assumed to be the winning neuron in this algorithm.

Step 8. Test stopping condition and continue with step 2.

From the procedure described above, it can be seen that this process of feature map forming is similar to the K-means clustering algorithm. No information concerning the correct class is needed during adaptive training.

Let us close this chapter with an example given by Kohonen. His example is for the mapping of a five-dimensional feature vectors. Figure 10.7a shows the training set samples of 32 different (five-dimensional input vectors labeled A to Z and 1 to 6. There are 70 neurons in the rectangular array. This array was trained in random order with vectors $\mathbf{x}_A$ to $\mathbf{x}_Z$ and $\mathbf{x}_1$ to $\mathbf{x}_6$. The subscripts denote the alphanumeric from A to Z, and 1 to 6. The learning rate control parameter decreases linearly with k from 0.5 to 0.04. After 10,000 training steps, the weights stabilized. When vector $\mathbf{x}_Z$ was input, the upper right corner neuron from Figure 10.7b produced the strongest response and was labeled Z. When vector $\mathbf{x}_A$ was input, the leftmost neuron of the second row produced the strongest response, and so forth. With these 32 five-dimensional vectors input to the network, 32 neurons were therefore labeled as shown in the 7 × 10 rectangular array. For details of this example see Kohonen (1984). The mathematical operation involved

Pattern	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	**1**	**2**	**3**	**4**	**5**	**6**
Components																																
x_1	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
x_2	0	0	0	0	0	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
x_3	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	3	3	3	6	6	6	6	6	6	6	6	6	6	6
x_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	1	2	3	4	2	2	2	2	2	2
x_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6

(a)

B	*C*	*D*	*E*	*	*Q*	*R*	*	*Y*	*Z*
A	*	*	*	*	*P*	*	*	*X*	*
*	*F*	*	*N*	*O*	*	*W*	*	*	**1**
*	*G*	*	*M*	*	*	*	*	**2**	*
H	*K*	*L*	*	*T*	*U*	*	**3**	*	*
*	*I*	*	*	*	*	*	*	**4**	*
*	*J*	*	*S*	*	*	*V*	*	**5**	**6**

(b)

FIGURE 10.7 Sample results for self-feature mapping. (a) List of five-dimensional pattern data; (b) feature map produced by SOFM. (From Kohonen, 1984.)

in SOFM is enormous. Highly parallel processing techniques need to be developed.

PROBLEMS

10.1 Repeat the example given on page 240 respectively with $\varepsilon = 0.3$. Compare their effectiveness in terms of number of cycles needed to suppress the weak nodes to zero.

10.2 Assume that the four characters A, E, P, and Y can be represented, respectively, by 3 × 5 matrices, as shown in Figure P10.2. Design a minimum Hamming distance classifier (with maxnet as the second layer in the classifier) to classify them.

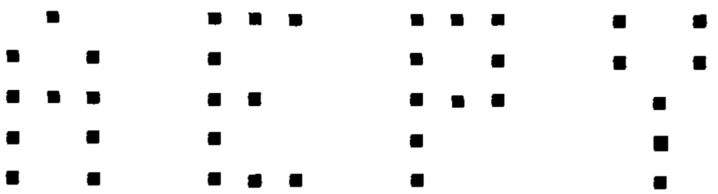


FIGURE P10.2

10.3 Write a program in C for a minimum Hamming distance classifier (with maxnet as the second layer in the classifier) to classify all the printed numerals, 0, 1, 2, ..., 9. Assume that they are represented with 5 × 7 matrices, as shown in Figure P10.3. The program should read input from 5 × 7 array into an **x** vector (a 35-tuple).

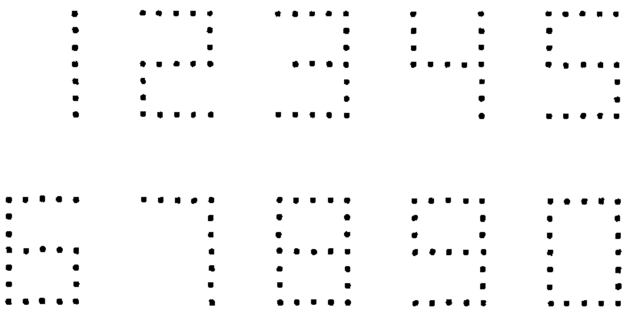


FIGURE P10.3

10.4 Consider a Kohonen self-organizing map with four cluster units and two input units, as shown in Figure P10.4. The initial weights were randomly set as follows:

$$\begin{array}{ll} w_{11} = 0.3 & w_{12} = 0.6 \\ w_{21} = 0.5 & w_{22} = 0.8 \\ w_{31} = 0.5 & w_{32} = 0.4 \\ w_{41} = 0.8 & w_{42} = 0.3 \end{array}$$

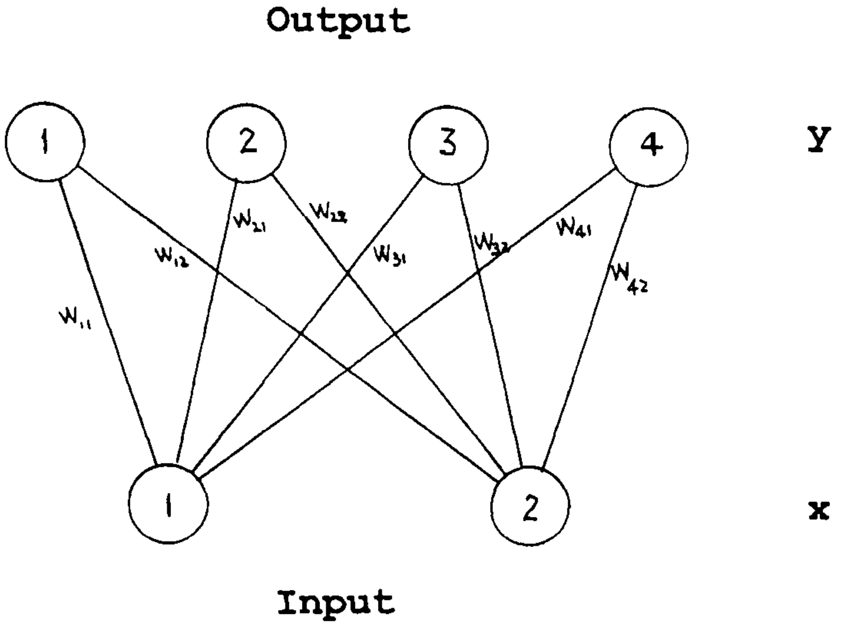


FIGURE P10.4

(a) Find the cluster unit $\mathbf{y}$ that is closest to the input $\mathbf{x} = (0.5, 0.3)$.

(b) Find the new weights for the winning unit when the learning rate ε is chosen as 0.1.

(c) Find also the new weights for the other three cluster units.

(d) Repeat (a) when a learning rate of 0.3 is used.

10.5 Write a computer program to implement a Kohonen self-organizing map neural net. Use the data shown in Figure 10.7 to test your program.

11

The Hopfield Model

11.1 THE HOPFIELD MODEL

The Hopfield model is one type of iterative (or recurrent) autoassociative network. It is a single-layer net with connections among units to form a closed loop. The output of each processing element (neuron) is fully connected to the inputs by weights. Positive weights are excitatory and will strengthen connections; negative weights are inhibitory, weakening connections. This feedback provides full trainability and adaptability, and the iterative (or recurrent) operation provides the necessary nonlinearity. With such a design, this net could retrieve a pattern that was stored in the memory to respond to the presentation of an incomplete or noisy version of that pattern. In this sense the Hopfield net was claimed to possess a property close to the brain.

In this model Hopfield (Hopfield, 1986) presents memory in terms of an energy function. He incorporates asynchronous processing for an individual processing element (neuron) so that for a given instant of time, only a single neuron updates its output. In other words, under asynchronous operation of the network, each element of the output vector is updated separately, while taking into account the most recent values for the elements that have already been updated. This asynchronous operation, other than the synchronous operation, is really needed for the net to work properly. Otherwise, the system would not stabilize to a fixed state. Kamp and Hasler (1990) have shown that if the synchronous state updating algorithm is chosen, it can lead to persistingly cycle states that consist of two complementary pattern states rather than a single equilibrium one, and these two complementary patterns correspond to identical energy levels. With proper system architecture designed and weights carefully selected for the model, the net could recall the desired response pattern when given an input stimulus that is similar, but not identical, to the training pattern.

Figure 11.1 shows a schematic diagram of a Hopfield net. It is a single-layer feedback neural network. In this diagram $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]$ is the n-dimensional input vector, $\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_n]$ is the output, and $u_1, u_2, \ldots, u_n$ are nodes representing the intermediate status of the output during iterations. The nodes contain hard-limiting nonlinearity. Binary input and output take on values +1 and −1. From this diagram we can see that there are feedbacks from the output of each node to all other nodes except itself during the iteration operation. These feedbacks are through the weights w_{ji} (from output of node i to input of node j; $i, j = 1, 2, \ldots, n$; and $i \neq j$). w_{ji} specifies the contribution of the output signal y_i of the neuron i to the net potential acting on neuron j. The net has symmetric weights with no self-connection, i.e.,

$$w_{ij} = w_{ji} \tag{11.1}$$

and

$$w_{ii} = 0 \tag{11.2}$$

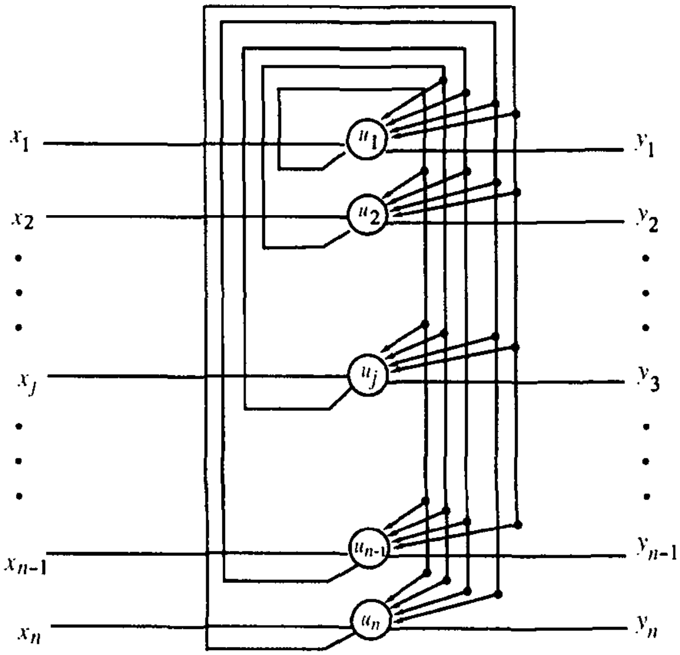


FIGURE 11.1 Architecture of the Hopfield net.

The net potential net_j acting on the neuron j is the sum of all postsynaptic potentials delivered to it, as illustrated in Figure 11.1, or

$$\text{net}_j = \sum_{\substack{i=1 \\ i \neq i}}^{n} w_{ji} y_i + x_j - \theta_j \qquad i = 1, 2, \ldots, n; j \neq i \tag{11.3}$$

where x_j is the external input to neuron j and y_i is the output of the neuron $i, i = 1, 2, \ldots, n, j \neq i$. θ_j is a threshold applied externally to neuron j. As stated at the beginning of this section, only asynchronous updating of the units is allowed to ensure the net converging to a stable set of activation. It follows that only one unit updates its activation at a time.

When we want to store an information (say pattern **x**) to the network, we apply the pattern $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]$ to the net. The network's output $\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_n]$ will be initialized accordingly. After this forward processing, the pattern $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]$ initially applied as input to the net is removed. Through the feedback connections the initialized output $\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_n]$ will become the updated input to the net. The first updated input forces the first updated output. This, in turn, produces the second updated input, and the second updated output response. This sequential updating process continues. When no new updated response is produced, the updating process will terminate. The network is then said to be in equilibrium. The output vector shown at the output will be the stored vector that most matches the input vector **x**.

11.2 AN ILLUSTRATIVE EXAMPLE FOR THE EXPLANATION OF THE HOPFIELD NETWORK OPERATION

Let us first explain the process by means of a numerical example and then generalize the process. Use a 5 × 5 matrix to code the alphanumeric characters from A to Z and 0 to 9. The code for the character C is

$$[1\ 1\ 1\ 1\ 1 \quad 1\ -1\ -1\ -1\ -1 \quad 1\ -1\ -1\ -1\ -1 \quad 1 -1\ -1\ -1\ -1 \quad 1\ 1\ 1\ 1\ 1]$$

01	02	03	04	05	*	*	*	*	*
06	07	08	09	10	*				
11	12	13	14	15	*				
16	17	18	19	20	*				
21	22	23	24	25	*	*	*	*	*
		(a)					(b)		

$$[1\ 1\ 1\ 1\ 1\quad 1\ -1\ -1\ -1\ -1\quad 1\ -1\ -1\ -1\ -1\quad 1\ -1\ -1\ -1\ -1\quad 1\ 1\ 1\ 1\ 1]$$

(c)

FIGURE 11.2 A 5 × 5 matrix for alphanumeric character coding. (a) Coding scheme; (b) character C; (c) codes for C.

(see Figure 11.2). For the case where only the binary code for character C is stored in the net, the weight matrix is:

$$\mathbf{W} = \left|\begin{array}{ccccccccccccccccccccccccc}
0 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 0 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 0 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 0 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 0 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 0 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 0 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 0 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 0 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 0 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 0 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 0 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 0
\end{array}\right|$$

Derivation of the above weight matrix will be discussed in Section 11.3. Let us use this weight matrix for the time being. Suppose we have here three noisy patterns, namely.

Noisy pattern 1. A noisy character C and its encoding. The signs Δ and ↑ indicate, respectively, the noise on the character image and also at its codes.

```
* * * * *
*     Δ
*   Δ
*
* * * * *
```

$$\mathbf{x}_1^T = [1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ \underset{\uparrow}{1}\ -1\ 1\ -1\ \underset{\uparrow}{1}\ -1\ -1\ 1\ -1\ -1\ -1\ -1\ 1\ 1\ 1\ 1\ 1]$$

Noisy pattern 2. A noisy character C and its encoding.

```
* * * * *
* Δ
*     Δ
*   Δ
* * * * *
```

$$\mathbf{x}_2^T = [1\ 1\ 1\ 1\ 1\ 1\ \underset{\uparrow}{1}\ -1\ -1\ -1\ 1\ -1\ -1\ \underset{\uparrow}{1}\ -1\ 1\ -1\ \underset{\uparrow}{1}\ -1\ -1\ 1\ 1\ 1\ 1\ 1]$$

Noisy pattern 3. A very noisy character C and its encoding.

```
* * * * *
*     Δ
*   Δ
*   Δ   Δ
* * * * *
```

$$\mathbf{x}_3^T = [1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ \underset{\uparrow}{1}\ -1\ 1\ -1\ \underset{\uparrow}{1}\ -1\ -1\ 1\ -1\ \underset{\uparrow}{1}\ -1\ \underset{\uparrow}{1}\ 1\ 1\ 1\ 1\ 1\}$$

Let us present the noisy pattern 1,

$$\mathbf{x}_1^T = [1\ 1\ 1\ 1\ 1\quad 1\ -1\ -1\ 1\ -1\quad 1\ -1\ 1\ -1\ 1\quad 1\ -1\ -1\ -1\ -1\quad 1\ 1\ 1\ 1\ 1],$$

to the network and see whether it can recognize it. The output of the network will be $\mathbf{x}_1^T\mathbf{W}$. It is

$$\mathbf{x}_1^T\mathbf{W} = [1\ 1\ 1\ 1\ 1\quad 1\ -1\ -1\ 1\ -1\quad 1\ -1\ 1\ -1\ -1\quad 1\ -1\ -1\ -1\ -1\quad 1\ 1\ 1\ 1\ 1]|\mathbf{W}|$$

After the activation of the signum function, output of the net is

$$\mathbf{y} = \text{sgn}\{\mathbf{x}_1^T\mathbf{W}\}$$
$$\Rightarrow [1\ 1\ 1\ 1\ 1\quad 1\ -1\ -1\ -1\ -1\quad 1\ -1\ -1\ -1\ -1\quad 1\ -1\ -1\ -1\ -1\quad 1\ 1\ 1\ 1\ 1]^T$$

where sgn is the signum function and is

$$\text{sgn}\{\mathbf{x}_1^T\mathbf{W}\} = \begin{cases} +1 & \text{if } \mathbf{x}^T\mathbf{W} > 0 \\ -1 & \text{if } \mathbf{x}^T\mathbf{W} < 0 \end{cases}$$

Thus, when the input noisy pattern

$$\mathbf{x}^T = [1\ 1\ 1\ 1\ 1\quad 1\ -1\ -1\ 1\ -1\quad 1\ -1\ 1\ -1\ -1\quad 1\ -1\ -1\ -1\ -1\quad 1\ 1\ 1\ 1\ 1]$$

is applied to the input of the network, the net produces the "known" pattern vector

$$[1\ 1\ 1\ 1\ 1\quad 1\ -1\ -1\ -1\ -1\quad 1\ -1\ -1\ -1\ -1\quad 1\ -1\ -1\ -1\ -1\quad 1\ 1\ 1\ 1\ 1],$$

which was stored in the network, as its response, thus recognizing the noisy pattern as the character C.

Let us try noisy pattern 2,

$$\mathbf{x}_2 = [1\ 1\ 1\ 1\ 1\quad 1\ 1\ -1\ -1\ -1\quad 1\ -1\ -1\ 1-1\quad 1\ -1\ 1\ -1\ -1\quad 1\ 1\ 1\ 1\ 1].$$

The output of the network will be $\mathbf{x}_2^T\mathbf{W}$:

$$\mathbf{x}_2^T\mathbf{W} = [1\ 1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ -1\ 1\ -1\ -1\ 1\ -1\ 1\ -1\ 1\ -1\ -1\ 1\ 1\ 1\ 1\ 1]|\mathbf{W}|$$

and the output of the net after the operation of the hard limiter activation is:

$$\begin{aligned}\mathbf{y} &= \operatorname{sgn}\{\mathbf{x}_2^T\mathbf{W}\}\\ &\Rightarrow [1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1\ -1\ -1\ -1\ 1\ -1\ -1\ -1\ -1\ 1\ 1\ 1\ 1\ 1]'\end{aligned}$$

As with the first testing input, the net recognizes the noisy pattern vector 2,

$$\mathbf{x}_2 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ -1\ 1\ -1\ -1\ 1\ -1\ 1\ -1\ 1\ -1\ -1\ 1\ 1\ 1\ 1\ 1],$$

as the "known" pattern

$$[1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1\ -1\ -1\ -1\ 1\ -1\ -1\ -1\ -1\ 1\ 1\ 1\ 1\ 1]$$

after the hard limiter, and identifies it as character C. This is also correct.

Let us try the third unknown pattern,

$$\mathbf{x}_3^T = [1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ 1\ -1\ 1\ -1\ 1\ -1\ -1\ 1\ -1\ 1\ -1\ 1\ 1\ 1\ 1\ 1\ 1],$$

that has more noise on it. The output of the network $y = \operatorname{sgn}\{\mathbf{x}_3^T\mathbf{W}\}$ after the hard limiter is

$$\begin{aligned}\mathbf{y} &= \operatorname{sgn}\{\mathbf{x}_3^T\mathbf{W}\}\\ &= [1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ 1\ -1\ 1\ -1\ 1\ -1\ -1\ 1\ -1\ 1\ -1\ 1\ 1\ 1\ 1\ 1\ 1]|\mathbf{W}|\end{aligned}$$

or

$$\begin{aligned}\mathbf{y} &= \operatorname{sgn}(\mathbf{x}_3^T\mathbf{W})\\ &\Rightarrow [1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1\ -1\ -1\ -1\ 1\ -1\ -1\ -1\ -1\ 1\ 1\ 1\ 1\ 1]'\end{aligned}$$

Again, the net produces the known pattern at the output. It means that the net still can recognize this unknown input pattern as character C even when it is very noisy.

11.3 OPERATION OF THE HOPFIELD NETWORK

From the testing results obtained from the illustrative example, it is obvious that the kernel of this process is to select the appropriate weight matrix. Keep in mind that in the Hopfield net (Figure 11.1) output of each neuron is fed back to all other neurons and that the weight matrix used is a symmetric matrix with zero diagonal entries. We also remember that this model is to recall the desired response pattern when presented an unknown pattern that is similar but not identical to the stored one. We can then summarize that in the operation of the Hopfield network there exist two phases, namely, the storage phase and the retrieval phase.

Storage phase. Suppose we like to store a set of l bipolar pattern vectors, S_μ, $\mu = 1, 2, \ldots, l$ as the patterns to be memorized by the net. According to Hebb's postulate of learning [Hebb 1949], the synaptic weight connected from neuron i to neuron j is given as

$$w_{ji} = \frac{1}{n}\sum_{\mu=1}^{l} \mathbf{s}_{\mu j}\mathbf{s}_{\mu i} \qquad \mu = 1, 2, \ldots, l; \; i, j = 1, 2, \ldots, n \tag{11.4}$$

where $1/n$ is the constant of proportionality for the $n \times n$ synaptic weight matrix. It is so used here to simplify the formulation of the weight matrix expression. w_{ii} is set as zero for the Hopfield network in the normal operation. Equation (11.4) can then be put in matrix form:

$$\mathbf{W} = \frac{1}{n}\sum_{\mu=1}^{l} \mathbf{s}_{\mu}\mathbf{s}_{\mu}^{T} - \frac{k}{n}\mathbf{I} \quad \mu = 1, 2, \ldots, l \tag{11.5}$$

where $\mathbf{W}$ is the weight matrix; $\mathbf{s}_{\mu}\mathbf{s}_{\mu}^{T}$ represents the outer product of the vector $\mathbf{s}_{\mu}$ with itself, $\mathbf{I}$ is an identity matrix; and l is the number of the pattern vectors stored in the memory. The weight matrix as obtained in previous section with character C stored to the net was exactly obtained in this way [see Eq. (11.5)].

When we want to store few more characters, say H, L, and E, into the net, the memory will have four stored coded pattern vectors and the weight matrix becomes

$$\mathbf{W} = \tfrac{1}{5}[\mathbf{S}_{\mathrm{C}}\mathbf{S}_{\mathrm{C}}^{T} + \mathbf{S}_{\mathrm{H}}\mathbf{S}_{\mathrm{H}}^{T} + \mathbf{S}_{\mathrm{L}}\mathbf{S}_{\mathrm{L}}^{T} + \mathbf{S}_{\mathrm{E}}\mathbf{S}_{\mathrm{E}}^{T}] - \tfrac{4}{5}\mathbf{I} \tag{11.6}$$

where $\mathbf{S}_{\mathrm{C}}$ is the pattern for character C stored in the net, and $\mathbf{S}_{\mathrm{C}}\mathbf{S}_{\mathrm{C}}$ has been found as:

$$\mathbf{S}_{\mathrm{C}}\mathbf{S}_{\mathrm{C}}^{T} =$$

$$\left.\begin{matrix}
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
\\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
\\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
\\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \\
\\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & 1
\end{matrix}\right| \tag{11.7}$$

Similarly, $\mathbf{S}_H$ stands for the pattern for character H. $\mathbf{S}_H\mathbf{S}_H^T$ is then

$$\mathbf{S}_H\mathbf{S}_H = \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \end{bmatrix} \tag{11.8}$$

or

$$\mathbf{S}_H\mathbf{S}_H = \begin{bmatrix}
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1
\end{bmatrix} \tag{11.9}$$

Let us consider only two characters C and H stored in the net for illustration. The weight matrix $\mathbf{W}_{\mathrm{CH}}$ is then

$$\mathbf{W}_{\mathrm{CH}} = \left(\frac{1}{5}\right)[\mathbf{S}_{\mathrm{C}}\mathbf{S}_{\mathrm{C}}^{T} + \mathbf{S}_{\mathrm{H}}\mathbf{S}_{\mathrm{H}}^{T}] - \left(\frac{2}{5}\right)\mathbf{I} \tag{11.10}$$

or

$$\mathbf{W}_{\mathrm{CH}} = \frac{1}{5}\begin{vmatrix}
0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 1 \\[1ex]
1 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 1 \\
-1 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & -1 \\
-1 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & -1 \\
-1 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & -1 \\
0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 \\[1ex]
1 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 \\
0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 \\
0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 \\
0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 \\[1ex]
1 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 1 \\
-1 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & -1 \\
-1 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & -1 \\
-1 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\
0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 \\[1ex]
1 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0
\end{vmatrix} \tag{11.11}$$

Retrieval phase. During the retrieval phase, let us present an n-dimensional unknown vector $\mathbf{x}$, which frequently represents an incomplete or noisy version of a stored vector in the net, to the Hopfield network. Information retrieval the proceeds in accordance with a dynamic rule. The feedback inputs to the jth neuron are equal to the weighted sum of neuron outputs y_i, $i = 1, 2, \ldots, n$. With the notation w_{ji} as the weights connecting the output of the ith neuron with the input of the jth neuron, net_j can be expressed as

$$\mathrm{net}_j = \sum_{\substack{i=1 \\ j \neq i}}^{n} w_{ji} y_i + x_j - \theta_j \qquad i, j = 1, 2, \ldots, n;\ j \neq i. \tag{11.12}$$

where x_j represents the external input to the jth neuron. x_j will be removed right after the iteration operation starts. In matrix form,

$$\mathbf{net} = \mathbf{W}\mathbf{y} + \mathbf{x} - \theta \tag{11.13}$$

where

$$\mathbf{net} = \begin{vmatrix} \text{net}_1 \\ \vdots \\ \text{net}_j \\ \vdots \\ \text{net}_n \end{vmatrix} \qquad \mathbf{x} = \begin{vmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{vmatrix} \qquad \mathbf{y} = \begin{vmatrix} y_1 \\ \vdots \\ y_j \\ \vdots \\ y_n \end{vmatrix} \qquad \theta = \begin{vmatrix} \theta_1 \\ \vdots \\ \theta_j \\ \vdots \\ \theta_n \end{vmatrix} \tag{11.14}$$

and

$$\mathbf{W} = \begin{vmatrix} 0 & w_{12} & \cdots & w_{1n} \\ w_{21} & 0 & & w_{2n} \\ \vdots & & \ddots & \\ w_{n1} & w_{n2} & \cdots & 0 \end{vmatrix} \tag{11.15}$$

There is a signum function (a hard limiter, abbreviated as "sgn") at the output of each neuron. This causes the following response of the jth neuron:

$$\begin{aligned} y_j &\Rightarrow -1 \qquad \text{if net}_j < 0 \\ y_j &\Rightarrow +1 \qquad \text{if net}_j > 0 \end{aligned} \tag{11.16}$$

Once again, note that the updating procedure described here will be continued in an asynchronous mode. For a given instant of time, only a single neuron is allowed to update its output, and only one entry in vector **y** is allowed to change. The next update in a series uses the already updated vector **y**. We can then formulate the update rule as follows:

$$\mathrm{y}_j(k+1) = \text{sgn}[\mathbf{w}_j^T \mathbf{y}(k) - \theta_j] \qquad j = 1, 2, \ldots, n; \; k = 0, 1, \ldots \tag{11.17}$$

where k represents the iteration step. Note that x_j is removed right after the iteration starts, and hence it no longer appears in the above expression. Note also that it is required in this asynchronous update procedure that once an updated entry of $\mathbf{y}(k+1)$ has been computed for a particular step, this update will substitute the current value $\mathbf{y}(k)$ for the computation of its subsequent update. The asynchronous (serial) updating procedure described here continues until there are no further changes to report. That is, starting with the input vector **x**, the network finally produces a time-invariant state vector **y** as shown below:

$$y_j = \text{sgn}\left[\sum_{\substack{i=1 \\ i \neq i}}^{n} w_{ji} y_i - \theta_j\right] \qquad i,j = 1, 2, \ldots, n; \; j \neq i \tag{11.18}$$

or in matrix form,

$$\mathbf{y} = \operatorname{sgn}[\mathbf{W}\mathbf{y} - \theta] \tag{11.19}$$

Example. Present

$$\mathbf{x} = [1\ 1\ 1\ 1\ 1\ \ 1\ -1\ -1\ -1\ \ -1\ \ 1\ -1\ -1\ -1\ \ -1\ \ 1\ -1\ -1\ -1\ \ -1\ \ 1\ 1\ 1\ 1\ 1]$$

to the net which has already had two patterns C and H stored in it. Test whether the net can recognize this unknown pattern **x**. Use W_{CH} as the weight matrix [see expression (11.11)] for this problem, and ignore the constant of proportionality, $\frac{1}{5}$, for simplicity.

$$\begin{aligned}\mathbf{x}^T\mathbf{W}_{CH} &= [1\ 1\ 1\ 1\ 1\ \ 1\ -1\ -1\ -1\ \ -1\ \ 1\ -1\ -1\ -1\ \ -1\ \ 1\ -1\ -1\ -1\ \ -1\ \ 1\ 1\ 1\ 1\ 1\}\mathbf{W}_{CH}\\ &= [12\ 7\ 7\ 7\ 12\ \ 12\ -12\ -12\ -12\ -7\ \ 12\ -7\ -7\ -7\ -9\ \ 12\ -12\ -12\ 12\ -9\ \ 12\ 7\ 6\ 7\ 11]\end{aligned}$$

After the signum function, we have

$$\begin{aligned}\mathbf{y}^1 &= \operatorname{sgn}\{\mathbf{x}^T\mathbf{W}_{CH}\}\\ &= [1\ 1\ 1\ 1\ 1\ \ 1\ -1\ -1\ -1\ \ -1\ \ 1\ -1\ -1\ -1\ \ -1\ \ 1\ -1\ -1\ -1\ \ -1\ \ 1\ 1\ 1\ 1\ 1]\end{aligned}$$

This is the code for character C. This signifies that the net can correctly identify that input pattern. Let us input another unknown pattern:

$$\mathbf{x} = [1\ -1\ -1\ -1\ 1\ \ 1\ -1\ -1\ -1\ 1\ \ 1\ 1\ 1\ 1\ 1\ \ 1\ -1\ -1\ -1\ 1\ \ 1\ -1\ -1\ -1\ 1]$$

to the network.

$$\begin{aligned}\mathbf{x}^T\mathbf{W}_{CH} &= [1\ -1\ -1\ -1\ 1\ \ 1\ -1\ -1\ -1\ 1\ \ 1\ 1\ 1\ 1\ 1\ \ 1\ -1\ -1\ -1\ 1\ \ 1\ -1\ -1\ -1\ 1]\mathbf{W}_{CH}\\ &= [12\ -11\ -11\ -11\ 12\ \ 12\ -12\ -12\ -12\ 11\ \ 12\ 11\ 11\ 11\ 11\ \ 12\ -12\ -11\ -12\ 11\ \ 12\\ &\qquad -11\ -8\ -11\ 12].\end{aligned}$$

After the signum function, we have

$$\begin{aligned}\mathbf{y}^1 &= \operatorname{sgn}\{\mathbf{x}^T\mathbf{W}_{CH}\}\\ &= [1\ -1\ -1\ \ -1\ 1\ \ 1\ -1\ -1\ -1\ 1\ \ 1\ 1\ 1\ 1\ 1\ \ 1\ -1\ -1\ -1\ \ -1\ \ 1\ -1\ -1\ -1\ 1\}\end{aligned}$$

The net identifies the input pattern as the character H. It is correct.

Let us input to the net another pattern

$$\mathbf{x} = [1\ 1\ 1\ 1\ 1\ \ 1\ -1\ -1\ 1\ \ -1\ \ 1\ -1\ 1\ -1\ \ -1\ \ 1\ -1\ 1\ -1\ 1\ \ 1\ 1\ 1\ 1\ 1].$$

which is a noisy pattern. Proceeding, we have

$$\begin{aligned}\mathbf{x}^T\mathbf{W}_{CH} &= [1\ 1\ 1\ 1\ 1\ \ 1\ -1\ -1\ 1\ \ -1\ \ 1\ -1\ 1\ -1\ \ -1\ \ 1\ -1\ 1\ -1\ 1\ \ 1\ 1\ 1\ 1\ 1]\mathbf{W}_{CH}\\ &= [8\ 7\ 7\ 7\ 8\ \ 8\ -8\ -8\ -10\ -7\ \ 7\ -7\ -9\ -7\ -5\ \ 8\ -8\ -10\ -8\ -9\ \ 8\ 7\ 7\ 7\ 8\end{aligned}$$

and

$$\begin{aligned}\mathbf{y}^1 &= \operatorname{sgn}\{\mathbf{x}^T\mathbf{W}\}\\ &= [1\ 1\ 1\ 1\ 1\ \ 1\ -1\ -1\ -1\ -1\ \ 1\ -1\ -1\ -1\ -1\ \ 1\ -1\ -1\ -1\ -1\ \ 1\ 1\ 1\ 1\ 1]\end{aligned}$$

This noisy pattern is recognized as character C.

The Hopfield net works quite well. One of the other reasons Hopfield model was so well received was that it could be implemented by integrated circuit hardware. However, it should be stressed that Hopfield model is a discrete time dynamic system. The dynamic system approach to cognitive task is still in an early development stage. The application of the dynamic models to real-time associative problem will require a great deal of further scientific development.

So far several typical neural networks for pattern recognition have been thoroughly discussed. It is our hope that readers have enough knowledge provided to start on this challenging field. Those particular paradigms chosen here were mainly because of their historical significance and their value in illustrating some neural network concepts. Those who are interested in a more comprehensive examination of some other models and the mathematics involved can refer to literature and/or books dedicated in the discussion of the neural networks.

PROBLEMS

11.1 (a) Write a program to implement a discrete Hopfield net to store the numerals shown below:

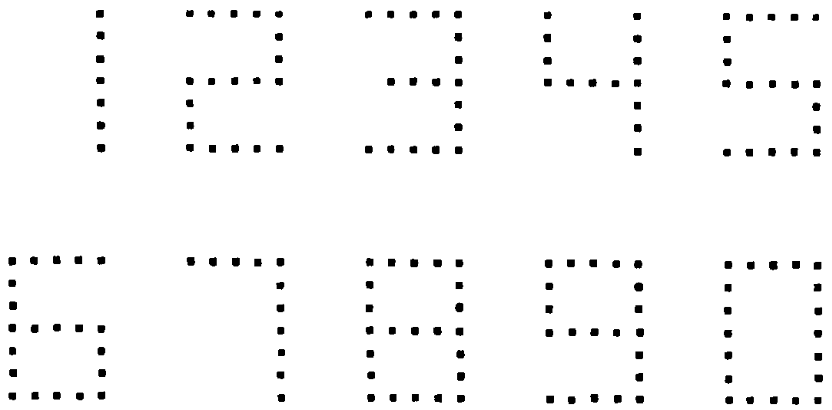


Figure P11.1

(b) How many patterns from the above list can be stored and recalled correctly?

(c) What is the ability of the net to respond correctly to noisy input?

This Page Intentionally Left Blank

Part III

Data Preprocessing for Pictorial Pattern Recognition

This Page Intentionally Left Blank

12

Preprocessing in the Spatial Domain

Image enhancement is an important step in the processing of large data sets to make the results more suitable for classification than were the original data. It accentuates and sharpens the image features, such as edges, boundaries, and contrast. The process does not increase the inherent information content in the data, but it does increase the dynamic range of the features. Because of difficulties experienced in quantifying the criteria for enhancement, and the fact that image enhancement is so problem oriented, no general approaches are available that can be used in every case, although many methods have been suggested.

The approaches suggested for enhancement can be grouped into two main categories: spatial processing and transform processing. In transform processing, the image function is first transformed to the transform domain and then processed to meet the specific problem requirements. Inverse transform is needed to yield the final spatial image results. On the other hand, with spatial domain processing, the pixels in the image are manipulated directly.

12.1 DETERMINISTIC GRAY-LEVEL TRANSFORMATION

Processing in the spatial domain is usually carried out pixel by pixel. Depending on whether the processing is based only on the processed pixel or takes its neighboring pixels into consideration, processing can be further divided into two subcategories: point processing and neighborhood processing. With this definition, the neighborhood used in point processing can be interpreted as being 1×1. In neighborhood processing, 3×3 or 5×5 windows are frequently used for the processing of a single pixel, and it is quite obvious that the computational time required will be greatly increased. Nevertheless, such an increase in computational time is sometimes needed to obtain local context information, which is useful for decision making in specific pixel processing for certain purposes. For example, to smooth an image, we need to use the similarity property for a smooth region; to detect the boundary we need to detect the sharp gray-level change between adjacent pixels.

By *point processing* we mean that the processing of a certain pixel in the image depends on the information we have on that pixel itself, without consideration of the status of its neighborhood. There are several ways to treat this problem, one of which is deterministic gray-level transformation, and another, histogram modification. Deterministic gray-level transformation is quite straightforward. A conversion table or algebraic expression will be stored, and the gray-level transformation for each pixel will be carried out either by table lookup or by algebraic computation, as shown schematically in Figure 12.1, in which $g(x, y)$ is the image after gray-level transformation of the original image $f(x, y)$. For an image of 512×512 pixels, 262,144 operations will be required. Figure 12.2 shows some of the deterministic gray-level transformations that could be used to meet various requirements. With the gray-level transformation function as shown in part (a), straight-line function 1 yields a brighter output than for the original, whereas straight-line function 2 gives a lower gray-level output for each pixel of the original image. Part (b) shows brightness stretching on the midregion of the image gray levels, part (c) gives a more eccentric action on this transformation which is useful in contrast stretching, and part (d) is the limiting transformation function, which yields a binary image (i.e., only two gray levels would exist in the image). Part (e) gives an effect opposite to that shown in part (b). In part (f), function 1 shows a dark region stretching transformation (i.e., dark becomes less dark, bright becomes less bright), and function 2 gives a bright region stretching transformation (i.e., lower gray-level outputs for lower gray levels of $f(x, y)$, but higher gray-level outputs for higher gray levels of $f(x, y)$. The sawtooth contrast scaling gray-level transformation function shown in part (g) can be used to produce a wide-dynamic-range image on a small-dynamic-range display. This is achieved by removing the most significant bit of the pixel value. Part (h) shows a reverse scaling, by means of which a negative of the original image can be obtained. Part (i) shows a thresholding transformation, where the height h can be changed to adjust the output dynamic range. Part (j) shows a level slicing contrast enhancement, which permits isolation of a band of input gray levels. Figures 12.3 to 12.13 shows some of the results obtained in applying the typical contrast stretching transformations to enhance the images.

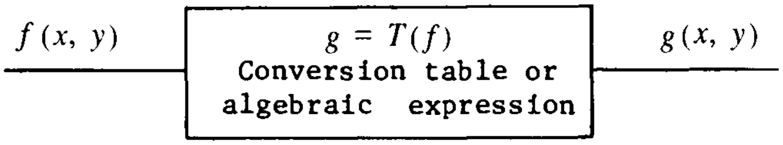


FIGURE 12.1 Schematic diagram of deterministic gray-level transformation.

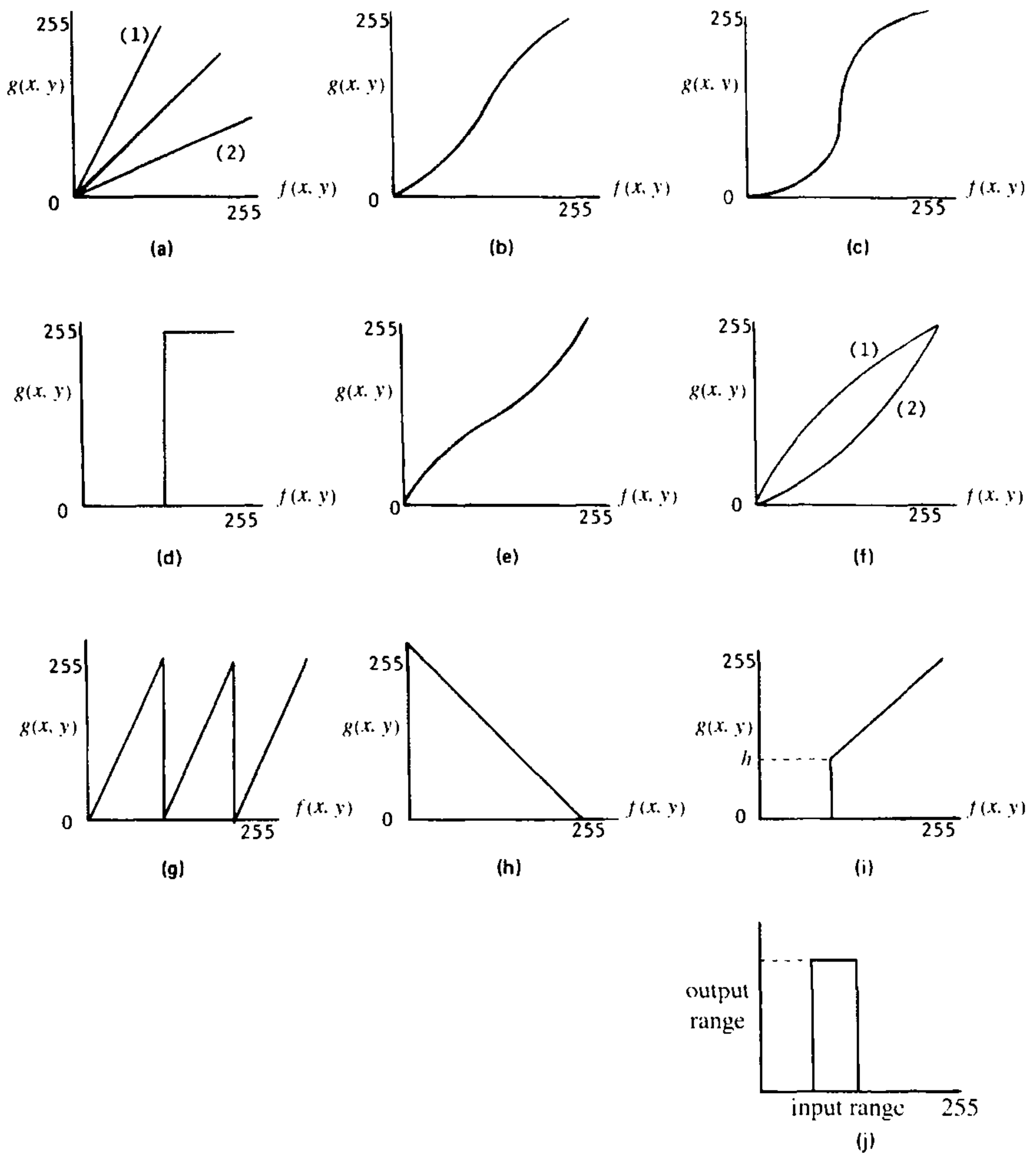


FIGURE 12.2 Various gray-level transformation functions.

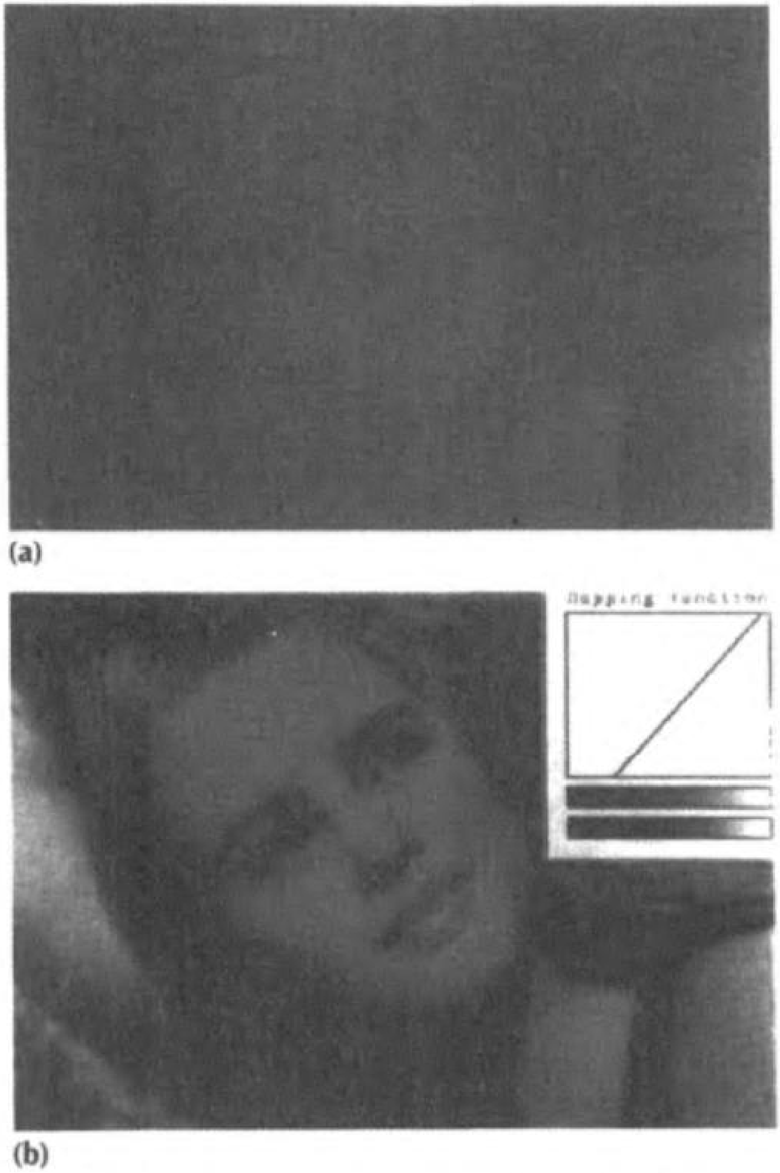


FIGURE 12.3 Linear contrast stretching. (a) Original image; (b) processed image.

12.2 GRAY-LEVEL HISTOGRAM MODIFICATION

12.2.1 Gray-Level Histogram

Consider an image $f(x, y)$ with discrete gray-level range $\{0, 1, 2, \ldots, 2^k - 1\}$, where k is a positive integer. The gray-level histogram $H(z)$ is the discrete graph plotted with the number of pixels at gray level z versus z, or

$$H(z) = \int\int [f(x, y) = z]\, dx\, dy \tag{12.1}$$

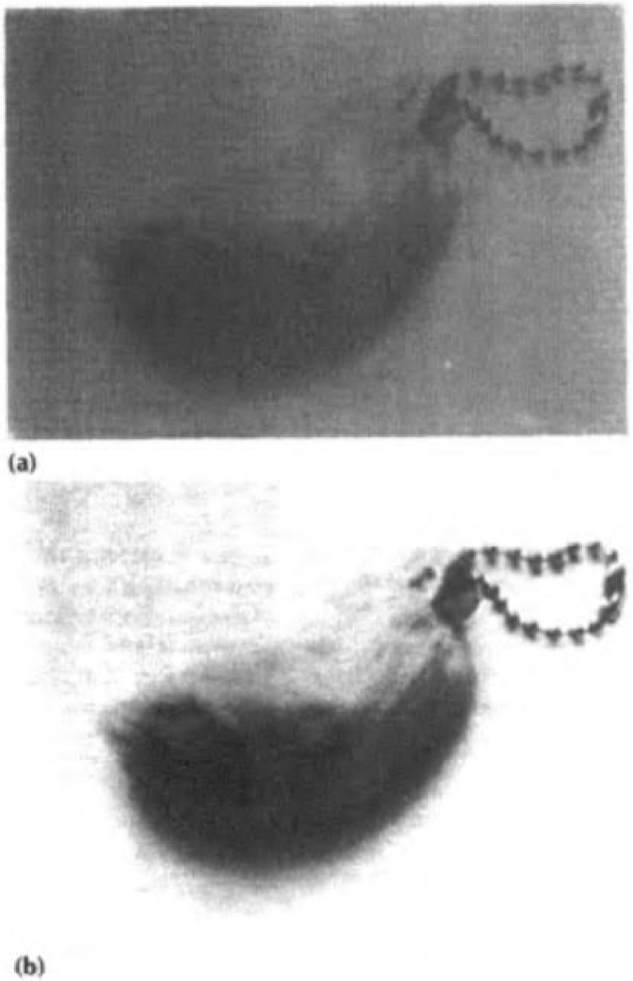


FIGURE 12.4 Linear contrast stretching. (a) Original image; (b) processed image.

It gives the distribution of the gray-level intensities over the image without reference to their locations, only to their frequencies of occurrence. So a histogram is a global representation of an image. For example, let $f(x, y)$ be the image

$$f(x, y) = \begin{vmatrix} 1 & 1 & 1 & 9 & 9 & 0 & 0 & 12 \\ 1 & 1 & 2 & 2 & 2 & 9 & 1 & 12 \\ 12 & 6 & 4 & 3 & 0 & 5 & 1 & 1 \\ 4 & 4 & 2 & 3 & 2 & 6 & 1 & 1 \\ 4 & 0 & 3 & 7 & 3 & 5 & 5 & 13 \\ 10 & 10 & 4 & 2 & 3 & 5 & 6 & 13 \\ 10 & 2 & 10 & 2 & 2 & 8 & 11 & 11 \\ 11 & 11 & 11 & 11 & 11 & 11 & 12 & 14 \end{vmatrix}$$

Then $f(x, y)$ has the histogram $H(z)$ shown in Figure 12.14.

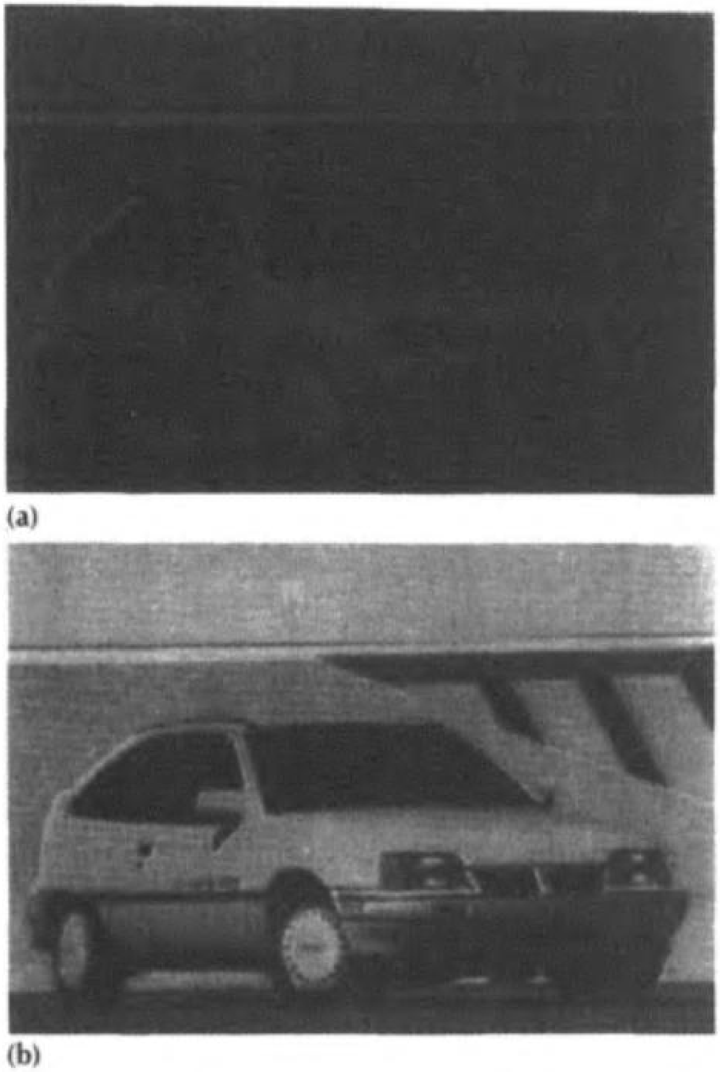


FIGURE 12.5 Linear contrast stretching. (a) Original image; (b) processed image.

Note that the $H(z)$ is a unary operator. The input is an image, while the output is an array:

$$[H(0) \quad H(1) \quad H(2) \quad \cdots \quad H(n)]$$

where $n = 2^k - 1$ and k is a positive integer. From the histogram function $H(z)$, the area function $A(z)$ can be computed. This is the area of the picture with gray level above threshold z.

$$A(z) = \int_z^{\infty} H(z)\,dz \tag{12.2}$$

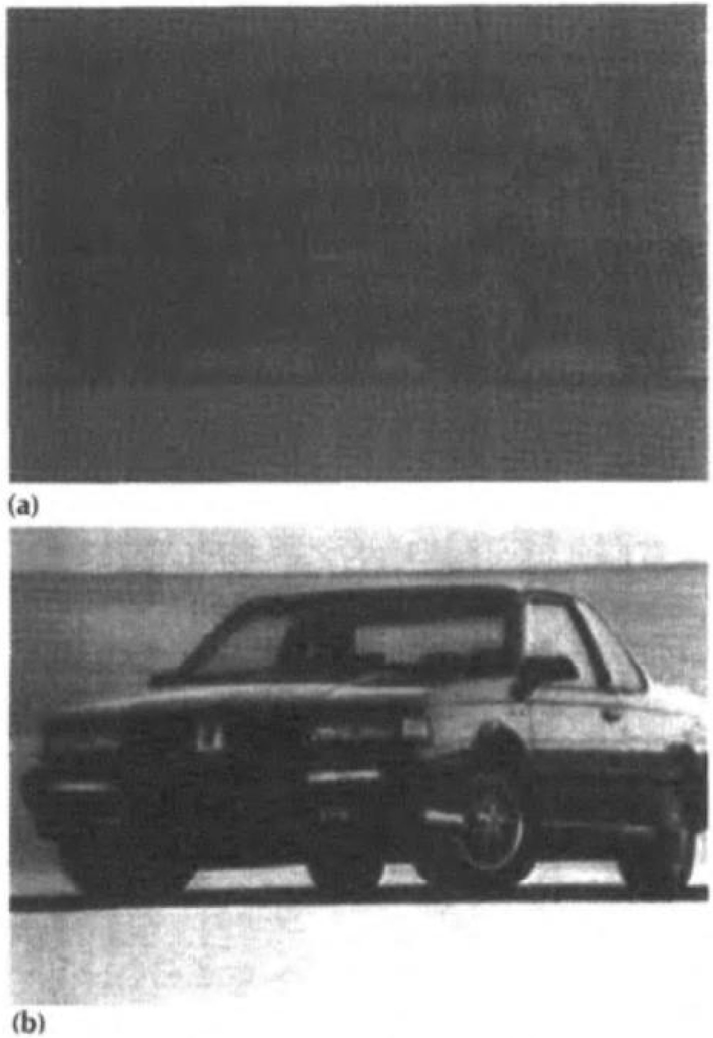


FIGURE 12.6 Linear contrast stretching. (a) Original image; (b) processed image.

Differentiating Eq. (12.2) gives

$$H(z) = \frac{dA(z)}{dz} \tag{12.3}$$

This can be interpreted using Figure 12.15, where A_1 is the area that contains all the pixels with gray level greater than z_1, or $A_1 = A(z_1)$. A_2 is the area containing all the pixels with gray level greater than z_2, or $A_2 = A(z_2)$, and z_2 is greater than z_1, say, $z_2 = z_1 + \Delta z$. We can then write

$$\lim_{\Delta z \to 0} \frac{A(z) - A(z + \Delta z)}{\Delta z} = \frac{d}{dz} A(z) = H(z) \tag{12.4}$$

which is the mathematical definition of the histogram (Castleman, 1979).

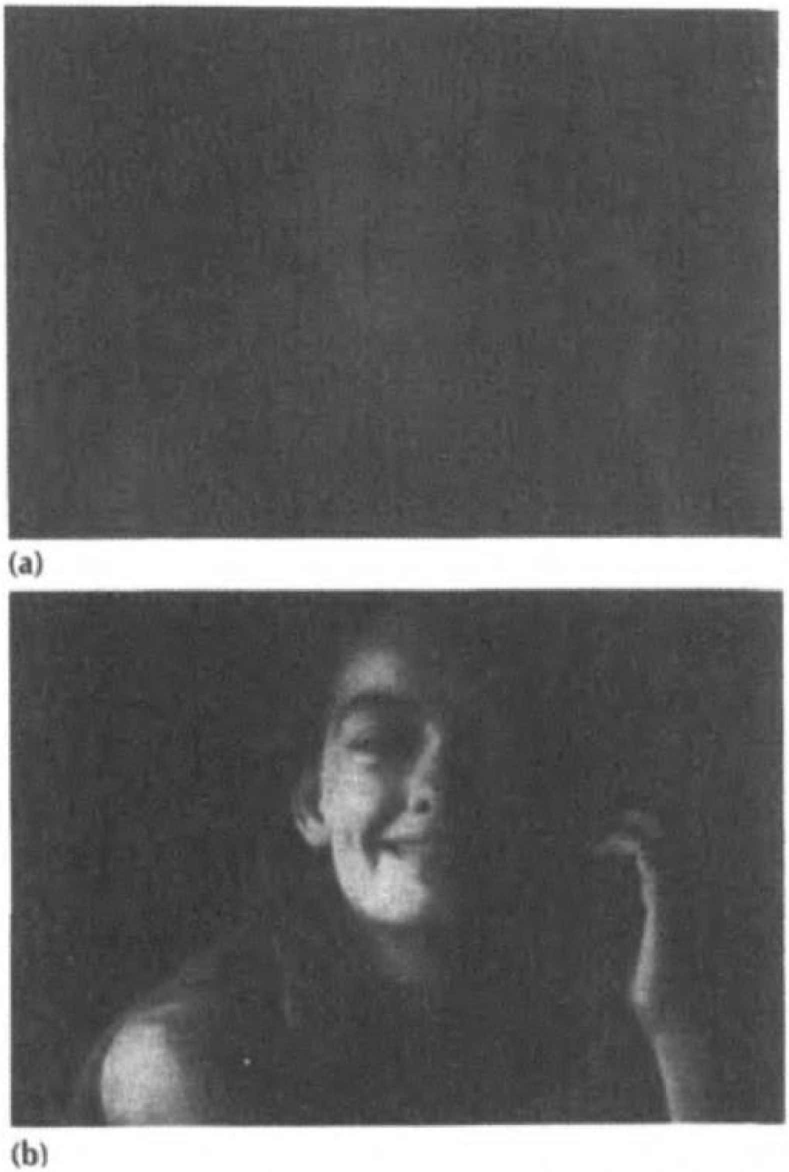


FIGURE 12.7 Linear contrast stretching. (a) Original image; (b) processed image.

12.2.2 Histogram Modification

As mentioned in Section 12.2.1, the histogram of an image represents the relative frequency of occurrence of the various gray levels in the image, or the probability density $P_r(r)$ versus r. Histogram modification techniques modify an image so that its histogram has a desired shape. This technique can be usec to improve the image contrast, and is another effective method used in point processing.

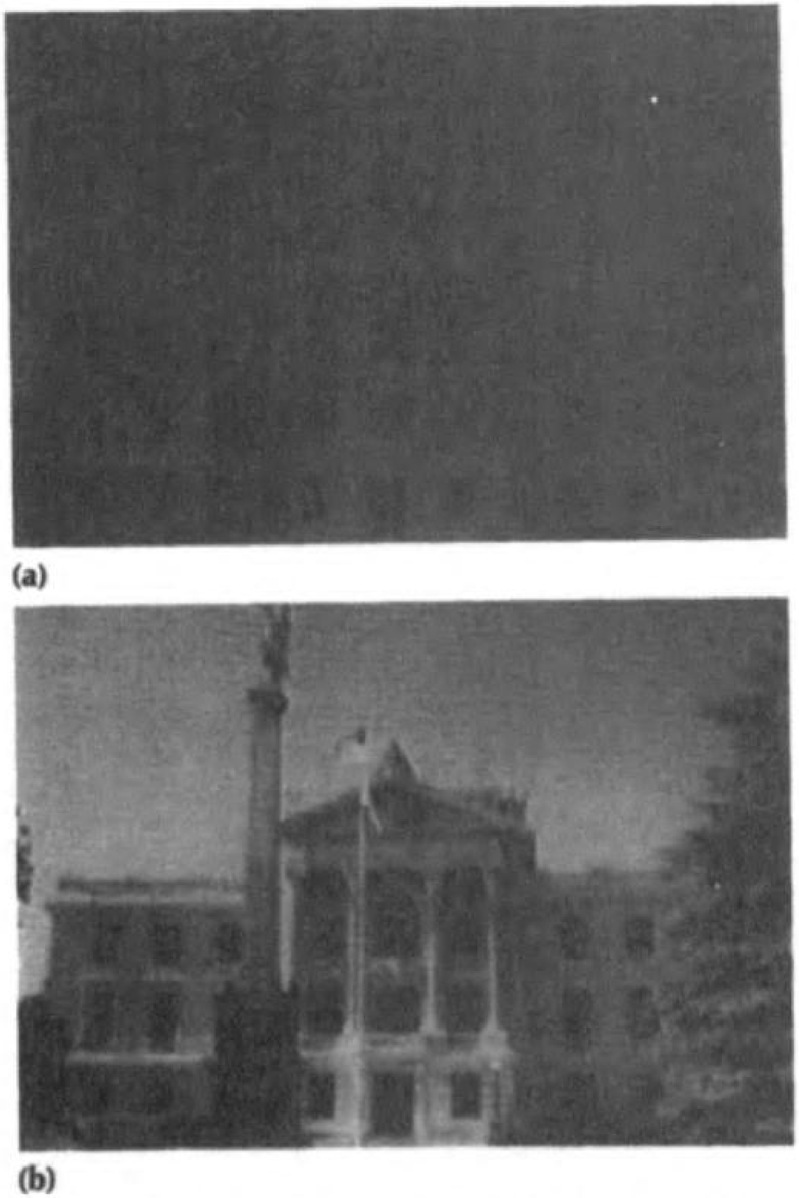


FIGURE 12.8 Lιnear contrast stretching. (a) Originai image; (b) processed ımage.

Figure 12.16a shows a histogram plot and Figure 12.16b shows the cumulative density function versus r plot. With the histogram a distribution of gray levels of pixels in an image can be described.

For an image of $N \times N$ pixels, the total number of pixels in the image is $\sum_{i=r_1}^{r_k} n_i = N^2$, where $r_1, \ldots, r_k$ are the gray levels and n_i is the number of pixels at gray level r_i. The histogram and the cumulative probability density function (CPDF) will be of the form shown in Figure 12.17a and b, respectively.

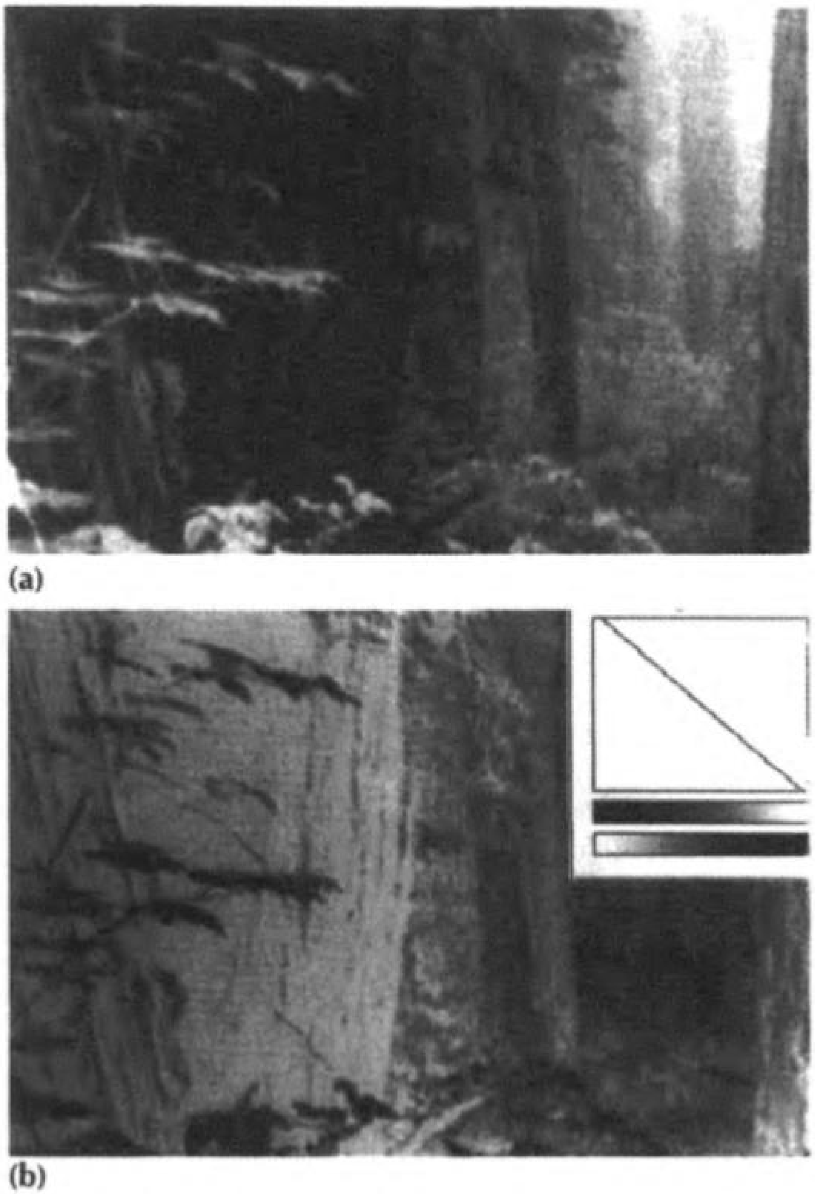


FIGURE 12.9 Reverse scaling. (a) Original image; (b) processed image.

It is obvious that $\sum_{r_1}^{r_k}(\text{PDF}) = 1$, and CPDF is a single-valued monotonic function. If the input image intensity variable $r = f(x, y), r_0 \leq r \leq r_J$, for the original image is mapped into the output image intensity $s = g(x, y), s_0 \leq s \leq s_k$, for the processed image such that the output probability distribution $P_s(s_k)$ follows some desired form for a given input probability distribution $P_r(r_j)$, we can relate them by Eqs. (12.5) and (12.6):

$$s = T(r) \qquad r_0 \leq r \leq r_J \tag{12.5}$$

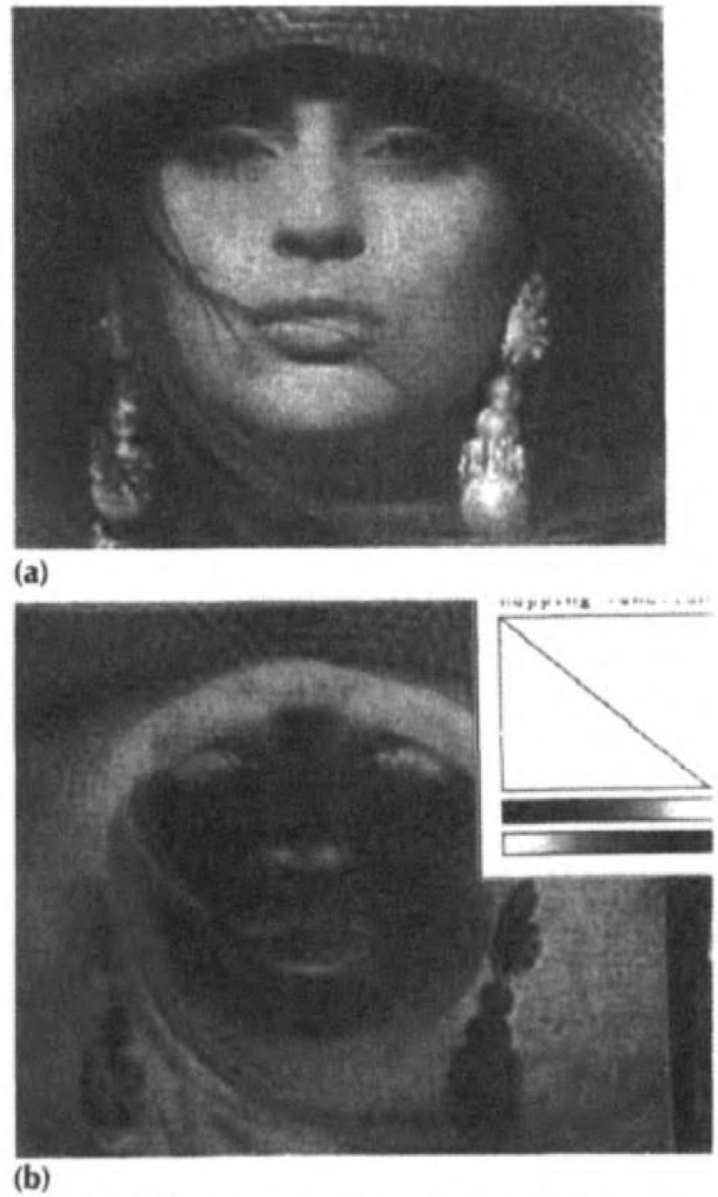


FIGURE 12.10 Reverse scaling. (a) Original image; (b) processed image.

or

$$r = T^{-1}(s) \qquad s_0 \leq s \leq s_K \tag{12.6}$$

where T is a transformation operator. This transformation function can be expressed in the form of a table, a functional curve or a mathematical expression. This transformation function must satisfy the following two conditions:

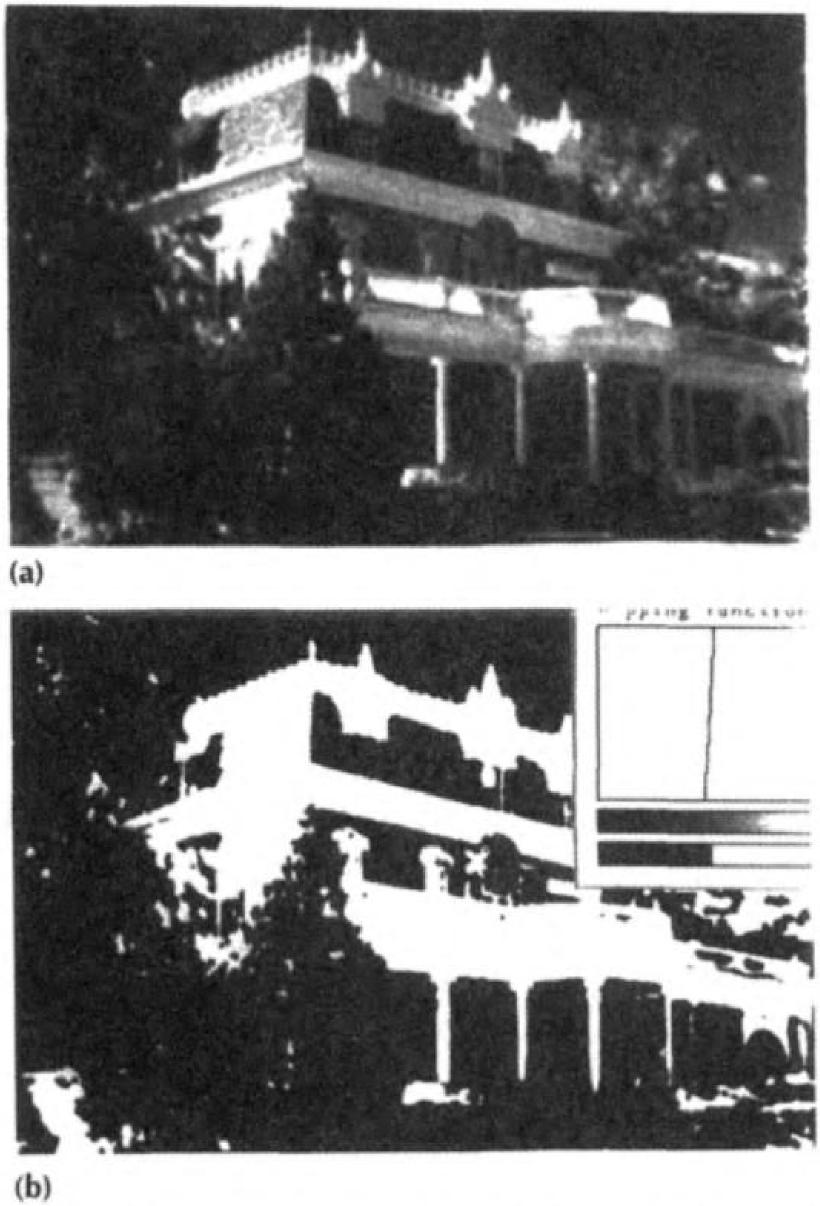
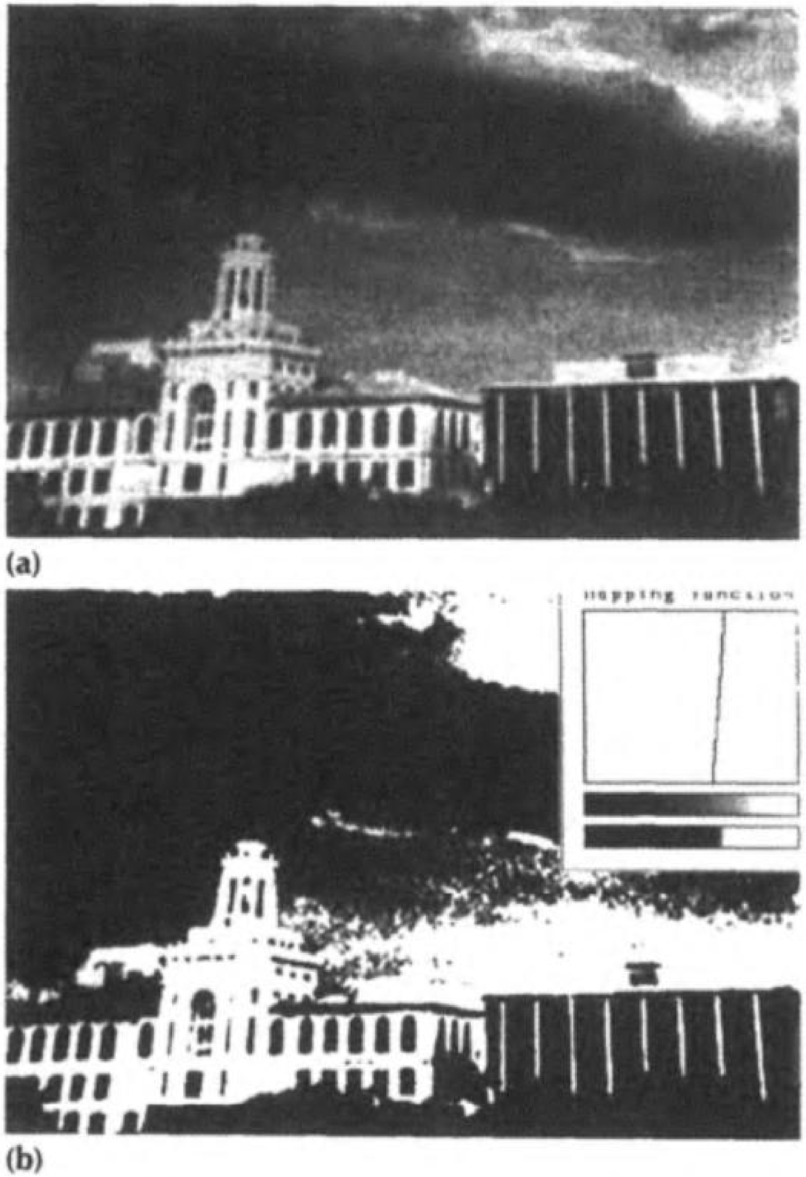


FIGURE 12.11 Binary image transformation. (a) Original image; (b) processed image.

1. It must be a single-value function and monotonically increasing to avoid ambiguous situation in the interval $0 \le r \le 1$, when normalized.
2. s, which is $T(r)$, should also be within the range between 0 and 1 for $0 \le r \le 1$.

However, the number of gray levels used in s is not necessarily equal to that used in r. This histogram modification problem can then be formulated as follows:



(a)

(b)

FIGURE 12.12 Binary image transformation. (a) Original image; (b) processed image.

1. Find the transformation function $T(r)$ to relate the PDF of the image on the original gray-level scale and the desired probability density distribution of the image on a new gray-level scale.
2. Or find the probability density function PDF of the image on a new gray-level scale, when the PDF of the image on the original gray-level scale and the transformation function are given.

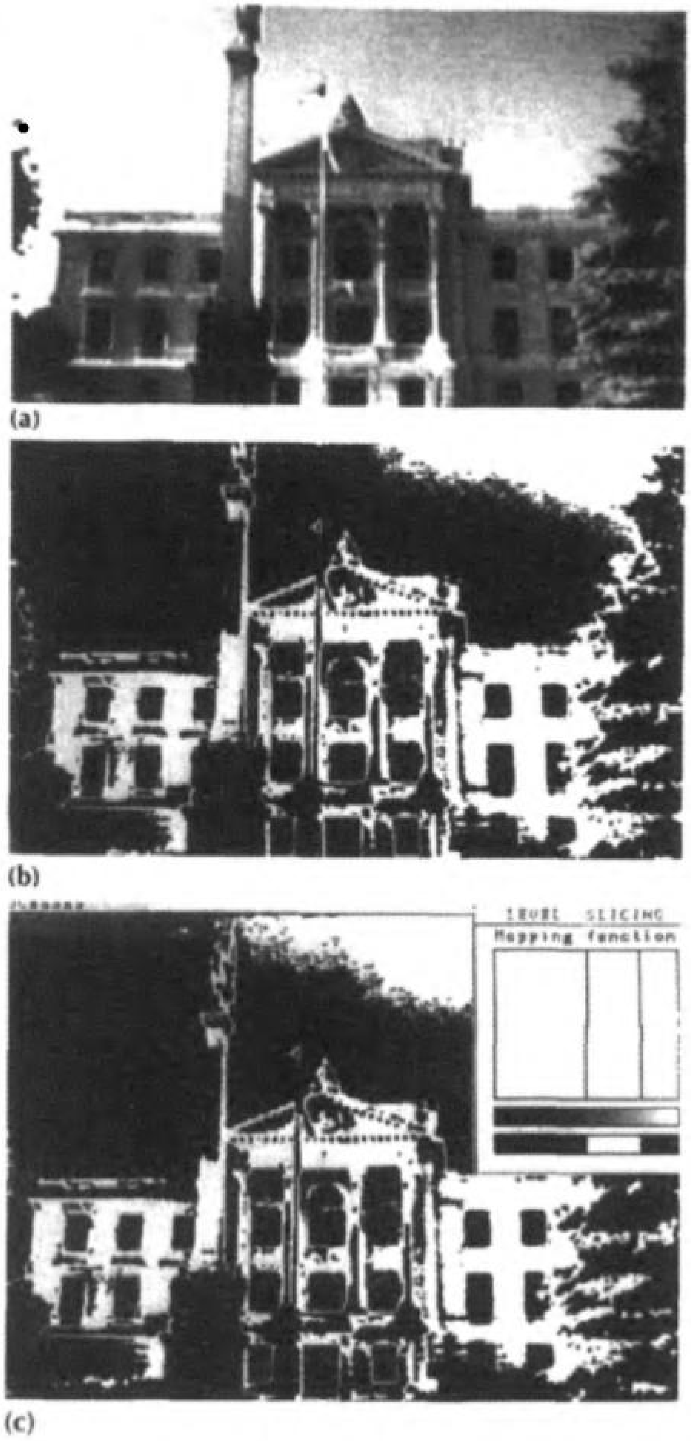


FIGURE 12.13 Level slicing contrast enhancement. (a) Original image; (b) processed image with mapping function shown on (c); (c) mapping function.

$$g(x, y) = \begin{cases} L & a \leq f(x, y) \leq b \\ 0 & \text{otherwise} \end{cases}$$

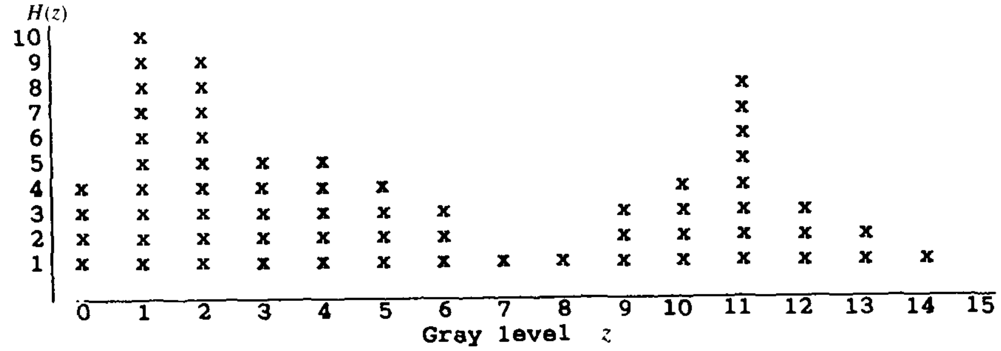


FIGURE 12.14 Histogram operation.

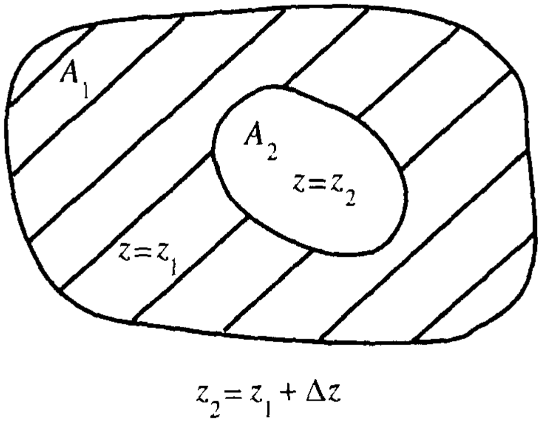


FIGURE 12.15 Image regions with different gray levels.

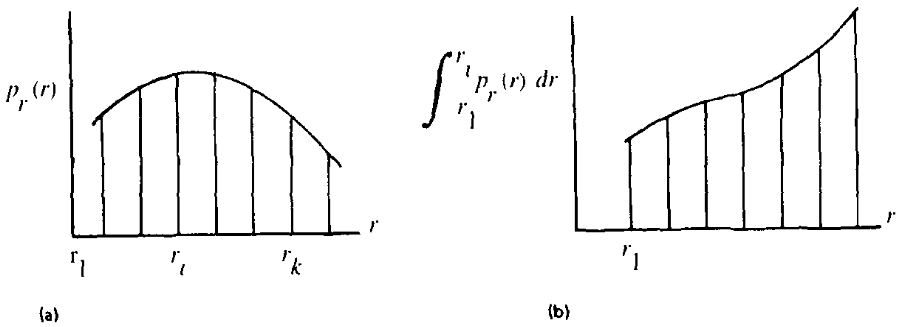


FIGURE 12.16 Probability density function and cumulative density function versus *r* plot. (a) $p_r(r)$ versus *r* plot (histogram); (b) cumulative probability density function versus *r* plot.

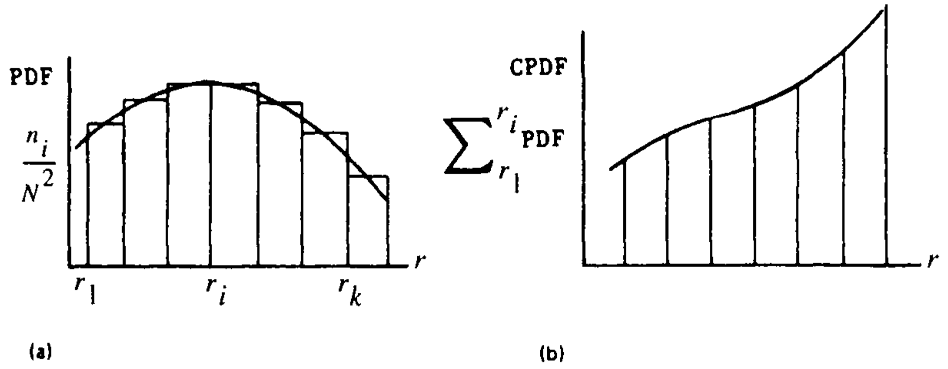


FIGURE 12.17 Probability density function and cumulative probability density function versus r plot. (a) Histogram; (b) cumulative probability density function versus r plot.

Note that this transformation is a gray-level transformation only. The number of gray levels may be different before and after the transformation, but there is no loss in pixel numbers. Hence

$$\sum_{j=0}^{J} p_r(r_j) = 1 \tag{12.7}$$

$$\sum_{k=0}^{K} p_s(s_k) = 1 \tag{12.8}$$

Equations (12.7) and (12.8) state that the input and output probability distributions must both sum to unity. Furthermore, the cumulative distributions for the input and output must equate for any input index j, that is,

$$\sum_{k} p_s(s_k) = \sum_{j} p_r(r_j) \tag{12.9}$$

Thus the probability that pixels in the input image have a pixel luminance value $\le r_j$ must be equal to the probability that pixels in the output image have a pixel luminance value $\le s_k$, as long as the transformation rule $s_k = T(r_j)$ is followed. The transformation in this case is monotonic. If for a given image the cumulative probability distribution $\sum_j p_r(r_j)$ is replaced by the cumulative histogram $\sum_j H_r(j)$, a solution for s_k in terms of r_j can be obtained by an inverted form of Eq. (12.9), as shown in Figure 12.18.

Although it is not impossible, it is very difficult to solve this problem analytically except for the very simple uniform output histogram case. Let us replace the summation by integration; we then have

$$\int_{s_{\min}}^{s} p_s(s)\,ds = \int_{r_{\min}}^{r} p_r(r)\,dr \tag{12.10}$$

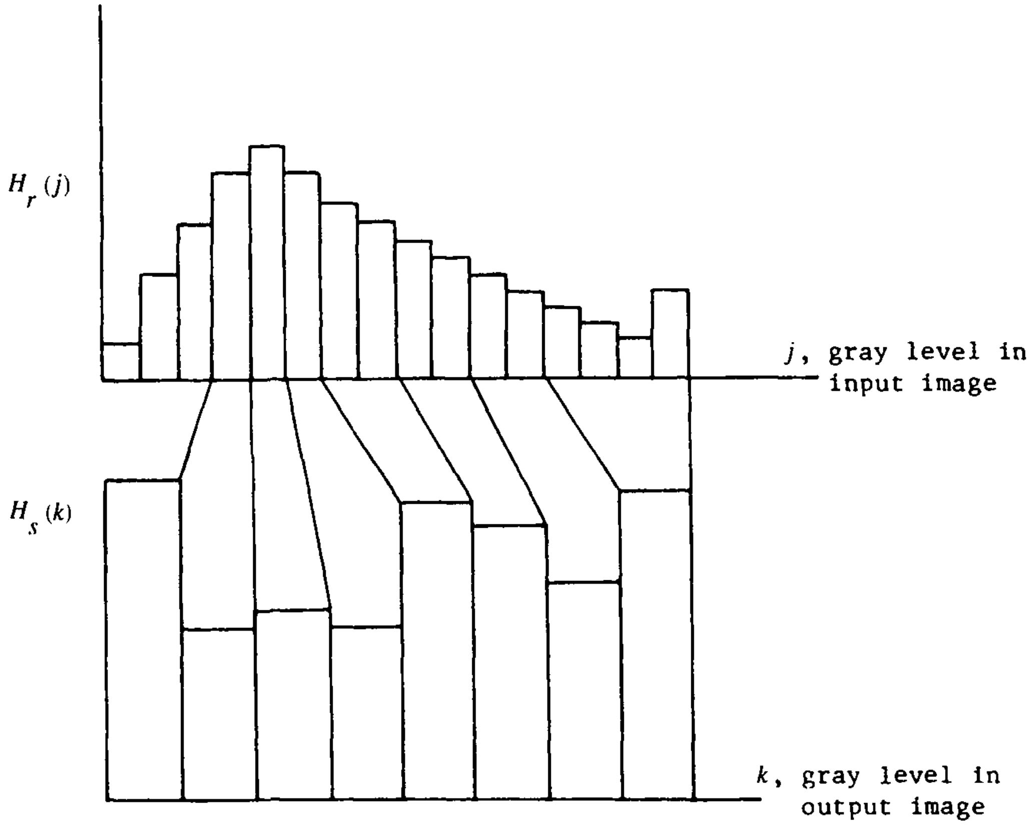

$H_r(j)$
$H_s(k)$
j, gray level in input image
k, gray level in output image

FIGURE 12.18 Histogram modification.

where the definite integral of $p_r(r)$ over r on the right-hand side is the cumulative distribution of the input image and equal, say, to $P_r(r)$.

For a uniform histogram,

$$p_s(s) = \text{const.} = \frac{1}{s_{\max} - s_{\min}} \qquad \text{for } s_{\min} \leq s \leq s_{\max} \tag{12.11}$$

Substitution of Eq. (12.11) in (12.10) gives

$$\int_{s_{\min}}^{s} \frac{1}{s_{\max} - s_{\min}} ds = \int_{r_{\min}}^{r} p_r(r)\, dr \tag{12.12}$$

or

$$\frac{s - s_{\min}}{s_{\max} - s_{\min}} = P_r(r) \tag{12.13}$$

from which we have

$$s = [s_{\max} - s_{\min}]P_r(r) + s_{\min} \tag{12.14}$$

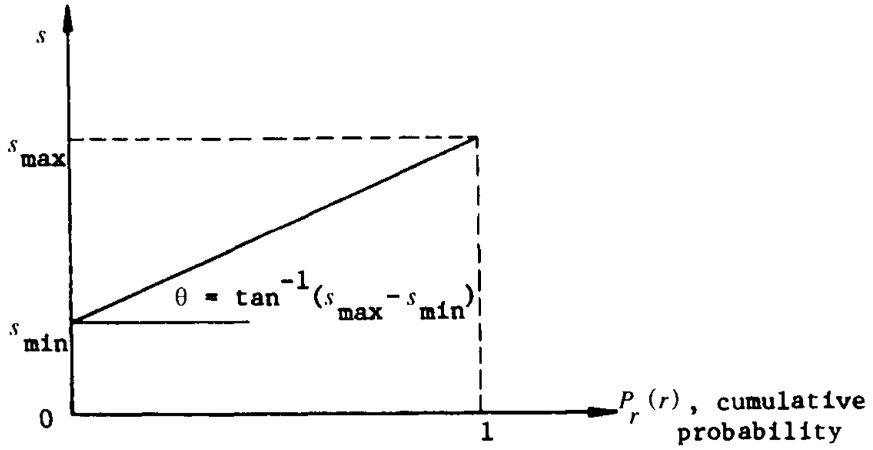

s
s max
s min
0
θ = tan^{-1}(s max - s min)
1
P r (r) , cumulative probability

FIGURE 12.19 Transfer function derived for histogram equalization.

which is the histogram equalization transfer function and is shown in Figure 12.19. $P_r(r)$, the cumulative probability distribution of the input image, can be approximated by its cumulative histogram or $P_r(r) \simeq \sum_j H_r(j)$. For an exponential output histogram,

$$p_s(s) = \alpha e^{-\alpha[s-s_{\min}]} \qquad s \geq s_{\min} \tag{12.15}$$

By substituting Eq. (12.15) in Eq. (12.10), we obtain

$$\int_{s_{\min}}^{s} \alpha e^{-\alpha[s-s_{\min}]}\, ds = \int_{r_{\min}}^{r} p_r(r)\, dr \tag{12.16}$$

or

$$1 - e^{-\alpha[s-s_{\min}]} = P_r(r) \tag{12.17}$$

The transfer function will then be

$$s = s_{\min} - \frac{1}{\alpha}\ln[1 - P_r(r)] \tag{12.18}$$

The procedure for conducting the histogram equalization can be summarized as consisting of the following steps:

1. Compute the average number of pixels per gray level.
2. Starting from the lowest gray-level band, accumulate the number of pixels until the sum is closest to the average. All of these pixels are then rescaled to the new reconstruction levels.
3. If an old gray-level band is to be divided into several new bands, either do it randomly or adopt a rational strategy—one being to distribute them by region. An example of a 16-gray-level 128 × 128 pixel image

TABLE 12.1 Example Image Showing Histogram Equalization Mapping

k	r_k	n_k	$p_r(r_k) = n_k / \sum n_k$	CPDF= $\sum P_r(r_k)$
0	0	300	0.0183	0.0183
1	1/15	1500	0.0916	0.1099
2	2/15	3500	0.2136	0.3235
3	3/15	3000	0.1831	0.5066
4	4/15	2125	0.1297	0.6363
5	5/15	1625	0.0992	0.7355
6	6/15	1250	0.0763	0.8118
7	7/15	900	0.0549	0.8667
8	8/15	650	0.0397	0.9064
9	9/15	550	0.0336	0.9400
10	10/15	325	0.0198	0.9598
11	11/15	200	0.0122	0.9720
12	12/15	150	0.0092	0.9812
13	13/15	140	0.0085	0.9897
14	14/15	97	0.0059	0.9956
15	1	72	0.0044	1.0000
		$\sum n_k = 16{,}384$	1.0000	

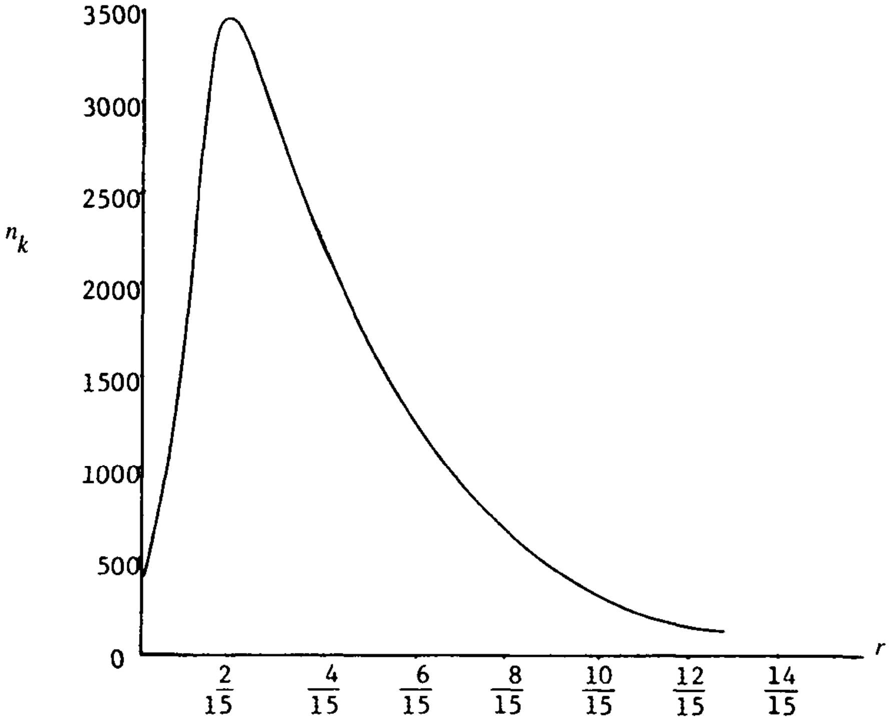


FIGURE 12.20 Original histogram of the example image.

	New gray levels																Summation of pixels
	0	1/15	2/15	3/15	4/15	5/15	6/15	7/15	8/15	9/15	10/15	11/15	12/15	13/15	14/15	15/15	
0	300																300
1/15	724	776															1500
2/15		248	1024	1024	1024	180											3500
3/15						844	1024	1024	108								3000
4/15									916	1024	185						2125
5/15											839	786					1625
6/15												238	1012				1250
7/15													12	888			900
8/15														136	514		650
9/15															510	40	550
10/15																325	325
11/15																200	200
12/15																150	150
13/15																140	140
14/15																97	97
1																72	72
	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	16384

FIGURE 12.21 Gray-level transformation matrix for the example image of Figure 12.20.

> is shown in Table 12.1. Sixteen equally spaced gray levels are assumed in this example. The average number of pixels per gray level is 16,384/16 = 1024. The histogram of this image is shown in Figure 12.20, and the gray-level transformation matrix can be formulated as shown in Fig. 12.21.

Figures 12.22a, 12.23a, and 12.24a show pictures that are barely visible due to the narrow range of values occupied by the pixels of this image, as shown by the histograms in Figures 12.22b, 12.23b, and 12.24b. After histogram equalization, considerable improvements were achieved. See Figures 12.22c and d, 12.23c and d, and 12.24c and d for the enhanced images and their equalized histograms.

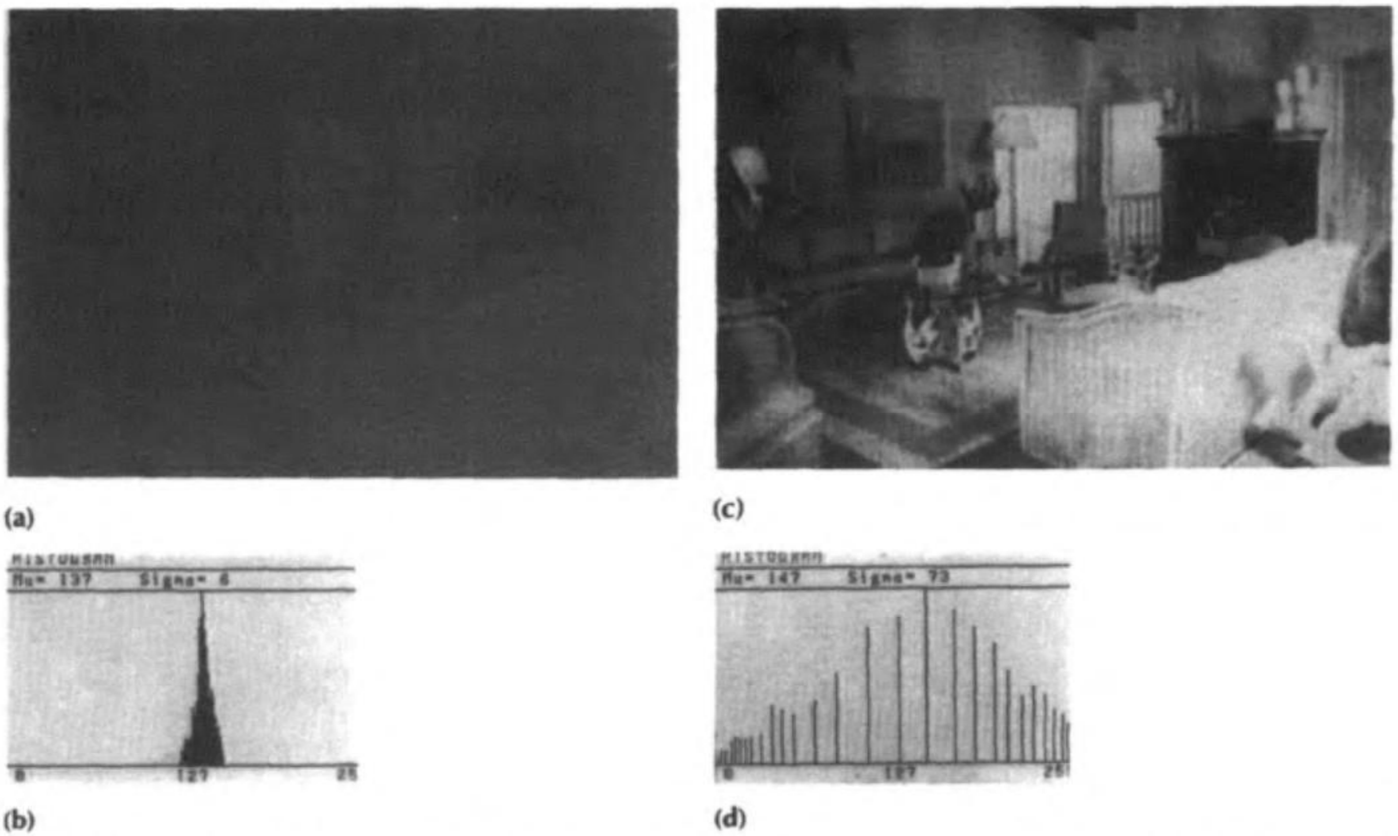
Although histogram equalization is a very useful tool, especially for the enhancement of a low-contrast image, this approach is limited to the generation of only one result (i.e., an approximation of a uniform histogram). In many cases it is desired to have a transformation such that probability density function of the output image matches that of a prespecified probability density function. An examplary application is to specify interactively particular histograms capable of highlighting certain gray-level ranges in an image. See Gonzalez and Wintz (1987, p. 157) for a set of images that shows an original semidark room viewed from a doorway, the image after the histogram equalization process, and the image after interactive histogram specification. The result obtained using histogram specification has the much more balanced appearance that we seek. In the image processed by histogram equalization, the contrast was somewhat high.

It is therefore our desire to develop a gray-level transformation such that the histogram of the output image matches the one specified. In other words, given two histograms, how can we find a gray-level transformation T so that one histogram can be converted into another? The procedure to achieve this is a modified version of that used for histogram equalization:

1. Find a transformation T_1 that will transform h_1 into a uniform histogram. This can be done by performing histogram equalization on the input image.
2. Find a transformation T_2 that will transform the output image with prespecified histogram h_2 to yield a uniform histogram. This yields a second transformation $s = T_2(r)$.
3. Obtain the inverse transformation described in step 2.
4. Combine the transformation T_1 and the inverse transformation T_2^{-1} to give the composite transform, which is

$$s = T_2^{-1}(T_1(r)) \tag{12.19}$$

Figure 12.25 shows the composite process for the histogram specification. Another application of histogram specification worthy of mention is for compar-


(a)

(b)

(c)

(d)

FIGURE 12.22 Considerable improvement in the image achieved through histogram equalization. (a) Original image; (b) original histogram; (c) processed image; (d) histogram after equalization.

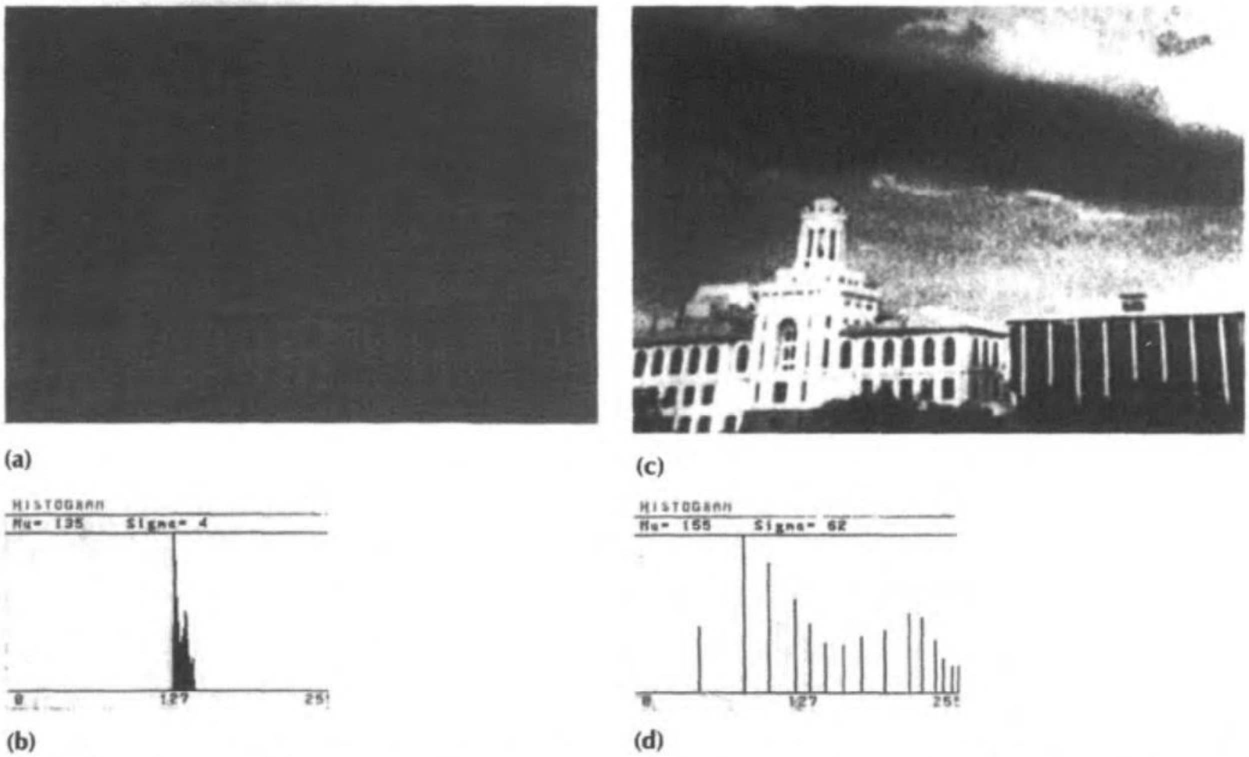


FIGURE 12.23 Considerable improvement in the image achieved through histogram equalization. (a) Original image; (b) original histogram; (c) processed image; (d) histogram after equalization.

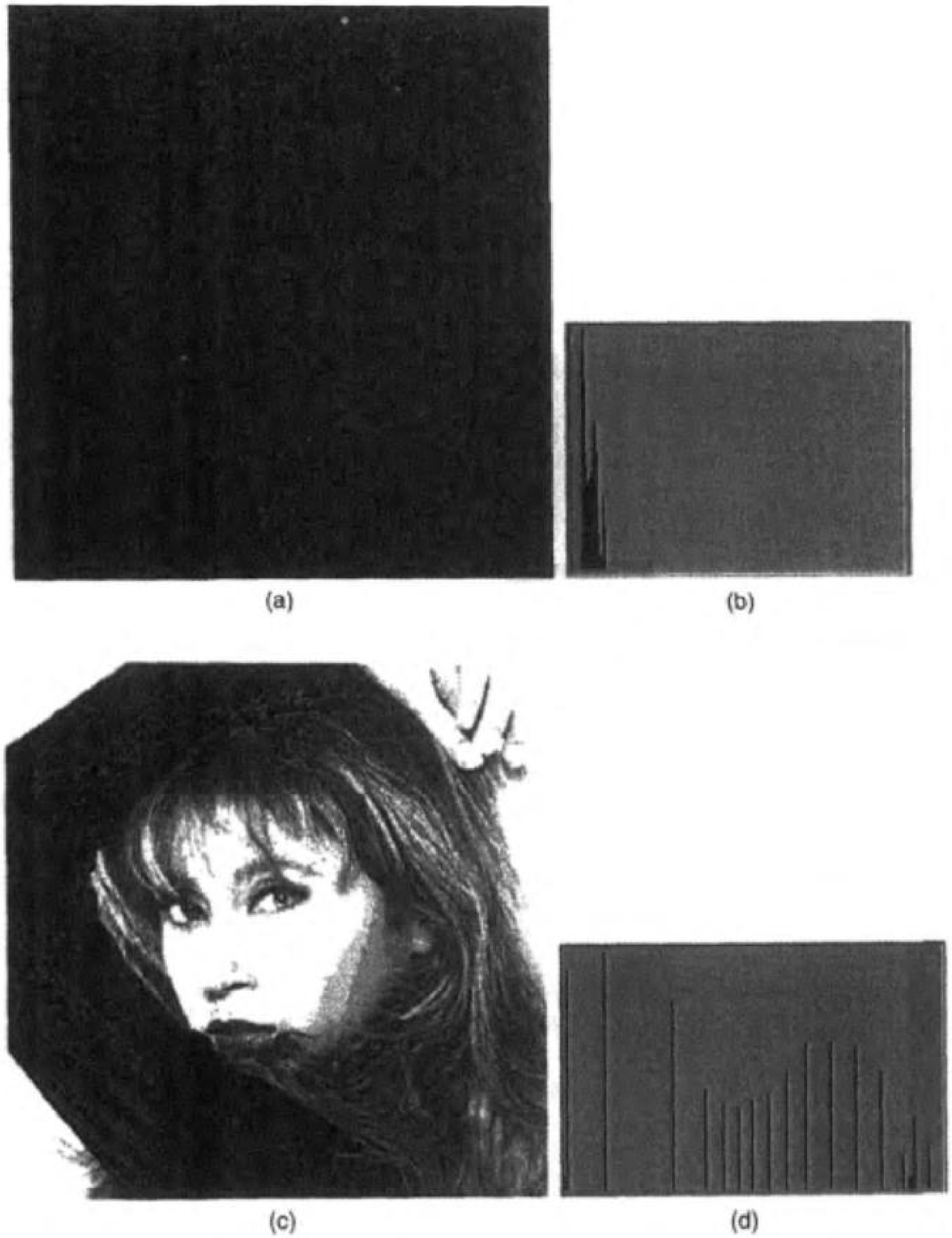


FIGURE 12.24 Considerable improvement in the image achieved through histogram equalization. (a) Original image; (b) original histogram; (c) processed image; (d) histogram after equalization.

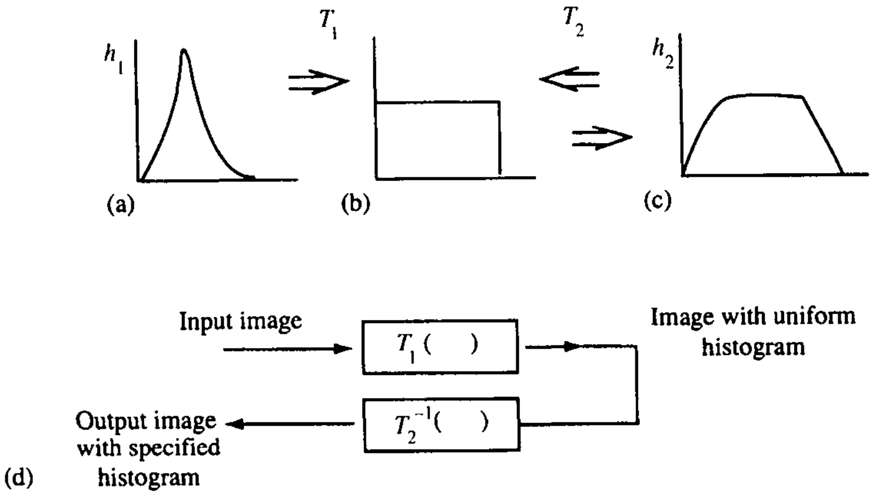


FIGURE 12.25 Composite process for histogram specification. (a) Histogram of the input image; (b) uniform histogram after transformation T_2; (c) desired histogram of the output image; (d) composite process for the histogram specification.

ison of the two images of a scene acquired under different illumination conditions.

12.2.3 HistogramThinning

So far we have discussed the use of histogram modification to enhance an image. Histogram modification can also be used to help segment objects for detection and/or identification. This is known as histogram thinning. Whereas in histogram equalization the histogram is flattened to achieve full dynamic range of the gray levels to get more details of the image, the goal of histogram thinning is to obtain the opposite effect on the image. The approach is to transform the input image into one with fewer number of gray levels without loss of detail.

The idea we use in this process is to thin each peak on the original histogram into a spike so that the gray levels belonging to each peak are now represented by a single gray level. The image is thus segmented into regions that correspond to the gray-level ranges of the original peaks. This histogram thinning process helps segment the image into isolated objects, and is useful in the image understanding process to differentiate and/or identify various objects (e.g., pond, grass, forest, highway) in remote sensing satellite and aerial images.

Figure 12.26 shows a histogram with two humps, each of which represents an object. In the histogram thinning process, it is desired to thin the original histogram (a) into a new histogram (b), so that these two objects can be separated more easily. The algorithm suggested is:

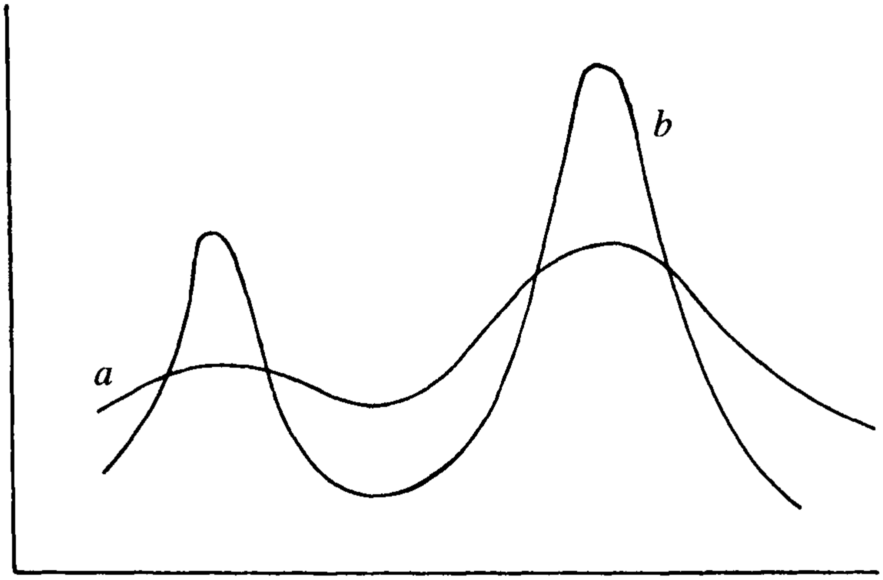


FIGURE 12.26 Histogram thinning. (a) Original histogram; (b) thinned histogram.

1. Calculate the initial histogram.
2. Starting from the lowest value of the histogram, search for the local peaks.
3. When a local peak is located, move a certain fraction of pixels toward the peak to "thin" the peak.
4. After the peak has been thinned, find the second peak and do the same, until no more local peaks exist.

The entire process is summarized in the flow diagram shown in Figure 12.27. A local peak of the histogram is said to appear at gray level i, if the number of pixels at i (say B_i) is greater than the average of the numbers of pixels at gray levels $i + r, r = 1, 2, \ldots, r$ (say A^+), and also greater than the average of the numbers of pixels at gray levels $i - r, r = 1, 2, \ldots, r$ (say A^-). That is, the local peak appears at the gray level i when B_i is greater than both A^+ and A^-, where

$$A^+ = \frac{1}{r}\sum_{n=1}^{r} B_{i+n}$$

and

$$A^- = \frac{1}{r}\sum_{n=1}^{r} B_{i-n}$$

Otherwise, move forward to look for a prospective local peak. When a local peak is found, a certain fraction of pixels is to be moved toward the local peak step by

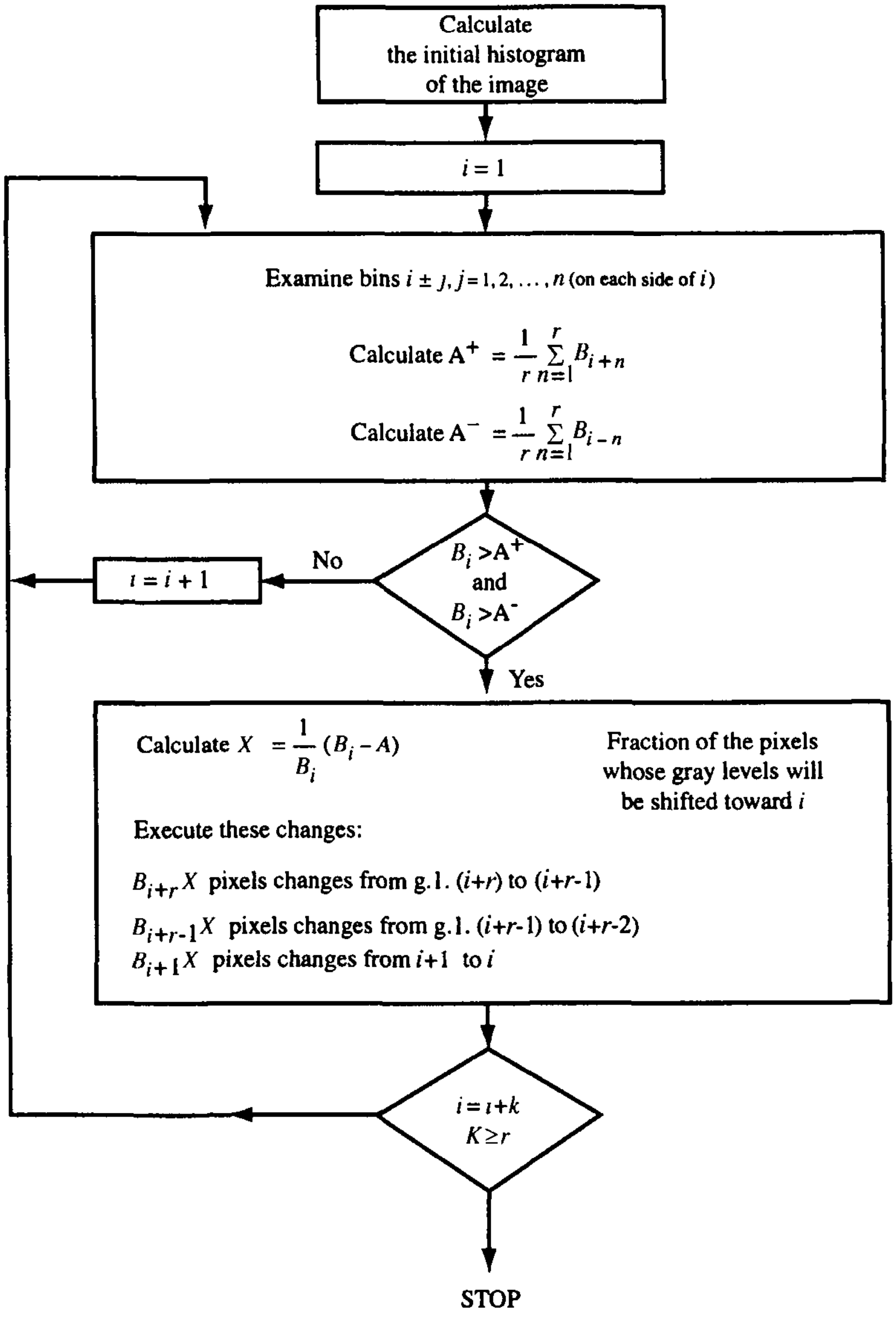

Calculate the initial histogram of the image
$i = 1$
Examine bins $i \pm j, j = 1, 2, \ldots, n$ (on each side of i)
Calculate $A^{+} = \frac{1}{r} \sum_{n=1}^{r} B_{i+n}$
Calculate $A^{-} = \frac{1}{r} \sum_{n=1}^{r} B_{i-n}$
No
$i = i + 1$
$B_i > A^{+}$ and $B_i > A^{-}$
Yes
Calculate $X = \frac{1}{B_i}(B_i - A)$
Fraction of the pixels whose gray levels will be shifted toward i
Execute these changes:
$B_{i+r}X$ pixels changes from g.l. $(i+r)$ to $(i+r-1)$
$B_{i+r-1}X$ pixels changes from g.l. $(i+r-1)$ to $(i+r-2)$
$B_{i+1}X$ pixels changes from $i+1$ to i
$i = i+k$
$K \geq r$
STOP

FIGURE 12.27 Flow diagram of the histogram thinning process. B_i = number of pixels in the ith histogram; A^{+} = average number of pixels at high end of the histogram; A^{-} = average number of pixels at low end of the histogram.

step to perform the histogram thinning process. This fraction (say X) is computed according to

$$X = \frac{B_i - A}{B_i}$$

where $A = \frac{1}{2}(A^+ + A^-)$.

12.3 SMOOTHING AND NOISE ELIMINATION

Neighborhood processing differs from point processing in that the local context information will be used for specific pixel processing. Pixels close to each other are supposed to have approximately the same gray levels except for those at the boundary. Noise can come from various sources. It can be introduced during transmission through the channel. This type of noise has no relation to the image signal. Its value is generally independent of the strength of the image signal. It is additive in nature and can be put in the following form:

$$f(x,y) = f'(x,y) + n(x,y) \tag{12.20}$$

where $f'(x,y)$ simply denotes the hypothetically noise-free image, n the noise, and $f(x,y)$ the noisy image. This kind of noise can be estimated by a statistical analysis of a region known to contain a constant gray level, and can be minimized by classical statistical filtering or by spatial ad hoc processing technique.

Another source of noise is the equipment or the recording medium. This kind of noise is multiplicative in nature, and its level depends on the level of the image signal. An example of this is noise from the digitizer, such as the flying spot scanner, TV raster lines, and photographic grain noise.

In practice, there frequently is a difference between an image and its quantized image. This difference can be classified as quantization noise and can be estimated.

In the smoothing of an image, we have to determine the nature of the noise points first. If they belong to fine noise, they are usually isolated points. This means that each noise point has nonnoise neighbors. These noise points, known as "salt-and-pepper noise," usually occur on raster lines.

After determining the nature of the noise, we can set up an approximate approach to eliminate the noise. Detection of the noise point can be done by comparing its gray level with those of its neighbors. If its gray level is substantially larger than or smaller than those of all or nearly all of its neighbors, the point can be classified as a noise point, and the gray level of this noise point can be replaced by the weighted average of the gray levels of its neighbors.

Coarse noise (e.g., a 2×2 pixel core) is rather difficult to deal with. The difficulty will be in detection. We treat such noise by various appropriate methods, and sometimes we need some a priori information.

Various sizes of neighborhood can be used for noise detection and for the smoothing process. A 3 × 3 pixel window is satisfactory in most cases, although a large neighborhood is sometimes used for statistical smoothing. For the border points (i.e., for the points on four sides of an image), some amendment should be added either by ignoring or by duplicating the side rows and columns.

Multiples of the estimated standard deviation of the noise can be chosen for the threshold by which the noisy point must differ from its neighborhood. The estimation of the standard deviation of the noise can be obtained by measuring the standard deviation of gray levels over a region that is constant in the nonnoisy image. It is assumed here that the noise has zero mean. Some other method, such as a majority count of the neighbors that have larger or smaller gray levels than the given point, is also adopted.

Based on the arguments we have presented so far, a decision rule can be set for the determination of the gray level of each point. Generally speaking, if a point is a noise point, its gray level can be replaced by the weighted average of its neighbors. If a point is not a noise point, its original gray level is still used.

There are some other possible ways of deciding whether the given point is a noise point:

1. Compare the gray level of the point with those of its neighbors. If the comparison shows that $|f - f_i| \geq \tau$, it will be considered a noise point, where f is the gray level of the point; $f_i, i = 0, \ldots, 8$, if a 3×3 neighborhood is used, are the gray levels of its neighboring points; and τ is the threshold chosen.
2. Compare the gray level of the point with the average gray level of its neighbors. If $|f - f_{\text{avg}}| \geq 0$, the point is a noise point. The advantage of this algorithm is simplicity of computation; its shortcoming is that some difficulties will be experienced in distinguishing isolating points from points on edges or on boundary lines.
3. A third way of determining whether the point is a noise point is called "fuzzy decision." Let p be the probability of the point being a noise point; then $(1 - p)$ will be the probability of the point not being a noise point. Then a linear combination of f, the gray level of that point, and f_{avg}, the average gray level of its neighbors ($f_{\text{avg}} = \sum f_i / k$) will give the gray level of the point under consideration, such as

$$g = (1 - p)f + p\frac{\sum f_i}{k} \tag{12.21}$$

A straightforward neighborhood averaging algorithm is discussed here for the smoothing processing. Let $f(x, y)$ be the noisy image. The smoothed image

$g(x, y)$ after averaging processing can be obtained from the following relation:

$$g(x, y) = \frac{1}{N} \sum_{n,m \in \Omega} \sum W_{n,m} f(n, m) \qquad \text{for } x, y = 0, 1, \ldots, N-1 \tag{12.22}$$

where Ω is the rectangular $n \times m$ neighborhood defined, $W_{n,m}$ the $n \times m$ weight array, and N the total number of pixels defined by Ω. A criterion can be formulated such that if

$$|f(x, y) - g(x, y)| \geq \tau \tag{12.23}$$

where τ denotes the threshold chosen, then $f(x, y)$ is replaced by $g(x, y)$. Otherwise, no change in the gray level will be made.

If, for example, the weight array $W_{n,m}$ and $f(x, y)$ are, respectively,

$$\frac{1}{N} W_{n,m} = \frac{1}{9} \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

and

$$f(x, y) = \begin{vmatrix} 100 & 120 & 100 \\ 120 & \enclose{circle}{200} & 90 \\ 100 & 90 & 100 \end{vmatrix}$$

and the threshold value chosen for the example is 40, then

$$g(x, y) = \begin{vmatrix} 100 & 120 & 100 \\ 120 & \enclose{circle}{113} & 90 \\ 100 & 90 & 100 \end{vmatrix}$$

where the circled pixel "200" is replaced by the average gray level "113" of the nine pixels. Note that the multiplication $\sum \sum W_{n,m} f(n, m)$ in Eq. (12.22) is not a matrix multiplication, but is

$$\begin{aligned} g(x, y) &= \frac{1}{N} \sum_{n,m=1}^{3} \sum W_{n,m} f(n, m) \\ &= \frac{1}{9} \begin{vmatrix} W_{11} f(1, 1) & W_{12} f(1, 2) & W_{13} f(1, 3) \\ W_{21} f(2, 1) & W_{22} f(2, 2) & W_{23} f(2, 3) \\ W_{31} f(3, 1) & W_{32} f(3, 2) & W_{33} f(3, 3) \end{vmatrix} \end{aligned} \tag{12.24}$$

where

$$W_{n,m} = \begin{vmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{vmatrix}$$

and

$$f(n,m) = \begin{vmatrix} f(1,1) & f(1,2) & f(1,3) \\ f(2,1) & f(2,2) & f(2,3) \\ f(3,1) & f(3,2) & f(3,3) \end{vmatrix}$$

Let us take another example for illustration. Let

$$f(x,y) = \begin{vmatrix} 100 & 120 & 100 \\ 120 & \enclose{circle}{140} & 90 \\ 100 & 90 & 100 \end{vmatrix}$$

where the gray level of the pixel to be processed is 140, while those of its neighbors are the same as in the preceding example. Since the absolute value of the difference between 140 and the average of the nine pixels (107) is less than the threshold, no change in gray-level value is made on the circled pixel. Move the mask over every pixel in the image, and a smoothed image can be obtained. The reason for using the threshold in the process is to reduce the blurring effect produced by the neighborhood averaging.

Some other masks that were suggested are shown in Figure 12.28. Note that in parts (a) and (b), different weights are given to the centering pixel and its neighbors. In parts (c) and (d), the averaging process is operated only on the neighboring pixels and different neighbors are taken into consideration. This smoothing process can be generalized as

$$g(x,y) = \begin{cases} \frac{1}{N} \sum\sum_{n,m \in \Omega} W_{n,m} f(n,m) & \\ & \text{if } \left| f(x,y) - \frac{1}{N} \sum\sum_{n,m \in \Omega} W_{n,m} f(n,m) \right| > \tau \\ f(x,y) & \text{otherwise} \end{cases} \tag{12.25}$$

$$W = \frac{1}{10} \begin{vmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

(a)

$$W = \frac{1}{16} \begin{vmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{vmatrix}$$

(b)

$$W = \frac{1}{4} \begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{vmatrix}$$

(c)

$$W = \frac{1}{8} \begin{vmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

(d)

FIGURE 12.28 Various masks frequently used in image smoothing.

Smoothing by neighborhood averaging is effective in removing the noise from an image. On the other hand, it introduces an adverse effect by losing part of the edge information, and blurring results. It is, of course, our desire to have the noise smoothed out but to retain the high-frequency contents of the image. From psychological studies we know that human eyes can tolerate more noise in areas of high signal strength (call them areas of high activity) than in areas of low signal strength (areas of low activity). By "low signal strength" we refer to the image region, with very little energy in the higher frequencies. It seems to be a good idea to process the image in such a way that in areas of high activity the image be left untouched, with only areas of low activity in the noisy image smoothed, that is, removing the random noises from an image selectively while retaining visual resolution. Statistical means can be applied to our case to determine whether a given area has high or low signal activity. Assume that the noise image contains gaussian noise with zero mean and standard deviation, and use the following notation for analysis:

I = noisy image
B = blurred image
$D = I - B$ = difference image

A blurred image resulting from neighborhood averaging may be regarded as a low-order fit to the original image. If the low-order fit to the data is perfect, the blurred image equals the original nonnoisy image. It follows that σ_D^2 will be equal to σ_n^2, where σ_D^2 is the variance in the difference image, and σ_n^2 is that in the original image. From this, a conclusion can be drawn: If $\sigma_D^2 \leq \sigma_n^2$, the blurred image is an adequate representation of the original image. This implies that the original signal has low signal activity. But if $\sigma_D^2 > \sigma_n^2$, the image cannot be well represented by the low-order function. Hence σ_n^2 can be used as a logical threshold to test σ_D^2 to see in which category of the signal activity the image region really belongs.

Based on the foregoing analysis, an algorithm can be designed to our satisfaction for generation of a new image N. Partition the image into a number of areas. In areas of low activity we pass their smoothed images, while in the area of high activity with very small values of σ_n^2/σ_D^2 (i.e., $\ll 1$), we just leave this image section I_i intact (i.e., as it is). For some areas with moderate values of the ratio σ_n^2/σ_D^2, we compute a weighted average of B_i and I_i for them:

$$N_i = (w_{Bi})B_i + (w_{Ii})I_i \tag{12.26}$$

where I_i is the ith image section of the original image, B_i is the blurred image of the ith image section, and N_i is the new image generated for the ith image section. w_{Bi} and w_{Ii} are, respectively, their weights which will be shown, respectively, as θ_i and $(1 - \theta_i)$.

The algorithm can then be summarized as follows:

1. Obtain B by blurring the noisy image I.
2. Compute the difference image D, where $D = I - B$. From this difference, we can compute σ_n^2, the variance of the noise.
3. Partition the image into $p \times q$ sections denoted as I_i.
4. For each section I_i, smooth it to obtain B_i and compute the difference image D_i and its variance σ_{Di}^2.
5. For each section, define θ_i as

$$\theta_i = \min\left\{1.0, \frac{\sigma_n^2}{\sigma_{Di}^2}\right\} \tag{12.27}$$

   where σ_n^2 is the variance of the original noisy image as a whole, which has been computed in step 2. σ_{Di}^2 is the variance of this ith difference image section.
6. Compute a new image for the ith section with

$$N_i = \theta_i B_i + (1 - \theta_i) I_i \tag{12.28}$$

7. Repeat this process for all these sections to obtain a new image.

12.4 EDGE SHARPENING

As just discussed, the smoothing of an image by neighborhood averaging is analogous to an integration process, In the integration process, part of the edge information is lost and blurring results. Differentiation is the reverse of integration, and therefore by using differentiation, a sharpening effect can be expected on the edge.

If we are given a two-dimensional image function $f(x, y)$, a vector gradient $\mathbf{G}[f(x, y)]$ can be formed as

$$\mathbf{G}[f(x, y)] = \left|\begin{matrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{matrix}\right| \tag{12.29}$$

The direction of this vector is toward the maximum rate of increase of the image function $f(x, y)$, while its magnitude is represented by

$$\mathbf{G} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2} \tag{12.30}$$

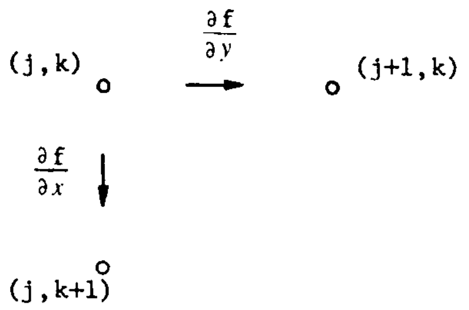


FIGURE 12.29 Digital implementation of a gradient operator.

For discrete images, the coordinate system is chosen as shown in Figure 12.29, with df/dx pointing in the vertical downward direction and df/dy in the horizontal rightward direction. Equation (12.30) can then be implemented digitally by

$$\mathbf{G} = [(f(j,k) - f(j+1,k))^2 + (f(j,k) - f(j,k+1))^2]^{1/2} \qquad (12.31)$$

which is commonly called a *three-point gradient*. This implementation is more accurate, but is computationally expensive. If the absolute values of the terms inside the brackets under the square root are taken for the value of $|G|$,

$$|G| \simeq |f(j,k) - f(j+1,k)| + |f(j,k) - f(j,k+1)| \qquad (12.32)$$

computational advantages can be achieved.

Another digital implementation for the gradient is called *Roberts' cross operator*. This is shown in Figure 12.30, where cross differences are used for the implementation as follows:

$$|G| \simeq |f(j,k) - f(j+1,k+1)| + |f(j,k+1) - f(j+1,k)| \qquad (12.33)$$

This is commonly called a *four-point gradient*.

It can easily be seen from Eqs. (12.31) to (12.33) that the magnitude of the gradient is large for prominent edges, small for a rather smooth area, and zero for a constant-gray-level area.

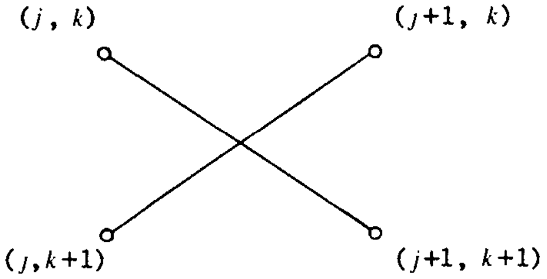


FIGURE 12.30 Robert's cross-gradient operator.

Several algorithms are available for performing edge sharpening:

Algorithm 1. *Edge Sharpening by Selectively Replacing a Pixel Point by its Gradient.* The algorithm can be put in the form shown in Figure 12.31. The computation of $|G[f(j,k)]|$ can be done using the three-point gradient or the four-point gradient expression. When the value of $|G[f(j,k)]|$ is greater than or equal to a threshold, replace the pixel by its gradient value, or by gray level 255. Otherwise, keep it in the original level or a zero (or low) gray level. The process will run point by point through the 262,144 points from top to bottom and from left to right on the image.

Different gradient images can be generated from this algorithm depending on the values chosen for the $g(j,k), j,k = 1,2,\ldots,N-1$, when its gradient $G[f(j,k)] \geq T$ and the values chosen for $g(j,k)$ when $G[f(j,k)] < T$. A binary gradient image will be obtained when the edges and background are displayed in 255 and 0. An edge-emphasized image without destroying the characteristics of smooth background can be obtained by

$$g(j,k) = \begin{cases} |G[f(j,k)]| & \text{if } |G[f(j,k)]| \geq T \\ f(j,k) & \text{otherwise} \end{cases}$$

Algorithm 2. *Edge Sharpening by Statistical Differencing.* The algorithm can be stated as follows:

(a) Specify a window (typically 7×7 pixels) with the pixel (j,k) as the center.

(b) Compute the standard deviation σ or variance σ^2 over the elements inside the window.

$$\sigma^2(j,k) = \frac{1}{N} \sum^{\sqrt{N}} \sum^{\sqrt{N}}_{j,k\in\text{window}} [f(j,k) - \bar{f}(j,k)]^2 \tag{12.34}$$

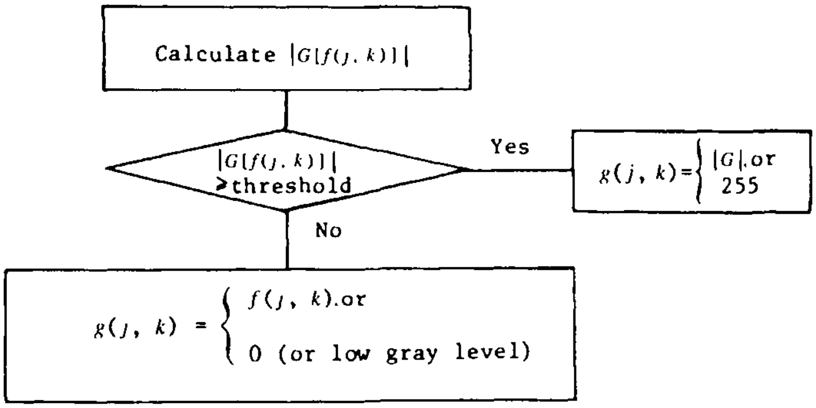


FIGURE 12.31 Algorithm for edge sharpening by selectively replacing pixel points by their gradients.

where N is the number of pixels in the window and $\bar{f}(j, k)$ is the mean value of the original image in the window.

(c) Replace each pixel by a value $g(j, k)$ which is the quotient of its original value divided by $\sigma(j, k)$ to generate the new image, such as

$$g(j, k) = \frac{f(j, k)}{\sigma(j, k)}$$

It can be noted that the enhanced image $g(j, k)$ will be increased in amplitude with respect to the original image $f(j, k)$ at the edge point and decreased elsewhere.

Algorithm 3. *Edge Sharpening by Spatial Convolution with a High-Pass Mask H.* For example,

$$g(j, k) = \sum_n \sum_m f(n, m)H(n, m, j, k) \tag{12.35}$$

Examples of high-pass masks for edge sharpening are

$$H = \begin{vmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{vmatrix} \quad H = \begin{vmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{vmatrix} \quad H = \begin{vmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{vmatrix} \tag{12.36}$$

The summation of the elements in each mask equals 1. The algorithm will be:

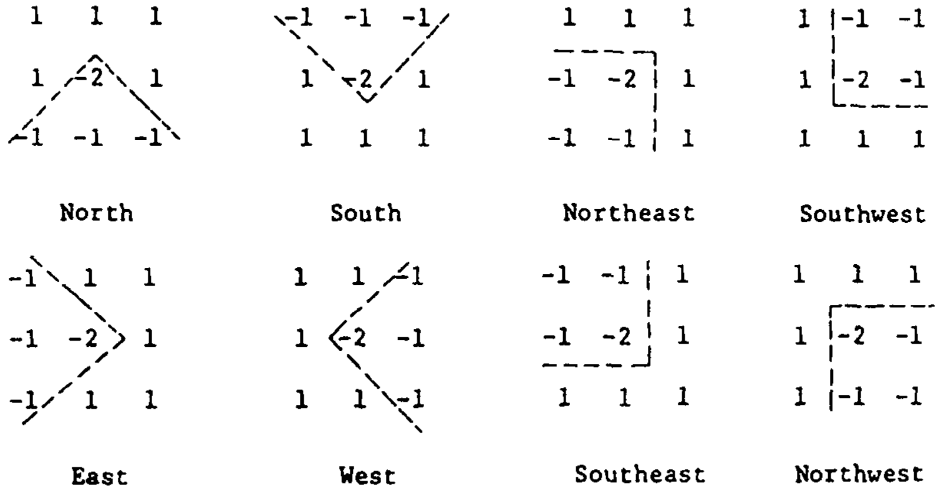
1. Examine all the pixels and compute $g(j, k)$ by convolving $f(j, k)$ with H.
2. Either use this computed value of $g(j, k)$ as the new gray level of the pixel concerned or use the original value of $f(j, k)$, depending on whether the difference between the computed $g(j, k)$ and the original $f(j, k)$ is greater than or less than the threshold chosen.

Other edge enhancement masks and operators have been suggested by various authors. These masks can be used to combine with the original image array to yield a two-dimensional discrete differentiation.

Compass gradient masks. These are so named because they indicate the slope directions of maximum responses, as shown by the dashed angle in Figure 12.32.

Algorithm 4. *Edge Sharpening with a Laplacian Operator.* Assume that the blurring process of an image may be modeled by the diffusion equation

$$\frac{\partial f}{\partial t} = k \nabla^2 f \tag{12.37}$$



$$\begin{matrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{matrix} \qquad \begin{matrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{matrix} \qquad \begin{matrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{matrix} \qquad \begin{matrix} 1 & -1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & 1 \end{matrix}$$

North South Northeast Southwest

$$\begin{matrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{matrix} \qquad \begin{matrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{matrix} \qquad \begin{matrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{matrix} \qquad \begin{matrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -1 \end{matrix}$$

East West Southeast Northwest

FIGURE 12.32 Compass gradient masks.

where $f = f(x, y, t)$ and $\nabla^2 f = \partial^2 f/\partial x^2 + \partial^2 f/\partial y^2$. Let

$$g = f(x, y, 0) \qquad \text{at } t = 0$$

be the unblurred image, and

$$f = f(x, y, \tau) \qquad \text{where } \tau > 0$$

be the blurred image. Then by Taylor's expansion, we have

$$f(x, y, 0) = f(x, y, \tau) + (0 - \tau)\frac{\partial f(x, y, \tau)}{\partial t} + \frac{(0 - \tau)^2}{2!}\frac{\partial^2 f(x, y, \tau)}{\partial t^2} + \cdots \tag{12.38}$$

Truncation at the second term gives

$$f(x, y, 0) \cong f(x, y, \tau) - \tau\frac{\partial f(x, y, \tau)}{\partial t} \tag{12.39}$$

Substituting (12.37) into (12.39) yields

$$g \cong f - k\tau\, \nabla^2 f \tag{12.40}$$

where $k\tau$ is a constant. Equation (12.40) can be interpreted as

$$\begin{aligned} \text{Unblurred image} &= \text{blurred image} - \text{positive constant} \\ &\quad \times \text{laplacian of the blurred image} \end{aligned} \tag{12.41}$$

In other words, we can simply use a subtractive linear combination of the blurred image and its laplacian to restore the unblurred image.

Figure 12.33 gives some processing results obtained with this method. The original image shown in Figure 12.33a was very bad, and the details of the face can hardly be identified. This image is improved, although still not very good, is shown in Figure 12.33b. This method is useful for those images taken in very bad environments and need a very fast process to improve it. It is possible to improve the image a little bit more if careful selection of the constant is made. Figure 12.34a and b shows the original image and the processed one by this algorithm.

Reference to Figure 12.35 gives ·

$$\frac{\partial f}{\partial x} = \frac{f_{x+1,y} - f_{x,y}}{\Delta x}$$

$$\frac{\partial f}{\partial x} = \frac{f_{x,y} - f_{x-1,y}}{\Delta x}$$

and

$$\frac{\partial f}{\partial y} = \frac{f_{x,y+1} - f_{x,y}}{\Delta y}$$

$$\frac{\partial f}{\partial y} = \frac{f_{x,y} - f_{x,y-1}}{\Delta y}$$

The laplacian of the image function $f(x, y)$ will then be

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{12.44}$$

or

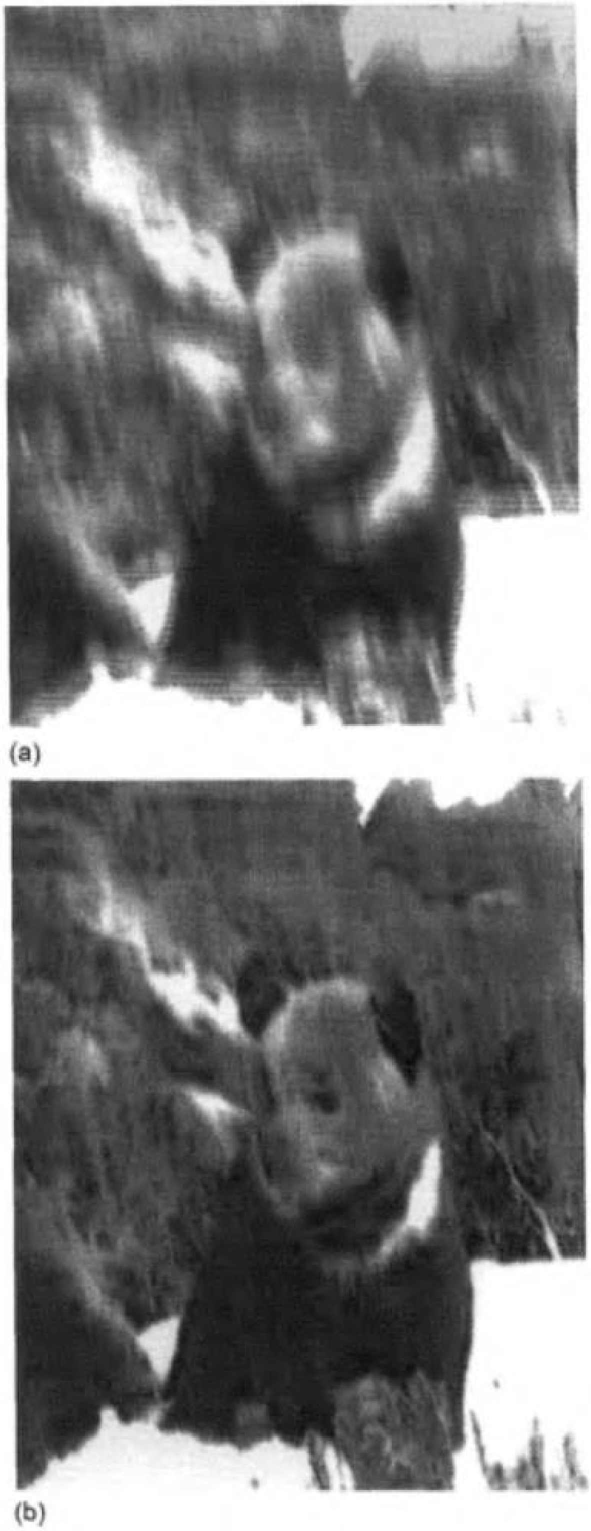
$$\nabla^2 f(x, y) = \frac{(f_{x+1,y} - f_{x,y}) - (f_{x,y} - f_{x-1,y})}{(\Delta x)^2} + \frac{(f_{x,y+1} - f_{x,y}) - (f_{x,y} - f_{x,y-1})}{(\Delta y)^2} \tag{12.45}$$

If both Δx and Δy are chosen to be unity, which is the usual practice in digital image processing, we have

$$\nabla^2 f(x, y) = 4(f_{\text{avg}} - f_{x,y}) \tag{12.46}$$

where

$$f_{\text{avg}} = \tfrac{1}{4}(f_{x+1,y} + f_{x-1,y} + f_{x,y+1} + f_{x,y-1}) \tag{12.47}$$



(a)

(b)

FIGURE 12.33 Processed results obtained with the method of edge sharpening with a laplacian operator. (a) Original image; (b) processed image.

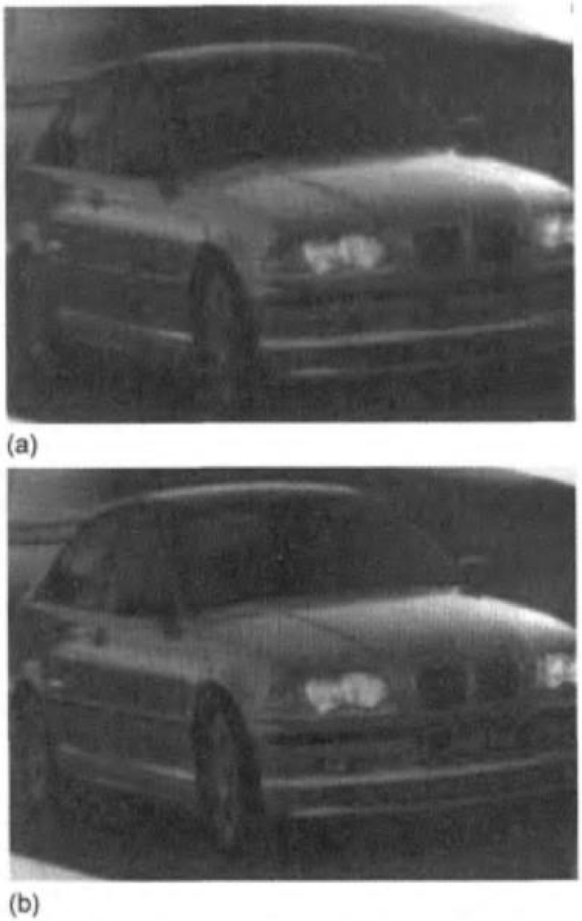


FIGURE 12.34 Processed results obtained with the method of edge sharpening with a laplacian operator. (b) Original image; (b) processed image.

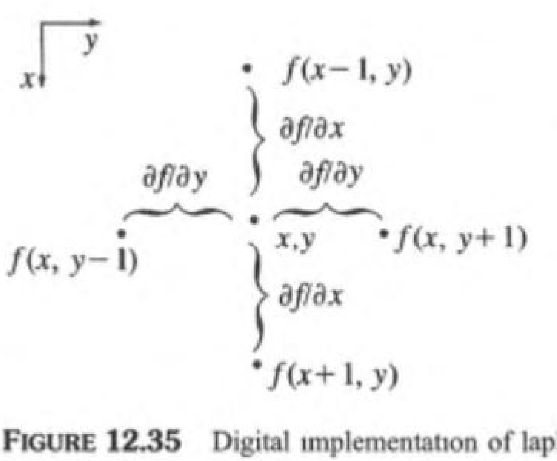


FIGURE 12.35 Digital implementation of laplacian operator.

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix} \qquad \begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix} \qquad \begin{matrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{matrix}$$

(a) **(b)** **(c)**

FIGURE 12.36 Laplacian masks.

is the average of the four neighbors of the image point under consideration. Equations (12.46) and (12.47) can further be put in a form as the convolution of a mask (called a laplacian mask) with the image function window

$$\nabla^2 f(x,y) = \begin{vmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{vmatrix} * \begin{vmatrix} 0 & f_{x-1,y} & 0 \\ f_{x,y-1} & f_{x,y} & f_{x,y+1} \\ 0 & f_{x+1,y} & 0 \end{vmatrix} \tag{12.48}$$

The laplacian mask can be used to detect lines, line ends, and points over edges. By convolution of an image with a Laplacian mask, we can obtain edge sharpening without regard to edge direction. The mask shown in part (b) of Figure 12.36 is obtained by adding the mask shown in part (a) to the result obtained when part (a) is rotated by 45°. The mask shown in part (c) is obtained by subtracting the mask in part (a) after it has been rotated by 45°, from twice the mask shown in part (a). The laplacian of pixels at the edge is high, but it is not as high as that at the noise point. This is because the edge is directional, whereas for the noise point, both $\nabla_x^2 f$ and $\nabla_y^2 f$ are high. With this property in mind, the noise points can be distinguished from the edge pixels. Some other measures will be provided for the detection of edges only.

Algorithm 5. *Nonlinear Edge Operators.* Among the nonlinear edge operators are the Sobel operator, Kirsch operator, and Wallis operator.

Sobel operator. This operator is a 3 × 3 window centered at (j,k), as shown in Figure 12.37. The intensity gradient at the point (j,k) is defined as either

$$s = (s_x^2 + s_y^2)^{1/2} \tag{12.49}$$

or

$$s = |s_x| + |s_y| \tag{12.50}$$

where s_x and s_y are, respectively, computed from its neighbors according to

$$s_x = (A_6 + 2A_5 + A_4) - (A_0 + 2A_1 + A_2) \tag{12.51}$$

$$s_y = (A_2 + 2A_3 + A_4) - (A_0 + 2A_7 + A_6) \tag{12.52}$$

$$\begin{array}{ccc} A_0 & A_1 & A_2 \\ A_7 & f(j,k) & A_3 \\ A_6 & A_5 & A_4 \end{array}$$

FIGURE 12.37 A 3 × 3 window for edge detection.

or the Sobel high-pass weighting masks are, respectively, W_x and W_y for computation of the horizontal and vertical components of the gradient vector in the x and y directions at the center point of the 3 × 3 window shown in Figure 12.37.

$$W_x = \begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{vmatrix} \tag{12.53}$$

and

$$W_y = \begin{vmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{vmatrix} \tag{12.54}$$

Convolving these masks with an image $f(x, y)$ over all the points on the image gives the gradient image. Figures 12.38 to 12.41 show the original images, the images comprising all the horizontal edge elements, those comprising all the vertical edge elements, as well as the complete images which combine all the edge elements responding to various directional masks.

Kirsh edge operator. Another 3 × 3 nonlinear edge enhancement algorithm was suggested by Kirsch (see Figure 12.37). The subscripts of its neighbors are made such that they are labeled in ascending order. Modulo 8 arithmetic is used in this computation. Then the enhancement value of the pixel is given as

$$G(j, k) = \max\left(1, \max_{i=0}^{7} [|5S_i - 3T_i|]\right) \tag{12.55}$$

where S_i and T_i are computed, respectively, from

$$S_i = A_i + A_{i+1} + A_{i+2} \tag{12.56}$$

and

$$T_i = A_{i+3} + A_{i+4} + A_{i+5} + A_{i+6} + A_{i+7} \tag{12.57}$$

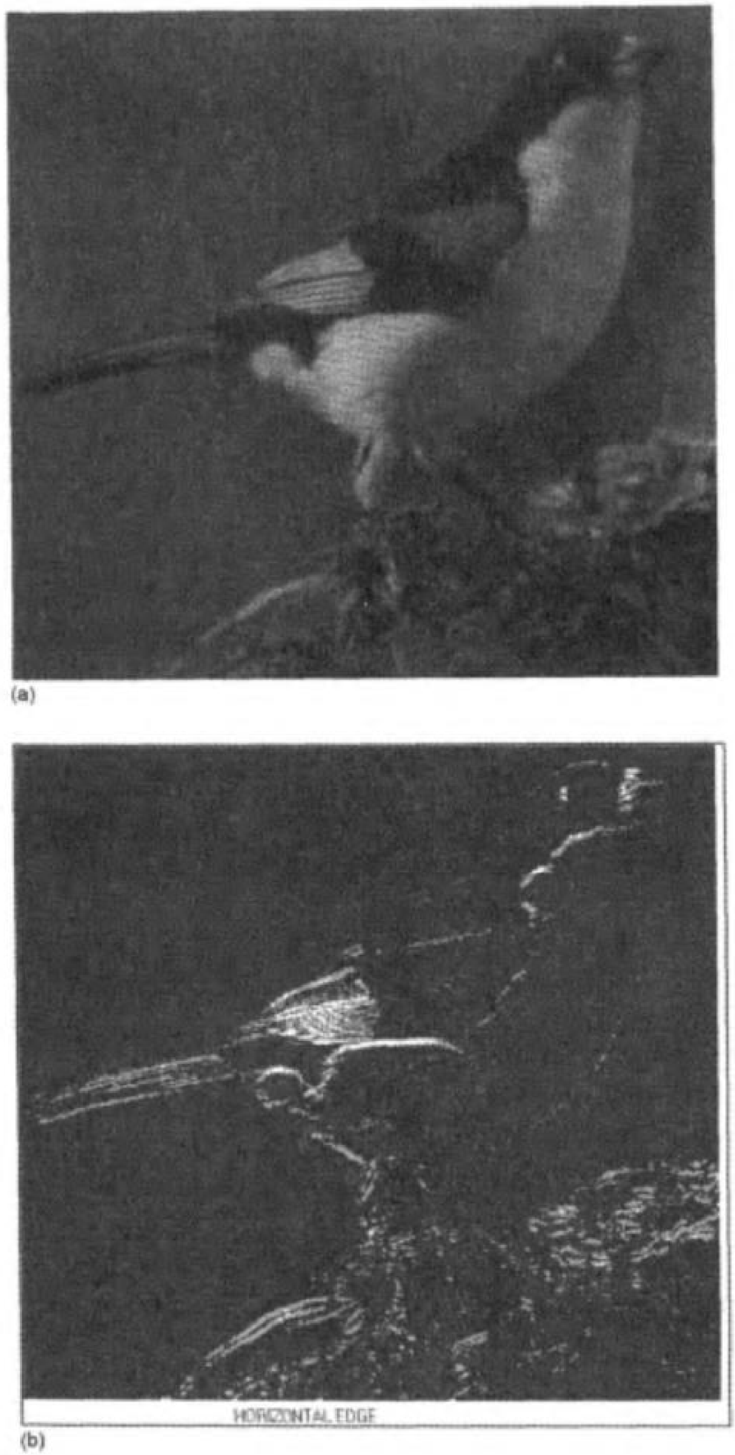


FIGURE 12.38 Gradient image obtained by applying Sobel edge operator. (a) Original image; (b) response of all horizontal edges; (c) response of all vertical edge elements; (d) response of all 45° edge elements; (e) response of all 135° edge elements; (f) response of all $(s_x + s_y)$ edge elements; (g) response of $(s_{45} + s_{135})$ edge elements; (h) complete gradient image.

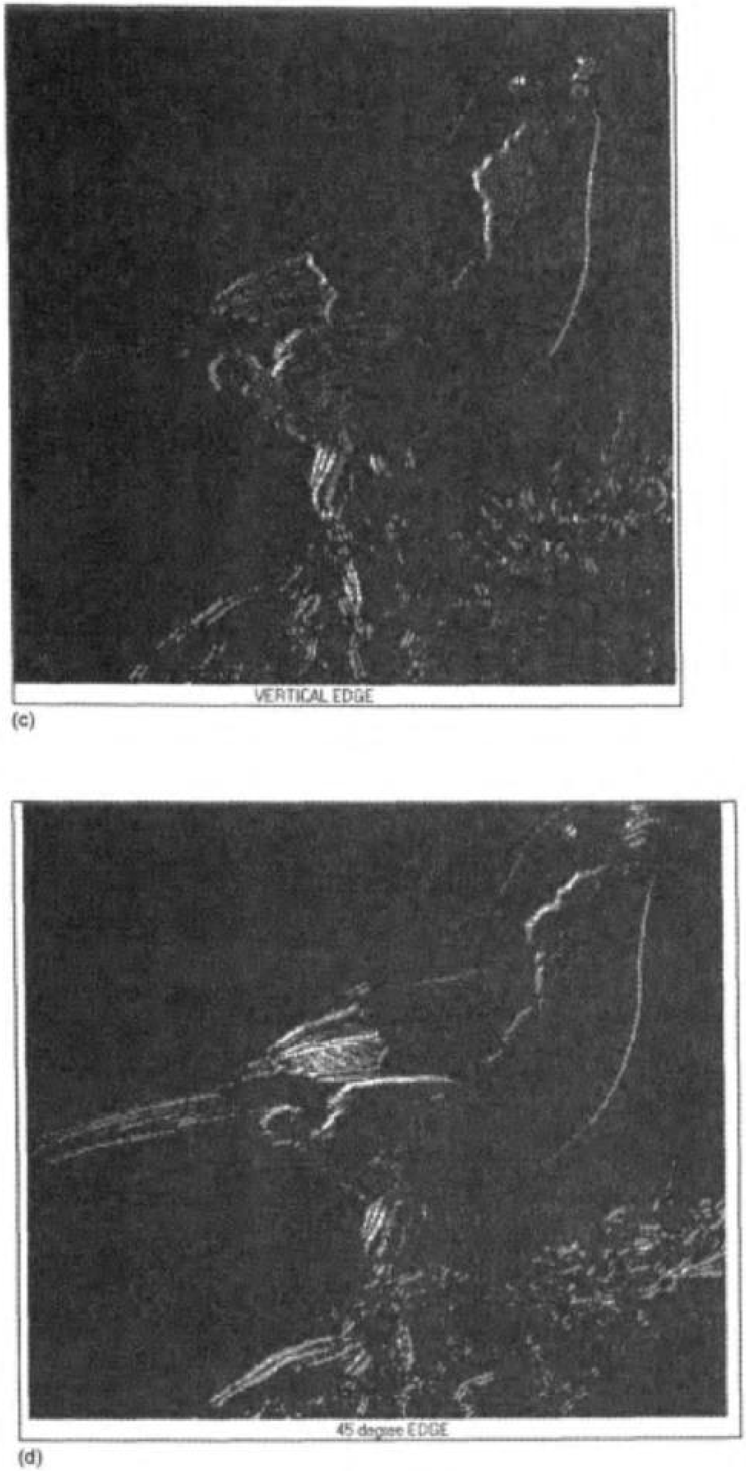


FIGURE 12.38 *Continued*

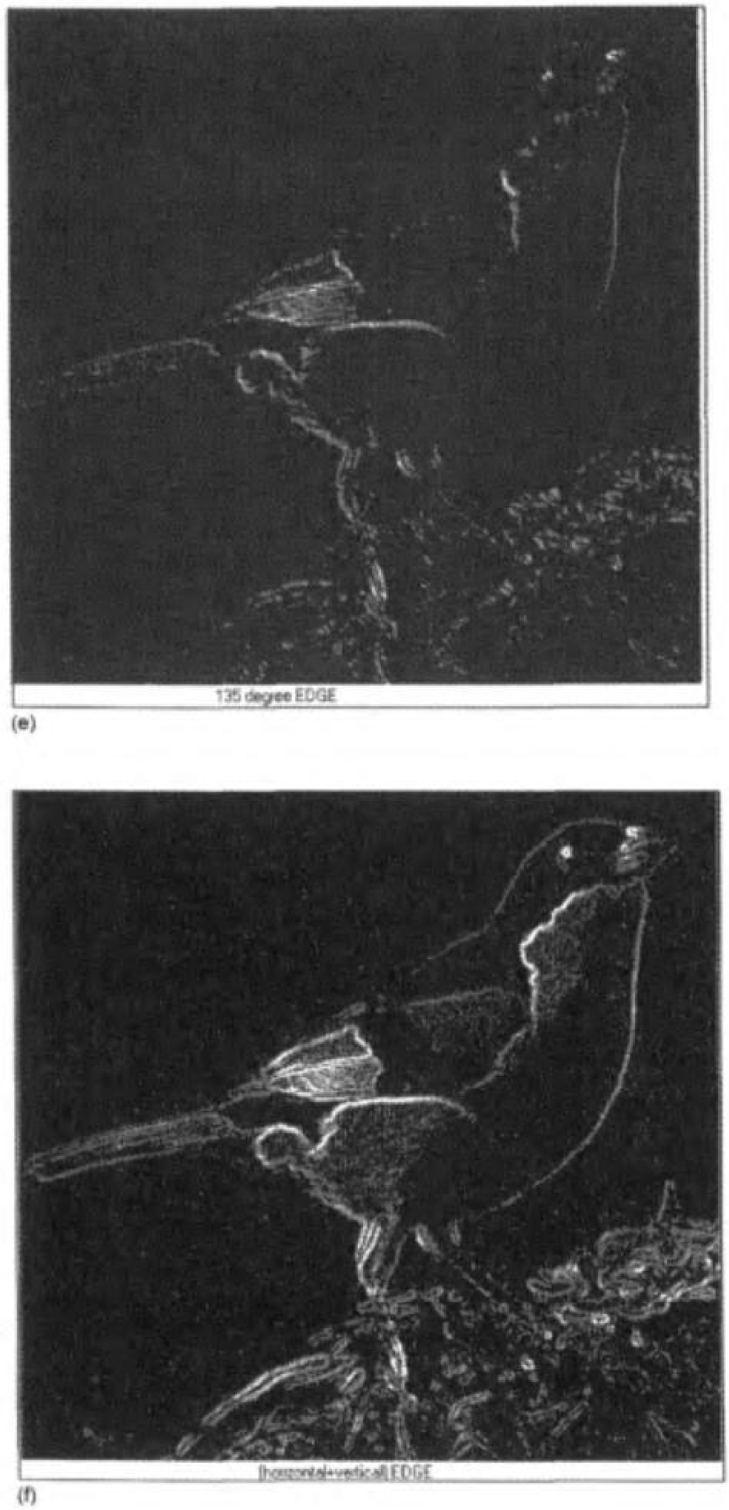


FIGURE 12.38 *Continued*

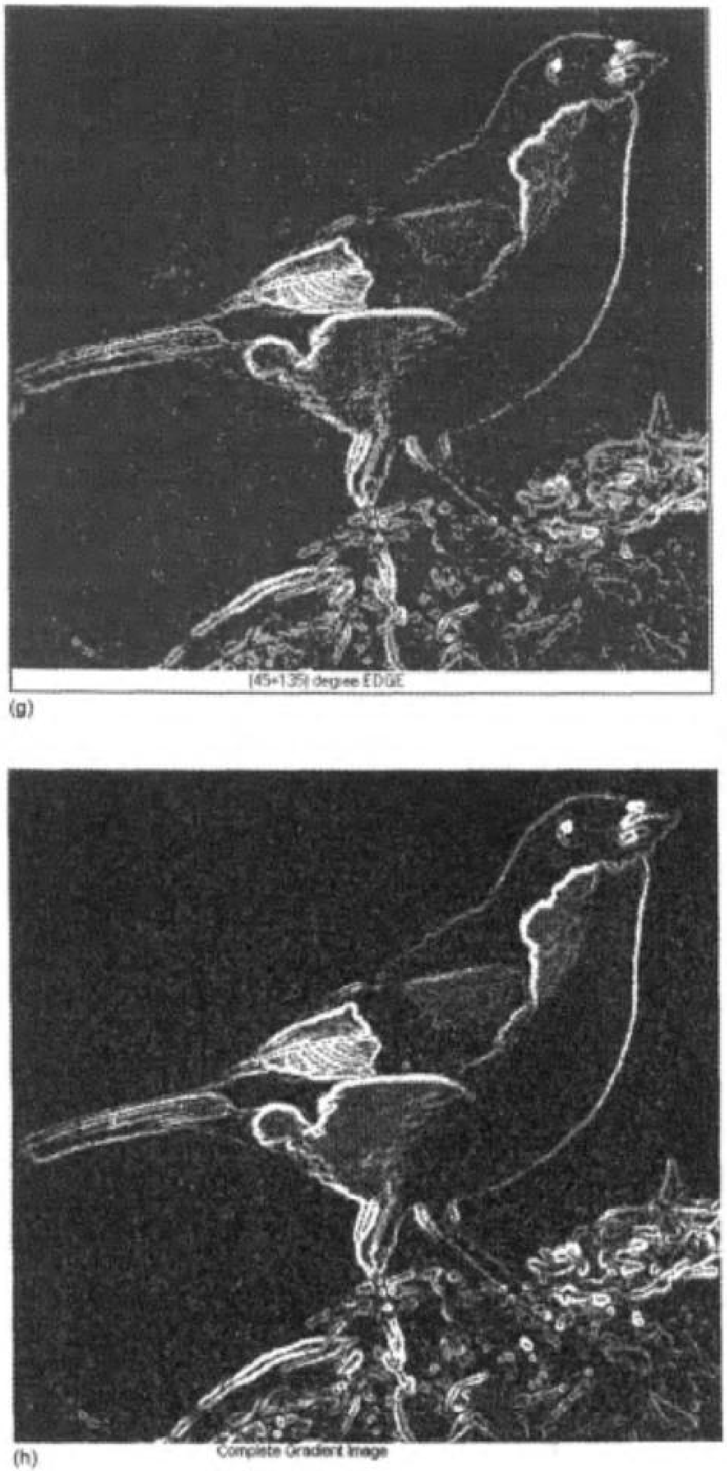


FIGURE 12.38 *Continued*

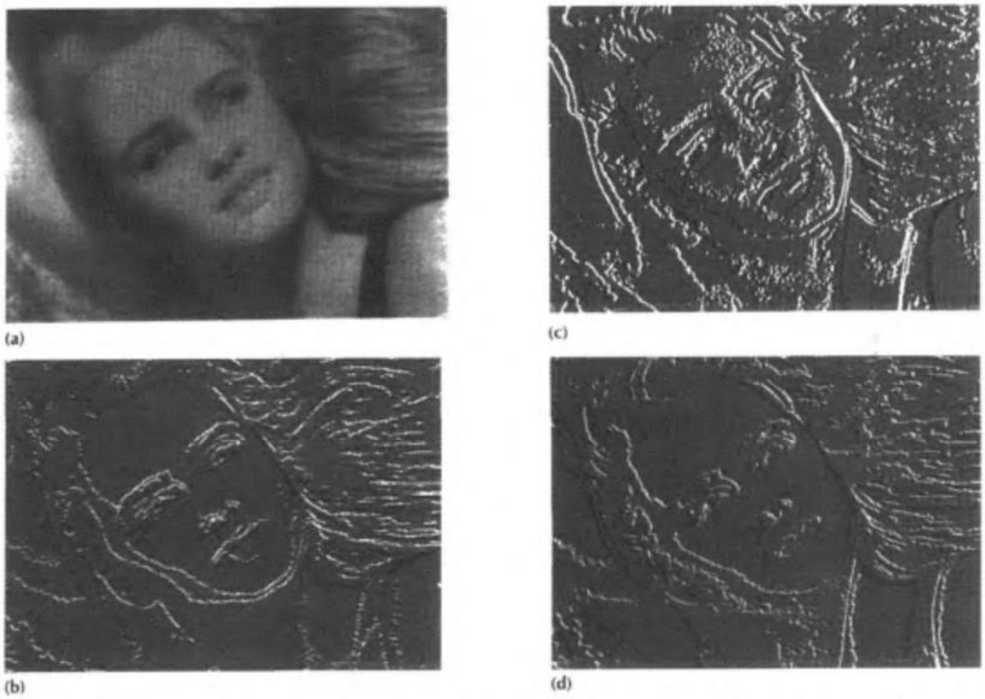
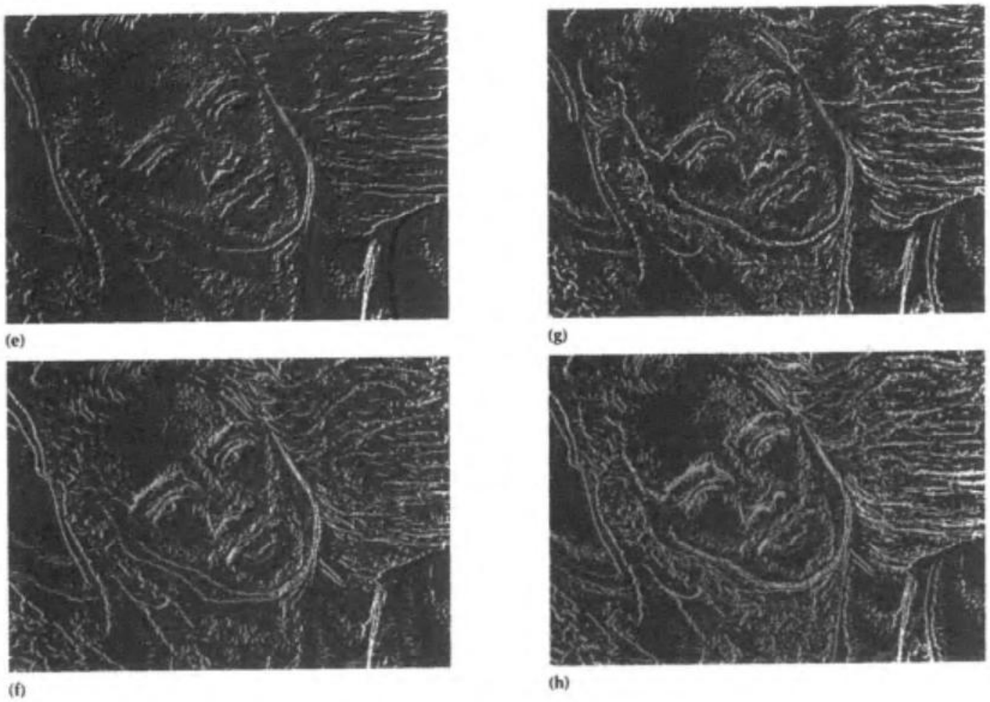


FIGURE 12.39 Gradient image obtained by applying Sobel edge operator. (a) Original image; (b) response of all horizontal edges; (c) response of all vertical edge elements; (d) response of all 45° edge elements; (e) response of all 135° edge elements; (f) response of all $(s_x + s_y)$ edge elements; (g) response of $(s_{45} + s_{135})$ edge elements; (h) complete gradient image.



(e)

(f)

(g)

(h)

FIGURE 12.39 *Continued*

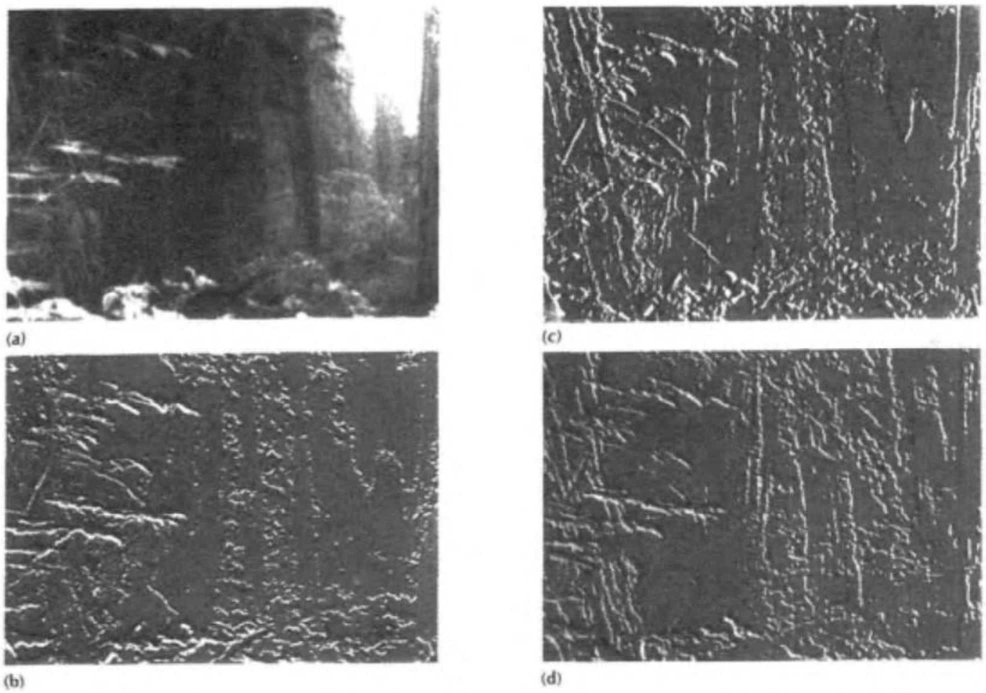


FIGURE 12.40 Gradient image obtained by applying Sobel edge operator. (a) Original image; (b) response of all horizontal edges; (c) response of all vertical edge elements; (d) response of all 45° edge elements; (e) response of all 135° edge elements; (f) response of all $(s_x + s_y)$ edge elements; (g) response of $(s_{45} + s_{135})$ edge elements; (h) complete gradient image.

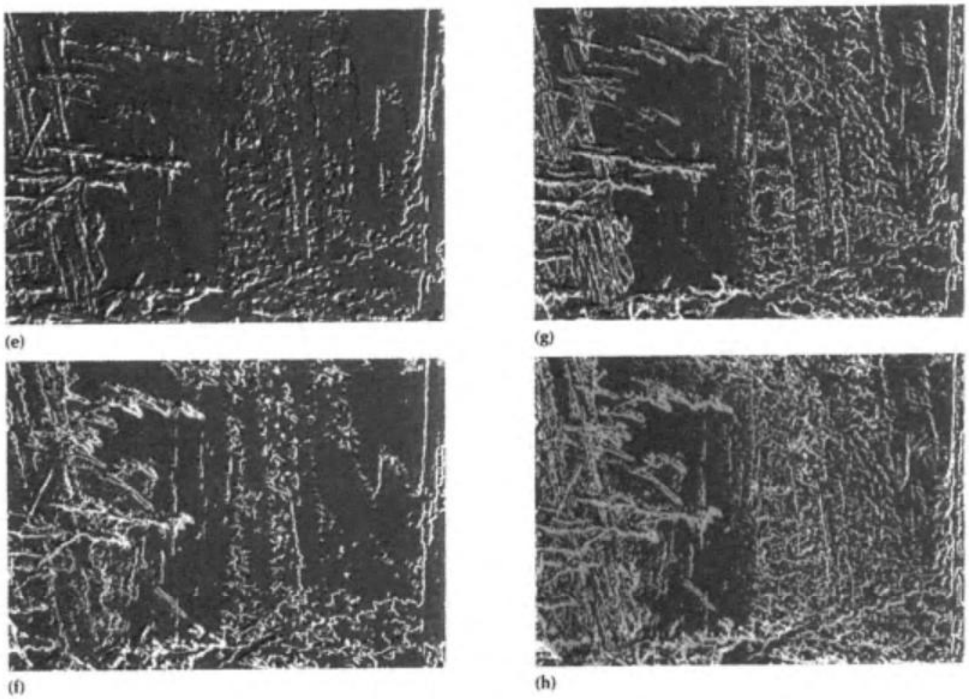


FIGURE 12.40 *Continued*

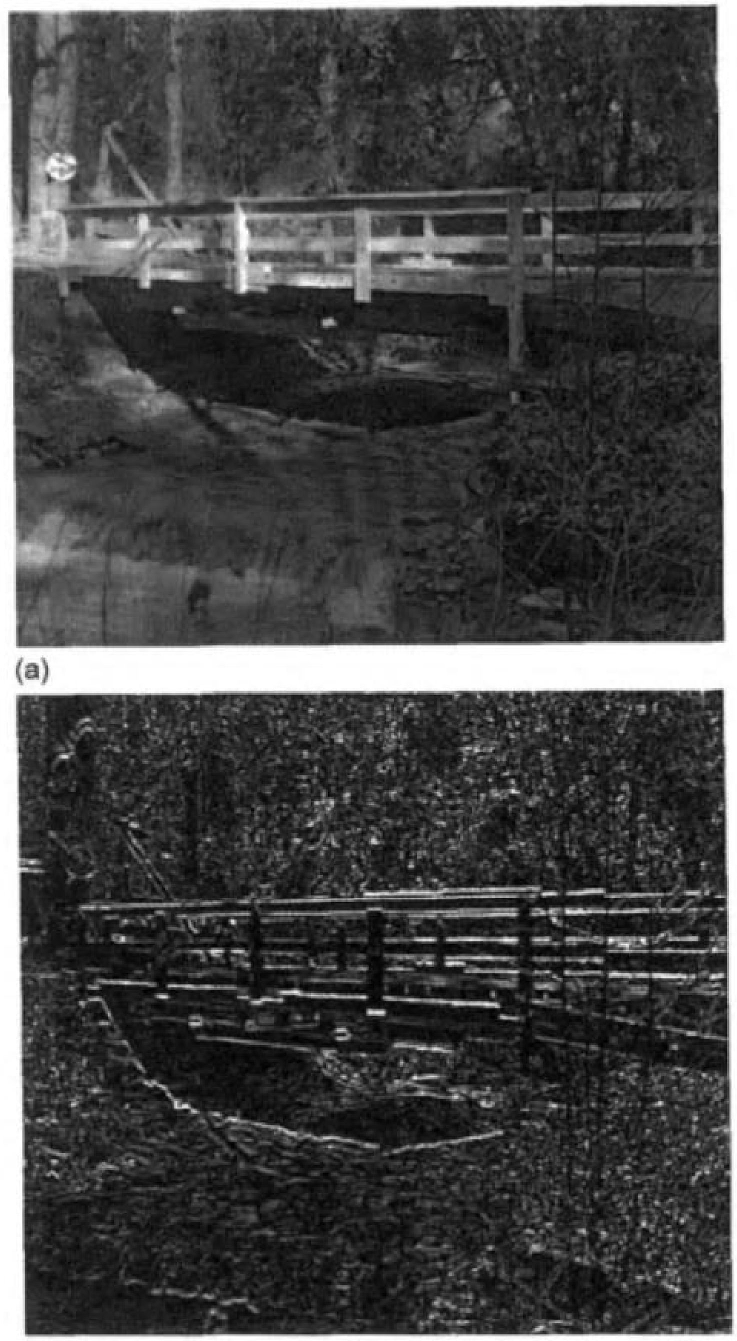
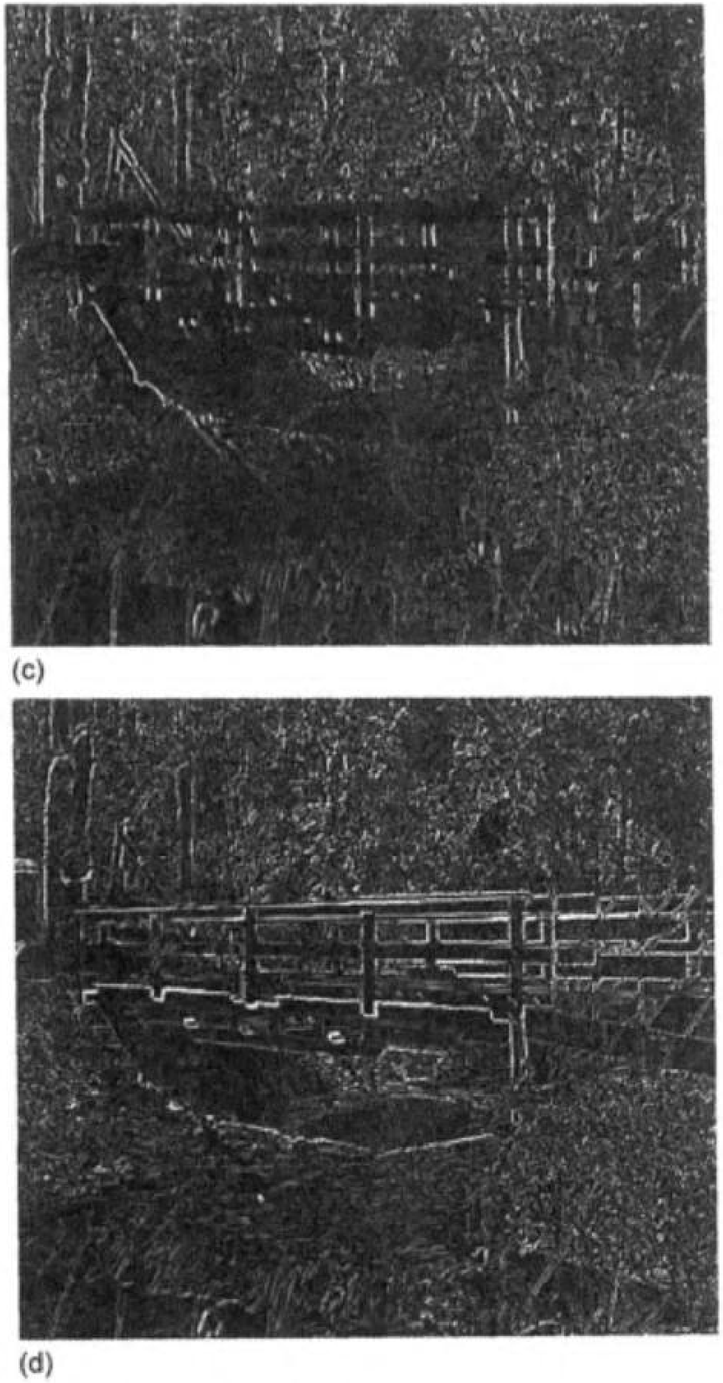


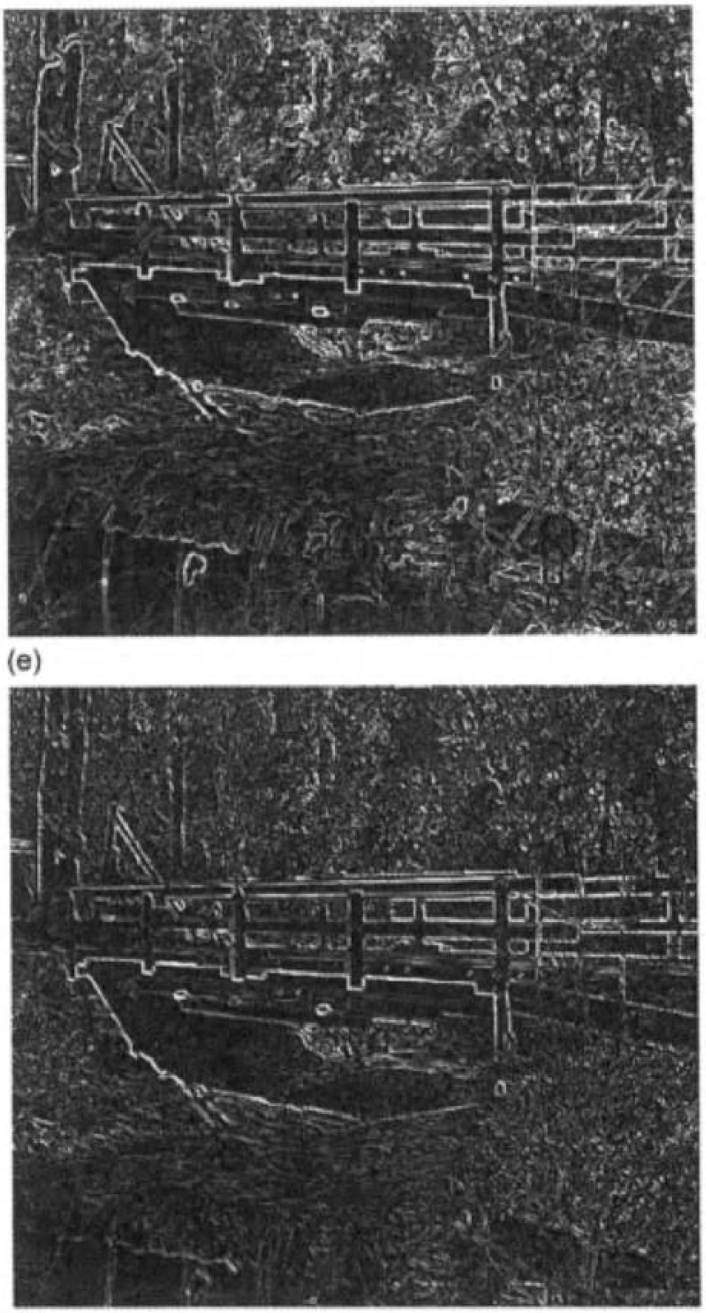
FIGURE 12.41 Gradient image obtained by applying Sobel edge operator. (a) Original image; (b) response of all honzontal edges; (c) response of all vertical edge elements; (d) response of all 45° edge elements; (e) response of all 135° edge elements; (f) response of all $(s_x + s_y)$ edge elements; (g) response of $(s_{45} + s_{135})$ edge elements; (h) complete gradient image.



(c)

(d)

FIGURE 12.41 *Continued*



(e)

(f)

FIGURE 12.41 *Continued*

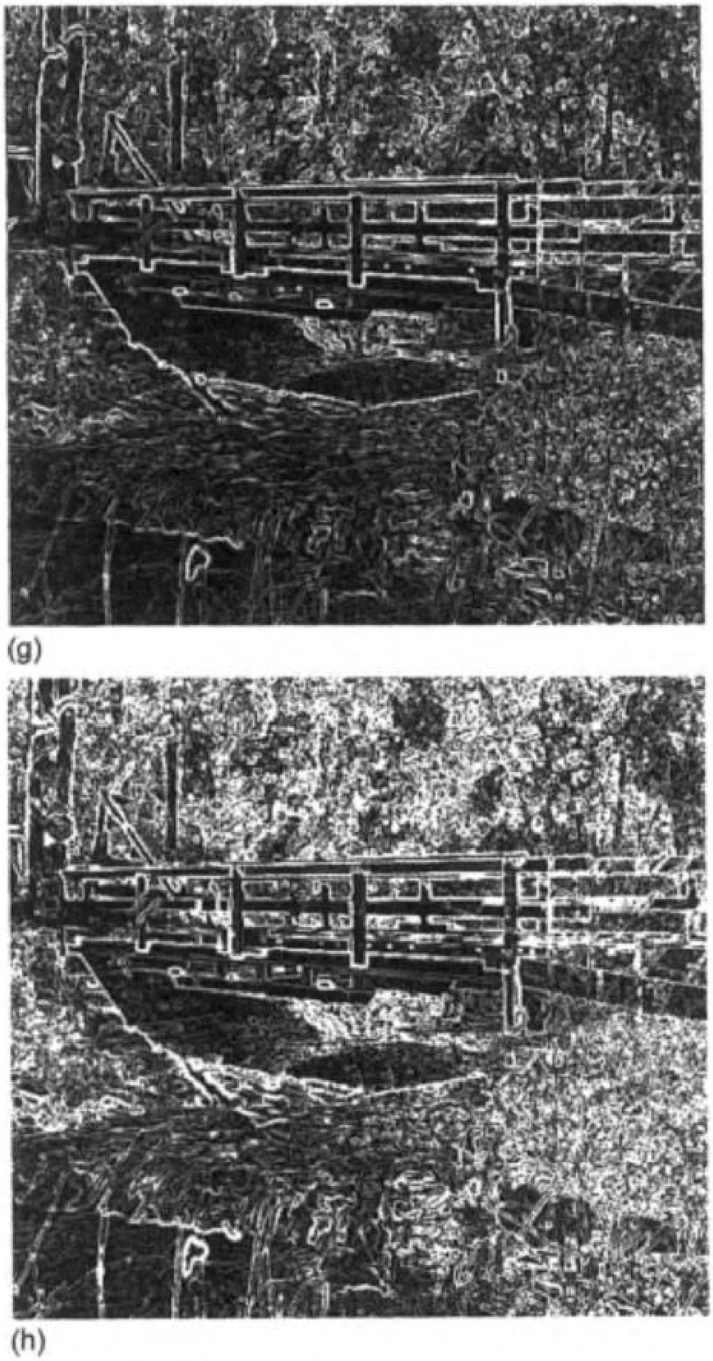


FIGURE 12.41 *Continued*

5	1	1
5	1	1
5	1	1

FIGURE 12.42 Illustrative window.

Take, for example, an illustrative window as shown in Figure 12.42. When $i = 2$,

$$S_2 = A_2 + A_3 + A_4 = 3$$

$$T_2 = A_5 + A_6 + A_7 + A_0 + A_1 = 17$$

and

$$|5S_2 - 3T_2| = 36$$

Similarly, we can get eight values for $|5S_i - 3T_i|$, $i = 0, \ldots, 7$, as follows:

i	S_i	T_i	$\lvert 5S_i - 3T_i \rvert$
0	7	13	4
1	3	17	36
2	3	17	36
3	3	17	36
4	7	13	4
5	11	9	28
6	15	5	60
7	11	9	28

Substitution of these values into Eq. (12.55) gives the gradient at the point $G(j, k) = 60$. It is not difficult to see from Eq. (12.55) that when the window is passed into a smoothed region (i.e., with the same gray level in the neighbors as the center pixel), $|5S_i - 3T_i| = 0$, $i = 0, \ldots, 7$. Then the gradient $G(j, k)$ at this center pixel will assume a value of 1. Basically, the Kirsch operator provides the maximal compass gradient magnitude about an image point [ignoring the pixel value of $f(j, k)$].

Wallis edge operator. This is a logarithmic laplacian edge detector. The principle of this detector is based on the homomorphic image processing. The assumption that Wallis made is that if the difference between the absolute value of the logarithm of the pixel luminance and the absolute value of the average of the logarithms of its four nearest neighbors is greater than a threshold value, an edge is assumed to exist.

Using the same window as that used by Kirsch (Figure 12.37), an expression for $G(j, k)$ can be put in the following form:

$$G(j, k) = \log[f(x, y)] - \tfrac{1}{4}\log[A_1 A_3 A_5 A_7] \tag{12.58}$$

or

$$G(j, k) = \frac{1}{4}\log\frac{[f(x, y)]^4}{A_1 A_3 A_5 A_7} \tag{12.59}$$

It can be seen that the logarithm does not have to be computed when compared with the threshold. The computation will be simplified. In addition, this technique is insensitive to multiplicative change in the luminance level, since $f(x, y)$ changes with A_1, A_3, A_5, and A_7 by the same ratio.

Algorithm 6. *Least-Square-Fit Operator.* Before the image is processed, it is smoothed. Referring to Figure 12.43, let $f(x, y)$ be the image model in the xy plane, and $z = ax + by + c$ be a plane that fits the four points shown. The error that resulted when $z = ax + by + c$ is taken as the image plane will be the square root of Error^2, shown by the following equation:

$$\begin{aligned}\text{Error}^2 = {} & [ai + bj + c - f(i,j)]^2 \\ & + [a(i+1) + bj + c - f(i+1,j)]^2 \\ & + [ai + b(j+1) + c - f(i,j+1)]^2 \\ & + [a(i+1) + b(j+1) + c - f(i+1,j+1)]^2 \end{aligned} \tag{12.60}$$

To optimize the solution, take $\partial(\text{Error}^2)/\partial a$, $\partial(\text{Error}^2)/\partial b$, and $\partial(\text{Error}^2)/\partial c$ and set these partial derivatives to zero. a, b and c can then be found:

$$a = \frac{f(i+1,j) + f(i+1,j+1)}{2} - \frac{f(i,j) + f(i,j+1)}{2} \tag{12.61}$$

$$b = \frac{f(i,j+1) + f(i+1,j+1)}{2} - \frac{f(i,j) + f(i+1,j)}{2} \tag{12.62}$$

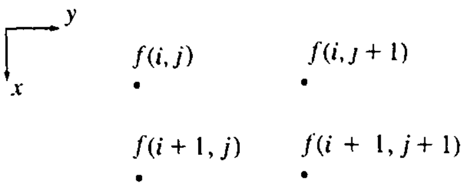


FIGURE 12.43 Window used for least-square-fit edge detection operator.

and

$$c = \tfrac{1}{4}[3f(i,j) + f(i+1,j) + f(i,j+1) - f(i+1,j+1) - ia - ib] \tag{12.63}$$

It can easily be seen that Eq. (12.61) gives the difference of the averages of pixels in two consecutive columns. Similarly, Eq. (12.62) gives the difference of the averages of pixels in two consecutive rows. The value of c given by Eq. (12.63) is more complicated, but this value is not needed in the edge detection. The gradient of the plane can then be found as

$$G = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2} \tag{12.64}$$

or

$$G = (a^2 + b^2)^{1/2} \tag{12.65}$$

which can further be approximated as

$$G = |a| + |b| \tag{12.66}$$

or

$$G = \max[|a|, |b|] \tag{12.67}$$

This gradient is usually called a digital gradient. Computation of the digital gradient is more complicated than that of the Roberts gradient. But it is less sensitive to noise, because in this process, averaging is done prior to differencing.

Algorithm 7. *Edge Detection via Zero Crossing.* Edges in image can be located through detection of the zero crossing of the second derivative of the edges. This approach applies especially well when the gray-level transition region broadens gradually rather than when there is an abrupt change. Figure 12.44a shows the case when the intensity of an edge appears as a ramp function. Parts (b) and (c) of the same figure show its first and second derivatives. Note that the second derivative crosses zero if an edge exists. Similar observations are obtained for a signal with smooth intensity change at the edge (see Figure 12.45).

As mentioned before, the laplacian operator is more sensitive to noise. Any small ripple in $f(x)$ is enough to generate an edge point and therefore a lot of artifact noises will be introduced when the laplacian operator is used alone (see Figure 12.46b). Due to this noise sensitivity the application of noise-reduction processing prior to edge detection is desirable when images with noisy background are processed. Notice that an edge point is different from a noise point in that at an edge point the local variance is sufficiently large. With this property in mind, the "false" edge points can be identified and discarded. Using a window

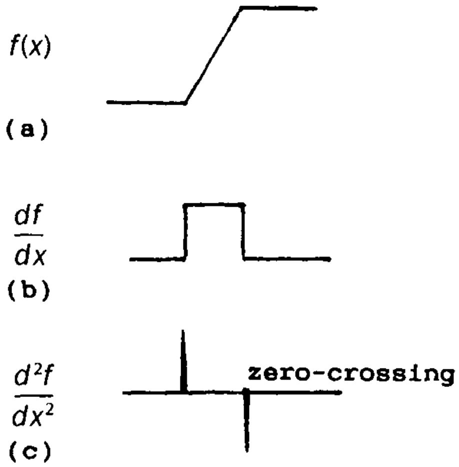


FIGURE 12.44 Edge detection via zero crossing.

$(2M+1)\times(2M+1)$, with M chosen around 2 or 3, the local variance $\sigma_f^2(i,j)$ can be estimated by

$$\sigma_f^2(i,j)=\frac{1}{(2M+1)^2}\sum_{k_1=i-M}^{i+M}\sum_{k_2=j-M}^{j+M}[f(k_1,k_2)-m_f(k_1,k_2)]^2 \tag{12.68}$$

where

$$m_f(i,j)=\frac{1}{(2M+1)^2}\sum_{k_1=i-M}^{i+M}\sum_{k_2=j-M}^{j+M}f(k_1,k_2) \tag{12.69}$$

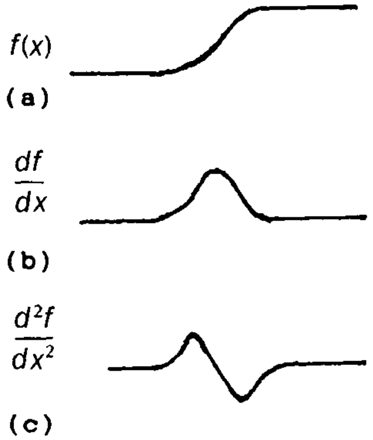


FIGURE 12.45 Edge detection via zero crossing.

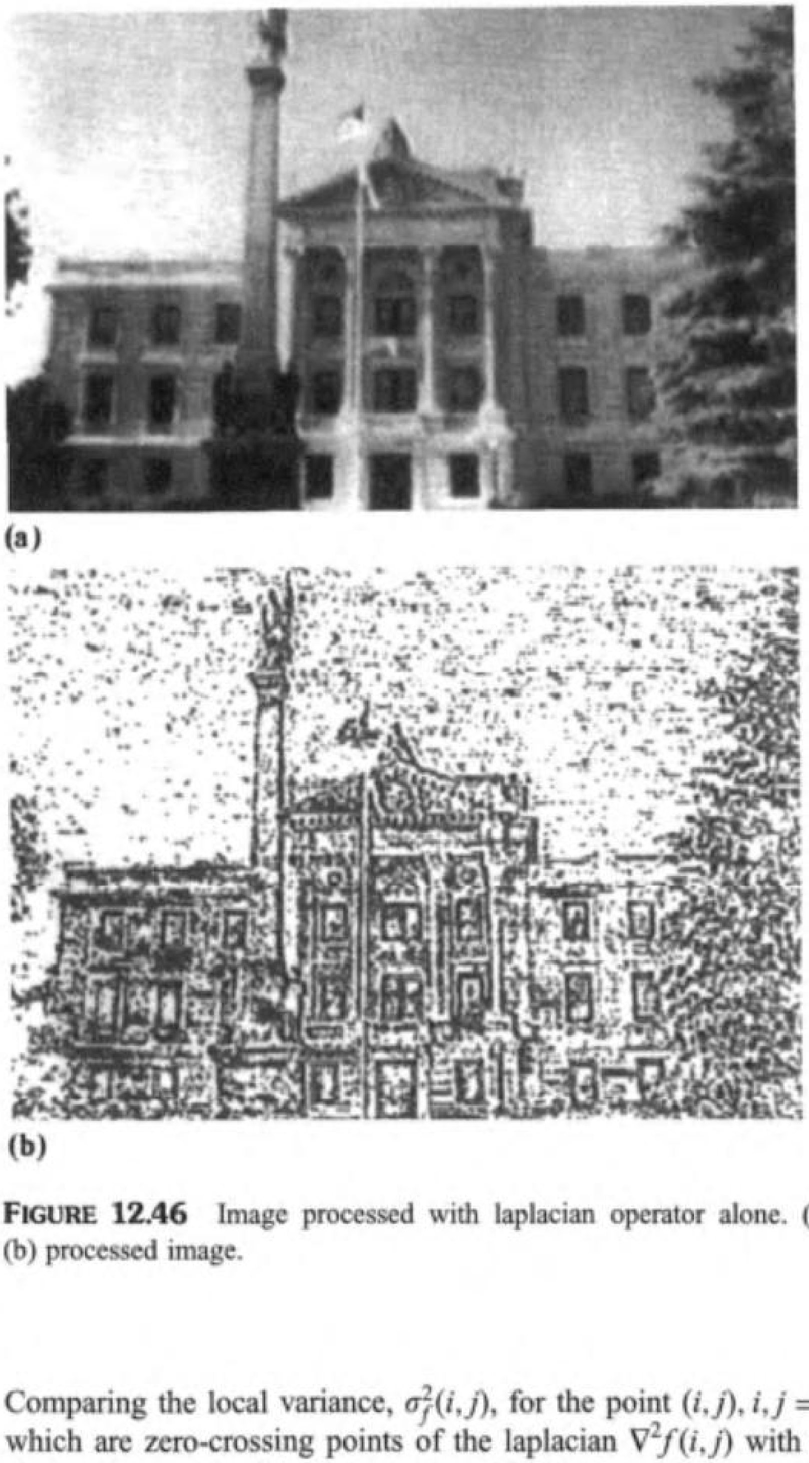


FIGURE 12.46 Image processed with laplacian operator alone. (a) Original image; (b) processed image.

Comparing the local variance, $\sigma_f^2(i,j)$, for the point $(i,j), i,j = 1, 2, \ldots, N-1$, which are zero-crossing points of the laplacian $\nabla^2 f(i,j)$ with an approximately chosen threshold will eliminate the "false" edge points accordingly.

Figure 12.47 shows the results when a laplacian operator associated with the local variance evaluation approach is applied to the image shown in Figure 12.46. Figure 12.48 is the block diagram of a laplacian operator associated with

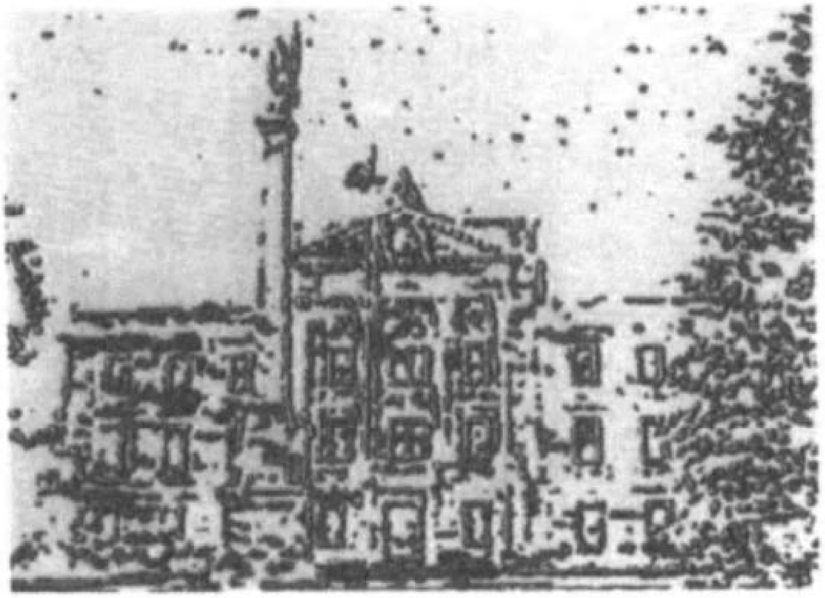


FIGURE 12.47 Result obtained after the application of the laplacian operator associated with local variance evaluation approach to the image shown in Figure 12.46a.

local variance evaluation for use with the window shown in Figure 12.49. Figure 12.50 is another example that illustrates this method.

Line and spot detection. It is clear that lines can be viewed as extended edges, and spots as isolated edges. Isolated edges can be detected by comparing

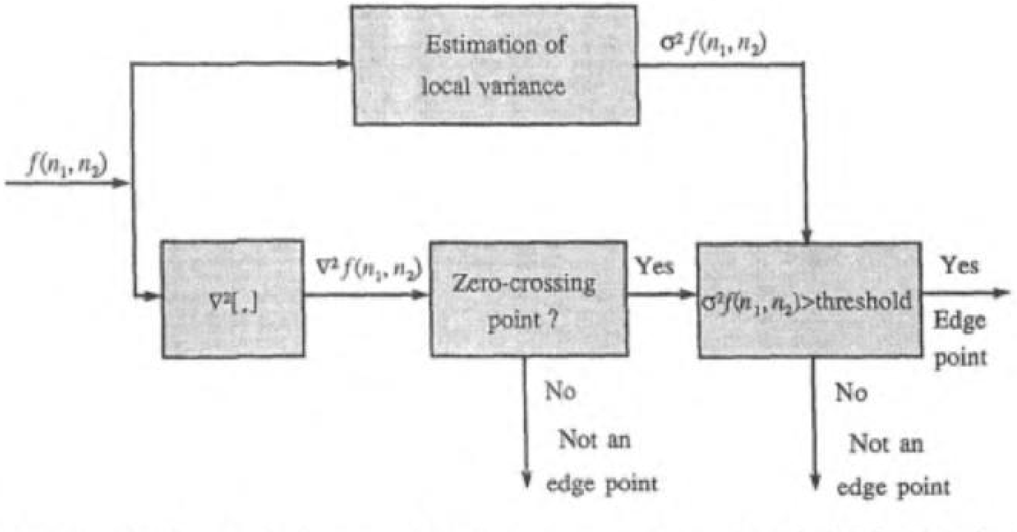


FIGURE 12.48 Block diagram of the laplacian operator associated with local variance evaluation.

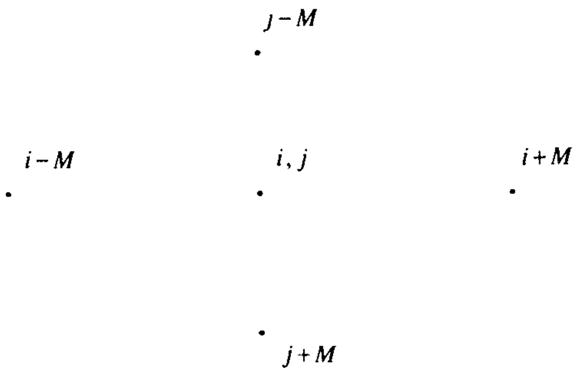


FIGURE 12.49 $(2M+1)\times(2M+1)$ window for local variance implementation.

the pixel value with the average or median of its neighborhood pixels. A 3 × 3 mask such as

$$W = \begin{vmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{vmatrix} \tag{12.70}$$

can make $\mathbf{W}^T\mathbf{x}$ substantially greater than zero at the isolated points. For line detection, the compass gradient operators shown in Figure 12.51 will respond to lines of various orientations with $\mathbf{W}_i^T\mathbf{x} > \mathbf{W}_j^T\mathbf{x}$ for all j ($j \neq i$) when $\mathbf{x}$ is closest to the ith mask. For the detection of combinations of isolated points and lines of various orientations, conceptually, we can use $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$, and $\mathbf{W}_4$ as the four masks for edge detection, and $\mathbf{W}_5, \mathbf{W}_6, \mathbf{W}_7$, and $\mathbf{W}_8$ as the four masks for line detection. By comparing the angle of the pixel vector $\mathbf{x}$, with its projections onto the "edge" subspace and that onto the "line" subspace, we can then decide to which subspace (edge or line subspace) the pixel $\mathbf{x}$ belongs, based on which of the angles is smaller. Obviously, magnitude of the projection of $\mathbf{x}$ onto the edge subspace is

$$\mathrm{MAG}_{\mathrm{edge}} = [(\mathbf{W}_1^T\mathbf{x})^2 + (\mathbf{W}_2^T\mathbf{x})^2 + (\mathbf{W}_3^T\mathbf{x})^2 + (\mathbf{W}_4^T\mathbf{x})^2]^{1/2} \tag{12.71}$$

where $\mathbf{W}_1^T\mathbf{x}, \mathbf{W}_2^T\mathbf{x}, \mathbf{W}_3^T\mathbf{x}$, and $\mathbf{W}_4^T\mathbf{x}$ represent, respectively, the projections of $\mathbf{x}$ onto the vectors $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$, and $\mathbf{W}_4$. Similar arguments apply to $\mathbf{W}_5, \mathbf{W}_6, \mathbf{W}_7$, and $\mathbf{W}_8$. The magnitude of projection of $\mathbf{x}$ onto the line subspace is

$$\mathrm{MAG}_{\mathrm{line}} = [(\mathbf{W}_5^T\mathbf{x})^2 + (\mathbf{W}_6^T\mathbf{x})^2 + (\mathbf{W}_7^T\mathbf{x})^2 + (\mathbf{W}_8^T\mathbf{x})^2]^{1/2} \tag{12.72}$$

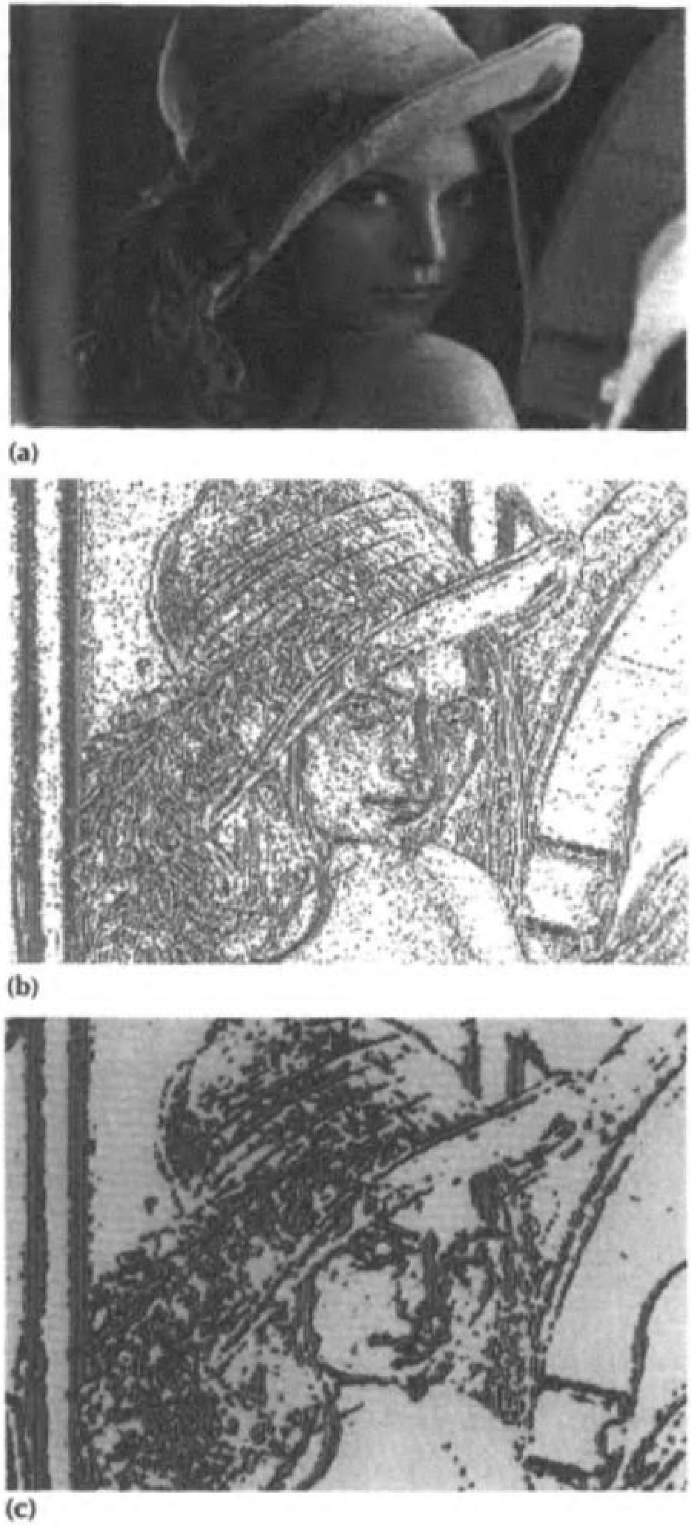


FIGURE 12.50 Edge detection by laplacian operator associated with local variance evaluation. (a) Original image; (b) processed with laplacian alone; (c) processed with laplacian operator taking the local variance into consideration.

$$\begin{array}{rrr} -1 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{array} \qquad \begin{array}{rrr} -1 & 2 & 1 \\ -1 & 2 & 1 \\ -1 & 2 & 1 \end{array} \qquad \begin{array}{rrr} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{array} \qquad \begin{array}{rrr} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{array}$$

(a) (b) (c) (d)

FIGURE 12.51 Compass gradient operators for line detection. (a) Responds to horizontal lines; (b) responds to vertical lines; (c) responds to 45°-oriented lines; (d) responds to 135 -oriented lines.

The angle between the pixel vector **x** with its projection onto the edge subspace is

$$\theta_{\text{edge}} = \cos^{-1} \frac{\left[\left(\sum_{i=1}^{4} \mathbf{W}_i^T \mathbf{x}\right)^2\right]^{1/2}}{|\mathbf{x}|} \tag{12.73}$$

and that between the vector **x** with its projection onto the line subspace is

$$\theta_{\text{line}} = \cos^{-1} \frac{\left[\left(\sum_{i=5}^{8} \mathbf{W}_i^T \mathbf{x}\right)^2\right]^{1/2}}{|\mathbf{x}|} \tag{12.74}$$

where $|\mathbf{x}| = [\sum_{i=1}^{8} (\mathbf{W}_i^T \mathbf{x})^2]^{1/2}$.

12.5 THINNING

Thinning is a necessary process in most pattern recognition problems because it offers a way to simplify the form for pattern analysis. In scanning an image, especially a text or drawing, high enough resolution is preferred to assure that no indispensable information is lost during digitization. In so doing, a width of more than two pixels will appear for each line. Thinning is the process to extract and apply additional constraints on the pixel elements that are to be preserved so that a linear structure of the input image will be recaptured without destroying its connectivity. See Figure 12.52 for the linear structure by medial axis transformation, which is covered in many books and is not discussed here.

A fast parallel algorithm for thinning digital patterns developed by Zhang and Suen (1984) is presented here, and application of their algorithm to various curved patterns is given. The same neighborhood notation used before for pixel

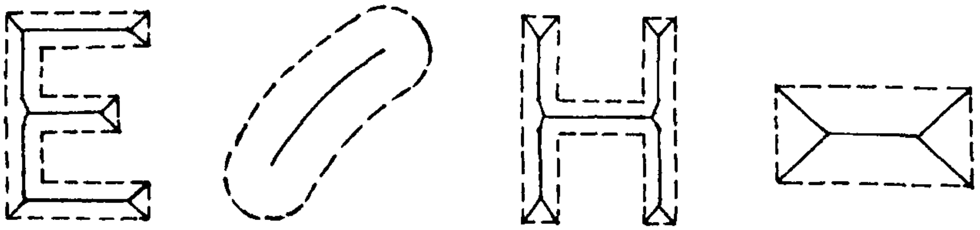


FIGURE 12.52 Linear structure of the silhouette by medial axis transformation.

$f(j,k)$ is redrawn in Figure 12.53. Let $N[f(j,k)]$ be the number of nonzero neighbors of $f(j,k)$, and $S[f(j,k)]$ is the number of $0 \rightarrow 1$ transitions in the ordered sequence $A_1, A_2, A_3, \ldots, A_7, A_0$. By following this definition, we have $N[f(j,k)] = 4$ and $S[f(j,k)] = 3$ for the window

$$\begin{array}{ccc} 0 & 0 & 1 \\ 1 & f(j,k) & 1 \\ 0 & 1 & 0 \end{array}$$

If the following conditions of *pass 1*,

$$\begin{array}{ll} (1) & 2 \leq N[f(j,k)] \leq 6 \\ (2) & S[f(j,k)] = 1 \\ (3) & A_1 \cdot A_3 \cdot A_5 = 0 \\ (4) & A_3 \cdot A_5 \cdot A_7 = 0 \end{array} \qquad (12.75)$$

are met, the point $f(j,k)$ is flagged for deletion; otherwise, it is not changed. Violation of condition (1) would take off the endpoint of the skeleton stroke, while violation of (2) would cause disconnection of segments of a skeleton. Satisfying conditions (3) and (4) as well as (1) and (2) means that they are a south boundary point or a northwest corner point in the boundary. From the point of view of thinning, they can be removed. Satisfying conditions (3′) and (4′) of

$$\begin{array}{ccc} A_0 & A_1 & A_2 \\ A_7 & f(j,k) & A_3 \\ A_6 & A_5 & A_4 \end{array}$$

FIGURE 12.53 Neighborhood of $f(j,k)$.

$$\begin{matrix}0&0&0\\0&f(j,k)&1\\0&1&1\end{matrix}\qquad\begin{matrix}1&1&1\\1&f(j,k)&1\\0&0&0\end{matrix}\qquad\begin{matrix}1&1&0\\1&f(j,k)&0\\0&0&0\end{matrix}$$

(a) **(b)** **(c)**

$$\begin{matrix}0&0&0\\1&f(j,k)&1\\1&1&1\end{matrix}\qquad\begin{matrix}0&1&1\\0&f(j,k)&1\\0&1&1\end{matrix}$$

(d) **(e)**

FIGURE 12.54 Point patterns deletable with the thinnıng process. (a) Northwest corner point; (b) south boundary poınt; (c) southeast corner point; (d) north boundary point; (e) west boundary point.

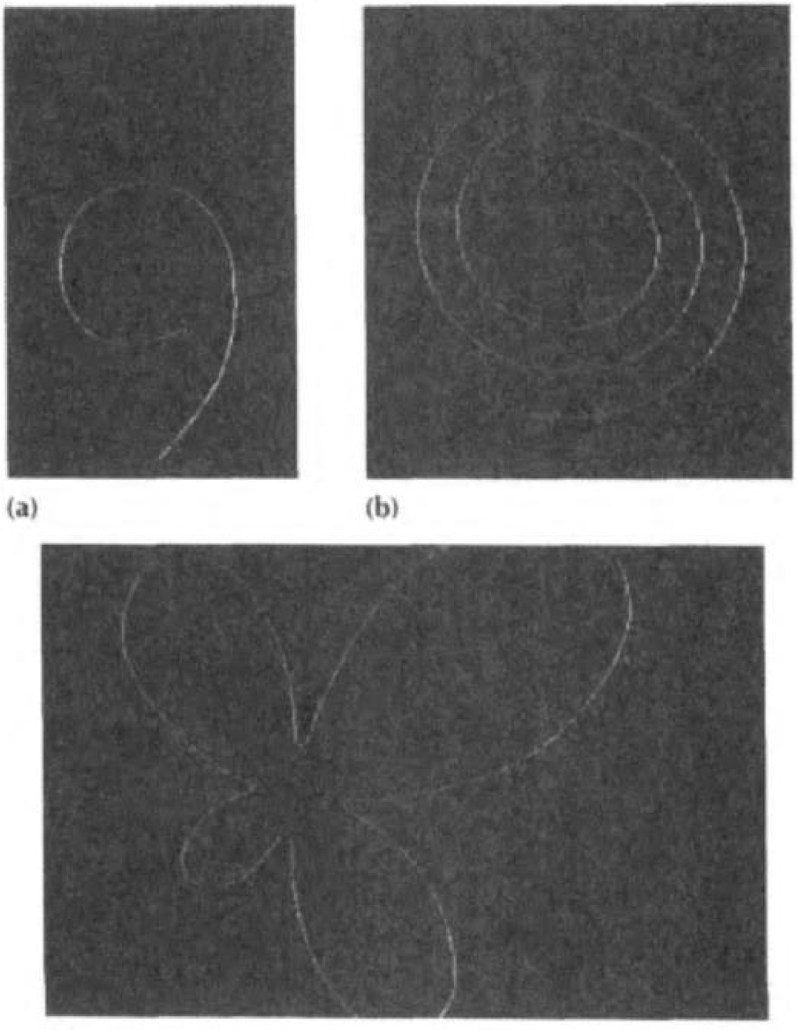


FIGURE 12.55 Some thinning results obtained using the algorithm suggested by Zhang and Suen. (a) Numeral "9" after thinning; (b) composite closed curve after thinning; (c) composite shape after the thinning process.

pass 2,

$$\begin{aligned}
&(1) \quad 2 \le N[f(j,k)] \le 6 \\
&(2) \quad S[f(j,k)] = 1 \\
&(3') \quad A_1 \cdot A_3 \cdot A_7 = 0 \\
&(4') \quad A_1 \cdot A_5 \cdot A_7 = 0
\end{aligned} \tag{12.76}$$

as well as (1) and (2) means that they are north or west boundary points or a southwest corner point. In either case $f(j,k)$ is not a part of the skeleton and should be removed. Figure 12.54 shows some cases in which conditions (1) to (4) imply. Figure 12.55 shows some thinning results by using this algorithm.

12.6 MORPHOLOGICAL PROCESSING

Morphological processing refers to an analysis of the geometrical structure within an image. Because the operations involved in the morphological processing relate directly to the shape, they prove to be more useful than convolution operations in industrial applications for defect identification. Morphological operations can be defined in terms of two basic operations, erosion and dilation.

12.6.1 Dilation

Dilation is a morphological transformation that combines two sets using vector addition of set elements. Suppose that object A and structuring element B are represented as two sets in two-dimensional Euclidean space. Then the dilation of A by B is defined as the set of all points c for which $c = a + b$:

$$\text{Dilation: } A \oplus B = \{c \mid c = a + b \text{ for some } a \in A \text{ and } b \in B\} \tag{12.77}$$

or

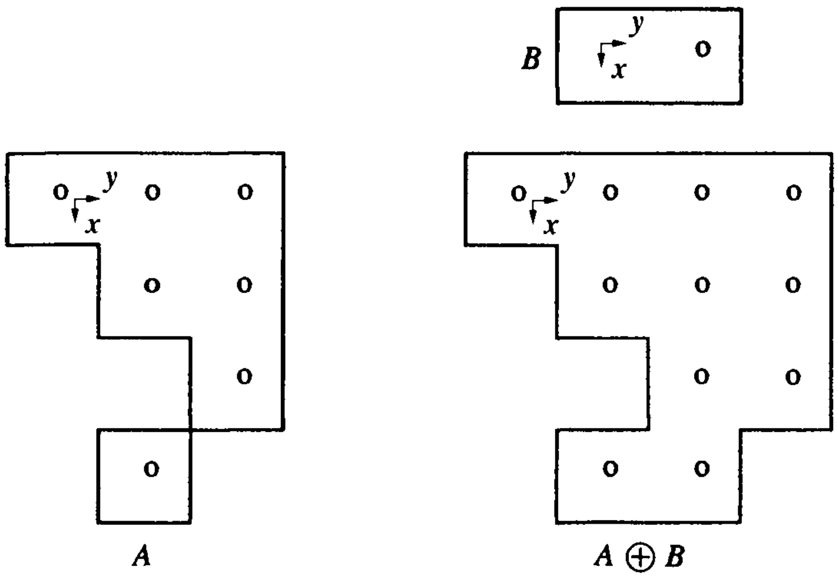
$$A \oplus B = \bigcup_{x \in B} a + x \tag{12.78}$$

where $A = \{a_1, a_2, \ldots, a_n\}$, and $B = \{b_1, b_2, \ldots, b_N\}$, and the operation symbol $\oplus$ denotes Minkowski addition.

To illustrate the dilation operation, consider the following examples.

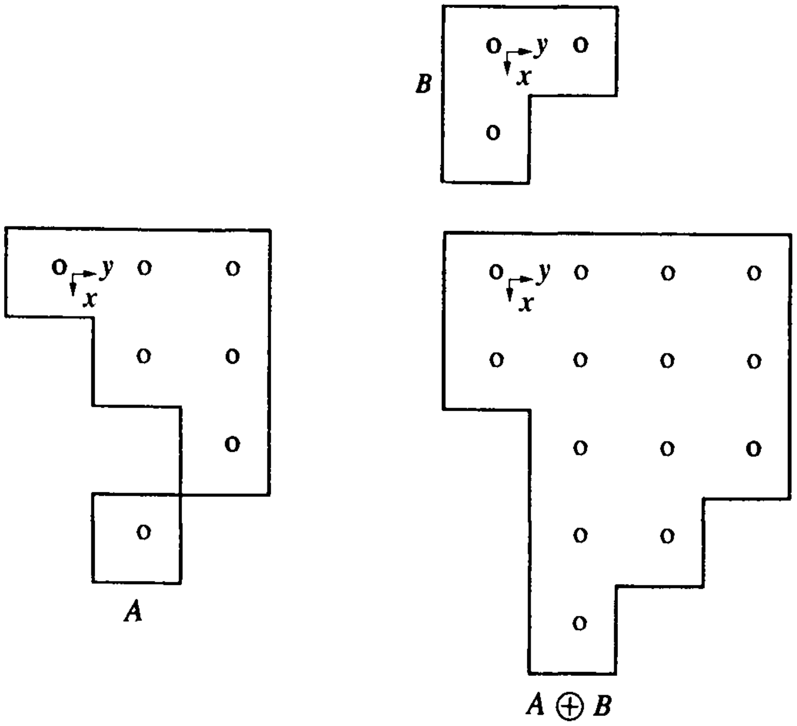
Example 1.

$$\begin{aligned}
A &= \{(0,0),(0,1),(0,2),(1,1),(1,2),(2,2),(3,1)\} \\
B &= \{(0,0),(0,1)\} \\
A \oplus B &= \{(0,0),(0,1),(0,2),(1,1),(1,2),(2,2),(3,1),(0,3),(1,3), \\
&\qquad (2,3),(3,2)\}
\end{aligned}$$



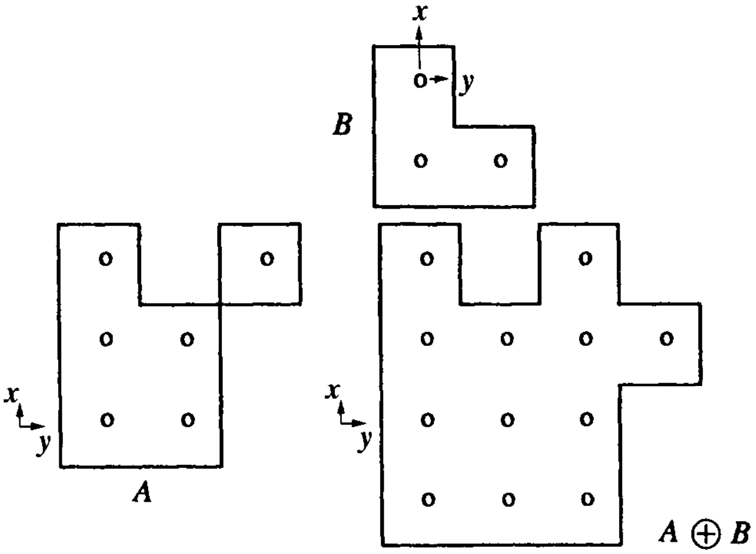
Example 2.

$$
\begin{aligned}
A &= \{(0,0),(0,1),(0,2),(1,1),(1,2),(2,2),(3,1)\}\\
B &= \{(0,0),(0,1),(1,0)\}\\
A \oplus B &= \{(0,0),(0,1),(0,2),(0,3),(1,0),(1,1),(1,2),\\
&\qquad (1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(4,1)\}
\end{aligned}
$$



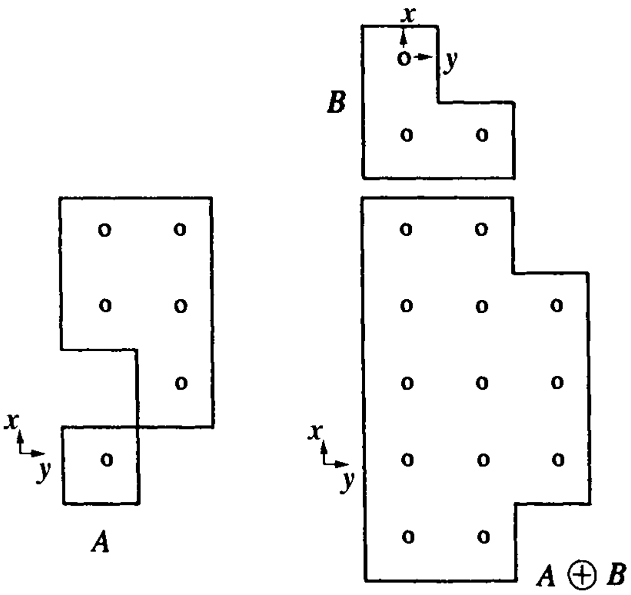
Example 3.

$A = \{(0, 1), (0, 2), (1, 1), (1, 2), (2, 1), (2, 3)\}$
$B = \{(0, 0), (-1, 0), (-1, 1)\}$
$A \oplus B = (0, 1), (0, 2), (0, 3), (1, 1), (1, 2), (1, 3), (2, 1),$
$(2, 3), (1, 4), (-1, 1), (-1, 2), (-1, 3)\}$



Example 4.

$A = \{(0, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2)\}$
$B = \{(0, 0), (-1, 0), (-1, 1)\}$



$$A \oplus B = \{(0,1),(1,2),(2,1),(2,2),(3,1),(3,2),(-1,1),(0,2),(1,1),$$
$$(-1,2),(0,3),(1,3),(2,3)\}$$

12.6.2 Erosion

Similarly, erosion is a morphological transformation that combines two sets using vector subtraction of the set elements. Suppose that object A and structuring element B are represented as two sets in two-dimensional euclidean space. Then the erosion of A by B is defined as the set of all points c for which $c + b \in A$ for every $b \in B$. That is,

$$A \ominus B = \{c | c + b \in A \text{ for every } b \in B\} \tag{12.79}$$

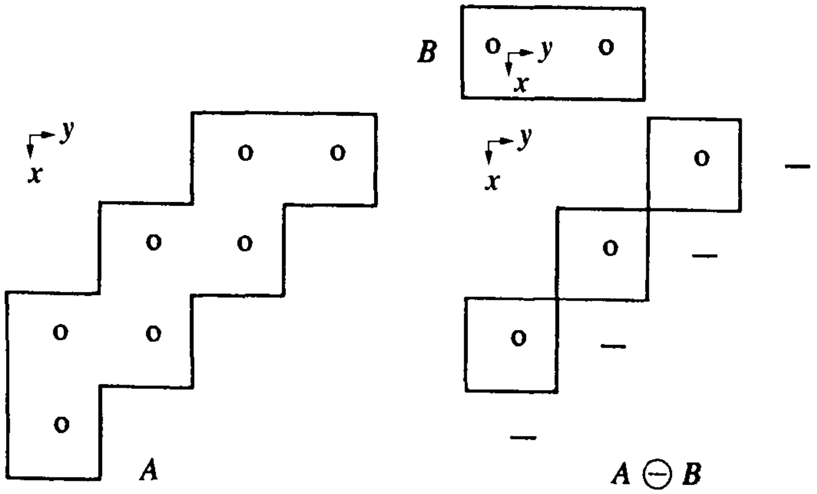
or

$$A \ominus B = \left\{c | c = a - b \text{ for every } b \in B\right\} \tag{12.80}$$

where $A = \{a_1, a_2, \ldots, a_n\}$, $B = \{b_1, b_2, \ldots, b_N\}$, and the operation symbol $\ominus$ denotes Minkowski subtraction. Let us take an example to illustrate the erosion operation.

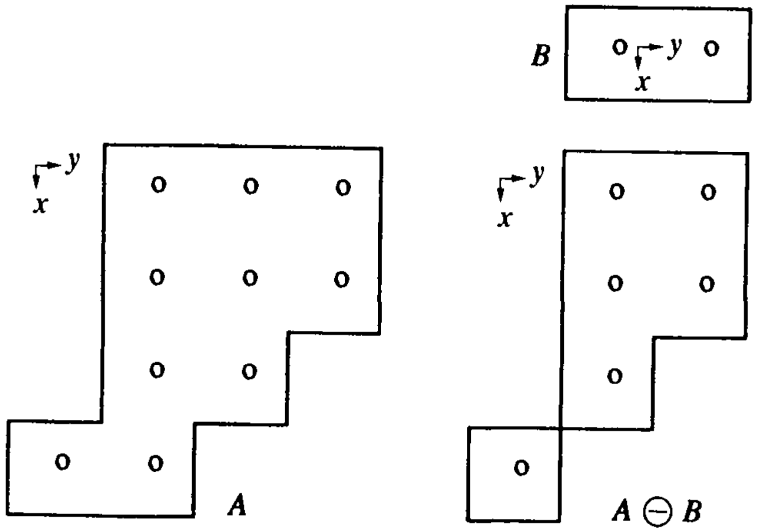
Example 1.

$A = \{(0,2),(0,3),(1,1),(1,2),(2,0),(2,1),(3,0)\}$
$B = \{(0,0),(0,1)\}$
$A \ominus B = \{(0,2),(1,1),(2,0)\}$
Note: The dashes are deleted by erosion.



Example 2.

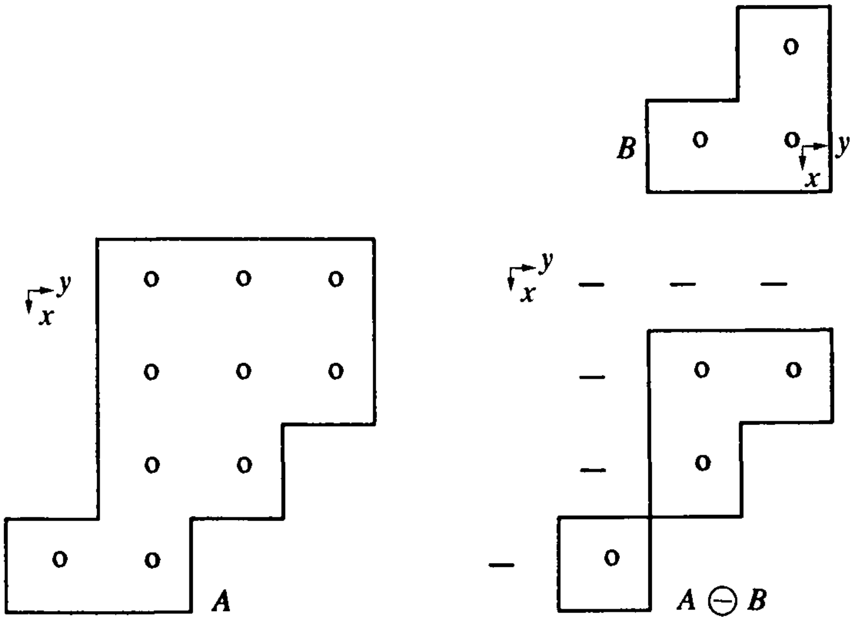
$A = \{(0, 1), (0, 2), (0, 3), (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (3, 0), (3, 1)\}$
$B = \{(0, 0), (0, 1)\}$



$A \ominus B = \{(0, 1), (0, 2), (1, 1), (1, 2), (2, 1), (3, 0)\}$

Example 3.

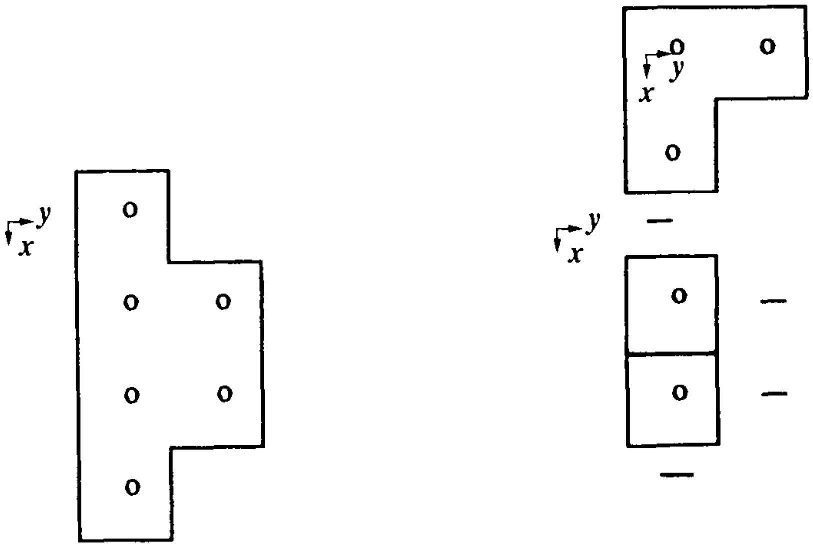
$A = \{(0, 1), (0, 2), (0, 3), (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (3, 0), (3, 1)\}$
$B = \{(0, 0), (0, -1), (-1, 0)\}$



$A \ominus B = \{(1,2), (1,3), (2,2), (3,1)\}$

Example 4.

$A = \{(0,1), (1,1), (1,2), (2,1), (2,2), (3,1)\}$
$B = \{(0,0), (0,1), (1,0)\}$
$A \ominus B = \{(1,1), (2,1)\}$



From the examples above it can be seen that dilation by a structuring element corresponds to swelling or expansion operation on the image. A notch in an image will be filled by this operation. When an image is dilated with a 3 × 3 structuring element, the operation is equivalent to a neighborhood operation. By contrast, the erosion operation provides a shrinking effect, and blobs will be eliminated.

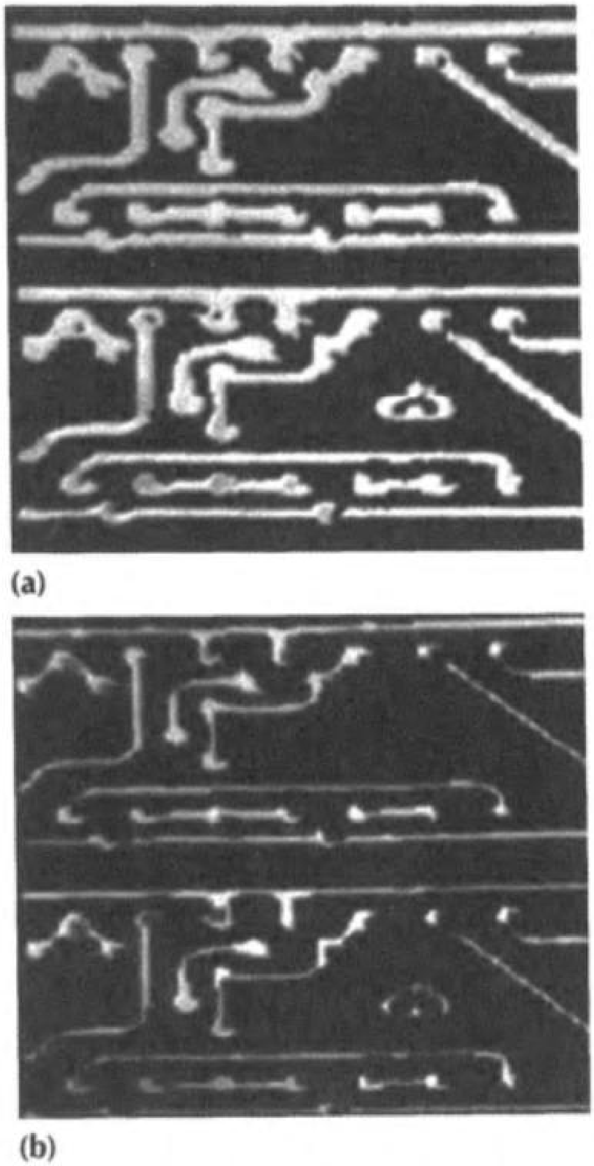
Frequently, these two operations (dilation and erosion) are used in pairs, either dilation of an image followed by erosion of the dilated result, or erosion of an image followed by dilation of the eroded result. By so doing we can eliminate the specific image details that are smaller than the structuring element (e.g., gaps, notches, blobs, etc.) while preserving the main geometric shape of the image.

An erosion operation on an image followed by a dilation operation on the eroded image is called opening and is defined as

$$\mathcal{C}(A, B) = \mathcal{D}[\mathcal{E}(A, B), B] \tag{12.81}$$

The opening operation will have a smoothing effect on the image. It can be used to smooth contours, suppress small islands, and sharpen caps of the image A. When dilation operation on the image is followed by an erosion on the dilated result, it is called closing and is defined as

$$\mathscr{C}(A, B) = \mathscr{E}[\mathscr{D}(A, B), B] \tag{12.82}$$



(a)

(b)

FIGURE 12.56 Line drawing after morphological processing. (a) Original drawing; (b) after morphological processing.

The closing operation can be used to block up narrow channels and thin lakes. It is ideal for the study of interobject distance. Figure 12.56 shows some results on a line drawing after the application of morphological processing.

12.7 BOUNDARY DETECTION AND CONTOUR TRACING

12.7.1 Boundary Extraction

A boundary can be viewed as a path formed by linking the edge elements together. After being linked together, edge pixels will give more meaningful information that can characterize the shape of an object and its geometric features, such as size and orientation. Therefore, an edge detection algorithm is frequently followed by an edge linking algorithm. Two edge pixels can be linked together if they are (1) very close to each other (i.e., in the 3 × 3 neighborhood), and (2) similar in the strength of response to a gradient operator and the direction of their gradient vectors.

Connectivity

Three types of connectivity are considered: 4-, 8-, and m-connectivity.

4-connectivity. For a pixel $p(x, y)$, where x and y are spatial coordinates, the set of pixel points with coordinates

$$(x+1, y), \quad (x-1, y), \quad (x, y+1), \quad (x, y-1) \tag{12.83}$$

is said to consist of the four 4-neighbors of the point $p(x, y)$ and is denoted by $N_4(p)$.

8-connectivity. For the pixel point $p(x, y)$ the set of pixel points with coordinates

$$(x+1, y+1), \quad (x+1, y-1), \quad (x-1, y+1), \quad (x-1, y-1) \tag{12.84}$$

is said to consist of the four diagonal neighbors of $p(x, y)$ and is denoted by $N_D(p)$. $N_8(p)$, the 8-neighbors, are defined as

$$N_8(p) = N_4(p) + N_D(p) \tag{12.85}$$

$N_8(p)$ is a set of the eight 8-neighbors of $p(x, y)$.

m-connectivity. Introduction of this connectivity is necessary to eliminate the multiple path when both a 4-connected and an 8-connected neighbor

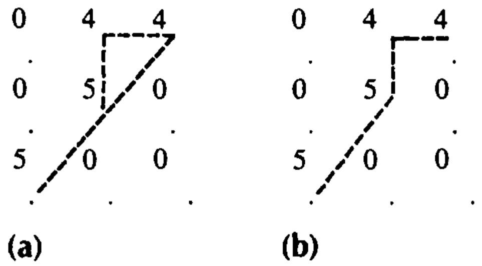


FIGURE 12.57 Linking of edge pixels. (a) Multiple-path connection resulted when 8-connectivity is used; (b) multiple-path connection eliminated when *m*-connectivity is used.

appear in the situation shown in Figure 12.57. Two edge pixels are said to be *m*-connected if either of the following conditions is satisfied:

$$\begin{array}{ll}(1) & q \text{ is in } N_4(p) \\ (2) & q \text{ is in } N_D(p) \text{ and the set } N_4(p) \cap N_r(q) \text{ is empty.}\end{array} \qquad (12.86)$$

12.7.2 Contour Tracing

In contour tracing we try to trace the boundary by properly ordering successive edge points. Many algorithms have been suggested for contour tracing. One suggested by Pavlidis (1982), implemented in our laboratory with some modification, works very well. An experimental result is given here for a miniature spring (see Figure 12.58). This algorithm can be described briefly as follows:

1. Find an initial pixel on the boundary by scanning the image from top to bottom and from left to right.
2. Once the initial pixel is found, we select the rightmost pixel among the successive pixels that belong to the neighborhood set.
3. Continue tracing until the current pixel is the same as the initial pixel.

Refer to Figure 12.59 for the tracing direction notation. First search the point along direction $P \rightarrow 5$. If the point at that location is in R, the contour set, set the current point at this location, and the next search direction will be westward as $P \rightarrow 4$. If the point at search direction $P \rightarrow 5$ is not in R, search the point along $P \rightarrow 6$ direction. If it is in R, set C to this point. Otherwise, search along direction $P \rightarrow 7$ of the current point C (i.e., the point P). If it is in R, the next point is found at the $P \rightarrow 7$ direction. If none of these is tue, change the search direction to $P \rightarrow 0$.

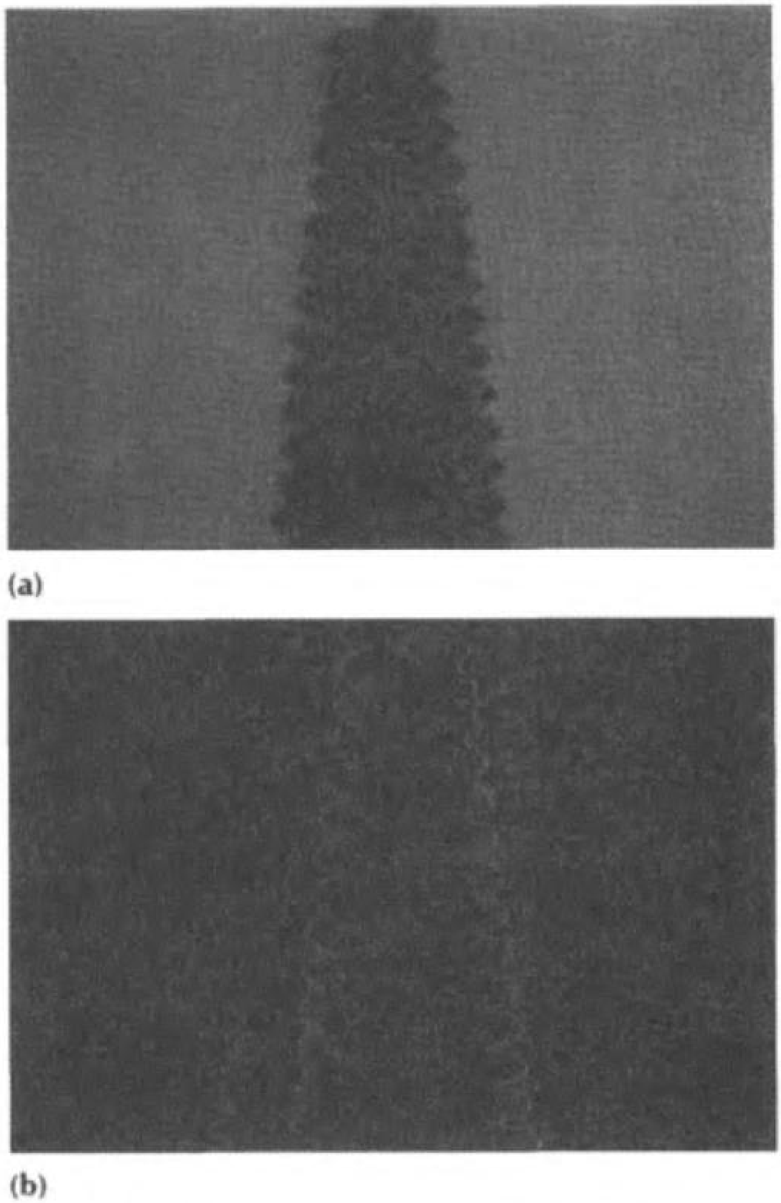


FIGURE 12.58 Some contour tracing results obtained using the method suggested by Pavlidis. (a) Original image; (b) contour of the object.

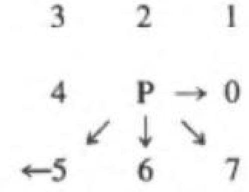


FIGURE 12.59 Notatıon used for contour tracıng algorıthm.

12.7.3 Global Analysis of the Boundary via Hough Transform

In this section global analysis via Hough transform will be discussed for the extraction and fitting of geometric shapes from a set of extracted image points. The idea behind using the Hough transform technique for geometric recognition is simple. It maps a straight line $y = ax + b$ in a cartesian coordinate system into a single point in the (ρ, θ) plane, or

$$\rho = x \cos \theta + y \sin \theta \tag{12.87}$$

For a point (x, y) in the cartesian coordinate plane, there will be an infinite number of curves in the (ρ, θ) plane (Figure 12.60). When two points (x_i, y_i) and (x_j, y_j) lie on the same straight line, the curves in the (ρ, θ) plane which correspond, respectively, to the two points (x_i, y_i) and (x_j, y_j) in the cartesian plane will intersect at a point. This intersection point determines the parameters of

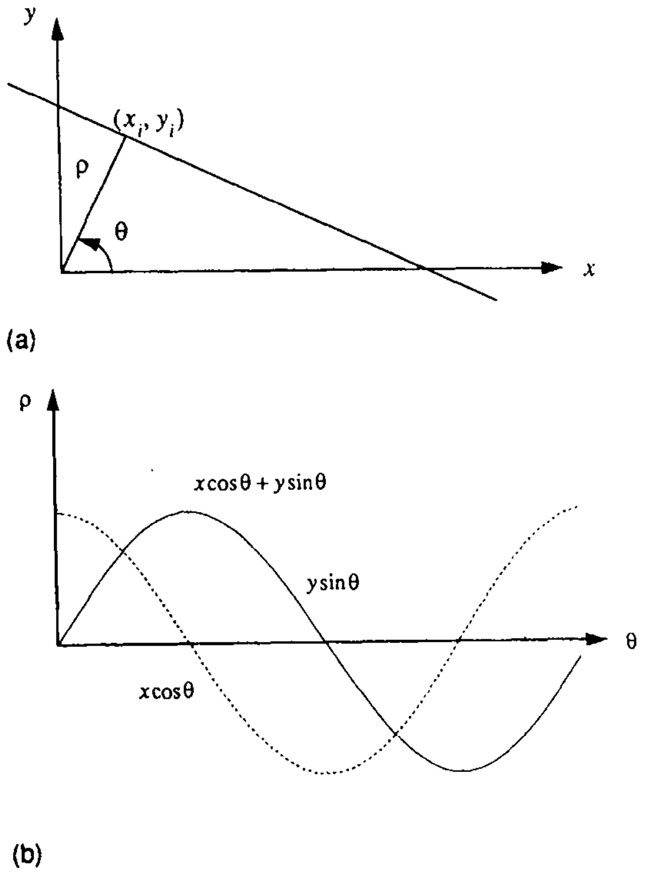


FIGURE 12.60 Hough transform. (a) Cartesian xy coordinate system; (b) (ρ, θ) parametric plane.

the line that joins these two points. Similar arguments apply to three collinear points. This property, which exists between the cartesian plane and the (ρ, θ) plane (or the parametric plane as it is usually called) will be useful in finding the line that fits points in the xy plane. In short, the Hough transform approach is to find the point of intersection of the $\rho\theta$ curves, each of which corresponds to a line in the cartesian xy plane.

Global analysis via Hough transform can be formulated as consisting of discretizing two-dimensional parameter space, which is (ρ, θ) space in our discussion, into finite intervals (called accumulator cells or two-dimensional bins), as illustrated in Figure 12.61, where $(\rho_{\max}, \rho_{\min})$ and $(\theta_{\max}, \theta_{\min})$ are the expected ranges of the values of ρ and θ. Each of these cells in the (ρ, θ) space is first initialized to zero. For each point (x_k, y_k) in the cartesian image plane, do the following computation. Let ρ equal each of the subdivision values (say, ρ_i) on the ρ axis and then solve for the corresponding θ_i. If there are N collinear points lying on a line,

$$\rho_i = x \cos \theta_i + y \sin \theta_i \tag{12.88}$$

there will be N sinusoidal curves that intersect at (ρ_i, θ_i) in the parametric space. Resulting peaks in the (ρ, θ) accumulator array therefore give strong evidence of the existence of lines in the image. The Hough transform can be generalized to detect circles as described by

$$(x - a)^2 + (y - b)^2 = r^2 \tag{12.89}$$

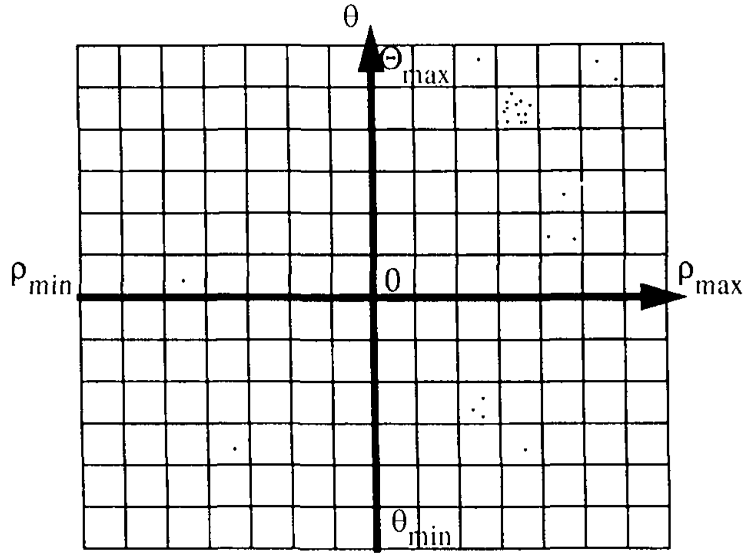


FIGURE 12.61 Parameter plane for use in Hough transform. *Note:* Number of points on the same line equals the count number for the (ρ, θ) count.

We now have three (instead of two) parameters, a, b, and r, to parametrize a circle. This results in a three-dimensional accumulator array, where each accumulator bin is indexed by a discrete value of (a, b, r).

12.7.4 Boundary Detection by a Sequential Learning Approach

This approach applies to the case where very slow spatial variations of certain importance occur with respect to figure-background contrast. These types of edges are often encountered in digital radiography. This method can be described briefly as follows: Use the information of those pixels that have been classified as on the boundary to do bayesian updating of the information on the successive pixels for class categorization ("figure" or "background"). This updating allows the system to "learn" the slow variations in the background level as well as on the gradient near the boundary of the figure region.

Denote the background region with the gray level as R_1 and the figure region as R_2. In both regions the gray level increases (or decreases) linearly with distance from the edge. In Figure 12.62, B and B^* are, respectively, the last pixel and the pixel preceding B which have already been classified as points on the boundary as shown. Note the difference in the numbering of the pixels neighboring B in the figure. The arrows B^*B show the direction of boundary tracing. If there is only one change, in the row or in the column, we follow the arrow shown in Figure 12.62a. Otherwise (i.e., there are changes in both row and column), we follow the tracing direction shown in Figure 12.62b. Define a loss function

$$L(P_j|P_i)$$

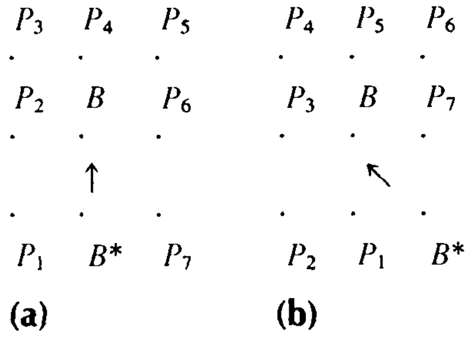


FIGURE 12.62 Numbering of the pixels which are neighboring to B.

as the loss (cost or penalty) due to mischoice of P_j as the next boundary pixel when it should be P_i. The conditional average loss or conditional average risk may be defined as

$$r_j(\mathbf{x}) = \sum_{i=1}^{7} L(P_j|P_i)p(P_i|\mathbf{x}) \tag{12.90}$$

that is, the average or expected loss of mischoosing $\mathbf{x}$ as P_j, when in fact it should be some other pixel, $P_i, i = 1, 2, \ldots, 7, i \neq j$. $\mathbf{x}$ is the vector with components $x_1, x_2, \ldots, x_7$, and $p(P_i|\mathbf{x})$ is the a posteriori probability of the next boundary pixel being P_i. By Bayes' theorem, Eq. (12.90) becomes

$$r_j(\mathbf{x}) = \sum_{i=1}^{7} L(P_j|P_i)\frac{p(\mathbf{x}|P_i)p(P_i)}{p(\mathbf{x})} \tag{12.91}$$

If the loss function is chosen so that

$$L(P_j|P_i) = \begin{cases} 0 & \text{for } j = i \\ 1 & j \neq i \end{cases} \tag{12.92}$$

we have

$$r_j(\mathbf{x}) = \sum \frac{p(P_i)p(\mathbf{x}|P_i)}{p(\mathbf{x})} \tag{12.93}$$

where $p(P_i)$ is the a priori probability of the next boundary pixel being P_i. $p(\mathbf{x}|P_i)$ is the likelihood function for $\mathbf{x}$ given that the P_i is the next boundary pixel, and $p(\mathbf{x}) = \sum_i p(\mathbf{x}|P_i)p(P_i), i = 1, 2, \ldots, 7$ is the probability that $\mathbf{x}$ occurs without regard to whether or not it is the correct next boundary pixel. Our job becomes to find an optimal decision that will minimize the average risk (or cost). Obviously, we will choose P_i if

$$r_i \leq r_j \qquad \forall j \tag{12.94}$$

or

$$p(P_j)p(\mathbf{x}|P_j) \leq p(P_i)p(\mathbf{x}|P_i) \qquad \forall j \tag{12.95}$$

Using normal density function for analysis.

$$p(\mathbf{x}|P_i) = \frac{1}{\sqrt{2\pi c}}\exp\left[-\frac{1}{2}(\mathbf{x} - \mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x} - \mathbf{m})\right] \tag{12.96}$$

where m is the mean and $\mathbf{C} = \sigma^2$ is the variance; or

$$p(\mathbf{x}|P_i) = \frac{1}{\sqrt{2\pi}\sigma_{vi}}\exp\left[\frac{-(x_v - m_{vi})^2}{2\sigma_{vi}^2}\right] \tag{12.97}$$

Taking the logarithm of the $p(\mathbf{x}|P_i)$ and denoting it by φ_i, it follows that we will choose P_i if

$$\varphi_i < \varphi_j \qquad \forall j \tag{12.98}$$

where

$$\varphi_i = b_i - \sum_{v=1}^{7} \left[\ln \sigma_{vi} + \frac{(x_v - m_{vi})^2}{2\sigma_{vi}^2} \right] \tag{12.99}$$

Let the direction of tracing be anticlockwise, thus leaving the background R_1 on its left. If the boundary pixel following B is P_i, the pixel

$$P_v \in R_1 \qquad \text{for } v \leq i \tag{12.100}$$

$$P_v \in R_2 \qquad \text{for } v > i \tag{12.101}$$

φ_i in (12.99) can then be decomposed as

$$\varphi_i = b_i - \sum_{v=1}^{i} \left[\ln \sqrt{C_b} + \frac{(x_v - m_b)^2}{2C_b} \right] - \sum_{v=i+1}^{7} \left[\ln \sigma_{vi} + \frac{(x_v - m_{vi})^2}{2\sigma_{vi}^2} \right] \tag{12.102}$$

where m_b and C_b are, respectively, the mean and variance in the background region, R_1, and are usually known; m_{vi} and σ_{vi}^2 are the mean and variance in the figure region, R_2, which need to be evaluated for updating of the information used to decide the next boundary pixel.

Figure 12.63 shows the process of updating the class parameters after finding the boundary pixel P_i. P_{i-1} and P_{i-2} are the two boundary pixels found previously. Q_2, Q_3, and Q_1 are the pixels needed for updating the class parameters. Readers may refer to (Ghalli, 1988) for details on computation of m_{vi} and σ_{vi}. Figures 12.64 and 12.65 show the results of boundary detection by this method on a radiograph of the hand and an angiograph of the head, respectively.

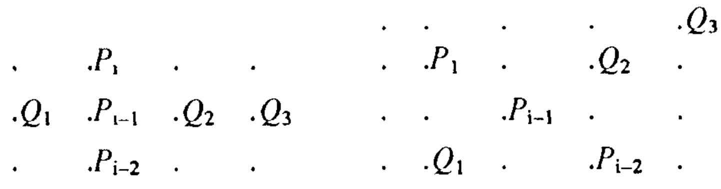


FIGURE 12.63 Updating of the class parameters m_{vi} and σ_{vi}.

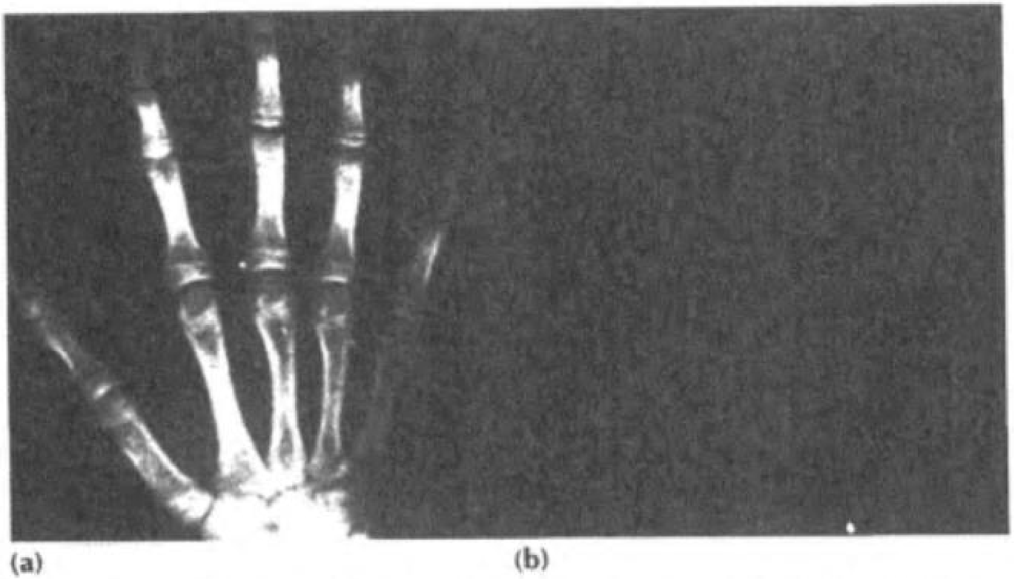


FIGURE 12.64 Results of boundary detection by the sequential method. (a) Radiograph of a hand; (b) boundaries obtained. (From Ghalli, 1988.)

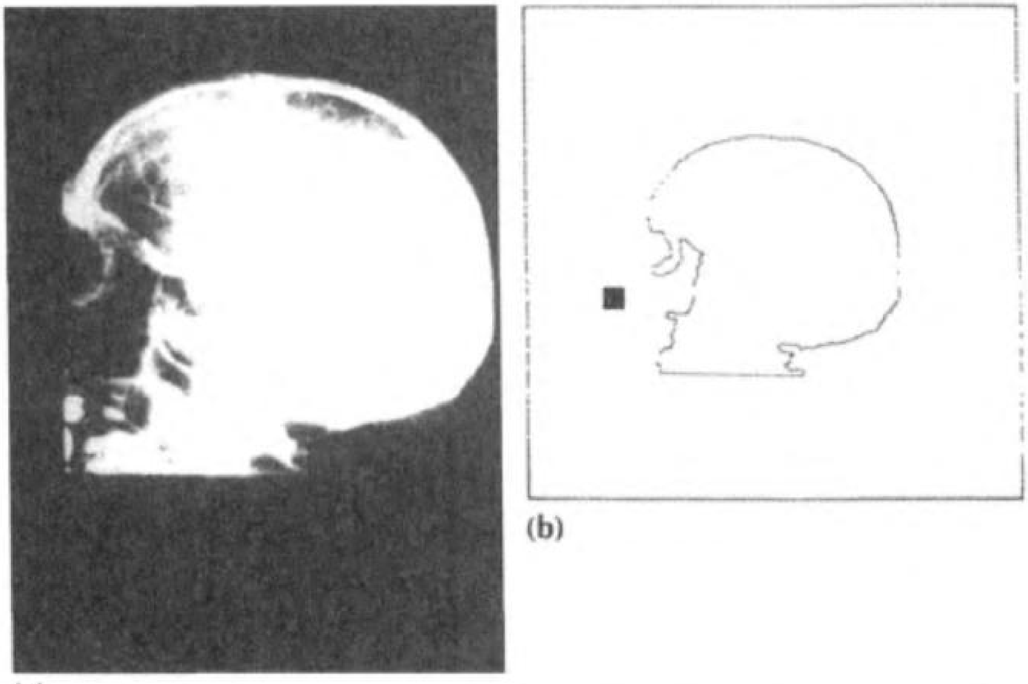


FIGURE 12.65 Boundary detection by the sequential learning method. (a) Angiograph of a head; (b) boundaries obtained. (From Ghalli, 1988.)

12.8 TEXTURE AND OBJECT EXTRACTION FROM TEXTURAL BACKGROUND

Texture analysis is one of the most important techniques used in the analysis and classification of images where repetition or quasi-repetition of fundamental image elements occurs. Such characteristics can easily be seen from remote sensing images obtained from an aircraft/satellite platform to images of cell cultures or tissue samples through microscope. So far, there is no precise definition of texture. It is evaluated qualitatively by one or more of the properties of coarseness, smoothness, granulation, randomness, and regularity. Nevertheless, the tonal primitive property and the spatial organization of the tonal primitives characterize a texture fairly well. There are three principal approaches to the texture description of a region: statistical, spectral, and structural.

Among the statistical approaches are autocorrelation functions, textural edgeness, structural elements, spatial gray-tone cooccurrence probabilities, gray-tone run lengths, and autoregressive models. Statistical approaches characterize the textures as smooth, coarse, grainy, and so on.

Spectral techniques (optical transform and digital transform) are based on properties of the Fourier spectrum. The image is analyzed globally by identifying the percentage energy of the peak. Calculation of the discrete Laplacian at the peak, area of the peak, angle of the peak, squared distance of the peak from the origin, and angle between the two highest peaks are involved.

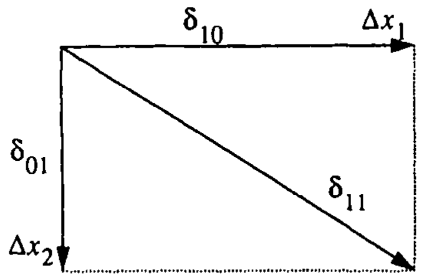
Structural approaches deal with the primitives and their spatial relationships. A primitive is usually defined as a connected set of cells characterized by attributes. The attributes may be gray tone, shape of the connected region, and/or the homogeneity of the local property.

"Spatial relationships" refers to adjacency of primitives, closest distance within an angular window, and so on. According to the spatial interaction between primitives, textures can be categorized as weak textures or strong textures. To distinguish between these two textures, the frequency with which the primitives cooccur in a specified spatial relationship is a good measure. Some investigators suggested using the number of edges per unit area (or edge density) for a texture measure. Others suggested using the gray-level run lengths as primitive, or using the number of extrema per unit area (extreme density) for a measure of texture.

12.8.1 Extraction of theTexture Features

As mentioned in previous paragraphs, many feature extraction methods have been proposed. One of these (called cooccurrence or gray-tone spatial dependence) will be discussed in more detail. This method considers not only the distribution of intensities, but also the position of pixels with equal or nearly equal intensity values. This cooccurrence matrix evolves from the joint probability density function of two pixel locations and is a second-order statistical measure of image intensity variation. As we will see later, it provides the basis for a number of textural features.

If we define the position operator P as follows:



we obtain the 3 × 3 matrices shown in Figure 12.67 for the sample images shown on Figure 12.66. Note that the size of the cooccurrence matrices is determined strictly by the number of distinct gray levels in the input image. If every element in the matrix is divided by the total number of point pairs in the image that satisfy the position operator δ_{10}, δ_{01}, or δ_{11} as indicated, a new matrix H (called the gray-level cooccurrence matrix) is formed, with the element h_{ij} as the estimate of the joint probability that a pair of points satisfying the position operator will have values (z_i, z_j). By choosing an appropriate position operator, it is possible to detect the presence of a given texture pattern. Nevertheless, this cooccurrence matrix H does not directly provide a single feature that may be used for texture discrimination.

0	0	0	0	1
0	0	0	1	1
0	0	1	1	2
0	1	1	2	2
1	1	2	2	2

(a)

1	0	0	0	0
1	1	0	0	0
2	1	1	0	0
2	2	1	1	0
2	2	2	1	1

(b)

0	1	2	1	0
0	1	2	1	0
0	1	2	1	0
0	1	2	1	0
0	1	2	1	0

(c)

1	2	1	2	1
1	2	1	2	1
1	2	1	2	1
1	2	1	2	1
1	2	1	2	1

(d)

FIGURE 12.66 Sample images with three gray levels, $z_0 = 0, z_1 = 1$, and $z_2 = 2$.

10	0	1	2
0	6	4	0
1	0	4	3
2	0	0	3

(a)

10	0	1	2
0	6	0	0
1	4	4	0
2	0	3	3

(b)

10	0	1	2
0	0	5	0
1	5	0	5
2	0	5	0

(c)

10	0	1	2
0	0	0	0
1	0	0	10
2	0	10	0

(d)

01	0	1	2
0	6	4	0
1	0	4	3
2	0	0	3

(e)

01	0	1	2
0	6	4	0
1	0	4	3
2	0	0	3

(f)

01	0	1	2
0	8	0	0
1	0	8	0
2	0	0	4

(g)

01	0	1	2
0	0	0	0
1	0	12	0
2	0	0	8

(h)

11	0	1	2
0	3	7	0
1	0	0	5
2	0	0	1

(i)

11	0	1	2
0	6	0	0
1	0	7	0
2	0	0	3

(j)

11	0	1	2
0	0	4	0
1	4	0	4
2	0	4	0

(k)

11	0	1	2
0	0	0	0
1	0	0	8
2	0	8	0

(l)

FIGURE 12.67 Sample cooccurrence matrices corresponding to textures shown in Figure 12.66 with position operators δ_{10}, δ_{01}, and δ_{11}. (a) For the image shown in Figure 12.66a; (b) for Figure 12.66b; (c) for Figure 12.66c; (d) for Figure 12.66d; (e) for Figure 12.66a; (f) for Figure 12.66b; (g) for Figure 12.66c; (h) for Figure 12.66d; (i) for Figure 12.66a; (j) for Figure 12.66b; (k) for Figure 12.66c; (l) for Fig. 12.66d.

A set of descriptors have been proposed by Haralick (1978) to be derived from the gray-level cooccurrence matrix as textural features. They include:

1. Uniformity: $$\sum_i \sum_j h_{ij}^2 \qquad (12.103)$$
2. Entropy: $$-\sum_i \sum_j h_{ij} \log h_{ij} \qquad (12.104)$$
3. Maximum probability: $$\operatorname*{Max}_{i,j} h_{ij} \qquad (12.105)$$

4. Contrast: $$\sum_i \sum_j (i-j)^2 h_{ij} \tag{12.106}$$

5. Inverse difference moment: $$\sum_i \sum_j \frac{h_{ij}}{(i-j)^2} \tag{12.107}$$

12.8.2 Segmentation of Textural Images

Segmentation of an image into homogeneous regions is one of the many intriguing topics in image processing. "Homogeneous" refers to the uniformity in some property, such as intensity or gray level, color, or texture. In most applications an image is segmented by the intensity criterion. But for some applications, segmentation by intensity criterion does not give satisfactory results, since an image such as a human portrait or outdoor scene gives a nonuniform intensity region but a homogeneous texture region.

In this section we discuss textural image segmentation without supervision. That is, no a priori knowledge or operator guidance is available, and no pixel classification based on feature statistics gathered over training areas can be applied. In what follows, a method using cooccurrence matrices as features for image segmentation is discussed. They randomly take 40 $N \times N$ subimages from the 256×256 original image. The value N is to be determined experimentally. From these subimages they computed cooccurrence matrices of size $G \times G$, where G is the number of gray levels in the image. Then divide the cooccurrence matrix into n equally sized squares and use the average value of the matrix elements in each square to form one component of the feature vector, such that a multidimensional feature space R^n is formed with the feature vector represented as

$$\mathbf{x} = \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix} \tag{12.108}$$

These feature vectors will cluster in the feature space if different textures appear in the image. The feature space R^n will then be partitioned into subspaces as $L_1, L_2, \ldots, L_k$, each corresponding to a texture class $1, 2, \ldots, k$. Parkkinen and Oja use rules similar to the discriminant functions discussed in Chapters 3 to 5 for the classification of texture images:

$$\text{If } \mathrm{DF}_i > \mathrm{DF}_j, \forall j \text{ not equal to } i, \text{ then } \mathbf{x} \in \omega_i \tag{12.109}$$

where DF stands for the discriminant function.

Worthy of mention is a work by Davis and Tychon (1986). In their research they used two statistical measures to generate a second-order picture from the original textural image. Then they processed the second-order image to produce boundaries between adjacent texture regions. The first measure they used is to determine the frequency characteristics pertaining to the texture. The more the rise and fall of the picture element values, the finer the texture. The second measure is to capture information on the contrast in a texture. This measure is defined as the means of the absolute differences between adjacent points:

$$C = \sum \frac{|d(i)|}{n-1} \qquad i = 1, 2, \ldots, n \tag{12.110}$$

where $d(i)$ is the gray-level difference of the adjacent pixels and is

$$d(i) = p(i) - p(i+1) \qquad i = 1, 2, \ldots, n \tag{12.111}$$

and $p(i)$ is the gray level of the point i. A high-contrast picture will have larger differences in gray levels and therefore produce a higher value for C.

Four vectors about each point in a textural scene in four orientations [0° (horizontal), 45° (right diagonal), 90° (vertical), and 135° (left diagonal) are to be analyzed with these two measures, and eight values are produced for each point in the textural image for use in the construction of its second-order picture. When

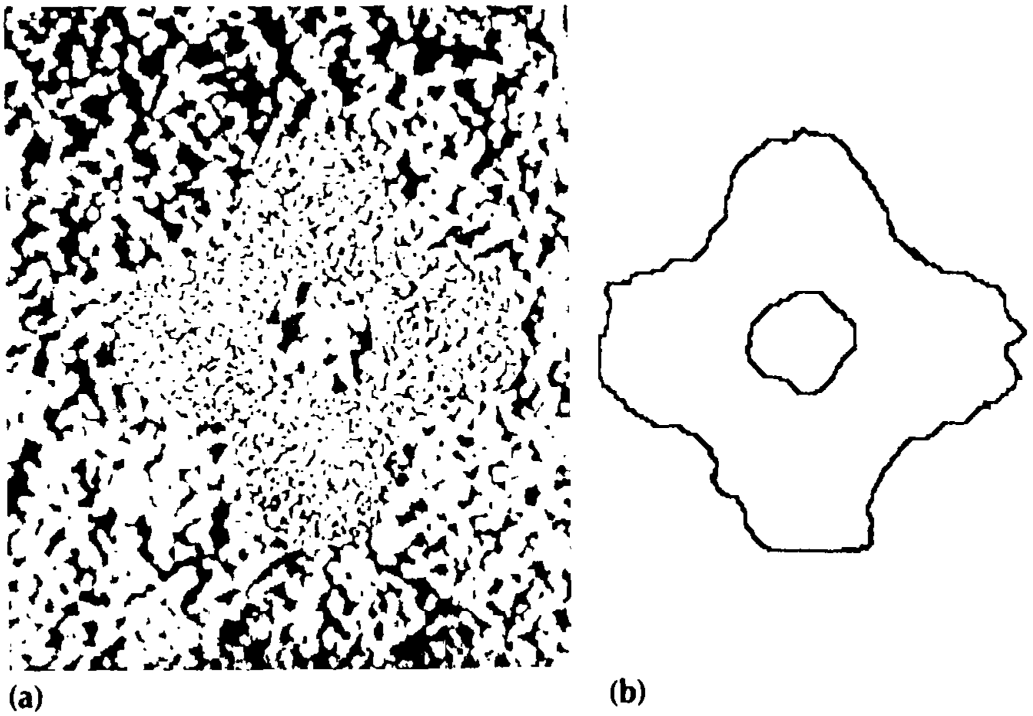


FIGURE 12.68 Textural image for segmentation. (a) Test image consisting of two different textures; (b) boundaries between these two textures extracted. (From Davies and Tychon, 1986.)

more than one binary edge picture has been created, they are OR'ed together to produce a resultant edge picture. Figure 12.68a shows a test image containing two different textures. The boundaries between the two different textures extracted by the method are shown in Figure 12.68b.

PROBLEMS

12.1 Write a program to perform the linear contrast stretching gray-level transformation shown in Figure P12.1. Take any picture; scan and digitize it to obtain a 512 × 512 image. With that image as a large data set, enhance it by a suitable contrast stretching transformation. Evaluate this linear contrast stretching algorithm by comparing the processed image with the original picture.

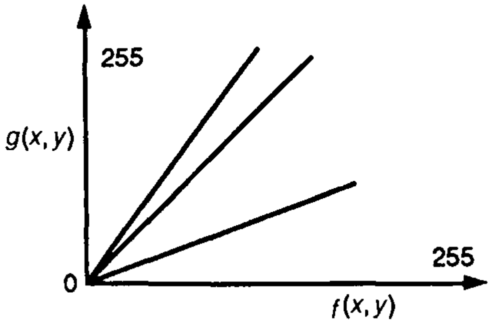


FIGURE P12.1

12.2 Write a program to perform the brightness stretching on the midregion of the image gray levels as shown in Figure P12.2, and process any one of the images given in Appendix A with your program.

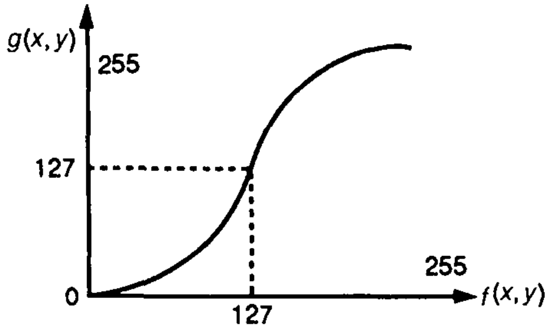


FIGURE P12.2

12.3 Write a program to perform the bileveling gray-level transformation shown in Figure P12.3, and process any of the images given in Appendix A with your program.

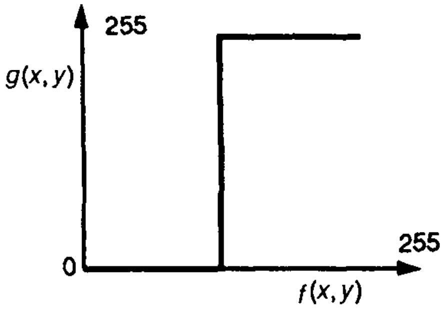


FIGURE P12.3

12.4 Write a program to perform the bright region (or dark region) gray-level transformation shown in Figure P12.4, and process any one of images given in Appendix A with your program.

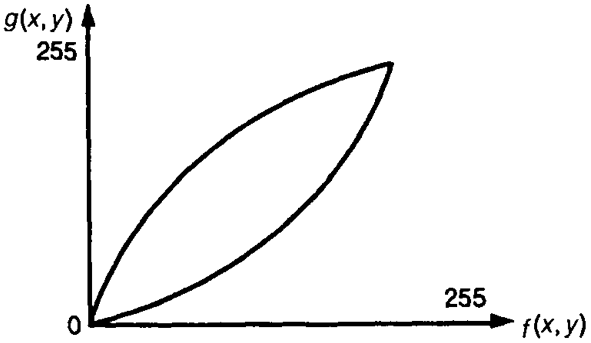


FIGURE P12.4

12.5 Write a program to perform level-slicing contrast enhancement to isolate a band of gray levels as shown in Figure P12.5, and process any one of the images given in Appendix A with your program.

12.6 Use any one of the images given in Appendix A to alter the data by processing each pixel in the image with the deterministic gray-level transformation shown in Figure P12.6, where 0 = dark and 255 = white. Use the altered data obtained as an example image for enhancement processing.

(a) Obtain a histogram of the example image.

(b) Obtain a processed image by applying the histogram equalization algorithm to these example image data.

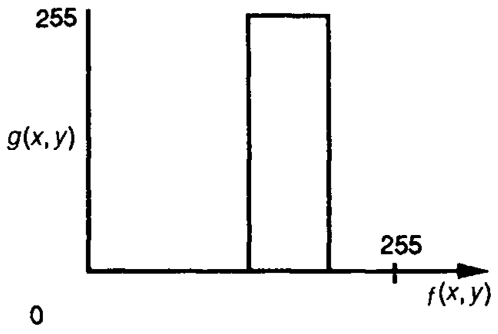


FIGURE P12.5

(c) Evaluate the histogram equalization algorithm by comparing the example image with the image after histogram equalization.

(d) Suggest a histogram specification transformation and see whether this can further improve the image.

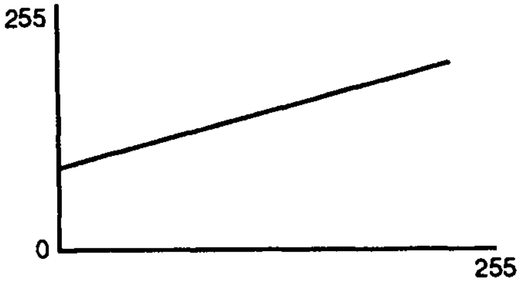


FIGURE P12.6

12.7 Given that an image has the histogram $p_r(r)$ as shown in Figure P12.7a, it is desired to transform the gray levels of this image so that

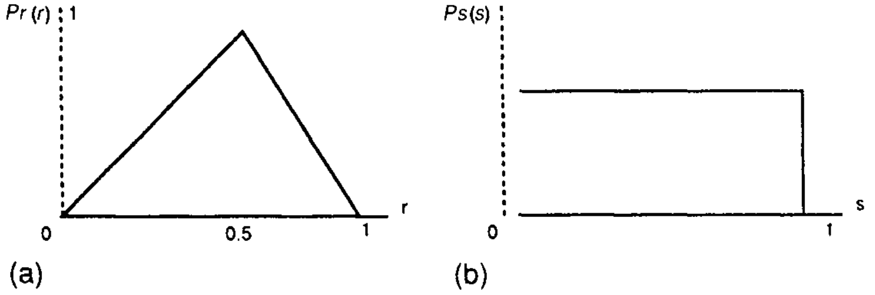


FIGURE P12.7

an equalized histogram is obtained. Find the transformation function $T(r)$.

12.8 An image has a PDF on the original gray-level scale as shown in Figure P12.8. A transformation function

$$T(r) = [S_{\max} - S_{\min}]P_r(r) + s_{\min}$$

is selected. What will be the PDF of the image on a new gray-level scale.

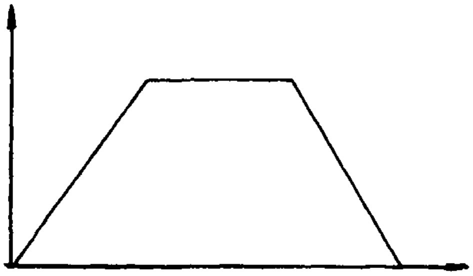


FIGURE P12.8

12.9 If the desired PDF is

$$P_s(s) = ae^{-a[S-S_{\min}]} \quad \text{and} \quad s \geq S_{\min}$$

Find $T(r)$ in terms of $P_r(r)$.

12.10 Using the histogram thinning algorithm shown on Fig. 12.27, perform thinning on the histogram shown in Figure P12.10.

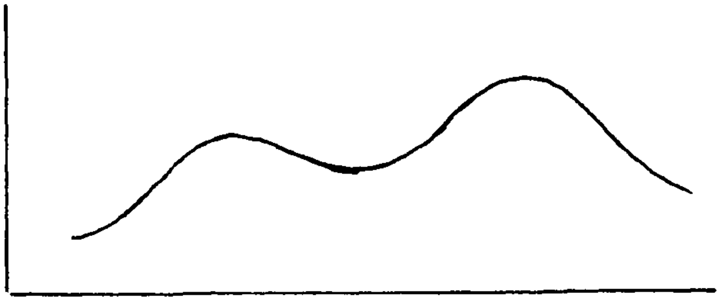


FIGURE P12.10

12.11 Corrupt one of the images given in Appendix A with gaussian noise (or some other forms of noise). Use it as an input image and process it with various masks, as shown in Figure 12.28.

12.12 Load any one of the images given in Appendix A into the computer and blur it with the following mask:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

With this image as an input image, obtain an edge enhancement image with

$$g(x,y) = \begin{cases} G[f(x,y)] & \text{if } G[f(x,y)] \geq \text{T} \\ f(x,y) & \text{otherwise} \end{cases}$$

where $G[f(x,y)]$ is the gradient of f at the point (x,y) and T is a nonnegative threshold. Compare the image after processing with the original image and note the improvement in image quality.

12.13 Repeat Problem 12.12 with another image from Appendix A or a grabbed image with a CCD camera.

12.14 Write programs for Robert's cross-gradient operator and the Sobel operator. Use these two problems to enhance the edge of the images from Problems 12.12 and 12.13. Compare and discuss their effectiveness.

12.15 Discuss the relative merits of the laplacian operator and the laplacian operator associated with local variance evaluation.

12.16 Write a program to implement the thinning algorithm for the figures shown in Figure P12.16.

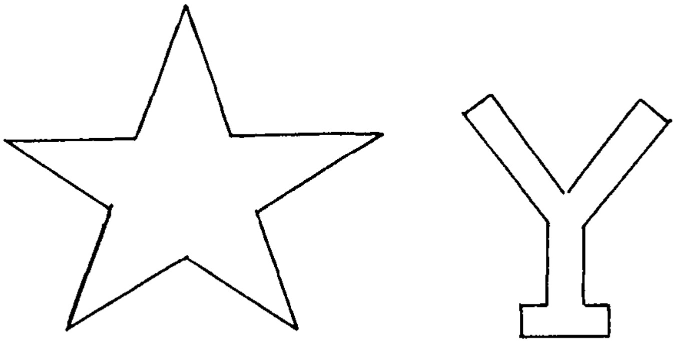


FIGURE P12.16

12.17 Dilate the figure shown in Figure P12.17a with the figure shown in Figure P12.17b.

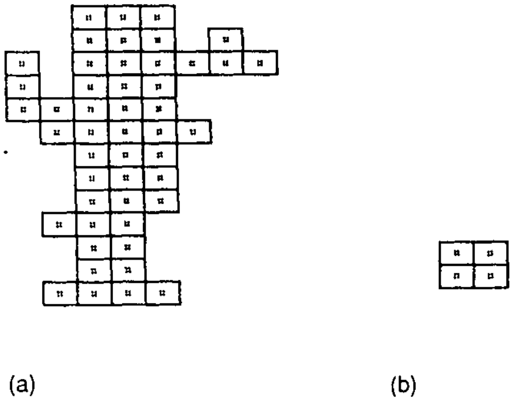


FIGURE P12.17

12.18 Find the Hough transforms of the figures shown in Figure P12.18.

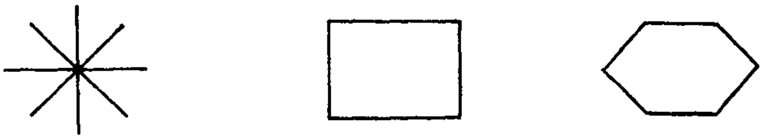


FIGURE P12.18

13

Pictorial Data Preprocessing and Shape Analysis

Pictorial data input for processing by computer falls into four different classes: (1) full gray-scale images in the form of TV pictures; (2) bilevel (black-and-white) pictures, such as a page of text; (3) continuous curves and lines, such as waveforms or region contours; and (4) sets of discrete points spaced far apart. In this chapter our discussions focus on images in the second, third, and fourth classes. Although when in printed or displayed form, images of the first class are also of bilevel microstructure, they are actually represented as full gray levels instead of simply black and white. Their gray levels are approximated by a halftone approach. The halftone technique is successful because the human visual system spatially integrates on and off the intensity value to create a perceived gray scale. We have discussed halftone images in previous chapters.

13.1 DATA STRUCTURE AND PICTURE REPRESENTATION BY A QUADTREE

It is well known that a large memory is required to store the pictorial data. For example, to store a frame of TV pictures with ordinary resolution, we would need 512 × 512, or 262,144, bytes. In addition to the volume required for storage of the pictorial data, access to memory is also an important problem for consideration.

A quadtree is a popular data structure in both graphics and image processing. It is a hierarchical data structure that provides quick access to memory for data retrieval. A quadtree is based on the principle of recursive decomposition of pictures. This technique is best used when the picture is a square matrix A, with dimensions of a power of 2, say 2^n. Then matrix A can be divided into four equal-sized quadrants, A_0, A_1, A_2, A_3, whose dimensions are half that of A. This process is repeated until blocks (possibly single pixels) are reached that consist entirely of either 1's or 0's. In this process of successive matrix decomposition, quadrants consisting of all white or all black pixels remain untouched. Only quadrants consisting of both black and white pixels are to be decomposed further. In other words, terminal nodes correspond to those blocks of the array for which further subdivision is unnecessary. The levels can be numbered starting with zero for the entire picture down to n for single pixels, as shown in Figures 13.1 and 13.2. Figure 13.3 shows a simple object coded with this quadtree structure. Black and white square nodes represent blocks consisting entirely of 1's and 0's, respectively. Circular nodes, also termed gray nodes, denote nonterminal nodes.

The pixel level is the lowest level in the quadtree representation of an image. $A, B, C, D, \ldots, P$ in Figures 13.4 and 13.5 are at a higher level. I, II, III, and IV are at an even higher level. The highest level, labeled as zero, represents the entire image. For an image of 8 × 8 pixels in our example, there are three levels. In general, for an image of $2^n \times 2^n$ pixels there will be n levels.

Conversion from a raster to a quadtree is conducted row by row starting from the first row and leftmost pixel. Take a 8 × 8 pixel image (Figure 13.6) as an example to illustrate the procedure.

1. Start from the first pixel of the first row (i.e., pixel 1).
2. When the first pixel is processed, we add a nonterminal node A, which is at a higher level. At the same time, we add three white nodes as its remaining sons, as shown in part (c) of Figure 13.7.
3. Ascend a NW link to reach node A and descend from A again to the NE son of A, or the eastern neighbor of node 1 (i.e., node 2). This will be the eastern edge of the block.
4. Try to add another eastern neighbor to it. If a common ancestor does not exist, a nonterminal node I is added, with its three remaining sons being white (see Figure 13.7f).
5. Descend along the retraced path. During this descent, a white node is converted to a gray node and four white sons are added (see Figure 13.7g).
6. Color the terminal node appropriately (see Figure 13.7g and h).

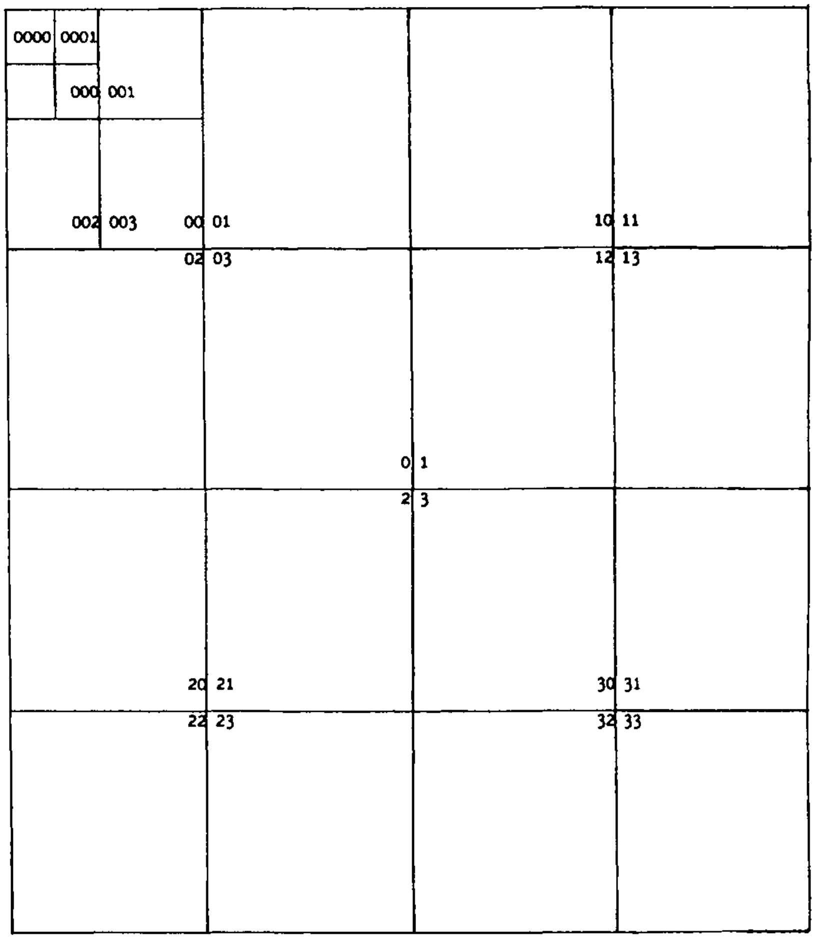


FIGURE 13.1 Numbering in the quadtree decomposition.

7. Go to the first pixel of the second row (even-numbered row), pixel 9 in this example, and repeat the process outlined in the preceding steps.
8. Merge the four pixels NW, NE, SW, and SE if they are all black.

Figures 13.8 and 13.9 shows the quadtree after processing the first and second rows of Figure 13.6.

13.2 DOT-PATTERN PROCESSING WITH VORONOI APPROACH

When processing visual information, dot patterns other than gray level or color images appear frequently. For example, the nighttime sky is a natural dot pattern. Locations of landmarks in cartography are also detected as dot patterns by airborne sensors. Objects in an image represented by locations of their spatial

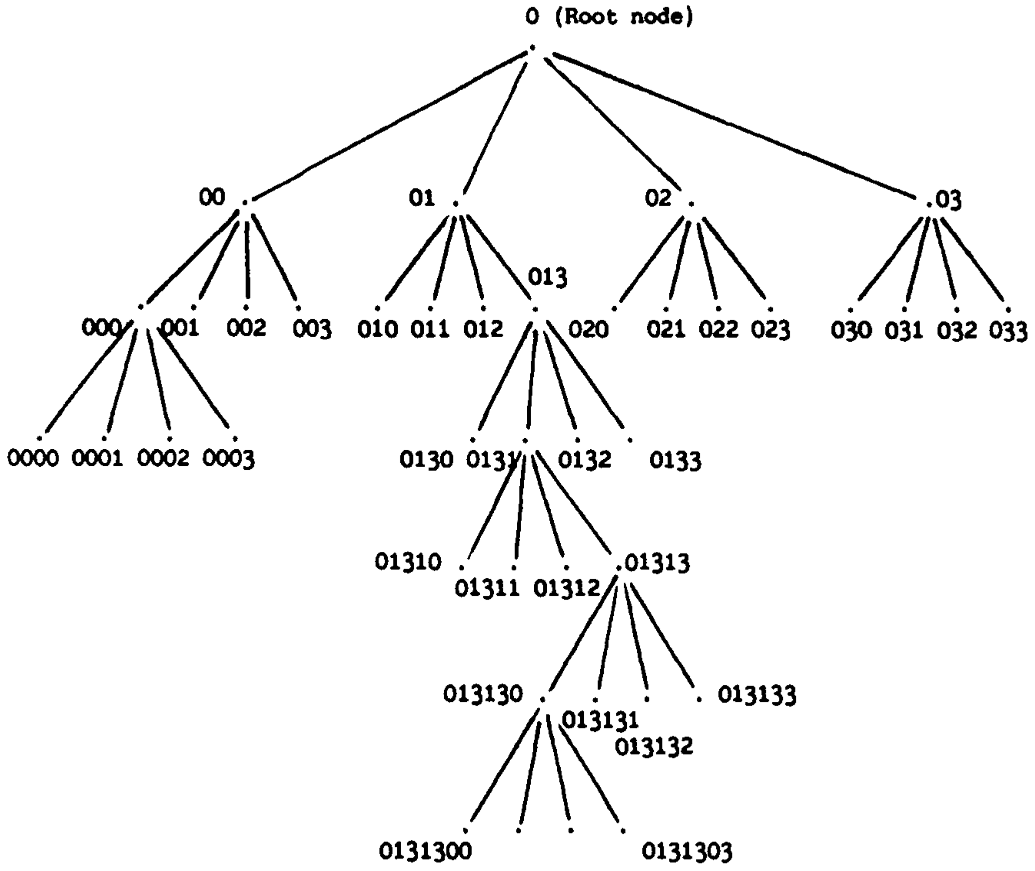


FIGURE 13.2 Quadtree and its addressing notation. Length = 2^7 = 128 pixels.

features, such as spots, corners, and so on, are also examples of dot patterns. Edge pixels obtained after edge detection also appear in the form of dots (edge pixels). Methods such as edge detection followed by edge linking, minimum spanning tree, and the Delaunay method are found to be effective in treating dot patterns. However, in many other applications, for example, (1) dot patterns with varying density, such as those shown in Figure 13.10; (2) dot patterns that appear in the form of a cluster with direction-sensitive point density, such as that shown in Figure 13.11; (3) dot patterns with curvelike and/or necklike clusters, as shown in Figure 13.12; and (4) dot patterns in the form of globular and nonglobular clusters, as shown in Figure 13.13, another method, called Voronoi tessellation, is found to be more effective.

13.2.1 Voronoi Tessellation

Voronoi tessellation consists of a Voronoi diagram together with incomplete polygons on the convex hull. Figure 13.14 shows Voronoi polygons, which are

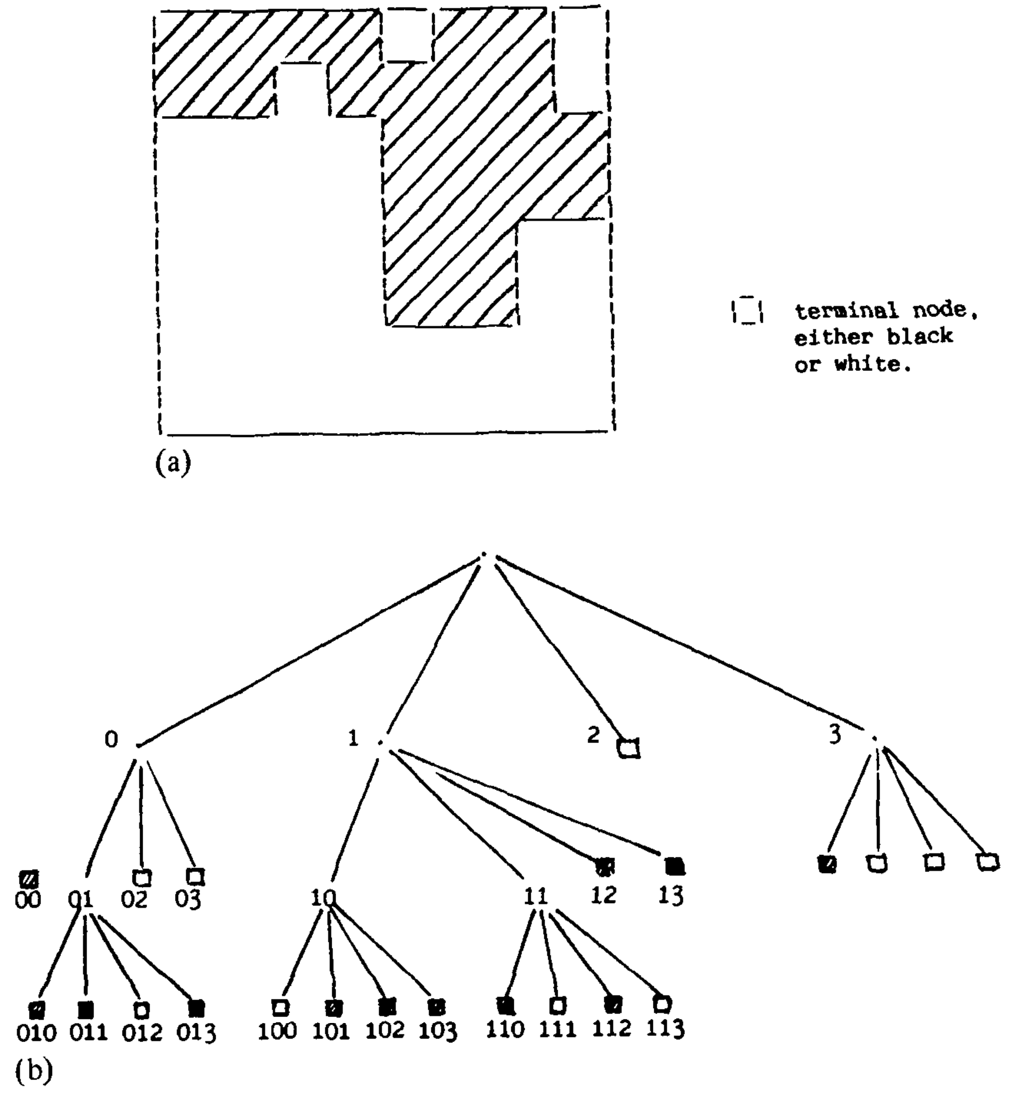


FIGURE 13.3 (a) Simple object; (b) its quadtree representation.

polygons (regions) containing dot points. Voronoi polygons associated with Voronoi neighborhood relationships among points can be used to analyze dot patterns. The same diagram (see Figure 13.14) includes Delaunay tessellation, the edges obtained by joining each point with its neighbors.

The Voronoi neighborhood is not defined on the euclidean plane, nor is it drawn based on the fixed radius concept. The reason for not so doing is the fact that approaches with the fixed radius concept are not sensitive to variations in the local point density. It is not difficult to note that in a dense dot pattern area, a point may have a large number of neighbors, whereas in a sparse region it may not have even a single neighbor. For that reason the fixed radius approach cannot

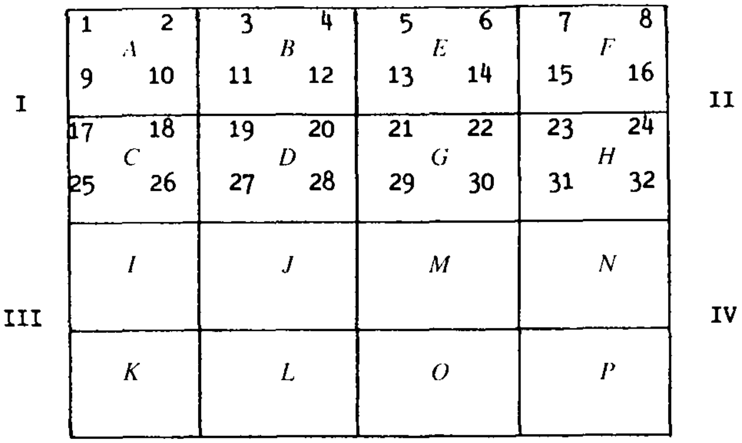


FIGURE 13.4 Recursive decomposition of an image for quadtree representation.

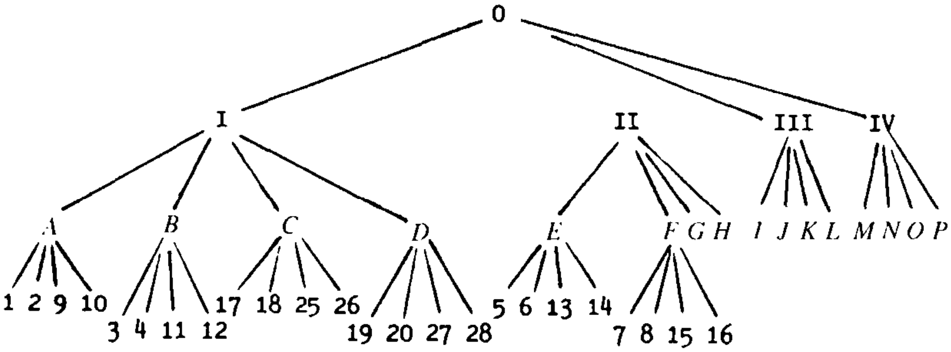
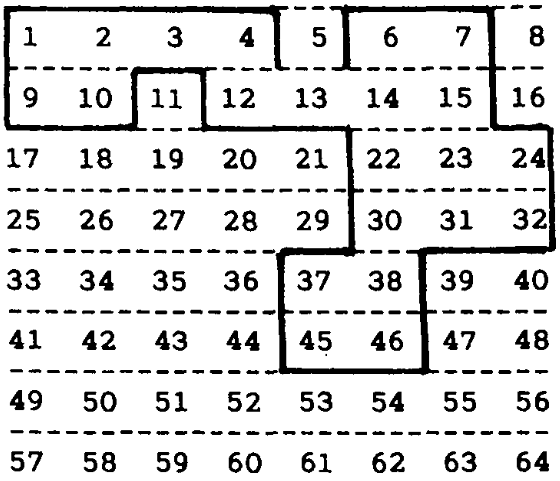


FIGURE 13.5 Hierarchical data structure in quadtree representation.



1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

FIGURE 13.6 An 8 × 8 pixel image.

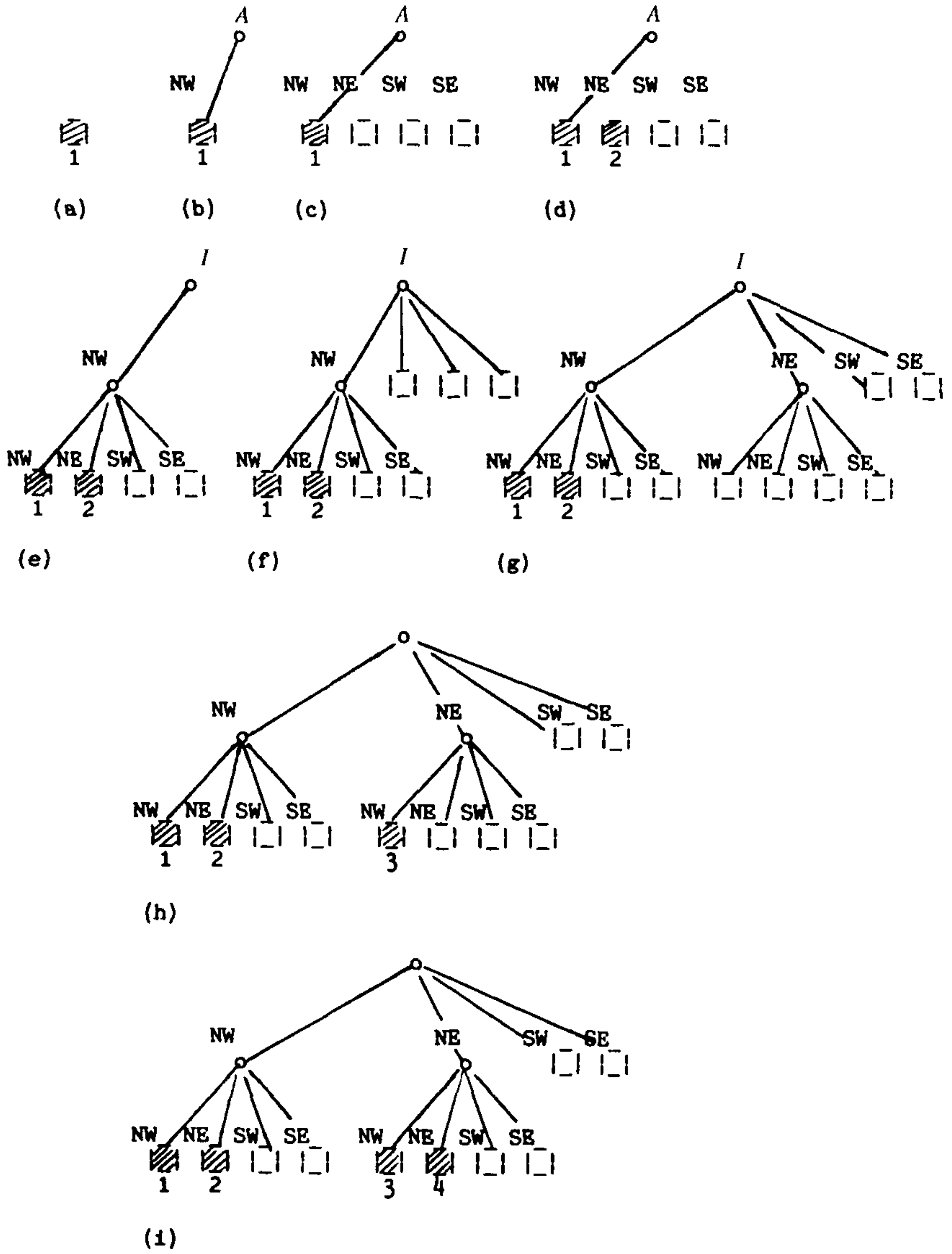


FIGURE 13.7 Intermediate trees in the process of obtaining a quadtree for the first part of the first row of the image shown in Figure 13.6. (From Samet, 1981.)

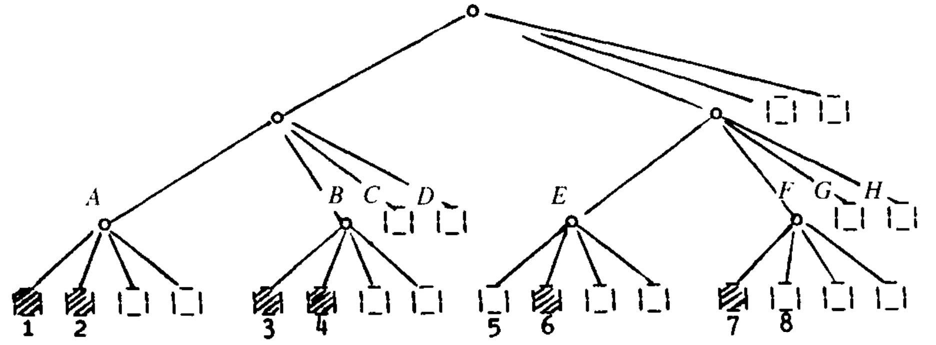


FIGURE 13.8 Quadtree after processing the first row of Figure 13.6. (From Samet, 1981.)

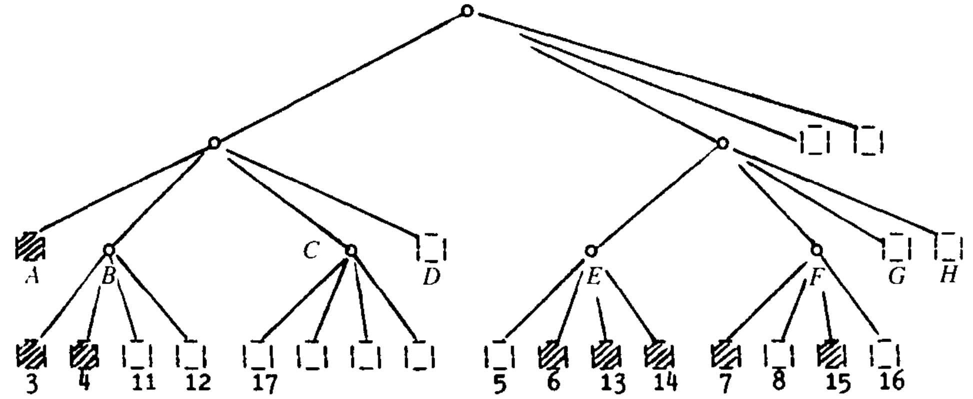


FIGURE 13.9 Quadtree after processing the second row of Figure 13.6. (From Samet, 1981.)

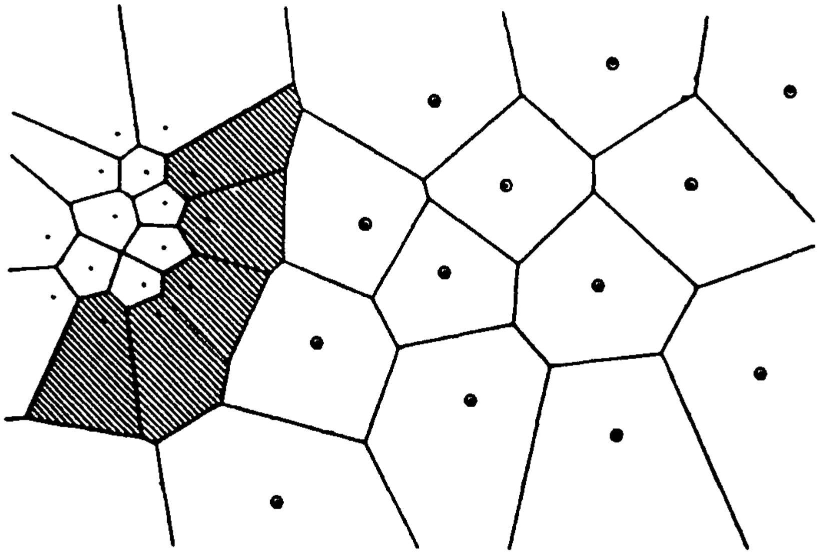


FIGURE 13.10 Dot pattern with varying intensity. (From Ahuja, 1982.)

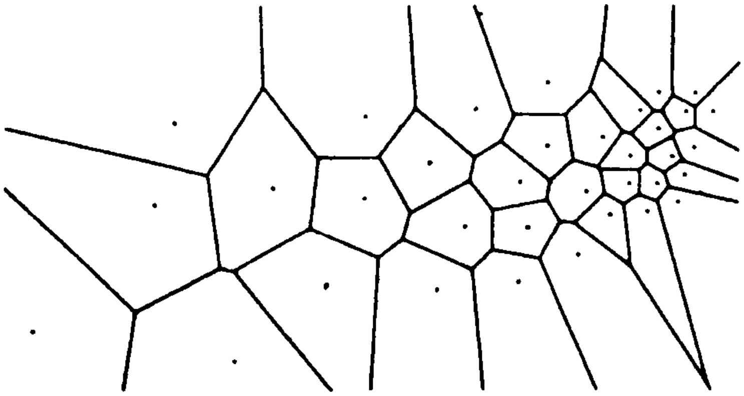


FIGURE 13.11 Dot patterns in the form of a cluster with direction-sensitive point density. (From Ahuja, 1982.)

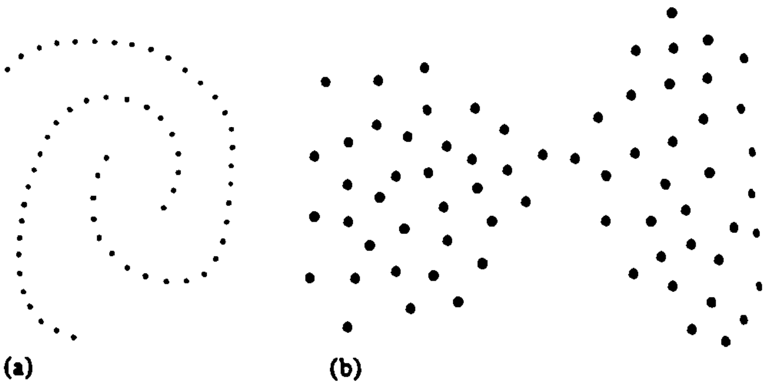


FIGURE 13.12 Dot patterns in the form of (a) a curvelike cluster and (b) clusters with a neck. (From Ahuja, 1982.)

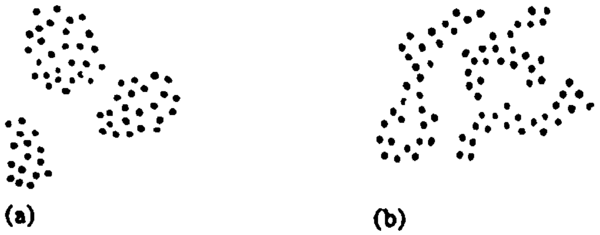


FIGURE 13.13 Dot patterns in the form of (a) globular and (b) nonglobular clusters. (From Ahuja, 1982.)

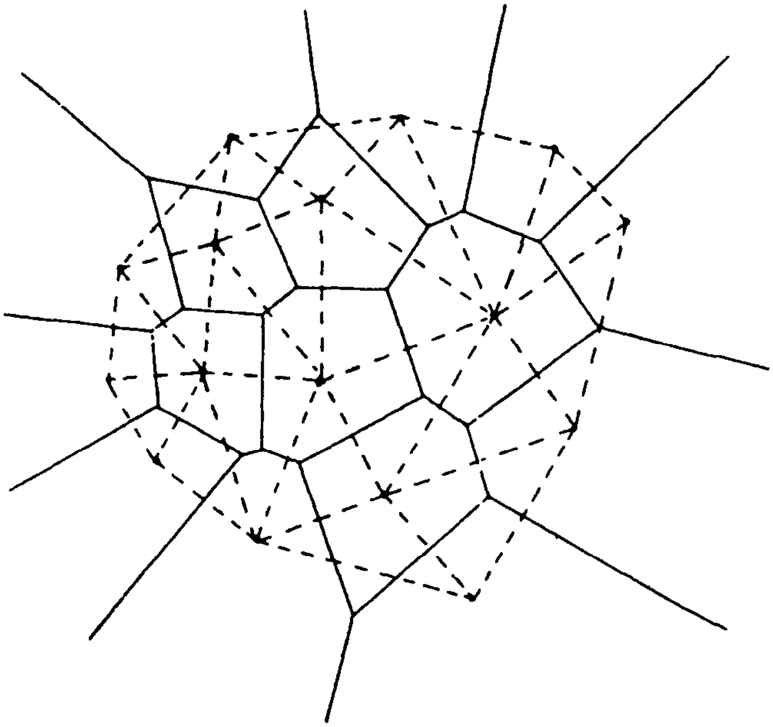


FIGURE 13.14 Voronoi tessellation defined by a given set of points. Dashed lines show the corresponding Delaunay tessellation. (From Ahuja, 1982.)

reflect the local structure very satisfactorily. Voronoi polygons are suggested for this purpose, to characterize various geometrical properties of these disjoint pictorial entities.

Let us take an example to illustrate how to use a Voronoi diagram to detect the boundary. Generally speaking, the boundary segments form multiple closed loops, and it is our intention to recover the medial axis of these closed boundaries. Let us use Voronoi diagram approach to perform this task. Consider a two-line segment image. Its Voronoi diagram can be constructed as shown in Figure 13.15, where a, b, c, and d are the endpoints of the two line segments ab and cd. The dashed curve (called a Voronoi edge) is the locus of the points that are equidistant from the two line segments. $B(a, c)$, $B((a, b), c)$, $B((a, b), (c, d))$, $B((a, b), d)$, and $B(b, d)$ represent, respectively, the subsegments of the curve, with $B(a, c)$, $B(b, d)$ being point-to-point subsegments, $B((a, b), c)$, $B((a, b), d)$ being line-to-point subsegments, and $B((a, b), (c, d))$ being the line-to-line subsegment. Note that the line-to-line subsegment is a line subsegment, while the others are curved subsegments. Figure 13.16a to c shows the step-by-step process of extracting the medial axes from the disjoint boundary segments. The steps are self-explanatory.

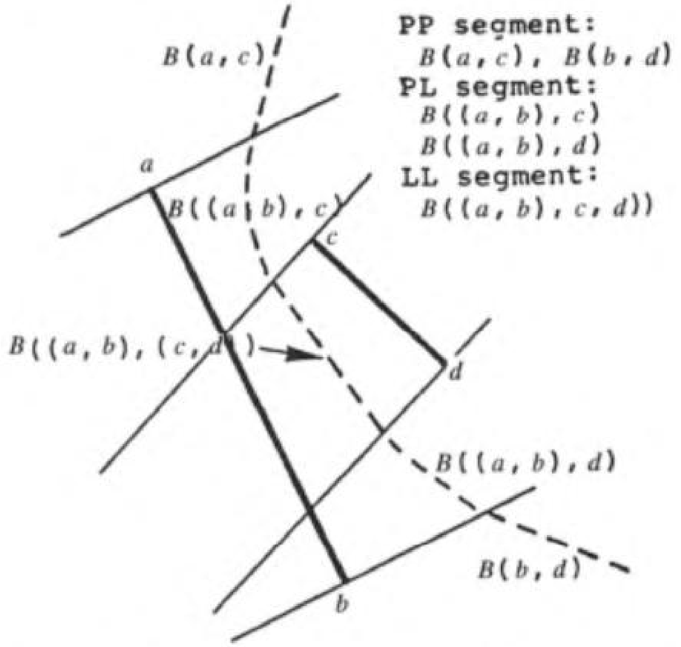


FIGURE 13.15 Voronoi diagram of two line segments. (From Matsuyama and Phillips, 1984.)

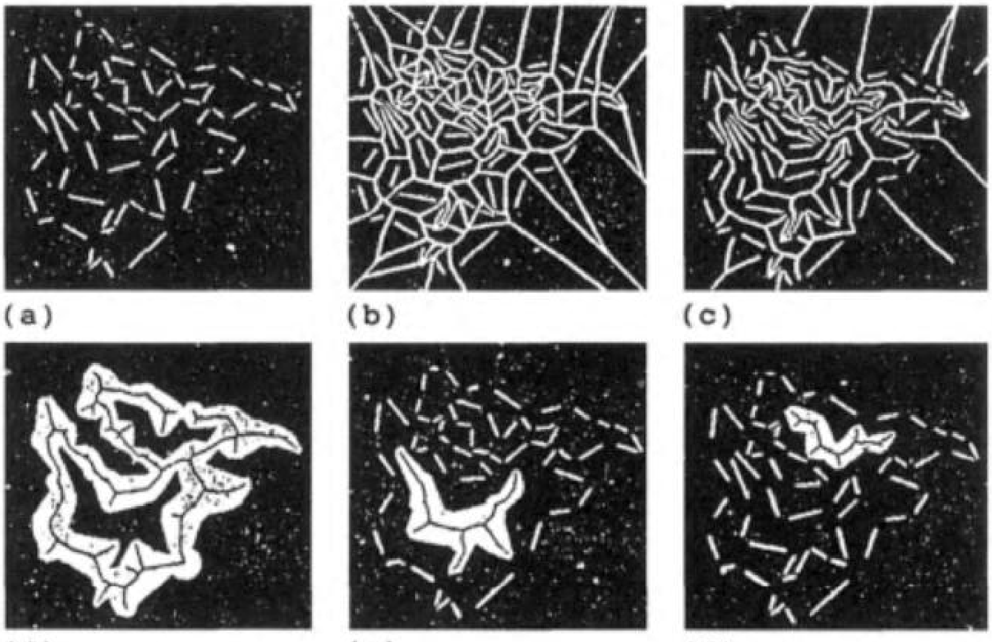


FIGURE 13.16 Step-by-step process ın extracting the medial axis from the disjoint boundary of a region. (a) Disjoint boundary segments; (b) Voronoi diagram; (c) extracted medial axis of the region; (d)–(f) three major medial axes and regions expanded from them. (From Matsuyama and Phillips, 1984.)

13.3 ENCODING OF A PLANAR CURVE BY CHAIN CODE

For some applications it may be adequate to represent a curve as a sequence of pixels. But for many other cases it might be much more desirable to have a more compact form to represent them, even with a mathematical description. The term "curve fitting" is usually used to refer to the process of finding a curve that passes through a set of points. A lot of studies has been devoted to curve fitting, namely, polynomial curve fitting, piecewise polynomial curve fitting, B-splines, polygonal approximations, concatenated arc approximations, and so on. We discuss some of them below. Figure 13.17 shows a common chain code notation using 8 directions and 48 directions. Figure 13.18 shows a skeletonized graph and its chain coding, where the exponents denote repeated symbols.

13.4 POLYGONAL APPROXIMATION OF CURVES

Polygonal approximation for a set of data points as suggested [see Pavlidis (1982) for a rigorous treatment of this subject] can be described briefly as follows. Subdivide the data points into groups, each of which is to be approximated by a polygonal side. Start by drawing a line L_1 to connect the first point P_1 and the last point P_j of the first group of points of size k_0, as shown on Figure 13.19. If the collinearity test succeeds on this group of points (k_0 in size), a new line L_2 is to be drawn to connect point P_j to point P_k. P_1 is then established as a breakpoint, and the new line L_2 becomes the current line. A merge check then follows by comparing the inclination angle of L_1 with that of line L_2. If the difference between these two inclination angles is small, a new line L_3, instead of L_1 and L_2, will be used to approximate these two sets of points. Otherwise, L_1 will be kept as a polygonal side. Repeat the process from the endpoint of L_1, point P_j, until the

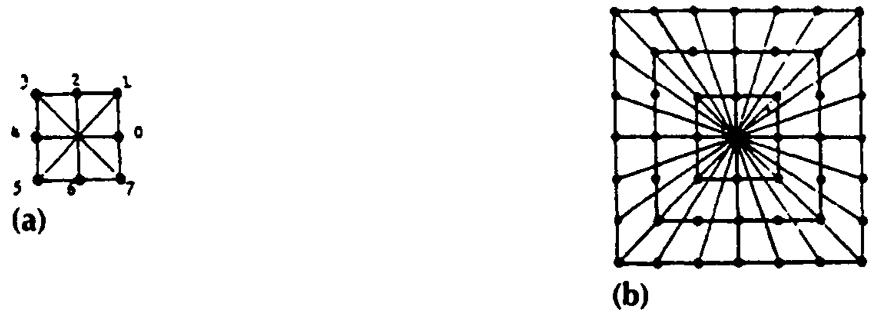
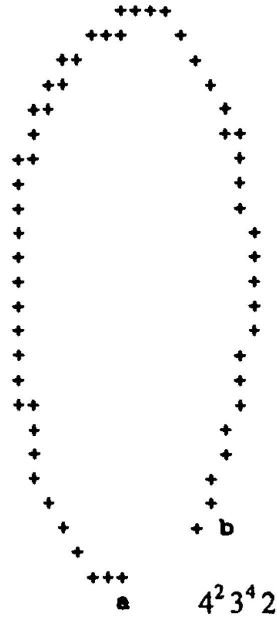


FIGURE 13.17 Chain codes. (a) Eight directions; (b) more than eight directions. (From Freeman, 1978.)



$$4^2 3^4 2^3 4\, 2^{10} 0\, 2^2 0\, 2\, 0\, 2\, 0\, 1\, 0^2 2\, 0^3 7^4 6\, 0\, 6^3 7\, 6^4 5\, 6^2 5\, 6\, 5\, 6\, 5$$

FIGURE 13.18 Graph represented by an eight-direction chain code.

complete set of the data points is successfully approximated by the polygonal sides.

Some mathematics are involved in this approximation. It is clear that the equation for a line passing through two points (x_1, y_1) and (x_2, y_2) is

$$y = y_1 \frac{x - x_2}{x_1 - x_2} + y_2 \frac{x_1 - x}{x_1 - x_2} \tag{13.1}$$

It follows that the line used to approximate the points (x_i, y_i), (x_{i+1}, y_{i+1}), $\ldots, (x_k, y_k)$ will be given by

$$x(y_i - y_k) + y(x_k - x_i) + y_k x_i - y_i x_k = 0 \tag{13.2}$$

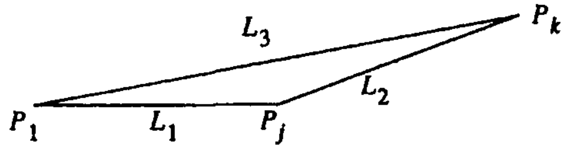


FIGURE 13.19 Polygon approximation approach.

For a point (u, v) that does not lie on the line, from geometry this point will be at a distance DIST from the line, where

$$\text{DIST} = \frac{u(y_i - y_k) + v(x_k - x_i) + y_k x_i - y_i x_k}{\sqrt{(y_i - y_k)^2 + (x_k - x_i)^2}}. \tag{13.3}$$

Sklansky and Gonzalez (1978) have suggested another method for the approximation of digitized curves. Their method is based on the minimum perimeter polygonal approximation. Suppose that we have a set of data points, say A, and are to find a polygonal approximation B for these data points such that the Hausdorff–euclidean distance

$$H(A, B) = \text{MAX}\left[\underset{x_1 \in B}{\text{MAX}}\ \underset{x_2 \in A}{\text{MIN}}\,|x_1 - x_2|,\ \underset{x_1 \in A}{\text{MAX}}\ \underset{x_2 \in B}{\text{MIN}}\,|x_1 - x_2|\right] \tag{13.4}$$

between the set of points A and the curve approximation B is less than a prespecified value ε, where $|x_1 - x_2|$ is the euclidean distance between x_1 and x_2 and is

$$|x_1 - x_2| = [(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2]^{1/2} \tag{13.5}$$

Figure 13.20 describes the process. Keep in mind from the previous discussion that the distance from any one point to the side of the polygon should be less than ε. So the first step in this process is to construct circles with the data points as center and ε as radius. As shown in Figure 13.20, $1, 2, 3, \ldots, 9$ are the data points, t_3 is the top tangent to the circle with point 3 as center and ε as radius, and b_3 is its bottom tangent. Any line segment lying inside or tangent to all the circular apertures is a valid line segment for the polygon we are looking for. But to obtain an optimum line segment for these data points, one more step should be taken.

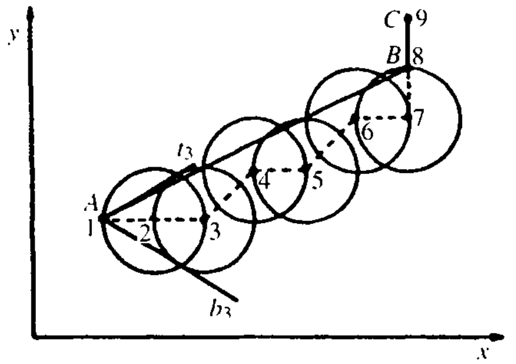


FIGURE 13.20 Rectified minimum perimeter polygons (RMPP) finder. (From Sklansky and Gonzalez, 1980.)

Draw two tangents, denoted by t_i and b_i, from the source point to the circular aperture of each point i. In Figure 13.20 only t_3 and b_3 are shown. Two cones, TCONE$_i$ and BCONE$_i$, are formed, which are defined, respectively, as the angles between t_i (or b_i) with the positive x direction. Find the smallest value of TCONE$_i$ (BCONE$_i$), $i = 1, 2, 3, \ldots, N$, inside which all the segments between the current data point and source point lie. The line segment corresponding to the smallest values of BCONE will be the optimum segment.

13.5 ENCODING OF A CURVE WITH B-SPLINE

A spline is a piecewise polynomial function used to describe a curve that can be divided into intervals. Each of these intervals is represented individually by a separate polynomial function, as given by

$$p(x) = p_1(x) + \sum_{i=1}^{k-1} q_i(x - x_i)^m \tag{13.6}$$

where $(x - x_i)^m$ is zero when $(x - x_i) \leq 0$, and q_i is a constant proportional to the mth derivative of $p(x)$ with respect to x at $x = x_i$. The piecewise polynomial function is called a linear spline when $m = 1$, a quadratic spline when $m = 2$, and a cubic spline when $m = 3$. In many practical applications, another form of spline representation, called B-splines, is used. This B-spline is zero at all subintervals except $m + 1$ of them. The linear and quadratic B-splines are defined, respectively, as

$$B_{i,1}(x) = \begin{cases} \dfrac{x - x_i}{x_{i+1} - x_i} & x_i \leq x \leq x_{i+1} \\[2ex] \dfrac{x_{i+2} - x}{x_{i+2} - x_{i+1}} & x_{i+1} \leq x \leq x_{i+2} \end{cases} \tag{13.7}$$

and

$$B_{i,2}(x) = \begin{cases} \dfrac{(x - x_i)^2}{(x_{i+2} - x_i)(x_{i+1} - x_i)} & x_i \leq x \leq x_{i+1} \qquad (13.8a) \\[3ex] \dfrac{(x - x_i)(x_{i+2} - x)}{(x_{i+2} - x_i)(x_{i+2} - x_{i+1})} + \dfrac{(x_{i+3} - x)(x - x_{i+1})}{(x_{i+3} - x_{i+1})(x_{i+2} - x_{i+1})} & \\ & x_{i+1} \leq x \leq x_{i+2} \qquad (13.8b) \\[3ex] \dfrac{(x_{i+3} - x)^2}{(x_{i+3} - x_{i+1})(x_{i+3} - x_{i+2})} & x_{i+2} \leq x \leq x_{i+3} \qquad (13.8c) \end{cases}$$

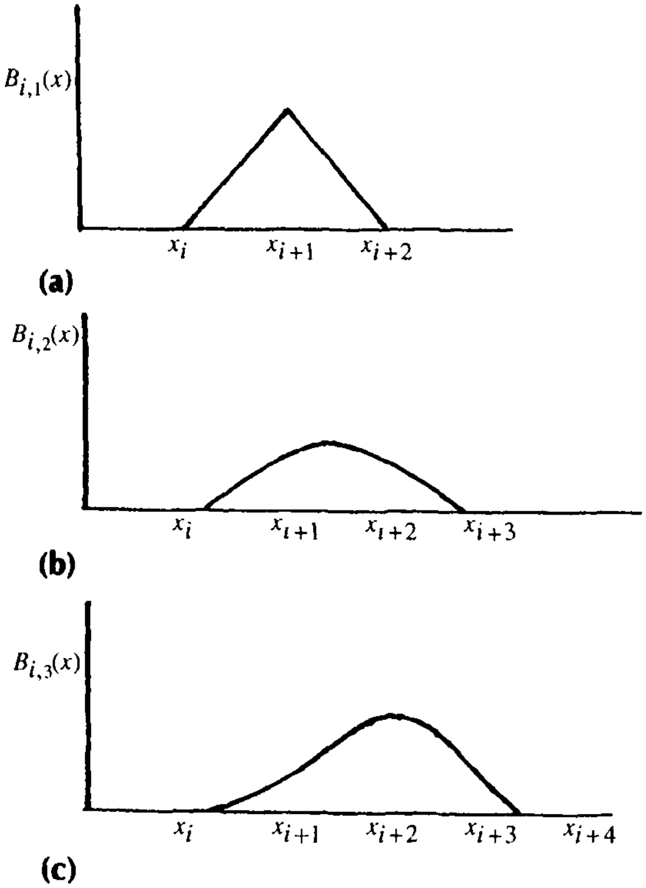


FIGURE 13.21 B-spline. (a) Linear ($m = 1$); (b) quadratic ($m = 2$); (c) cubic ($m = 3$).

From Eqs. (13.7) and (13.8) it is not difficult to see that the mth-degree B-spline can be generated via the following recursion:

$$B_{i,m}(x) = \frac{x - x_i}{x_{i+m} - x_i} B_{i,m-1}(x) + \frac{x_{i+m+1} - x}{x_{i+m+1} - x_{i+1}} B_{i+1,m-1}(x) \qquad (13.9)$$

Some examples of linear, quadratic, and cubic B-splines are shown in Figure 13.21. Figure 13.22a shows a set of sampled spline boundary points, and Figure 13.22b shows its quadratic B-spline interpolations.

13.6 SHAPE ANALYSIS VIA MEDIAL AXIS TRANSFORMATION

Shape analysis is a fundamental problem in pattern recognition and computer vision. With the term "shape" we refer to the invariant geometrical properties

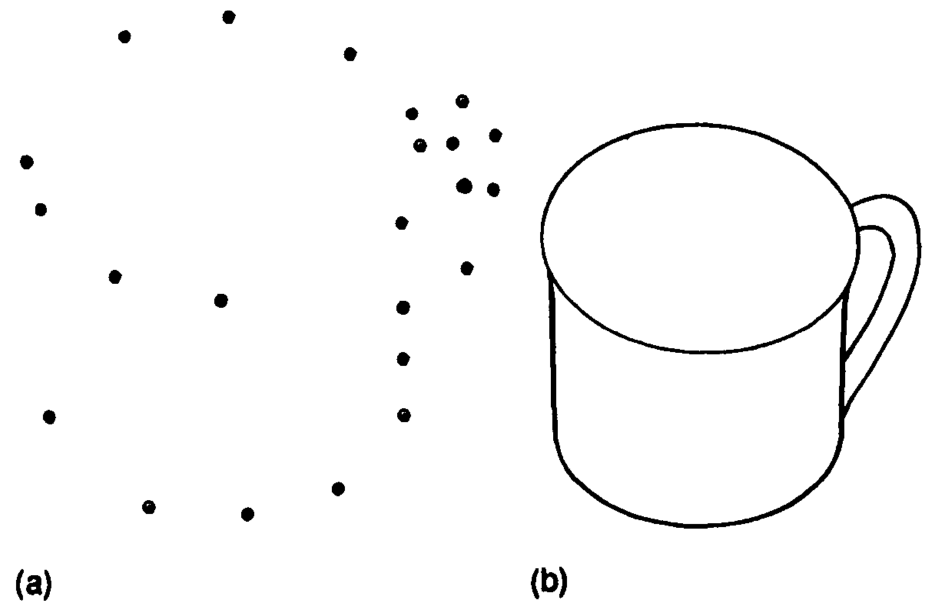


FIGURE 13.22 B-spline curve fitted through points of original boundaries. (a) Given points; (b) quadratic B-spline interpolation.

among a set of spatial features of an object. The contour of the silhouette conveys a lot of information about these geometrical features, which are useful for the analysis and recognition of an object. This information is normally considered to be independent of scale and orientation. Many different approaches have been suggested for pictorial shape analysis and recognition. Some of them are discussed below.

Medial axis transformation is a method proposed for shape description by Blum (1964) and studied by many others. The medial axis of a figure is also called a symmetrical axis, or the skeleton of a figure, which is the union of all the skeletal points. A skeletal point is defined as follows. Given a region *R*, say a simple rectangle having a boundary *B* as shown in Figure 13.23a, the skeletal point is defined as that point which has two equal nearest neighbors on *B* (i.e., no other points will give a distance that is less than the one between the point and its two neighbors). This describes a circle that is completely enclosed by the figure, as shown in Figure 13.23. From this definition it is obvious that the circles centered at points on the axis with radii specified by the radius function are tangent to at least two boundary points. With such a representation it is then possible to recover the original figure by taking the union of all the circles centered on the points comprising the axis, each with a radius given by a radius function. Obviously, computation of the medial axis involved will be on the order of the square of the number of the boundary edges of the figure.

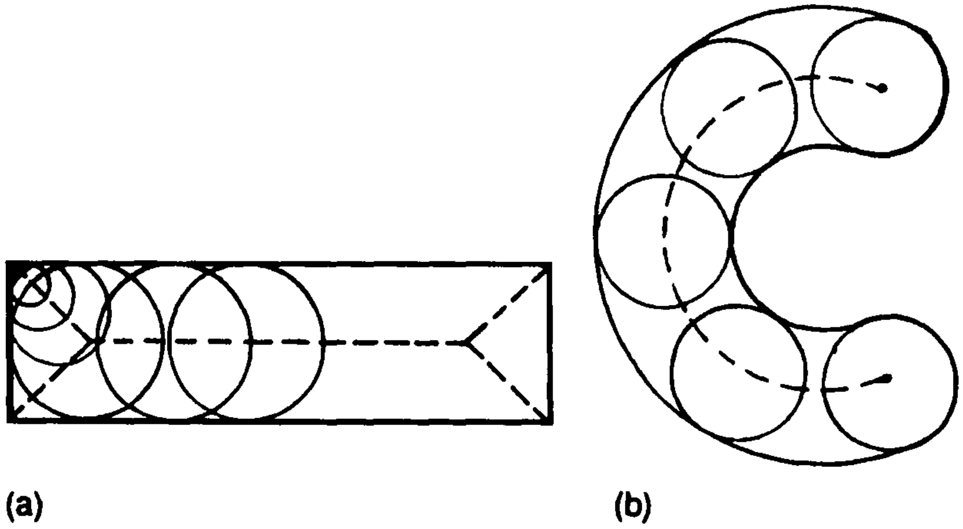


FIGURE 13.23 Some shapes and their medial axes. (a) Medial axis of a rectangle; (b) medial axis of a telocentric chromosome shape.

Another way of obtaining the medial axis transformation of a planar shape is via a Voronoi diagram. First draw the Voronoi diagram of the polygon as shown in Figure 13.24. Denote those vertices of the polygon as convex vertices when the internal angle at the vertex is less than 180°, and as reflex vertices if the internal angle at that vertex is greater than 180°. So in Figure 13.24, vertices within the dashed circles are reflex vertices. Then remove all the Voronoi edges incident with each reflex vertex. We will obtain a medial axis of the polygon as shown in Figure 13.25.

13.7 SHAPE DISCRIMINATION USING FOURIER DESCRIPTOR

In this section we show that Fourier descriptors (FDs) can be used on a quantitative basis for the description of a shape. After normalization, FDs can be matched to a test set of FDs regardless of its original size, position, and orientation. A Fourier descriptor (Persoon and Fu, 1977) is defined as follows. Refer to Figure 13.26 and assume that the curve is a clockwise-oriented closed curve and is represented parametrically as a function of arc length l as $(x(l), y(l))$, and that the angular direction of the curve at point l is represented as $\theta(l)$, with the arc l varying from 0 to L. When the curve is moving along from the starting point

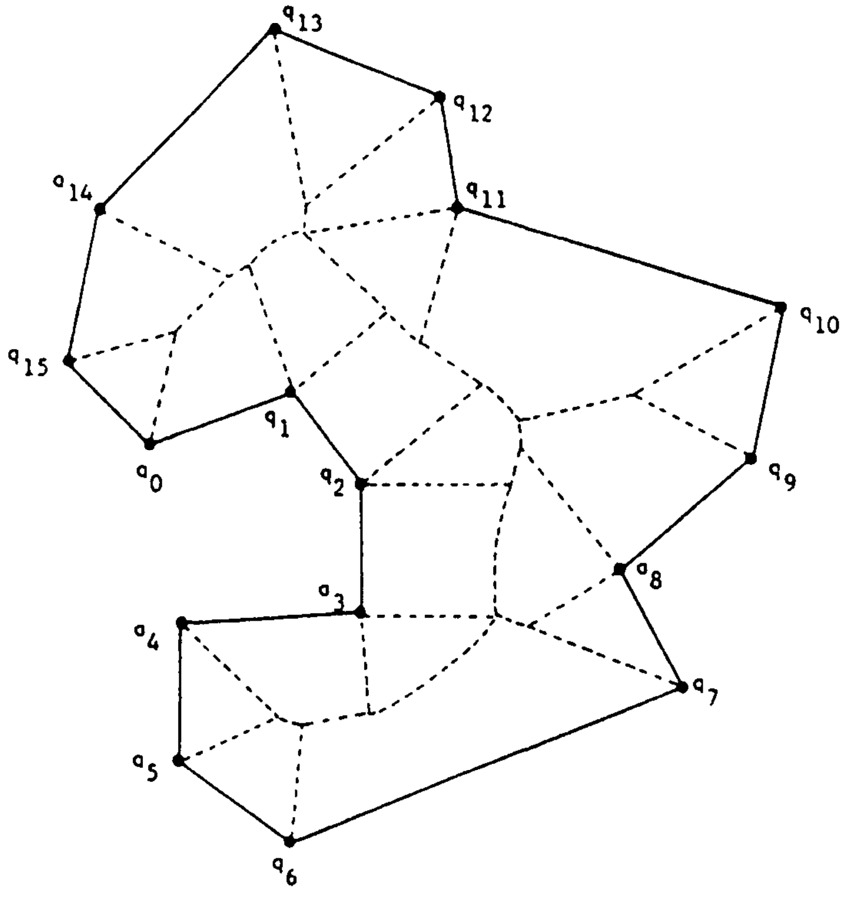


FIGURE 13.24 Polygon and its computer-generated Voronoi diagram. (From Lee, 1982.)

to point l, there will be an angular change in direction of the curve (between the starting point and the point l). Let us denote this angular change by $\phi(l)$, where

$$\phi(l) = \theta(l) - \theta(0)$$

For a closed curve, $\phi(0) = 0$ and $\phi(L) = -2\pi$. In other words, $\phi(l)$ changes from 0 to -2π. We can then define a new function $\phi^*(t)$ such that $\phi^*(0) = \phi^*(2\pi) = 0$ with t domain in $[0, 2\pi]$ as follows:

$$\phi^*(t) = \phi\left(\frac{Lt}{2\pi}\right) + t \tag{13.10}$$

It is not difficult to see that $\phi^*(t) = 0$ when $t = 0$, and $\phi^*(t)$ also equals 0 when $t = 2\pi$, due to the fact that there is a net angular change of -2π for a closed curve.

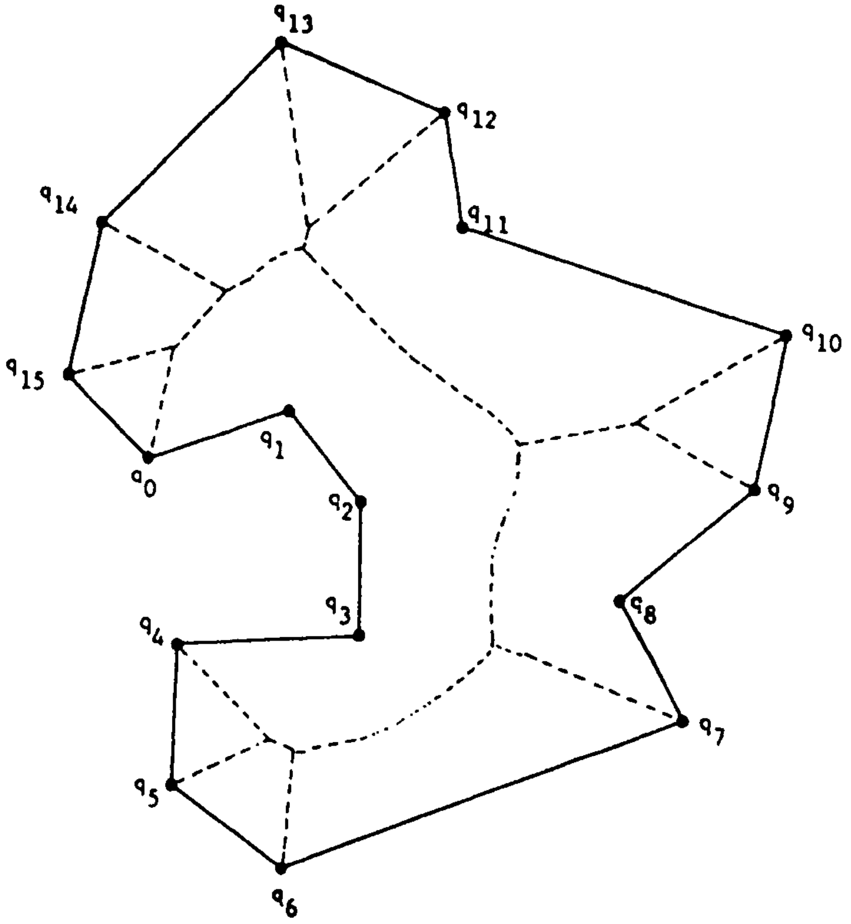


FIGURE 13.25 Computer-generated medial axis of the polygon shown in Figure 13.24. (From Lee, 1982.)

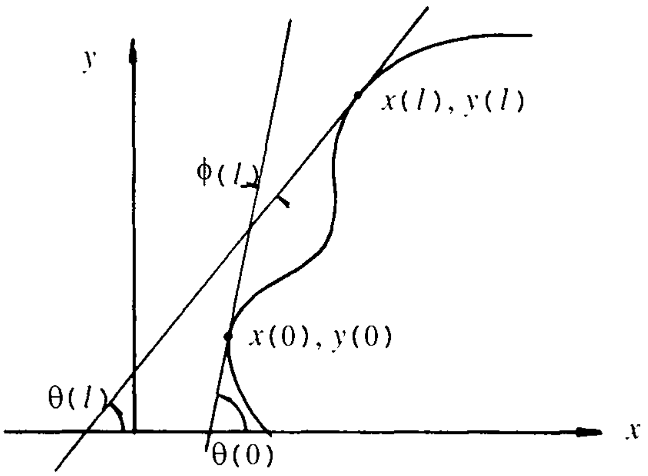


FIGURE 13.26 Definition of the Fourier descriptor.

When the function $\phi^*(t)$ is expanded in a Fourier series and the coefficients are arranged in the amplitude/phase angle form, Eq. (13.10) becomes

$$\phi^*(t) = \mu_0 + \sum_{k=1}^{\infty} A_k \cos(kt - \alpha_k) \tag{13.11}$$

The complex coefficients (A_k, α_k), $k = 1, 2, 3, \ldots$, are called Fourier descriptors (FDs) of the curve (or boundary when describing the shape). These coefficients can be used for the analysis of shape similarity or symmetry. Figure 13.27 shows the effect of the truncation and quantization of the Fourier

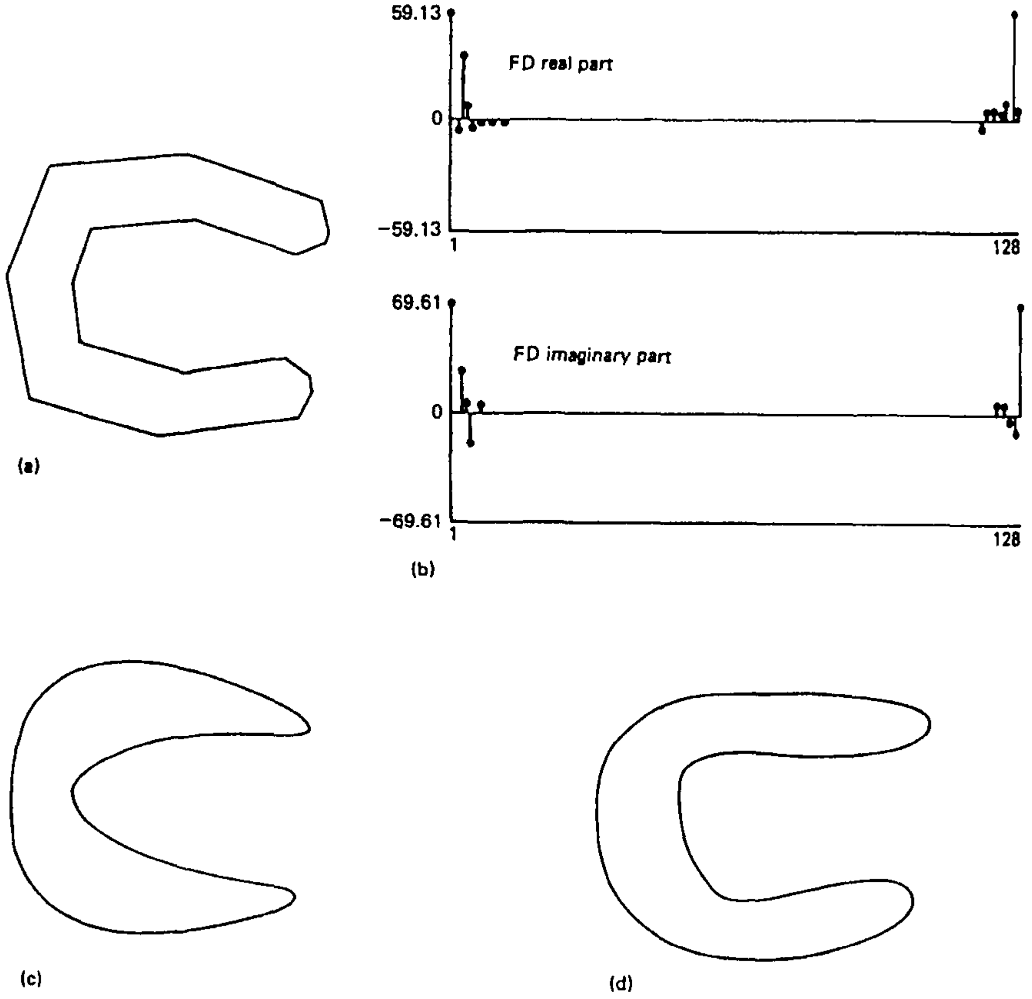


FIGURE 13.27 Fourier descriptor. (a) Given shape; (b) FDs, real and imaginary components; (c) shape derived from largest five FDs; (d) derived from all FDs quantized to 17 levels each. (From Jain, 1989.)

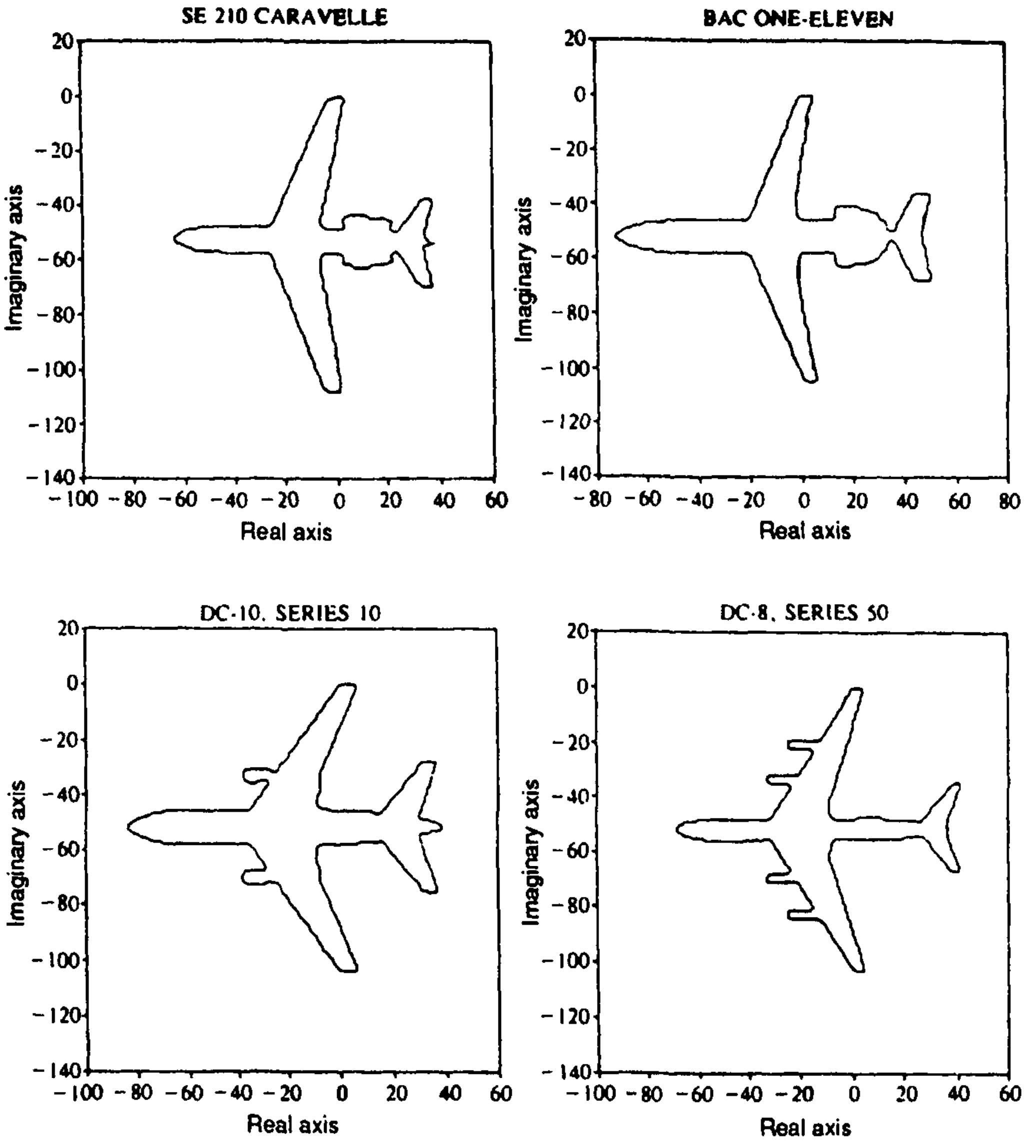


FIGURE 13.28 Shapes obtained by using Fourier descriptors. (Courtesy of T. Wallace, Electrical Engineerng Department, Purdue University.)

descriptor. Figure 13.28 shows some shapes obtained by using a Fourier descriptor.

13.8 SHAPE DESCRIPTION VIA THE USE OF CRITICAL POINTS

The basis of this method is to divide a curve into segments and then use relatively simple features to characterize the segments. The key to this method is an effective segmentation scheme. What we expect on the scheme are that (1) some critical points should be properly chosen and detected for use with segmentation, and (2) the curve description should be independent of scale and orientation of the curve.

Freeman (1978) has expanded the critical point concept. In addition to the traditional maxima, minima, and points of inflection, he suggests including the discontinuities in curvature, endpoints, intersections (junctions), and points of tangency as critical points, since these points are all well defined to a certain degree and fortunately are not affected by scale and orientation.

Use of a line segment scan was suggested to extract the discontinuities. Consider a chain to be represented by $[a_i]_1^n, a_i \in [0, 1, \ldots, 7]$. Define a straight-line segment L_i^s such that the initium of a_{i-s+1} is connected to the terminus of a_i. If a_{ix} and a_{iy} represent the x and y components of the chain links, respectively, L_i^s will be given by

$$L_i^s = [(X_i^s)^2 + (Y_i^s)^2]^{1/2} \tag{13.12}$$

where

$$X_i^s = \sum_{j=i-s+1}^{i} a_{jx}$$

and

$$Y_i^s = \sum_{j=i-s+1}^{i} a_{jy}$$

and the inclination angle of the line segment is

$$\theta_i^s = \tan^{-1} \frac{Y_i^s}{X_i^s} \tag{13.13}$$

Variation in the angle θ_i^s, as L_i^s scans over the chain, will provide the inside of the shape of the curve. A plot of δ_i^s versus i with δ_i^s defined as

$$\delta_i^s = \theta_{i+1}^s - \theta_{i-1}^s$$

is shown for a particular example shown in Figure 13.29a. This plot is independent of orientation.

13.9 SHAPE DESCRIPTION VIA CONCATENATED ARCS

In previous sections several approaches have been introduced for the effective representation of a curve. Nevertheless, the representation still looks cumbersome from a mathematical point of view. In this section an algorithm is designed to

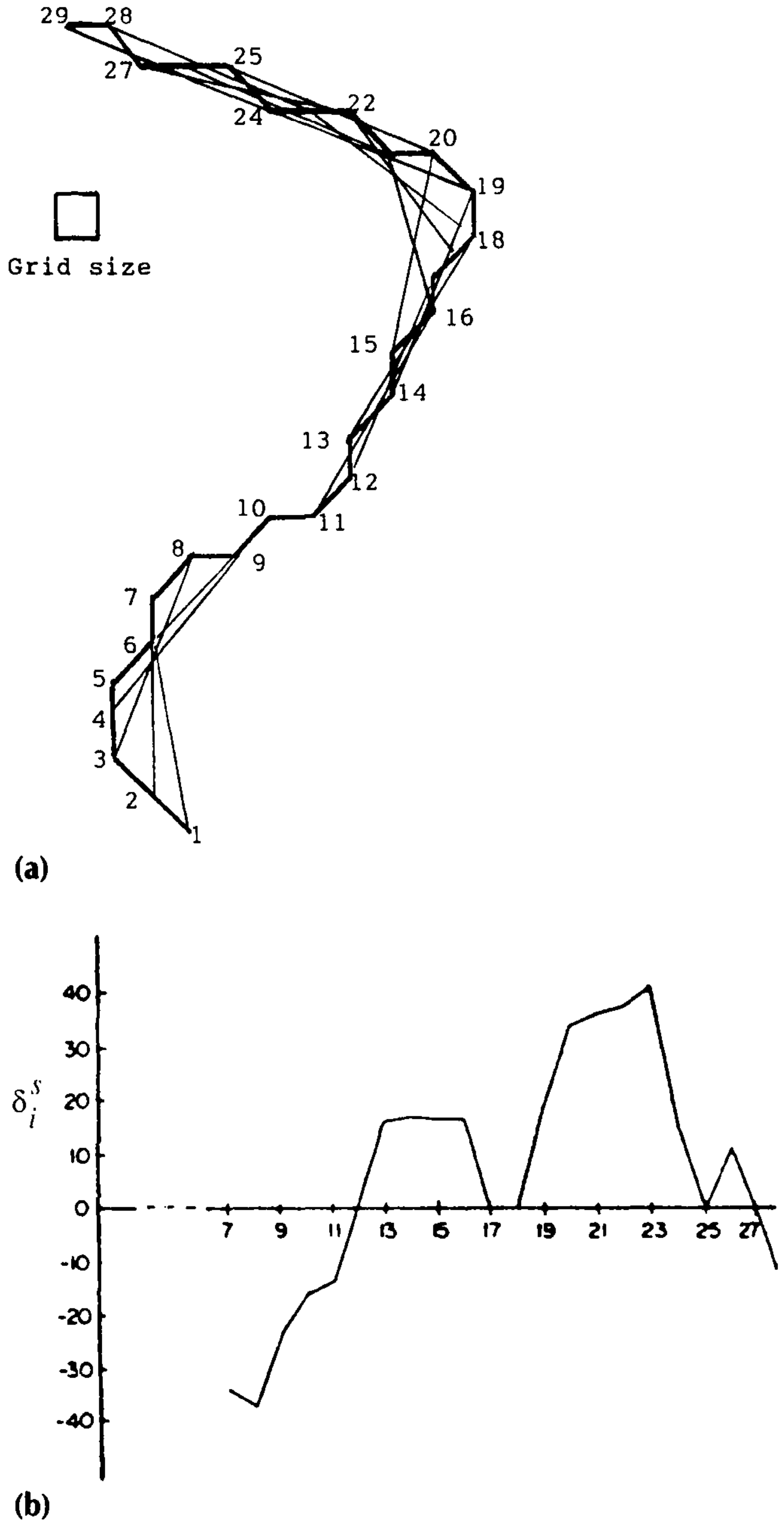


Figure 13.29 Line segment scan. (a) Chain being scanned ($s = 5$); (b) plot of incremental curvature as a function of i. (From Freeman, 1978.)

generate automatically a concise and rather accurate representation for curves in terms of concatenated arcs. The major idea involved in this approach is to detect, efficiently and effectively, the appropriate breakpoints on the curves for concatenated sections.

It is not very difficult to visualize that any composite curve can be approximated with satisfactory accuracy by a finite number of piecewisely concatenated circular arc segments exhibiting first-degree geometric continuity. (Note that a straight-line segment may also be considered as a circular arc in the sense that it has a radius of infinity in length with a center located at infinity.) The ellipse *abcd* shown in Figure 13.30a, for example, can be approximated by four arc segments $\widehat{ab}$, $\widehat{cd}$, $\widehat{bc}$, and $\widehat{da}$. Arcs $\widehat{ab}$ and $\widehat{cd}$ can be drawn with radius r_1 and centers c_1 and c_2 located on the major axis of the ellipse, while $\widehat{bc}$ and $\widehat{da}$ can be drawn with r_2 and centers c_3 and c_4, respectively. Similarly, we can find arcs to fit different portions of a parabola, whose equation is given by $y^2 = ax$ (see Figure 13.30b). For the portion P_1OP_2 of the parabola, where x^2 is very small in comparison with y^2 (which is usually the case near the vertex of a parabola), that

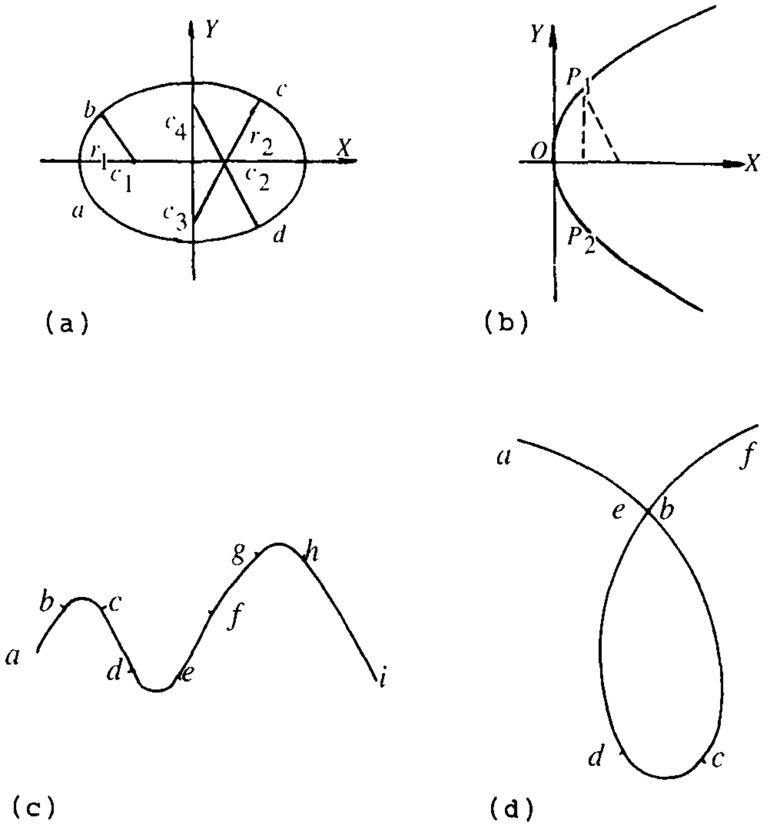


FIGURE 13.30 Approximation of curves by concatenated arcs. (a) Ellipse; (b) parabola; (c, d) composite curves. (From Honnenahalli and Bow, 1988.)

portion can also be approximated by a circular arc. A similar approximation can be applied to other curves, such as higher-order polynomial curves and transcendental curves. By following these arguments, any interpolation of an ordered set of points by a smoothed curve (see Figure 13.30c and d) can be broken down into a sequence of simpler curve segments, and accordingly can be approximated by concatenated arcs. Note that the accuracy of the approximation achieved depends greatly on the process of segmentation on the curve. More specifically, it depends on how accurate the breakpoints can be detected.

Our problem then becomes the following. Given a curve (any kind of curve), we are to segment it into several portions such that each can be represented by a circular arc. Now the first thing we ought to do is to locate accurately all the breakpoints, which are defined in this section as the points at which circularity of the curve changes.

Consider $[x_i, y_i]$, $i = 1, 2, \ldots, n$, to be the data points on the curve. Let P_k, P_l, and P_m be the three guiding points used for breakpoint searching. If the number of consecutive points in curve section P_kP_l equals that in P_lP_m, and all these data points have the same rate of inclination change, $\mathscr{K} = |d\phi/dt|$, of the tangents, they must lie on the arc drawn with the center situated at the intersection of perpendicular bisectors of the two chords $\overline{P_kP_l}$ and $\overline{P_lP_m}$. By following this argument, an algorithm can then be proposed:

1. Start from the first point, P_k, on the curve.
2. Select points P_l and P_m such that the number of points between P_k and P_l equals that between P_l and P_m. Compute the radius of curvature and the center of the circle of curvature for this curve portion (Figure 13.31). Note that the choice of P_l during the first trial depends on the radius of the curvature of the arc and has to be iterated several times for accuracy (see the experimental results and performance evaluation at the end of this section).
3. Set the new guiding point, P_l', at P_m of the preceding trial, and set the guiding point P_m' equal to $2P_l' - 1$, where P_l, P_m and P_l', P_m' are the guiding points for the previous and current runs, respectively.
4. Repeat the process by following the guiding point generating scheme shown in Table 13.1 until P_m crosses the breakpoint for the first time. When the error on the center coordinates is greater than a preset threshold, it signifies that P_m has overshot the breakpoint (see Figure 13.31d). Once the breakpoint is overshot, P_l is to be moved halfway between the previous two P_l's, as shown in Figure 13.31e. The new P_m is $2P_l - 1$, as before. Once again the center and errors in the x and y directions are computed. From this point onward, P_l is moved forward or backward until the breakpoint is reached, depending on whether P_m is within or beyond the breakpoint.

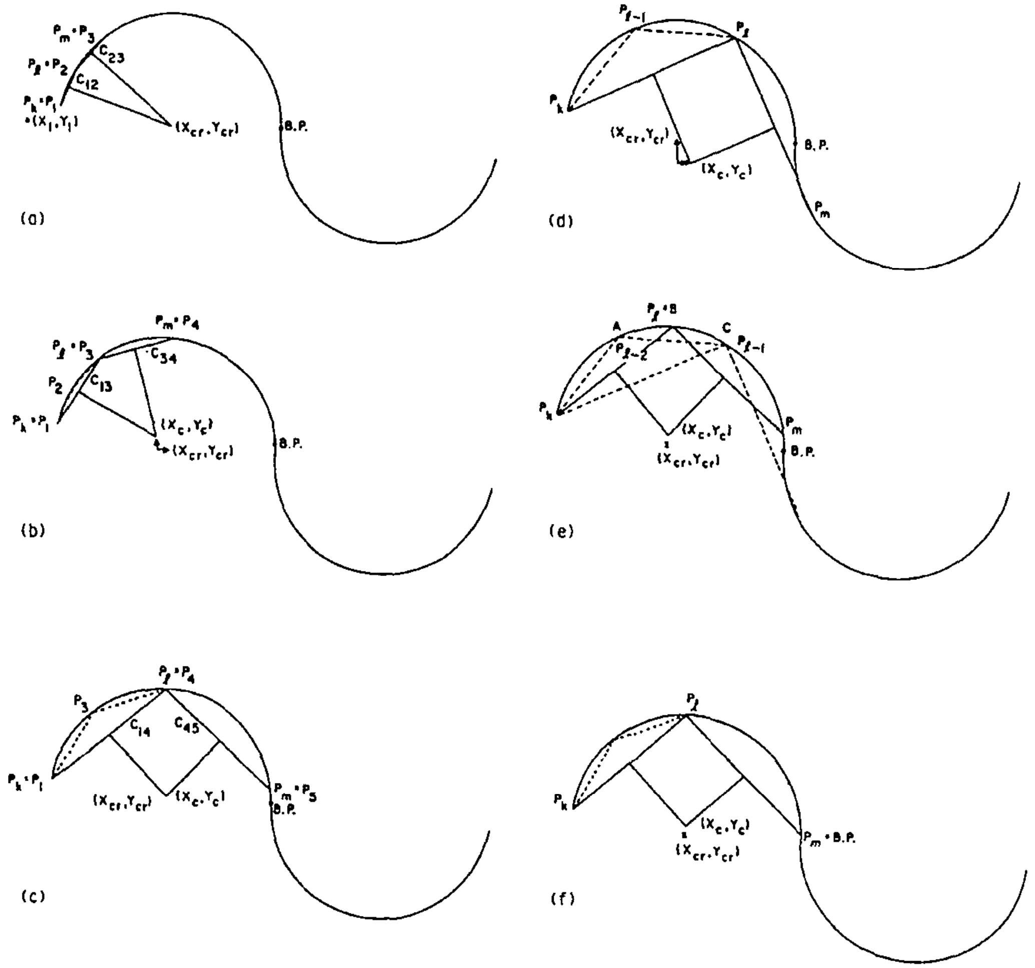


FIGURE 13.31 Breakpoint detection process. (a) Construction to obtain (x_{cr}, y_{cr}); (b, c) when error in center coordinates computation is less than zero; (d) when error greater than $(0 + 0.5)$; (e) backtracking of P_l and P_m; (f) P_m at breakpoint. (From Honnenahalli and Bow, 1988.)

It has been found that after the breakpoint is overshot for the first time, it typically takes less than five backward and forward movements to locate the actual breakpoint. After the breakpoints have been located accurately, the curve can be represented by concatenated curve segments, each of which can be described by an arc. The representation of a curve can then be expressed as

$$C\colon \mathop{\mathbb{C}}_{i=1}^{n} C(i) \tag{13.14}$$

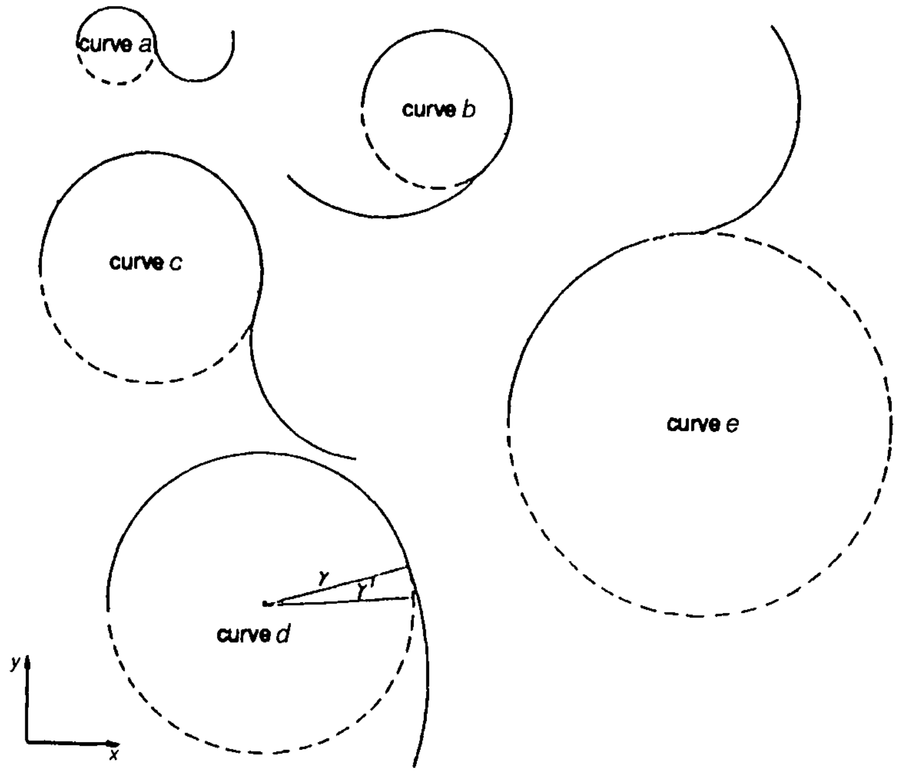


FIGURE 13.32 Shapes of different radii used for the algorithm evaluation. (From Honnenahalli and Bow, 1988.)

TABLE 13.1 Guiding-Point Generating Scheme

Trial	P_k	P_l	P_m
1	P_1	P_2	P_3
2	P_1	P_3	P_5
3	P_1	P_5	P_9
4	P_1	P_9	P_{17}
.	.	.	.
.	.	.	.
.	.	.	.

Column 1: All $k = 1$.
Column 2: $P_l = P_m$ of the previous trial.
Column 3: $P_m = 2P_l - 1$.

Source: Honnenahalli and Bow (1988).

where ℂ stands for concatenation of all the circular arcs, and $C(i)$ is the equation for curve segment i and is given by

$$(x_i - x_{Ci})^2 + (y_i - y_{Ci})^2 = R_i^2 \tag{13.15}$$

for all points between $P_{ia}(x_{ia}, y_{ia})$ and $P_{ib}(x_{ib}, y_{ib})$, where $P_{ia}(x_{ia}, y_{ia})$ and $P_{ib}(x_{ib}, y_{ib})$ are the breakpoints corresponding to segment i, while R_i and $P_{Ci}(x_{Ci}, y_{Ci})$ are, respectively, the radius and the center of the ith curve segment. This algorithm has been applied to a wide variety of curves. Results show that most curves (if not any) can be processed successfully with this algorithm and can finally be represented satisfactorily by a sequence of concatenated arcs.

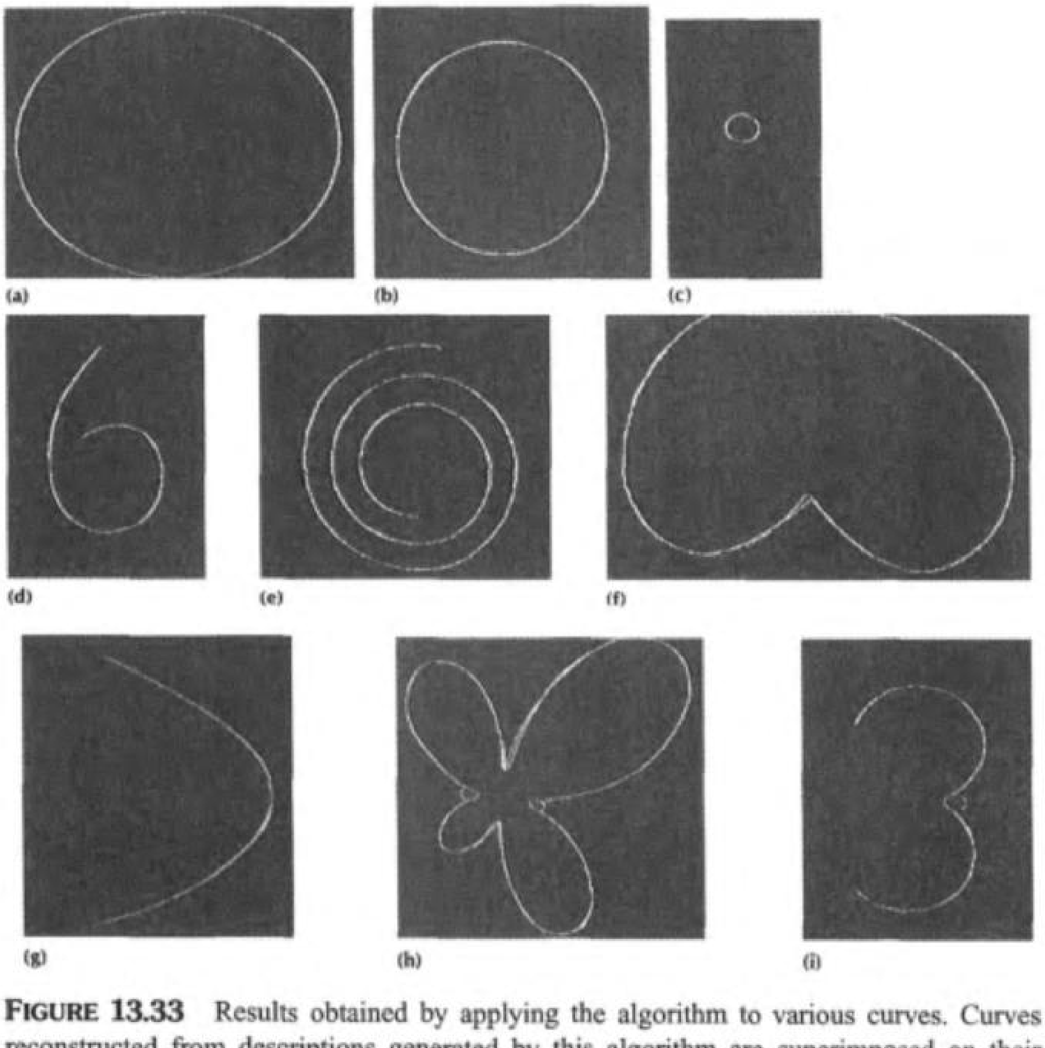


FIGURE 13.33 Results obtained by applying the algorithm to various curves. Curves reconstructed from descriptions generated by this algorithm are superimposed on their original graphics. Very close matchmgs were found in (a) to (e) and (g). Some discrepancies are noted in (f), (h), and (i). (From Bow et al., 1988.)

Both curve graphics on paper-based documents and curve silhouettes from three-dimensional objects were input to our system either through an AST Turboscan scanner or a high-resolution CCD camera with a frame grabber. A processing sequence followed to extract the curves from the graphics in discrete form: that is, skeletonization of the curve graphics, linking of those data points that have connectivity relationships, and so on. An ordered set of points was then produced for each curve. A breakpoint detection algorithm was then applied to the ordered set of points so as to break the curve down into describable curve segments. Each can be represented by only a few parameters.

Curves reconstructed from their machine-generated descriptions were superimposed on the original graphics. Figure 13.33 shows some results. In these figures the reconstructed curves were overlaid on the original curves for an effectiveness evaluation of this approach. Very close matchings were found.

13.10 IDENTIFICATION OF PARTIALLY OBSCURED OBJECTS

What we have discussed so far is how to describe a shape that is fully observable. But in the real world, two or more overlapped objects are frequently encountered. One example is that of two workpieces on a production line, with one partially overlaid by the other. Partially occluded objects participating in a composite scene are another example. How to identify these two objects from an occluded image is a problem of finding the best fit between two boundary curves. This problem is of central importance in applications of pattern recognition and computer vision. Note that a contour usually represents one object. However, when occlusion occurs, the contour could represent merged boundaries of several objects. Several approaches are coded here to illustrate solutions to such types of problems.

Let us first make the problem simple. The approaches suggested below focus on converting the object boundary (curve) into a form such as shape signature strings, dominant points (landmarks), or sides of polygons that approximate the curve. Then use the least-square-error fit to find the longest matching (i.e., to find the subcurve that appears in the scene as well as in the model object). The reason we follow such a procedure is simply because, in such a situation, no use can be made of the global object characteristics, and therefore stresses are put on the local properties of the curves.

In Wolfson's (1990) shape signature string approach, a curve (the model and scene curve) is first represented by characteristic strings of real numbers. They are supposed to possess the invariant property in translation and rotation within a local region:

$$C_i \qquad i = 1, 2, \ldots, n \tag{13.16}$$

Curvature is a good shape signature for use with a curve. Therefore, an accumulated angle change $\theta(s_i)$, $i = 1, 2, \ldots, n$, versus path length along the curve plot is first constructed. When the path length is sampled at an equal interval, Δs, we then have

$$\Delta\theta(s_i) = \theta(s_i + \Delta s) - \theta(s_i) \qquad i = 1, 2, \ldots, n \tag{13.17}$$

With the conversion above, the effect of rotation and translation will be eliminated. Similar numerical strings will represent similar subcurves. With these two shape signature strings as input, find the long substrings (may be several in number) that are common in both strings. If found, keep the starting and ending points of these substrings for later analysis. Each pair of these long-matching substrings corresponds to the subcurves in the cartesian coordinate plane.

Then transform one of subcurves relative to the other in terms of rotation and translation to give the best least-square-error fit of one curve to the other by minimizing

$$\sum_{j=1}^{n} |T\mathbf{u}_j - \mathbf{x}|^2 \tag{13.18}$$

where T represents the rotational and translational transformations operated on one of the curves, which is usually the curve to be matched with the model. This is to align these two curves along their matching subportions. Figure 13.34 shows some results obtained with this method on the overlapping scene of pliers and scissors.

Ansari and Delp (1990) tried to represent each object by a set of landmarks that possess shape attributes. Examples of landmarks (sometimes called dominant points) are corners, holes, protrusions, high curvature points, and so on. Their approach works as follows. Given a scene consisting of partially occluded objects, landmark matching (or dominant point matching) is to be performed between the two landmarks, one from the model object and the other from the scene. These landmarks should be ordered in a sequence that corresponds to consecutive points along the object boundary. Ansari and Delp use a local shape measure called sphericity, which possesses the property that any invariant function under a similarity transformation must be a function of the sphericity. Figure 13.35 shows the landmarks obtained from the cardinal curvature points. When using this approach to match landmarks in a model with those in a scene, it is required that at least three landmarks in a scene that correspond to those of the model must be detectable. In addition, part of the sequential order of the detectable landmarks must also be preserved. Ansari and Delp claimed that as long as more than half of its landmarks in the scene can be detected in the correct sequential order, the object in a scene can be recognized. Interested readers may refer to Ansari and Delp (1990) for details of their approach.

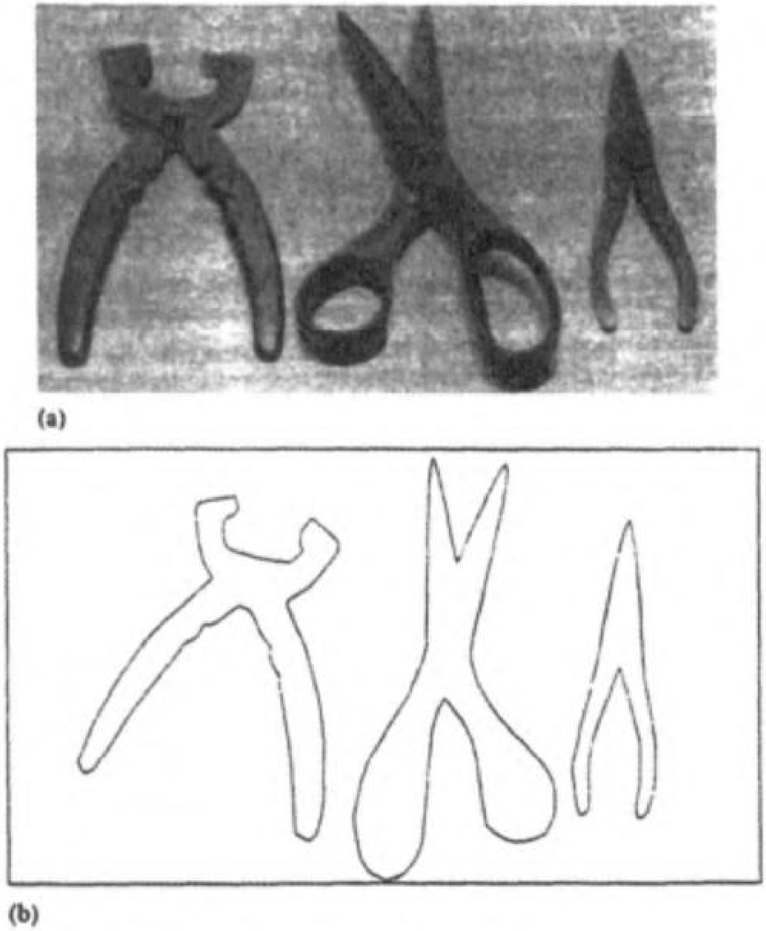


FIGURE 13.34 Model objects and their boundary curves. (From Wolfson, 1990.)

13.11 RECOGNIZING PARTIALLY OCCLUDED PARTS BY THE CONCEPT OF SALIENCY OF A BOUNDARY SEGMENT

The problem of recognizing an object from a partially occluded boundary image is of considerable interest in industrial automation. When objects are occluded, many shape recognition methods that use global information will fail. In this section we describe a method that combines the ideas we have discussed in Chapters 3 and 4 with digital image processing. First, the image is processed and thinned. Then useful features along the boundary are extracted and the effectiveness of the features evaluated. Weighting coefficients are applied to the individual features. Finally, a discriminant function is computed to determine the category to which the object belongs.

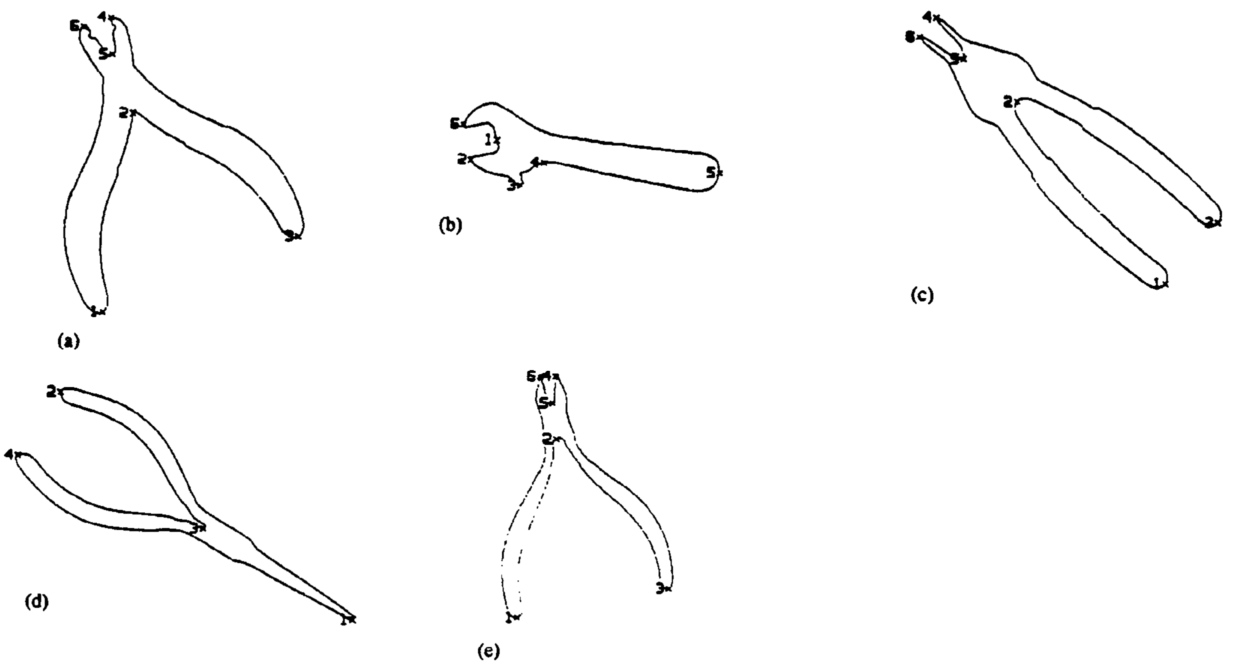


FIGURE 13.35 Landmarks of a library of objects obtained from the cardinal curvature points. (a) Wire stripper; (b) wrench; (c) specialty plier; (d) needle-nose plier; (e) wire cutter. (From Ansari and Delp, 1990.)

As we discussed in previous chapters, the boundary of an object carries a lot of information about its shape. So the traditional method focuses on this information, and a match is to be made between the boundary of the image and the boundary of the model. The computation required by this method is, no doubt, excessive. If the objects are intermixed or partially occluded, the aforementioned match will no longer be realized.

In the approach we are discussing here, the concept of saliency of a boundary segment is introduced. Instead of finding a best match of a template to the boundary as a whole, the matching is done on the subtemplates (i.e., the salient features of the boundary), which can distinguish the object to which it belongs from other objects that might be present. To make the method more effective, choice of the salient features is important. Their choice depends greatly on the set of objects being considered. When the visible part of the object matches some set of subtemplates with enough combined saliency, the partially occluded object can then be determined. In the extraction of the subtemplates, the boundary image is first transformed from its x–y coordinate system to a θ–s coordinate representation, with θ (the angle of slope of the boundary image segments) plotted against s (the arc length along the boundary). A matching process is conducted on the θ–s representation of subtemplates and boundary image segments of the object.

Take an example to illustrate the matching process. Figure 13.36 shows a boundary image containing a partially occluded part between b_k and b_l, with its θ–s representation traced counterclockwise along the boundary. A heavy line in the complete template shown in Figure 13.37 shows the subtemplate chosen as the feature for matching, which is not occluded. The matching can be thought of as a minimization problem with $\theta_{\tau i}(s)$, the θ–s graph of subtemplate τ_i, moving along the s axis direction in the θ–s graph of the boundary image $\theta_B(s)$ at regularly spaced pixels, that is, minimizing

$$\gamma_{ij} = \sum_{p=1}^{h}[\theta_{Bj}(s_p) - \theta_{\tau i}(s_p) + \theta_{ij}]^2 \tag{13.19}$$

where h is the number of pixels in the subtemplate τ_i and in the particular boundary interval B_j. θ_{ij} in (13.19) is a variable added to make γ_{ij} to be a minimum. It can easily be shown that it is the difference between the average slope of the boundary segment and that of the subtemplate, that is, $\bar{\theta}_{ij} - \bar{\theta}_{\tau i}$, as shown in Figure 13.38. The shift in θ corresponds to an angular rotation of the subtemplate to the average orientation of the boundary image segment. Define a matching coefficient

$$C_{ij} = \frac{1}{1+\gamma_{ij}^{*}} \qquad 0 \le C_{ij} \le 1 \tag{13.20}$$

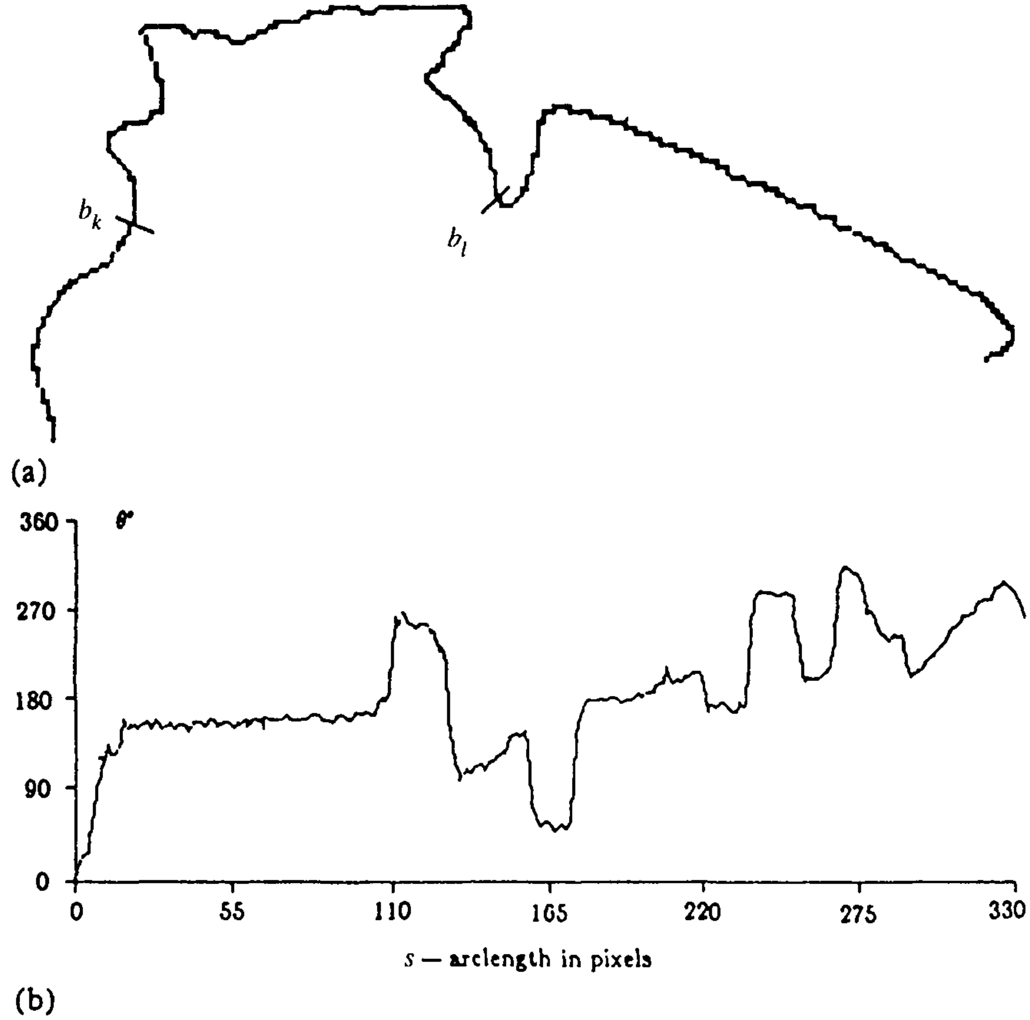


FIGURE 13.36 Boundary image and its θ–s representation. (From Turney et al., 1985.)

where γ_{ij}^* is the minimum value of γ_{ij}. A C_{ij} value close to 1 implies a good match; otherwise, a poor match. Select the weighting coefficient w_i properly with each C_{ij} to match the template τ_t. If the combined saliency of some set of subtemplates is over a certain value (which is prespecified), the partially occluded object is said to be identified.

Figure 13.39 is taken from part of a figure in Turney et al. (1985) to show the computer result in recognizing the two keys when one occludes the other. This approach is useful in industrial inspection, where the types of parts are almost always known a priori.

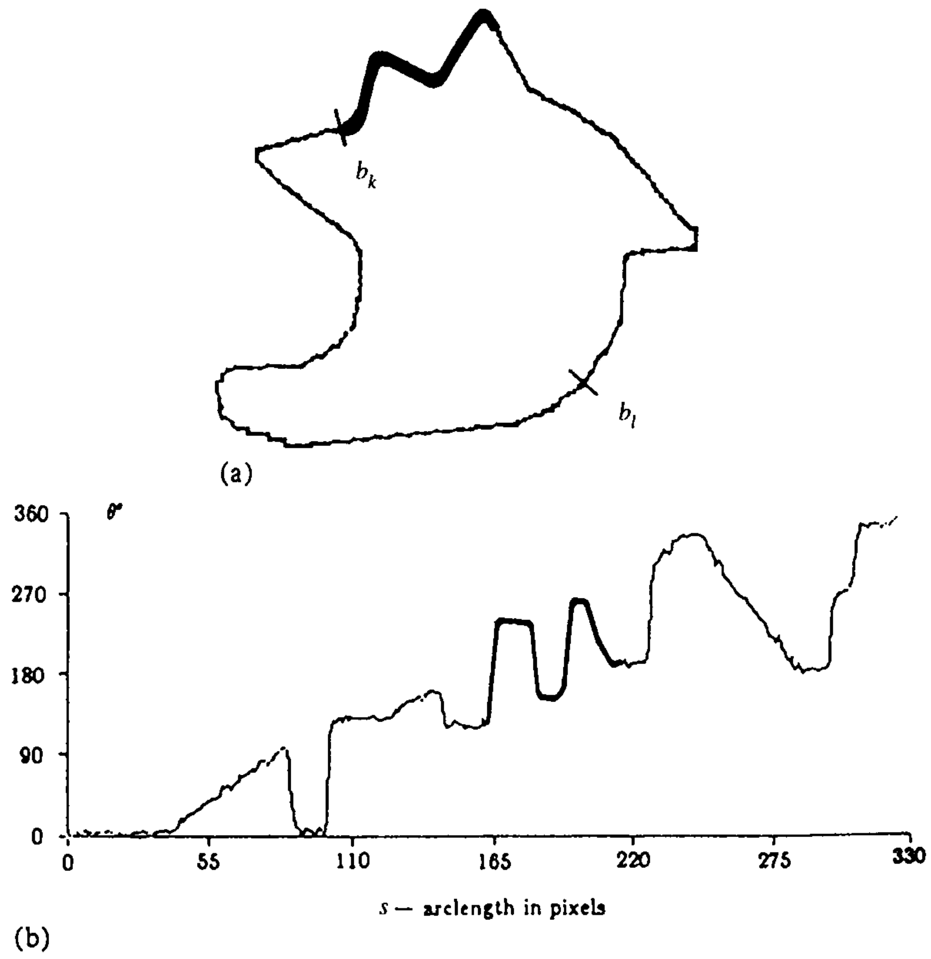


FIGURE 13.37 Template and its θ–s representation. (From Turney et al., 1985.)

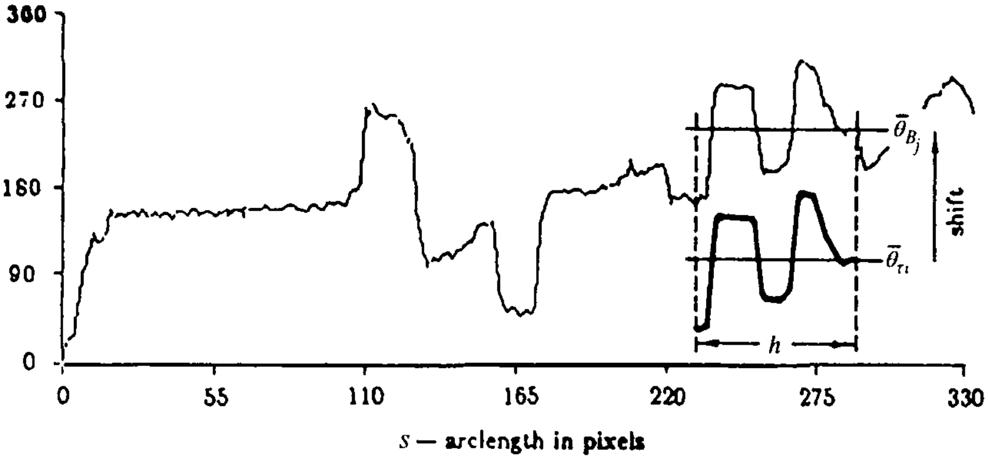


FIGURE 13.38 Matching in θ–s space. (From Turney et al., 1985.)

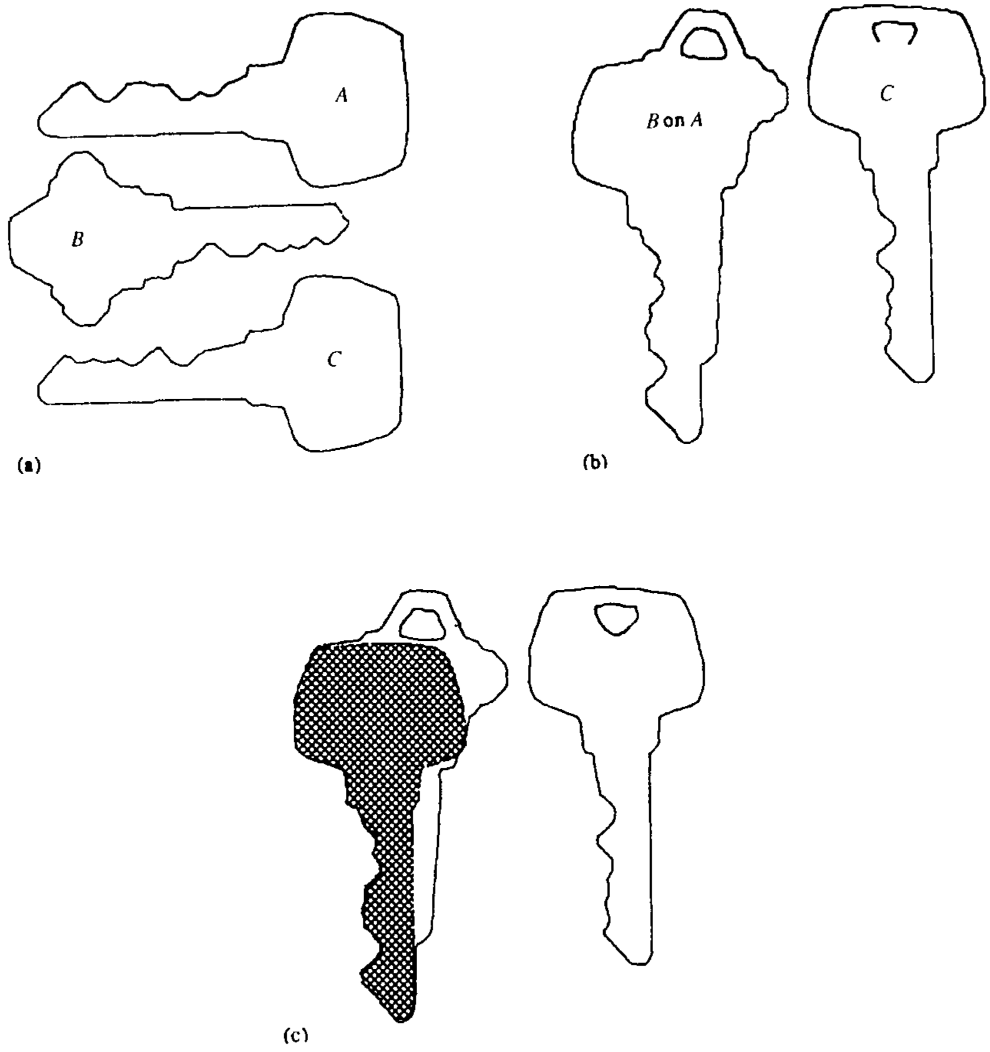


FIGURE 13.39 Three keys experiment. (From Turney et al., 1985.)

PROBLEMS

13.1 For an image of 512 × 512 pixels, determine the total number of levels and the total number of nodes needed for the entire quadtree representation of the image.

13.2 Following the method suggested in Sec. 13.2, delineate the medial axis from the disjoint boundary segments of the two figures shown in Figure P13.2.

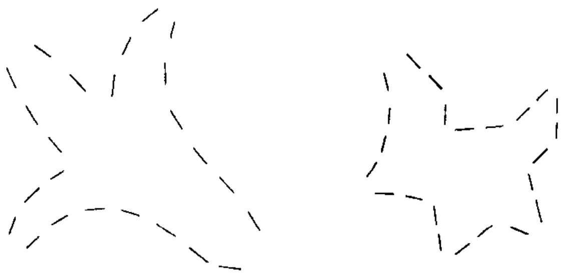


FIGURE P13.2

13.3 Find the medial axis of (a) a circle; (b) a square; (c) an equilateral triangle; (d) a submedian chromosome; and (e) a telocentric chromosome.

13.4 A chain code representation of a boundary can be obtained with the following algorithm:

1. Start at any boundary pixel point.
2. Find the nearest edge pixel and code its orientation.
3. Continue until there are no more boundary pixel points.

Write a program for its implementation and run the program for the contour shape shown in Figure P13.4.

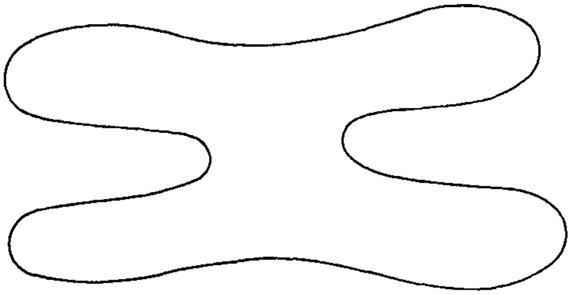


FIGURE P13.4

13.5 Following the algorithms suggested in Sec. 13.9, write a program to determine all the breakpoints and represent the closed curve shown in Figure P13.5 with concatenated arcs.

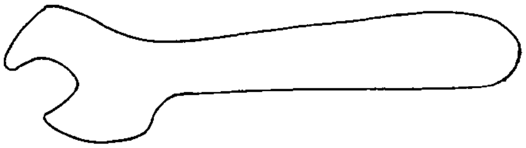


FIGURE P13.5

14

Transforms and Image Processing in theTransform Domain

The necessity of performing pattern recognition problems by computer lies on the large set of data to be dealt with. The preprocessing of large volumes of these data into a better form will be very helpful for more accurate pattern recognition. A two-dimensional image is a very good example of problems with a large data set. The preprocessing of an image can be carried out in one of two domains: the spatial domain and the transform domain. Figure 14.1 shows the general configuration of digital image processing in the spatial domain or in the Fourier domain and the relationship between them.

When an image is processed in the spatial domain, the processing of digitized image is carried out directly either by point processing or by neighborhood processing for enhancement or restoration. But if an image is to be processed in the transform domain, the digitized image will first be transformed by discrete Fourier transform (DFT) or fast Fourier transform (FFT). Processing will then be carried out on the image in the transform domain. After the image is processed, an inverse operation of the FFT [called the inverse fast Fourier transform (IFFT)] will be carried out on the result to transform it back to an image in the spatial domain.

Image transform and image inverse transform are two intermediate processes. They are linked such that image processing can be carried out in the transform domain instead of in the spatial domain. In so doing, three objectives are expected:

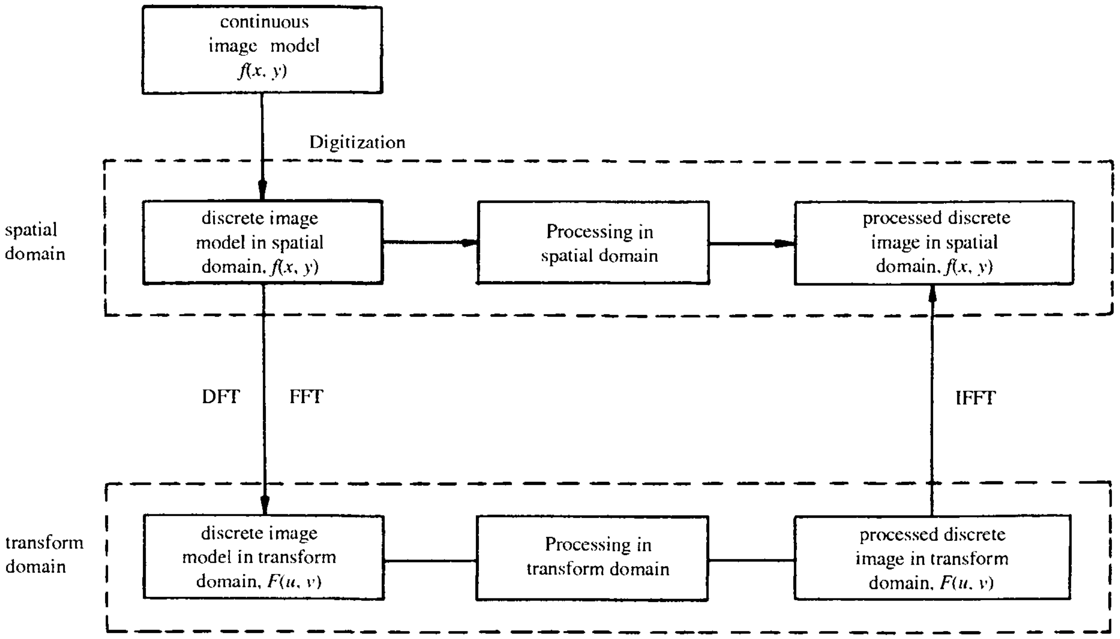

continuous image model $f(x, y)$
Digitization
spatial domain
discrete image model in spatial domain, $f(x, y)$
Processing in spatial domain
processed discrete image in spatial domain, $f(x, y)$
DFT
FFT
IFFT
transform domain
discrete image model in transform domain, $F(u, v)$
Processing in transform domain
processed discrete image in transform domain, $F(u, v)$

FIGURE 14.1 Schematic diagram of image processing.

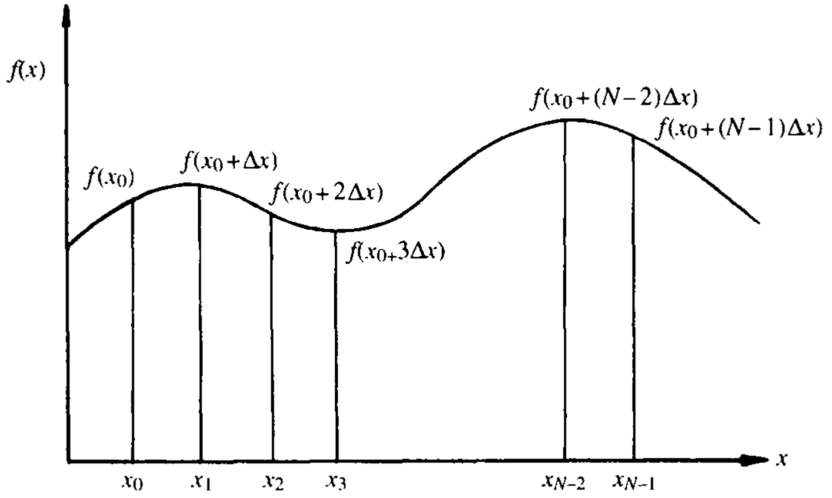


FIGURE 14.2 Discretization of a continuous function.

1. Processing might be facilitated in the transform domain for some operations, such as convolution and correlation.
2. Some features might be more obvious and easier to extract in that domain.
3. Data compression might be possible, thus reducing the on-line and off-line storage requirements and also the bandwidths requirements in transmission.

14.1 FORMULATION OF THE IMAGE TRANSFORM

If a continuous function $f(x)$ as shown in Figure 14.2 is discretized into N samples Δx apart, such as $f(x_0), f(x_0 + \Delta x), f(x_0 + 2\Delta x), \ldots, f(x_0 + (N - 1)\Delta x)$, the function $f(x)$ can be expressed as

$$f(x) = f(x_0 + x\Delta x) \qquad x = 0, 1, \ldots, N - 1 \tag{14.1}$$

With this in mind we have the discrete Fourier transform pair*

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N} \qquad u = 0, 1, \ldots, N - 1 \tag{14.2}$$

$$f(x) = \sum_{u=0}^{N-1} F(u)e^{j2\pi ux/N} \qquad x = 0, 1, \ldots, N - 1 \tag{14.3}$$

*A proof of these results is very lengthy and is beyond the scope of this book. Details of the derivation can be found in Brigham (1974).

where N is the number of samples taken from the function curve. This corresponds to the transform pair for a continuous one-dimensional function:

$$F(j\omega) = \int_{-\infty}^{\infty} e^{-j\omega t} f(t)\, dt \tag{14.4}$$

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{j\omega t} F(j\omega)\, d\omega \tag{14.5}$$

Extending this to two-dimensional functions, we have the Fourier transform pair for a continuous function as

$$F(u, v) = \iint\limits_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)}\, dx\, dy \tag{14.6}$$

and

$$f(x, y) = \iint\limits_{-\infty}^{\infty} F(u, v) e^{j2\pi(ux+vy)}\, du\, dv \tag{14.7}$$

where x and y are spatial coordinates and $f(x, y)$ is the image model; while u and v are the spatial frequencies and $F(u, v)$ is the frequency spectrum. The corresponding discrete transform pair for the two-dimensional function will be

$$F(u, v) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux+vy)/N}$$

$$u, v = 0, 1, 2, \ldots, N = 1 \tag{14.8}$$

and

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux+vy)/N}$$

$$x, y = 0, 1, 2, \ldots, N - 1 \tag{14.9}$$

The frequency spectrum $F(u, v)$ can be computed if the appropriate values of x, y, u, v, and $f(x, y)$ are substituted into Eq. (14.8). Clearly, the computation is rather cumbersome, and spectrum computation by computer is suggested when N becomes large.

There are quite a few transformation techniques available, among them the Fourier transform represented by Eq. (14.8). If the exponential factor

$e^{-j2\pi(ux+vy)/N}$ is replaced by a more general function, $g(x, y; u, v)$, Eq. (14.8) becomes

$$F(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) g(x, y; u, v) \tag{14.10}$$

where $F(u, v)$ is an $N \times N$ transformed image array if $f(x, y)$ is an $N \times N$ array of numbers used to represent the discrete image model as follows:

$$[f] = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0, N-1) \\ f(1,0) & & \dots & f(1, N-1) \\ \vdots & & & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1, N-1) \end{bmatrix} \tag{14.11}$$

The function $g(x, y; u, v)$ in Eq. (14.10) is the forward transform kernel. Correspondingly, we can write its inverse transform:

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) h(x, y; u, v) \tag{14.12}$$

where $h(x, y; u, v)$ is the inverse transform kernel. For the case of Fourier transform, the inverse transform kernel is $e^{j2\pi(ux+vy)}$. Equations (14.10) and (14.12) form a transform pair. The transformation is unitary* if the following orthonormality conditions are met:

$$\sum_u \sum_v (x, y; u, v) g^*(x_0, y_0; u, v) = \delta(x - x_0, y - y_0) \tag{14.13}$$

$$\sum_u \sum_v h(x, y; u, v) h^*(x_0, y_0; u, v) = \delta(x - x_0, y - y_0) \tag{14.14}$$

$$\sum_x \sum_y g(x, y; u, v) g^*(x, y; u_0, v_0) = \delta(u - u_0, v - v_0) \tag{14.15}$$

$$\sum_x \sum_y h(x, y; u, v) h^*(x, y; u_0, v_0) = \delta(u - u_0, v - v_0) \tag{14.16}$$

where the superscript * denotes a complex conjugate and the Dirac delta function is

$$\delta = \begin{cases} \infty & \text{at } x = x_0, y = y_0 \\ 0 & \text{elsewhere} \end{cases} \tag{14.17}$$

or

$$\delta = \begin{cases} \infty & \text{at } u = u_0, v = v_0 \\ 0 & \text{elsewhere} \end{cases} \tag{14.18}$$

*A is said to be a *unitary matrix* if the matrix inverse is given by A^{*T}. A real unitary matrix is called an *orthogonal matrix*. For such a matrix, $A^{-1} = A^T$.

These can easily be proved by substituting $e^{-j2\pi(ux+vy)/N}$ and $e^{+j2\pi(ux_0+vy_0)/N}$, respectively, for $g(x, y; u, v)$ and $g^*(x_0, y_0; u, v)$ in Eq. (14.13).

Two-dimensional transformation is a very tedious mathematical operation, and therefore lots of effort has been spent in simplifying it. The separability property of the transformation is very effective for this purpose. The transformation is "separable" if its kernel can be written as

$$g(x, y; u, v) = g_{\text{col}}(x, u)g_{\text{row}}(y, v) \qquad \text{forward transform} \tag{14.19}$$

$$h(x, y; u, v) = h_{\text{col}}(x, u)h_{\text{row}}(y, v) \qquad \text{inverse transform} \tag{14.20}$$

A separable unitary transform can thus be computed in two steps:

1. Transform column-wise or one-dimensional transform along each column of the image $f(x, y)$:

   $$P(u, y) = \sum_{x=0}^{N-1} f(x, y)g_{\text{col}}(x, u) \tag{14.21}$$

   where $g_{\text{col}}(x, u)$ is the forward column transform kernel and is $e^{-j2\pi ux/N}$ for the Fourier transform.
2. Transform row-wise, or one-dimensional unitary transform along each row of $P(u, y)$:

   $$F(u, v) = \sum_{y=0}^{N-1} P(u, y)g_{\text{row}}(y, v) \tag{14.22}$$

   where $g_{\text{row}}(y, v)$ is the forward row transform kernel and is $e^{-j2\pi vy/N}$ for the Fourier transform. Thus a two-dimensional transform may be computed in two steps, each being a one-dimensional transform. If an efficient and effective one-dimensional transform algorithm is set up, it can be used repeatedly for a two-dimensional transformation.

14.2 FUNCTIONAL PROPERTIES OF THE TWO-DIMENSIONAL FOURIER TRANSFORM

As mentioned in Section 14.1, the Fourier transform for a continuous function is

$$F(u, v) = \iint_{-\infty}^{\infty} f(x, y)e^{-j2\pi(ux+vy)}\, dx\, dy \tag{14.23}$$

This transform function is in general complex and consists of two parts, such as

$$F(u, v) = \text{Real}(u, v) + j\ \text{Imag}(u, v) \tag{14.24}$$

or in polar form,

$$F(u, v) = |F(u, v)|\phi(u, v) \tag{14.25}$$

where $F(u, v) = [\text{Real}^2(u, v) + \text{Imag}^2(u, v)]^{1/2}$ = Fourier transform of $f(x, y)$,

$$\phi(u, v) = \tan^{-1}\left[\frac{\text{Imag}(u, v)}{\text{Real}(u, v)}\right] = \text{phase angle}$$

and $|F(u, v)|^2 = E(u, v)$ = energy spectrum of $f(x, y)$.

The inverse transform of $F(u, v)$ gives $f(x, y)$. The inverse transform forms a pair with Eq. (14.23).

$$f(x, y) = \mathscr{F}^{-1}[F(u, v)] = \iint\limits_{-\infty}^{\infty} F(u, v)e^{j2\pi(uv+vy)}\, du\, dv \tag{14.26}$$

This transform pair can be shown to exist if $f(x, y)$ is continuous and integrable and $F(u, v)$ is also integrable. If we have a simple rectangular bar object with uniform intensity, that is, $f(x, y) = f_0$, shown shaded on Figure 14.3, its Fourier spectrum can then be computed accordingly to Eq. (14.23) as follows:

$$\begin{aligned} F(u, v) &= \iint\limits_{-\infty}^{\infty} f(x, y)e^{-j2\pi(ux+vy)}\, dx\, dy \\ &= f_0 \int_0^{x_0} e^{-j2\pi ux} dx \int_0^{y_0} e^{-j2\pi vy}\, dy \\ &= f_0 \frac{(e^{-j\pi ux_0} - e^{j\pi ux_0})e^{-j\pi ux_0}}{-j2\pi u} \times \frac{(e^{-j\pi vy_0} - e^{j\pi vy_0})e^{-j\pi vy_0}}{-j2\pi v} \\ &= f_0 x_0 y_0 \frac{\sin \pi ux_0}{\pi ux_0} e^{-j\pi ux_0} \frac{\sin \pi vy_0}{\pi vy_0} e^{-j\pi vy_0} \end{aligned} \tag{14.27}$$

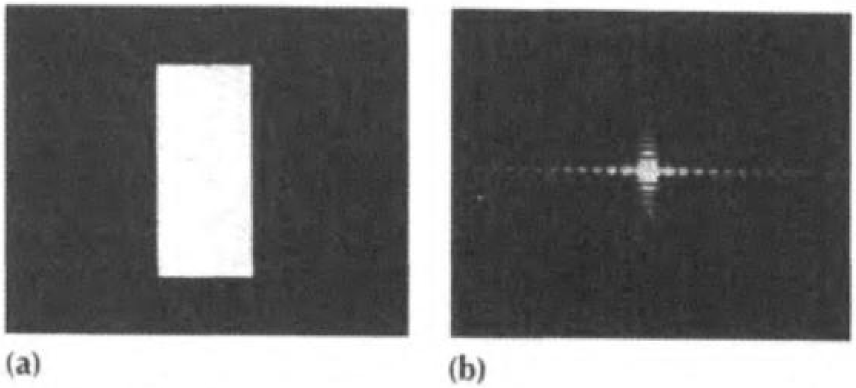


FIGURE 14.3 Fourier spectrum of a simple rectangular bar object with uniform intensity: (a) object; (b) spectrum.

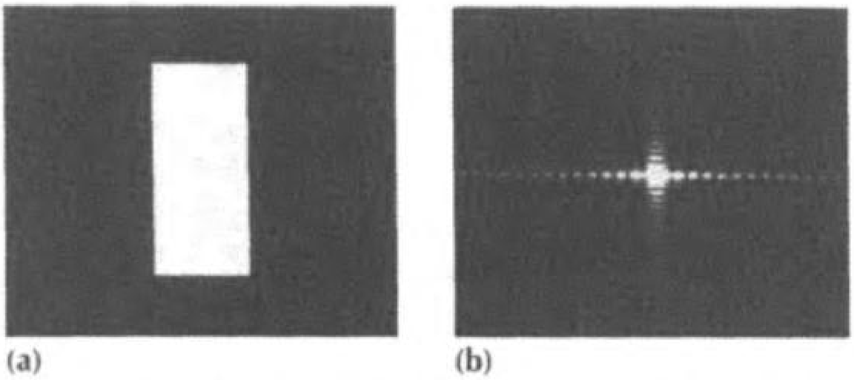


FIGURE 14.4 Computer printout of the Fourier spectrum of a simple vertical rectangular bar object with uniform intensity. (a) object; (b) spectrum.

or

$$F(u, v) = f_0 x_0 y_0 \operatorname{sinc}(\pi u x_0) e^{-j\pi u x_0} \operatorname{sinc}(\pi v y_0) e^{-j\pi v y_0} \tag{14.28}$$

if $\operatorname{sinc}(\pi u x_0)$ and $\operatorname{sinc}(\pi v y_0)$ substitute, respectively, for $(\sin \pi u x_0)/\pi u x_0$ and $(\sin \pi v y_0)/\pi v y_0$. Figure 14.3b shows the plot of the intensity of the spectrum $F(u, v)$, from which we can see clearly that the spectrum in the intensity plot varies as a sinc function.

Some additional examples of two-dimensional functions and their spectra are shown in Figures 14.4 to 14.7. The same number of pixels in the x and y

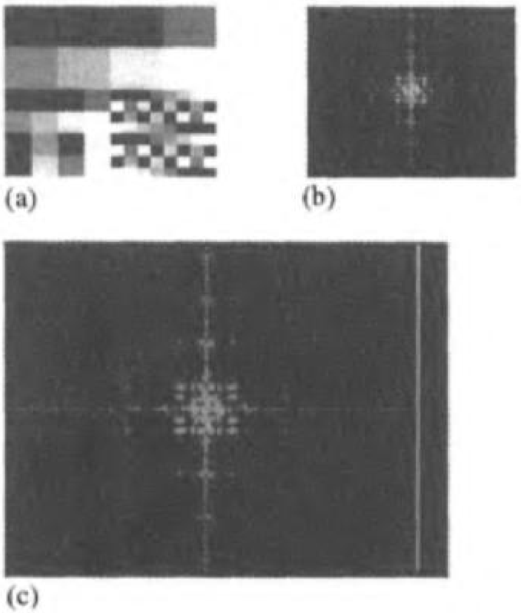


FIGURE 14.5 Computer printout of the Fourier spectrum of a simpie regular pattern object. (a) Object; (b) its spectrum; and (c) center portion of the spectrum enlarged.

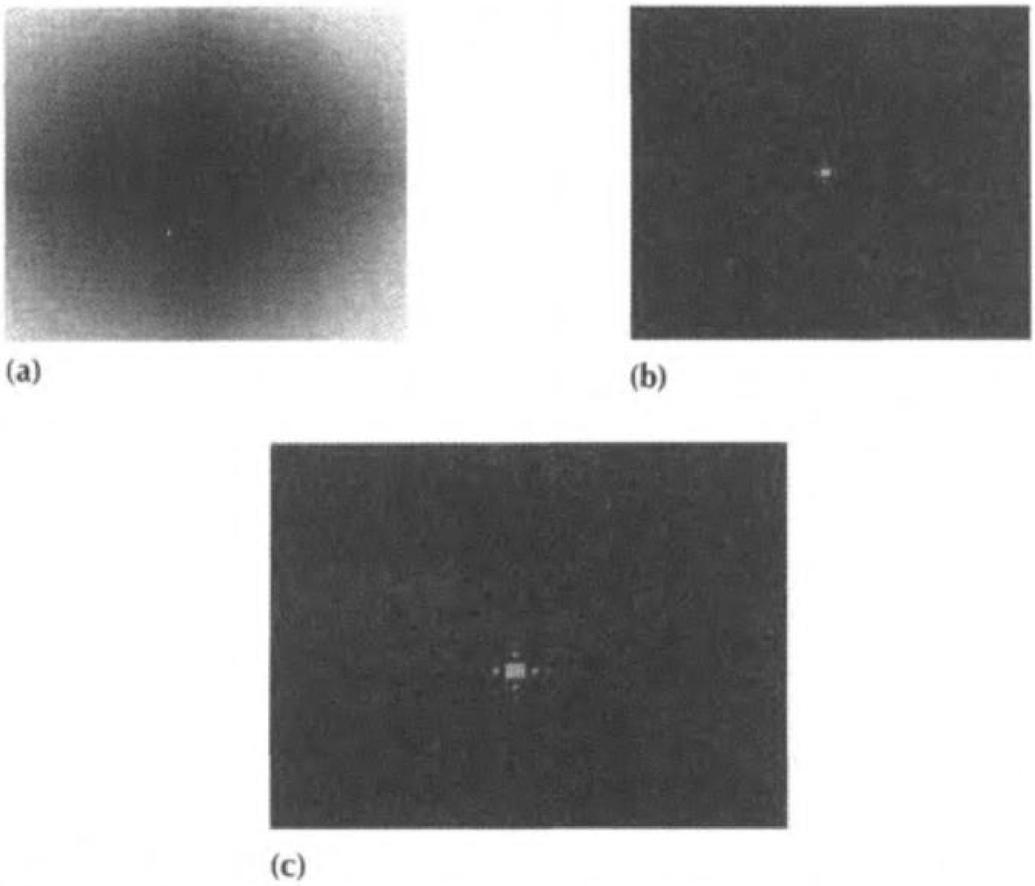


FIGURE 14.6 Computer printout of the Fourier spectrum of a simple image with gaussian distributed intensity. (a) Image; (b) its Fourier spectrum; and (c) center portion of the spectrum enlarged.

directions of the images and their spectra are the same, but due to the imperfection of the monitor (i.e., not exactly the same unit length of a pixel in the x and y directions), the resulting images and spectra are flattened, as shown in Figures 14.4 to 14.7. The same applies to the later images.

Following are some properties of the Fourier transform that are worthy of discussion.

14.2.1 Kernel Separability

The Fourier transform given in Eq. (14.23) can be expressed in separate form as follows:

$$F(u, v) = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(x, y) e^{-j2\pi ux} \, dx \right] e^{-j2\pi vy} \, dy \tag{14.29}$$

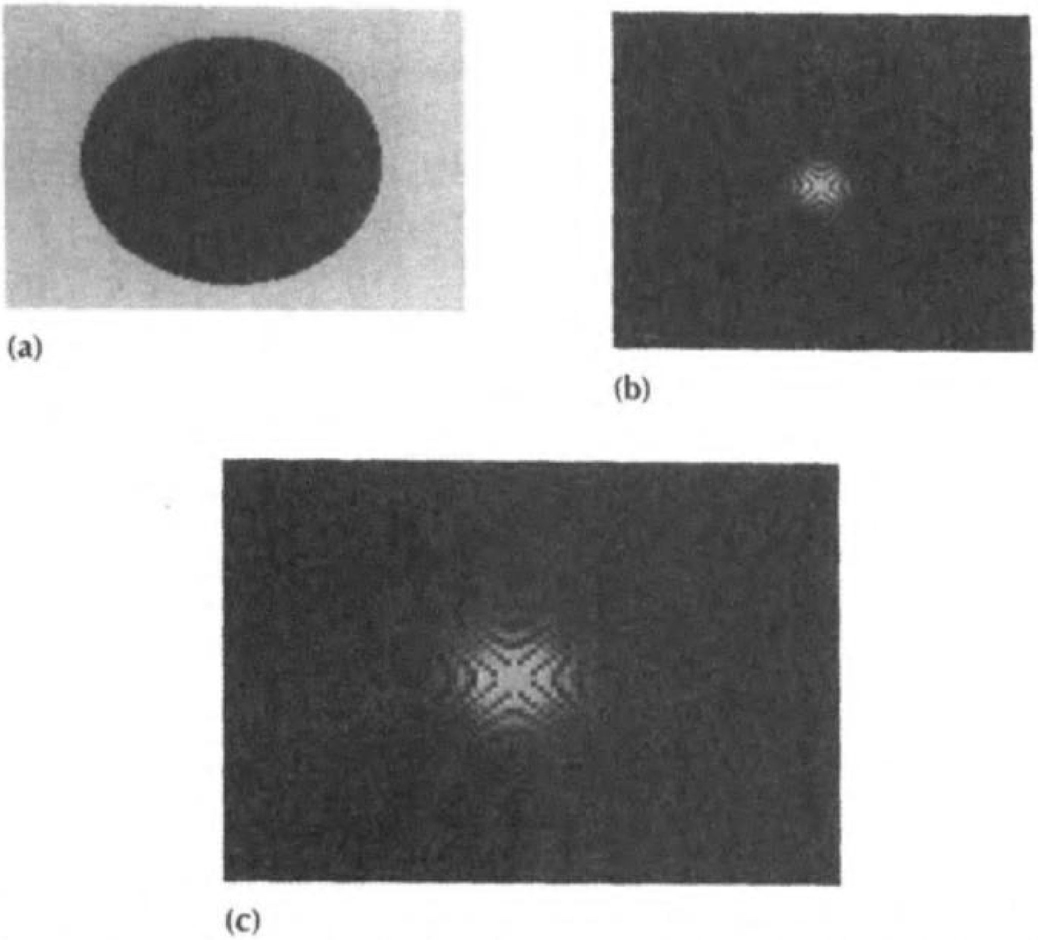


FIGURE 14.7 Computer printout of the Fourier spectrum of a simple object with uniform intensity. (a) Object; (b) its Fourier spectrum; and (c) center portion of the spectrum enlarged.

The integral in brackets is $F_y(u, y)$ (row-wise transform). $F(u, v)$ can then be expressed as

$$F(u, v) = \int_{-\infty}^{\infty} F_y(u, y)e^{-j2\pi vy}\, dy \tag{14.30}$$

or

$$F(u, v) = \int_{-\infty}^{\infty} F_x(x, v)e^{-j2\pi ux}\, dx \tag{14.31}$$

where

$$F_x(x, v) = \int_{-\infty}^{\infty} f(x, y)e^{-j2\pi vy}\, dy \text{ (column-wise transform)}$$

The principal significance of the kernel separability is that a two-dimensional Fourier transform can be separated into two computational steps, each a one-dimensional Fourier transform—which is much less complicated then the two-dimensional transform. Thus

$$\mathscr{F}(u, v) = \mathscr{F}_x\{\mathscr{F}_y[f(x, y)]\} \tag{14.32}$$

or

$$\mathscr{F}(u, v) = \mathscr{F}_y\{\mathscr{F}_x[f(x, y)]\} \tag{14.33}$$

where $\mathscr{F}_x$ and $\mathscr{F}_y$ represent, respectively, the column-wise and row-wise transformations.

This is also true for the inverse Fourier transform. Since the same one-dimensional Fourier transform is employed in these two steps, more effort can be concentrated on the design of the algorithm to make it more effective.

14.2.2 Linearity

The Fourier transform is a linear operator and possesses distributivity and scaling properties. Thus

$$\begin{aligned}\mathscr{F}[a_1 f_1(x, y) + a_2 f_2(x, y)] &= a_1\mathscr{F}[(f_1(x, y)] + a_2\mathscr{F}[f_2(x, y)]\\ &= a_1 F_1(u, v) + a_2 F_2(u, v)\end{aligned} \tag{14.34}$$

$$\mathscr{F}[f(ax, by)] = \frac{1}{|ab|}F\left(\frac{u}{a}, \frac{v}{b}\right) \tag{14.35}$$

Equation (14.35) can be easily proved by direct substitution in Eq. (14.23) of ax, by, u/a, and v/y, respectively, for x, y, u, and v.

14.2.3 Periodicity and Conjugate Symmetry

It can be easily proved by substituting $u + N$ for u or $v + N$ for v in Eq. (14.23) that the Fourier transform and the inverse Fourier transform are periodic and have a period of N. Thus we have

$$F(u, v) = F(u + N, v) = F(u, v + N) = F(u + N, v + N) \tag{14.36}$$

and

$$f(x, y) = f(x + N, y) = f(x, y + N) = f(x + N, y + N) \tag{14.37}$$

By the similar method, the conjugate symmetric property of the Fourier transform that can also be proved such that

$$F(u, v) = F^*(-u, -v)$$

By the same token, the magnitude plot of the Fourier transform is symmetrical to the origin; or

$$|F(u, v)| = |F(-u, -v)| \tag{14.38}$$

14.2.4 Rotation Invariant

If polar coordinates (ρ, θ) and (r, ϕ) are introduced for the rectangular coordinate, the image function and its transform become $f(\rho, \theta)$ and $F(r, \phi)$, respectively. It can easily be shown by direct substitution into the Fourier transform pair that

$$\mathscr{F}\{f(\rho, \theta + \Delta\theta)\} = F(r, \phi + \Delta\phi) \tag{14.39}$$

where $\Delta\phi = \Delta\theta$. That is, when the image function $f(x, y)$ is rotated by $\Delta\theta$, its Fourier transform is also rotated by the same angle $\Delta\theta$ (i.e., $\Delta\phi = \Delta\theta$). In other words, the same angle rotations occur in the spatial and the transform domains. See Figures 14.8 and 14.9 for illustrations.

One more thing we would like to add about the Fourier transform is that most of the information about the image object in the spatial domain concentrates on the central part of the spectrum. The intensity plot of the spectrum contains no positional information about the object, since the phase angle information is discarded in that plot. Complete reconstruction of the original image can be obtained when the real and imaginary spectrum data are included. Figures 14.10a and 14.11a show two identical objects placed in different spatial positions. They give exactly the same spectra as those shown in Figures 14.10b and 14.11b. Figures 14.10c and 14.11c are their zoomed spectra, shown for comparison. Concentration of the spatial image information content in the Fourier spectrum is discussed in Section 14.4.3.

14.2.5 Translation

The Fourier spectra shown in Figure 14.12b for the images shown in part (a) of the figure concentrate on the four corners. This introduces difficulties in obtaining an overall view of the spectra. Figure 14.12c shows the same spectra after the origin of the transformation plane has been translated to the point (u_0, v_0), which is $(N/2, N/2)$.

Let $F(u, v)$ be the Fourier transform of $f(x, y)$. If we multiply $f(x, y)$ by an exponential factor, $\exp[j2\pi(u_0x + v_0y)/N]$, and then take the transform of the

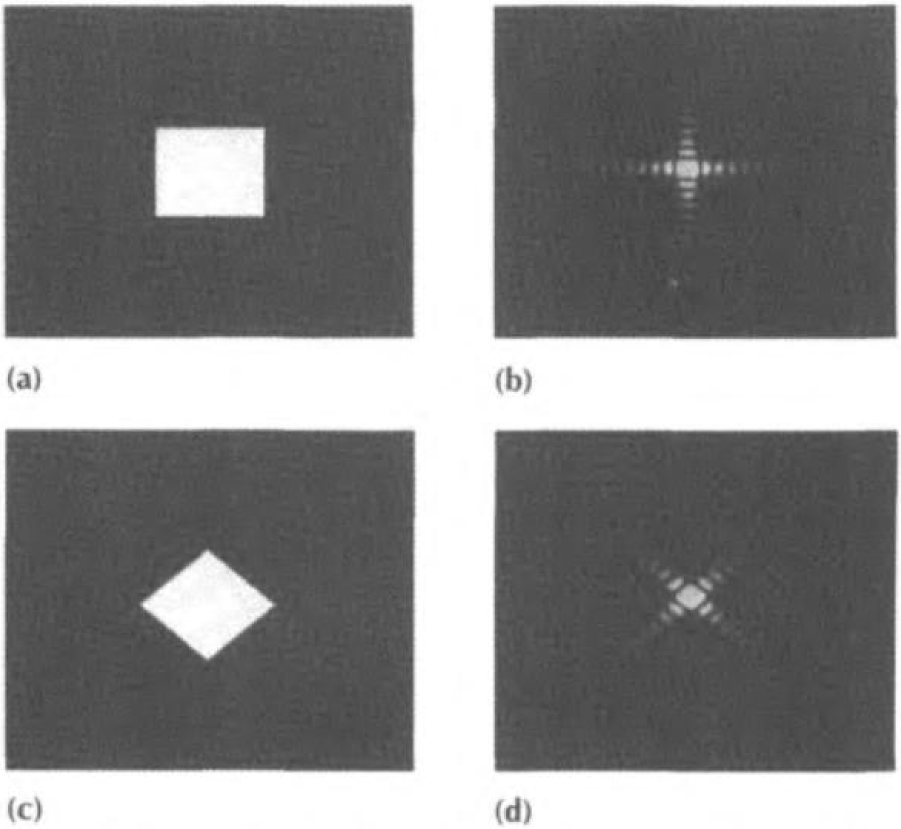


FIGURE 14.8 Rotation invariant property of the Fourier transform. (a) Simple image; (b) its spectrum; (c) same image as in (a), but rotated by an angle; (d) corresponding spectrum of the rotated image shown in (c).

product, the origin of the frequency plane will be shifted to the point (u_0, v_0). Thus

$$f(x, y) \Leftrightarrow F(u, v) \tag{14.40}$$

$$\underbrace{f(x, y)\exp\frac{[j2\pi(u_0x + v_0y)]}{N}}_{\text{in the spatial domain}} \Leftrightarrow \underbrace{F(u - u_0, v - v_0)}_{\text{in the frequency domain}} \tag{14.41}$$

From (14.41) it is noted that $F(u - u_0,\ v - v_0)$ is exactly the same in shape as $F(u, v)$, but shifted by a distance of (u_0, v_0).

The same applies to the inverse transformation operation. If we multiply $F(u, v)$ by an exponential factor $\exp[-j2\pi(ux_0 + vy_0)/N]$ and take the inverse transform of their product, we obtain $f(x - x_0, y - y_0)$. Thus

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v)\exp\left[\frac{-j2\pi(ux_0 + vy_0)}{N}\right] \tag{14.42}$$

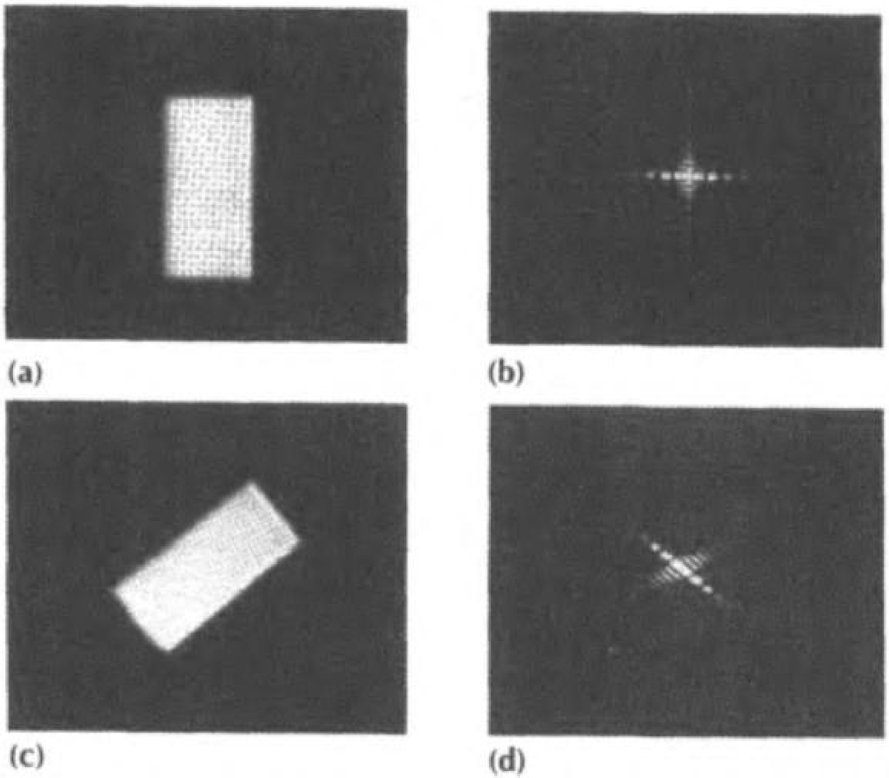


FIGURE 14.9 Rotation invariant property of the Fourier transform. (a) simple image; (b) its spectrum; (c) same image as in (a), but rotated by an angle; (d) corresponding spectrum of the rotated image shown in (c).

That is, the entire spatial image is translated to the new position. This is done by moving the origin of the spatial image to the point (x_0, y_0). Equations (14.41) and (14.42) form a translational transform pair.

To make the spectrum easier to read and analyze, we usually move the center of the frequency plane to $(u_0, v_0) = (N/2, N/2)$ instead of $(u_0, v_0) = (0, 0)$. By so doing, the exponential multiplication factor is

$$\exp\left[\frac{j2\pi(u_0x + v_0y)}{N}\right] = \exp[j\pi(x + y)] \\ = (-1)^{x+y} \tag{14.43}$$

We then have the centering property of the transformation,

$$f(x, y)(-1)^{x+y} \Leftrightarrow F\left(u - \frac{N}{2}, v - \frac{N}{2}\right) \tag{14.44}$$

The double arrows in Eqs. (14.40) to (14.44) indicate the correspondences between the image functions and their Fourier transformations, and vice versa.

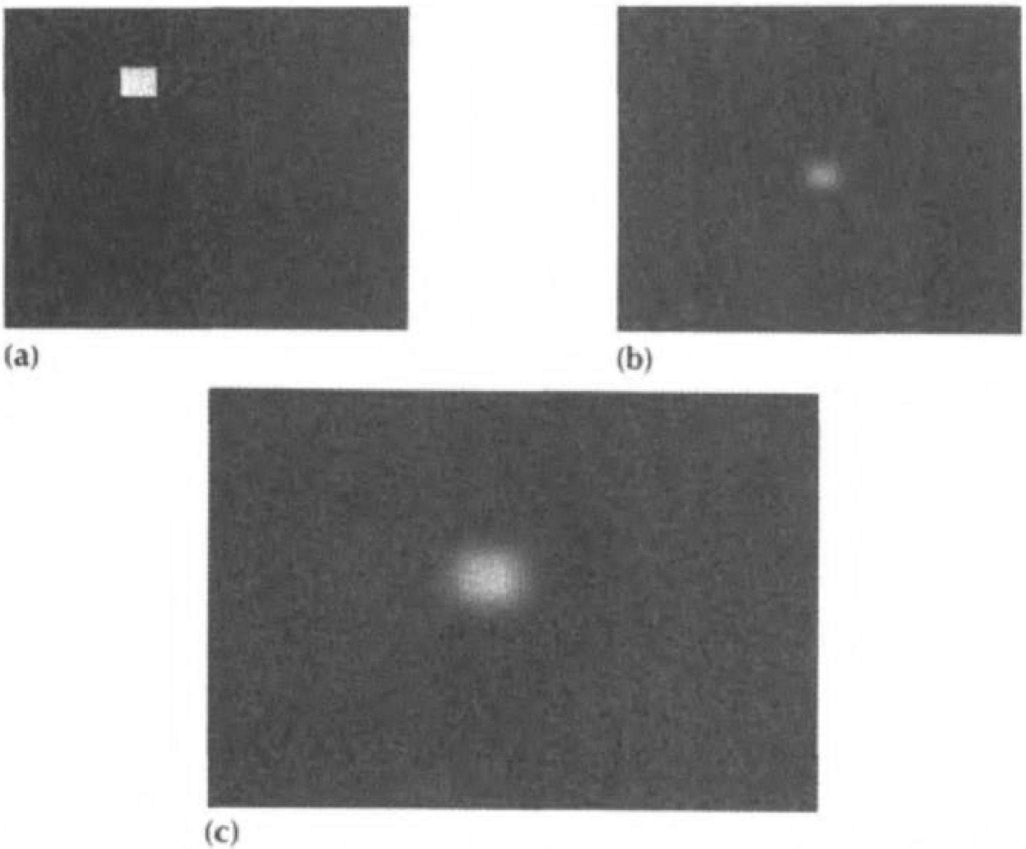


FIGURE 14.10 Computer printout of the Fourier spectrum of a simple object. (a) Object; (b) its Fourier spectrum; and (c) center portion of the spectrum enlarged.

Figures 14.12 and 14.13 give, respectively, several other images and some regular patterns and their corresponding Fourier transforms: part (a) in the figures shows the images in the spatial domain; (b) their Fourier spectra without translation; and (c) their Fourier spectra after shifting to the center of the frequency planes.

It should be pointed out that there is no change in the magnitude of the spectrum even with a shift in $f(x, y)$, since

$$\left| f(x, y) \exp\left[\frac{j2\pi(u_0 x + v_0 y)}{N}\right]\right| = |f(x, y)|$$

and

$$\left| F(u, v) \exp\left[\frac{-j2\pi(u x_0 + v y_0)}{N}\right]\right| = |F(u, v)|$$

and therefore there is no change in the spectrum display except for a translation. This is because the spectrum display is usually limited regarding the display of its magnitude.

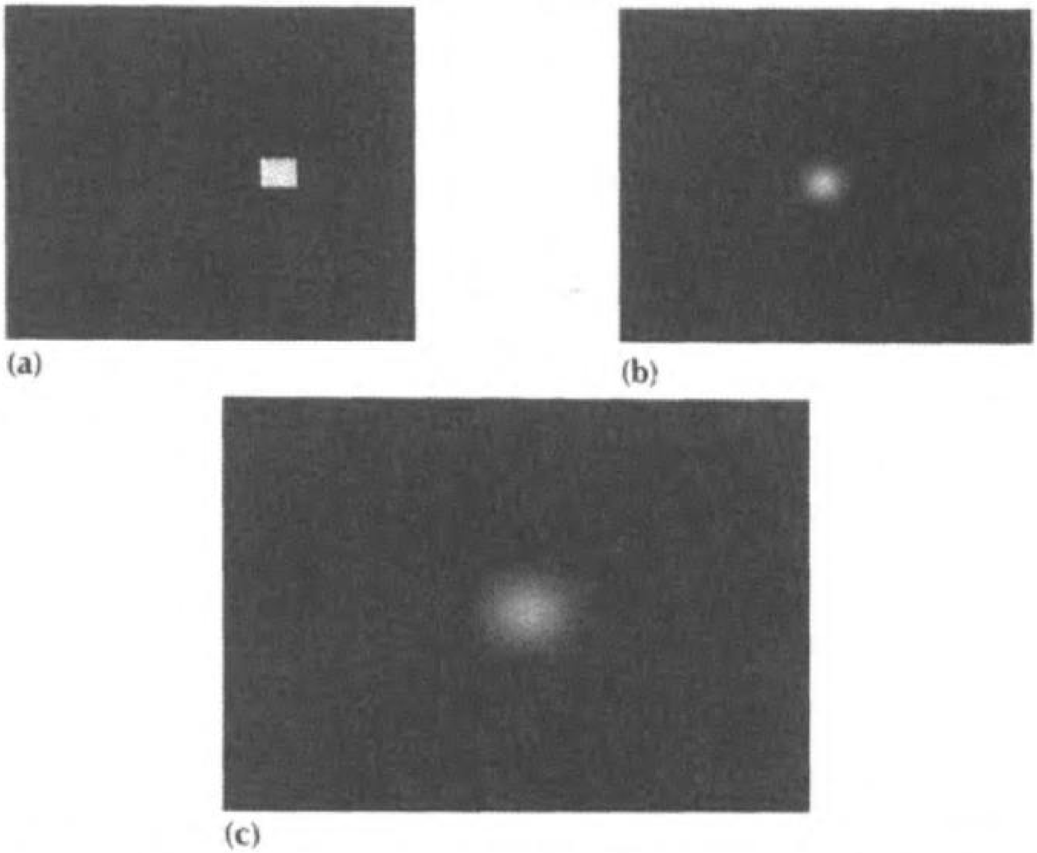


FIGURE 4.11 Computer printout of the Fourier spectrum of the same object as shown in Figure 14.10, but placed at a different position. (a) object; (b) its Fourier spectrum; and (c) center portion of the spectrum enlarged.

14.2.6 Correlation and Convolution

In the processing of images by the Fourier transform technique, correlation and convolution are the important operations. Emphasis on the clarification of the difference between them will be the main subject of this section. Definitions given hereafter are valid only for deterministic functions. Correlation of two continuous functions $f(x)$ and $g(x)$ is defined by

$$f(x) \circ g(x) = \int_{-\infty}^{\infty} f(\alpha)g(x+\alpha)\,d\alpha \tag{14.45}$$

where α is a dummy variable of integration. The correlation is called *auto-correlation* if $f(x) = g(x)$, and *cross-correlation* if $f(x) \neq g(x)$. Convolution, by definition is

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)\,d\alpha \tag{14.46}$$

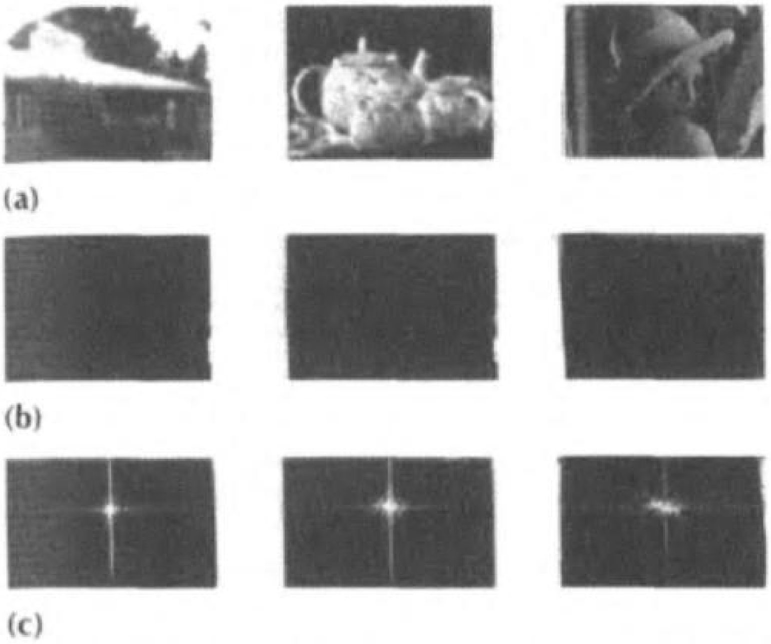


FIGURE 14.12 Original Fourier spectra and those after the origin is translated to $(N/2, N/2)$. (a) Original image; (b) Fourier spectrum of the image as shown in (a); (c) same Fourier spectrum after the origin is translated to the point $(N/2, N/2)$.

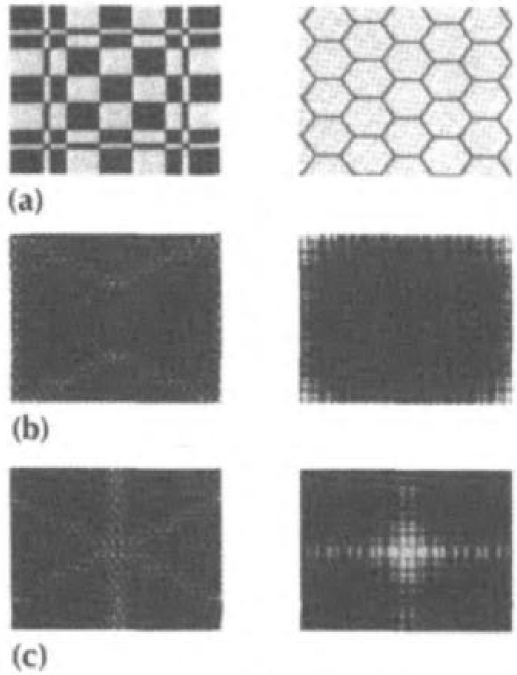


FIGURE 14.13 Original Fourier spectra and those after the origin is translated to $(N/2, N/2)$. (a) Original image; (b) Fourier spectrum of the image as shown in (a); (c) same Fourier spectrum after the origin is translated to the point $(N/2, N/2)$.

The forms for correlation and convolution are similar, the only difference between them being the following. In convolution, $g(\alpha)$ is first folded about the vertical axis and then displaced by x to obtain a function $g(x - \alpha)$. This function is then multiplied by $f(\alpha)$ and integrated from $-\infty$ to ∞ for each value for displacement x to obtain the convolution. (See Figure 14.14, which is self-explanatory.) In correlation, $g(\alpha)$ is not folded about the vertical axis and is directly displaced by x to obtain $g(x + \alpha)$. Figure 14.14e indicates the integral $\int f(\alpha)g(x + \alpha)\,d\alpha$ on the shaded areas shown in part (c), which is the correlation of $f(x)$ and $g(x)$; part (f) indicates the integral $\int f(\alpha)g(x - \alpha)\,d\alpha$ over the shaded area in part (d), which is the convolution of $f(x)$ and $g(x)$.

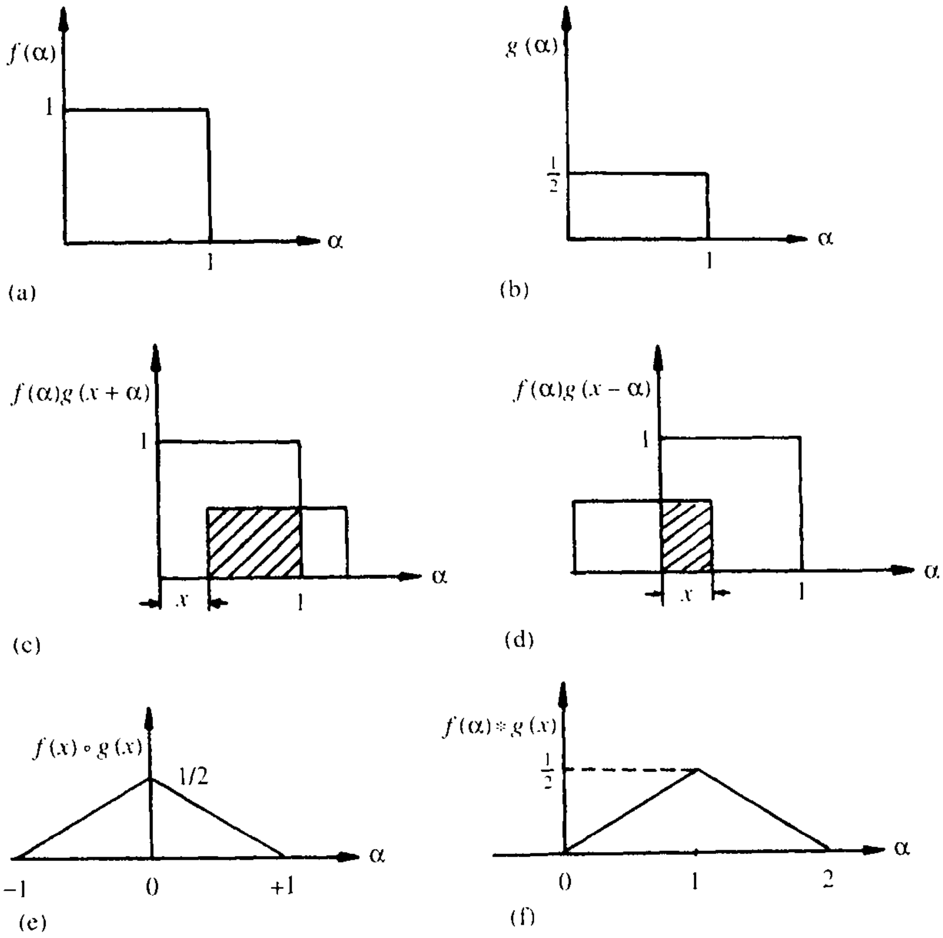


FIGURE 14.14 Graphical illustration of the difference between correlation and convolution.

By the same definition, the correlation for a two-dimensional case can then be expressed as

$$f(x, y) \circ g(x, y) = \int_{-\infty}^{\infty}\int f(\alpha, \beta) g(x + \alpha, y + \beta)\, d\alpha\, d\beta \tag{14.47}$$

After discretization, the correlation between $f(x, y)$ and $g(x, y)$ becomes

$$f(x, y) \circ g(x, y) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) g(x + m, y + n)$$

$$\text{for } x = 0, 1, \ldots, N - 1; y = 0, 1, \ldots, N - 1 \tag{14.48}$$

If the Fourier transform of $f(x, y)$ is $F(u, v)$ and that of $g(x, y)$ is $G(u, v)$, the Fourer transform of the correlation of two functions $f(x, y)$ and $g(x, y)$ is the product of their Fourier transforms with one of them conjugated. Thus

$$f(x, y) \circ g(x, y) \Leftrightarrow F(u, v) G^*(u, v) \tag{14.49}$$

which indicates that the inverse transform of $F(u, v)G^*(u, v)$ gives the correlation of the two functions in the (x, y) domain. An analogous result is that the Fourier transform of the product of two functions $f(x, y)$ and $g(x, y)$ with one of them conjugated is the correlation of their Fourier transforms, and is formally stated as

$$f(x, y) g^*(x, y) \Leftrightarrow F(u, v) \circ G(u, v) \tag{14.50}$$

where * represents the complex conjugate. These two results together constitute the correlation theorem. Similarly, we can derive the convolution theorem as follows:

$$f(x, y) * g(x, y) \Leftrightarrow F(u, v) G(u, v) \tag{14.51}$$

and

$$f(x, y) g(x, y) \Leftrightarrow F(u, v) * G(u, v) \tag{14.52}$$

This states that the Fourier transform of the convolution of two functions is equal to the product of the Fourier transforms of the two functions; and conversely, the Fourier transform of a product of two functions is equal to the convolution of the Fourier transforms of the two functions. These two relations constitute the convolution theorem. This theorem is very useful in that complicated integration in the spatial domain can be completed by comparatively simpler multiplication in the Fourier domain.

One of the principal applications of correlation in image processing is in the area of template matching. The correlations of the unknown with those images of known origin are computed and the largest correlation indicates the closest match. This is sometimes called the method of maximum likelihood.

14.3 SAMPLING

In digital image processing by Fourier transformation, an image function is first sampled into uniformly spaced discrete values in the spatial domain and is then Fourier transformed. After the processing is completed in the Fourier domain, the results are converted back to the spatial image by inverse transformation. What interests us most is the relations between the sampling conditions and the recovered image from the set of sampled values. Let us start with one-dimensional case.

14.3.1 One-Dimensional Functions

If we have a function $f(x)$ which is the envelope of a string of impulses on Figure 14.15a, we have a corresponding transform $F(u)$, as shown in Figure 14.15b. The string of impulses shown in part (a) is actually a sampled version of $f(x)$ with train of impulses Δx apart or

$$f(x) \sum_{k=-\infty}^{\infty} \delta(x - k\,\Delta x) = \sum_{k=-\infty}^{\infty} f(k\,\Delta x)\delta(x - k\,\Delta x) \tag{14.53}$$

$F(u)$ convoluted at interval $1/\Delta x$ is shown in part (b). This is because the Fourier transform of a string of impulses will be another string of impulses a distance $1/\Delta x$ apart, where Δx is the distance between impulses of the original string.

Similarly, if we sample $F(u)$ with an impulse train $S(u)$ Δu units apart between impulses, we will have $f(x) \sum \delta(x - k\ \Delta x)$ convoluted at interval $1/\Delta u$ and periodic with period $1/\Delta u$ in the spatial domain. If N samples of $f(x)$ and $F(u)$ are taken, and the spacings of Δu are selected such that the interval $1/\Delta u$ just covers N samples in the x domain and interval $1/\Delta x$ just covers N samples in the frequency domain, then

$$\frac{1}{\Delta u} = N\,\Delta x \tag{14.54}$$

or

$$\Delta u = \frac{1}{N}\frac{1}{\Delta x} \tag{14.55}$$

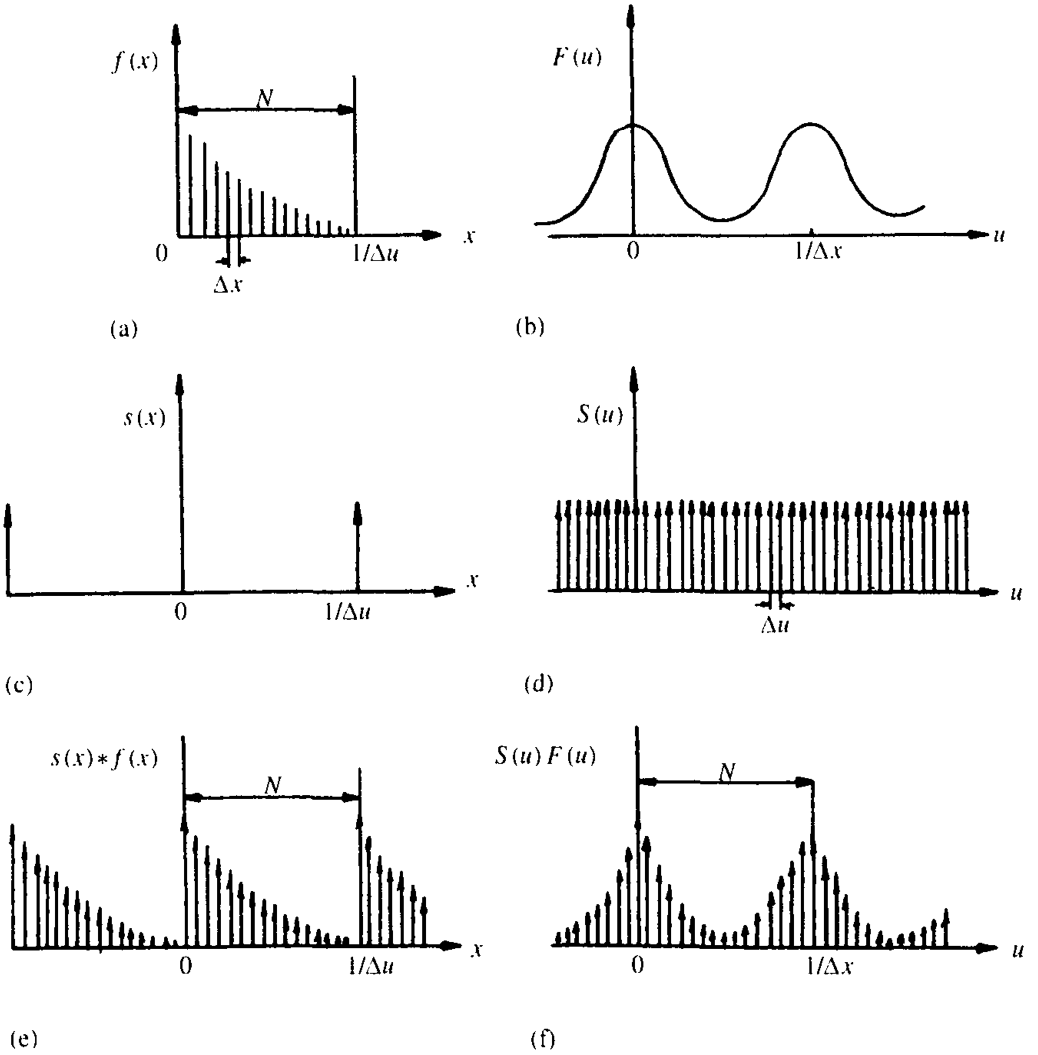


FIGURE 14.15 Sampling on a one-dimensional function.

14.3.2 Two-Dimensional Functions

For the two-dimensional case, the image function is $f(x, y)$. The Fourier transform pair is

$$F(u, v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y)e^{-j2\pi(ux/M+vy/N)}$$

$$u = 0, 1, \ldots, M-1; v = 0, 1, \ldots, N-1 \qquad (14.56)$$

and

$$f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)e^{j2\pi(ux/M+vy/N)}$$
$$x=0,1,\ldots,M-1;y=0,1,\ldots,N-1 \tag{14.57}$$

Let Δx and Δy be the separations of strings in the spatial domains, and Δu and Δv be those in the frequency domains;

$$\begin{aligned}\Delta u&=\frac{1}{M\,\Delta x}\\ \Delta v&=\frac{1}{N\,\Delta y}\end{aligned} \tag{14.58}$$

can be worked out for this two-dimensional DFT by the analogous analysis developed for this one-dimensional case. With these relationships, the image function and its two-dimensional Fourier transform will be periodic with $M \times N$ uniformly spaced values. For $N \times N$ square array samples, that is, $M = N$, we have

$$\begin{aligned}\Delta u&=\frac{1}{N\,\Delta x}\\ \Delta v&=\frac{1}{N\,\Delta y}\end{aligned} \tag{14.59}$$

These relationships between the sample separations guarantee that the two-dimensional period defined by $1/\Delta u \times 1/\Delta v$ just be covered by $N \times N$ uniformly spaced samples in the spatial domain, and the period defined by $1/\Delta x \times 1/\Delta y$ will be covered by $N \times N$ samples in the frequency domain. It is noted that the constant multiplicative terms may be grouped arbitrarily. If they are regrouped this way such that $NF(u,v) \Rightarrow F(u,v)$, the Fourier transform pair will have the following form:

$$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)e^{-j2\pi(ux+vy)/N}$$
$$u,v=0,1,\ldots,N-1 \tag{14.60}$$

$$f(x,y)=\frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}F(u,v)e^{j2\pi(ux+vy)/N}$$
$$x,y=0,1,\ldots,N-1 \tag{14.61}$$

The notions of impulse sheets and two-dimensional impulses suggested by Lendaris et al. (1970) will be introduced for the discussion of two-dimensional functions. An impulse sheet is defined such that it has an infinitive length in one direction and its cross section has the usual delta-function properties. The cross section of the impulse sheet is presumed to remain the same along the sheet's entire length. The intersection of two impulse sheets results in a two-dimensional impulse located at their intersection.

The impulse sheets and two-dimensional impulses have the following properties:

1. The two-dimensional transform of an impulse sheet is an impulse sheet centered at the origin and in the direction orthogonal to the direction of the original impulse sheet.
2. The two-dimensional Fourier transform of an infinitive array of uniformly spaced parallel impulse sheets is an infinite string of impulses along the direction orthogonal to the impulse sheet direction, with a spacing inversely proportional to the impulse sheet separation, and with one of the impulses located at the origin.
3. Conversely, the two-dimensional Fourier transform of a string of uniformly spaced impulses is an array of parallel impulse sheets whose direction is orthogonal to the impulse string, whose separation is inversely proportional to the impulse separation, and one of whose impulse sheets goes through the origin.
4. The two-dimensional Fourier transform of an infinite lattice-like array of impulses is an infinite lattice-like array of impulses whose dimensions are inversely related to those of the original lattice, with an impulse at the origin.
5. When convolving a function with an array of impulses, the function is simply replicated at the locations of each of the impulses.
6. The convolution theorem as derived for one-dimensional functions also holds for two-dimensional functions.

Relatively Large Aperture

If the image consists of an array of uniformly spaced parallel straight lines with a scan circular aperture, the Fourier transform of the combination will be the convolution of their respective Fourier transforms. Thus

$$\begin{aligned}&\mathscr{F}[(\text{a straight line} * \text{a string of impulses})(\text{circular aperture function})]\\&= \mathscr{F}(.) * \mathscr{F}(.)\end{aligned}$$

where the first $\mathscr{F}(.)$ is the Fourier transform of the first set of parentheses, while the second $\mathscr{F}(.)$ is the Fourier transform of the second set of parentheses. The result will be a string of impulses with a separation equal to the reciprocal of the spacing of the lines and in a direction perpendicular to these parallel lines. If the aperture is relatively large with respect to the spacing of the parallel lines, the separation of the string of impulses will be large, as shown in Figure 14.16b. The

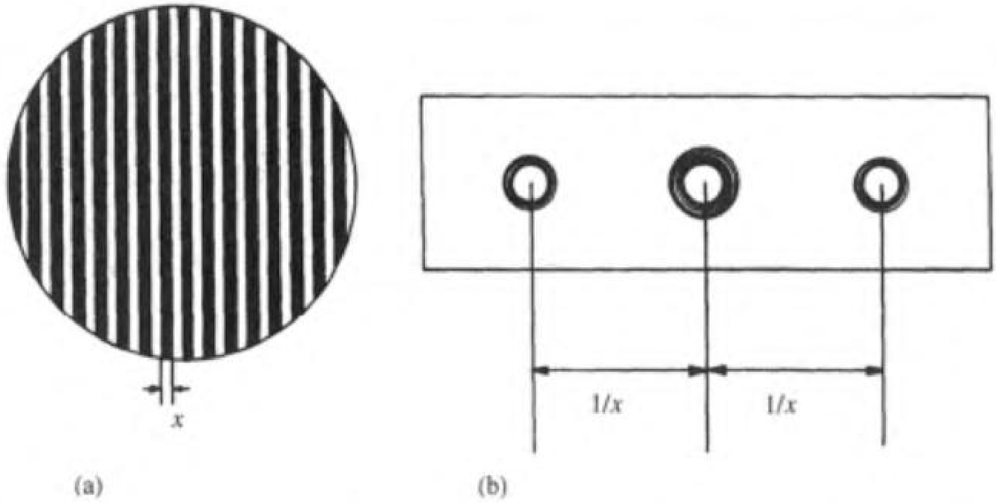


FIGURE 14.16 Fourier transform of a uniformly spaced parallel straight lines with a scan circular aperture. (a) Uniformly spaced parallel straight lines; (b) Fourier transform of (a).

Airy disk is the Fourier transform of the circular aperture, and is replicated on each impulse. A computer plot is shown in Figure 14.17. Some additional examples are shown in Figures 14.18 to 14.25.

Figure 14.26 shows another example, which consists of an array of rectangles with a relatively smaller rectangular aperture. The Fourier transform of this image will be the Fourier transform of the combination

[(A rectangle * array of impulses)](rectangular aperture function)

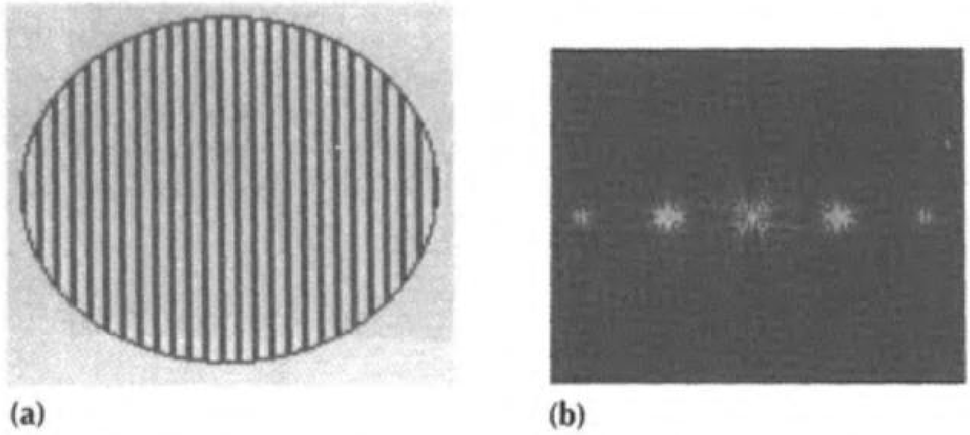


FIGURE 14.17 Computer printout of the Fourer transform of uniformly spaced parallel straight lines with a scan circular aperture. (a) uniformly spaced parallel straight lines; (b) Fourier transform of (a).

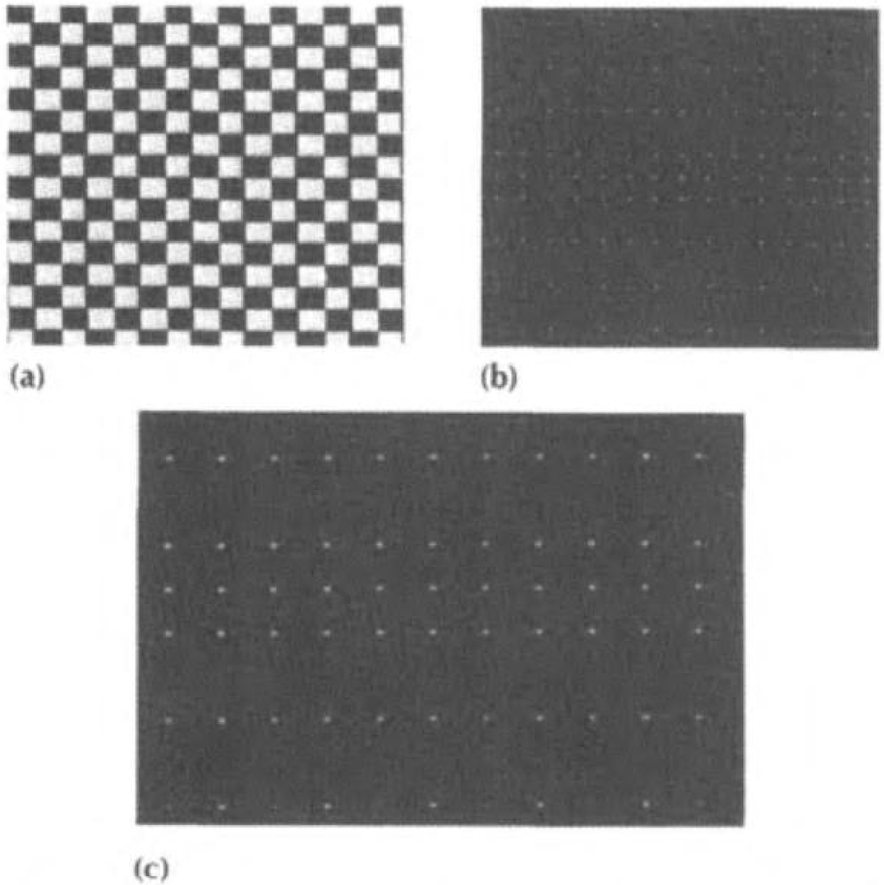


FIGURE 14.18 Arrays of squares with a relatively large square aperture. (a) array of squares; (b) spectrum; (c) center portion of the spectrum enlarged.

The result of Fourier transform of this array of rectangles (i.e., those inside the brackets) will be the product of Fourier transform of the rectangle shown in Figure 14.26b and the Fourier transform of the array of impulses. The Fourier transform of the array of impulses is another array of impulses. Multiplication of the Fourier transform of the rectangle with an array of impulses gives a "sampling" Fourier transform of the rectangle (Figure 14.26c).

By the convolution theorem, the product of the rectangular aperture function and the array of rectangles in the spatial domain will give a convolution of their respective spectra. Therefore, the Fourier transform of the rectangular aperture will be replicated at each sampling point of the Fourier transform in Figure 14.26c.

Relatively Small Aperture

Let us take the example shown in Figure 14.27. This array of six squares can be viewed as the result of convolving one square with an infinite array of impulses,

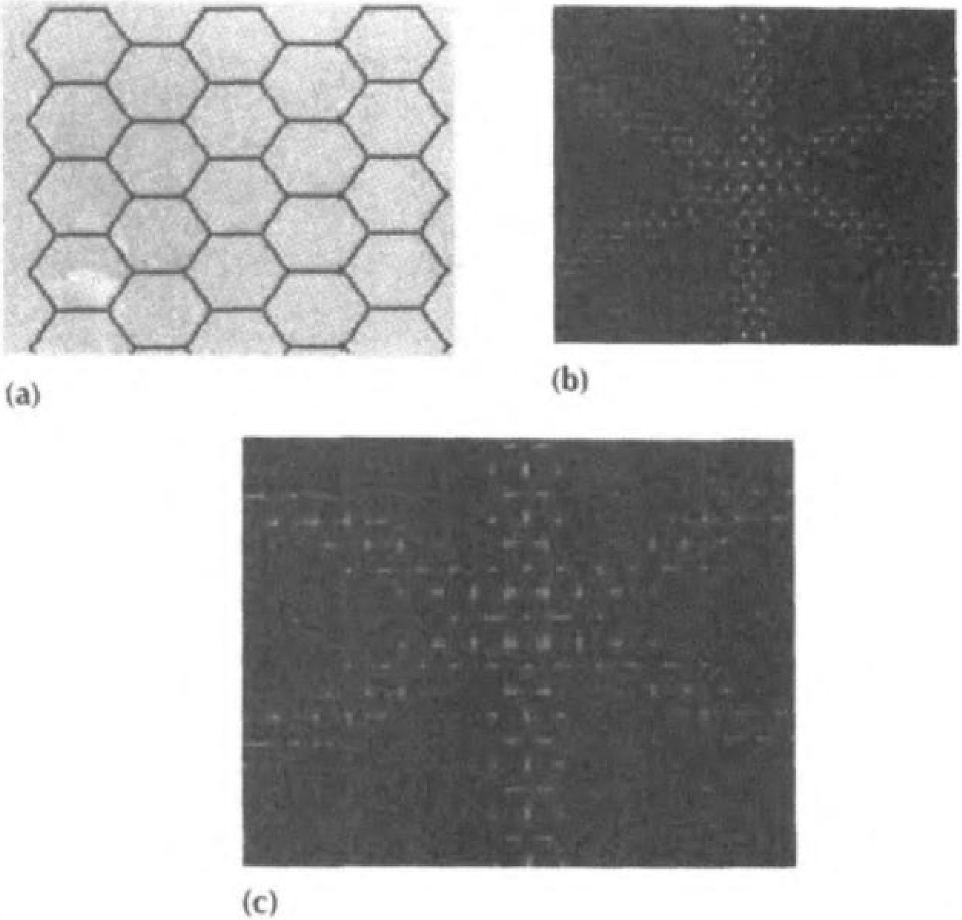


FIGURE 14.19 Arrays of hexagons with a relatively large square aperture. (a) Array of hexagons; (b) its spectrum; (c) center portion of the spectrum enlarged.

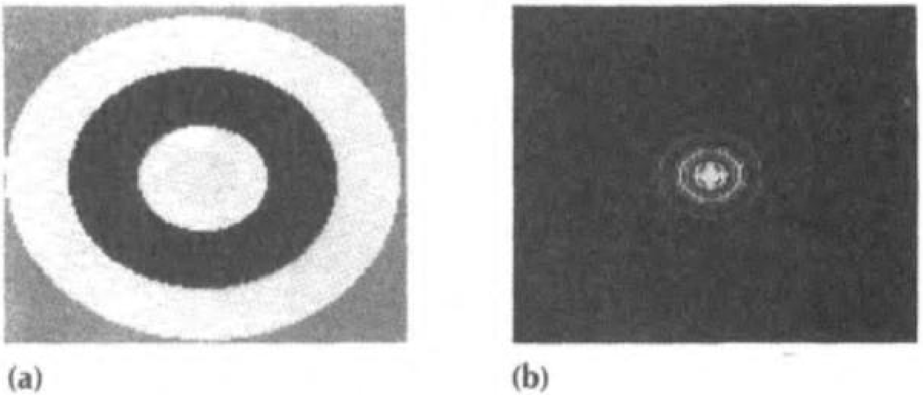


FIGURE 14.20 Fourier spectrum of a pattern with a relatively large square aperture. (a) Simple pattern; (b) its spectrum.

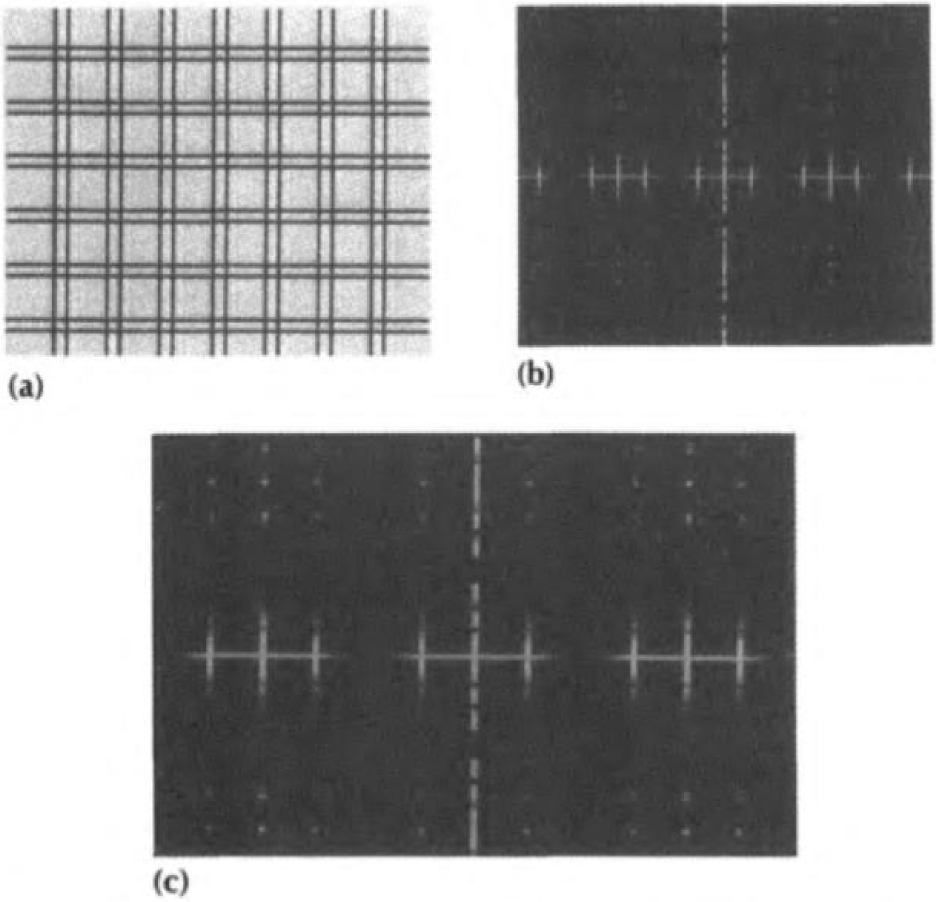


FIGURE 14.21 Fourier spectrum of a pattern with a relatively large square aperture. (a) Simple regular pattern; (b) its spectrum; (c) center portion of the spectrum enlarged.

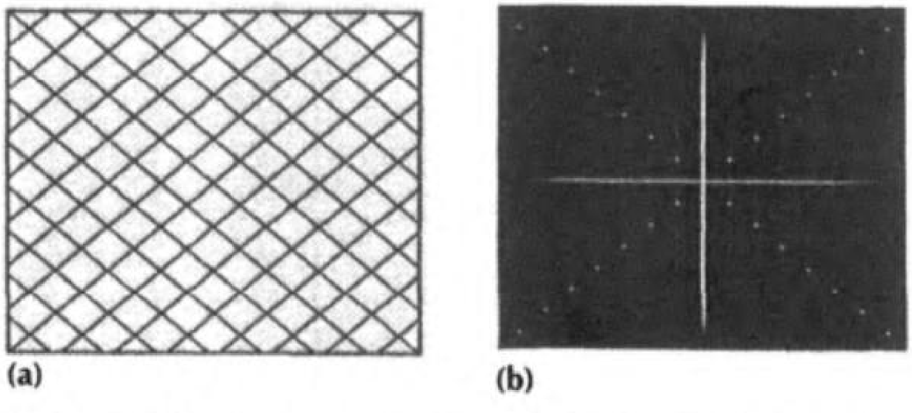


FIGURE 14.22 Fourier spectrum of a pattern with a relatively large square aperture. (a) Simple regular pattern; (b) its spectrum.

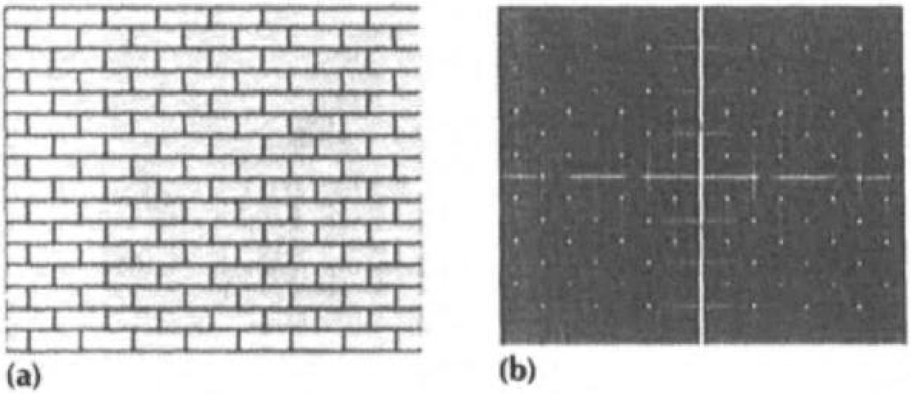


FIGURE 14.23 Fourier spectrum of a pattern with a relatively large square aperture. (a) Simple regular pattern; (b) its spectrum.

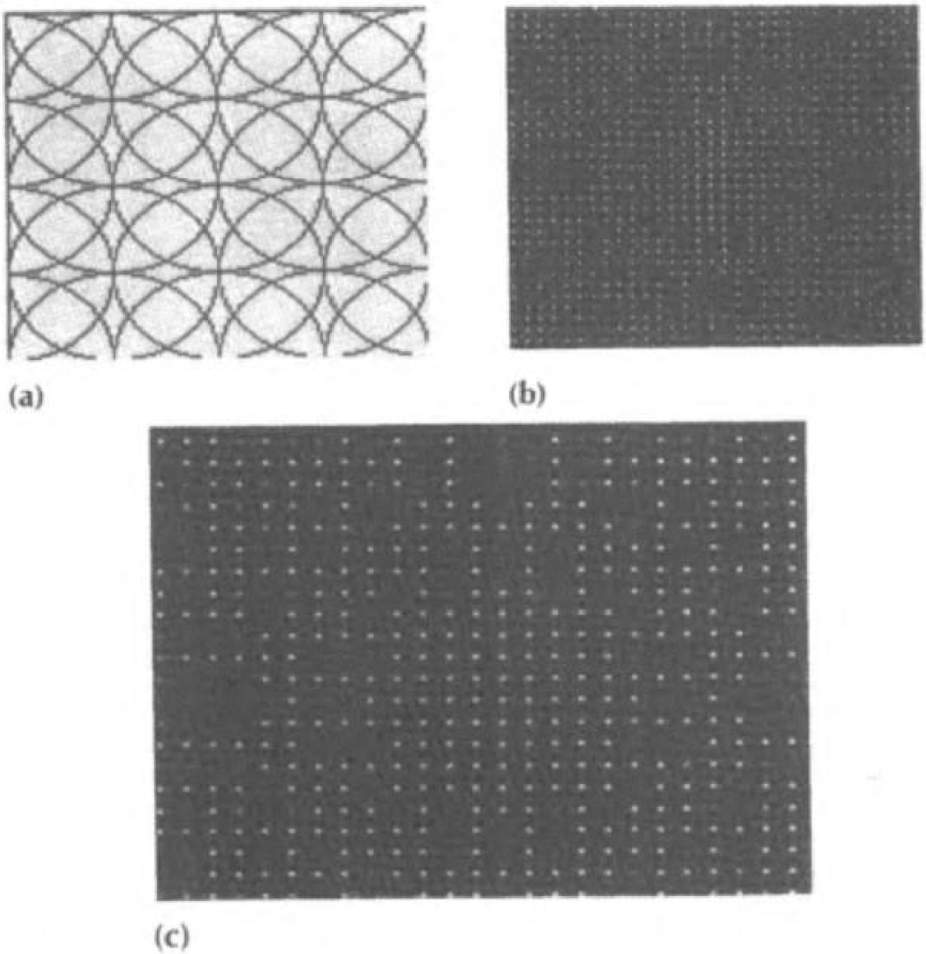


FIGURE 14.24 Fourier spectrum of a pattern with a relatively large square aperture. (a) Pattern, (b) its spectrum; (c) center portion of the spectrum enlarged.

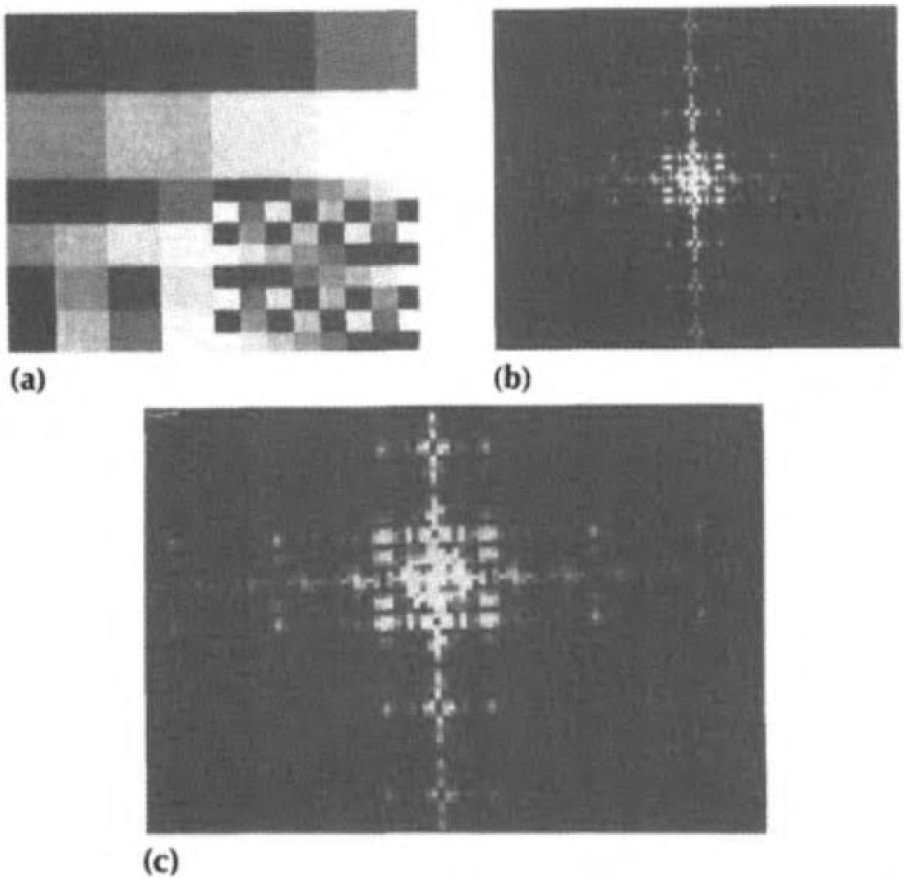


FIGURE 14.25 Fourier spectrum of a pattern with a relatively large square aperture. (a) Complicated pattern; (b) its spectrum; (c) center portion of the spectrum enlarged.

with the result multiplied by an aperture to allow only the six squares to appear; that is,

[(A square * infinite array of impulses)](rectangular aperture)

According to the mathematical operation above, its corresponding Fourier transform will then be

[(FT of the square)(FT of the impulse array)] * (FT of rectangular aperture)

Since the impulse spacing in the spatial domain is relatively large with respect to the aperture, the Fourier transform of this array of impulses (which turns out to be another array of impulses) will have relatively narrow spacing, thus giving a sampled version of the Fourier transform of the square. Convolution of this multiplication with the Fourier transform of the rectangular aperture will have the Fourier transform of the aperture replicated at each sample point.

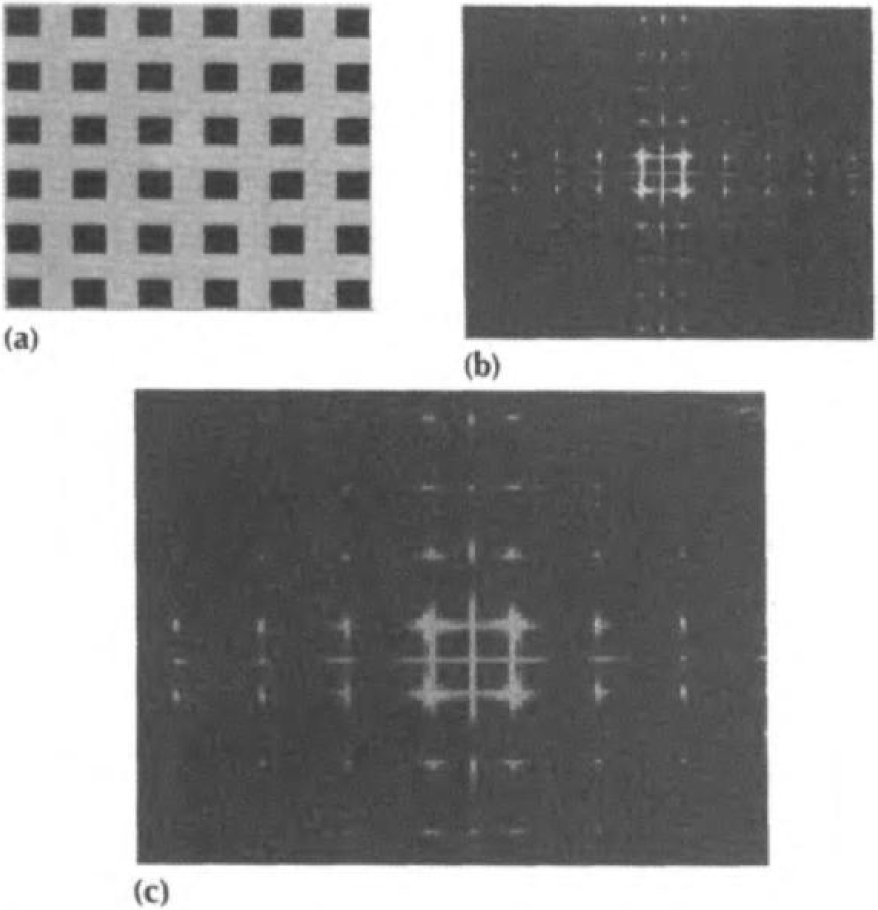


FIGURE 14.26 Arrays of squares with a relatively small aperture. (a) Array of squares; (b) its spectrum; (c) center portion of the spectrum enlarged.

In extracting primitives of Chinese characters by the Fourier transform technique, a similar problem appears. Figure 14.28a shows a primitive consisting of two parallel bars in a vertical direction and three bars in a horizontal direction. Figure 14.28b shows its spectrum, and Figure 14.28c shows the central portion of the Fourier transform of part (a). Another primitive, different from that shown in Figure 14.28a by having one more bar in the vertical direction, is shown in Figure 14.29a. Because of the differences in the number of bars in the vertical direction, a difference in the Fourier transform can also be noted.

Figure 14.30a shows a parking lot. For the same reason as before, this can be viewed as a small rectangle convoluted with an infinite string of uniformly spaced impulses multiplied by the scan aperture so as to cause only a small number of rectangles to appear. Figure 14.30b and c show, respectively, its spectrum and the central portion of the spectrum. More spectra for their corresponding images are shown in Figures 14.31 to 14.34.

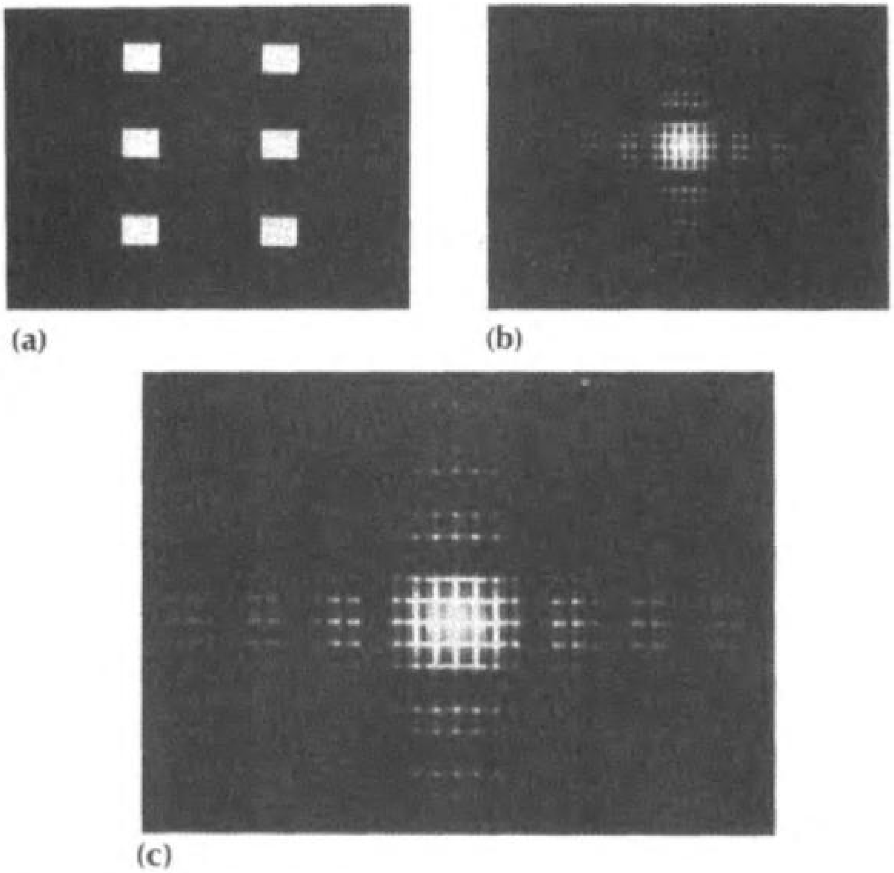


FIGURE 14.27 Arrays of squares with a relatively small aperture. (a) Array of squares; (b) its spectrum; (c) center portion of the spectrum enlarged.

14.3.3 Applications

Sampling is a good tool to use to reduce the amount of information to be processed. The Fourier transform is particularly well suited to sampling because of the following features:

1. Most of the information from the original imagery will be on the central portion of the spectrum. From the translation property of the Fourier transform,

$$f(x-x_0, y-y_0) \Leftrightarrow F(u, v)e^{-j2\pi(ux_0+vy_0)/N} \tag{14.62}$$

   It is interesting to note that a shift in $f(x, y)$ does not affect the magnitude of its transform. A linear object in the original imagery gives rise to a spectrum along a line centered on the central axis. Similarly, a circular object gives rise to a spectrum as concentric annular rings centered on the central axes. A latticelike object gives

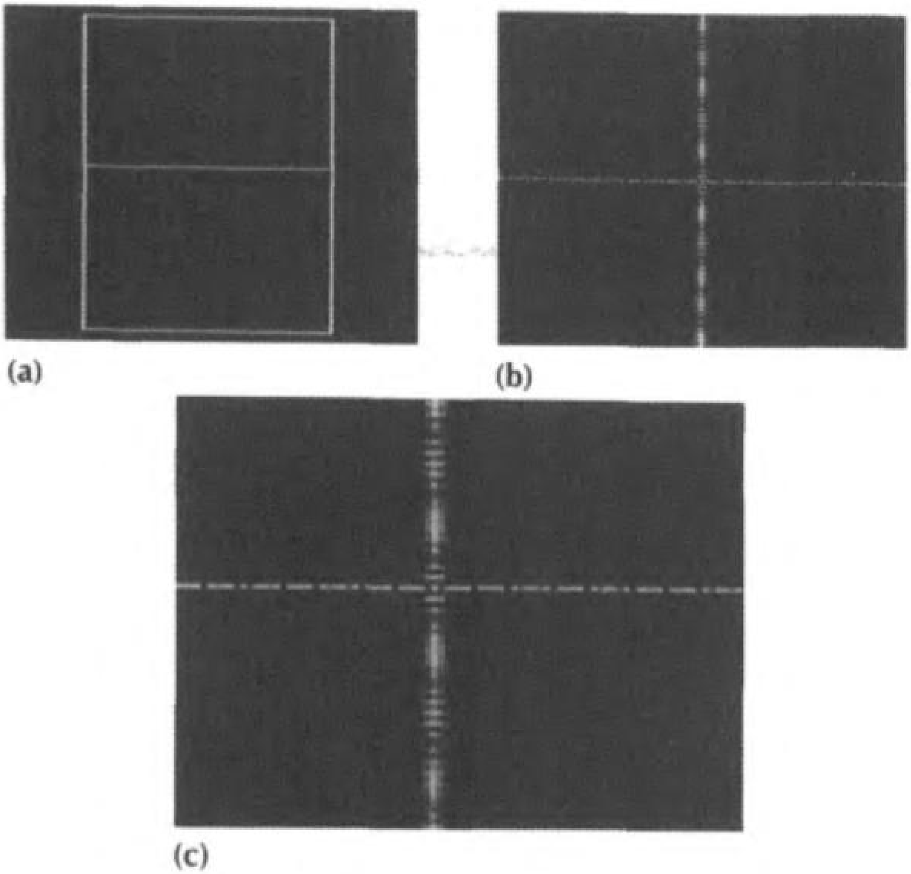


FIGURE 14.28 Fourier spectrum of an ideograph with a relatively small aperture. (a) A Chinese character; (b) its spectrum; (c) center portion of the spectrum enlarged.

rise to a latticelike spectrum in the diffraction patterns. However, no reference point like this exists in the original spatial image. Scanning and processing of the whole area are then necessary to obtain the object information in the image.

2. From the linearity property of Fourier transform as described by

$$\mathscr{F}[af_1(x,y)+f_2(x,y)] = aF_1(u,v)+f_2(u,v) \tag{14.63}$$

where $F_1(u,v) = \mathscr{F}[f_1(x,y)]$ and $F_2(u,v) = \mathscr{F}[f_2(x,y)]$, the Fourier spectrum can be well interpreted by superposition of component spectra from their corresponding separable spatial image functions.

Several sampling devices were suggested by Lendaris et al. (1970) to measure the amount of light energy falling within specified areas of the Fourier spectrum. With an annular-ring sampling device as shown in Figure 14.35a, the total light energy of the Fourier spectrum measured along a circle centered on the optical axis corresponds to one frequency in all directions. With a set of annular-

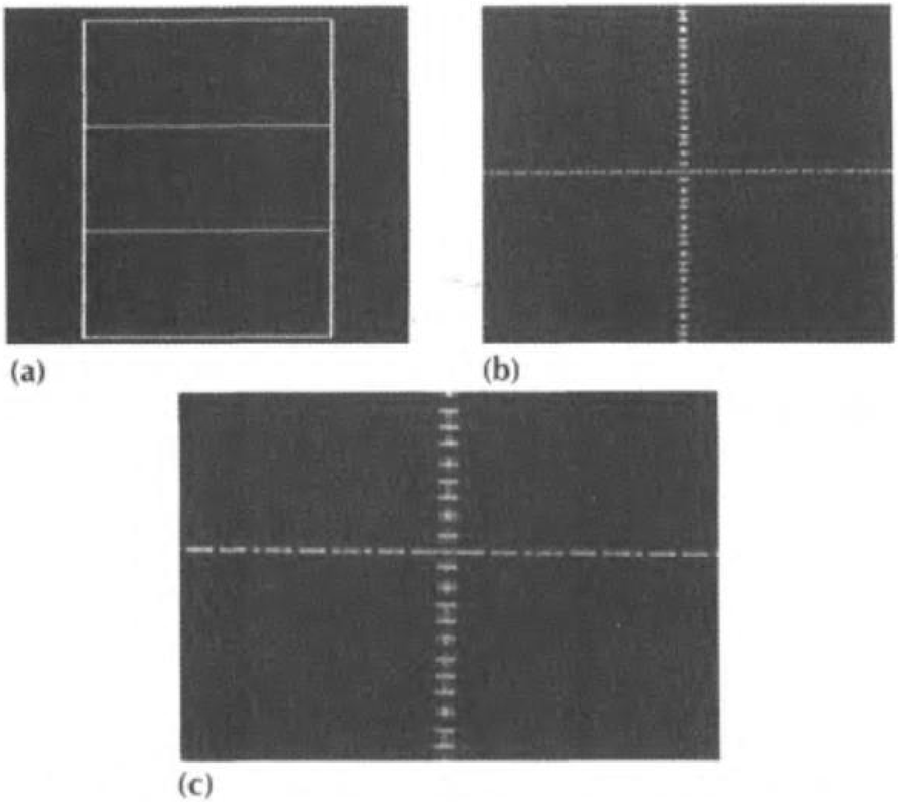


FIGURE 14.29 Fourier spectrum of another ideograph with a relatively small aperture. (a) Another Chinese character; (b) its spectrum; (e) center portion of the spectrum enlarged.

ring sampling windows, a spatial frequency profile of the contents of the scan area can be obtained simultaneously. The device can be used to detect the regularity.

Figure 14.35b shows another sampling device (a wedge-shaped sampling window) in which the light energy of the spectrum along a radial line (which corresponds to a single direction in the spectrum) can be measured, and gives a direction profile of the contents of the scan area simultaneously. This device can be used to find the principal directions. These sample signatures, obtained either from the annular ring or from a wedge-shaped sampling device (or from both) are useful for pattern recognition.

14.3.4 Image Information Content in the Fourier Spectrum: A Practical Example

As discussed in Section 14.3.3, every component object in a scan area has its own spectrum. All these spectra will be superimposed and centered on the central

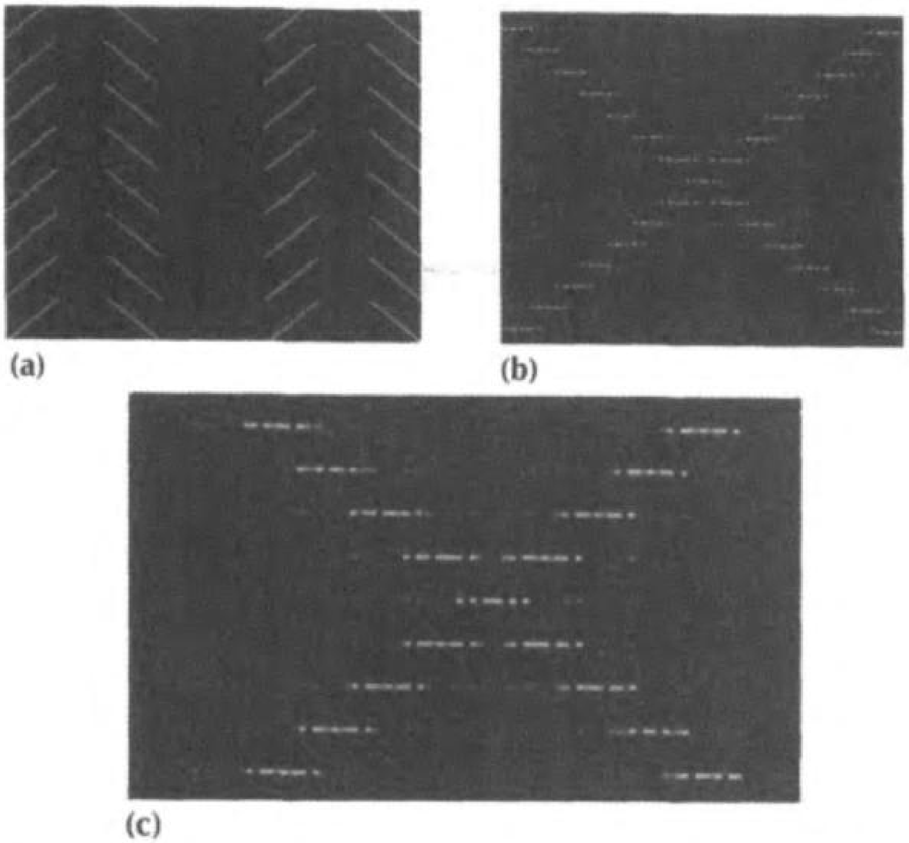


FIGURE 14.30 Fourier spectrum with a relatively small aperture. (a) A parking lot; (b) its spectrum; (c) center portion of the spectrum enlarged.

axes. It is interesting to know how many percentages of data taken from the central spectrum portion will be enough to preserve the image quality. No general answer can be given, since this is highly problem dependent. For some cases the picture quality is the most important requirement, and therefore a larger percentage of the spectrum data should be used in the processing to preserve both the low-frequency and all the high-frequency components of the image. But in other cases, the processing speed will be the first criterion, and even some sacrifice of image quality will be tolerable. Numerous examples can be enumerated, one of which is air reconnaissance. All we need is to search the desired objects within the scan area at a very fast speed and then focus our analysis on a smaller area to get more details. The first thing of concern is the speed; that is what is usually required in real-time processing or pseudo-real-time processing.

The complex image shown in Figure 14.36a has been Fourier transformed and different percentages (5%, 10%, and 20%) of its spectrum were taken to restore the images. The processing results are shown in Figure 14.36b to e. Part

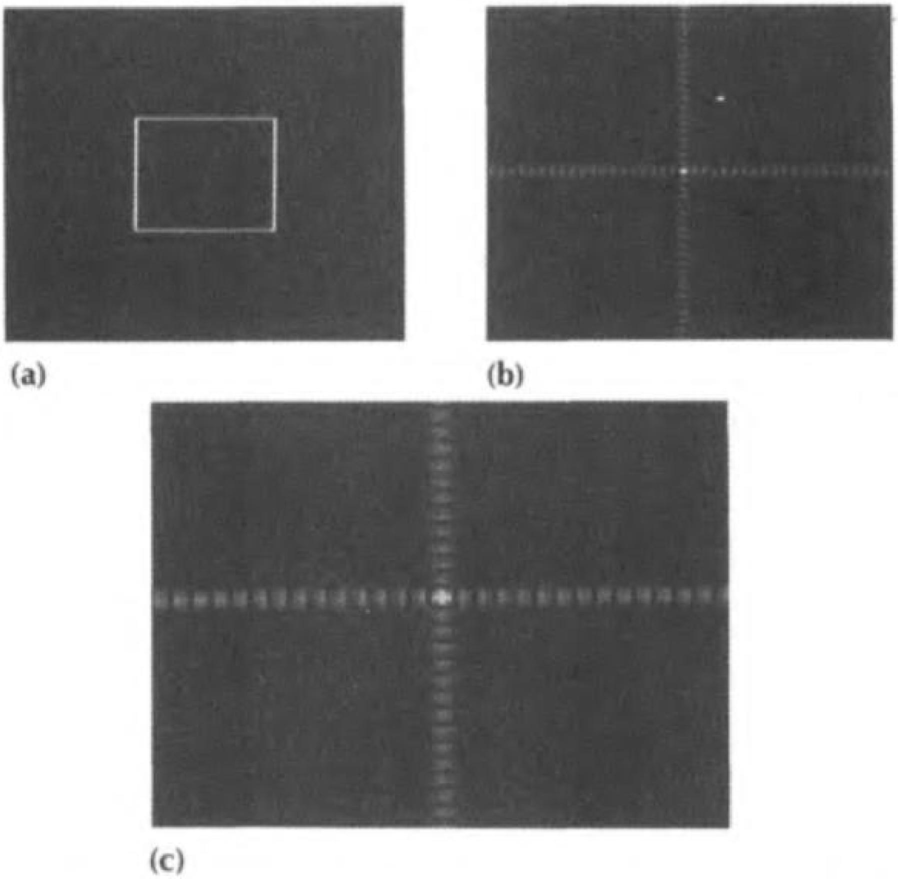


FIGURE 14.31 Fourier spectrum with a relatively small aperture. (a) A simple pattern; (b) its spectrum; (c) center portion of the spectrum enlarged.

(a) is the original image, (b) is its spectrum, and (c), (d), and (e) show, respectively, the restored images when 95%, 90%, and 80% of the spectrum information far away from the center was discarded. It can be seen that lots of boundary information has been lost in the restored image shown in (c) (i.e., when 95% of the spectrum data was discarded in the restoration process). But restored images like those shown in part (e) are sometimes acceptable for applications where high processing speed is of primary concern.

More examples are given below. Figures 14.37 and 14.38 are for two regular patterns, and Figures 14.39 and 14.40 are for two images. Images restored with different percentages of spectrum data far away from center discarded are indicated. It is interesting to see that parts (c) of Figures 14.37 and 14.40 are all obtained by discarding 95% of their spectra, but it looks that Figures 14.37c and 14.38c are much more blurred than Figures 14.39c and 14.40c. As a matter of fact, they should be the same. The only difference is that the blurring effects in

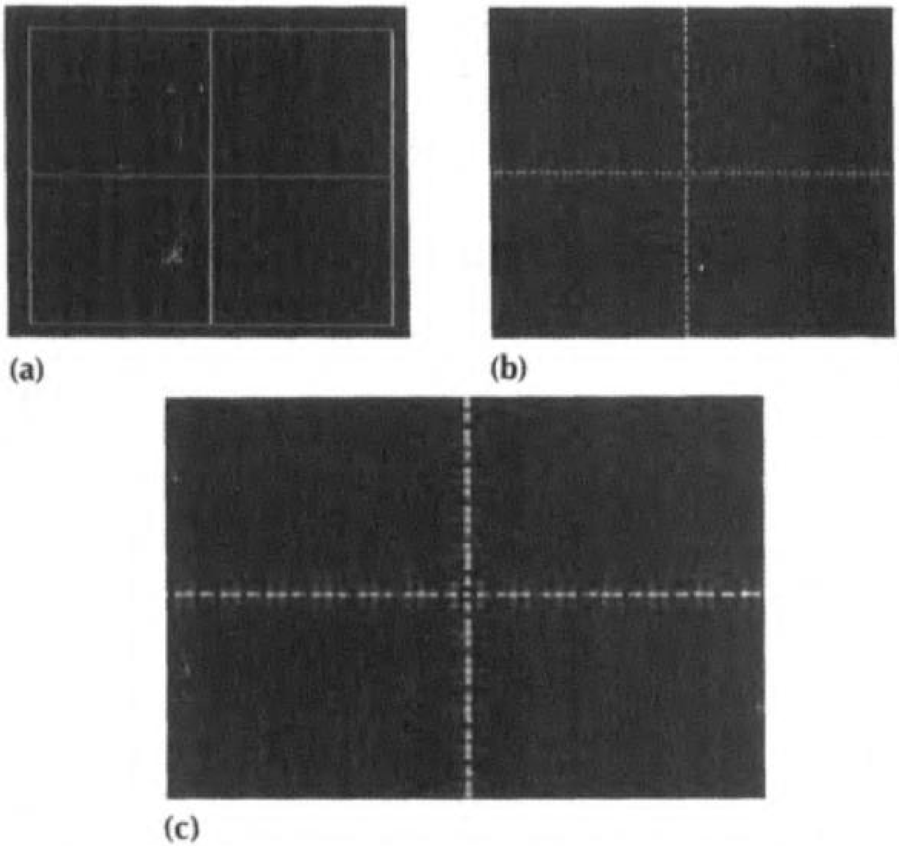


FIGURE 14.32 Founer spectrum of a simple pattern with a relatively small aperture. (a) A simple pattern; (b) its spectrum; (c) center portion of the spectrum enlarged.

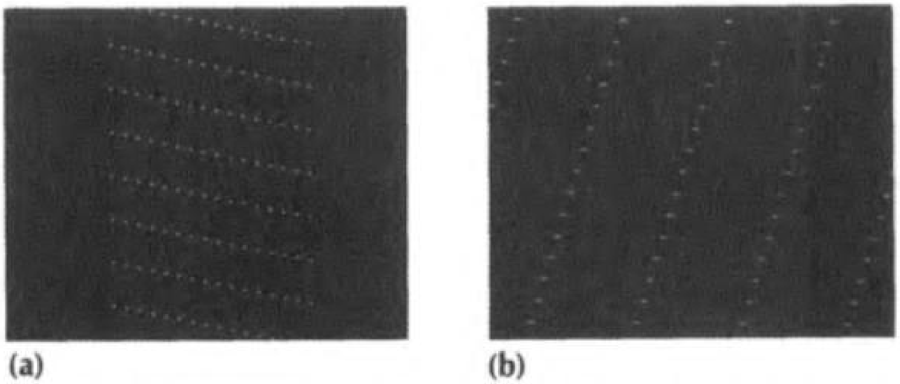


FIGURE 14.33 Founer spectrum of the simple pattern image with a relatively small aperture. (a) A simple pattern of parallel lines; (b) its spectrum.

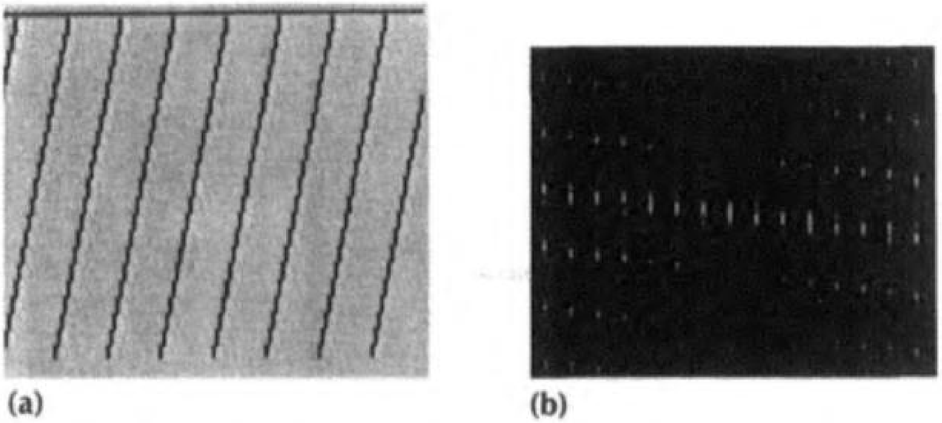


FIGURE 14.34 Fourier spectrum of an image with a relatively small aperture. (a) A simple pattern; (b) its spectrum.

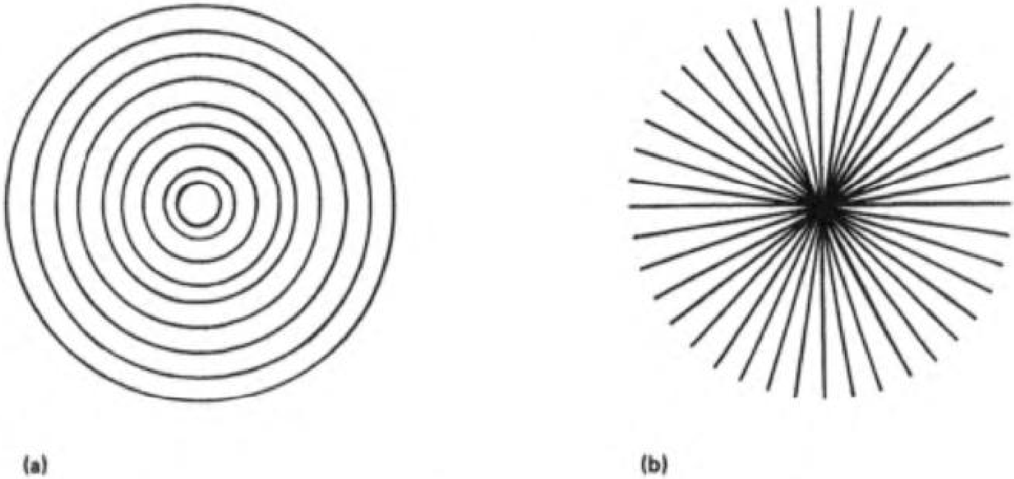
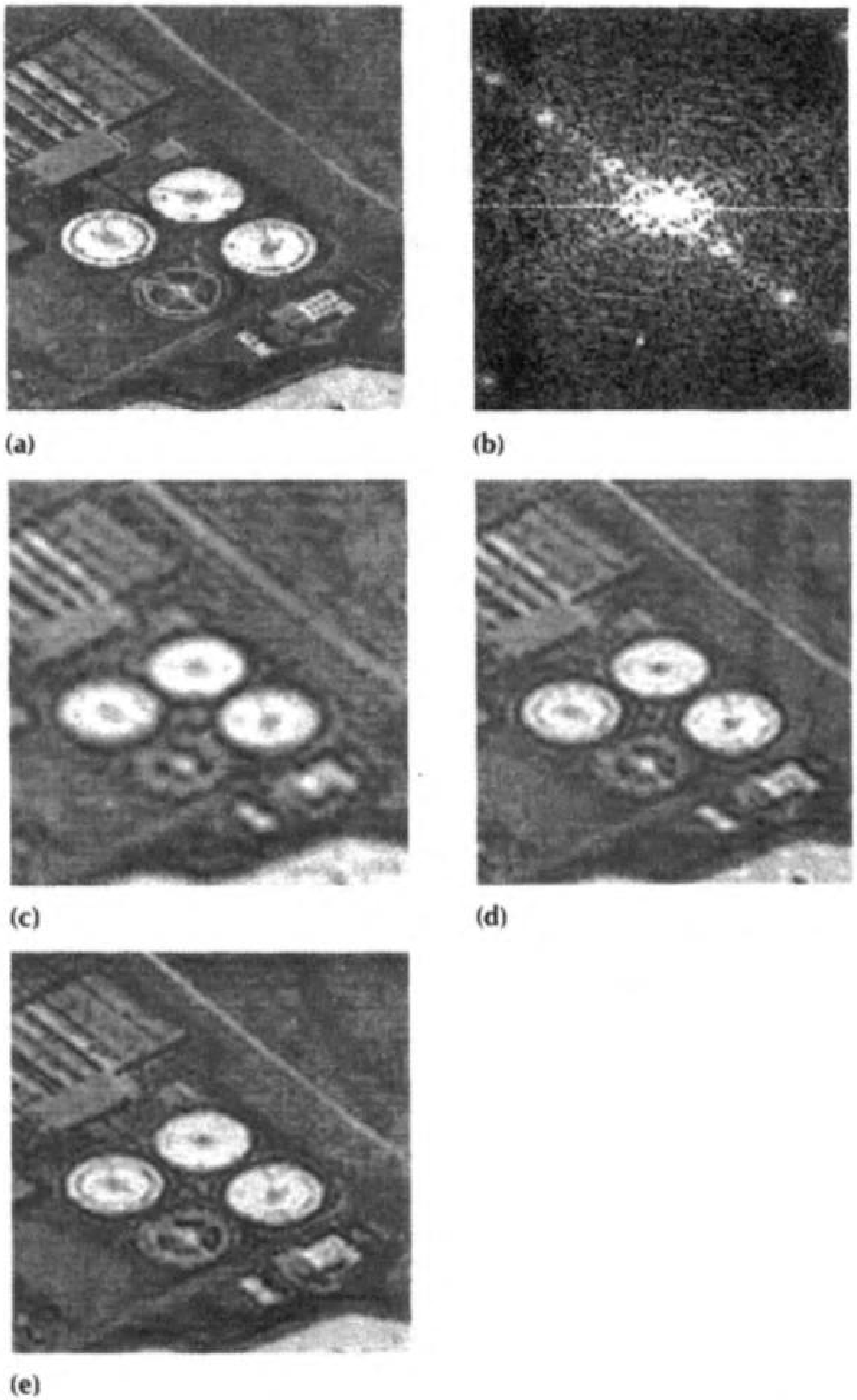


FIGURE 14.35 Sampling devices. (a) Annular-ring sampling device; (b) wedge-shaped sampling device.

the regular patterns are more sensitive to human vision than are those in the images.

14.4 FAST FOURIER TRANSFORM

The Fourier transformation technique is a very effective tool, although a great deal of computation is needed to carry out these transformations. This makes the Fourier transformation technique impractical unless the computation can be simplified.



(a)

(b)

(c)

(d)

(e)

FIGURE 14.36 Information content in the Fourier spectrum. (a) Original image; (b) spectrum; (c) restored image with 95% of spectrum data far away from center discarded; (d) restored image with 90% of spectrum data far away from center discarded; (e) restored image with 80% of spectrum data far away from center discarded.

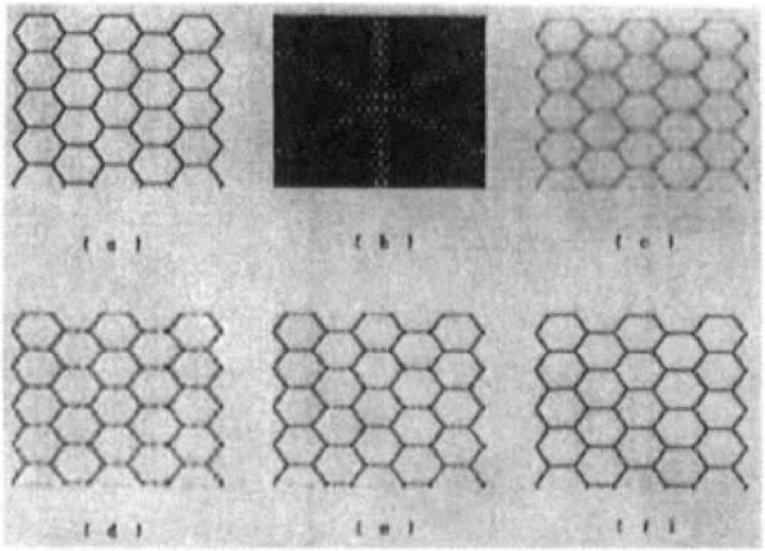


FIGURE 14.37 Information content in the Fourier spectrum. (a) Original image; (b) spectrum; (c) restored image with 95% of spectrum data far way from center discarded; (d) restored image with 90% of spectrum data far away from center discarded; (e) restored image with 80% of spectrum data far away from center discarded; (f) restored image with 50% of spectrum data far away from center discarded.

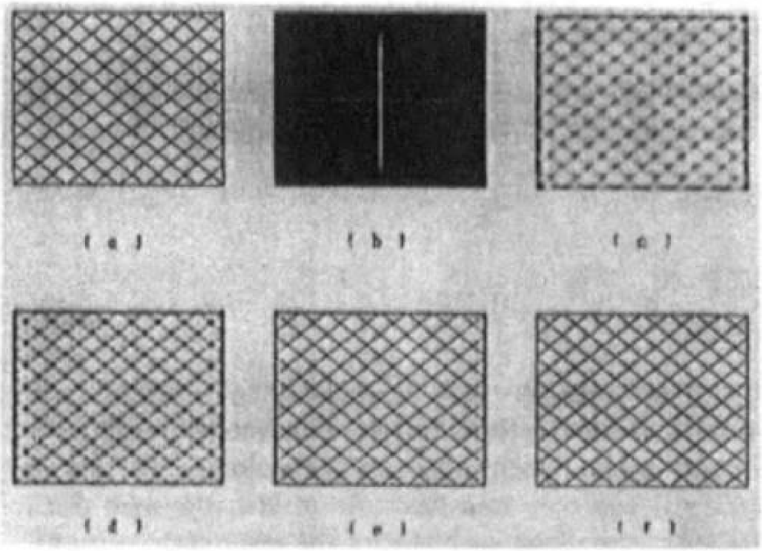


FIGURE 14.38 Information content in the Fourier spectrum. (a) Original image; (b) spectrum; (c) restored image with 95% of spectrum data far away from center discarded; (d) restored image with 90% of spectrum data far away from center discarded; (e) restored image with 80% of spectrum data far away from center discarded; (f) restored image with 50% of spectrum data far away from center discarded.

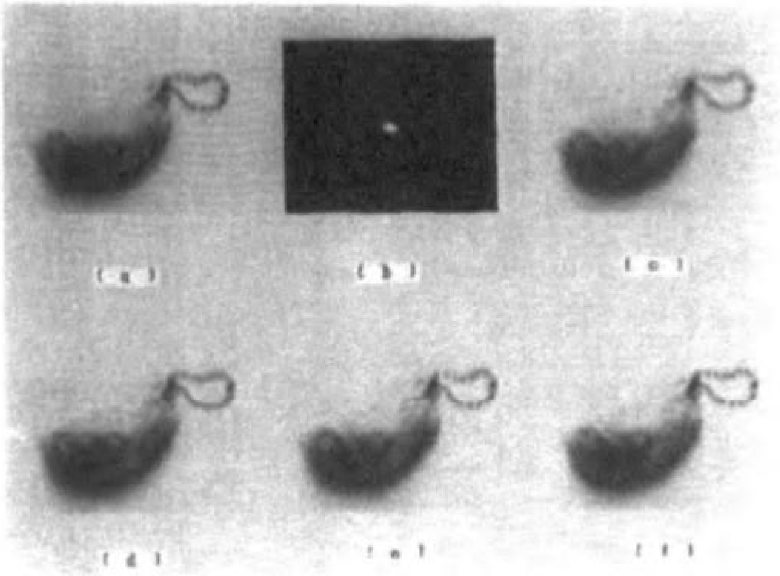


FIGURE 14.39 Information content in the Fourier spectrum. (a) Original image; (b) spectrum; (c) restored image with 95% of spectrum data far away from center discarded; (d) restored image with 90% of spectrum data far away from center discarded; (e) restored image with 80% of spectrum data far away from center discarded; (f) restored image with 50% of spectrum data far away from center discarded.

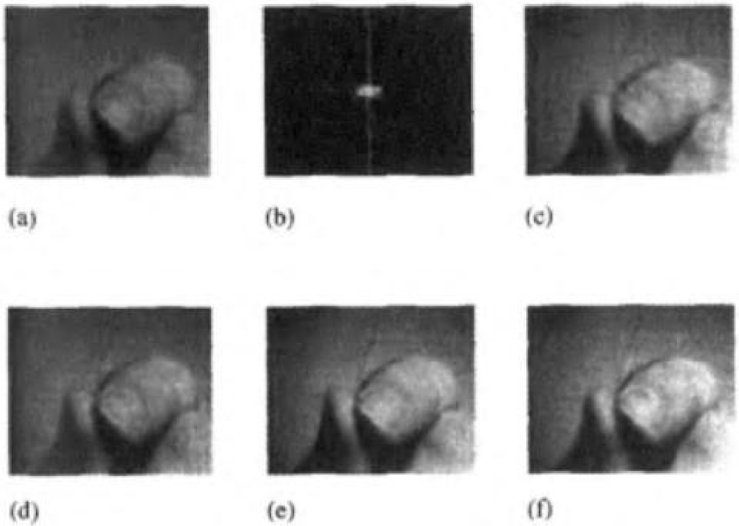


FIGURE 14.40 Information content in the Fourier spectrum. (a) Original image; (b) spectrum; (c) restored image with 95% of spectrum data far away from center discarded; (d) restored image with 90% of spectrum data far away from center discarded; (e) restored image with 80% of spectrum data far away from center discarded; (f) restored image with 50% of spectrum data far away from center discarded.

14.4.1 DFT of a Two-Dimensional Image Computed as a Two-Step One-Dimensional DFT

The separability of the kernel property can he used in the simplification of the transformation process. Rewrite Eqs. (14.32) and (14.33) as follows:

$$F(u, v) = \mathscr{F}_y\{\mathscr{F}_x[f(x, y)]\} \tag{14.64}$$

and

$$F(u, v) = \mathscr{F}_x\{\mathscr{F}_y[f(x, y)]\} \tag{14.65}$$

In other words, the Fourier transformation operation on the image function $f(x, y)$ can be performed in two steps: first, transform the two-dimensional image function column-wise i.e., perform one-dimensional transform along each column of the image function $f(x, y)$, and then transform the results row-wise (i.e., perform one-dimensional transform along each row of the resulting spectrum), as indicated by Eq. (14.64). A different order of transformation can also be taken: transform $f(x, y)$ row-wise first, and then transform the resulting spectrum column-wise, as indicated by Eq. (14.65).

Similarly, the inverse Fourier transform can also be performed in two steps, such as

$$f(x, y) = \mathscr{F}^{-1}_{u,v}[F(u, v)] = \mathscr{F}^{-1}_u\{\mathscr{F}^{-1}_v[F(u, v]\} \tag{14.66}$$

and

$$f(x, y) = \mathscr{F}^{-1}_v\{\mathscr{F}^{-1}_u[F(u, v)]\} \tag{14.67}$$

The complex conjugate properties in the arithmetic operation can also be used to simplify the transformation process. The conjugate of $[f(x, y)]^*$ can be written according to Eq. (14.61) as

$$[f(x, y)]^* = \frac{1}{N}\left[\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u, v)e^{j2\pi(ux+vy)/N}\right]^* \tag{14.68}$$

where $e^{j2\pi(ux+vy)/N}$ is the inverse transformation kernel. Equation (14.68) can further be written as

$$[f(x, y)]^* = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u, v)^* e^{-j2\pi(ux+vy)/N} \tag{14.69}$$

from which it is interesting to know that the inverse transformation kernel in Eq. (14.68) is converted to a forward transformation kernel in Eq. (14.69). That is, it is possible to use the same forward transformation kernel to do the inverse transformation. For a real function, where $[f(x,y)]^* = f(x,y)$, Eq. (8.69) then becomes

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F^*(u,v)e^{-j2\pi(ux+vy)/N} \tag{14.70}$$

Comparison with Eq. (14.60) shows that the same forward transformation algorithm can be used to do the inverse transform, as far as the Fourier transform $F(u,v)$ is conjugated to $F^*(u,v)$. Arguments similar to those used in Eqs. (14.64) and (14.65) are also valid for Eq. (14.70); thus

$$f(x,y) = \mathscr{F}_{u,v}[F^*(u,v)] = \mathscr{F}_u\{\mathscr{F}_v[F^*(u,v)]\} \tag{14.71}$$

Note that unlike Eq. (14.67), $\mathscr{F}_u$ and $\mathscr{F}_v$ in Eq. (14.71) are forward Fourier transforms. A conclusion can then be drawn that an inverse discrete Fourier transform may be computed as the discrete Fourier transform of the conjugate, and can also be computed as a two-step one-dimensional discrete Fourier transform. One-dimensional discrete Fourier transform will then be the nucleus of the discrete Fourier transform and the inverse discrete Fourier transform.

14.4.2 Method of Successive Doubling

As discussed in previous sections, a two-dimensional Fourier transform can be separated into two computational steps, each of which is a one-dimensional Fourier transform, and an inverse two-dimensional Fourier transform may be computed as the discrete Fourier transform of the conjugate and can also be computed as a two-step one-dimensional discrete Fourier transform. Thus, a one-dimensional discrete Fourier transform will be the nucleus of the computation and more effort should be concentrated on its algorithm design to make it more effective.

The discrete Fourier transform for one-dimensional function is rewritten as follows from Eq. (8.2):

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N} \qquad u = 0, 1, \ldots, N-1 \tag{14.72}$$

Regrouping in the same manner as in Eqs. (14.60) and (14.61) the constant multiplication factor for the convenience of analysis and letting $W = e^{-j2\pi/N}$, the discrete Fourier transform becomes

$$F(u) = \sum_{x=0}^{N-1} f(x)W^{ux} \qquad u = 0, 1, \ldots, N-1 \tag{14.73}$$

giving a system of N simultaneous equations, corresponding to N different values of u. It can be easily seen from Eq. (14.73) that N complex multiplications and N complex additions are needed for each equation, or a total of $2N^2$ complex arithmetic operations for the whole array [or a total of N^2 operations when $f(x)$ is a real function]. When N becomes large, the number of computations involved in the Fourier transform is terribly great. Hence, computation must be simplified to make the transformation technique practical. It became so only when the fast Fourier transform (FFT) was suggested in 1965. The fundamental principle in FFT algorithm is based on the decomposition of DFT computation of a sequence of length N into successively smaller DFTs.

Assume that $N = 2^L$, where L is a positive integer; Eq. (14.73) can then be broken into two parts:

$$F(u) = \sum_{\substack{\text{even}\\ x}} f(x)W_N^{ux} + \sum_{\substack{\text{odd}\\ x}} f(x)W_N^{ux} \qquad u = 0, 1, \ldots, N-1 \tag{14.74}$$

where $W_N = e^{-j2\pi/N}$. There are still N terms (or a sequence of N terms), in each equation of the system above. Equation (14.74) can, in turn, be put in the following form with r equal to a positive integer:

$$F(u) = \sum_{r=0}^{(N/2)-1} f(2r)W_N^{2ru} + \sum_{r=0}^{(N/2)-1} f(2r+1)W_N^{(2r+1)u} \tag{14.75}$$

or

$$F(u) = \sum_{r=0}^{(N/2)-1} f(2r)(W_N^2)^{ru} + W_N^u \sum_{r=0}^{(N/2)-1} f(2r+1)(W_N^2)^{ru} \tag{14.76}$$

The first summation on the right-hand side consists of a sequence of $N/2$ terms. Note from the definition of W_N that

$$W_N^2 = (e^{-j2\pi/N})^2 = e^{-j2\pi/(N/2)} = W_{N/2} \tag{14.77}$$

Use $W_{N/2}$ as the kernel for the sequence of $N/2$ terms; then we have

$$F(u) = \sum_{r=0}^{(N/2)-1} f(2r)(W_{N/2})^{ru} + W_N^u \sum_{r=0}^{(N/2)-1} f(2r+1)(W_{N/2})^{ru} \tag{14.78}$$

or

$$F(u) = G(u) + W_N^u H(u) \tag{14.79}$$

where

$$G(u) = \sum_{r=0}^{(N/2)-1} f(2r)(W_{N/2})^{ru}$$

and

$$H(u) = \sum_{r=0}^{(N/2)-1} f(2r+1)(W_{N/2})^{ru}$$

are two $(N/2)$-point discrete Fourier transforms. Note also that

$$(W_{N/2})^{u+N/2} = (W_{N/2})^{u} \tag{14.80}$$

Both $G(u)$ and $H(u)$ are periodic in u with a period of $N/2$. Careful analysis of Eqs. (14.74) through (14.80) reveals some interesting properties of these expressions. It is noted in Eqs. (14.78) and (14.79) that an N-point discrete transform can be computed by dividing the original expression into two parts. Each part corresponds to a $(N/2)$-point discrete transform computation. Obviously, the computation time required for $(N/2)$-point DFT will be more greatly reduced than that for N-point DFT.

By continuing this analysis, the $(N/2)$-point discrete transform can be obtained by computing two $(N/4)$-point discrete transforms, and so on, for any N that is equal to an integer power of 2. The implementation of these equations constitutes the successive-doubling FFT algorithm.

The implementation (Sze, 1979) of Eq. (14.78) is shown in Figure 14.41 for $N = 8$. Inputs to the upper $(N/2)$-point DFT block are $f(x)$'s for even values of x, while those for the lower $(N/2)$-point DFT block are $f(x)$'s for odd values of x. Substitution of values 0, 1, 2, and 3 for u in

$$G(u) = \sum_{r=0}^{(N/2)-1} f(2r)(W_{N/2})^{ru} \tag{14.81}$$

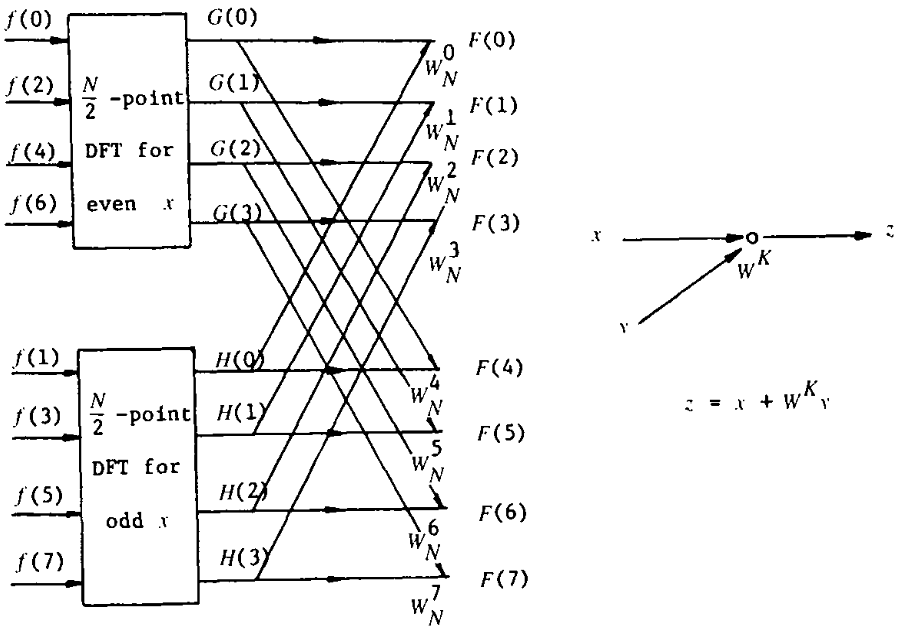


FIGURE 14.41 Implementation of Eq. (14.78) for $N = 8$.

and

$$H(u) = \sum_{r=0}^{(N/2)-1} f(2r+1)(W_{N/2})^{ru} \tag{14.82}$$

gives $G(0), \ldots, G(3)$ and $H(0), \ldots, H(3)$, which combine according to the signal flow graph indicated in the figure to give the Fourier transforms $F(u)$'s, $u = 0, 1, \ldots, 7$. $W_N^0, \ldots, W_N^7$ on the graph indicate the multiplying factors on $H(u)$'s, $u = 0.1, \ldots, 3$, needed in Eq. (14.79). Note that $G(u)$ and $H(u)$ are periodic in u with a period of $N/2$, which is 4 in this case [i.e., $G(4) = G(0)$; $G(5) = G(1)$; $H(4) = H(0)$; $H(5) = H(1)$; etc.]. Thus $F(7) = G(7) + W_N^7 H(7) = G(3) + W_N^7 H(3)$. With this doubling algorithm, the number of complex operations needed for the eight-point DFT is reduced. The total number of complex operations required before using the doubling algorithm is 8^2 or 64, whereas that needed after the doubling algorithm is used is $8 + 2(8/2)^2$ or 40, where $(8/2)^2$ is the number of mathematics operations needed for each of the (8/2)-pint DFTs, and 8 (the first term in the expression) is the number of addition operations needed. Replacing $f(2r)$ by $g(r)$ and letting $r = 2l$ in Eq. (14.78), we then separate $G(u)$ into two parts, one for even r's and the other one for odd r's. We then have

$$G(u) = G_1(u) + W_{N/2}^u G_2(u) \tag{14.83}$$

where

$$G_1(u) = \sum_{l=0}^{(N/4)-1} g(2l) W_{N/4}^{lu} \tag{14.84}$$

represents that part of $G(u)$ for even values of r, and

$$G_2(u) = \sum_{l=0}^{(N/4)-1} g(2l+1) W_{N/4}^{lu} \tag{14.85}$$

represents that part of $G(u)$ for odd values of r. $G_1(u)$ and $G_2(u)$ are periodic with period of $N/4$. Similarly, we have

$$H(u) = H_1(u) + W_{N/2}^u H_2(u) \tag{14.86}$$

where $H_1(u)$ represents that part of $H(u)$ for even values of r, and $H_2(u)$ for odd values of r. Again $H_1(u)$ and $H_2(u)$ are periodic with a period of $N/4$. Then the $(N/2)$-point DFT is decomposed as shown in Figure 14.42. The upper $(N/4)$-point DFT block implement Eq. (14.84) and the lower $(N/4)$-point DFT block implements Eq. (14.85).

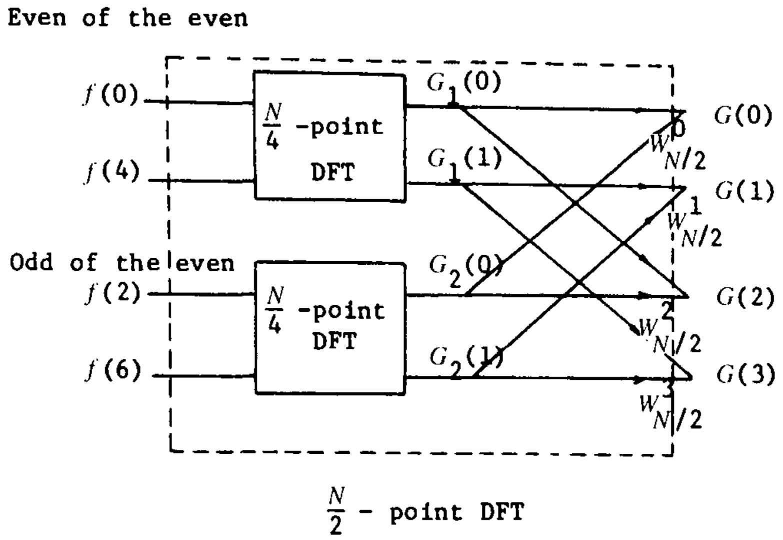


FIGURE 14.42 Implementation of Eq. (14.83) for $N = 8$.

The decomposition of the DFT computation keeps on going until $(N/2^{L-1})$-point DFT becomes a two-point DFT. For the case we have now, that is, $N = 8$, $(N/4)$-point DFT is a two-point DFT, where

$$\begin{aligned} G_1(0) &= f(0) + f(4)W_{N/4}^0 = f(0) + f(4) \\ G_1(1) &= f(0) + W_{N/4}^1 f(4) = f(0) - f(4) \end{aligned} \tag{14.87}$$

The complete 8-point DFT (or FFT for the case $N = 8$) is implemented as shown in Fig. 14.43.

By means of the successive-doubling algorithm, the total number of complex operations changes from the original N^2 to $N + 2(N/2)^2$, and then to $N + 2[N/2 + 2(N/4)^2]$, and so on, depending on the number of stages into which the N-point DFT can be decomposed. If N is large and equal to 2^L, then the number of stages is L, and the number of complex operations changes from N^2 to $N + N + \cdots + N = N \times L = N \log_2 N$. For the example just given, the total number of complex operations will be $N \times L = 8 \times 3 = 24$.

To maintain the structure of this algorithm, the inputs to the DFT block must be arranged in the order required for successive applications of Eq. (14.78). For the FFT computation of an eight-point function $\{f(0), f(1), \ldots, f(7)\}$, inputs with even arguments $f(0), f(2), f(4), f(6)$ are used for the upper $(N/2)$-point DFT (four-point DFT in this case), while those with odd arguments $f(1), f(3), f(5), f(7)$ are used for the lower four-point DFT. Each four-point transform is computed as two-point transforms. We must divide the first set of inputs into its even part $\{f(0), f(4)\}$ and odd part $\{f(2), f(6)\}$, and divide the second set of inputs into $\{f(1), f(5)\}$ as the even part and $\{f(3), f(7)\}$ as the odd part. That is to

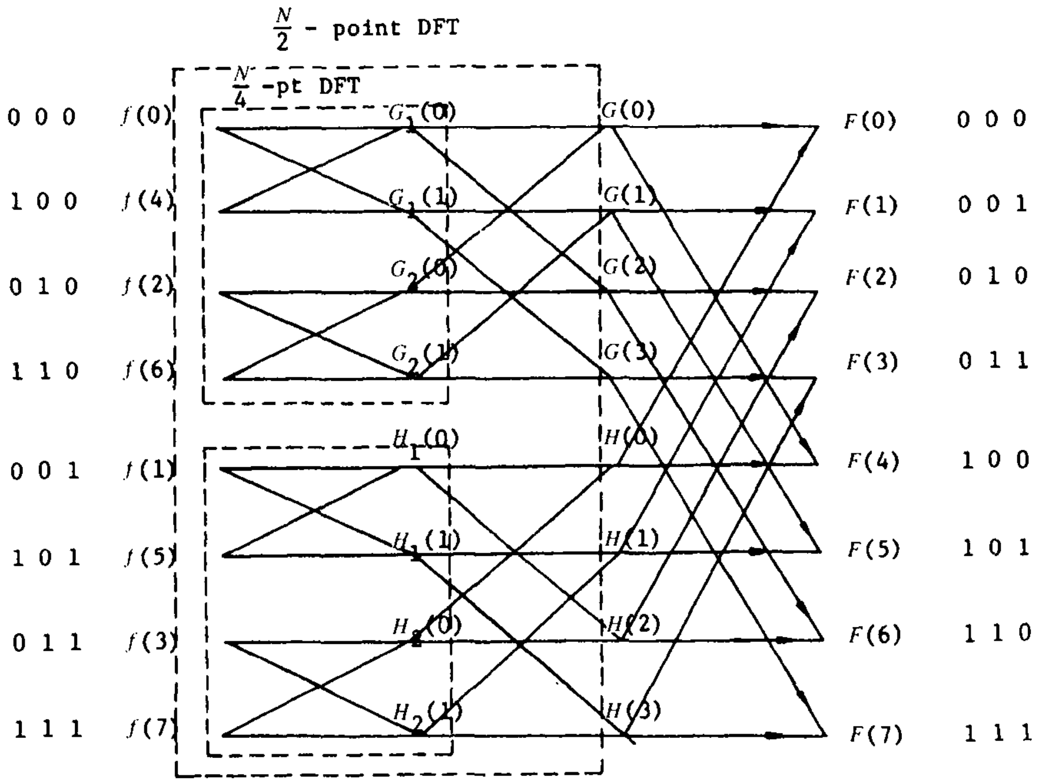


FIGURE 14.43 Complete eight-point DFT.

say, we must arrange the inputs in the order $\{f(0), f(4), f(2), f(6), f(1), f(5), f(3), f(7)\}$ for the successive-doubling algorithm of an eight-point function as shown in Figure 14.44. It is not difficult to note that the input and output are related by a "bit-reversal" order, as shown in Table 14.1. Note that in Figure 14.43, $W_N = e^{-j2\pi/N}$ and $N = 8$; we then have $W_N^0 = 1$, $W_N^4 = -1$, $W_N^5 = -W_N^1$, $W_N^6 = -W_N^2$, and $W_N^7 = -W_N^3$. Utilizing these relations, Figure 14.45 results. By the same argument if $N = 16$, the number of stages is $\log_2 16$ or 4, and the number of complex operations is $16 \log_2 16$ or 64. The reordering of the inputs

TABLE 14.1 Example of Bit Reversal and Recording of Inputs for FFT Algorithm

					Bit-reversal				
Input order	Argument	Binary-coded			binary-coded			New argument	Outputorder
f(0)	0	0	0	0	0	0	0	0	F(0)
f(4)	4	1	0	0	0	0	1	1	F(1)
f(2)	2	0	1	0	0	1	0	2	F(2)
f(6)	6	1	1	0	0	1	1	3	F(3)

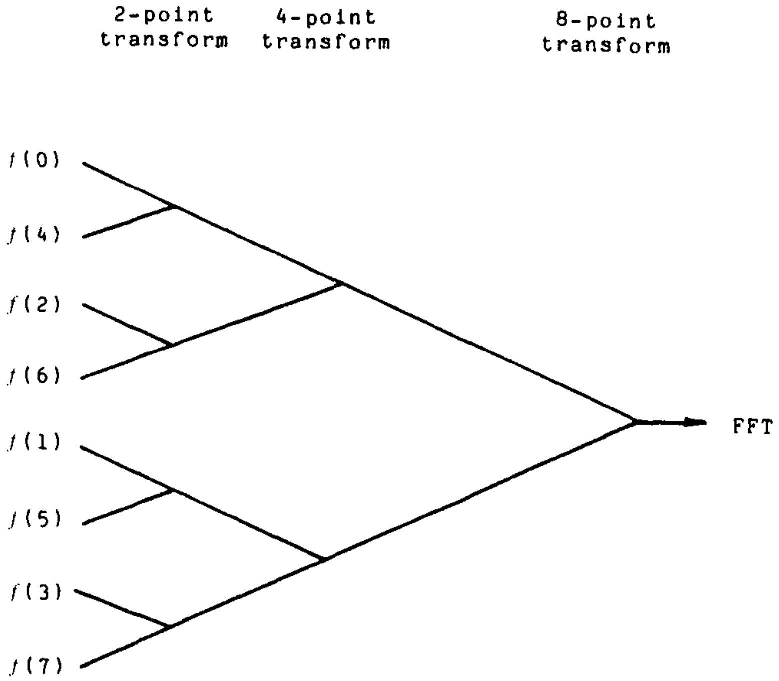


FIGURE 14.44 Reordering of inputs for successive-doubling method.

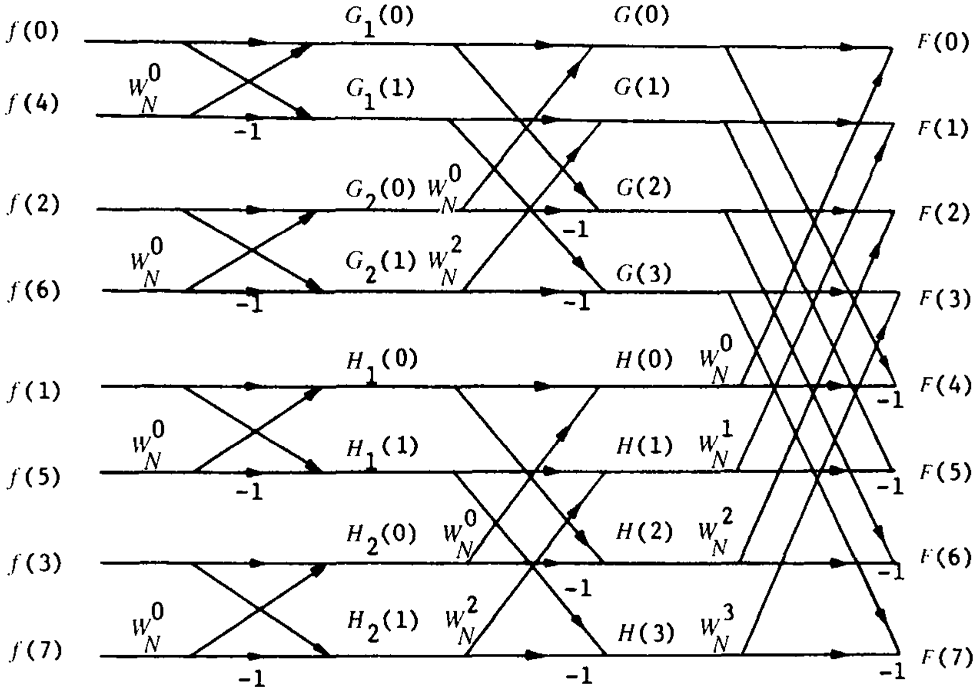


FIGURE 14.45 Bit-reversal-ordering relationship between input and output in the FFT algorithm.

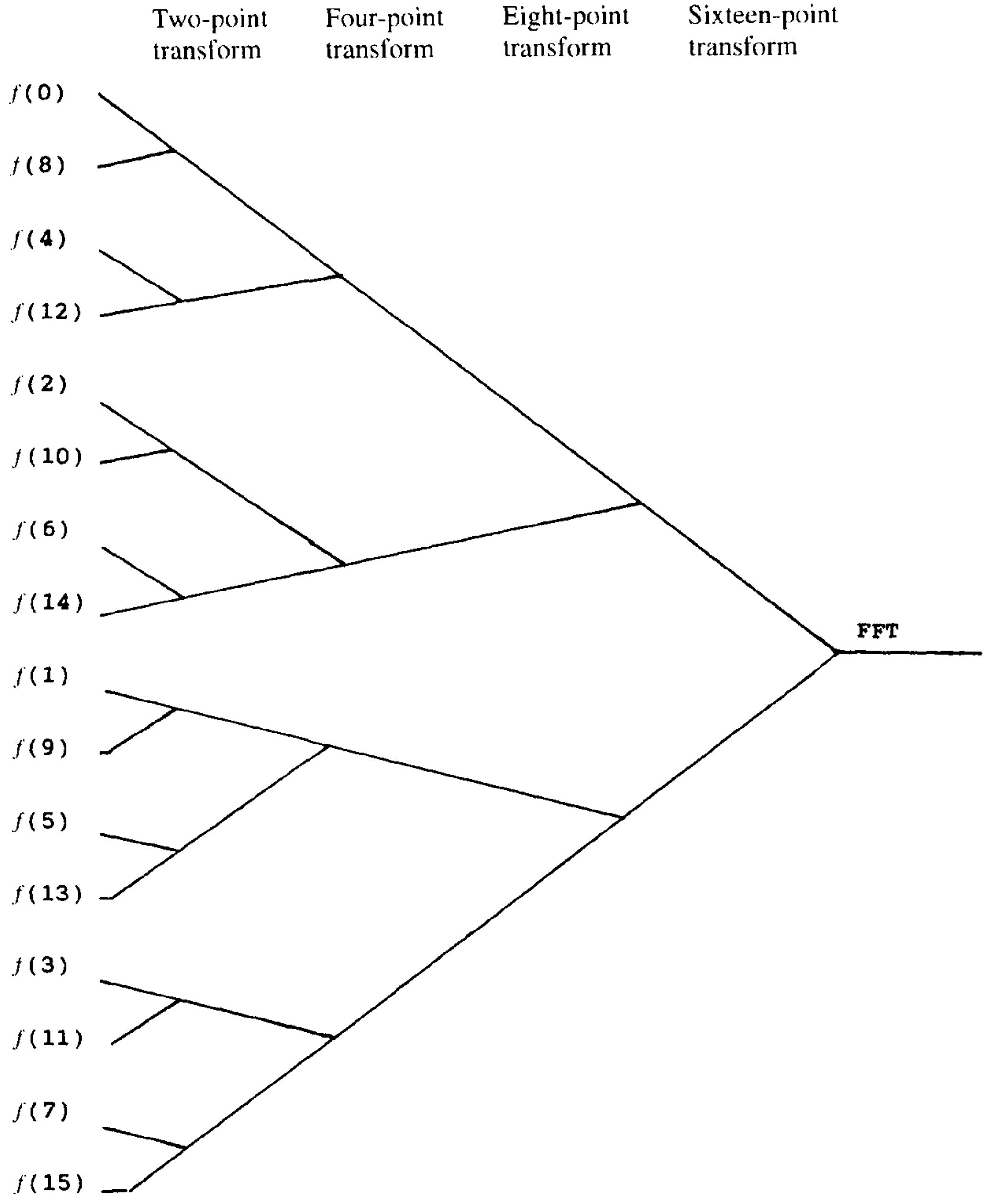


FIGURE 14.46 Reordering of inputs for successive-doubling method when $N = 16$.

for the successive-doubling algorithm and its implementation are shown, respectively, in Figures 14.46 and 14.47.

As mentioned earlier, the Fourier transform technique became widely used only after the effective successive-doubling FFT implementation was suggested in 1969 by Cooley et al. Figure 14.48 shows a FORTRAN implementation of the FFT algorithm suggested by them. This program consists of four parts, the first part being parameter specification (from lines 1 to 6). The second part (i.e., from

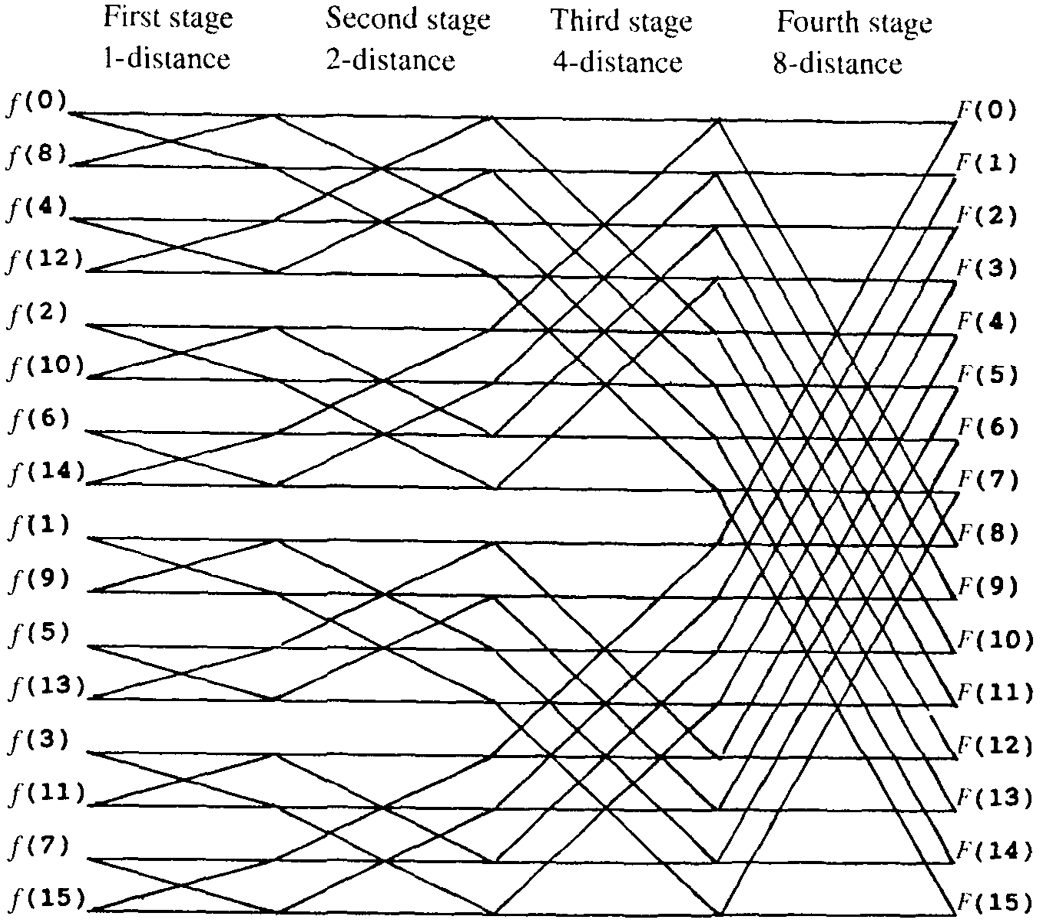


FIGURE 14.47 Bit-reversal order relationship between input and output in the FFT algorithm when $N = 16$.

lines 7 to 18), including the "DO 3" loop, takes care of the bit-reversal-order processing of the input data for later successive-doubling computations. The third part of the program (from lines 19 to 30), including the "DO 5" loop, performs successive-doubling calculations as required. In the final part, the "DO 6" loop divides the results by N. Readers can analyze this program using $N = 16$ and see the detailed steps in getting the input data reordered, such as

$$F(2) \leftrightarrow F(9)$$
$$F(3) \leftrightarrow F(5)$$
$$F(4) \leftrightarrow F(13)$$
$$F(6) \leftrightarrow F(11)$$
$$F(8) \leftrightarrow F(15)$$
$$F(12) \leftrightarrow F(14)$$

```
          SUBROUTINE FFT(F,LN)
          COMPLEX F(1024),U,W,T,CMPLX
          PI=3.141593
          N=2**LN
          NV2=N/2
          NM1=N-1
          J=1
          DO 3 I=1,NM1
             IF(I.GE.J) GO TO 1
             T=F(J)
             F(J)=F(I)
             F(I)=T
1            K=NV2
2            IF(K.GE.J) GO TO 3
             J=J-K
             K=K/2
             GO TO 2
3            J=J+K
          DO 5 L=1,LN
             LE=2**L
             LE1=LE/2
             U=(1.0, 0.0)
             W=CMPLX(COS(PI/LE1),-SIN(PI/LE1))
             DO 5 J=1,LE1
                DO 4 I=J,N,LE
                   IP=I+LE1
                   T=F(IP)*U
                   F(IP)=F(I)-T
4                  F(I)=F(I)+T
5               U=U*W
             DO 6 I=1,N
6               F(I)=F(I)/FLOAT(N)
             RETURN
             END
```

FIGURE 14.48 A FORTRAN implementation of the successive-doubling FFT algorithm. (Adapted from Cooley et al., 1969.)

where ↔ indicates the exchange of the two input functions; or

$$
\begin{aligned}
f(1) &\leftrightarrow f(8)\\
f(2) &\leftrightarrow f(4)\\
f(3) &\leftrightarrow f(12)\\
f(5) &\leftrightarrow f(10)\\
f(7) &\leftrightarrow f(14)\\
f(11) &\leftrightarrow f(13)
\end{aligned}
$$

These exchanges are shown in symbolically Figure 14.49.

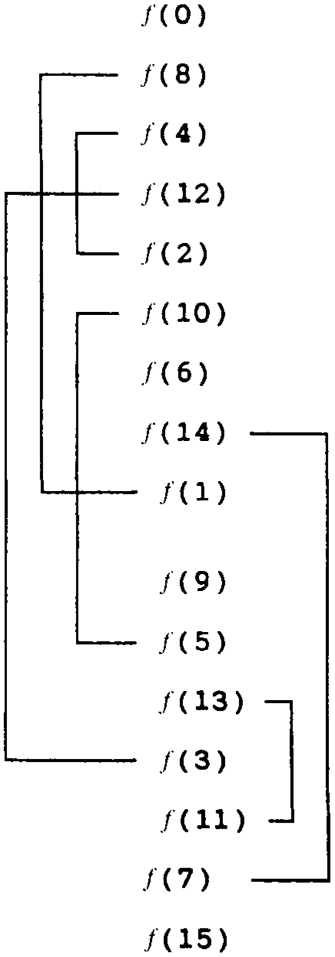


FIGURE 14.49 Input data after bit-reversal processing for $N = 16$.

For $N = 16$ the "big DO 5" loop repeats four times. This corresponds to $\log_2 16$, or four stages. During the first stage, eight butterfly computations are performed between $f(0)$ and $f(8)$, $f(4)$ and $f(12)$, $f(2)$ and $f(10)$, $f(6)$ and $f(14)$, $f(1)$ and $f(9)$. $f(5)$ and $f(13)$, $f(3)$ and $f(11)$, and between $f(7)$ and $f(15)$. Note that inputs used for computation are at a "one-distance" apart. During the second stage, eight butterfly computations are performed. But the computations are now carried out at a "two-distance" apart, with the first four inputs, $f(0)$, $f(8)$, $f(4)$, and $f(12)$, as a subgroup; $f(2)$, $f(10)$, $f(6)$, and $f(14)$ as another subgroup; and so on, as shown clearly in Figure 14.47. In the third-stage computation, the "four-distance" spacings are chosen such that butterfly computations are carried out between $f(0)$ and $f(2)$, $f(8)$ and $f(10)$, $f(4)$ and $f(6)$, and $f(12)$ and $f(14)$. In the fourth stage, which is the last stage when $N = 16$, "eight-distance" butterfly computations are performed: between $f(0)$ and $f(1)$, $f(8)$ and $f(9)$, $f(4)$ and $f(5)$, $f(12)$ and $f(13)$, $f(2)$ and $f(3)$, and so on.

Our earlier discussion on two-dimensional, forward, and inverse FFT provided the necessary information for their implementation. Remember that the same forward FFT is applicable to the inverse transform by using the complex conjugate of the Fourier transform as the input to the FFT subroutine. A FORTRAN implementation of the successive-doubling two-dimensional FFT algorithm can be designed as shown in Figure 14.50 by taking advantage of the ingenious one-dimensional FFT as suggested by Cooley et al.

Owing to the fact that $W_N^{r+N/2} = -W_N^r$, the number of complex multiplications needed for the successive-doubling FFT computational configuration would be further reduced by a factor of 2. Therefore, the total number of complex operations needed for one-dimensional DFT would be $(N \log_2 N)/2$. For a two-dimensional $N \times N$ image function $f(x, y)$, we need $(N \log_2 N)/2$ computational operations for one value of u, and therefore $(N \times N \log_2 N)/2$ for N values of u. By the same reasoning, we need $(N \times N \log_2 N)/2$ operations for N values of v. The total number of complex operations will then be $N^2 \log_2 N$. But if the two-dimensional Fourier transform is evaluated directly, $N^2(N^2) = N^4$ complex

```
C       **** FFT MAIN PROGRAM ****

C       CALL FFT SUBROUTINE FOR X DIRECTION

            DO 20 I=1,128
            DO 30 J=1,128
            F(J)=AR(I,J)*((-1)**(I+J))
30      CONTINUE
        LN=7
            CALL FFT(F,LN)
            DO 40 J=1,128
            AR(I,J)=128*F(J)
40      CONTINUE
20      CONTINUE

C       CALL FFT SUBROUTINE FOR Y DIRECTION

            DO 50 J=1,128
                DO 60 I=1,128
                F(I)=AR(I,J)
60      CONTINUE
        LN=7
            CALL FFT(F,LN)
            DO 70 I=1,128
            AR(I,J)=F(I)
70      CONTINUE
50      CONTINUE

            END
```

FIGURE 14.50 A FORTRAN implementation of the two-dimensional FFT algorithm.

TABLE 14.2 Computational Advantage Obtained for Various Values of N

N	L	Conventional two-dimensional FFT, N^4	Two-dimensional FFT, $N^2 \log_2 N$	Computational advantage, $N^2/\log_2 N$
2	1	16	4	4.00
4	2	256	32	8.00
8	3	4.096	192	21.33
16	4	6.554×10^4	1024	64.00
32	5	1.049×10^6	5120	204.8
64	6	1.678×10^7	2.458×10^4	682.67
128	7	2.684×10^8	1.147×10^5	2341.0
256	8	4.295×10^9	5.243×10^5	8192.0
512	9	6.872×10^{10}	2.359×10^6	2.913×10^4
1024	10	1.100×10^{12}	1.049×10^7	1.049×10^5

operations will be required. Table 14.2 shows a comparison of N^4 versus $N^2 \log_2 N$ for various values of N.

Thus far, discussions have emphasized the forward FFT. As discussed in Section 14.4.1, the inverse transform can be performed using the same transformation algorithm as long as $F(u, v)$ is conjugated to $F^*(u, v)$.

14.5 OTHER IMAGE TRANSFORMS

The Fourier transform is just one of the transformation techniques frequently used in image processing. Other transformation techniques have also been shown to be very effective: Walsh transforms, Hadamard transforms, Karhunen-Loéve transforms, and so on. Like Fourier transforms, all of these transforms are reversible; that is, both forward and inverse transforms can be operated on functions that are continuous and integrable, thus making it possible to process an image in the transform domain.

14.5.1 Walsh Transform

If the function

$$g(x, u) = \frac{1}{N}\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)} \tag{14.88}$$

is used for the forward transform kernel in the generalized transformation equation (14.10), the transformation is known as the Walsh transform. Thus the Walsh transform of a function $f(x)$ is

$$W(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)} \tag{14.89}$$

where N is the number of samples and is assumed to be 2^n, with n as a positive integer. $b_k(z)$ represents the kth bit in the binary representation of z with the zeroth bit as the least significant one. For example, if $n = 4$, $z = 13$ (1 1 0 1 in binary representation), then $b_0(z) = 1$, $b_1(z) = 0$, $b_2(z) = 1$, and $b_3(z) = 1$. The kernel for $n = 4$ is

$$\begin{aligned} g(x, u) &= \frac{1}{N}\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)} \\ &= \frac{1}{N}[(-1)^{b_0(x)b_3(u)+b_1(x)b_2(u)+b_2(x)b_1(u)+b_3(x)b_0(u)}] \end{aligned}$$

By substituting $x = 5$, $u = 7$ in the expression above, we obtain the value of the kernel for the circled entry in Figure 14.51 as

$$\begin{aligned} g(5, 7) &= \frac{1}{N}[(-1)^{1\times0+0\times1+1\times1+0\times1}] \\ &= \frac{1}{N}[(-1)^1] = -\frac{1}{N} \end{aligned}$$

which is a negative value. It can be seen from Figure 14.51 that the kernel is symmetrical and orthogonal, and therefore the inverse kernel $h(x, u)$ is identical to the forward kernel, except for the constant multiplicative factor $1/N$. Hence we have

$$h(x, u) = \prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)} \tag{14.90}$$

The inverse Walsh transform is then

$$f(x) = \sum_{u=0}^{N-1} W(u)\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)} \tag{14.91}$$

Let us start from the smallest N ($N = 2$) and see how the array builds up with the Walsh transformation kernel. When $N = 2$ (or $n = 1$), Eq. (14.88) becomes

$$g(x, u) = \frac{1}{N}(-1)^{b_0(x)b_0(u)}$$

u \ x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
1	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-
2	+	+	+	+	-	-	-	-	+	+	+	+	-	-	-	-
3	+	+	+	+	-	-	-	-	-	-	-	-	+	+	+	+
4	+	+	-	-	+	+	-	-	+	+	-	-	+	+	-	-
5	+	+	-	-	+	+	-	-	-	-	+	+	-	-	+	+
6	+	+	-	-	-	-	+	+	+	+	-	-	-	-	+	+
7	+	+	-	-	-	⊖	+	+	-	-	+	+	+	+	-	-
8	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
9	+	-	+	-	+	-	+	-	-	+	-	+	-	+	-	+
10	+	-	+	-	-	+	-	+	+	-	+	-	-	+	-	+
11	+	-	+	-	-	+	-	+	-	+	-	+	+	-	+	-
12	+	-	-	+	+	-	-	+	+	-	-	+	+	-	-	+
13	+	-	-	+	+	-	-	+	-	+	+	-	-	+	+	-
14	+	-	-	+	-	+	+	-	+	-	-	+	-	+	+	-
15	+	-	-	+	-	+	+	-	-	+	+	-	+	-	-	+

FIGURE 14.51 Values of the Walsh transformation kernel for $N = 16$.

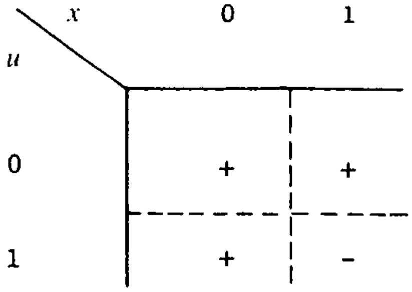
The simplest kernel in the Walsh transformation will be that as shown in Figure 14.52. For $N = 4$ (or $n = 2$), Eq. (14.88) becomes

$$g(x, u) = \frac{1}{N}(-1)^{b_0(x)b_1(u)+b_1(x)b_0(u)}$$

The corresponding Walsh transformation kernel will be the array shown in Figure 14.53. Following the same process of arithmetic substitution, the arrays formed for the Walsh transformation kernel for $N = 8$ (or $n = 3$), and for $N = 16$ (or $n = 4$) are shown in Figures 14.54 and 14.51, respectively.

Extending the Walsh transformation as derived for the two-dimensional case, we obtain the transformation kernel pair as follows:

$$g(x, y; u, v) = \frac{1}{N}\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)} \qquad (14.92)$$



u \ x	0	1
0	+	+
1	+	-

FIGURE 14.52 Values of the Walsh transformation kernel for $N = 2$.

and

$$h(x, y; u, v) = \frac{1}{N}\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)} \tag{14.93}$$

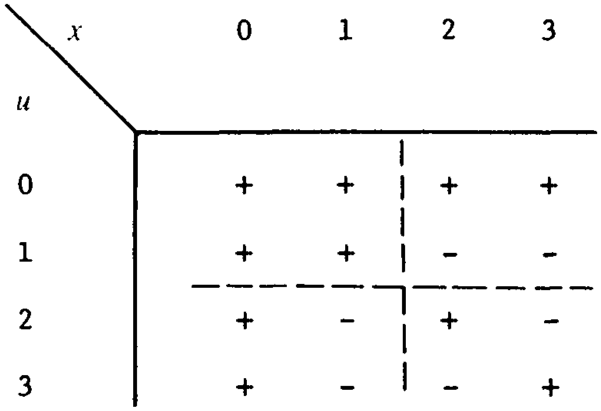
As discussed in Eq. (14.90), the same kernel can be used for both forward and inverse transformation, so we can then write the Walsh transform pair as follows:

$$W(u, v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y)\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)} \tag{14.94}$$

and

$$f(x, y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} W(u, v)\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)} \tag{14.95}$$

Equations (14.94) and (14.95) demonstrate that one algorithm can be used for the computation of both the forward and inverse two-dimensional Walsh transforms.



u \ x	0	1	2	3
0	+	+	+	+
1	+	+	-	-
2	+	-	+	-
3	+	-	-	+

FIGURE 14.53 Values of the Walsh transformation kernel for $N = 4$.

u \ x	0	1	2	3	4	5	6	7
0	+	+	+	+	+	+	+	+
1	+	+	+	+	-	-	-	-
2	+	+	-	-	+	+	-	-
3	+	+	-	-	-	-	+	+
4	+	-	+	-	+	-	+	-
5	+	-	+	-	-	+	-	+
6	+	-	-	+	+	-	-	+
7	+	-	-	+	-	+	+	-

FIGURE 14.54 Values of the Walsh transformation kernel for $N = 8$.

It is obvious from Eqs. (14.94) and (14.95) that the transformation kernels $g(x, y; u, v)$ and $h(x, y; u, v)$ are symmetrical and separable, or

$$g(x, y; u, v) = g_1(x, u)g_2(y, v) \tag{14.96}$$

$$h(x, y; u, v) = h_1(x, u)h_2(y, v) \tag{14.97}$$

where

$$g_1(x, u) = h_1(x, u) = \frac{1}{\sqrt{N}} \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)}$$

and

$$g_2(y, v) = h_2(y, v) = \frac{1}{\sqrt{N}} \prod_{i=0}^{n-1} (-1)^{b_i(y)b_{n-1-i}(v)}$$

That is, both the computation of a two-dimensional Walsh transform $W(u, v)$ and the computation of its inverse transform can be done by successive applications of a one-dimensional Walsh transform, and one algorithm can be used for all those computations. The procedures in computation will be the same as for Fourier transform. Analogous to the fast Fourier transform, a fast algorithm in the form of successive doubling can also be written for the Walsh transform. If the multiplying factors 1, W, $W^2, \ldots,$ in FFTs are all omitted, the algorithm for the fast Walsh transform (FWT) and that of the fast Fourier Transform will be similar,

```
      SUBROUTINE FWT(F,LN)
      REAL F(1024),T
      N=2**LN
      NV2=N/2
      NM1=N-1
      J=1
      DO 3 I=1,NM1
         IF(I.GE.J) GO TO 1
         T=F(J)
         F(J)=F(I)
         F(I)=T
1        K=NV2
2        IF(K.GE.J) GO TO 3
         J=J-K
         K=K/2
         GO TO 2
3        J=J+K
      DO 5 L=1,LN
         LE=2**L
         LE1=LE/2
         DO 5 J=1,LE1
            DO 4 I=J,N,LE
               IP=I+LE1
               T=F(IP)
               F(IP)=F(I)-T
4              F(I)=F(I)+T
5           CONTINUE
      DO 6 I=1,N
6        F(I)=F(I)/FLOAT(N)
      RETURN
      END
```

FIGURE 14.55 A FORTRAN implementation of the successive-doubling FWT algorithm.

and FORTRAN implementation of the successive-doubling FFT algorithm shown in Figure 14.48 can be used for the FWT with U, W, and PI deleted and the word "COMPLEX" changed to "REAL" (see Figure 14.55).

14.5.2 HadamardTransform

If the function

$$g(x, u) = \frac{1}{N}(-1)^{\sum_{i=0}^{n-1} b_i(x) b_i(u)} \tag{14.98}$$

is used for the forward transform kernel in the generalized transformation equation (14.10), the transformation is known as the Hadamard transform.

Thus the Hadamard transform pair is

$$F_H(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)(-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(x)b_i(u)} \qquad u = 0, 1, \ldots, N-1 \tag{14.99}$$

$$f(x) = \sum_{u=0}^{N-1} F_H(u)(-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)} \qquad x = 0, 1, \ldots, N-1 \tag{14.100}$$

where N is the number of samples and is also assumed to be 2^n, with n a positive integer. The same arguments on $b_k(z)$ as those used for the Walsh transform also apply to this transform.

Some properties of the H matrices are useful in their generation:

Property 1. A Hadamard matrix is a square matrix whose rows (and columns) are orthogonal with elements either $+1$ or -1. For an $N \times N$ matrix,

$$H_N H_N^T = NI \tag{14.101}$$

and

$$H_N = H_N^T \tag{14.102}$$

where H_N and H_N^T denote, respectively, a Hadamard matrix and its transpose, and I is an identity matrix. The lowest-order H matrix (i.e., for $N = 2$) is defined as

$$H_2 = \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix} \tag{14.103}$$

Property 2. $H_N^{-1} = \dfrac{1}{N} H_N$

Property 3. A simple recursive algorithm can be used for the construction of the Hadamard transformation matrices, namely,

$$H_{2N} = \begin{vmatrix} H_N & H_N \\ H_N & -H_N \end{vmatrix} \tag{14.105}$$

where H_N and H_{2N} represents matrices of order N and $2N$, respectively. If "+" and "−" are used, respectively, for the "+1" and "−1" entries for notation simplification, then

$$H_2 = \begin{vmatrix} + & + \\ + & - \end{vmatrix}$$

and

$$H_4 = \left|\begin{array}{cc:cc} + & + & + & + \\ + & - & + & - \\ \hdashline + & + & - & - \\ + & - & - & + \end{array}\right|$$

By the same recursive relation, we can write H_8 as

$$H_8 = \begin{vmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{vmatrix} = \begin{array}{cc|cc|cc|cc|c}
 & & & & & & & & \text{Sequence} \\
+ & + & + & + & + & + & + & + & 0 \\
+ & - & + & - & + & - & + & - & 7 \\
\hline
+ & + & - & - & + & + & - & - & 3 \\
+ & - & - & + & + & - & - & + & 4 \\
\hline
+ & + & + & + & - & - & - & - & 1 \\
+ & - & + & - & - & + & - & + & 6 \\
\hline
+ & + & - & - & - & - & + & + & 2 \\
+ & - & - & + & - & + & + & - & 5
\end{array} \qquad (14.106)$$

and H_{16} as

$$H_{16} = \begin{vmatrix} H_8 & H_8 \\ H_8 & -H_8 \end{vmatrix}$$

$$= \begin{array}{cccc|cccc|cccc|cccc|c}
 & & & & & & & & & & & & & & & & \text{Sequence} \\
+ & + & + & + & + & + & + & + & + & + & + & + & + & + & + & + & 0 \\
+ & - & + & - & + & - & + & - & + & - & + & - & + & - & + & - & 15 \\
+ & + & - & - & + & + & - & - & + & + & - & - & + & + & - & - & 7 \\
+ & - & - & + & + & - & - & + & + & - & - & + & + & - & - & + & 8 \\
\hline
+ & + & + & + & - & - & - & - & + & + & + & + & - & - & - & - & 3 \\
+ & - & + & - & - & + & - & + & + & - & + & - & - & + & - & + & 12 \\
+ & + & - & - & - & - & + & + & + & + & - & - & - & - & + & + & 4 \\
+ & - & - & + & - & + & + & - & + & - & - & + & - & + & + & - & 11 \\
\hline
+ & + & + & + & + & + & + & + & - & - & - & - & - & - & - & - & 1 \\
+ & - & + & - & + & - & + & - & - & + & - & + & - & + & - & + & 14 \\
+ & + & - & - & + & + & - & - & - & - & + & + & - & - & + & + & 6 \\
+ & - & - & + & + & - & - & + & - & + & + & - & - & + & + & - & 9 \\
\hline
+ & + & + & + & - & - & - & - & - & - & - & - & + & + & + & + & 2 \\
+ & - & + & - & - & + & - & + & - & + & - & + & + & - & + & - & 13 \\
+ & + & - & - & - & - & + & + & - & - & + & + & + & + & - & - & 5 \\
+ & - & - & + & - & + & + & - & - & + & + & - & + & - & - & + & 10
\end{array} \qquad (14.107)$$

The H matrix formed from the recursive construction algorithm is unordered in sequence (i.e., the number of sign changes in the rows/columns is unordered). This can be reordered by making a change in the kernel, the details of which are discussed later.

The two-dimensional Hadamard transform can be formulated as

$$F_H(u, v) = H(u, v)f(x, y)H(u, v) \tag{14.108}$$

where $F_H(u, v)$ is the Hadamard transform of $f(x, y)$ and $H(u, v)$ is the $N \times N$ symmetric Hadamard transformation matrix. The inverse Hadamard transform of $F_H(u, v)$ is

$$H(u, v)F_H(u, v)H(u, v) \tag{14.109}$$

or

$$H(u, v)H(u, v)f(x, y)H(u, v)H(u, v)$$

after substitution of $F_H(u, v)$ from Eq. (14.108). By using the relation expressed on Eq. (14.101), we have

$$H(u, v)F_H(u, v)H(u, v) = N^2 f(x, y) \tag{14.110}$$

Thus

$$f(\mathbf{x}, \mathbf{y}) = \frac{1}{N^2}H(u, v)F_H(u, v)H(u, v) \tag{14.111}$$

which forms a Hadamard transformation pair with Eq. (14.108).

In order to put the sequence in increasing order, let us let the forward transformation kernel be of the following form:

$$g(x, y; u, v) = \frac{1}{N}(-1)^{\sum_{i=0}^{n-1}[b_i(x)p_i(u)+b_i(y)p_i(v)]} \tag{14.112}$$

The Hadamard transform becomes

$$F_H(u, v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y)(-1)^{\sum_{i=0}^{n-1}[b_i(x)p_i(u)+b_i(y)p_i(v)]} \tag{14.113}$$

where both x and u are in binary representation. $b_k(z)$ represents the kth bit in the binary representation of z with the zeroth bit as the least significant one. $p_l(r)$ is defined as follows:

$$\begin{aligned}
p_0(u) &= b_{n-1}(u) \\
p_1(u) &= b_{n-1}(u) + b_{n-2}(u) \\
p_2(u) &= b_{n-2}(u) + b_{n-3}(u) \\
&\vdots \\
p_{n-1}(u) &= b_1(u) + b_0(u)
\end{aligned} \tag{14.114}$$

and

$$\sum_{i=0}^{n-1} p_i(u) = b_0(u) \qquad (\text{mod } 2) \tag{14.115}$$

where $b_{n-1}(u)$ represents the leftmost binary bit of u; $b_{n-2}(u)$, the next-leftmost bit of u; and so on. The summations in Eqs. (14.114) and (14.115) are performed in modulo 2 arithmetic. Similar arguments apply to $p_i(v)$, $i = 0, 1, \ldots, n-1$.

Example. For the one-dimensional Hadamard transform, compute the values of the ordered Hadamard kernel for $N = 8$ (or $n = 3$).

Solution. When $u = 2$ (0 1 0 in binary representation) and $x = 6$ (1 1 0 in binary), we have

$$p_0(u) = 0$$
$$p_1(u) = b_2(u) + b_1(u) = 0 + 1 = 1$$
$$p_2(u) = b_1(u) + b_0(u) = 1 + 0 = 1$$

$$\begin{aligned}\sum_{i=0}^{n-1} b_i(x)p_i(u) &= b_0(x)p_0(u) + b_1(x)p_1(u) + b_2(x)p_2(u)\\ &= 0 + 1 + 1\\ &= 2\\ &= 0 \text{ (mod-2 arithmetic)}\end{aligned}$$

Hence, the entry (when $\mu = 2$, $x = 6$) in the Hadamard kernel is $(-1)^0$ or $+$.

When $u = 5$ (1 0 1 in binary representation) and $x = 4$ (1 0 0), we have

$$p_0(u) = 1$$
$$p_1(u) = b_2(u) + b_1(u) = 1 + 0 = 1$$
$$p_2(u) = b_1(u) + b_0(u) = 0 + 1 = 1$$

$$\begin{aligned}\sum_{i=0}^{n-1} b_i(x)p_i(u) &= b_0(x)p_0(u) + b_1(x)p_1(u) + b_2(x)p_2(u)\\ &= 0 + 0 + 1 = 1\end{aligned}$$

Hence, the entry (when $\mu = 5$, $x = 4$) in the Hadamard kernel is $(-1)^1$ or $-$.

An ordered Hadamard transformation kernel can be constructed as shown in (14.116).

$u \backslash x$	0	1	2	3	4	5	6	7	Sequence
0	+	+	+	+	+	+	+	+	0
1	+	+	+	+	−	−	−	−	1
2	+	+	−	−	−	−	+	+	2
3	+	+	−	−	+	+	−	−	3
4	+	−	−	+	+	−	−	+	4
5	+	−	−	+	−	+	+	−	5
6	+	−	+	−	−	+	−	+	6
7	+	−	+	−	+	−	+	−	7

(14.116)

By comparing (14.116) with (14.106), we can see that the sequence in (14.116) is ordered.

As can be seen from Eq. (14.112), the kernel for the two-dimensional ordered Hadamard transform is separable. Thus

$$
\begin{aligned}
F_H(u,v) &= \frac{1}{N}\sum_{y=0}^{N-1}\left[\sum_{x=0}^{N-1} f(x,y)(-1)^{\sum_{i=0}^{n-1} b_i(x)p_i(u)}\,(-1)^{\sum_{i=0}^{n-1} b_i(y)p_i(v)}\right] \\
&= \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} F_H(u,y)(-1)^{\sum_{i=0}^{n-1} b_i(y)p_i(v)}
\end{aligned}
\tag{14.117}
$$

where $F_H(u,y)$ is a one-dimensional Hadamard transform. Analogous to the two-dimensional FFT, the one-dimensional Hadamard transform can be successively used for the two-dimensional transformation and a fast algorithm can also be established for it. Analogous to the Fourier transform, the Hadamard transform as expressed by

$$
F_H(u) = \sum_{x=0}^{N-1} f(x)(-1)^{\sum_{i=0}^{n-1} b_i(x)p_i(u)} \tag{14.118}
$$

can be decomposed into the sum of two series of terms as

$$
\begin{aligned}
F_H(u) = &\sum_{r=0}^{(N/2)-1} f(2r)(-1)^{\sum_{i=0}^{n-1} p_i(u)b_i(2r)} \\
&+ \sum_{r=0}^{(N/2)-1} f(2r+1)(-1)^{\sum_{i=0}^{n-1} p_i(u)b_i(2r+1)}
\end{aligned}
\tag{14.119}
$$

Noting that

$$\sum_{i=1}^{n-1} b_i(2r+1) = \sum_{i=1}^{n-1} b_i(2r) \tag{14.120}$$

and

$$\begin{array}{ll} \text{For even } x\text{:} & b_0(x) = b_0(2r) = 0 \\ \text{For odd } x\text{:} & b_0(x) = b_0(2r+1) = 1 \end{array} \tag{14.121}$$

we have

$$\begin{aligned} \sum_{i=0}^{n-1} p_i(u) b_i(2r+1) &= p_0(u) b_0(2r+1) \\ &\quad + \sum_{i=1}^{n-1} p_i(u) b_i(2r+1) \end{aligned} \tag{14.122}$$

or

$$\sum_{i=0}^{n-1} p_i(u) b_i(2r+1) = p_0(u) + \sum_{i=1}^{n-1} p_i(u) b_i(2r+1) \tag{14.123}$$

since we know that $b_0(2r+1)$ is definitely equal to 1. With knowledge of Eqs. (14.120) and (14.121), Eq. (14.123) can be put in the following form:

$$\sum_{i=0}^{n-1} p_i(u) b_i(2r+1) = p_0(u) + \sum_{i=0}^{n-1} p_i(u) b_i(2r) \tag{14.124}$$

Equation (14.119) then becomes

$$F_H(u) = \left[\sum_{r=0}^{(N/2)-1} f(2r) + (-1)^{p_0(u)} \sum_{r=0}^{(N/2)-1} f(2r+1) \right] (-1)^{\sum_{i=0}^{n-1} p_i(u) b_i(2r)} \tag{14.125}$$

$$F_H(u) = G(u) + (-1)^{b_{n-1}(u)} H(u) \tag{14.126}$$

where

$$G(u) = \sum_{r=0}^{(N/2)-1} f(2r)(-1)^{\sum_{i=0}^{n-1} p_i(u) b_i(2r)} \tag{14.127}$$

$$H(u) = \sum_{r=0}^{(N/2)-1} f(2r+1)(-1)^{\sum_{i=0}^{n-1} p_i(u) b_i(2r)} \tag{14.128}$$

and

$$b_{n-1}(u) = p_0(u) \tag{14.129}$$

The sign of $H(u)$, dominated by $(-1)^{b_{n-1}(u)}$, is positive for $u < N/2$ and negative for $u > N/2$. Following the decomposition procedure, implementation of these equations constitutes the successive FHT algorithm.

14.5.3 Discrete Karhunen-LoéveTransform

We discussed the discrete Karhunen-Loéve transform in detail in Section 7.3. What are we going to add here is the application of this transform in image processing. Let us put the $N \times N$ matrix $f(x, y)$,

$$f(x,y) = \begin{vmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & & & \\ \vdots & & & \vdots \\ f(N-1,0) & & \cdots & f(N-1,N-1) \end{vmatrix} \tag{14.130}$$

into the form of an N^2-element vector as expressed by

$$\mathbf{x}_i = \begin{vmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ij} \\ \vdots \\ x_{iN^2} \end{vmatrix} \tag{14.131}$$

where $\mathbf{x}_i$, $i = 1, 2, \ldots, K$, represent image samples, and $x_{i1}, x_{i2}, \ldots, x_{iN^2}$ correspond, respectively, to $f(0, 0)$, $f(0, 1), \ldots, f(N-1,\ N-1)$ of the ith image sample. The transform can then be treated as a statistical problem. Following the discussions in Chapter 5, we have

$$\mathbf{C}_x = E\{(\mathbf{x} - \mathbf{m}_x)(\mathbf{x} - \mathbf{m}_x)^T\} \tag{14.132}$$

where $\mathbf{C}_x$ is the covariance matrix, and $\mathbf{m}_x$ the mean value of $\mathbf{x}$, both of which can be approximated by

$$\mathbf{m}_x \simeq \frac{1}{K}\sum_{i=1}^{k} \mathbf{x}_i \tag{14.133}$$

and

$$\mathbf{C}_x \simeq \frac{1}{K}\sum_{i=1}^{K} \mathbf{x}_i\mathbf{x}_i^T - \mathbf{m}_x\mathbf{m}_x^T \tag{14.134}$$

where K is the number of image samples, $\mathbf{m}_x$ is an N^2 vector, and $\mathbf{C}_x$ is an $N^2 \times N^2$ matrix. The problem we now have is to transform the original image vector $\mathbf{x}$ into a new image vector $\mathbf{y}$ so that the covariance matrix $\mathbf{C}_y$ will be a diagonal one. Thus we have

$$\mathbf{y} = \mathbf{B}(\mathbf{x} - \mathbf{m}_x) \tag{14.135}$$

where $(\mathbf{x} - \mathbf{m}_x)$ represents the centralized vector, and the $N^2 \times N^2$ matrix $\mathbf{B}$ is chosen such that its rows are eigenvectors of $\mathbf{C}_x$; thus

$$\mathbf{B} = \begin{vmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \vdots \\ \mathbf{e}_i \\ \vdots \\ \mathbf{e}_{N^2} \end{vmatrix} = \begin{vmatrix} e_{11} & & \cdots & e_{1N^2} \\ e_{21} & & \cdots & e_{2N^2} \\ \vdots & & & \vdots \\ e_{i1} & & & e_{iN^2} \\ \vdots & & & \vdots \\ e_{N^21} & e_{N^22} & \cdots & e_{N^2N^2} \end{vmatrix} \tag{14.136}$$

where $\mathbf{e}_i = [e_{i1}, \ldots, e_{iN^2}]$ is the ith eigenvector of $\mathbf{C}_x$ and e_{ij} is the jth component of the ith eigenvector. The new covariance matrix $\mathbf{C}_y$ is then

$$\begin{aligned} \mathbf{C}_y &= E\{B(\mathbf{x} - \mathbf{m}_x)(\mathbf{x} - \mathbf{m}_x)^T\mathbf{B}^T\} \\ &= \mathbf{B}\mathbf{C}_x\mathbf{B}^T \end{aligned} \tag{14.137}$$

which is a diagonal matrix, for the reasons given below. Since

$$\mathbf{y} = \mathbf{B}(\mathbf{x} - \mathbf{m}_x) \tag{14.138}$$

Then

$$\mathbf{x} - \mathbf{m}_x = \mathbf{B}^T\mathbf{y} \tag{14.139}$$

where $\mathbf{y} = [y_1, y_2, \ldots y_p]$ and $\mathbf{B}$ is an orthogonal matrix. Let $\mathbf{B}_r$ denote the rth column of $\mathbf{B}$ (and $\mathbf{B}_r$ the rth row of $\mathbf{B}^T$). Then $\mathbf{B}_1$ is chosen first in such a way that the variance of y_1 is maximized; $\mathbf{B}_2$ is chosen so that the variance of y_2 is maximized subject to the condition that y_2 is uncorrelated with y_1; and similarly for the remaining y's. The variance of y_r is maximized subject to the condition that y_r is uncorrelated with $y_1, y_2, \ldots, y_{r-1}$. Let us denote the variance of y_r by λ_r. Since $\mathbf{y}_r = \mathbf{B}_r^T\mathbf{x}$, we have

$$\lambda_r = \mathbf{B}_r^T\mathbf{C}_x\mathbf{B}_r \tag{14.140}$$

As the y's are uncorrelated, we also have

$$\mathbf{B}_r^T\mathbf{C}_x\mathbf{B}_s = 0 \qquad \text{for } r \neq s \tag{14.141}$$

This means that

$$\mathbf{B}^T\mathbf{C}_x\mathbf{T} = \Lambda \tag{14.142}$$

which is diagonal with elements $\lambda_1, \lambda_2, \ldots, \lambda_p$ arranged in order of magnitude.

$$\mathbf{C}_y = \begin{vmatrix} \lambda_1 & & & 0 & & \\ & \lambda_2 & & & & \\ & & \ddots & & & \\ & & & \lambda_i & & \\ 0 & & & & \ddots & \\ & & & & & \lambda_{N^2} \end{vmatrix} \tag{14.143}$$

with elements equal to the eigenvalues of $\mathbf{C}_x$ (λ_i, $i = 1, 2, \ldots, N^2$), where λ_i is the variance of the ith element of $\mathbf{y}$ along eigenvector $\mathbf{e}_i$. $\mathbf{x}$ can be reconstructed from $\mathbf{y}$ by using

$$\mathbf{x} = \mathbf{B}^T\mathbf{y} + \mathbf{m}_x \tag{14.144}$$

This is because $\mathbf{B}^{-1} = \mathbf{B}^T$ for the orthonormal vectors.

The Karhunen-Loève transform is useful in data compression and image rotation applications. But this transform has the drawback of not being separable, and therefore no fast algorithm exists for computing the transform.

14.6 ENHANCEMENT BY TRANSFORM PROCESSING

14.6.1 Low-Pass Filtering

As discussed at the beginning of this chapter, image enhancement can also be carried out by the transform method. In this method the image $f(x, y)$ is first transformed into $F(u, v)$, and then processed in the transform domain to meet our requirements. Filtering is one of the processes most frequently used in the transform domain. Since convolution in the spatial domain is converted to simpler multiplication in the transform domain, the processing work required is greatly simplified. On the other hand, extra work will be introduced in the transform and inverse transform of the image function to yield the final spatial image that is expected. A trade-off between these two is therefore needed in making the choice as to the domain in which we are going to work.

The entire procedure in transform processing can be put in block form as shown in Figure 14.56, where $f(x, y)$ and $\hat{f}(x, y)$ represent, respectively, the original image and the expected processed image. $H(u, v)$ is the process expected to be used in the transform domain, and $G(u, v)$ is the result after processing in the transform domain, which can be represented by

$$G(u, v) = F(u, v)H(u, v) \tag{14.145}$$

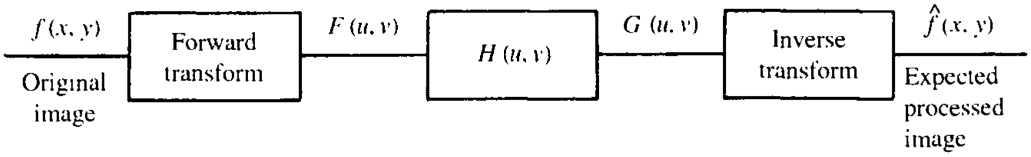


FIGURE 14.56 Block diagram of image processing in the transform domain.

Filtering is one of the processes used to advantage in the transform domain. Digital filtering can be implemented ideally in the transform domain because only pure mathematics is involved rather than physical components.

Various kinds of filters are available. They can be grouped into two main categories: low-pass filters and high-pass filters. As we know, the high-frequency information content of the spectrum (say, a Fourier spectrum) is contributed primarily by the edges and sharp transitions, while the low-frequency information content is contributed by the brightness and the image texture. Depending on what we expect of the processed images, either high-pass or low-pass filters will be chosen to fit the requirements.

Among the low-pass filters, there are for conventional use the ideal low-pass filters, Butterworth filters, exponential low-pass filters, trapezoidal filters, and others. For the ideal low-pass filter shown in Figure 14.57a, the transfer function is

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases} \tag{14.146}$$

where D is the distance from point (u, v) to the origin of the frequency plane such that $D = (u^2 + v^2)^{1/2}$, and D_0 is the cutoff frequency, a specified nonnegative quantity, the value of which depends on what we required in the processed image. D_0 may be obtained by decreasing D until the energy passed exceeds a prescribed percentage of the total energy.

For the Butterworth low-pass filter shown in Figure 14.57b, the transfer function is

$$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}} \tag{14.147}$$

where n is known as the order of the filter, and D_0 is the cutoff frequency, which is defined at the open point on the abscissa where $H(u, v)$ is equal to the one-half of its maximum value. The image processed with this Butterworth filter will be expected to have less blurring effect, since some of the high-frequency-component information will be included in the tail region of this filter, as can be seen in Figure 14.57b.

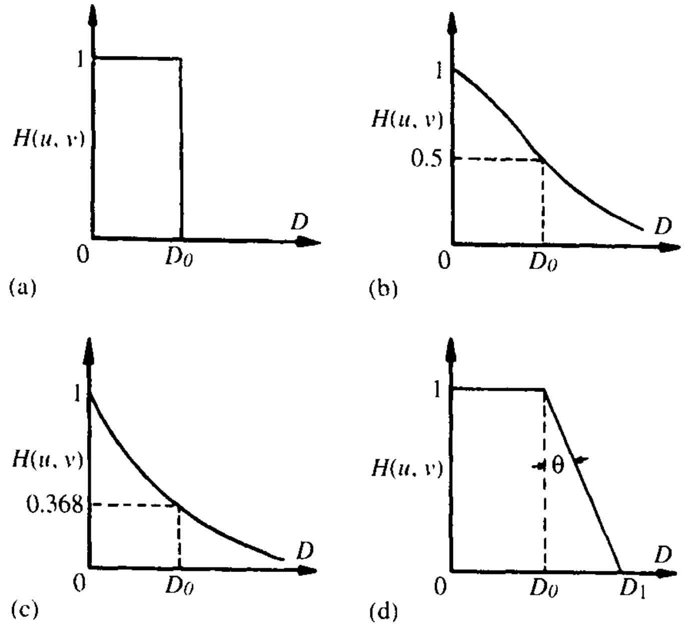


FIGURE 14.57 Low-pass filters. (a) ideal; (b) Butterworth; (c) exponential; (d) trapezoidal.

The exponential low-pass filter is shown in Figure 14.57c. Its transfer function is

$$H(u, v) = e^{-[D(u,v)/D_0]^n} \tag{14.148}$$

The cutoff frequency D_0 is defined at the point on the abscissa where $H(u, v)$ drops to a point equal to 0.368 of its maximum value. n in the transfer function is the variable controlling the rate of decay of the H function. More blurring will be expected from the exponential low-pass filter than from the Butterworth filter, since less high-frequency component information is included in the processed image.

A trapezoidal filter as shown in Figure 14.57d is halfway between an ideal low-pass filter and a completely smooth filter. Depending on the slope of the tail of this trapezoid, the high-frequency-component information content will be different, and therefore the blurring effects will be different for different cases. From the functional diagram shown in Figure 14.57d. $H(u, v)$ assumes a value

$$1 - (D - D_0)\cot\theta = \frac{D_1 - D}{D_1 - D_0}$$

when D falls at a point between D_0 and D_1. Thus we obtain the transfer function of the trapezoidal filter as

$$H(u, v) = \begin{cases} 1 & D < D_0 \\ \dfrac{D - D_1}{D_0 - D_1} \ \text{if} & D_0 \leq D \leq D_1 \\ 0 & D > D_1 \end{cases} \tag{14.149}$$

Figure 14.58 illustrates the blurring that occurred after the processing of an ideal low-pass filter.

14.6.2 High-Pass Filtering

Similar to low-pass filters, we have ideal high-pass filters, Butterworth high-pass filters, exponential high-pass filters, and trapezoidal high-pass filters. As implied by the name "high-pass" the lower-frequency-component information is attenuated without disturbing the high-frequency information. These kinds of filters are generally used to achieve edge sharpening. The transfer functions for these filters are shown in Figure 14.59. Contrary to the transfer function shown by Eq.

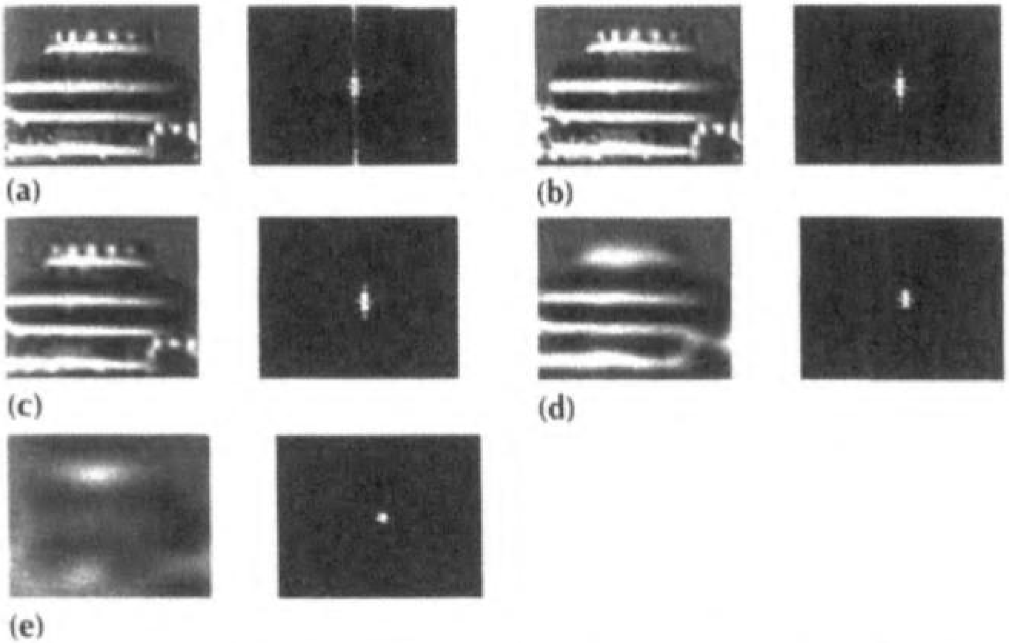


FIGURE 14.58 Blurring process for an ideal low-pass filter. (a) Original image; (b) when $D_0 = 0.25$; (c) when $D_0 = 0.15$; (d) when $D_0 = 0.08$: (e) when $D_0 = 0.04$.

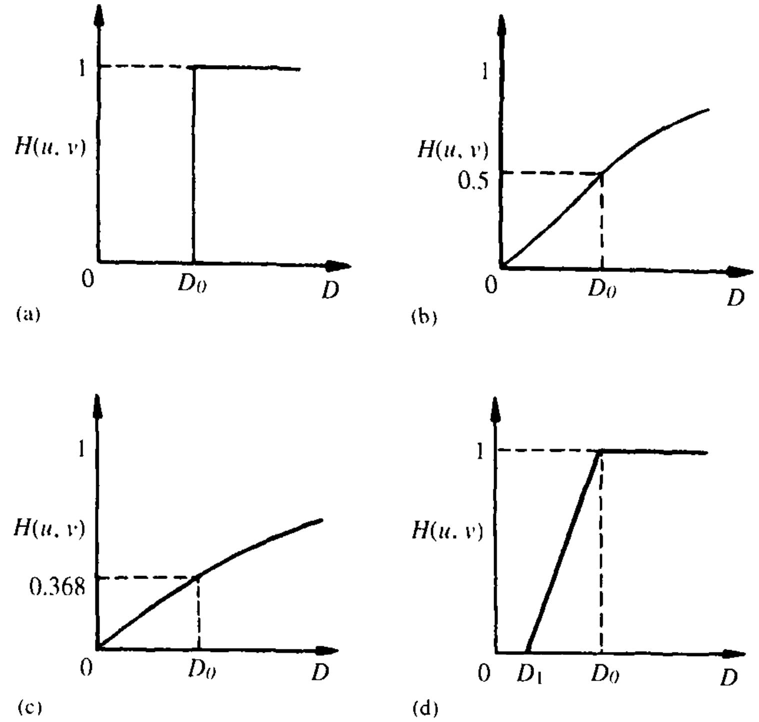


FIGURE 14.59 High-pass filters. (a) Ideal; (b) Butterworth; (c) exponential (d) trapezoidal.

(14.146), the transfer function of an ideal high-pass filter is given by the following relation:

$$H(u, v) = \begin{cases} 0 & \text{if } D < D_0 \\ 1 & \text{if } D > D_0 \end{cases} \tag{14.150}$$

where $D = D(u, v) = (u^2 + v^2)^{1/2}$. That of the Butterworth high-pass filter is

$$H(u, v) = \frac{1}{1 + [D_0/D(u, v)]^{2n}} \tag{14.151}$$

Note the difference between this expression and Eq. (14.147). Values of $H(u, v)$ increase with an increase in $D(u, v)$. $H(u, v) = 0$ when $D(u, v)$ is very small; $H(u, v) = 1$ when $D(u, v)$ is much greater than D_0; and $H(u, v) = 0.5$ when $D(u, v) = D_0$ and $n = 1$.

For exponential high-pass filters, the transfer function will be represented by

$$H(u, v) = e^{-[D_0/D(u,v)]^n} \tag{14.152}$$

$H(u, v)$ is zero when $D(u, v) = 0$, and the cutoff frequency D_0 is then defined at the point on the abscissa when $H(u, v) = e^{-1}$ or 0.368 of the maximum value of $H(u, v)$. $H(u, v)$ increases as D increases, and equals 1 when D approaches ∞ as the limit. That is, more high-frequency-component information will be included in the processing, but low-frequency-component information will be suppressed.

Analogous arguments can be applied to the high-pass trapezoidal filter. The transfer function of this filter can be derived similarly as follows:

$$H(u, v) = \begin{cases} 0 & \text{if} \quad D < D_1 \\ \dfrac{D - D_1}{D_0 - D_1} & \text{if} \quad D_0 \geq D \geq D_1 \\ 1 & \text{if} \quad D > D_0 \end{cases} \tag{14.153}$$

Comparison of these four transfer functions shows that the high-frequency-component emphasis increases in the order: Butterworth high-pass filter, exponential high-pass fitter, and trapezoidal high-pass filter, but the preservation of low-frequency information is in the reverse order for these filters. The proper choice of filter is largely problem dependent. Figures 14.60 and 14.61 show the results obtained by applying ideal high-pass filtering.

14.6.3 EnhancementThrough Clustering Specification

Although lots of approaches have been suggested, to date, no general procedure can be followed for image enhancement. Approaches available for image enhancement are very problem oriented. Nevertheless, a more-or-less generalized approach for image enhancement is still being sought. An approach by clustering specification inspired from bionics has been suggested by Bow and Toney (1983). The basics of this approach is quite intuitive. This follows from what we generally expect on a processed image:

1. *Object—distinctive:* It is expected that all the desirous objects should be included in the processed image and, in addition, those separate objects should be as distinct from one another as possible.
2. *Details—discernible:* Fine details of the desirous objects are expected to be discernible as well as possible.

That is, in viewing and in analyzing an image, the first thing to do is usually to separate the objects from the whole image and then focus our attention on the details of each of the objects. Following such bionic requirements, this algorithm consists of first applying the natural clustering method to identify the objects and then allocating appropriate dynamic ranges for each individual object in the order of their importance, so as to be able to fully utilize the gray levels to delineate the

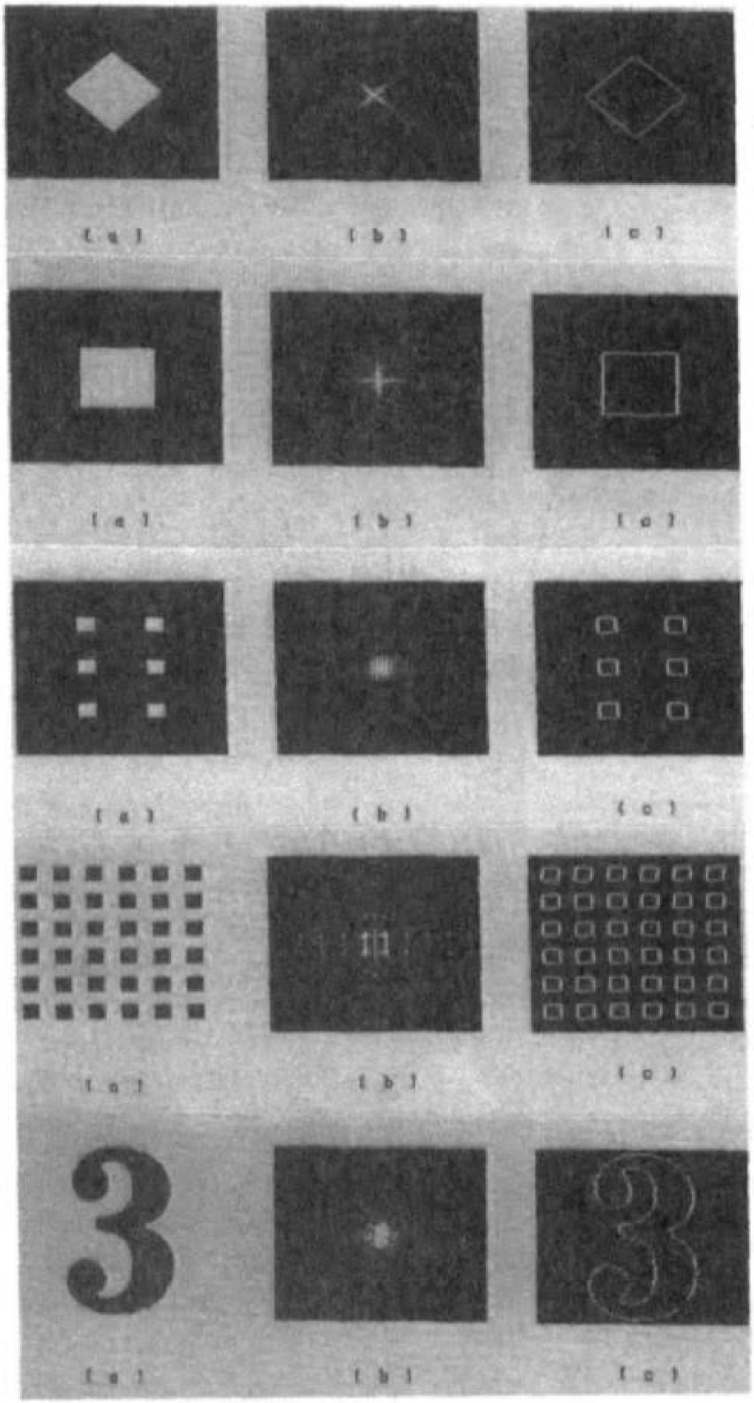


FIGURE 14.60 Ideal high-pass filtering process. (a) Original images; (b) spectra; (c) processed images.

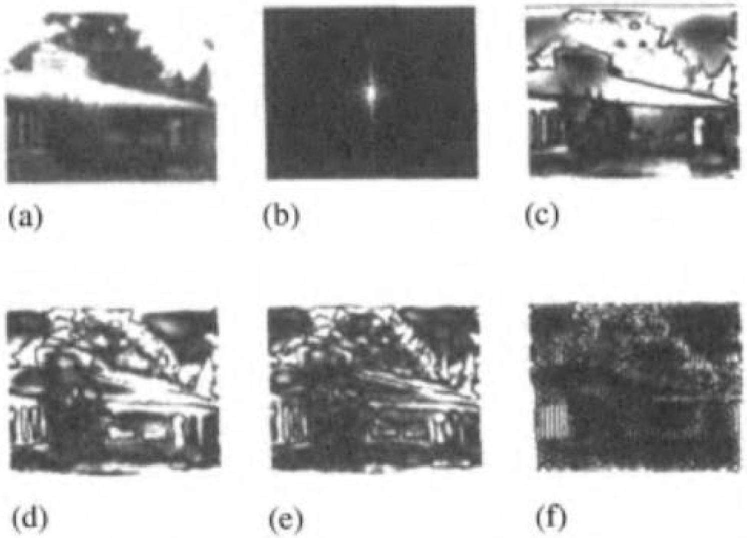


FIGURE 14.61 Examples of ideal high-pass filtering. (a) Original image; (b) spectrum; (c) when $D_0 = 0.02$; (d) when $D_0 = 0.05$; (e) when $D_0 = 0.07$; (f) when $D_0 = 0.20$.

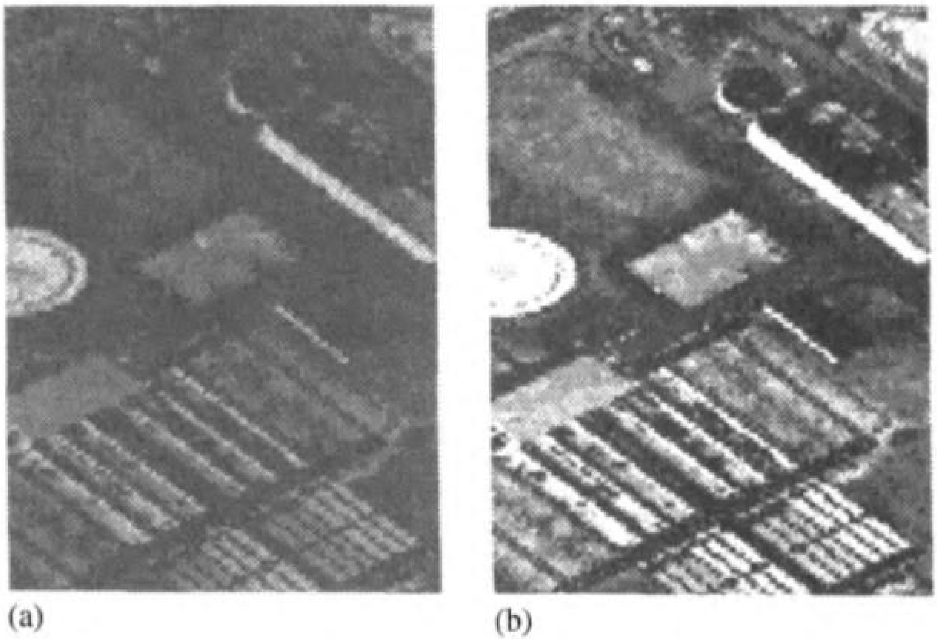


FIGURE 14.62 Image enhancement through clustering specification. (a) Original image; (b) image after processing.

details of the objects of interest, leaving the other objects, such as the background, suppressed in the picture. Any clustering approach can be used for clustering purposes. Two of these have been used for illustration: the extreme point clustering approach and the ISODATA algorithm.

Figure 14.62b shows the image after processing of the original image, shown in Figure 14.62a. Remember that the same image has been studied in Chapter 5. By using this method, a general routine procedure can be followed and the same result obtained with much less human intervention.

PROBLEMS

14.1 Given that the Fourier transform of $f(x, y)$ is $F(u, v)$, prove that the Fourier transform of $f(ax, by)$ is

$$\frac{1}{|ab|}F\left(\frac{u}{a}, \frac{v}{b}\right)$$

14.2 Show that the Fourier transform of a rectangular function with magnitude A (Figure P14.2) is a sinc function. Sketch the resulting Fourier spectrum.

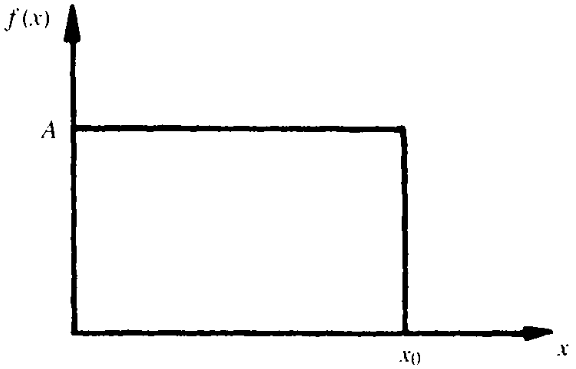


FIGURE P14.2

14.3 Consider the following 3×2 array $f(\zeta, \eta)$ and the 2×2 array $h(\zeta, \eta)$:

$$\eta\uparrow\begin{array}{|ccc} 2 & 1 & 4 \\ 1 & 3 & 5 \\ \hline \end{array}\rightarrow \zeta \qquad\qquad \eta\uparrow\begin{array}{|cc} 1 & 1 \\ 1 & -1 \\ \hline \end{array}\rightarrow \zeta$$

$$f(\zeta, \eta) \qquad\qquad\qquad h(\zeta, \eta)$$

Show the various steps for obtaining the convolution of these two arrays. If $f(\zeta, \eta)$ and $h(\zeta, \eta)$ are arrays of sizes $(M_1 \times N_1)$ and $(M_2 \times N_2)$, respectively, what will be the size of the resulting array?

14.4 Assume that x and y are continuous variables. Show that:

(a) The Fourier transform of the partial derivative with respect to x of an image function $f(x, y)$ is

$$\mathcal{F}\left[\frac{\partial f(x, y)}{\partial x}\right] = j2\pi uF(u, v)$$

and that with respect to y of $f(x, y)$ is

$$\mathcal{F}\left[\frac{\partial f(x, y)}{\partial y}\right] = j2\pi vF(u, v)$$

(b) The Fourier transform of the Laplacian of an image function, $f(x, y)$, is equal to

$$-(2\pi)^2(u^2 + v^2)F(u, v)$$

14.5 Indicate the principal difference between the convolution operation and correlation.

14.6 For an image function $f(x, y)$, $x, y = 0, 1, 2, \ldots, N-1$, prove that the average brightness of the image can be found as $F(0, 0)$, where $F(.,.)$ is the Fourier spectrum of the image $f(x, y)$.

14.7 When an image $f(x, y)$ is multiplied by $(-1)^{x+y}$ before transformation, the center of the frequency plane is moved to $(N/2, N/2)$. If the unitary DFT of $f(x, y)$ has its region of support as shown in Figure P14.7, what would be the region of support of the unitary DFT of $(-1)^{x+y}f(x, y)$?

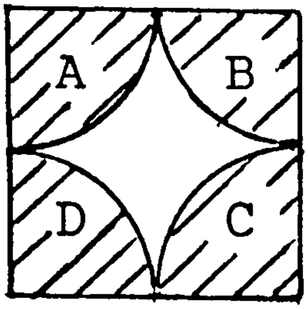


FIGURE P14.7

14.8 Discuss the effect of the size of the aperture on the Fourier spectrum of an image.

14.9 The two-dimensional Fourier transform of an image function

$$f(x, y) \qquad x, y = 0, 1, 2, \ldots, N-1$$

can be implemented by

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \exp\left[-j\frac{2\pi}{N}(ux + vy)\right]$$
$$u, v = 0, 1, 2, \ldots, N-1$$

Someone suggests an algorithm to speed up the transform process by partitioning the image function $f(x, y)$ into 16 smaller subarrays and performing two-dimensional FFT on the 16 subarrays. Would that be a good idea? If yes, explain why it works. If no, explain why not.

14.10 Reorder the inputs for the successive doubling method when $N = 32$, and draw the structure of the computation.

14.11 Note that in Fig. 14.45, the f's and F's are ordered differently in order to retain the butterfly computation format. Work out the flow graph of the FFT algorithm with the inputs in natural order.

14.12 Derive an equivalent algorithm for the FFT decimation-in-frequency decomposition of an eight-point DFT computation.

14.13 Write a program to generate, display, and print out some regular patterns for later processings (e.g., forward and inverse FFT).

14.14 Write a program for two-dimensional FFT for one of the images (Figures A.1 to A.15) in Appendix A, and also for the patterns generated in Problem 14.13.

(a) Obtain the Fourier spectrum with two-dimensional forward transform and restore the original image (or pattern) with the two-dimensional inverse transform.

(b) Rotate the pattern by an angle and transform it with two-dimensional FFT. Compare the spectrum obtained with the one obtained in part (a).

(c) Rotate the image chosen from any one of the images given in Appendix A by an angle and transform it with two-dimensional FFT. Compare the spectrum with the one obtained in part (a).

14.15 Use the program obtained from Problem 14.14.

(a) Translate the pattern as generated in Problem 14.13 by a distance, and transform it with two-dimensional FFT.
Compare the spectrum obtained with the one obtained in part (a).

(b) Translate the image chosen from any one of the images given in Appendix A by an angle, and transform it with two-dimensional FFT. Compare the spectrum with the one obtained in part (a).

14.16 Use the Fourier spectrum (a) obtained from Problem 14.14 and/or 14.15. Discuss the information content in the Fourier spectrum of a regular pattern and of an image by:

(a) Restoring the pattern/image with 90% of the spectrum data far away from center discarded.

(b) Restoring the pattern/image with 90% of the spectrum data far away from center discarded..

(c) Restoring the pattern/image with 80% of the spectrum data far away from center discarded.

(d) Restoring the pattern/image with 50% of the spectrum data far away from center discarded.

14.17 (a) Construct the unordered H matrix for $N = 16$ and mark the number of sign changes in each row.

(b) Rearrange the H matrix in part (a) so that the sequence is in increasing order.

14.18 Write a Program for the FHT.

14.19 Use an alternative method to prove Eq. (14.143). Start with

$$\mathbf{y} = \mathbf{B}(\mathbf{x} - \mathbf{m}_x)$$

and

$$\mathbf{B} = \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1N} \\ e_{21} & e_{22} & \cdots & e_{2N} \\ \vdots & & & \\ e_{N^21} & e_{N^22} & \cdots & e_{N^2N^2} \end{bmatrix}$$

remembering that $\mathbf{e}_i = [e_{i1}, e_{i2}, \ldots, e_{in}]$ is the ith eigenvector of $\mathbf{C}_x$ and e_{ij} is the jth component of the ith eigenvector. Prove that $\mathbf{C}_y$ is a diagonal matrix.

14.20 Write a program for each of the following filters:

(a) Ideal low-pass filter

(b) Butterworth low-pass filter

(c) Exponential low-pass filter

(d) Trapezoidal low-pass filter

Apply these filters to one of the images in Appendix A and the pattern that you generated. Discuss the results obtained when used for processing.

14.21 Write a program for each of the following filters:

(a) Ideal high-pass filter

(b) Butterworth high-pass filter

(c) Exponential high-pass filter

(d) Trapezoidal high-pass filter

Apply these filters to one of the images in Appendix A and the pattern that you generated. Discuss the results obtained when used for processing.

15

Wavelets and Wavelet Transform

15.1 INTRODUCTION

15.1.1 Why Wavelet for Image Processing

Before selecting an algorithm to process an image, first we must determine which type of information in the image we are most interested in, the local information or the global information. The approaches discussed in Chapter 12 are useful for the local information extraction by use of the neighborhood information of pixels in an image. When our interest is in global information (i.e., global image properties, both geometry and intensity based), those algorithms described in Chapter 14 are good choices. In these algorithms the discrete image data is represented in form of a matrix, and a separable linear transform, implemented as multiplication of the image function by a transformation matrix, is used to generate a set of basis functions for the representation of the entire image. Fourier transform, Walsh transform, Hadamard transform, etc., are examples of this type of transform. When approaches of the image transform are adopted, it is assumed that underlying images possess some characteristics that may be related to the transformed basis functions and that the whole image is treated as a single entity and cannot be processed by parts.

Image transform (e.g., two-dimensional Fourier transform) is, indeed, very powerful and effective for image analysis. It transforms the two-dimensional image signal from spatial domain to transform domain in which many characteristics of the image signal are revealed. However, due to the fact that the transformation kernel {e.g., $\exp[-j2\pi(ux + vy)]$ in the Fourier transform} is a global function, the double integration process, in the definition of the Fourier transform, cannot be carried out until the entire image (or a continuous sequence of images) in the whole of the real spatial axes $(-\infty, \infty)$ is known. This means that a small perturbation of the function at any point along the spatial axes influences every point on the frequency axes, and vice versa. If we imagine the image function $f(x, y)$ as the modulating function for $\exp[-j2\pi(ux + vy)]$, a perturbation at any point on the xy axes will propagate through the entire u, v axes. Put in other words, the Fourier spectrum does not provide any of the location information about the image signal. From the Fourier spectrum we have no way to tell where did the event occur. So the Fourier transform is good for a still image or a single-frame image, but not for nonstationary or transitory characteristics like trends. To overcome this deficiency, we need an approach which can perform both the location- and frequency-domain analyses on an image. By means of such an approach we can then extract the local frequency contents of this image. For this reason, we have the short-time Fourier transform (STFT).

Short time Fourier transform (STFT) is a time-frequency analysis for a one-dimensional signal, or a location-frequency analysis for a two-dimensional image. STFT can be briefly interpreted as the following. When we want to know the local frequency contents of a two-dimensional signal in the neighborhood of a desired location in a planar image, we can remove the desired portion from the given image by a window function, and then proceed the Fourier transform of that removed portion. Because of the windowing nature of the STFT, this transform is also referred to as the windowed Fourier transform. Advantage of this transform is that some information about "where" in the xy spatial domain and with "what" frequency content of the signal are provided. Nevertheless, this transform still has the shortcoming in that once a window is selected, the size of the window is fixed for all frequencies, and the location-frequency resolution is fixed throughout the processing. The question then arises as to the possibility of having a windowing technique with variable-sized regions. With such a technique, a window with larger region coverage (lower in frequency) can be chosen to acquire the lower frequency information, and a window with smaller region coverage (higher in frequency) to acquire the high-frequency components of the two-dimensional signal. If so, it will be much more suitable for the image processing. This objective leads to the development of the wavelet functions $\psi(x, y)$, and the wavelet transform, where the signal is decomposed into various scales of resolution, rather than frequencies. Multiresolution divides the frequencies into octave bands, from ω to 2ω, instead of uniform bands from ω to $\omega + \Delta\omega$. Figure 15.1a and b shows the time-frequency plot for the short-time Fourier transform and the time-scale plot for the wavelet analysis. From the figure it can be seen that short-time intervals are natural for high frequencies.

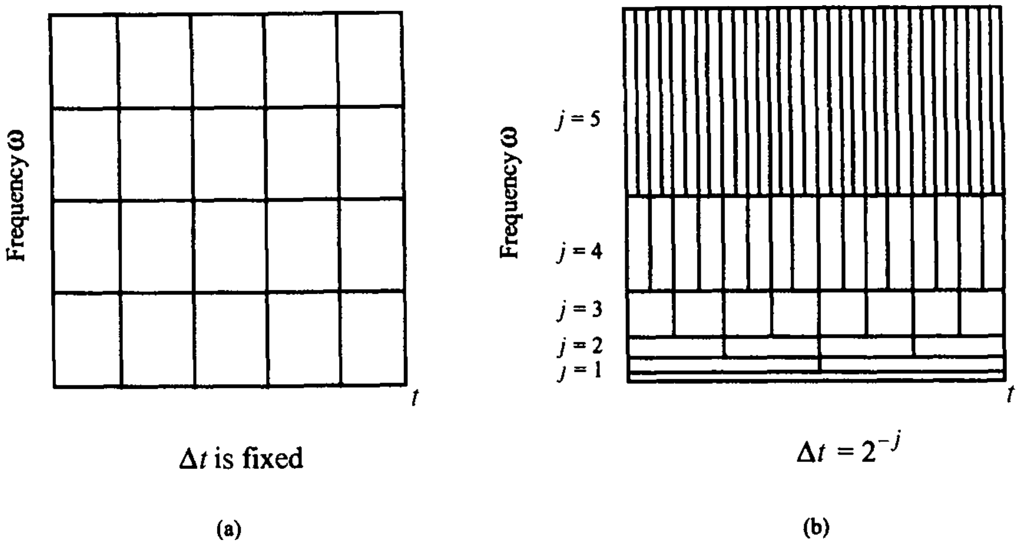


FIGURE 15.1 (a) Time-frequency plot for short time Fourier transform (STFT). (b) Time-scale plot for wavelet transform.

15.1.2 Why Wavelet Analysis Is Effective

The word *wavelet* (small wave) originates from the French name *ondelettes*. Similar to the sinusoid, it has the oscillating wavelike characteristic shown in Figure 15.2. However, it differs from the sinusoid in that it is a waveform of effectively limited duration that has an average value of zero. Because of this property, the wavelet expansion allows a more accurate local description and separation of signal characteristics. On the contrary, a Fourier coefficient represents a component that lasts for all time from $-\infty$ to $+\infty$. For this reason, a temporary event, for example, a delta function, would require an

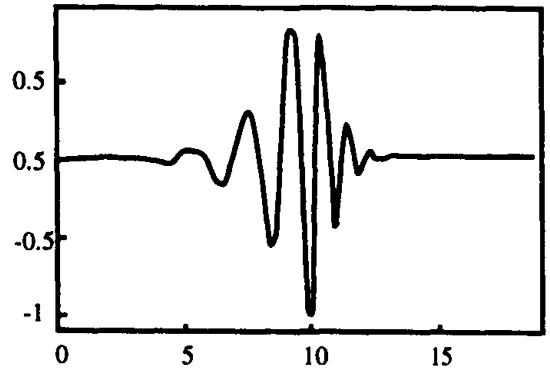


FIGURE 15.2 A wavelet in practical use.

infinite number of sinusoidal functions that combine constructively, while interfering with one another destructively to produce zero at all points except at $t \neq 0$. However, this is not the case when we use a wavelet, because a wavelet coefficient represents a component that is itself local. Hence, the wavelet expansion and wavelet transform are especially suitable for the processing of an image where most details could hardly be represented by functions, but could be matched by the various versions of the mother wavelet with various translations and dilations. Wavelet is really a good mathematical tool to extract the local features of variable sizes, variable frequencies, and at variable locations in an image.

As we will see later, the number of the wavelet coefficients drop off rapidly for image signals. This is another reason why the wavelet transform is so effective in image compression.

15.2 WAVELETS AND WAVELET TRANSFORM

Analogous to the Fourier transform we have continuous wavelet transform (CWT), wavelet series expansion, and discrete wavelet transform (DWT). However, we should recall that there is a main difference between Fourier transform and wavelet transform. The orthogonal basis functions used in Fourier transform are $\sin(k\omega_0 t)$ and $\cos(k\omega_0 t)$. They extend from minus infinity to positive infinity, and are also nonzero over their entire domain. Hence, the Fourier transform does not have compact support. Neither does the Short time Fourier transform, even though it could provide the time-frequency information. This is because in STFT, as mentioned in the previous section, once the window is chosen, the window size will be fixed for all frequencies.

However, the wavelet transform has compact support, since in the wavelet transform the basis function used is

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \tag{15.1}$$

which satisfies two conditions. One is the admission condition

$$\int_{-\infty}^{\infty} \frac{|\psi(\omega)|^2}{\omega} d\omega < \infty \tag{15.2}$$

and the other is

$$\int_{-\infty}^{\infty} \psi(t)\, dt = 0 \tag{15.3}$$

It is obvious from the above conditions that by reducing the scale parameter, the support of $\psi_{j,k}$ is reduced in time and frequency.

The continuous wavelet transform of a function $f(t) \in L^2$ with respect to some analyzing wavelet ψ_{jk} is defined as

$$W_\psi f(t) = \langle f, \psi_{j,k} \rangle = \int_{-\infty}^{\infty} f(t)\psi_{j,k}(t)\,dt \tag{15.4}$$

and

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \tag{15.5}$$

Both j and k in the above expression are real numbers. The variable j reflects the scale (width of a particular basis function), while k specifies its translated position along the t axis. The wavelet transform coefficients are calculated as the inner products of the function being transformed with each of the basis functions. The inverse transform is in the form of

$$f(t) = \sum_j \sum_k c_{j,k}\psi_{j,k}(t) \tag{15.6}$$

where

$$c_{j,k} = \langle f(t), \psi_{j,k} \rangle = \int_{-\infty}^{\infty} f(t)\psi_{j,k}(t)\,dt \tag{15.7}$$

This set of expansion coefficients $c_{j,k}$ are called the discrete wavelet transform (DWT) of $f(t)$, and the above expression (15.6) is the inverse discrete wavelet transform (IDWT). They form a transformation pair.

For a function $f(x, y)$, the wavelet functions and the two-dimensional continuous wavelet transform are, respectively,

$$\psi_{j,kx,ky}(x, y) = 2^j\psi(2^j x - k_x, 2^j y - k_y) \tag{15.8}$$

$$W_\psi f(x, y) = \langle f(x, y), \psi_{j,kx,ky} \rangle = \int\int_{-\infty}^{\infty} f(x, y)\psi_{j,kx,ky}(x, y)\,dx\,dy \tag{15.9}$$

where k_x and k_y are, respectively, translations in x and y coordinate axes. The inverse two-dimensional CWT is

$$f(x, y) = \sum_j \sum_{k_x} \sum_{k_y} c_{j,kx,ky}\psi_{j,kx,ky}(x, y) \tag{15.10}$$

where

$$c_{j,kx,ky} = \langle f(x, y)\psi_{j,kx,ky}(x, y) \rangle = \int\int_{-\infty}^{\infty} f(x, y)\psi_{j,kx,ky}(x, y)\,dx\,dy \tag{15.11}$$

15.3 SCALING FUNCTION AND WAVELET

15.3.1 Scaling Function and theVector Space Spanned by the Scaling Function

The idea of multiresolution is frequently used in the discussion of a wavelet system. There are two functions needed to be defined: the scaling function and the wavelet. Let us first define the scaling function and then define the wavelet in terms of it. Say we have a basic scaling function:

$$\varphi(t) \in L^2 \tag{15.12}$$

where L^2 is the space of all functions $f(t)$ with a well-defined integral of the squares of modulus of the function:

$$\int_{-\infty}^{\infty} |f(t)|^2 \, dx < \infty \tag{15.13}$$

and a set of scaling functions $\varphi_k(t)$:

$$\varphi_k(t) = \varphi(t-k) \qquad k \in \mathbf{Z} \tag{15.14}$$

which is generated by the basic function $\varphi(t)$ with k translates. $\mathbf{Z}$ is the set of all integers from $-\infty$ to ∞. Thus, we can define the vector space of signals S as the functions, all of which can be expressed by

$$f(t) = \sum_k c_k \varphi_k(t) \tag{15.15}$$

Let us use v_0 to represent the space spanned by these functions, i.e.,

$$v_0 = \operatorname*{span}_k \, (\varphi_k(t)) \qquad \text{for all integers from } -\infty \text{ to } \infty$$

v_0 is then the space spanned by the $\varphi_{jk} = 2^{j/2}\varphi(2^j t - k)$, which is generated by changing the time scale of the scaling functions from $\varphi(t)$ to $\varphi(2^j t)$, $j = 1, 2, \ldots$ to increase the size of the subspace spanned, and also by translates for all values of k. The above expression implies that if $f(t) \in v_j$, $f(t)$ can then be expressed

$$f(t) = \sum_k c_k \varphi(2^j t - k) \qquad \text{for } f(t) \in v_j \tag{15.16}$$

For $j < 0$, the scaling function $\varphi_{jk}(t)$ is wider and is translated in larger steps. These wider scaling functions can represent coarse information. When $j > 0$, the scaling function $\varphi_{jk}(t)$ becomes narrower and is translated in smaller steps. This narrower scaling function can represent finer details. Note that $f(t) = \sum_k c_k \varphi(2^j t - k)$ is now in v_j, i.e., in the signal space spanning over $\phi_k(2^j t)$. The spaces have the scaling property

$$f(t) \in v_j \iff f(2t) \in v_{j+1} \tag{15.17}$$

This implies that the space that contains signals of high resolution will also contain those of lower resolution. Thus,

$$v_{-\infty} \subset \cdots \subset v_{-1} \subset v_0 \subset v_1 \subset v_2 \subset \cdots \subset L^2 \tag{15.18}$$

or

$$v_j \subset v_{j+1} \qquad \text{for all } j \in \mathbf{Z} \tag{15.19}$$

with $v_{-\infty} = \{0\}$ and $v_{\infty} = L^2$. This means that if $\varphi(t)$ is in v_0, it is also in v_1. It also means that $\varphi(t)$ can be expressed in terms of a weighted sum of shifted $\phi(2t)$ as

$$\varphi(t) = \sum_n h(n)\sqrt{2}\varphi(2t - n) \qquad n \in \mathbf{Z} \tag{15.20}$$

where $h(n)$ in the above recursive equation are the scaling function coefficients, a sequence of real or perhaps complex numbers, and the constant $\sqrt{2}$ is added here to maintain the norm of the scaling function with the scale of 2. For the Haar scaling function, $h(0) = 1/\sqrt{2}$ and $h(1) = 1/\sqrt{2}$, we then have

$$\varphi(t) = \varphi(2t) + \varphi(2t - 1) \tag{15.21}$$

15.3.2 Wavelet and the Vector Space Spanned by the Wavelet Function

As we discussed in the previous paragraph, the space v_0 spanned by $\varphi(t - k)$ is included in the space v_1 which is spanned by $\varphi(2t - k)$. Increasing the scale will allow greater and greater details to be realized, and so the higher resolution space v_j, which is the space spanned by $\varphi(2^j t - k)$, will better approximate the functions. Nevertheless, if we introduce another set of functions that span the differences between the spaces spanned by scaling functions of various scale rather than using the scaling function $\varphi_{jk}(t)$ alone, the signal will be much more better described or represented. These functions are called wavelets $\psi_{jk}(t)$. These two sets of functions are orthogonal. The orthogonality property among these two sets of functions provides advantages in making it possible to simplify the coefficient computation.

Let us define the orthogonal complement of v_j in v_{j+1} as W_j. This means that all members of v_j are orthogonal to all members of W_j, or

$$v_j = v_{j-1} \oplus W_{j-1} \qquad \text{for all } j \in \mathbf{Z} \tag{15.22}$$

and

$$v_{j-1} \perp W_{j-1} \tag{15.23}$$

where $\oplus$ and $\perp$ denote, respectively, superposition and orthogonality. Thus we have

$$\begin{aligned} v_{j-1} &= v_{j-2} \oplus W_{j-2} \\ &\vdots \\ v_2 & \quad v_1 \oplus W_1 \\ v_1 &= v_0 \oplus W_0 \end{aligned} \tag{15.24}$$

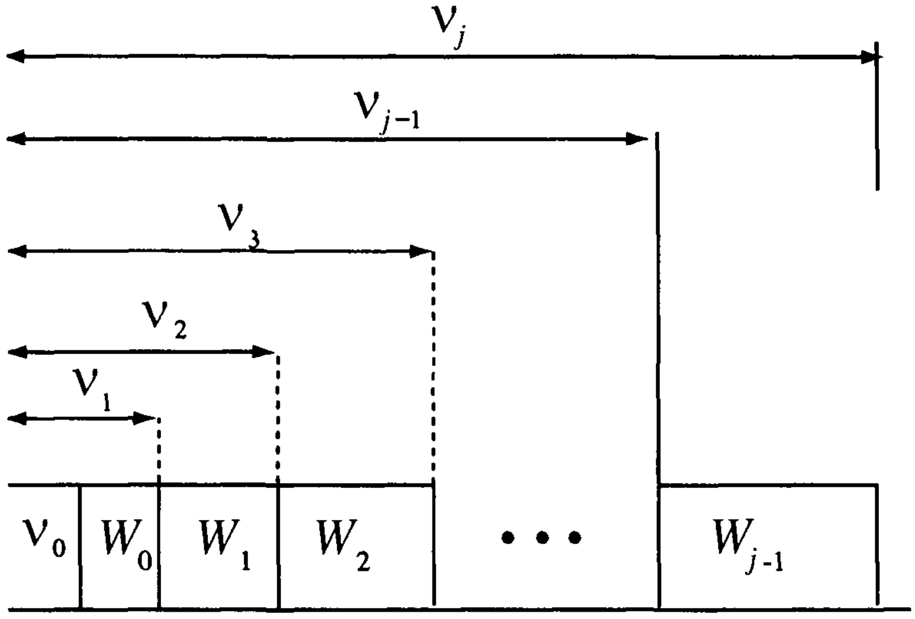
and

$$v_j = v_0 \oplus W_0 \oplus W_1 \oplus W_2 \oplus \cdots \oplus W_{j-1} \tag{15.25}$$

where v_0 is the initial space spanned by the scaling function $\varphi(t-k)$, and W_β, $\beta = 0, 1, 2, \ldots, j-1$, are the orthogonal components (or disjoint differences). This is shown in Figure 15.3.

By the same reason discussed in Eq. (15.20), when $W_0 \subset v_1$, these wavelets can be represented in terms of a weighted sum of the shifted scaling function $\varphi(2t)$ as

$$\psi(t) = \sum_n h_1(n)\sqrt{2}\varphi(2t-n) \qquad n \in \mathbf{Z} \tag{15.26}$$



$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset L_2$$

FIGURE 15.3 Scaling function and wavelet vector spaces.

for some set of coefficients $h_1(n)$. For the Haar scaling function, $h_1(0) = 1/\sqrt{2}$ and $h_1(1) = -1/\sqrt{2}$. We then have

$$\psi(t) = \varphi(2t) - \varphi(2t - 1) \tag{15.27}$$

With both the scaling function and wavelets we can then represent a large class of signals as

$$f(t) = \sum_k c_{j_0,k}\varphi_{j_0,k}(t) + \sum_{j=j_0}^{\infty} \sum_k d_{jk}\psi_{j,k}(t) \tag{15.28}$$

or

$$f(t) = \sum_k c_{j_0,k}2^{j_0/2}\varphi(2^{j_0}t - k) + \sum_k \sum_{j=j_0}^{\infty} d_{jk}2^{j/2}\psi(2^j t - k) \tag{15.29}$$

where $c_{j_0,k}$ and d_{jk} are the coefficients which can be computed by inner products as

$$c_{j_0,k} = \langle f(t), \varphi_{j_0,k}(t)\rangle = \int f(t)\varphi_{j_0,k}(t)\, dt \tag{15.30}$$

and

$$d_{jk} = \langle f(t), \psi_{jk}(t)\rangle = \int f(t)\psi_{jk}(t)\, dt \tag{15.31}$$

It should be repeated here that there is an orthogonality requirement for the scaling functions and wavelets. With this requirement it would make the coefficient computation simple. Secondary, it is possible for the scaling function and wavelets to have compact support, because the wavelet coefficients drop out rapidly as j and k increase. The signal can therefore be efficiently and effectively represented by a small number of the coefficients. This is why the DWT is efficient for signal and image compression. To give a clear picture of the scaling functions and wavelets as well as their relationships, the Haar function, which is an odd rectangular pulse pair, might be the best one for explanation (see Figures 15.4 and 15.5).

Note that the complete sets of wavelets at any scale completely cover the interval. Recalling that

$$\psi_n = 2^{j/2}\psi(2^j x - k) \tag{15.32}$$

as the wavelet is scaled down by a power of 2, its amplitude is scaled up by a power of $2^{1/2}$ to maintain orthonomality.

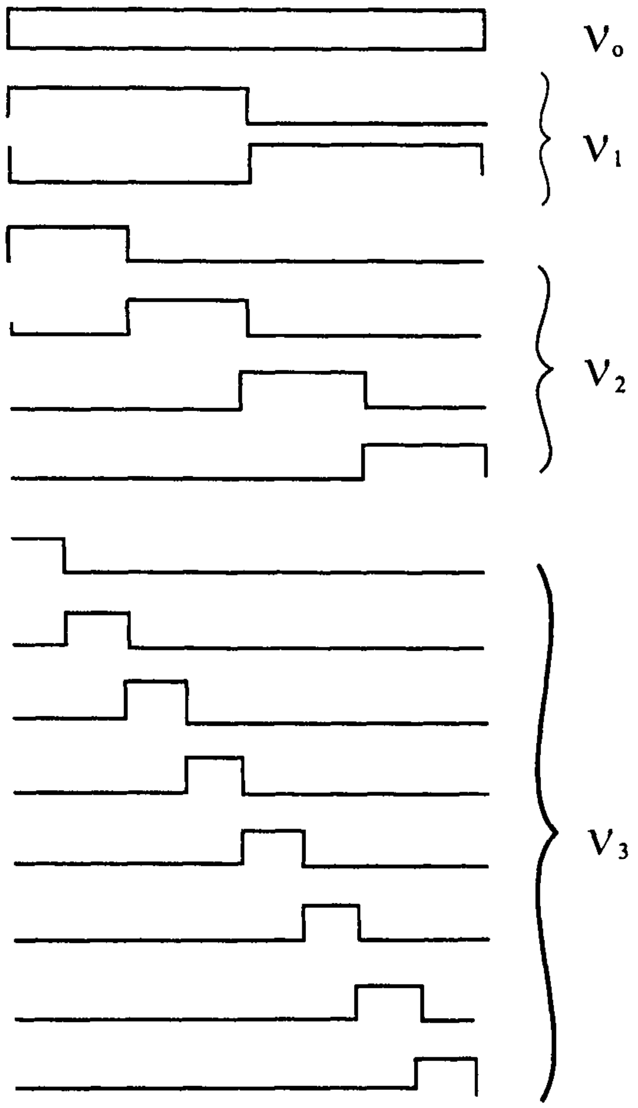


FIGURE 15.4 Haar scaling functions that span v_j.

15.4 FILTERS AND FILTER BANKS

Filter is a linear time-invariant operator $\mathbf{h}(n)$. It performs the convolution process. If vectors $\mathbf{x}(n)$ and $\mathbf{y}(n)$ denote, respectively, the input and output vectors, they can then be related as

$$\mathbf{y}(n) = \sum_k \mathbf{h}(k)\mathbf{x}(n-k) \qquad (15.33)$$

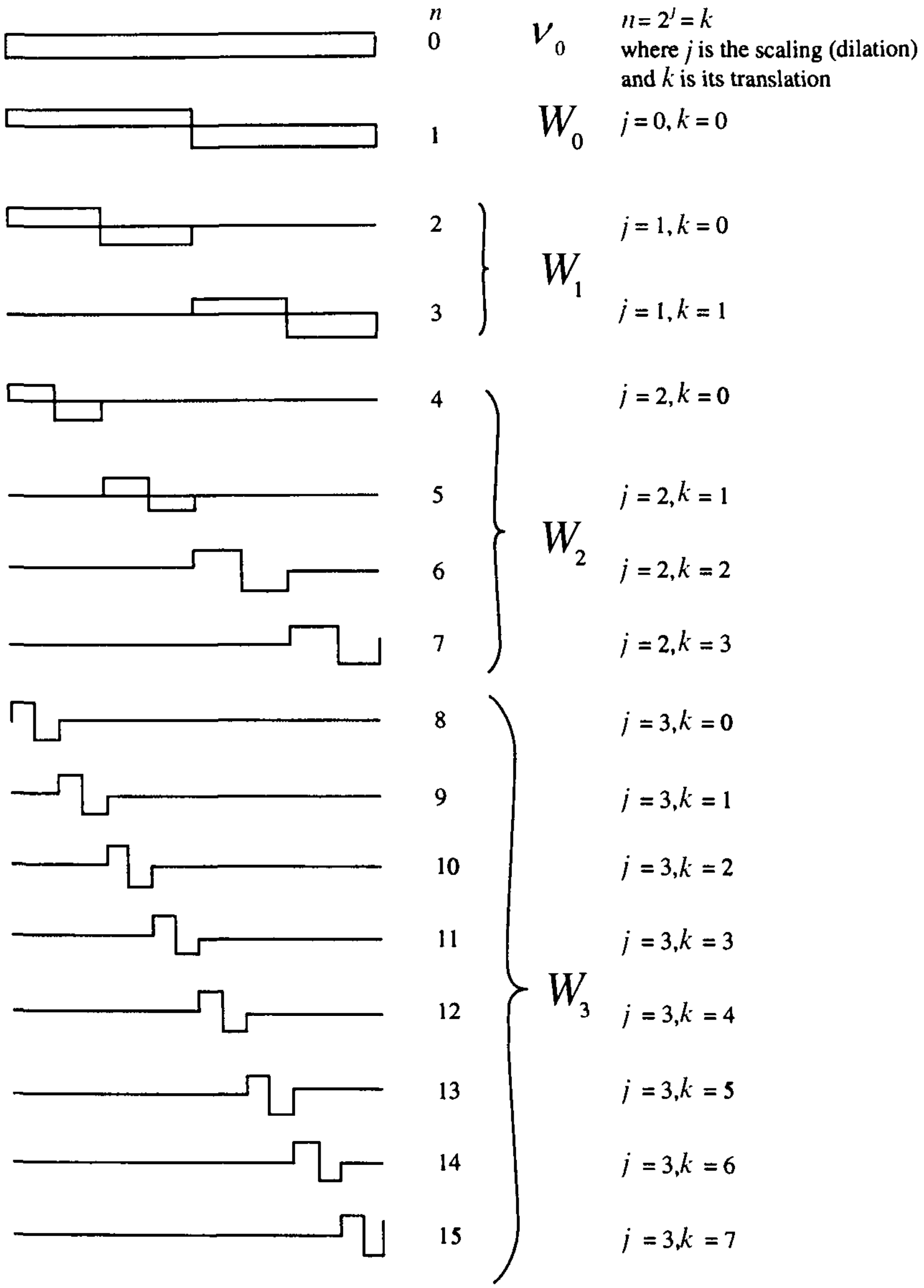


FIGURE 15.5 Haar scaling function and wavelets decomposition.

or

$$\mathbf{y} = \mathbf{h} * \mathbf{x} \tag{15.34}$$

where the symbol $*$ represents convolution. When the vector $\mathbf{x}(n-k)$ is a unit impulse at $n = k$ [i.e., $\mathbf{x}(n-k) = 0$, except when $n = k$], we then have

$$\mathbf{y}(n) = \mathbf{h}(n) \qquad \text{for } n = 0, 1, 2, \ldots \tag{15.35}$$

and $\mathbf{y}(n)$ is the impulse response.

A signal usually consists of low-frequency and high-frequency contents. The low-frequency content is usually the most important part of the signal, as it gives the signal its identity. The high-frequency content imparts flavor or nuance. There are two technical terms conventionally used in the wavelet analysis, namely, *approximations A* and *details D*. *Approximations* refers to the high-scale-factor, low-frequency components of the signal, which can be matched with the stretched wavelets, while *details* refers to the low-scale-factor, high-frequency components which are to be matched by the compressed wavelets. These two component parts of the signal can be separately extracted through a filter bank. A filter bank is a set of filters used to separate an input signal into frequency bands for analysis. For our case two filters are usually chosen for the bank, the high-pass and the low-pass filter. The high-scale-factor, low-frequency components of the signal can pass through the low-pass filter, while the high-frequency components of the signal (i.e., the low-scale-factor components) are singled out at the output of the high-pass filter.

15.4.1 Decimation (or Downsampling)

As we discussed in the last paragraph, the high-frequency and low-frequency components of a signal can be separated by a filter bank, which consists of a high-pass and low-pass filter. Output of the low-pass filter will retain the high-scale-factor, low-frequency components, while that of the high-pass filter retains the low-scale-factor, high-frequency components. At the outputs of the filter bank, the total number of samples will obviously be double. To keep the number of samples the same as before, a *decimation* (or called *downsampling*) by 2 will be needed. Simply select only the even samples to perform this process. The input $\mathbf{x}(n)$ and output $\mathbf{y}(n)$ of the *downsampler* is related by

$$\mathbf{y}(n) = \mathbf{x}(2n) \qquad \text{for } n \in \mathbf{Z} \tag{15.36}$$

or

$$\mathbf{y}(n) = \sum_k \mathbf{x}(n)\delta(n-2k) \qquad k \in \mathbf{Z} \tag{15.37}$$

where $\delta(n-2k)$ is a sequence of unit impulses. If

$$\mathbf{x}(n) = \ldots, x(-4), x(-3), x(-2), x(-1), x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8), \ldots$$

then

$$\mathbf{y}(n) = \ldots y(-3), y(-2), y(-1), y(0), y(1), y(2), y(3), y(4), y(5), y(6), y(7), y(8), \ldots$$

or

$$\begin{vmatrix} \vdots \\ y(-2) \\ y(-1) \\ y(0) \\ y(1) \\ y(2) \\ y(3) \\ y(4) \\ \vdots \end{vmatrix} = \begin{vmatrix} \vdots & & & & & & & & & & & & \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \vdots & & & & & & & & & & & & \end{vmatrix} \begin{vmatrix} \vdots \\ x(-4) \\ x-3) \\ x(-2) \\ x(-1) \\ x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \\ x(8) \\ \vdots \end{vmatrix}$$

or

$$[\mathbf{y}] = [\text{Downsampling} \downarrow 2][\mathbf{x}] \tag{15.38}$$

Figure 15.6 shows a stage of filter bank including downsampling by 2, where S denotes the original signal $\mathbf{x}(n)$, $n = -\infty, \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots, \infty$. cA, the approximation, is the low-frequency component part, while cD, the

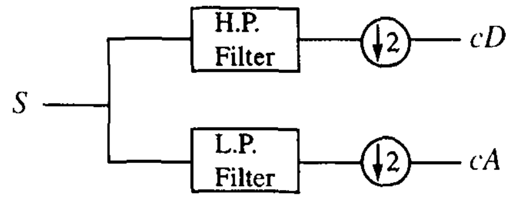


FIGURE 15.6 One stage of the filter bank downsampling by 2.

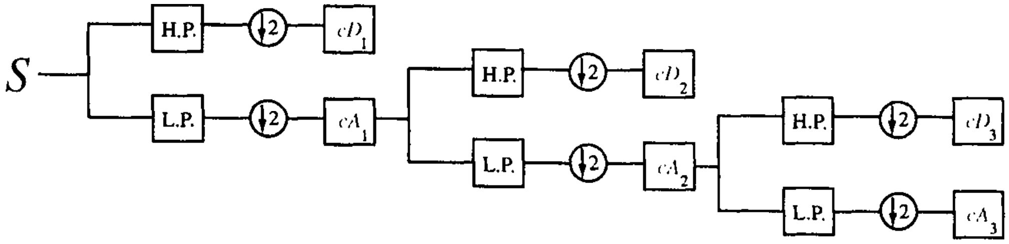


FIGURE 15.7 Decomposition of a signal into lower resolution components.

details, the high-frequency part of the original signal both at half-resolution. This decomposition process can be iterated with successive approximation being decomposed in turn. The algorithm gives as outputs cD_1, cD_2, cD_3, etc, which compose the wavelet coefficient set. The signal S is thus broken down into many lower-resolution components, as shown in Figure 15.7. cD_2 in Figure 15.7 is at a half-resolution of cD_1; cD_3 is at a half-resolution of cD_2; etc. They are the wavelet coefficients and give the "details" information of the signal. The decomposition can proceed until the individual details information consists of a single sample or a single pixel in our image processing application. However, a suitable number of levels should be chosen based on the individual problem. To meet our image processing need, three levels of decomposition is usually appropriate.

Three aspects are involved in the wavelet analysis and synthesis: (a) Break up a signal, either one-dimensional or two-dimensional, to represent it as a set of wavelet coefficients—this is what we conventionally call discrete wavelet transform (DWT); (b) modify the wavelet coefficients—processing in wavelet transform domain. For the purpose of denoising, we can look for those undesirable components which are similar to the noise and remove them. Similarly, for the purpose of data compression, we can ignore those transform coefficients that are insignificant. (c) Reconstruct the signal from the coefficients after their modification. This process is conventionally referred to as inverse discrete wavelet transform (IDWT).

15.4.2 Interpolation (or Upsampling)

When we reconstruct the signal from the wavelet coefficients, upsampling (or interpolation) followed by convolution is involved. In simple words, upsampling is a process to lengthen the components of a signal by inserting zero's between samples. Mathematically, the results yielded after the upsampling are

$$x'(n) = \begin{cases} y(n/2) & \text{for } n = -\infty, \ldots, -4, -2, 0, 2, 4, 6, \ldots, \infty \text{ (even } n) \\ 0 & \text{for } n = -\infty, \ldots, -5, -3, -1, 1, 3, 5, \ldots, \infty \text{ (odd } n) \end{cases}$$

$x'(n)$ and $y(n)$ are, respectively, the output and input of the processing segment. Put in matrix form, we have

$$\begin{bmatrix} x'(-2) \\ x'(-1) \\ x'(0) \\ x'(1) \\ x'(2) \\ x'(3) \\ x'(4) \\ x'(5) \\ x'(6) \\ x'(7) \\ x'(8) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y(-1) \\ 0 \\ y(0) \\ 0 \\ y(1) \\ 0 \\ y(2) \\ 0 \\ y(3) \\ 0 \\ y(4) \end{bmatrix}$$

or

$$[\mathbf{x}'] = [\text{Upsampling} \uparrow 2][\mathbf{y}] \tag{15.39}$$

Figure 15.8 shows the signal reconstruction procedure.

Figures 15.6 and 15.7 show the operation of convolution followed by downsampling performed on the signal S, and Figure 15.8 shows the reverse operation sequence of upsampling followed by convolution in reconstructing the signal. These two sets of operations are the most important building blocks in algorithms for both the discrete wavelet transform and inverse discrete wavelet transform.

When no modifications are made on the wavelet coefficients, the original function should be recovered perfectly from the components at different scales to make the wavelet transform meet the reversibility requirement. We should note that we cannot choose any shape and call it wavelet for the analysis. We are compelled to choose a shape determined by the quadrature mirror decomposition

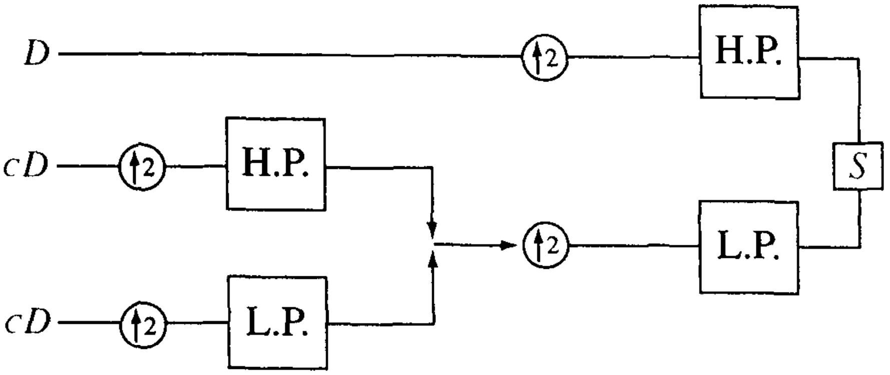
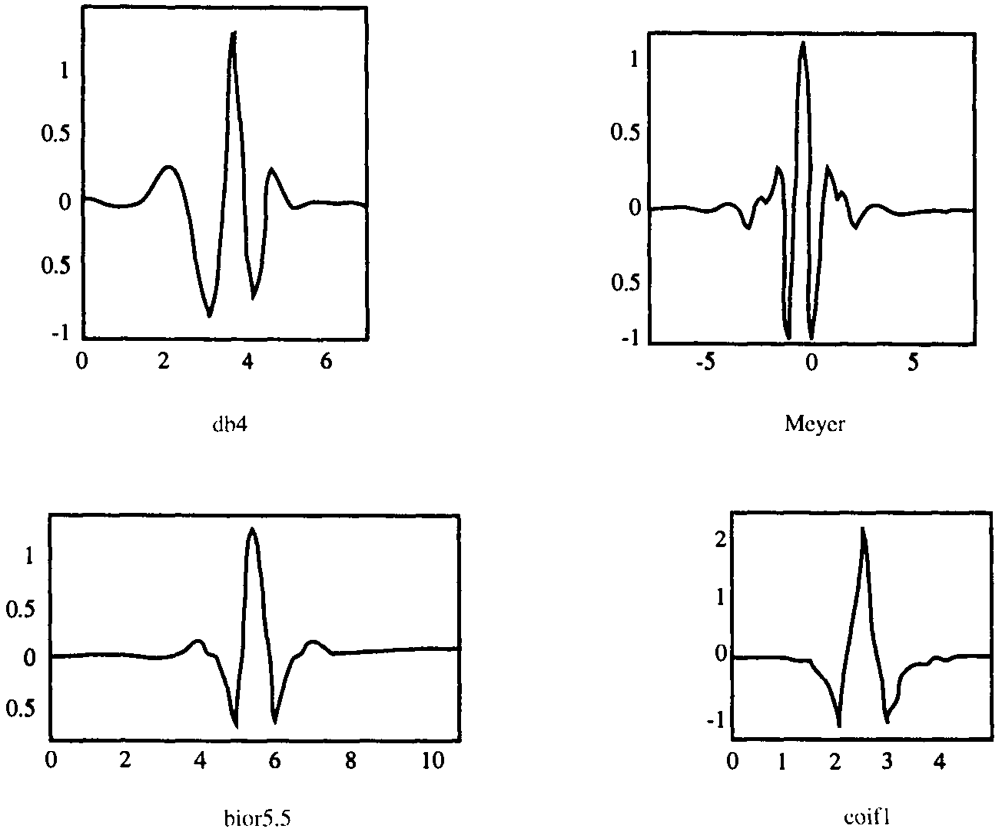


FIGURE 15.8 Signal reconstruction.



db4

Meyer

bior5.5

coif1

FIGURE 15.9 Examples of the wavelets proposed.

filters. It is therefore more practical for us to design the appropriate quadrature mirror filters first and then create the waveform. Many forms of wavelet have been proposed. To name a few for illustration, some of them are shown in Figure 15.9.

15.5 DIGITAL IMPLEMENTATION OF DWT

15.5.1 One-Dimensional Discrete Wavelet Transform (DWT)

The definition of the continuous wavelet transform as discussed in Section 15.2 is reproduced here for convenience:

$$W_\psi f(t) = \langle f, \psi_{j,k} \rangle = \int_{-\infty}^{\infty} f(x)\psi_{j,k}(t)\, dt \qquad (15.40)$$

and

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \tag{15.41}$$

The process of wavelet transform is actually a process to compute the set of coefficients. Each of these coefficients is the inner product of input function $f(t)$ and one version of the basic ψ function. Let us put the above expression in the following form:

$$c\text{ (scale, position)} = \int_{-\infty}^{\infty} f(t)\psi(\text{scale, position})\, dt$$

where c is the wavelet coefficients, $f(t)$ is the signal and ψ(scale, position) represents the various versions of the basic wavelet ψ scaled at j and translated by a distance k. This means that the wavelet coefficient c(scale, position) represents the degree of similarity between the input function $f(t)$ and that particular version of the basic function. The set of expansion coefficients, $c_{j,k}$ with scale at 2^j, and translated with k, $j, k = 0, 1, 2, \ldots$ can be used as amplitude weighted factor on the basis functions to represent the function $f(t)$, or

$$f(t) = \sum_j \sum_k c_{j,k}\psi_{j,k}(t) \tag{15.42}$$

Since the basis functions are carefully selected and are orthogonal (or orthonomal) among one another, the inner product taken between any two basis functions is zero. This indicates that these two functions are completely dissimilar. A signal is made up of constituent components, which are, respectively, similar to some but not all of the various basis functions. These components will manifest themselves in large coefficients for those basis functions which they are similar to, but small, even zero, for the rest of the basis functions. Hence, except for a few, most coefficients will be small. For this reason the signal can then be represented compactly by only a small number of transform coefficients. This is the heart of the discrete wavelet transform.

Processing in the wavelet transform domain works in such a way that when an undesirable component (say, noise) is similar to one or a few of these basis functions, then it will be easy for us to look for them. The denoising process can then be performed by simply reducing or even setting to zero the corresponding transform coefficients, and then reconstructing the signal according to the expression (15.42) with the new values of the transform coefficients.

Figure 15.10 shows the multiresolution pyramid decomposition to generate the wavelet coefficients. $c_{j,k}$ in this figure represents the coefficients of signal component in the original signal, which is assumed in the space v_j, and is

$$c_{j,k} = \langle f, \phi_{j,k}\rangle = \int f(t)\varphi(2^j t - k)\, dt \tag{15.43}$$

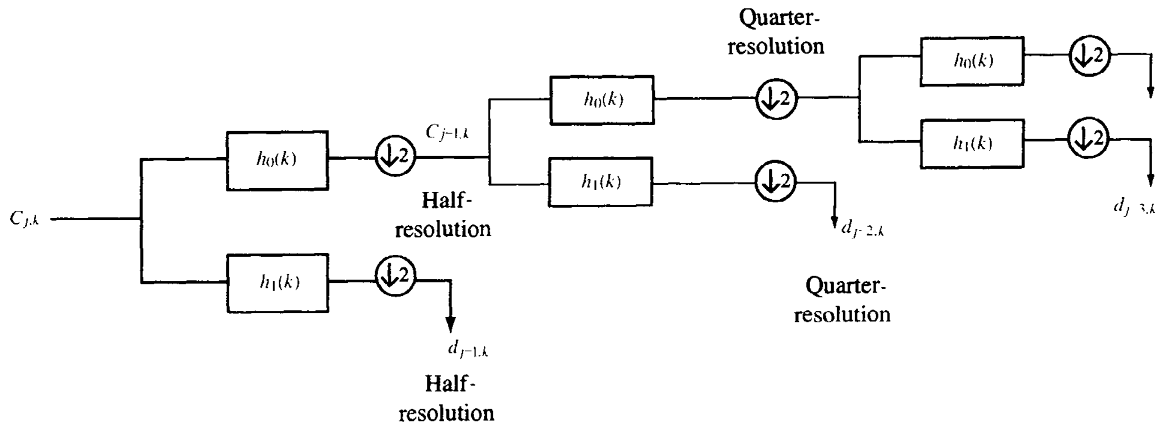


FIGURE 15.10 Multiresolution pyramid decomposition.

and the function $f(t)$ can be expressed as

$$f(t) = \sum_k c_{j-1,k}\varphi_{j-1,k}(t) + \sum_k d_{j-1,k}\psi_{j-1,k}(t) \tag{15.44}$$

where $c_{j-1,k}$ are the coefficients of half-resolution low-frequency signal components. $d_{j-1,k}$ are the coefficients of half-resolution high-frequency signal components, and represents the "detail" or difference between the original signal $c_{j,k}$ and its downsampled approximation signal $c_{j-1,k}$. $d_{j-2,k}$ is the quarter-resolution high-frequency component, and $d_{j-3,k}$ is the $\frac{1}{8}$-resolution high-frequency component, etc. If $N = 2^m$, after m iterations a signal with N samples will become a single data point.

Figure 15.11 shows the discrete approximations, respectively, after the first, second, and third decomposition of a speech signal. Figure 15.12 shows the discrete wavelet representations $d_{j-1,n}$, $d_{j-2,n}$, and $d_{j-3,n}$ of the same speech signal. Figure 15.13 gives the comparison of the reconstructed signal and original signal. They are the same.

Inverse discrete wavelet transform can be implemented in the reverse order, which is shown in Figure 15.14, and self-explanatory.

15.5.2 Two-Dimensional Discrete Wavelet Transform

All we have discussed so far is the one-dimensional discrete wavelet transform. The concept developed to represent a one-dimensional signal with wavelets and approximation function can be extended to represent a two-dimensional signal.

Let us go back to the expression of the two-dimensional continuous wavelet transform.

$$W_\psi f(x, y) = \int\int_{-\infty}^{\infty} f(x, y)\psi_{j,kx,ky}(x, y)\, dx\, dy \tag{15.45}$$

where k_x, k_y represent, respectively, the translations along the two axes x and y. $\psi_{j,kx,ky}(x, y)$, $k_x, k_y = 0, 1, 2, \ldots$ represent, respectively, the various versions of the basic wavelet ψ scaled at 2^j, $j = 0, 1, 2, \ldots$ and translated by distances k_x and k_y. Each of these filter versions is a two-dimensional impulse response, and would, respectively, respond only (or primarily) to the objects of different sizes on the particular location on the image. In other words, if the image is band limited to an interval over which at least one $\psi_j(u, v)$ is nonzero, then $f(x, y)$ could be recovered from that filter output alone.

To represent a two-dimensional signal (or an image) we use two-dimensional wavelets and a two-dimensional scaling function. The two-dimensional

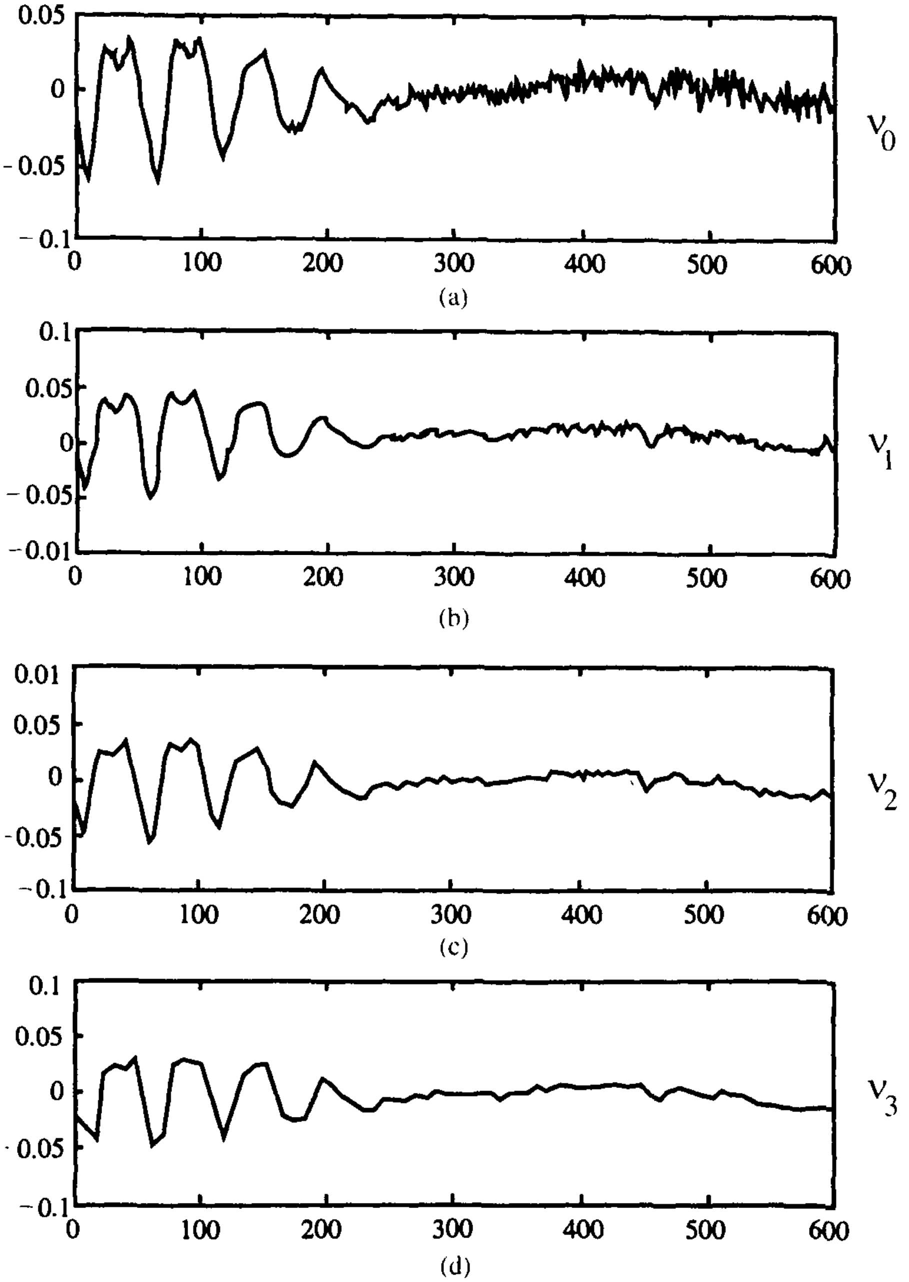


FIGURE 15.11 Discrete approximations after the first, second, and third decomposition of a speech signal. (a) The original speech signal; (b) approximation component after the first decomposition; (c) approximation component after the second decomposition; (d) approximation component after the third decomposition.

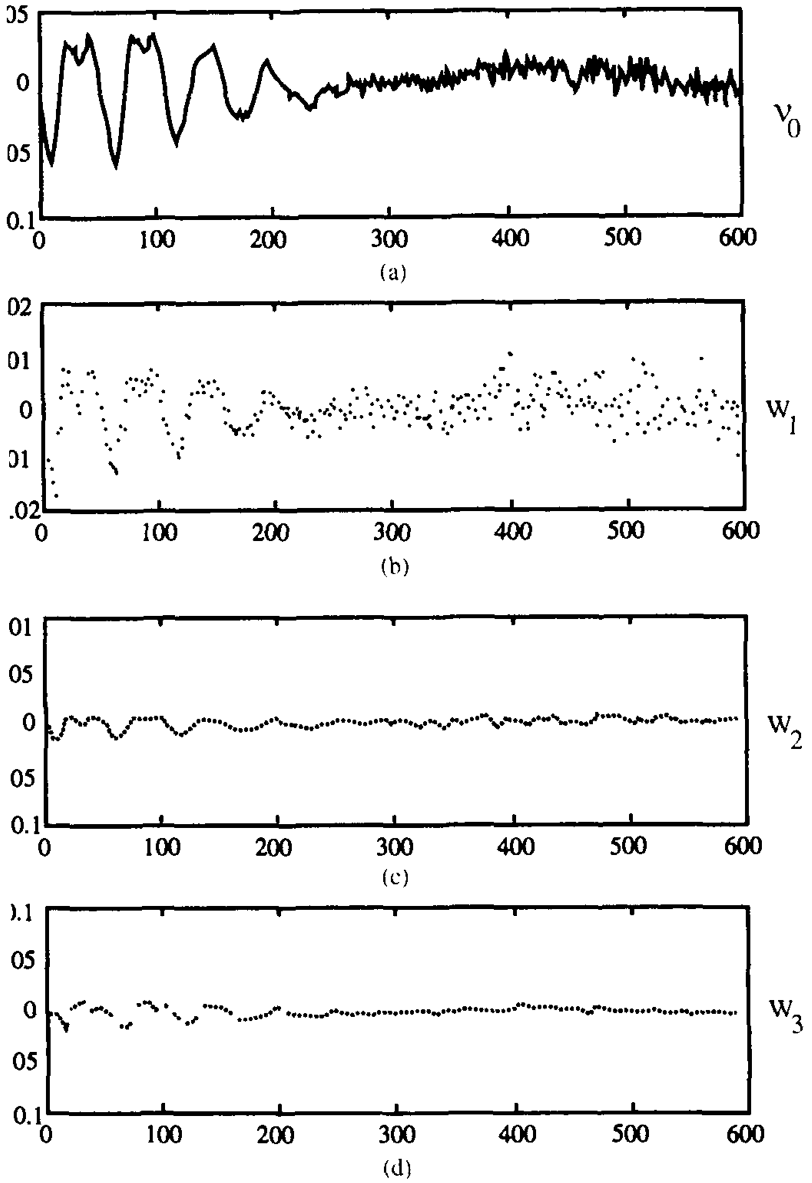


FIGURE 15.12 Discrete wavelet representations $d_{j-1,n}$, $d_{j-2,n}$, and $d_{j-3,n}$ of the speech signal shown in Figure 15.11.

scaling function is an orthogonal basis function at scale 2^j for the image function, $f(x, y)$:

$$\varphi_{kx,ky} = \varphi(2^j x - k_x, 2^j y - k_y) \qquad (15.46)$$

For the case where the two-dimensional scaling function is separable, we have

$$\varphi_{kx,ky} = \varphi(2^j x - k_x)\varphi(2^j y - k_y) \qquad (15.47)$$

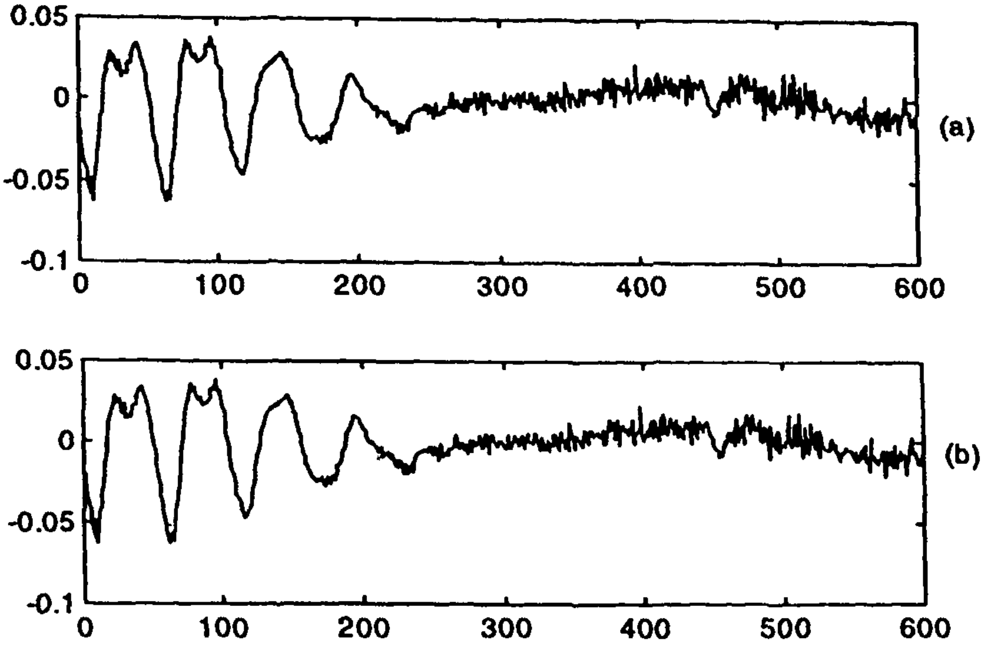


FIGURE 15.13 Comparison of the reconstructed signal with the original signal. (a) Original signal at resolution 1; (b) reconstruction of the original signal from the wavelet representation in Figure 15.12.

If $\psi_{kx,ky}(x,y)$ is its companion wavelet, then we can construct three different two-dimensional wavelets in addition to the above two-dimensional approximation function as follows:

$$\psi^1_{kx,ky}(x,y) = \varphi(2^jx - k_x)\psi(2^jy - k_y) \tag{15.48}$$

$$\psi^2_{kx,ky}(x,y) = \psi(2^jx - k_x)\varphi(2^jy - k_y) \tag{15.49}$$

$$\psi^3_{kx,ky}(x,y) = \psi(2^jx - k_x)\psi(2^jy - k_y) \tag{15.50}$$

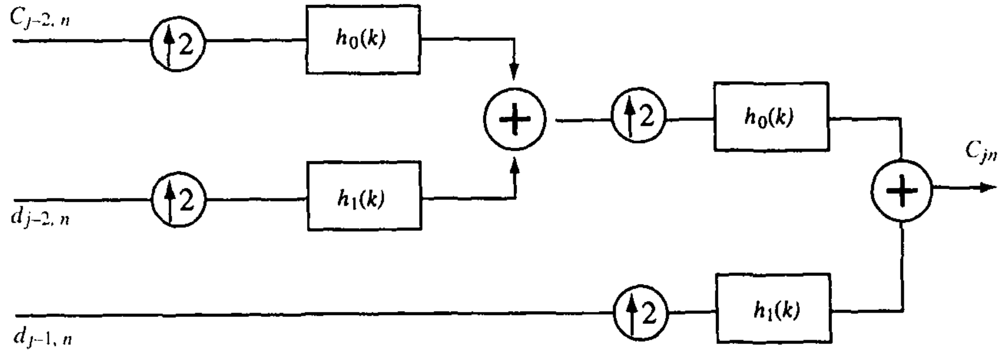


FIGURE 15.14 Multiresolution reconstruction structure.

The superscripts on the symbols ψ's are indices only. $\psi^l_{kx,ky}(x,y)$, $l = 1, 2, 3$, are all wavelets, since they satisfy the following condition:

$$\int\int_{-\infty}^{\infty} \psi_{kx,ky}(x,y)\,dx\,dy = 0 \qquad \text{for } l = 1, 2, 3 \tag{15.51}$$

Let us start with the image $f(x,y)$, i.e., $j = 0$, and the scale $2^j = 2^0 = 1$. The image can be expanded in stages. At each stage of the transform, the image is decomposed into four quarter-size images, each of which is formed by inner products of $f(x,y)$ with one of the wavelet basic images, followed by downsampling operation in x and y by a factor of 2. For the first stage decomposition (i.e., $j = -1$), we have

$$f^0_{-1}(x,y) = c^{\mathrm{LL}}_{-1,kx,ky} = \langle f(x,y)\varphi(2^{-1}x - k_x)\varphi(2^{-1}y - k_y)\rangle \tag{15.52}$$

$f^0_{-1}(x,y)$ is the first subimage giving the approximation coefficients at a coarse resolution of 2^{-1}. The other three subimages are, respectively,

$$f^1_{-1}(x,y) = d^{\mathrm{LH}}_{-1,kx,ky} = \langle f(x,y), \varphi(2^{-1}x - k_x)\psi(2^{-1}y - k_y)\rangle \tag{15.53}$$

$$f^2_{-1}(x,y) = d^{\mathrm{HL}}_{-1,kx,ky} = \langle f(x,y), \psi(2^{-1}x - k_x)\varphi(2^{-1}y - k_y)\rangle \tag{15.54}$$

$$f^3_{-1}(x,y) = d^{\mathrm{HH}}_{-1,kx,ky} = \langle f(x,y), \psi(2^{-1}x - k_x)\psi(2^{-1}y - k_y)\rangle \tag{15.55}$$

where $d^{\mathrm{LH}}_{-1,kx,ky}$, $d^{\mathrm{HL}}_{-1,kx,ky}$ and $d^{\mathrm{HH}}_{-1,kx,ky}$ are the detail coefficients at a coarse resolution of 2^{-1}.

The two-dimensional image function $f(x,y)$ can then be expressed as the sum of functions as shown below:

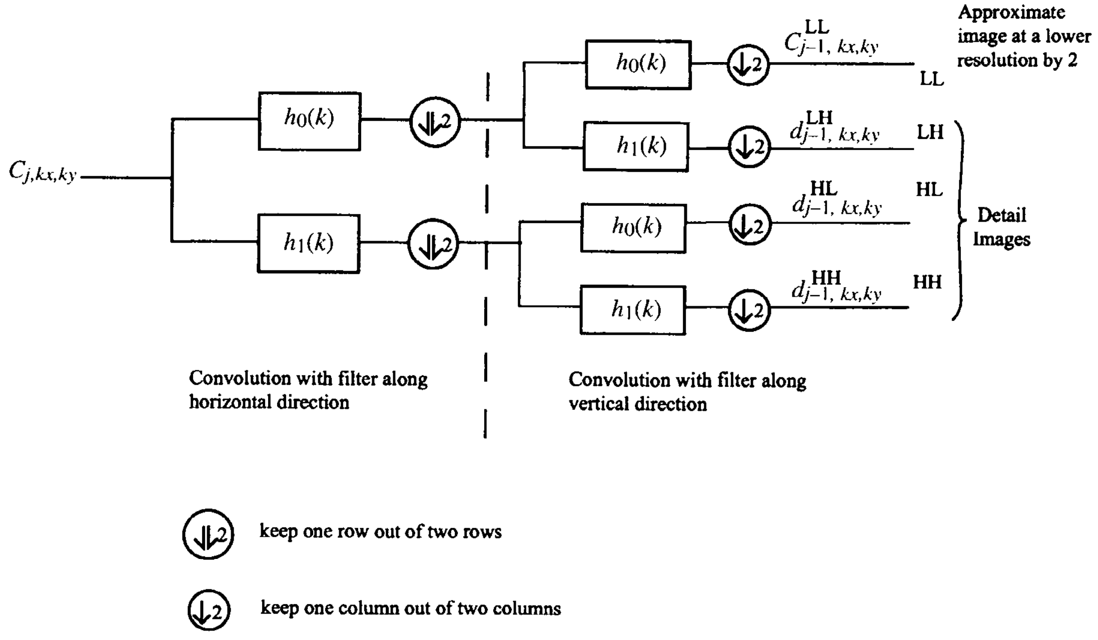
$$\begin{aligned} f(x,y) = {} & \sum_{kx}\sum_{ky} c^{\mathrm{LL}}_{-1,kx,ky}\varphi_{-1,kx,ky} + \sum_{kx}\sum_{ky} d^{\mathrm{LH}}_{-1,kx,ky}\psi^{\mathrm{LH}}_{-1,kx,ky} \\ & + \sum_{kx}\sum_{ky} d^{\mathrm{HL}}_{-1,kx,ky}\psi^{\mathrm{HL}}_{-1,kx,ky} + \sum_{kx}\sum_{ky} d^{\mathrm{HH}}_{-1,kx,ky}\psi^{\mathrm{HH}}_{-1,kx,ky} \end{aligned} \tag{15.56}$$

where

$$c^{\mathrm{LL}}_{-1,kx,ky} = \int\int_{-\infty}^{\infty} f(x,y)\varphi_{-1,kx,ky}(x,y)\,dx\,dy \tag{15.57}$$

Figure 15.15 shows one stage of the decomposition of an image. $c^{\mathrm{LL}}_{-1,kx,ky}$, $d^{\mathrm{LH}}_{-1,kx,ky}$, $d^{\mathrm{HL}}_{-1,kx,ky}$, and $d^{\mathrm{HH}}_{-1,kx,ky}$ can be computed with separable signal filters along the two coordinate axes. The two-dimensional DWT can be viewed as a one-dimensional DWT along the x and y axes.

Figure 15.16 shows the spectral distributions of the scaling function and each of the wavelets in the frequency domain. They occupy different regions of the two-dimensional spectral plane. The spectral bands that are labeled LL, LH, HL, and HH correspond to the spectra of the two-dimensional approximation function and wavelets $\psi^1_{kx,kx}(x,y)$, $\psi^2_{kx,ky}(x,y)$, and $\psi^3_{kx,ky}(x,y)$. The symbols L and



$C_{j,kx,ky}$
$h_0(k)$
$h_1(k)$
$h_0(k)$
$h_1(k)$
$h_0(k)$
$h_1(k)$
$C^{LL}_{j-1,\ kx,ky}$
$d^{LH}_{j-1,\ kx,ky}$
$d^{HL}_{j-1,\ kx,ky}$
$d^{HH}_{j-1,\ kx,ky}$
LL
LH
HL
HH
Approximate image at a lower resolution by 2
Detail Images
Convolution with filter along horizontal direction
Convolution with filter along vertical direction
keep one row out of two rows
keep one column out of two columns

FIGURE 15.15 One stage of the wavelet decomposition of an image.

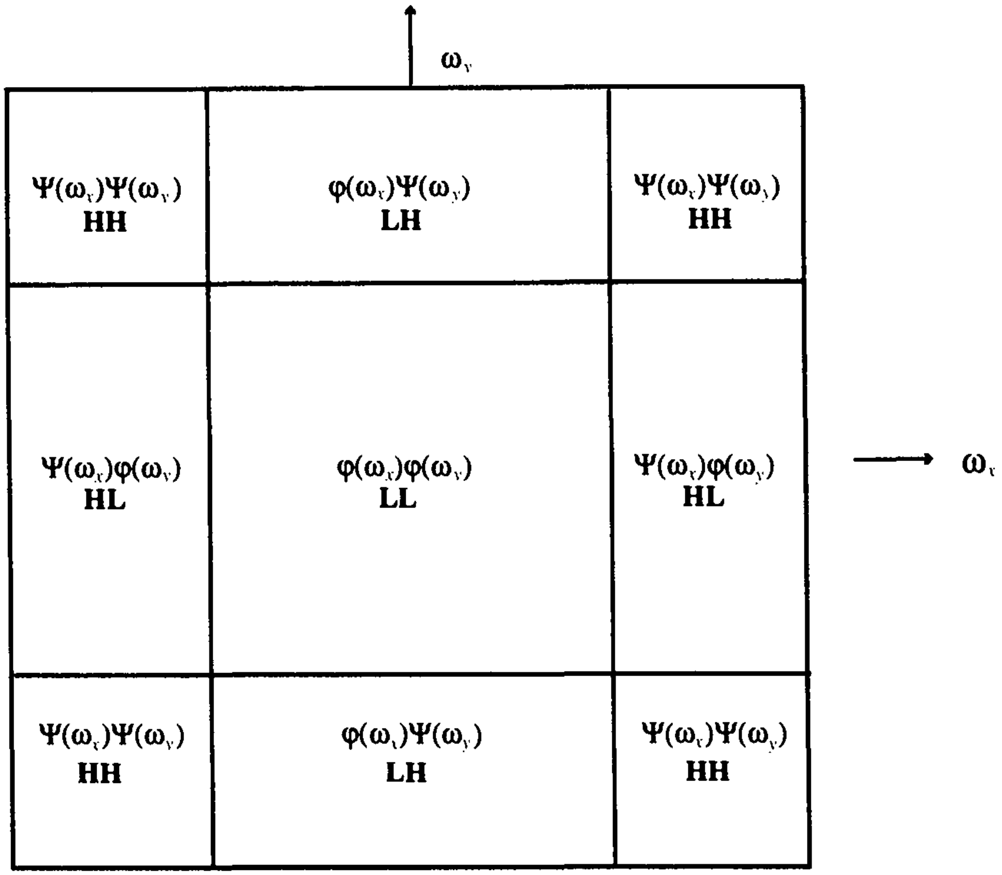

ω_y
$\Psi(\omega_x)\Psi(\omega_y)$
HH
$\varphi(\omega_x)\Psi(\omega_y)$
LH
$\Psi(\omega_x)\Psi(\omega_y)$
HH
$\Psi(\omega_x)\varphi(\omega_y)$
HL
$\varphi(\omega_x)\varphi(\omega_y)$
LL
$\Psi(\omega_x)\varphi(\omega_y)$
HL
ω_x
$\Psi(\omega_x)\Psi(\omega_y)$
HH
$\varphi(\omega_x)\Psi(\omega_y)$
LH
$\Psi(\omega_x)\Psi(\omega_y)$
HH

FIGURE 15.16 Spectral distribution of the scaling function $\varphi_{kx,ky}$ and the three wavelets, $\psi^1_{kx,ky}(x, y)$, $\psi^2_{kx,ky}(x, y)$, and $\psi^3_{kx,ky}(x, y)$, in the frequency domain.

H refer to whether the processing filter is lowpass or highpass. The region labeled LL represents the low-frequency contents in both x and y directions. It is the spectral distribution of the scaling function. That labeled LH represents low-frequency spectral contents in x and high-frequency contents in y direction. It is the spectral distribution of the wavelet $\psi^1_{kx,ky}(x, y)$. That labeled HL represents high spectral contents in x and low-frequency contents in y direction. It is the spectral distribution of the wavelet $\psi^2_{kx,ky}(x, y)$. The spectral distribution in the region labeled HH is from the wavelet $\psi^3_{kx,ky}(x, y)$, high-frequency contents in both x and y directions The approximation subimage, LL, can continue to decompose, and lower-resolution approximation subimages and detail subimages are obtained. Figures 15.17 and 15.18 show, respectively, the three-level wavelet decomposition of the standard image "Bridge", and "Lenna" with the center of the spectral domain shifted.

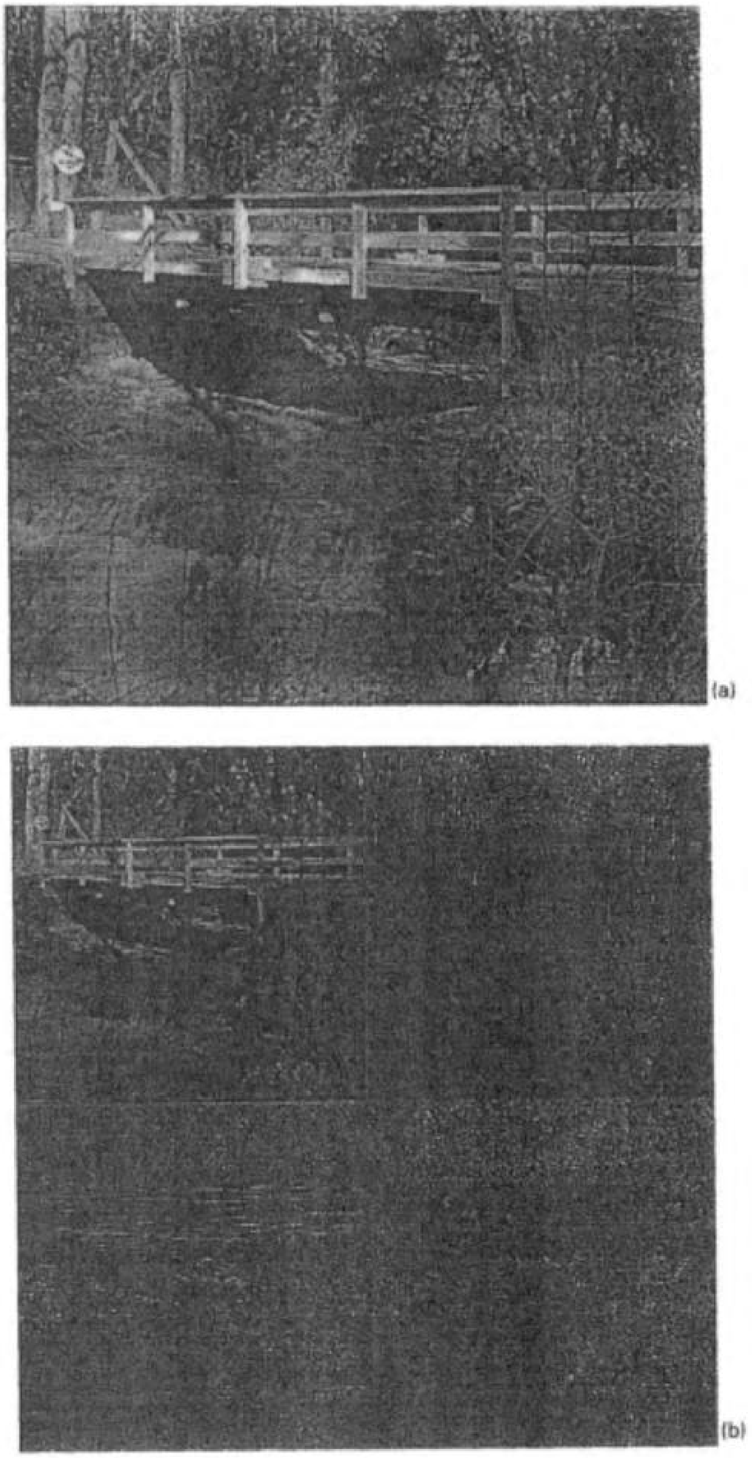


FIGURE 15.17 (a) The original image of the standard image "Bridge." (b) The one-level wavelet decomposition of the standard image "Bridge." (c) The two-level wavelet decomposition of the standard image "Bridge." (d) The reconstructed image with the IDWT.

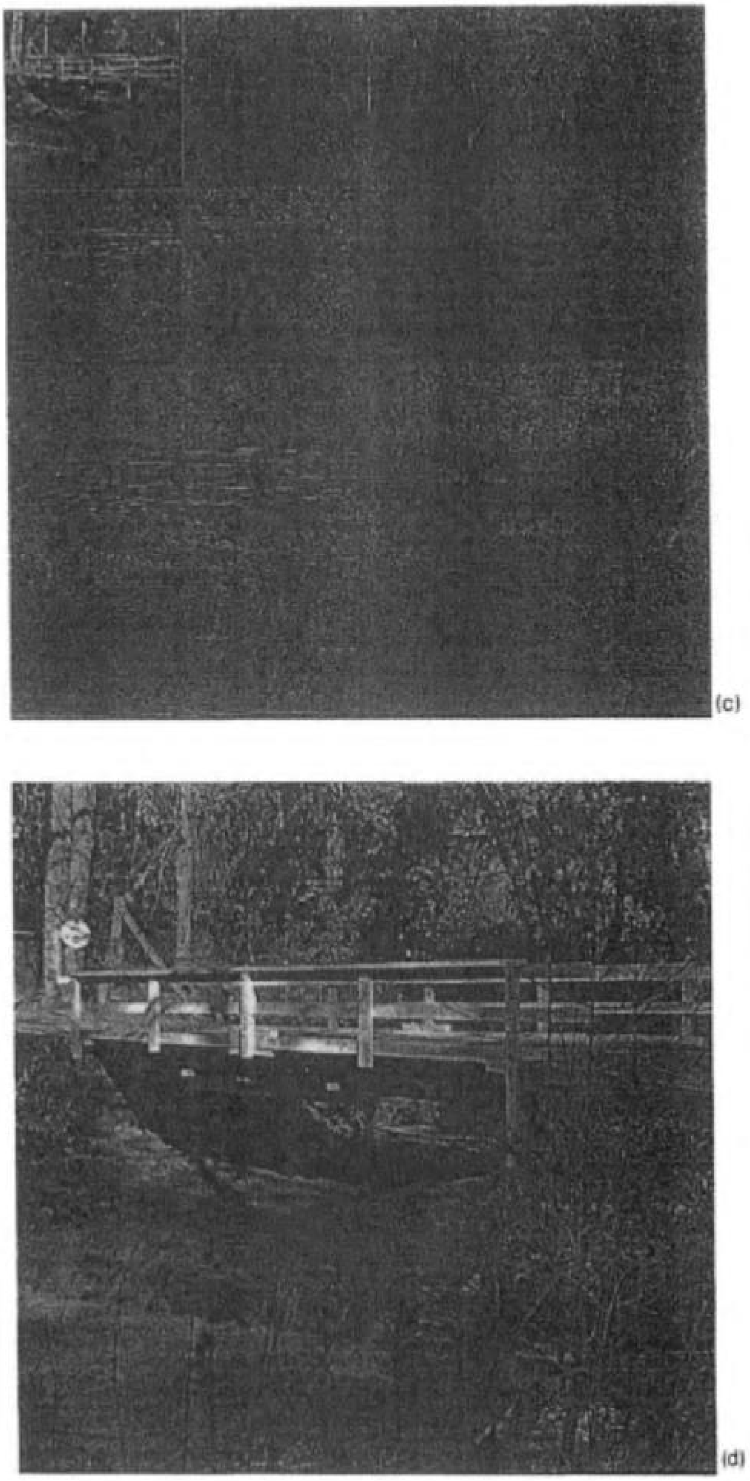


FIGURE 15.17 *Continued*

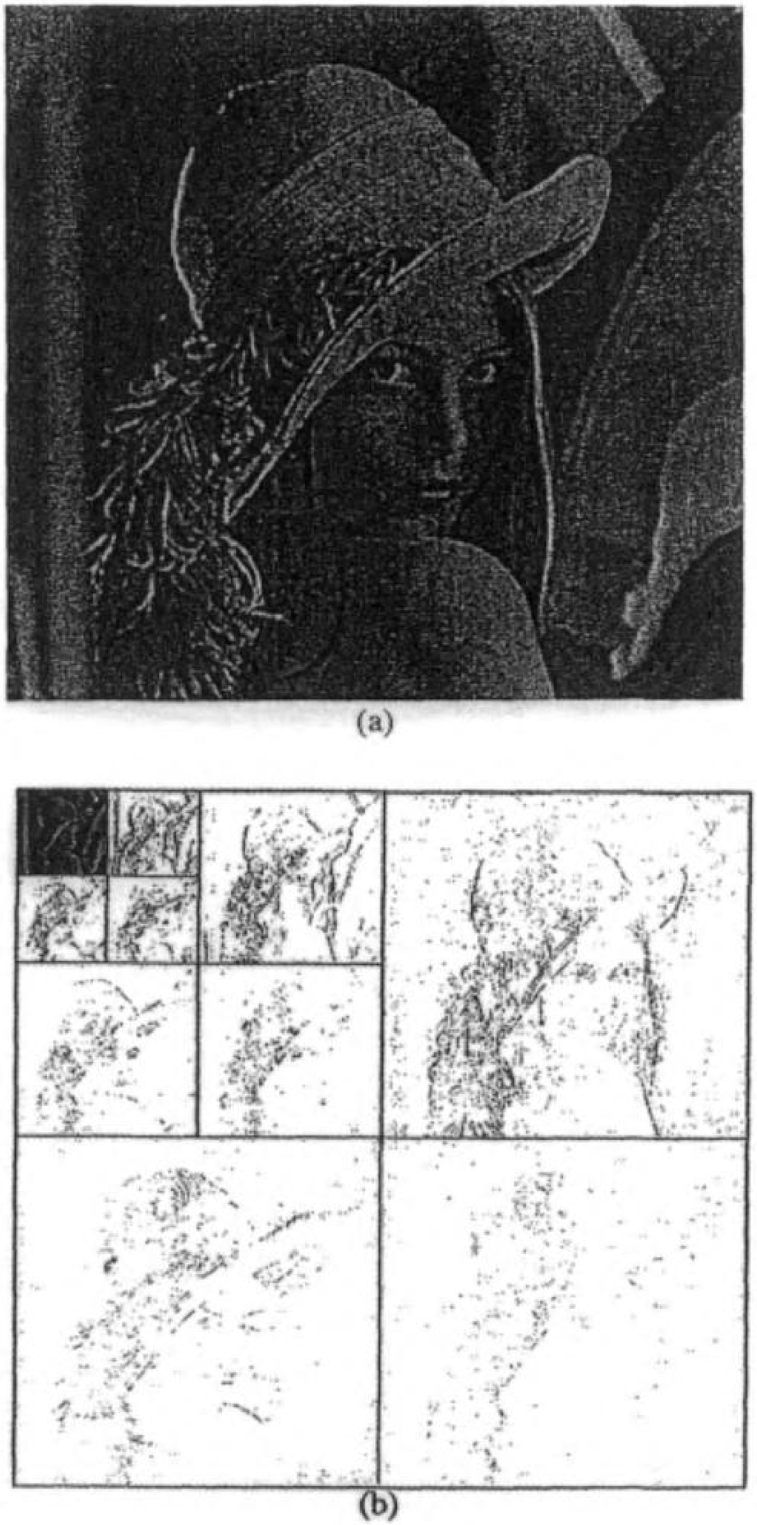


FIGURE 15.18 (a) The original image of the standard image "Lenna." (b) The wavelet decomposition of the standard image "Lenna," shown in (a).

Part IV

Applications

This Page Intentionally Left Blank

16

Exemplary Applications

Readers should now be aware that the two disciplines of pattern recognition and image processing are very intimately related. Although they each have their own applications, they frequently come together as "PRIP," with digital image processing as data preprocessing for pattern recognition. This has been a ramification of artificial intelligence (AI), but it is growing so fast that there are lots of independent activities going on and there even are specific professional societies (e.g., International Pattern Recognition Society and its subsidiary national societies in countries all over the world).

The reason this discipline could grow so fast is strong support from applications. PRIP can be applied to various areas to solve existing problems. In turn, the various requirements posed during the process of resolving practical problems motivate and speed up development of this discipline.

PRIP has a lot of military applications: ground-to-air surveillance in the detection and identification of incoming planes; air-to-ground surveillance in monitoring enemy troop deployment, reconnoitering enemy airfields, monitoring additions or changes in the enemy's surface and underground military installations; and coast and border surveillance.

In addition, PRIP has many civil applications. Some medical applications are:

Examination of cytological smears, to assist in the detection and on-time treatment of breast cancer, cervical cancer, and so on
Computerized tomography to help locate the tumor inside the brain through three-dimensional scanning
Interpretation of electrocardiograms (EKGs) and electroencephalograms (EEGs)
Enhancement and transmission of chest x-ray negatives

Examples of industrial applications:

Automated inspection, sorting, and/or pickup of machine parts in production lines
Automated warehouse, to help robots to pick up parts from shelves or bins
Automated inspection of printed circuit boards (PCBs) to locate broken lines, short circuits between lines, soldering cavities, and so on
Automated pin soldering of PCBs under a microscope
Nondestructive testing using metallography while casting to detect blow holes, deformations, and so on
Drug tablet inspection
Button inspection
Chocolate bar inspection

Examples of forensic applications:

Fingerprint identification
Face identification
Radar-timed speed monitoring

Examples of use of remote sensing images:

Railway line development
City planning
Pollution control
Forest fire monitoring and control, especially for hidden fires
Agricultural applications such as crop inventory and monitoring and management over a wide agricultural area

Other applications:

Seismic wave analysis for earthquake prediction
Petroleum deposit exploration (artificial earthquakes)
Acquisition and interpretation of a geological picture for mineral ore deposit
Weather forecasting
Archiving and retrieval of documents, including mixed text/graphics

There is no way to discuss all these applications in this limited space. Only a few are discussed in the following sections. Interested readers can refer to related periodicals and proceedings for other applications.

16.1 DOCUMENT IMAGE ANALYSIS

16.1.1 Recognition and Description of Graphics

There is no doubt of the use of computers in archiving and retrieval of documents. What remains would be how to store and retrieve them more effectively, especially in the case of mixed text/graphics data. Two problems are involved: (1) effective separation of character strings from intermixed text/graphics documents; and (2) graphics understanding and the generation of succinct descriptions of them. In this section a generic graphics interpretation system for the generation of a description of the contents of a paper-based line drawing is described. Frequently, graphics in engineering documents are diversified to a vast extent. But if we focus only on the shape primitives and their structural relationships, they can be grouped into only a few categories:

Graphics consisting mainly of polygonal shapes
Graphics consisting mainly of curved shapes
Graphics consisting mainly of special or user-defined shapes
Graphics consisting mainly of higher-order curves and/or polylines
Combination of one or several of the above

In addition, these entities have various attributes associated with them. Examples of such attributes are thickness of lines and filling details for graphical primitive. Very frequently, these entities overlap with one another and obscure some of the lines. Graphics are typically annotated with text strings.

In the system developed [Bow and Kasturi (1990), and Kasturi, Bow, et al. (1990)], a mixed text/graphics document is first digitized at a resolution of 12 pixels per millimeter to generate a 2048 × 2048 pixel binary image. To obtain a graphics description file of the highest possible level with minimum operator interaction, the following operations are performed on this image: (1) separation of text strings from graphics and (2) automated generation of a structural description for the graphics.

Separation ofText Strings from Graphics

The algorithm described in this section [Fletcher and Kasturi (1988), Bow and Kasturi (1990)] is used to separate text strings of various orientations, font styles, and character sizes. This segmentation algorithm is based on grouping collinear connected components of similar size and does not recognize individual characters. There are two principal steps in this algorithm: (1) connected component generation and (2) collinear component grouping in the Hough domain. These segmentation steps are described briefly in the following sections.

Connected component generation. The connected components in the digitized image are isolated by raster-scanning the image and growing the components as they are found. The algorithm keeps track of the top-, bottom-, left-, and rightmost pixels corresponding to the smallest enclosing rectangle of each component, and the percent of pixels within this rectangle that are of the foreground type. The rectangles enclosing the components in Figure 16.1 are shown in Figure 16.2. Note that each component of a text character is enclosed in a separate rectangle except for characters that are connected to graphics (the letters T and p inside the table). The connected component data are used by other

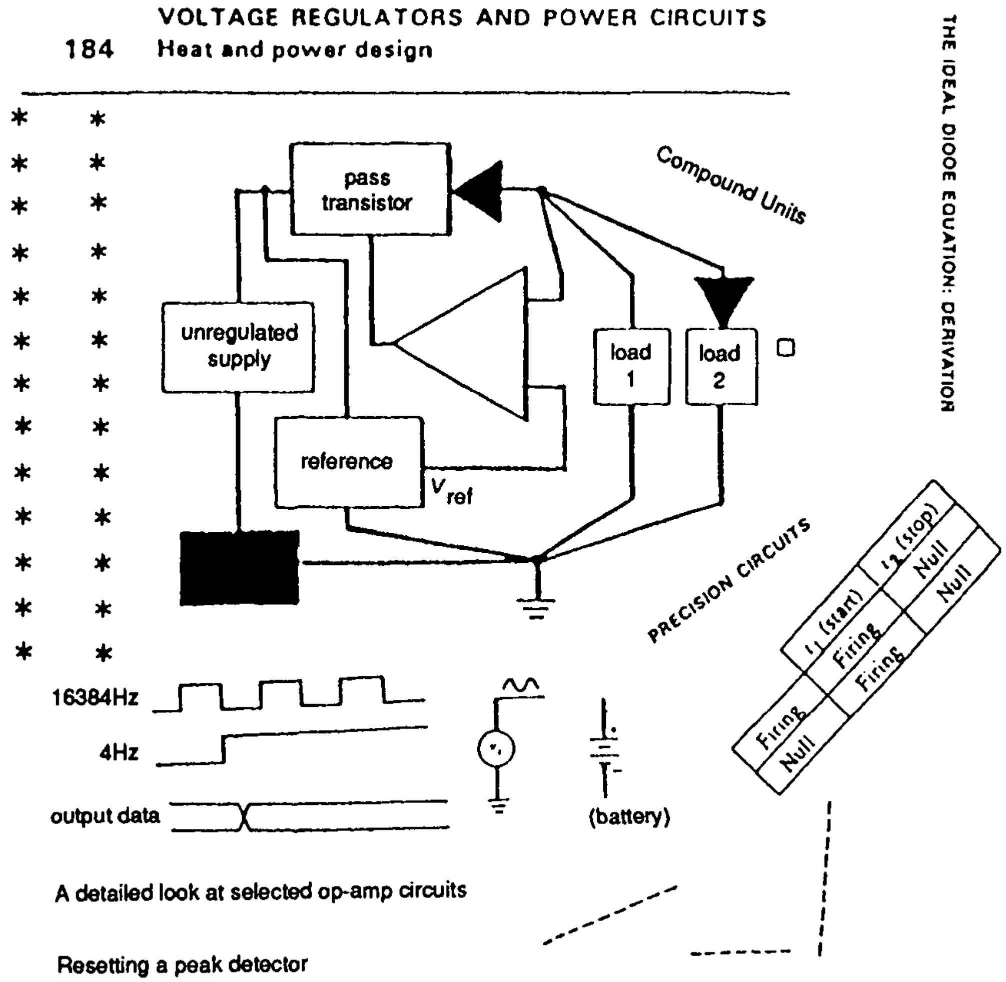


FIGURE 16.1 Test image 1. (From Bow and Kasturi, 1990.)

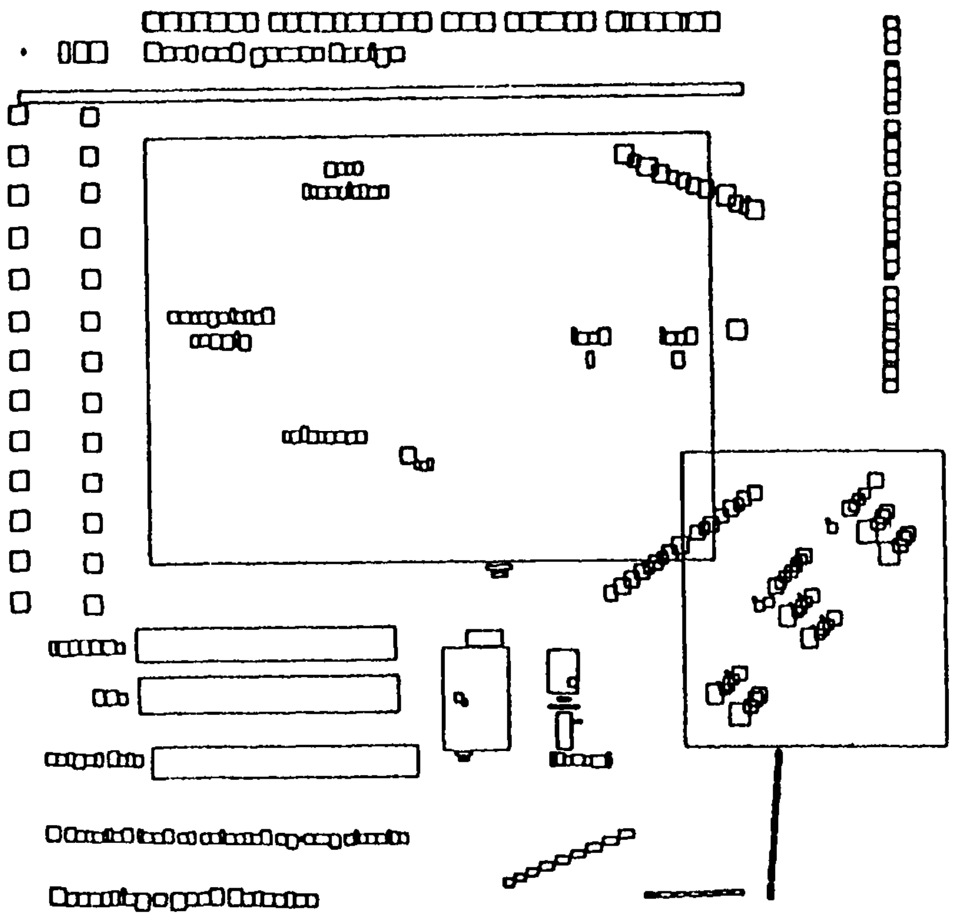


FIGURE 16.2 Connected components in test image 1. (From Bow and Kasturi, 1990.)

stages of the text segmentation algorithm, thereby minimizing the operations on the large image array.

An area filter is designed to identify those components that are very large compared to the average size of the connected components in the image, and to group them into graphics, since in a mixed text/graphics image, such components are probably appropriately so categorized. By obtaining a histogram of the relative frequency of occurrence of components as a function of their area, an area threshold, which broadly separates the larger graphics from the text components, is chosen. Consequently, connected components, which are enclosed by rectangles with a length-to-width ratio larger than 10 (e.g., long horizontal or vertical straight lines), are marked as graphics instead of text characters. With these filters the two large rectangles and the two thin rectangles are removed from Figure 16.2 as graphics.

Collinear component grouping in the Hough domain. Let us define a text string as a group of at least three characters which are collinear and satisfy certain proximity criteria. The collinear characters are then identified by applying the Hough transform to the centroids of the connected components. In this implementation, the angular resolution θ in the Hough domain is set at 1 degree, whereas the spatial resolution ρ is set at 0.2, which is the average height of the connected components, thus providing a threshold for the noncollinearity of the components of the text. The Hough domain is scanned first only for horizontal and vertical strings, then for all others. When a potential text string is identified in the Hough domain, a cluster of cells centered around the primary cell is extracted. This is done to ensure that all characters belonging to a text string are grouped together.

To decrease the time spent in performing the Hough transform, a pyramidal reduction of the resolution is used. In this case the reduction in resolution is 3 in ρ and 2 in θ, producing an array that is at each level one-sixth the size of the preceding level. The method of reducing the resolution is a maximum operator, so that the maximum string length at the base is represented in the top level. There are a total of five levels in this implementation, reducing the resolution by a total factor of $6^4 = 1296$. This means that scanning of the Hough domain is reduced computationally by a factor of 1296, but the individual string extraction time will be increased. This condition is almost always satisfied in a typical document.

After a collinear string is extracted from the Hough domain, it is checked by an area filter, which is similar to the one discussed earlier, so that the ratio of the largest to smallest component in the group is less than 5. This is necessary to prevent large components, which do not belong to the string under consideration (but having their centroids in line with the string) from biasing the thresholds, such as intercharacter and interword gaps. The components are further analyzed to group them into words and phrases using the following rules:

1. Intercharacter gap less than or equal to A_h
2. Interword gap less than or equal to $2.5 \times A_h$

Here A_h is the local average height and is computed using the four components that are on either side of each gap.

The algorithm described above classifies broken lines (e.g., dashed lines) and repeated characters (e.g., a string of asterisks) as text strings since they satisfy all the heuristics for text strings. It is desirable to label such components as graphics. On the other hand, text strings in which some of the components are connected to graphics are incorrectly segmented (e.g., underlined, words, in which characters with descenders touch the underline). Thus the strings that are identified are further processed to refine the segmentation [Fletcher and Kasturi (1988)].

Three more operations are involved in this algorithm: (1) separation of solid graphical components, (2) skeletonization and boundary tracking, and (3) segmentation into straight-line curves. The data generated include coordinates of endpoints, line type (straight or curved, continuous or dashed), line source (outline of solid object or core line of thin entity), and line thickness. For dashed lines, the segment and gap lengths are also included. For curved lines, a second file containing an ordered list of pixels using an eight-direction chain code is created. These files are processed by the graphics recognition and description algorithm described in the following section.

Automated Generation of a Structural Description for the Graphics

Necessity in implementing heuristic concepts to the graphics understanding for their succinct description. What comes so naturally to human segmentation of an image into meaningful objects is an extremely computationally intensive and ambiguous task for the computer. As a matter of fact, the human segmentation is an outcome of a very complicated process which we do not really realize. What the computer thinks about a graphic is actually a 2048 × 2048 or 512 × 512 bitmap. If a computer is taught to search a shape, it will search out all the closed shapes that might form from the given straightline or curved segments. The number of complex polygons generated might count up to several tenfold what they should be. Obviously, this does not lead to a succinct description, but on the contrary, will complicate it. The computer should be further taught to recognize the graphics more effectively and efficiently than human segmentation does. Heuristic concepts should therefore be implanted in the algorithm enable the system to understand the graphics exactly as a human expert does. The system is so equipped that it can generate indispensable loops (i.e., loops with minimum redundency). These loops will then serve as input to the next-higher-level processor to generate a structural description of the graphics.

Heuristic concepts will be introduced in two separate levels of processing. Some will be introduced during the closed-loop searching process to constrain the search to indispensable loops only. No doubt, this is done under the condition that no information will be lost. Other heuristic concepts would be left to higher-level processing, where decomposition on the complicated segmented images will be conducted.

Automated generation of forms with minimum redundancy from graphics consisting mainly of polygons and straight-line segments [Bow and El-Masri (1987), Bow and Zhou (1988)]. Input data obtained from the preprocessing part of the system are in the form of a group of line segments. For each line segment we can establish two neighbor lists: the head neighbor list and the tail neighbor list. Once these neighbor lists are built up, we no longer care about the length varieties or the orientation of the line segments.

Some definitions are helpful for the process. A line segment is designated as a terminal line segment *T* if it does not have a neighbor at its head, tail, or neither of them. If a line segment has one and only one neighbor at both its head and tail, it is designated as a path line segment *P*. If a line segment has two or more than two neighbors at either its head or its tail, it is designated as a branch line segment *B*. Note that some of the line segments would chain-cluster into a loop, while others would not. Let us designate those chain clusters that cannot form closed loops as PATH-1 and these that would probably form a loop as PATH-2. Both PATH-1 and PATH-2 are orderly sequences of line segments. PATH-1 will be in string form, such as T, TB, TPB, TPPB, TPPPB, . . . , and TT, TPT, TPPT, TPPPT, and so on, while PATH-2 will come out as a segment string with B's at both ends, such as B, BB, BPB, BPPB, . . . , or a continual and closed segment string of P's, such as PPP, PPPP, . . . (see Figure 16.3 for details).

Searching of PATH-2 can be started at any branch line segment as long as it has not been grouped into previous PATH-2's. If all the branch line segments have been included in PATH-2's and there are still path line segments left untraced, search for a new PATH-2 starting at any one of the path line segments. This will result in a string form such as PPP, PPPP, . . . , indicating that they are isolating forms.

A careful glance at the graphics shown in Figure 16.4 will show that an enormous number of loops could be traced out from these line segments. Even with this simple graphic, over 100 different polygons can be obtained. This would definitely increase the complexity instead of helping to obtain a succinct graphic description. In this sample graphic, 21 polygons (Figure 16.4b and c) are sufficient to keep all the information. Our problem now becomes to design an

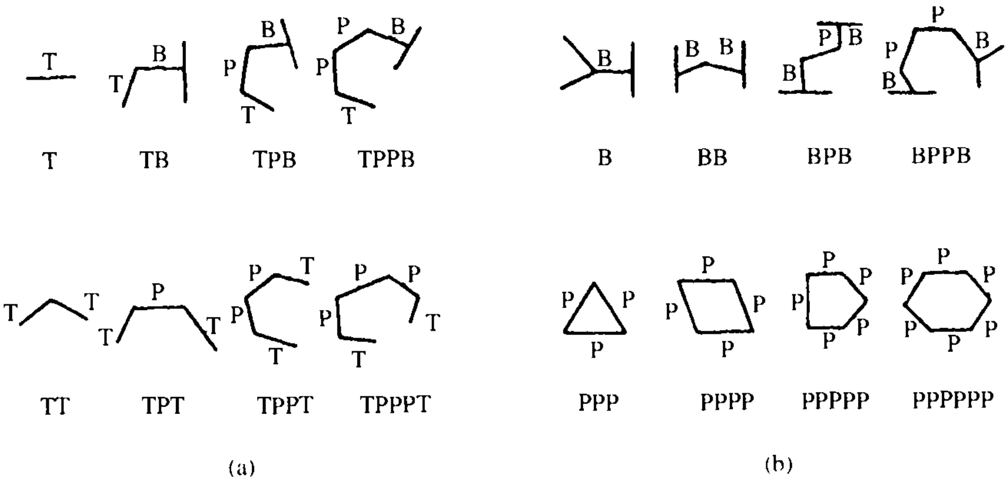


FIGURE 16.3 Two types of chain clusters from line segments. (a) PATH-1 clusters; (b) PATH-2 clusters. (From Bow and Zhou, 1988.)

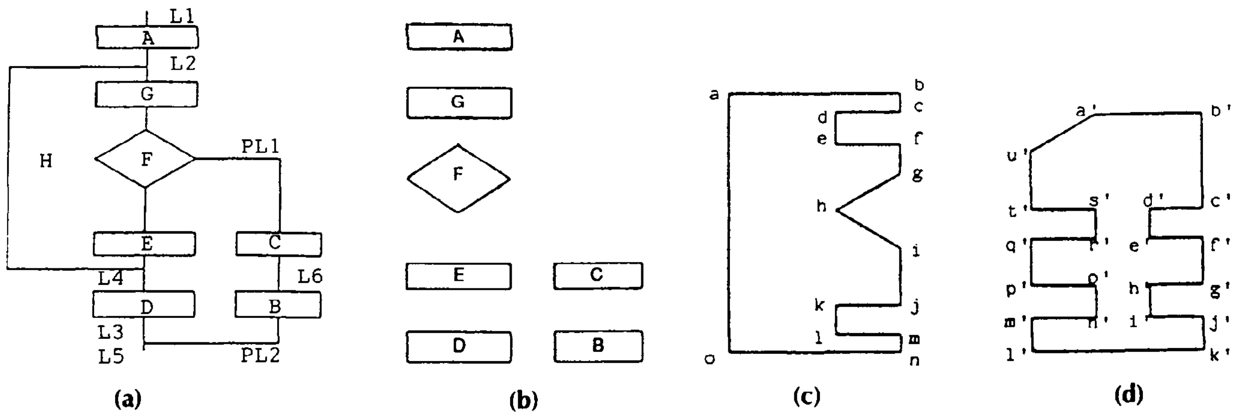


FIGURE 16.4 Polygonal loop generation. (a) Original graphic; (b) regular primitive forms generated by the system; (c) complex form generated by the system; (d) another complex form generated by the system. (From Bow and El-Masri, 1987.)

algorithm to generate these 21 polygons (no more and no less). Loops generated can be classified into three categories: (1) self-loop (SELFLOP), (2) simple loop (SIMLOP), and (3) complex loop (CMXLOP).

SELFLOP is a loop formed with a single PATH-2 sequence, PPP, PPPP, . . ., or BPB, BPPB, BPPPB, ... SIMLOP is defined here as a loop composed of edges less than and up to six in number, with the hope that it would come out to be a more or less regular form. CMXLOP is a loop composed of more than six edges. In the graphic shown in Figure 16.4a, there are 10 SELFLOPs, 7 SIMLOPs, and over 100 CMXLOPs. A SIMLOP may be a simple shape such as a triangle, rectangle, or rhombus, but that is not so for a CMXLOP, which may come out as a barely described polygonal figure.

Automating the generation of their structural description. As defined and generated by this system, SIMLOPs can be nothing but such forms as triangles, rectangles, trapezoids, rhombus, parallelograms, regular pentagons, regular hexagons, and horizontal hexagons. Once these forms were recognized, they (except the irregular ones) can be specified by only a few parameters, thus greatly reducing storage and increasing ease of retrieval.

A CMXLOP is a barely described polygonal figure. However, in many cases a CMXLOP can be ingeniously described by a human expert as a combination of two or more primitive forms grouped together in an overlapped, nested, intersected, perspective, subtractive, or additive mode. So some ideas can be transplanted to the system from human perception and cognition. To explain our approach, it may be helpful to use an example.

For the graphic shown in Figure 16.4a, the SIMLOPs and CMXLOPs generated by our system are listed in parts (b), (c), and (d). The seven SIMLOPs can be machine-recognized and stay where they are. The CMXLOP shown on (c) consists of 15 edges and is obviously not a convex hull. Note also that the edges sets [*cd*, *de*, *ef*], [*gh*, *hi*], and [*jk*, *kl*, *lm*] of this CMXLOP share edge with SIMLOPs *G*, *F*, and *E* in the form of segmented images.

If vertices *c* and *f*, *g* and *i*, and *j* and *m* are connected together by *cf*, *gi*, and *jm*, respectively, the CMXLOP *abcde* · · · *klmno* turns out to be a convex hull, and the figure looks as a rectangle *abno* overlapped by three primitive shapes: rectangle *G*, rhombus *F*, and another rectangle *E* with portions of the rectangle hidden. Similarly for the CMXLOP shown in Figure 16.4d, a convex hull can be obtained by connecting $c'f'$, $g'j'$, $m'p'$, and $q't'$. The CMXLOP can then be interpreted as a polygon $a'b'k'l'u'$ overlapped by four rectangles *E*, *D*, *C*, and *B*.

After analysis by this system, the graphic shown on Figure 16.4a can then be intepreted as composed of nine primitive forms, among which the rectangles *A* and *G*, rhombus *F*, and rectangles *E* and *D* are positioned in a top-down spatial relationship. They are symmetrical with respect to the right edge *bn* of the rectangle *abno*, with the other three edges *no*, *oa*, and *ab* forming a feedback loop. Rectangles C and B are in a spatial top-down relation and situated side by side with rectangles E and D, respectively. C and B are positioned such that they are symmetrical with respect to the right vertical edge $b'k'$ of the polygon as a branch loop.

For the same graphic there exists another interpretation which is as succinct as the preceding one. That is, instead of interpreting it as overlapping of primitives G, F, and E on the rectangle *abno* and E, D, C, and B on polygon $a'b'k'l'u'$, we can break these two CMXLOPs into two sets of line segments [ab, bc, fg, ij, mn, no, oa] and [$a'b'$, $b'c'$, $f'g'$, $j'k'$, $k'l'$, $l'm'$, $p'q'$, $t'u'$, $u'a'$], and use them to link the seven primitive shapes together in the correct order. Which one of the interpretations above we should follow will depend on the specific subgraphic and its relation to other subgraphics. The decision making will be left to a higher-level processing. This will, of course, challenge the system to show its intelligence.

Take another graphic (Figure 16.5a) for interpretation. SIMLOPs and CMXLOPs intelligently generated by our system are shown in Figure 16.5b and c. Obviously, there is no need for CMXLOP #3 to exist. The reason is that nearly all of the edges (except $b'c'$) of this CMXLOP #3 are already included either in the other three CMXLOPs or in some of the SIMLOPs, namely,

$f'e'$	included in	ab
$e'd'$	included in	bc
$a'f'$	included in	$a'''l'''$
$a'b'$	included in	$a'''b'''$
$d'c'$	included in	rectangle #0

This can be machine-checked without too much difficulty. For CMXLOP #1, six of the 12 edges are included in SIMLOPs #M and #L, namely

$i''j''$, $j''k''$, $k''l''$	included in SIMLOP M
$c''d''$, $d''e''$, $e''f''$	included in SIMLOP L

For this CMXLOP #1 we have to weigh and see which method of interpretation is more advantageous to our succinct description: Keep it in a convex hull shape, or break it down into line segments? Keeping it as a convex hull (i.e., by connecting $i''l''$ and $c''g''$) would lead to missing one line segment, $f''g''$. Furthermore, it cannot provide us with new, meaningful information, since there is no such reference pattern in our system.

Based on the facts listed above, there is not much benefit if we keep this CMXLOP #1 for graphic interpretation. Instead, we can decompose this CMXLOP #1 into polyline ($f''g''$, $g''h''$, $h''i''$) and line segments ($l''a''$, $a''b''$, and $b''c''$), where $a''b''$ will finally combine with other line segments to give a long line segment, L22. Similar arguments can be applied to CMXLOP #2 and #4. CMXLOP #2 will be replaced by line segments (ab, bc, fg, hi, la). CMXLOP #4

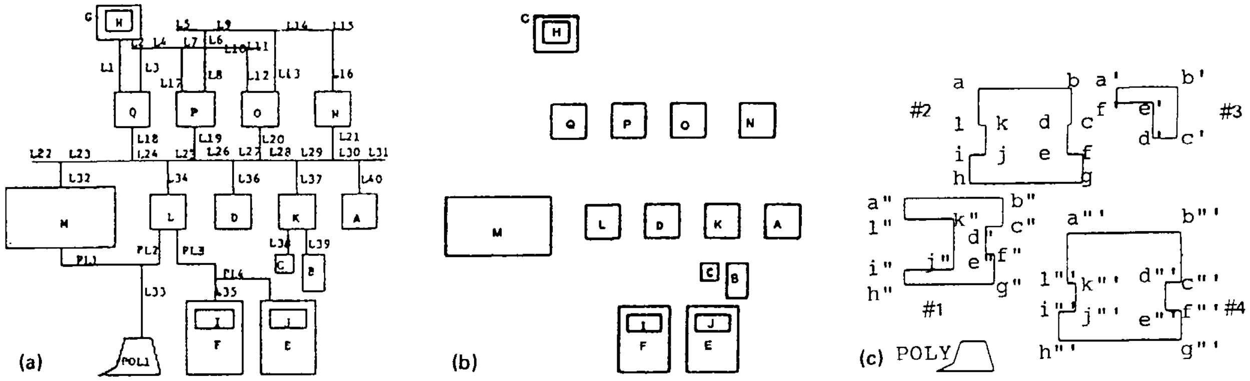


FIGURE 16.5 Decomposition on complex loop. (a) Original graphic; (b) regular primitive forms generated by the system; (c) complex forms generated by the system. (From Bow and El-Masri, 1987.)

will be replaced by line segments ($a'''b'''$, $b'''c'''$, $f'''g'''$, $g'''h'''$, $h'''i'''$, $l'''a'''$). The results obtained from the original graphic after the decomposition procedure will be 17 rectangles and one polygon linked together by 18 line segments in appropriate spatial relationships which can be specified clearly. Note that some of the line segments will be further combined to form long line segments by using their collinearity property.

Ten different graphics of different degrees of complexity were used as samples for the experiments. Figure 16.6a shows the input graphic with an unknown misalignment angle. Figure 16.6b and c, respectively, show the description generated for the graphic, and the graphic reconstructed from the structural description. Figure 16.7 shows the results obtained on the other graphics. Note that the complex loops generated by the system have finally been broken down into segments. Scenes from computer vision on overlapping objects have also been taken for experiments.

16.1.2 Logic Circuit Diagram Reader for VLSI-CAD Data Input

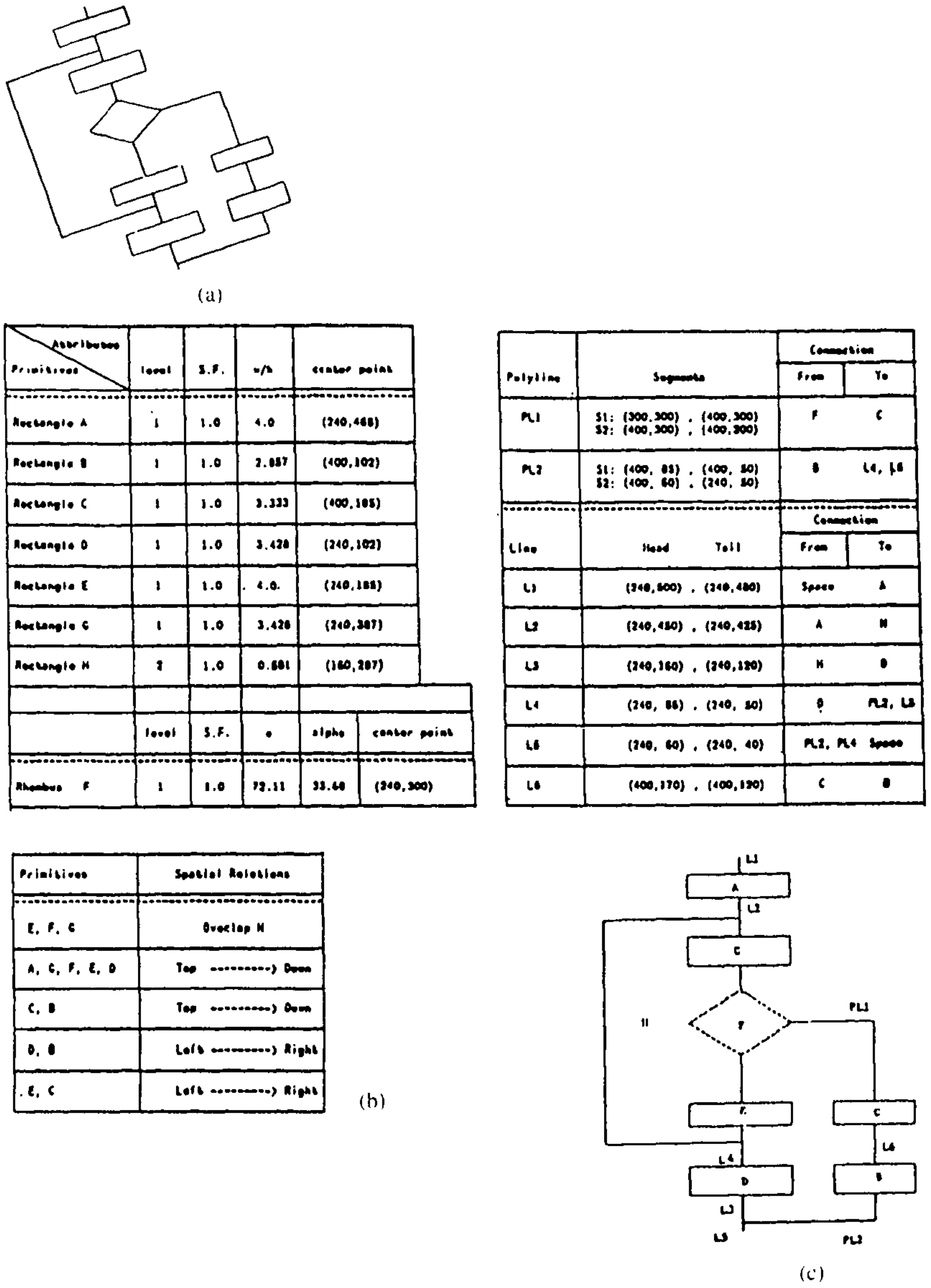
Although CAD systems have come into wide use to speed up engineering design, skillful work must still be done on drawing paper. In this section a logic circuit diagram reader using a pattern recognition technique for the automatic input of data into CAD system is discussed.

In a logic circuit diagram, many symbol patterns appear. The total number of these patterns may be on the order of several hundreds (see Figure 16.8). Nevertheless, some of them are very similar in morphology. If we analyze these symbols from the morphological point of view, most of the logic symbols can be grouped into two main categories. One consists of loop(s), such as triangles, rectangles, diamonds, and user-defined symbols, while the other does not. Let us designate symbols with loop(s) as loop symbols and the others as loop-free symbols.

Although the set of symbols used in the drawing is predefined and also drawn with a template, variations in their size, orientation and position exist. All these complicate the solution to this problem. Okazaki et al. (1988) suggested a two-stage recognition procedure for its solution: symbol segmentation and symbol identification.

Symbol Segmentation

Symbol segmentation is the first task and a key stage in this approach, wherein the legitimacy of a candidate loop is ascertained and the relevant region of the diagram is delimited. In this process a minimum region sufficient for symbol identification [called the minimum region for analysis (MRA)] is isolated from

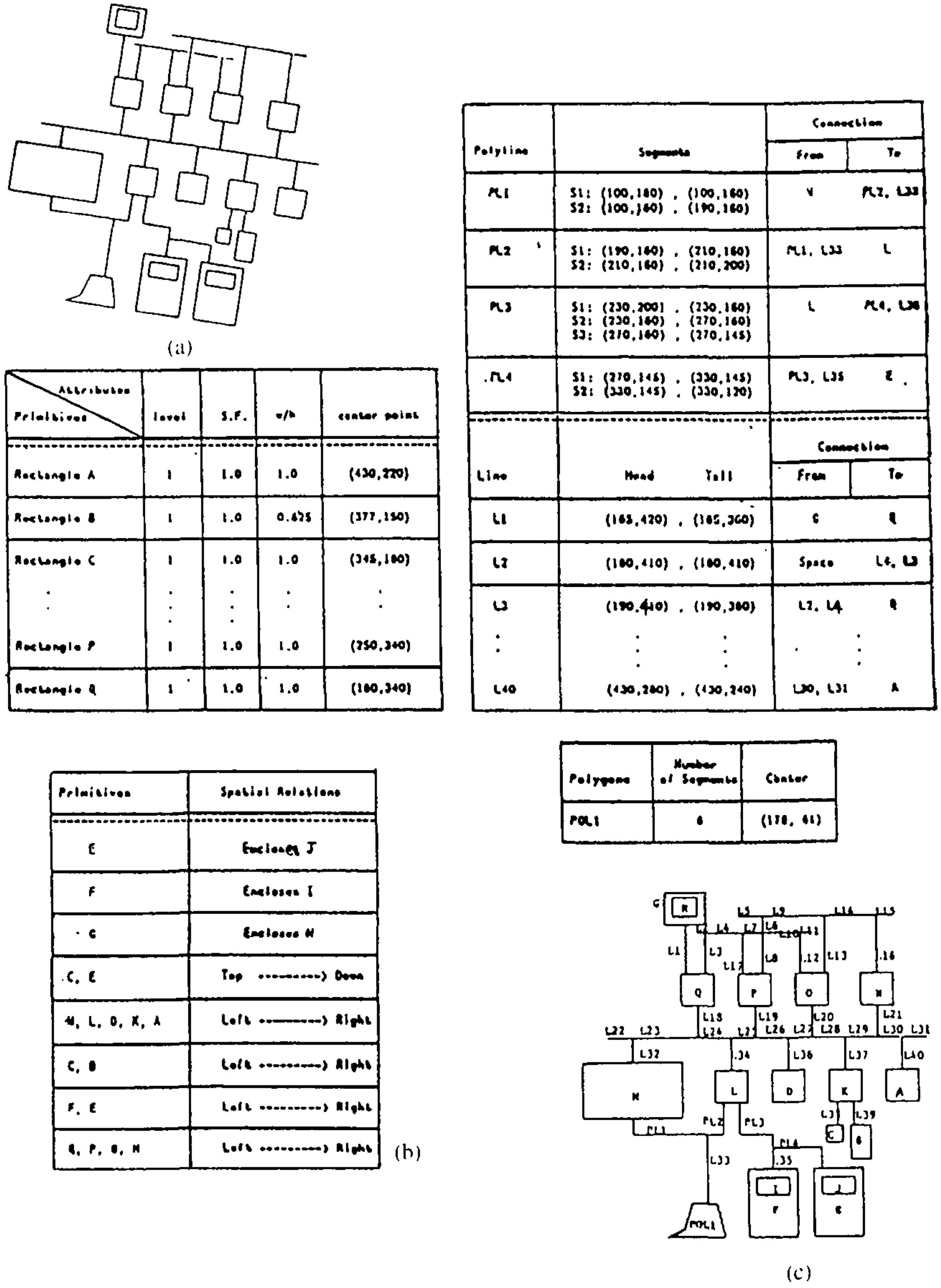


Attributes / Primitives	level	S.F.	w/h	center point	
Rectangle A	1	1.0	4.0	(240,468)	
Rectangle B	1	1.0	2.857	(400,102)	
Rectangle C	1	1.0	3.333	(400,185)	
Rectangle D	1	1.0	3.428	(240,102)	
Rectangle E	1	1.0	. 4.0.	(240,188)	
Rectangle G	1	1.0	3.428	(240,387)	
Rectangle H	2	1.0	0.881	(160,287)	
	level	S.F.	a	alpha	center point
Rhombus F	1	1.0	72.11	33.69	(240,300)

Polyline	Segments	Connection	
		From	To
PL1	S1: (300,300) , (400,300) S2: (400,300) , (400,200)	F	C
PL2	S1: (400, 85) , (400, 50) S2: (400, 60) , (240, 50)	B	L4, L5
		Connection	
Line	Head Tail	From	To
L1	(240,500) , (240,480)	Space	A
L2	(240,450) , (240,425)	A	H
L3	(240,160) , (240,120)	H	D
L4	(240, 85) , (240, 50)	D	PL2, L5
L5	(240, 60) , (240, 40)	PL2, PL4	Space
L6	(400,170) , (400,130)	C	B

Primitives	Spatial Relations
E, F, G	Overlap H
A, G, F, E, D	Top ---------) Down
C, B	Top ---------) Down
D, B	Left ---------) Right
. E, C	Left ---------) Right

FIGURE 16.6 Generation of a description for a misaligned input graphic and reconstruction of the graphic from the description. (a) Input graphic (misaligned); (b) description generated for the graphic shown on (a); (c) reconstructed graphic from the description. (From Bow and El-Masri, 1987.)



Attributes / Primitives	level	S.F.	w/h	center point
Rectangle A	1	1.0	1.0	(430,220)
Rectangle B	1	1.0	0.625	(377,150)
Rectangle C	1	1.0	1.0	(345,180)
⋮	⋮	⋮	⋮	⋮
Rectangle P	1	1.0	1.0	(250,340)
Rectangle Q	1	1.0	1.0	(180,340)

Polyline	Segments	Connection	
		From	To
PL1	S1: (100,180) , (100,160) S2: (100,160) , (190,160)	N	PL2, L33
PL2	S1: (190,160) , (210,160) S2: (210,160) , (210,200)	PL1, L33	L
PL3	S1: (230,200) , (230,160) S2: (230,160) , (270,160) S3: (270,160) , (270,145)	L	PL4, L38
.PL4	S1: (270,145) , (330,145) S2: (330,145) , (330,120)	PL3, L35	E

Line	Head Tail	Connection	
		From	To
L1	(185,420) , (185,360)	G	Q
L2	(180,410) , (180,410)	Space	L4, L3
L3	(190,410) , (190,380)	L2, L4	Q
⋮	⋮ ⋮	⋮	⋮
L40	(430,280) , (430,240)	L30, L31	A

Polygons	Number of Segments	Center
POL1	6	(178, 41)

Primitives	Spatial Relations
E	Encloses J
F	Encloses I
G	Encloses H
C, E	Top ---------) Down
N, L, D, K, A	Left ---------) Right
C, B	Left ---------) Right
F, E	Left ---------) Right
Q, P, O, H	Left ---------) Right

FIGURE 16.7 Generation of a description for a misaligned input graphic and reconstruction of the graphic from the descnption. (a) Input graphic (misaligned); (b) description generated for the graphic shown on (a); (c) reconstructed graphic from the description. (From Bow and El-Masri, 1987.)

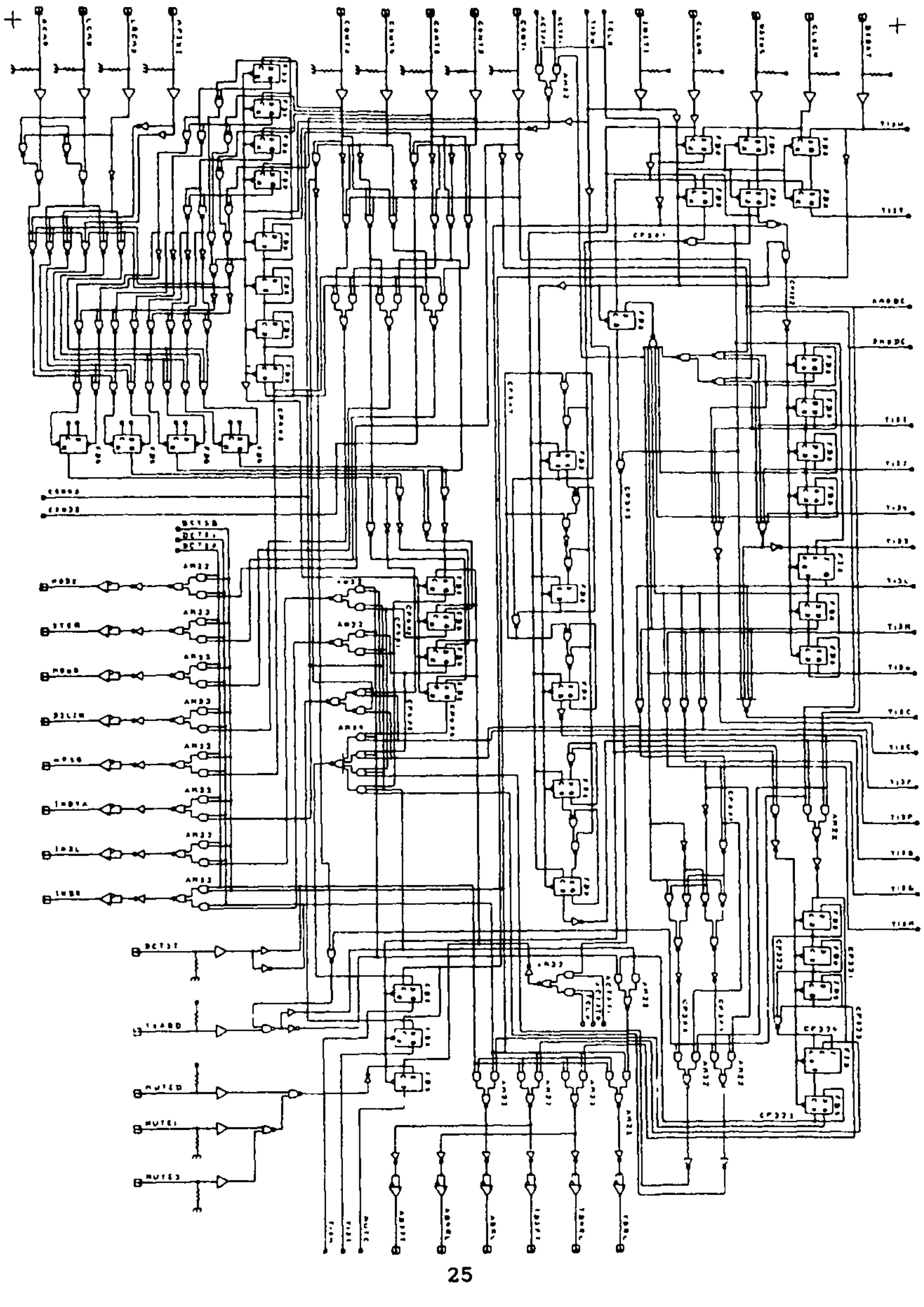


FIGURE 16.8 Logic circuit diagram.

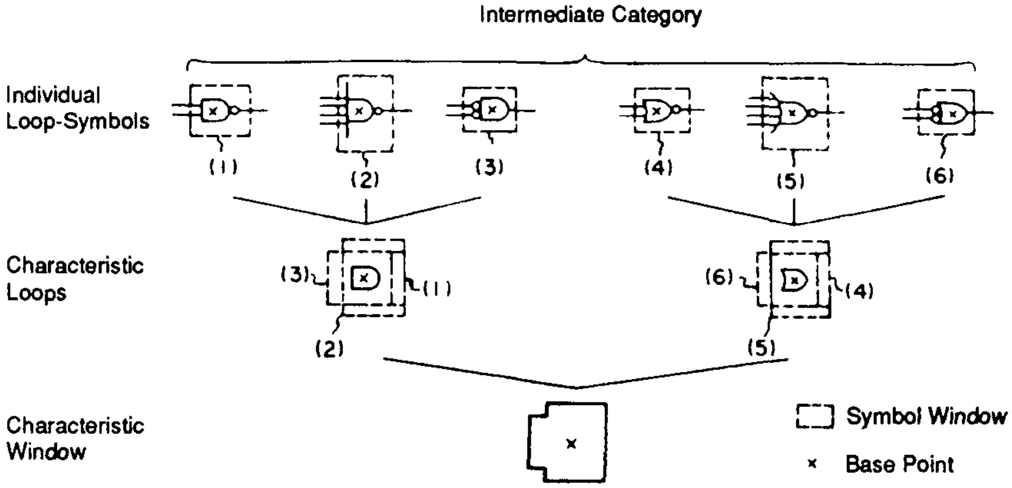


FIGURE 16.9 Concepts used in calculation of MRA. (From Okazaki et al., 1988.)

the drawing for later identification. Of course, this region could have four orientations and their mirror images. Logic symbols conventionally appearing in drawings can be grouped into a certain number of intermediate categories, each of which is represented by a characteristic window. This is the minimum region that can include all the components of the symbols within that intermediate category. The step-by-step determination of the characteristic window is shown in Figure 16.9 and is self-explanatory. Several features need to be extracted from the candidate loop: loop area, number of occurrences of each of the four mask patterns which approximate the length of the corresponding oblique line (see Figure 16.10), and the x and y lengths of the rectangle windows, which just fit the candidate.

Symbol Identification

After the MRA has been isolated, identifying the exact symbol type will proceed. Template matching is a simple way to perform this task. Templates are prepared only for the individual primitive loop patterns. Two groups of primitive templates are suggested, one for radical symbols, the other for auxiliary symbols, as shown in Figure 16.11. Take a simple NOR gate, , for illustration; this will be matched by the templates "OR" and "o" as well as with some additional information about their spatial relationships. The first template to be matched depends on the intermediate category. Once this template matching is successful for a radical symbol, the number of auxiliary symbols to be matched and their location are limited. Further matching processing can be guided, thus simplifying the entire process.

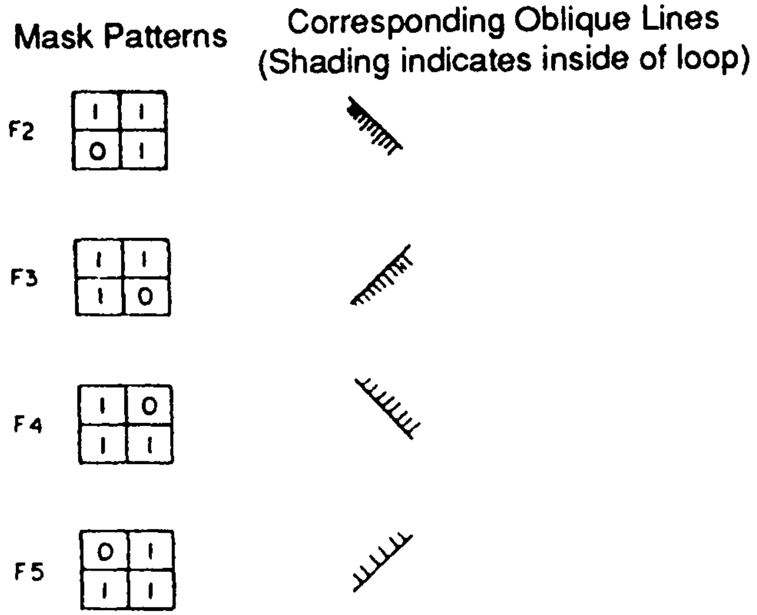

Mask Patterns
Corresponding Oblique Lines
(Shading indicates inside of loop)
F2
1 1
0 1
F3
1 1
1 0
F4
1 0
1 1
F5
0 1
1 1

FIGURE 16.10 Four mask patterns. (From Okazaki et al., 1988.)

The entire operation of the logic circuit diagram reader can thus be described by the recognition flow diagram shown in Figure 16.12. The loop-free symbol and rectangle recognition can be done according to the method described in Section 16.1. After the primitive elements (i.e., symbols, character strings, functional rectangles, and lines) are recognized by the reader, they can be converted into appropriate formats as CAD data input.

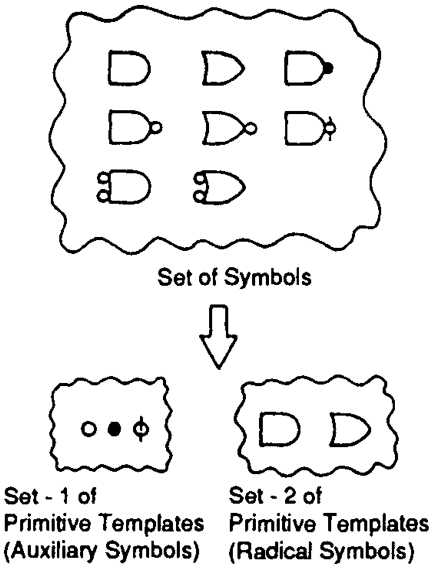


FIGURE 16.11 Primitive template set for matching. (From Okazaki et al., 1988.)

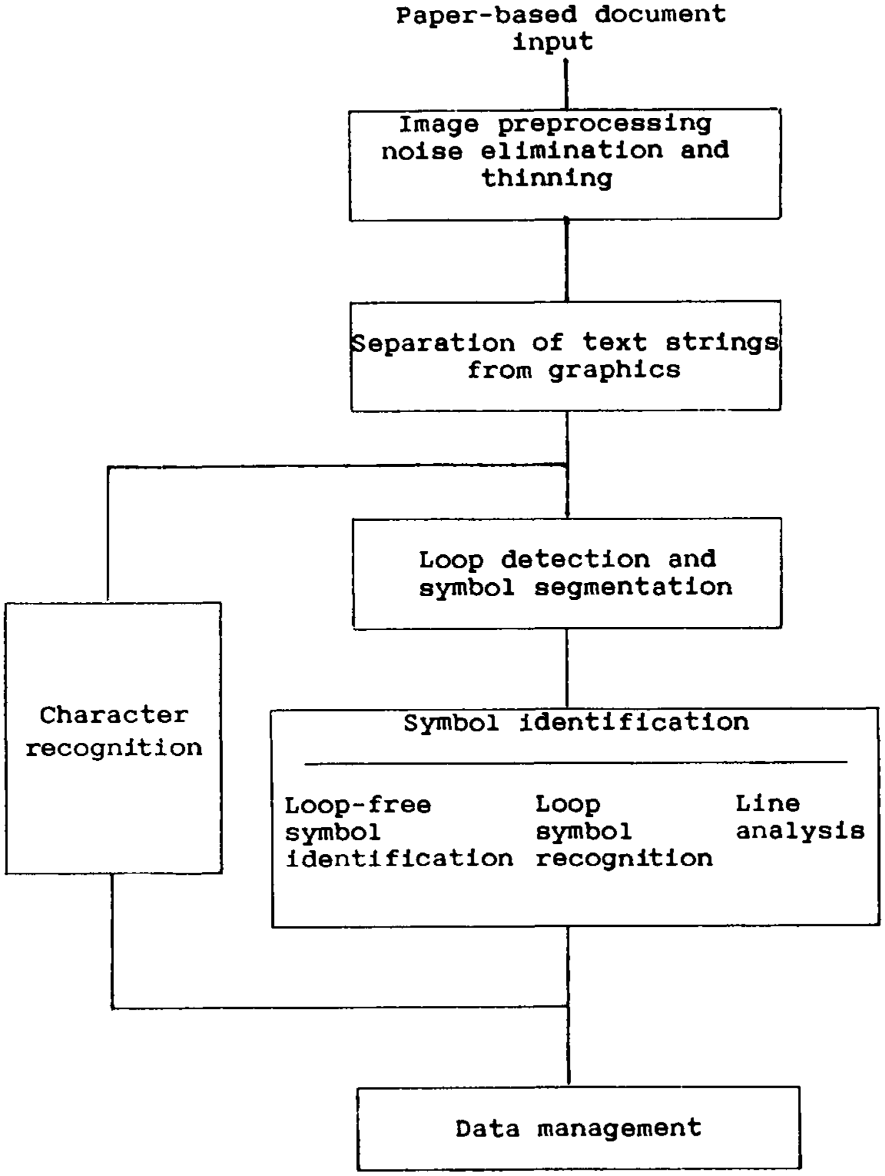

Paper-based document input
Image preprocessing noise elimination and thinning
Separation of text strings from graphics
Loop detection and symbol segmentation
Character recognition
Symbol identification
Loop-free symbol identification
Loop symbol recognition
Line analysis
Data management

FIGURE 16.12 Recognition flow in the logic circuit diagram reader.

16.2 INDUSTRIAL INSPECTION

16.2.1 Automated Detection and Identification of Scratches and Cracks on a Crystal Blank

Automation is no longer imaginary; it now plays an important role in industry. A lot of successful experience has been obtained in modern factories producing steel, automobiles, and the like, where direct human intervention is no longer necessary. In many cases computerized inspection coordinates very well with robotics and has become an important link in the integrated manufactured system. However, we should be aware that there is still a lot of work for human beings, and although it may not be as heavy as that noted above, it is even more tiresome to the human operators. Inspection of defects in crystal blanks in the crystal manufacturing industry is a very good example.

Crystal resonators have been used widely in many applications, from satellite communication to timekeeping in daily life. But unfortunately, due to the inherent brittleness of crystals when thin, quite a number of samples are rejected because of imperfections resulting from grinding, cleaning, and welding. This problem becomes much more serious when attempts are made to raise resonator frequency to 45 MHz or higher.

To assure the quality, to maintain a high productivity rate, to lower the cost per unit, and to achieve even higher resonant frequencies, conscientious pre-inspection of crystal blanks to screen out the imperfect ones will be a very important procedure. This inspection task is still in a very primitive stage and is carried out by a human operator. To sort out the imperfections of submils in width and a few mils in length from a 0.2756-in.-diameter crystal blank with a speed of 3000 pieces in an 8-hour shift (or less than 10 seconds per piece of crystal blank) under strong illumination would be a terrible job. Misclassification due to tired vision or lack of attention due to continuous exposure to strong illumination is understandable and unavoidable. Due to the fast development of image processing, such monotonous, tiresome work can be taken over by an image processing and pattern recognition system from the human operators, leaving them to perform higher-level jobs (e.g., statistical analysis).

Automated inspection of crystal blanks is very challenging because it competes with human operators' inspection speed, and their discriminating capability as to the existence of defects and their categories. That is, it is required that an automated inspection system (AIS) be able to do whatever a human operator can do. Not only that—it is also required that an automated inspection system be able to detect and categorize defects that are not visible to the naked eye even with a magnifier. At present, this situation may not be deemed to be very serious, but it will definitely be so when working toward 45 MHz or an even higher natural resonant frequency—when the crystal blank will be much thinner, and consequently the tolerable width of cracks and scratches will be much smaller.

Detection and Identification of Scratches and Cracks on Unpolished Crystal Blanks

Some investigations on the detection and identification of scratches and cracks on unpolished crystal blanks have been carried out [Bow, Chen, and Newell (1989)]. Figure 16.13 shows a digitized magnified image (×65) of a portion of an

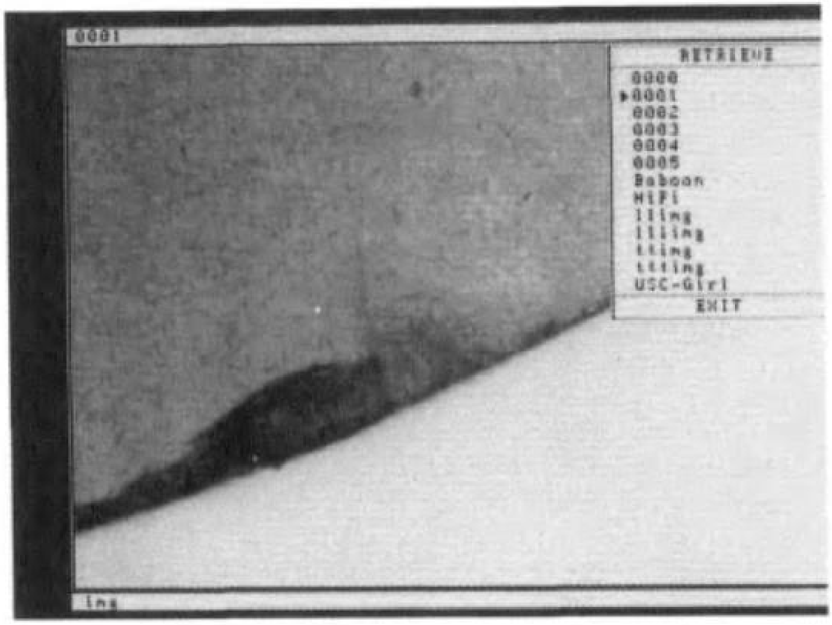


FIGURE 16.13 Magnified digitized image (×65) of an unpolished crystal with a tiny crack and an edge feature. (From Bow et al., 1989.)

unpolished crystal with a tiny crack 0.002 in. wide and 0.005 in. long (pointed to by an arrow in the figure). This is the typical image from which we are to detect cracks and scratches. Undoubtedly, unpolished quartz blanks are translucent and textural in nature. The black region in the picture (called an edge feature) looks very bad, but it is not a problem of concern to us since it does not cause trouble to the crystal. The detrimental part is the very thin line located inside the black region and extending upward, shown by the letter "C." This is what we call a crack. This crack is so thin that it can be seen with the naked eye under a magnifier only when special illumination set at a proper angle is used to make the reflection of light match the observer's line of sight.

Due to the importance and ubiquity of such image data, lots of work have been done to look for models and approaches to their representation and processing [Haralick (1978)]. These approaches are successful in an analysis of multispectral scanner images obtained from aircraft or satellite platforms and microscopic images of cell culture or tissue samples. However, in the detection of surface imperfections, we are confronted with problems that combine texture analysis and image segmentation. The imperfections in quartz blanks are usually very thin. Practically, they break the homogeneity of the textural pattern locally into two or more textural regions. For this reason the textural regions so formed are very similar in most aspects. This makes it extremely difficult to segment these textural regions from one another, but an approach has been proposed [Bow, Chen (1989)]. Difficulties in processing this image come from (1) very small differences between the gray levels of the objects (cracks, scratches, etc., in this case) we are looking for and those of portions of the pixels in the textural background (the difference is as small as three to four gray levels based on 256-gray-level representations with 0 for complete darkness and 255 for complete brightness), and (2) very small differences among the textural structures on the statistical parameters that are to be differentiated.

There is also another feature in our problem—that no a priori information is available regarding partitioning of the component textural structures. As can be seen from Figure 16.13, there are four textural structures on the image:

1. Image background
2. Crystal textural background
3. Crack-affected region
4. Edge features

Among these four textural structures, the most difficult task in processing will be segmentation between textures 2 and 3. The approach suggested is first to separate the image background from the rest of the image. Next, apply to the remaining data the three-step procedure suggested:

1. Segment the edge feature plus the crack-affected region from the crystal background.
2. Segment the crack-affected region from the crystal textural background.
3. Delineate the boundary between the edge feature and the crack-affected region.

A histogram of the image shown in Figure 16.14 shows that the pixels within the crack region are completely mixed up with those within the crystal textural background. Their gray levels are so close that they might not be differentiated from each other if a simple gray-level thresholding method were used. Our approach then is first to segment the edge feature plus the crack-affected region from the crystal background. Unavoidably, artifact noise will be introduced, as shown in Figure 16.15. Morphological erosion, dilation, and coarse noise elimination follow, and the image shown in Figure 16.16 results.

Superimposing the original image on the image shown in Figure 16.16 yields the image shown in Figure 16.17, from which we can see that the four textural structures—image background, crystal textual background, crack-affected region, and edge feature—are clearly separated. From Figure 16.17 we can see a very thin line (crack) extending upward between the edge feature and the crack-affected region. This is the crack. Figure 16.18 is the image shown in Figure 16.17 but with the textural background marked by a gray level other than 255. In this figure (Figure 16.18) the edge feature and the (crack-affected region appear with different gray levels. The edge feature and crack on the quartz blank are eventually located, which is shown on Figure 16.19, and identified as a crack

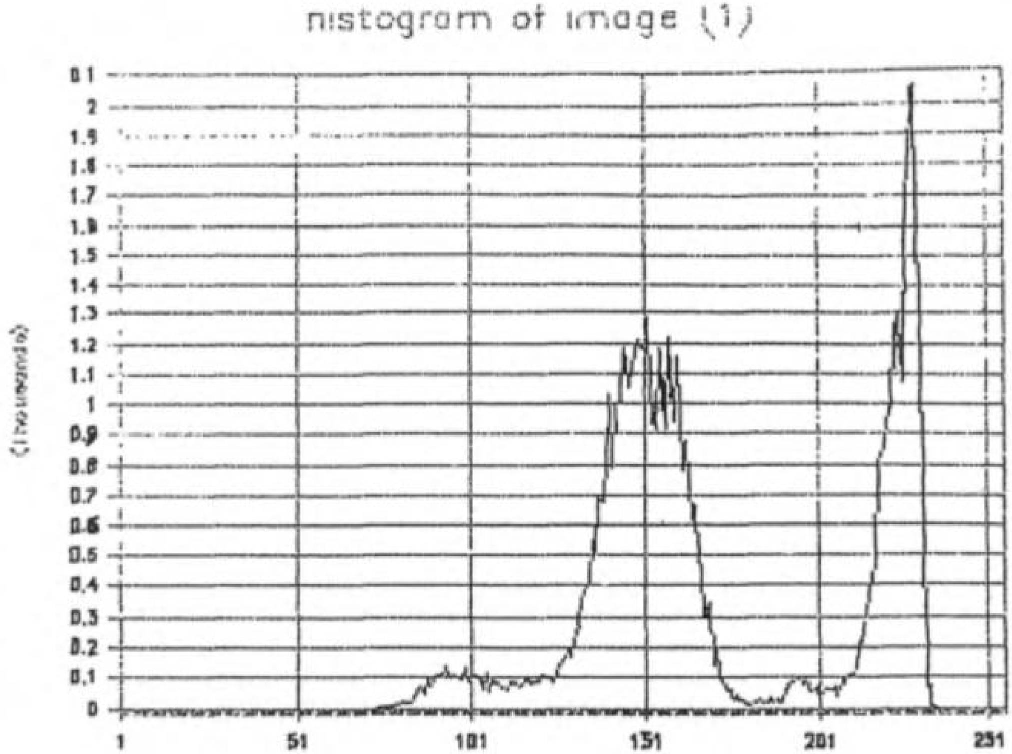


FIGURE 16.14 Histogram of the sample image shown in Figure 16.13. (From Bow et al., 1989.)

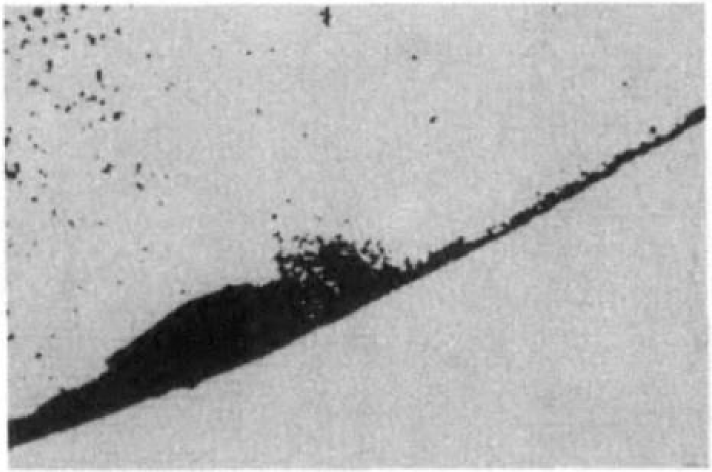


FIGURE 16.15 Image obtained after the edge feature plus the crack-affected region have been segmented from the crystal background. (From Bow et al., 1989.)

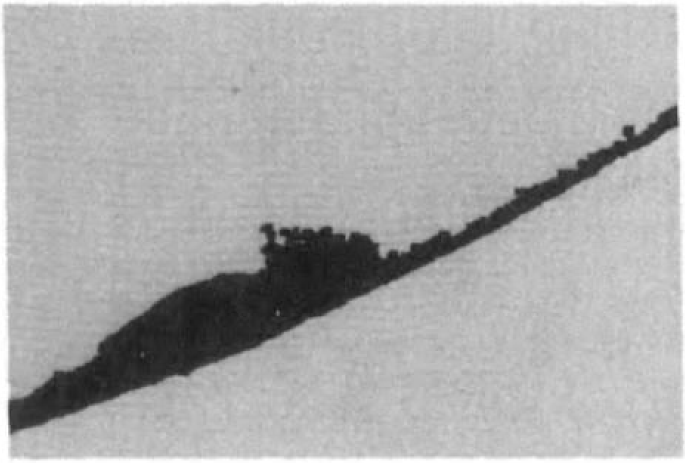


FIGURE 16.16 Image obtained after morphological processing and coarse noise elimination on the image shown in Figure 16.15. (From Bow et al., 1989.)

by the computer. This approach has been used on many other crystal images. In each figure, the original magnified digitized image is shown in (a), and the processed image is shown in (b), where the imperfection was delineated. "F" indicates an imperfection; "P" indicates a good product, and "b" and "t" represent the image background and crystal texture, respectively.

Detection and Identification of Cracks and Scratches on a Polished Crystal Blank

Defect model on a polished crystal blank. There will be no difficulty in the detection of cracks and scratches of discernible size nor in their screening,

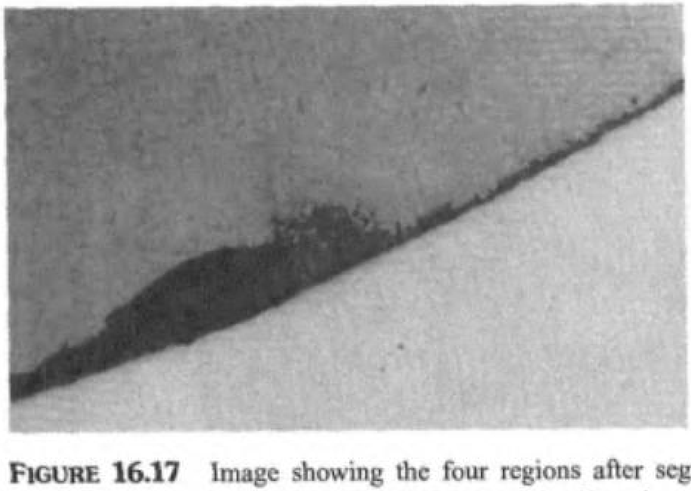


FIGURE 16.17 Image showing the four regions after segmentation. (a) Image background; (b) crystal textual background; (c) crack-affected region; (d) edge feature. (From Bow et al., 1989.)

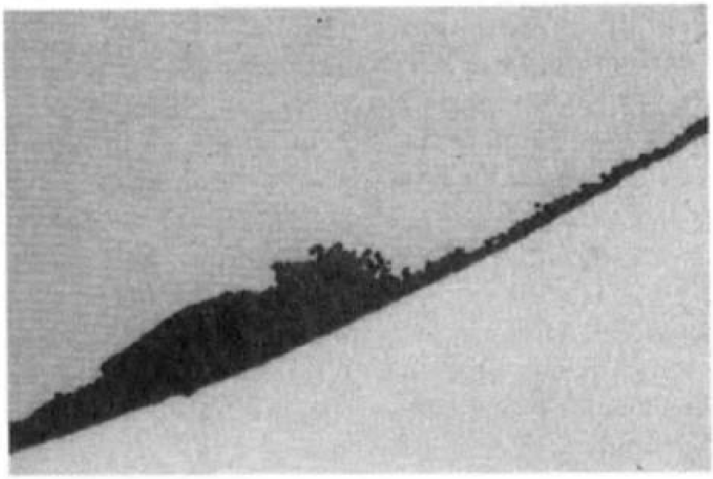


FIGURE 16.18 The image shown in Figure 16.17 but with the textural background marked with a gray level other than 255. (From Bow et al., 1989.)

since they will obviously be rejected despite which category, crack or scratch, they belong to. However, there will be differences in dealing with extraordinarily fine defects. Cracks, no matter how fine they are or where they are located, are always unacceptable. However, scratches of the same size may be accepted. Sometimes, we are inclined to keep them to avoid wasting material or labor time. We are not concerned, for example, about tiny scratches that are close to the edge. But for a scratch located somewhere in the center of a blank, whether or not it is accepted depends on its depth. How to decide whether it is acceptable or unacceptable is complicated. However, on the eve of fifth-generation computers, many empirical judgments are involved in the quantification criterion.

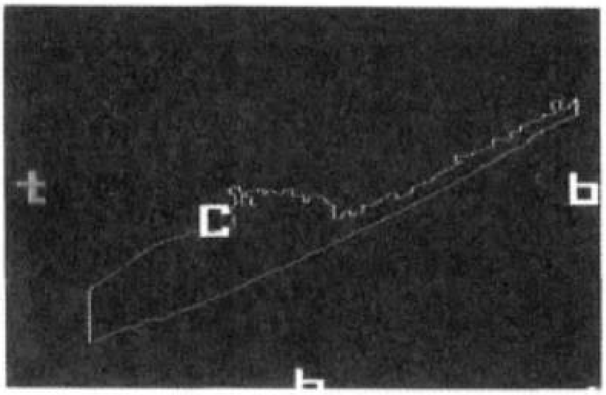


FIGURE 16.19 Locating and identifying the defect on a crystal. "C" represents a crack; "b" and "t" represent the image background and crystal textural background, respectively.

As mentioned above, what we are interested in is not the larger cracks and/or scratches, since they will obviously be rejected and will easily be detected and screened, especially when a computer is used as an aid. Our interest focuses on effective methods for detecting and categorizing extraordinarily fine defects—more specifically, into which category such defects fall. That is, an objective criterion should be established for their classification so that they can be handled properly. This should contribute positively to the productivity rate and quality assurance in quartz crystal manufacture.

It is therefore necessary to establish a criterion to define fine scratches and cracks precisely based on their morphology, and to develop a digital image processing and pattern recognition algorithm. Resolving such a difficult task, previously thought achievable only by an human operator, will make it possible for the resonator to advance toward even higher frequencies. From the data collected from a large number of crystal samples known to contain cracks and/or scratches, two phenomena were observed. Based on the observations, two approaches have been suggested to differentiate the two inherently different defects based on their magnifier images.

In deciding what approach should be used, several considerations were followed: that innovation in the mechanical equipment should be kept to a minimum, and that the computerized system should be made as compact, and thus as inexpensive, as possible. It follows that:

1. Feature measurements should be kept few in number and should be direct (i.e., directly obtainable from the picture element).
2. The measure should attempt to determine the inherent characteristics in crystal and should not require training samples.
3. The algorithm should be fast enough to compete with a human operator.
4. The system should be able to detect submil defects and to differentiate scratches from cracks. The detectability should be comparable to that of a highly experienced operator when he or she is functioning at the highest level.

One of the approaches proposed by [Bow, Chen, and Chen (1991)] is based on optical observation. Figure 16.20a and b show microscopic images of a crystal blank with scratch defects obtained with light incident from the scratch side and other side of the crystal blank, respectively. Figure 16.21 is the same as Figure 16.20 but for a crystal blank with a crack defect. Comparing Figure 16.21a with 16.21b shows that the widths and cross-sectional areas of the crack on the images taken from either side are the same.

By contrast, from Figure 16.20 it is clear that the width of the scratch is wider when observed with light incident from the scratch side. This is shown in Figure 16.22, where the refraction introduced by the crystal material is included.

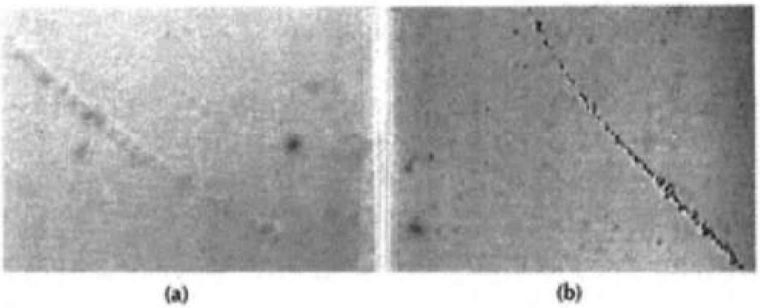


FIGURE 16.20 Microscopic image of a crystal blank with a scratch defect. (a) Image taken with light incident from the scratch side; (b) image taken with light incident from the other side of the crystal blank. (From Bow et al., 1990.)

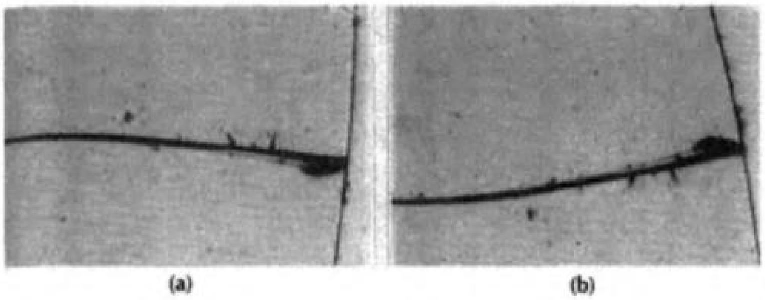


FIGURE 16.21 Microscopic image of a crystal blank with a crack defect. (a) Image taken with light incident from the crack side; (b) image taken with light incident from the other side of the crystal blank.

In Figure 16.22, the apparent widths of the scratch observed when the illumination is from the scratch side or from the opposite side are shown. It is clear that the differences in apparent widths come primarily from refraction of the crystal. This is an effective method to use to differentiate a scratch or a crack. It can also be used to compute the depth of the scratch indirectly from the thickness of the crystal, index of refraction of the crystal, which is 1.458, and the apparent widths of the scratch measured from the images when viewed from the two different

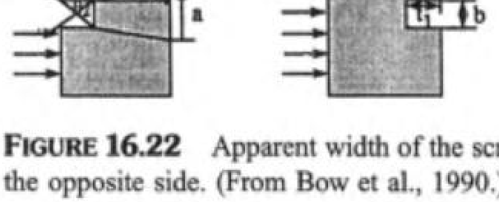


FIGURE 16.22 Apparent width of the scratch as observed from the scratch side or from the opposite side. (From Bow et al., 1990.)

sides. The only argument against this approach is that more work will be involved in the design of a mechanism for obtaining these two magnified images with light illuminated from one side in one case and from the other side in the other case.

Due to the negative effect of the first approach, as mentioned above, another approach has been proposed in which illumination comes from only one side. Of course, we have no idea at all as to which side the scratch actually lies on, so all we try to do is to keep away from the variations introduced by this factor. We concentrate on the morphology of the scratch or crack. From a large number of enlarged images of quartz samples with defects, we have found that scratches differ in morphology from cracks. Since cracks run all the way through a crystal, a solid boundary will appear in the image, whereas scratches which are superficial and usually caused by nonuniform stress, show on magnified image as intermittently connected pits (some of them light, some of them heavy). Nevertheless, the pits lie on the same line or, in general, lie on the same curve. This valuable information provides us with a basis to establish a criterion for differentiating a scratch from a crack: If the defect detected appears in the form of an obviously clear-cut continuous line (or broken lines), it is identified as a crack. If it appears as intermittently line-shaped (or in general, curve-shaped) dot clusters, it is a scratch.

Algorithm to identify extraordinarily fine cracks and scratches on a polished crystal blank: Laplacian operation plus zero crossing. Several algorithms have been developed regarding this problem, one of which was found to be very effective. As mentioned earlier, the scratches and cracks we are looking for are so fine that the intensity change was not as sharp as we expected. For this reason, instead of using a gradient-based method, we used the local extremum of $f'(x)$ as the criterion to detect the boundary of the defects. That is, the method of Laplace plus zero crossing was adopted in our approach. Because this operation is sensitive to noise, reduction of the resulting sizable artifact noise is desirable prior to edge detection. The laplacian operator is given by

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{16.1}$$

and its discrete implementation can be reduced to the form of a convolution operation as

$$\begin{aligned}\nabla^2 f(x,y) &= f_{xx}(n_1,n_2) + f_{yy}(n_1,n_2)\\ &= f(n_1+1,n_2) + f(n_1-1,n_2) + f(n_1,n_2+1)\\ &\quad + f(n_1,n_2-1) - 4f(n_1,n_2)\end{aligned} \tag{16.2}$$

where $f_{xx}(n_1, n_2)$ and $f_{yy}(n_1, n_2)$ represent, respectively, the second derivatives with respect to x and y and $f(n_1, n_2)$, $f(n_1 + 1, n_2)$, $f(n_1 - 1, n_2)$, $f(n_1, n_2 + 1)$, and $f(n_1, n_2 - 1)$ are the gray-level intensities of the pixel concerned and its four neighors. Figures 16.23 and 16.24 show the microscopic images of two polished crystal blanks, one with a scratch and the other with a crack. Figures 16.23b and 16.24b show the images obtained after laplacian and zero-crossing processing, respectively.

Due to the extremely slight difference in gray level in the defect area on the polished blank, lots of artifact noise after processing is unavoidable. Elimination of this artifact noise will be the next important step.

The laplacian-based methods discussed above frequently generate many undesirable "false" edge elements, especially when the local variance is small. Consider a special case where a uniform background region is assumed: that is, $f(n_1, n_2)$ is constant over that region. Consequently, $\nabla^2 f(n_1, n_2)$ equals zero and no edges will be detected. Any small perturbation of $f(n_1, n_2)$ is likely to generate false edge elements. The method selected to remove these false edge elements is to set up a threshold such that the local variance at the point should be sufficiently large. See Figure 16.25 for a schematic diagram of the process. The local variance $\sigma_f^2(n_1, n_2)$ at the pixel (n_1, n_2) can be estimated by

$$\sigma_f^2(n_1, n_2) = \frac{1}{(2M + 1)^2} \sum_{K_1=n_1-M}^{n_1+M} \sum_{K_2=n_2-M}^{n_2+M} [f(K_1, K_2) - m_f(K_1, K_2)]^2 \tag{16.3}$$

where

$$m_f(n_1, n_2) = \frac{1}{(2M + 1)^2} \sum_{K_1=n_1-M}^{n_1+M} \sum_{K_2=n_2-M}^{n_2+M} f(k_1, k_2) \tag{16.4}$$

with M typically chosen around 2. Since $\sigma_f^2(n_1, n_2)$ is compared with a threshold, the scaling factor $1/(2M + 1)^2$ in (16.3) can be removed from the expression. In addition, computation of the local variance σ_f^2 needs to be done only on the pixels (n_1, n_2), the Laplacian of which cross zero. Figures 16.23c and 16.24c show the results after the application of local variance thresholding of the two images.

After conducting the processing mentioned in the preceding two sections, defects in a crystal blank are detected. Basing the detection on the inherent characteristics of cracks (that they run all the way through the crystal cross section), it is not too difficult to sort out the cracks. But for scratches, one more processing step should be done to make sure of their existence. We have to ascertain all the discrete dots or dot clusters lying on the same straight line or lying on the same arc with a relatively large radius of curvature. Modified Hough transform is used for this purpose. The results are shown in Figure 16.23d, where the existing scratch is delineated with its category identified.

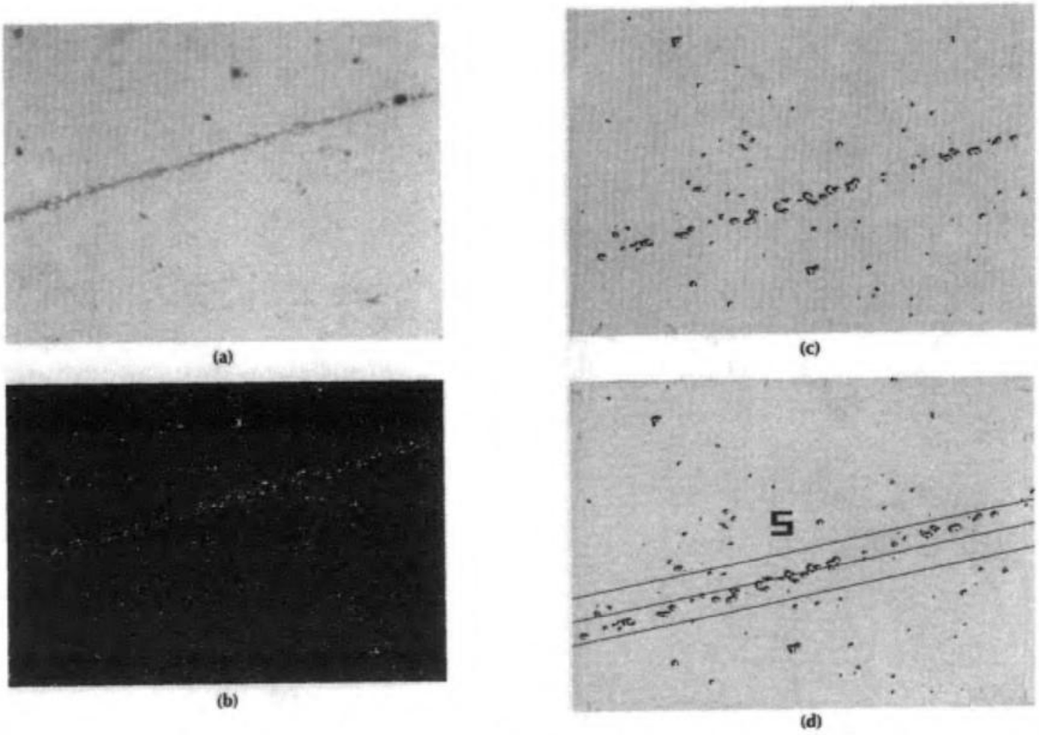


FIGURE 16.23 Detection and identification of a scratch on a polished crystal blank. (a) Microscopic image of a crystal blank with a scratch; (b) after processed by the approach of Laplace plus zero crossing; (c) after elimination of artifact noise; (d) delineation of the existing defect with its category identified. (From Bow and Chen, 1990.)

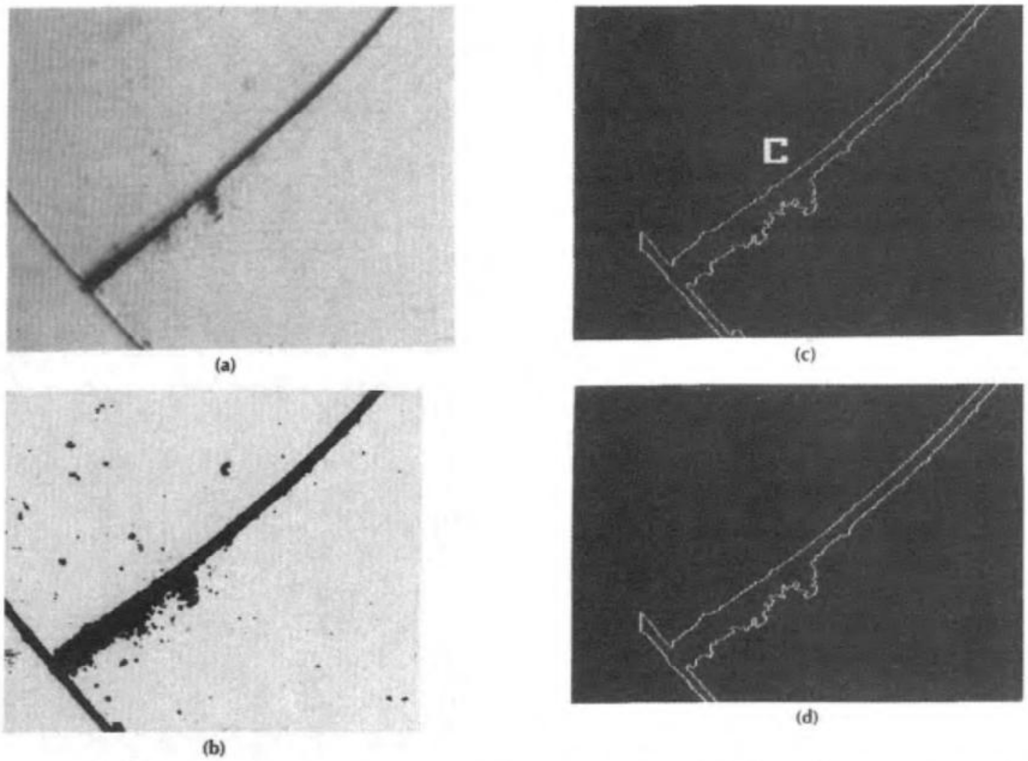


FIGURE 16.24 Detection and identification of a crack on a polished crystal blank. (a) Microscopic image of a crystal blank with a crack; (b) after processed by the approach of Laplace plus zero crossing; (c) after elimination of artifact noise; (d) delineation of the existing defect with its category identified. (From Bow and Chen, 1990.)

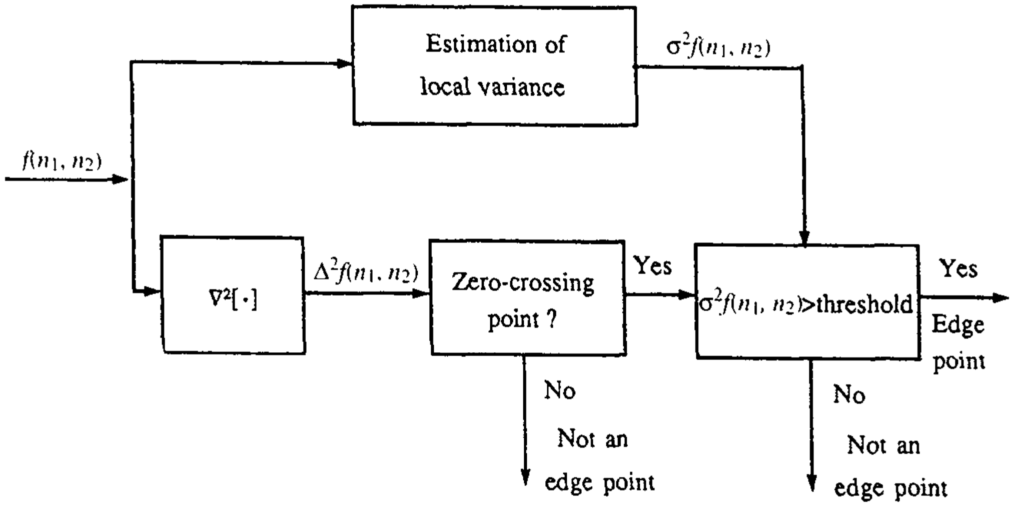


FIGURE 16.25 Laplacian-based edge detection taking the local variance into consideration.

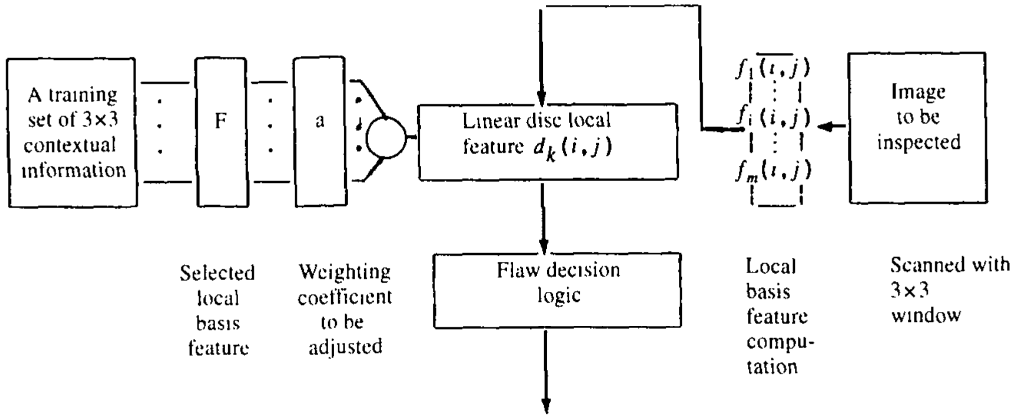
16.2.2 Automatic Inspection of Industrial Parts with X-rays

In this section an example using x-ray images for the automatic detection of flaws in cast aluminum wheels is given. Flaws such as cavities due to gas bubbles or shrink holes down to a size of 1 mm are to be detected for quality assurance. Three positions in a wheel, one for the hub, one for the spokes, and one for the road wheel, are to be checked. The automatic x-ray inspection system (see Figure 16.26) consists of three principal parts: (1) a precision object-handling mechanism, (2) an x-ray tube and image-data acquisition system, and (3) an image processing system. The image is 512 x 512 pixels in size.

First, we have to have some a priori information about the flaws. Voids frequently appear in the cast aluminum wheels. In x-ray images, voids show up as bright regions with respect to their neighborhood. The majority of flaws in aluminum casting look like isolated, roundish bubbles, which frequently are grouped together in clusters. When a void exists, the thickness of the object is locally reduced, and consequently, the attenuation of the x-ray is smaller and follows the exponential law

$$dS(x) = \mu_{\text{eff}}(x) S_0 \exp(-\mu_{\text{eff}}(x) x)\, dx \tag{16.5}$$

where $S(x)$ is the transmitted intensity, which is a function of the thickness x, and S_0 is the initial intensity, and μ_{eff} is the effective attenuation coefficient. This characteristic makes it difficult to detect flaws in regions where thick material must be penetrated. Because of the quantum nature of radiation, x-ray imaging is inherently noisy, giving rise to a signal-dependent noise component in the x-ray


A training set of 3×3 contextual information
F
a
Linear disc local feature $d_k(i,j)$
$f_1(i,j)$
$f_i(i,j)$
$f_m(i,j)$
Image to be inspected
Selected local basis feature
Weighting coefficient to be adjusted
Flaw decision logic
Local basis feature compu-tation
Scanned with 3×3 window

$$F = \begin{bmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ \cdot \\ f_m \end{bmatrix} \qquad a = \begin{bmatrix} a_{k0} \\ a_{k1} \\ \cdot \\ \cdot \\ \cdot \\ a_{km} \end{bmatrix} \qquad d_k(i,j) = a_{k0} + a_{k1}f_1 + a_{k2}f_2 + \cdots + a_{km}f_m$$

FIGURE 16.26 Schematic diagram of an automated x-ray inspection system for the detection of flaws in an aluminum casting.

intensity. Hence averaging over several frames is performed before the segmentation process.

If we select some local features that not only characterize the pixels themselves, but include local contextual information, we can assume that the image pixels of a segment form a cluster in the feature space. In this sense the segmentation turns out to be a pixel classification problem. Let us formulate a discriminant function $d_k(i,j)$, $k = 1, 2, \ldots, K$, such that if

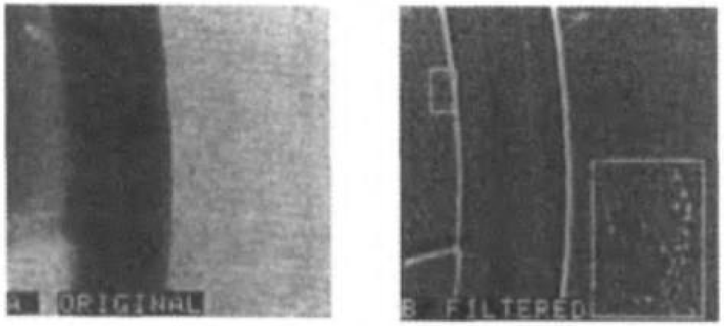
$$d_k(i,j) > \tau \qquad i,j = 1, 2, \ldots, N \tag{16.6}$$

then

$$P(i,j) \in \omega_k \qquad k = 1, 2, \ldots, K$$

with ω_k denoting class k. For the detection of cavities in aluminum castings discussed here, selection of these discriminant features has to be tailored to the problem at hand. The effectiveness of each feature should be evaluated to determine whether it should be included in the segmentation subsystem.

Features from linear filtering operations after enhancement of the signal for flaws while suppressing regular features of the projection image can be good candidates. Features from nonlinear filtering operations (e.g., median filters) can also be used. We can also define a local model to characterize the gray-level variations of the flaws and use the parameters of the model as features. After the



(a) (b)

FIGURE 16.27 (a) Original ımage of a wheel; (b) image filtered with DOG (difference of gaussians) mask. (From Boerner and Stretcker, 1988.)

features are selected, we can formulate a polynomial classifier for the flaw detection problem as follows:

$$d_k(i,j) = a_{k0} + a_{k1} f_1(i,j) + a_{k2} f_2(i,j) + \cdots + a_{km} f_m(i,j) \tag{16.7}$$

where

$d_k(i,j)$ = discriminant functions (features) for class k at location $P(i,j)$

$k = 1, 2, \ldots, K$

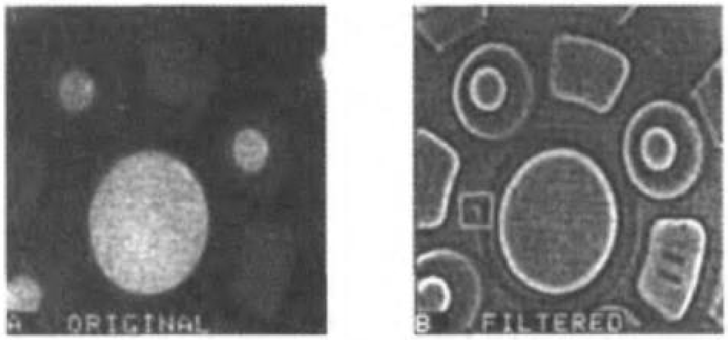
$$d_k(i,j) = \begin{cases} 1 & \text{if } P(i,j) \in \omega_k \\ 0 & \text{otherwise} \end{cases}$$

$f_m(i,j)$ = mth basis feature at location $P(i,j)$ $\qquad m = 1, 2, \ldots, M$

a_{km} = coefficients in the discriminant functions $d_k(i,j)$

$K = 1, 2, \ldots, K;\ m = 1, 2, \ldots, M$

The parameters $a_{k0}, a_{k1}, \ldots, a_{km}$ can be determined by means of a training set of pixels which are known to belong to category ω_k. Figures 16.27 and 16.28 show



(a) (b)

FIGURE 16.28 (a) Original image (x-ray projection of the region around the hub); (b) image filtered with DOG mask. (From Boerner et al., 1988.)

some results obtained with the algorithm described for flaw detection through x-ray imaging of an aluminum cast wheel and hub.

16.3 REMOTE SENSING APPLICATIONS

16.3.1 Autonomous Control of Image Sensor for Optimal Acquisition of Ground Information for Dynamic Analysis

Through constant improvements over the past 30 years, use of an image sensor has been successful in the detection and conversion of low-light-level signals. Nevertheless, human users from various branches of science and technology look forward to having an intelligent sensor that can adjust itself to optimal working conditions. The information base on which such an adjustment is made will be that of previous image segments acquired.

In this section we discuss a very specific problem, optimal acquisition of ground information for dynamic analysis. However, it will not be difficult to see that many problems similar to this (e.g., on-board data preediting) will also be realizable. We focus on an algorithm that will permit us to greatly increase the scanning range of a stripmap acquisition system without modifying its existing structure. This problem originates from the following facts. First, it is agreed that it is very effective and also very beneficial and favorable to acquire ground information from a satellite for either military or civilian purposes. However, due to the fixed orbit of the satellite and the fixed mode of sensor scanning, the way in which the satellite acquires ground information is in the form of a swath, as shown in Figures 16.29 and 16.30. It is known that two consecutive swaths of information scanned are not contiguous geographically. In addition, two geographically contiguous swaths are scanned at times that differ by several days. Very frequently, the part of the target area of greatest interest falls either to the left or right outside the current swath. Postflight matching of two or three swaths is thus unavoidable for target interpretation, and therefore on-line processing will not be possible. This will be all right (very inefficient, though), when dealing only with a static target, but the situation will become very serious if the information sought is for the dynamic analysis of strategic military deployment, for example. Even when monitoring a slowly changing flood, information obtained in this way would be of little use.

A desire has thus arisen to enlarge the viewing range of the scanner so that we can acquire in a single flight all the ground information of interest now located across two or three swaths. This would not only permit on-time acquisition and on-line processing of the relevant time-varying scene information, but would save a lot of postflight ground processing.

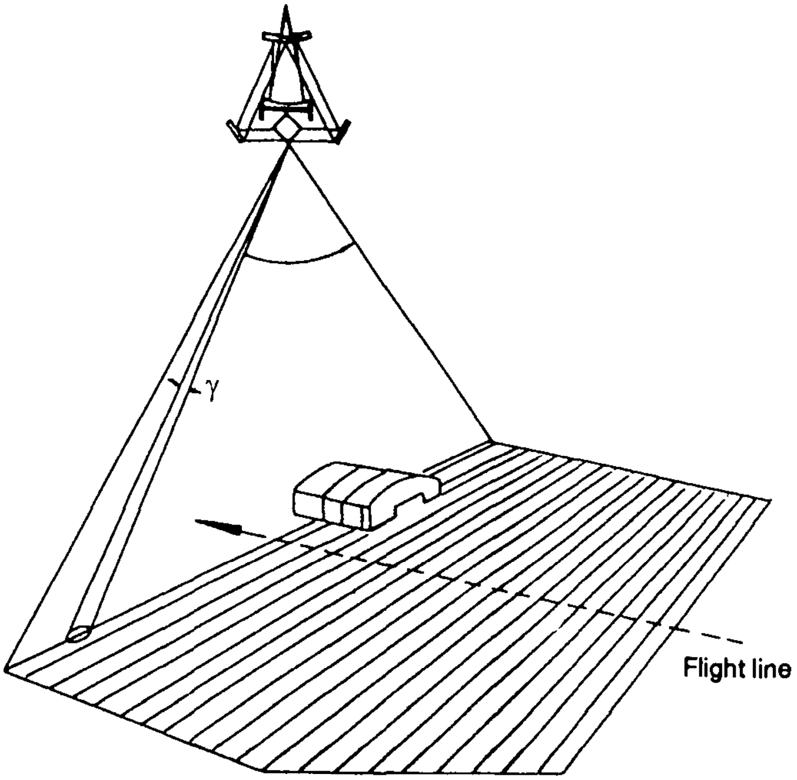


FIGURE 16.29 Mode of data acquisition by stripmap scanner. (From Bow, 1986.)

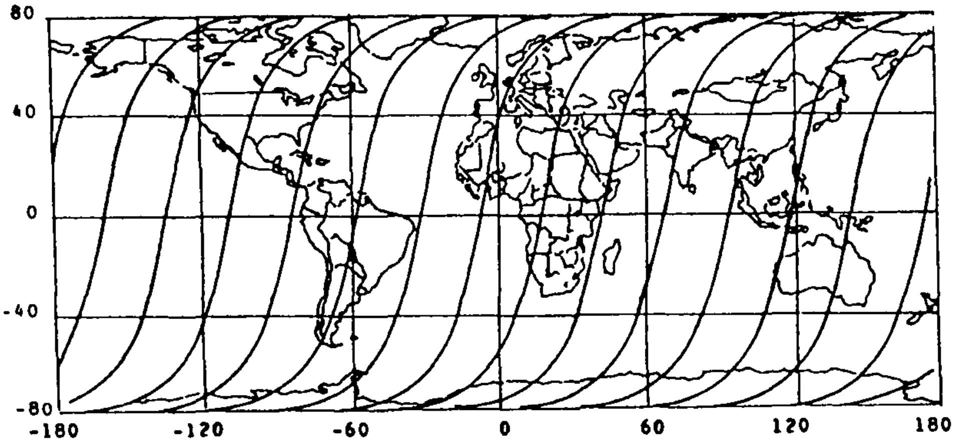


FIGURE 16.30 Orbit of LANDSAT D. (From Bow, 1986.)

The range of scanning of a stripmap sensor is highly limited by the instrument design. In the author's opinion, implementing the sensor with artificial intelligence (AI) will be a prospective solution to improving the overall performance of the sensor system. Based on the on-line processing of image segment data acquired from previous scans, the viewing angle of the sensor is to be adjusted automatically to track the target without changing the mainframe. This is to simulate the tracking action of the human eyeball and to enlarge the sensing scanning range. According to Bow (1986) and Bow, Yu, and Zhou (1986), the scanning range can be enlarged to two to three times that of the target's design value.

Autonomous eyeball-like tracking works on the traditional Chinese principle that if you want to get hold of something, you have to sacrifice something else. In so doing, you will obtain as much useful information as possible. What the useful information means depends on the problem being studied. The system discussed here will be something like that shown on Figure 16.31. Such an AI-implemented range-enlargement sensing system should be able to grab and recognize the object of interest, predict and track its positional change, and control the viewing angle of the sensor ahead of time. Association of the pattern recognition technique with the spectral characteristics of objects forms the kernel of this intelligent system. Interested objects can be detected in the spectral band specified. Tracking on the target can then be implemented by successively positioning the sensor at the center of the centroid C_i.

Four sets of measurements can be obtained at a certain interval of time:

$$\begin{aligned}
\mathrm{SN}_i &= \mathrm{MSN}_i + \mathrm{ESN}_i \\
\mathrm{ML}_i &= \mathrm{MML}_i + \mathrm{EML}_i \\
\mathrm{LM}_i &= \mathrm{MLM}_i + \mathrm{ELM}_i \\
\mathrm{RM}_i &= \mathrm{MRM}_i + \mathrm{ERM}_i \qquad i = 0, 1, \ldots, N-1
\end{aligned} \tag{16.8}$$

where

SN_i = number of target pixels obtained during the ith scan

ML_i = centroid of the target area

$$= \sum_j \frac{j\delta(j)}{\mathrm{SN}_i} \quad \text{and}$$

$$\delta(j) = \begin{cases} 1 & \text{target} \\ 0 & \text{otherwise} \end{cases}$$

LM_i = location of the leftmost pixel of the target area of scan i

RM_i = location of the rightmost pixel of the target area of scan i

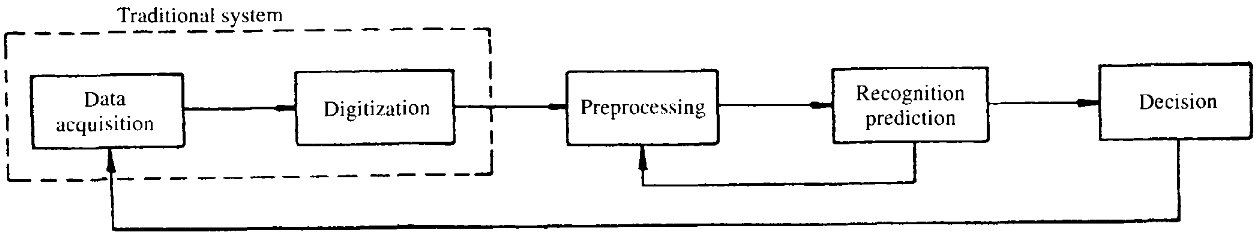


FIGURE 16.31 Intelligent data acquisition system. (From Bow, 1986.)

MSN_i, MML_i, MLM_i, and MRM_i are referred to as slowly varying components, while ESN_i, EML_i, and ELM_i, ERM_i are referred to as their random disturbance with zero means. Equation (16.8) can be generalized as

$$f_i = m_i + e_i \tag{16.10}$$

where m_i represents a varying trend signifying that the target area is going to expand, contract, or remain unchanged. It also shows whether the target area is expected to shift leftward or rightward. e_i is the disturbance of f_i, from which we can differentiate whether it is a random disturbance or spots of comparative significance.

$\{m_i\}$ can be approximated by inference with $g(t) = \sum_j a_j b_j(t)$, $j = 1, 2, \ldots, K$, to match $\{f_i\}$; that is,

$$\begin{aligned} m_i &= g(t_i) \\ e_i &= f_i - g(t_i) \qquad i = 0, 1, 2, \ldots, N-1 \end{aligned} \tag{16.11}$$

and $K \ll N$.

If we let $X_{ij} = b_j(t_i)$, where $i = 0, 1, 2, \ldots, N-1$ and $j = 1, 2, \ldots, K$, this turns out to be a simple model for linear inference. In matrix form,

$$F = XA + E \tag{16.12}$$

and

$$A = \begin{vmatrix} a_1 \\ a_2 \\ \vdots \\ a_K \end{vmatrix} \qquad F = \begin{vmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{vmatrix} \qquad E = \begin{vmatrix} e_0 \\ e_1 \\ \vdots \\ e_{N-1} \end{vmatrix}$$

$$X = \begin{vmatrix} X_{01} & X_{02} & \cdots & X_{0K} \\ X_{11} & & \cdots & X_{1K} \\ \vdots & & & \\ X_{N-1,1} & & \cdots & X_{N-1,K} \end{vmatrix} \tag{16.13}$$

A can be evaluated by

$$\frac{\partial}{\partial a_j}(F - XA)^T(F - XA) = 0 \qquad j = 1, 2, \ldots, K \tag{16.14}$$

Taking into consideration the property of orthogonality, we obtain

$$\hat{A} = (X^T X)^{-1} X^T F = X^T F \tag{16.15}$$

and the estimated value of $MF = (m_0, m_1, \ldots, m_{N-1})^T$ is then

$$M\hat{F} = X\hat{A} = XX^T F \tag{16.16}$$

Inference of the target area variation (e.g., the evaluation of $\{\mathrm{LM}_i\}$ and $\{\mathrm{RM}_i\}$ when $i > N - 1$) can be approximated by

$$\begin{aligned} \mathrm{MLM}_p &= \mathrm{MLM}_{N-1} + \mathrm{GLM} \cdot (P - N + 1) \\ \mathrm{MRM}_p &= \mathrm{MRM}_{N-1} + \mathrm{GRM} \cdot (P - N + 1) \end{aligned} \tag{16.17}$$

where GLM and GRM are the rate of change of $\{\mathrm{MLM}_i\}$ and $\{\mathrm{MRM}_i\}$, respectively, at $i = N - 1$. As can be seen from the expressions above, the larger the P, the less accurate the approximation will be; and therefore P should be restricted to a certain value. The approximation is acceptable if both GLM and GRM remain greater (or less) than zero during the interval $(N - 1, P)$. The range within which both GLM and GRM remain greater (or less) than zero is

$$P - N + 1 \leqslant \frac{N}{2} \tag{16.18}$$

or

$$P \leqslant \frac{3N}{2} - 1 \tag{16.19}$$

A simulation experiment has been conducted to evaluate the realizability of this approach. Thematic mapper (TM) data obtained from LANDSAT D on the Susquehanna River in the state of Pennsylvania and on the Yangtze River (the longest river in China) were taken for the experiment. Figures 16.32a and 16.33a

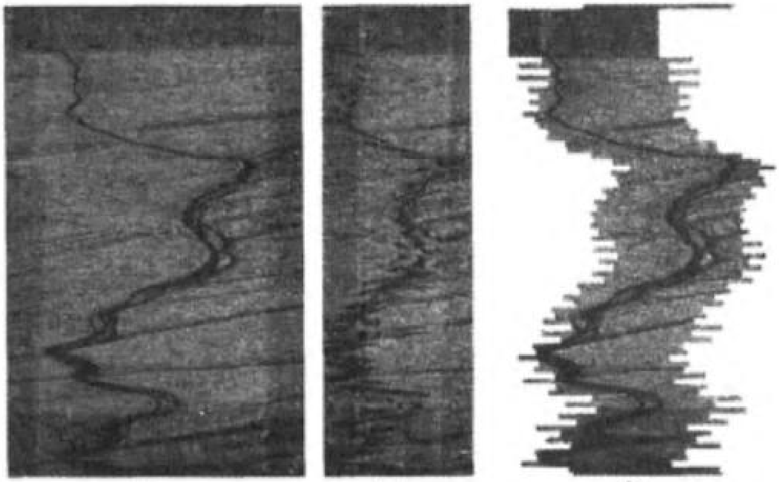


FIGURE 16.32 Simulation experiment and results on Susquehanna River. (From Bow et al., 1986.)

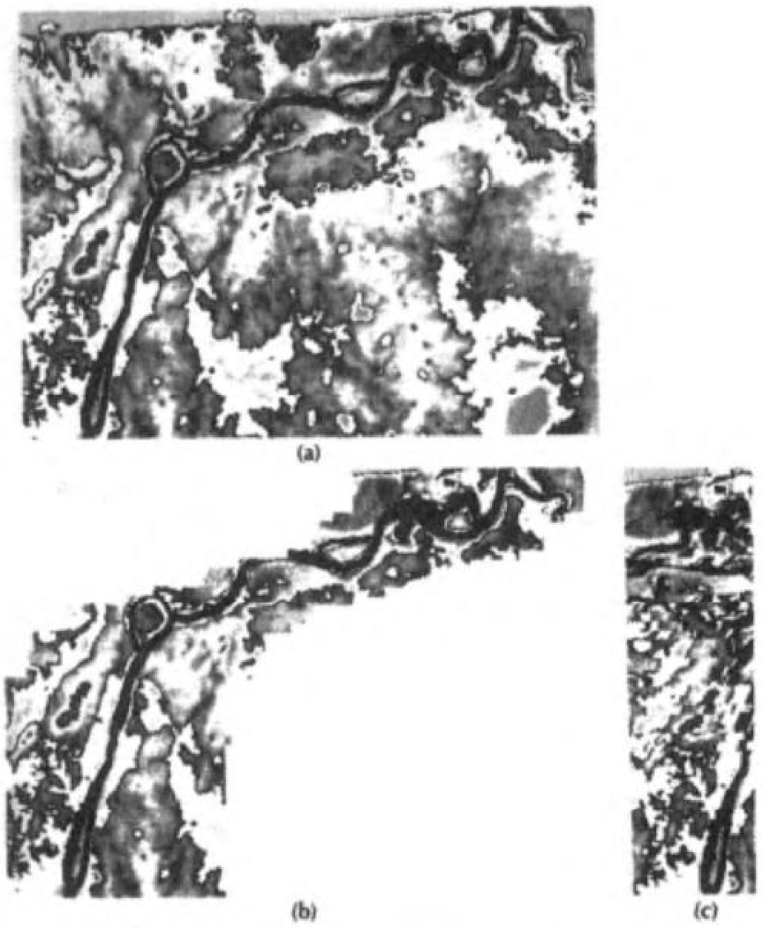


FIGURE 16.33 Simulation experiment and results on Yangtze River. (From Bow et al., 1986.)

show the images, which are at least two times wider than the scanner view. Figures 16.32b and 16.33b show the changes in the scanner view for the optimal acquisition of useful information (tracking on the river). Figures 16.32c and 16.33c show the stored data for later image restoration. From these figures we can see that in a single flight we could acquire ground information across as wide an area as three swaths.

16.4 VISION USED FOR CONTROL

16.4.1 Traffic Flow Measuring Using an Image Processing Technique

Another example of the image processing application is described here, use of the image processing technique to measure road traffic flow so as to maintain a smooth flow and safe driving conditions. This kind of measurement has conventionally been done using ultrasonic or magnetic sensors installed at various locations. However, the accuracy achieved on parameters such as vehicle volume, average vehicle speed, and so on, is relatively low. By using an image processing technique we can perform the job directly from road images over a large area of road, by using a sequence of images, thus improving the accuracy of the measurements as well as the estimation of some useful parameters.

When using an image processing technique, problems such as the effects of vehicle shadows and the occurrence of a traffic jam at dusk would exist. However, the system discussed here will take care of them. Two steps will be involved in the image processing: (1) to extract vehicle candidates from the road image and check whether or not they are vehicles, taking into consideration the occurrence of shadows and changes in brightness, particularly at dusk; and (2) to measure the traffic flow parameters at high speed.

To extract a vehicle from the image, the following method involving background subtracting with spatial differentiation is adopted. Figure 16.34 shows a schematic diagram of the processing system together with photographs.

Figure 16.34a, b, and c shows the input image, background image, and subtracted image $|f - g|$, respectively. Figure 16.34d shows the spatially differentiated image of Figure 16.34c. Figure 16.34e and f shows, respectively, the extracted image obtained from the differentiated images and that obtained directly from the subtracted image. It can be seen in Figure 16.34e that objects (a), (b), and (d) in Figure 16.34c are deleted from the processed image as candidates; only object (c) is extracted as a vehicle. Thus the adverse effect of vehicle shadows has been successfully eliminated. By contrast, in Figure 16.34f, objects (e), (f), and (h), which represent parts of vehicle shadows, would have been extracted as candidates for vehicles, and consequently, incorrect judgments would result.

For high-speed processing the image input, background subtraction, spatial differentiation, and binarization operations are pipelined. Other measurements, such as the average speed of vehicles and their spatial occupancy can be made similarly. It was reported by Takatoo (1989) that it was possible to track a vehicle running at 150 km/h (roughly 93 mph).

16.4.2 Autonomous Navigation System for Robotic Vehicle Road Following

To navigate adequately through its environment, an autonomous vehicle must plan its actions, perceive its surroundings, execute its plans, and adapt to the environment. The integrated navigation system consists of three parts: (1) hardware components, such as vehicle, sensors, and computer hardware; (2) a vision subsystem, composed of video data processing and range data processing; and (3) a reasoning subsystem. This reasoning subsystem is actually an executive

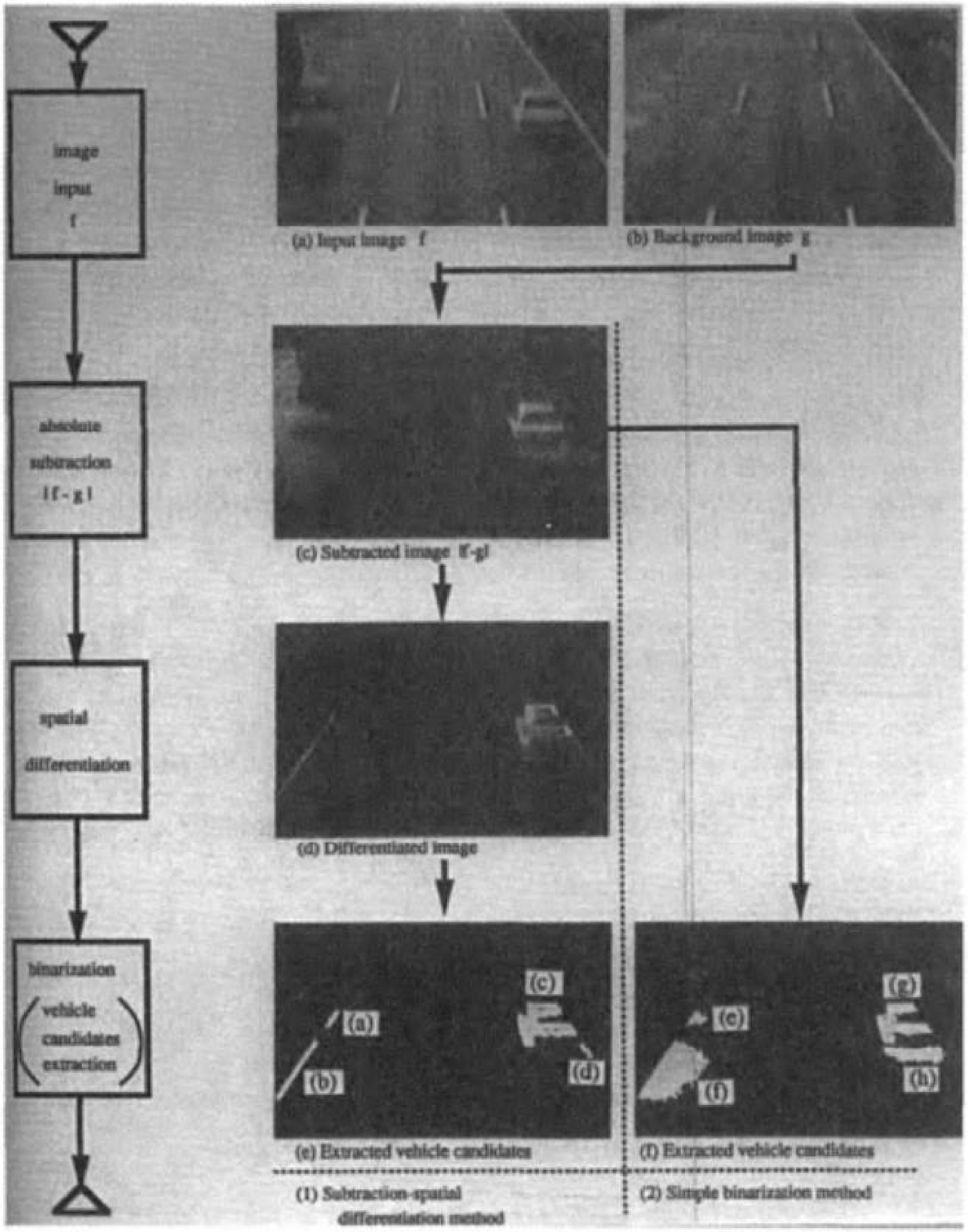


FIGURE 16.34 Vehicle extraction using background subtraction and spatial differentiation. (From Takatoo et al., 1989.)

controller, coordinating the other subsystems to accomplish the goal specified by the script. No doubt, a knowledge base is part of this reasoning subsystem. However, vision is a very important input resource for reasoning. These two subsystems are intimately tied together. The vision subsystem supplies a description of the observed road to the reasoning subsystem, while reasoning subsystem passes the position update message back and at the same time provides visual cues to the visual subsystem.

The primary vision (or perception) task is to provide a description of the world. This information should be rich enough to facilitate road following, obstacle avoidance, landmark recognition, and cross-country navigation. There is such as strategic computing program in the Defense Advanced Research Projects Agency (DARPA). For this project, several industrial laboratories and universities have developed prototype systems equipped with reasoning and perception capability. Some of the approaches used in road following, especially in the vision subsystem, are reviewed in this section.

The road-following algorithms that have been developed can be summarized in the following steps: visual information acquisition, segmentation of road/nonroad regions, extraction of road boundaries, and building a scene model. The sensor system includes a color camera for road following, a sonic image sensor for obstacle avoidance, an infrared sensor for target detection, and a laser range scanner for range determination.

Segmentation of ROAD/NOROAD Region

Due to the wide variation in the spectral characteristics of objects, one spectral band road image does not contain enough information for classification of the two main classes, ROAD and NOROAD. Comparatively, the blue color band image usually possesses better discriminatory power to extract ROAD segments, especially for a gravel road. However, in this application, a red-green-blue color image is used to achieve better performance. Kuan et al. (1988) showed that the optimal color transformation for a typical gravel road image is

$$Y = 0.175R - 0.030G + 0.795B$$

where R, G, and B are, respectively, the red, green, and blue component image samples of the scene, and Y is the optimal composite image constructed for gravel road identification. In general, Y can be expressed as

$$Y = w_1 R + w_2 G + w_3 B$$

and the discriminatory power $J(w_1, w_2, w_3)$ of a transformation $\mathbf{w}$ can be defined as

$$J(w_1, w_2, w_3) = \frac{(m_r^y - m_{\text{nr}}^y)^2}{v_r^2 + v_{\text{nr}}^2} \tag{16.20}$$

where $\mathbf{w} = (w_1, w_2, w_3)$ is the optimal projection transformation, m_r^y and m_{nr}^y are the means of the projected road and noroad samples, respectively, and v_r^2 and v_{nr}^2 are the variances in the projected road and noroad samples. RGB is the three-dimensional color feature space. With $\mathbf{w} = (w_1, w_2, w_3) = (0.175, -0.030, 0.795)$ for the gravel road, the discriminatory power is computed as 1.267. For a typical paved road image, the optimal color transformation changes to

$$Y = -0.35R + 0.200G + 0.450B$$

with the discriminatory power of the transformed image Y as 2.58. In contrast with this figure, the discriminatory power computed only from the blue image is 1.990.

Experience obtained by Turk et al. (1988) has shown that good segmentation is achievable without using the green band for their VITS system, so the problem can be reduced conceptually to finding the slope of a line in a two-dimensional red-blue plane, rather than finding the normal of a plane in RGB space. Figure 16.35 shows a scatter diagram of the red and blue components of a road image constructed by properly choosing the threshold in the following expression:

$$I'(i,j) = \begin{cases} 1 & \text{if } w_1 R(i,j) + w_2 G(i,j) + w_3 B(i,j) + \lambda < \theta \\ 0 & \text{otherwise} \end{cases} \tag{16.21}$$

The binary images I' (a function of the threshold λ) shown in Figure 16.35b, c, and d are the results obtained that correspond respectively, to the separating lines marked (b), (c), and (d).

Extraction of Road Boundaries

RGB images derived from the previous paragraphs will first be transformed into a single transform color image according to the parameter $\mathbf{w}$ derived in the preceding section. A pixel classification technique basing on the probabilistic method can be used to segment the image into ROAD/NOROAD regions. Let

$P(x|\text{ROAD})$ = conditional probability that pixel $\mathbf{x}$ is from the road class

$P(x|\text{NOROAD})$ = conditional probability that pixel $\mathbf{x}$ is from the noroad class

$P(\text{ROAD})$ = a priori probability of a pixel being road

$P(\text{NOROAD})$ = a priori probability of a pixel being noroad

$P(\text{ROAD})$ AND $P(\text{NOROAD})$ can be approximated by the area of the road and noroad regions in the image.

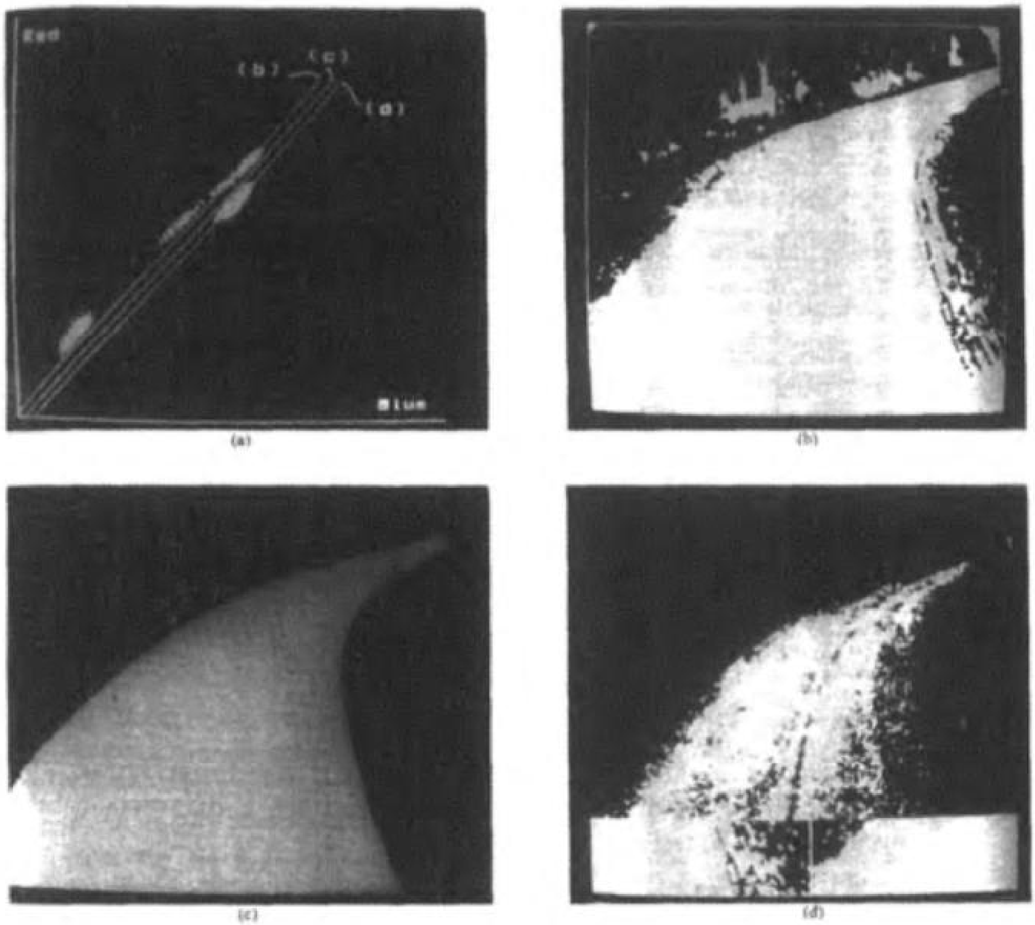


Figure 16.35 (a) Scattergram. (b)–(d) Result of thresholding at different values. (From Turk et al., 1988.)

Denote $P(\text{NOROAD})/P(\text{ROAD})$ by β; a pixel $\mathbf{x}$ is classified as "ROAD" if the ratio is

$$\frac{P(x|\text{ROAD})}{P(x|\text{NOROAD})} > \beta$$

Otherwise, it is classified as "NOROAD." Results obtained after applying this pixel classification algorithm are shown in Figures 16.36 and 16.37. Figure 16.36b is the segmented road region for the gravel road image shown in Figure 16.36a, and Figure 16.37b that for the paved road image shown in Figure 16.37a.

Building a Scene Model

Road regions extracted by the image segmentation algorithm described in the previous sections are what the vision system really sees. Reasoning in geometry

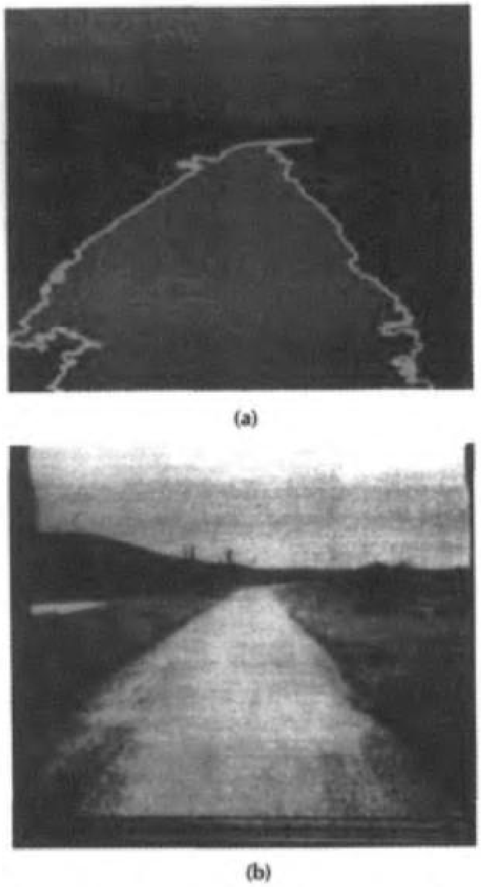


FIGURE 16.36 Typical gravel road image. (From Kuan et al., 1988.)

for the road model introduced would provide the vision system with perception capability. Generic information about road construction is useful in road segmentation. For example, if a road edge exists on the left side of the road, there should also be one on the right side. When the road turns on a curve, both the left and right sides of the road change direction smoothly. With the exception of the end of the road, the continuity between road boundaries exists in two successive images, and so on. This information is useful for geometric reasoning, especially in situations where there are different lighting conditions, seasonal changes, puddles or shadows on the road, and so on.

If every edge element of a roadside has support from the other side, they are used in the final road interpretation. When all the road edge points are selected in the image, a scene model or a three-dimensional description of the road can be established and sent to the reasoning subsystem for trajectory calculation. Various approaches, such as a flat-earth geometry model and a hill-and-dale geometry

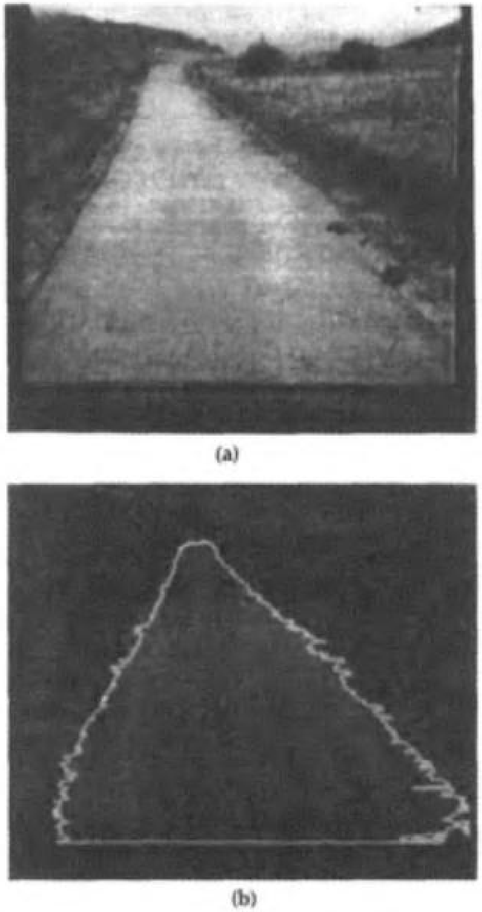


FIGURE 16.37 Typical paved road image. (From Kuan et al., 1988.)

model, have been suggested for this task. Interested readers are referred to [Kuan et al. (1988)] for details. According to this report, a field test has been performed by driving a vehicle down the road at 8 to 10 km/h with a traveling distance of up to 4.5 km along a paved road, and 0.4 mile along a gravel road that passes through several intersections and gentle curves. The processing time was 4 to 5 seconds per image and the visibility limit was set at 40 m. Road tests were conducted under both clear and cloudy skies.

Similar work has been in progress at General Motors Research Laboratories on a lane-sensing system for vehicle automatic guidance. Figure 16.38 shows how a collision warning signal is generated with a vision-steered radar concept.

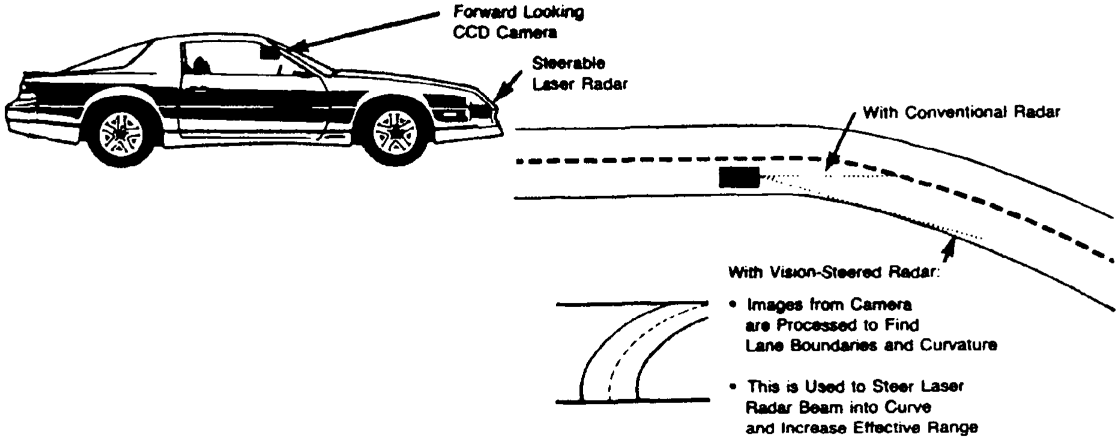


FIGURE 16.38 Collision warning: vision-steered radar concept. (Courtesy of General Motor Research Laboratory, LANELOK, 1990.)

This Page Intentionally Left Blank

Part V

Practical Concerns of Image Processing and Pattern Recognition

This Page Intentionally Left Blank

17

Computer System Architectures for Image Processing and Pattern Recognition

17.1 WHAT WE EXPECT TO ACHIEVE FROM THE POINT OF VIEW OF COMPUTER SYSTEM ARCHITECTURE

There is a very famous Chinese saying that seeing for oneself is a hundred times better than hearing from others. This implies that the information content in a picture is tremendously rich—and it really is. However, to store a picture in computer memory or to transmit a picture by computer, the amount of information that must be processed is extremely large. Let us take for illustration an image of 512 × 512 pixel resolution; 512 × 512, or 262,144, bytes are needed to store one image. To process this image in real time, we need to process 512 × 512 × 8 × 30 bits, or 62.9 Mbits, in 1 second. If the resolution of the image is to be upgraded a little, say to 1024 × 1024 pixels, an even larger memory (1.05 Mbytes for one image, which is equivalent to storing a 250-page book) and a higher processing speed (31.5 Mbytes/s) are needed. If an average of 15 floating-point operations (FLOPS) are assumed to process a pixel in an image, there will be 1.05 × 15, or 15.73, MFLOPS (millions of FLOPS) for a single processing function such as edge detection. As we know, many other functional operations are involved in an image processing algorithm, so an adequate image analysis computer is expected to perform at least 1 G FLOPS or higher with a memory bandwidth of at least 2 Gbytes for present and future applications. Although computing capacity continues to evolve at an astonishing speed, yet in terms of cost-effectiveness and affordability, the development and popularization of the image processing and understanding system are restricted.

This is a problem of great concern, especially in real-time processing. We could not turn our hope into reality by developing a computer that works faster. The best way is to distribute the work over an ensemble of processors to achieve speedup as a result of concurrent implementation. This is the subject of parallel processing. Diverse opinions exist as to how much speedup can be achieved by parallel processing. Some predictions are low; others are high. These might come from a different basis for the ideas in processor design. The most optimistic prediction (Hwang and Briggs, 1984) can reach a value of $n/\log_e n$, where n is the number of processors used. When $n = 1000$, the speedup value is 144.8. When $n = 10{,}000$, the speedup can go as high as 1086. In such cases the system architecture comprises a large group of processors, each with simple processing functions. Such an idea is applicable to image processing, where the task typically forms the bulk of the computation and possesses comparatively few well-characterized operations. In addition, for point processing and neighborhood processing, which occur so frequently among the various algorithms, the data structure of the image can be partitioned into small blocks for concurrent processing. This strongly supports such an architecture.

17.2 OVERVIEW OF SPECIFIC LOGIC PROCESSING AND MATHEMATICAL COMPUTATION

No general principles exist for systematic application of parallel processing techniques to image processing. Two approaches can be followed: dedicated implementation and reconfigurable networks for the connection of multiple programmable processors. Dedicated implementation can provide high-speed performance, but the range is narrow. The reconfigurable network provides flexibility—but achieved at the expense of increased complexity and some sacrifice in processing speed. The complexity comes primarily from communication and interaction between parallel image processing tasks. As a matter of fact, both of these two approaches have valuable distinguishing features and have absorbed the good points of others into their systems to make their systems more successful.

To summarize, it seems indicated to form a system with a large number of processor elements, regardless of the form—whether a dedicated implementation or a reconfigurable network architecture. For this reason, the discussion in this chapter assumes that a multiple-processor structure is used.

In general, an algorithm in image processing can be divided into tasks, each of which is divided into subtasks. Among the subtasks, some can be used for these problems (or algorithms) and can be used in other problems (or algorithms). Proper sequencing of these subtask operations in their respective algorithm depends on the particular problem tackled. They may be operated concurrently or governed sequentially by precedence constraints. The output of one subtask may be the input of another. Outputs of several subtasks may be integrated as an input for another. The output of one subtask can also be fed back as the input of several others. And on and on. Communication and interactions among the subtasks should be designed in a very flexible fashion and be software controlled.

Among the arithmetic computations and logic operations within each subtask, there may be two categories. One uses only data of its own or data from the local area; the other might require information about the pixels, which spread over the entire image. For the first category of processing, which includes point processing and neighborhood processing, the image can be partitioned obviously and easily into a disjoint data subset. The entire processing operation can easily be broken down into subtasks, each operated on a separate disjoint data subset. There should not be any problem in concurrent processing on the subtask level. The only thing to worry about is interconnection of the output of the subtask with the next subtask. Of course, a data and resource dependency problem still exists at the subtask level (more on this below). But for the other category, where a task is operated on data of the entire image (i.e., the data needed cannot simply be partitioned as described above, as in FFT operation), parallel processing can also be carried on except that the data have to be transformed into a different mode.

In intertask communication and interaction, two dependencies, the resource and data dependencies, need to be considered. "Resources" here refers to the processing elements, memory units, and I/O devices, as shown in Figure 17.1.

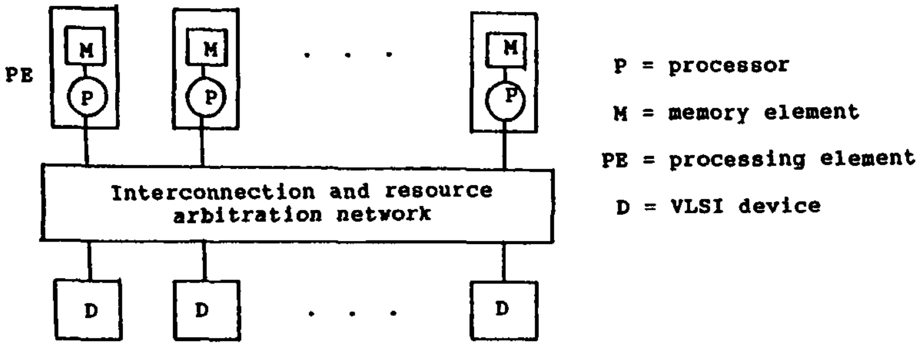


FIGURE 17.1 Multiple-processing element system.

Linking of the processing elements is through the interaction network, whose function is to establish data/instruction paths between subsets (or partitions) of the resources. These partitions can function independently or interdependently. Proper partition of the resources and proper scheduling of the subtasks are expected to achieve a better performance (i.e., the least total processing time) for a certain application. For another application, another partition and interconnection within the partition should be established.

For a good architecture, the system for pattern recognition and image processing applications should be able to function in a single-instruction–multiple-data-stream (SIMD) mode, and operate synchronously with replicated arithmetic/logic units. No doubt, a high degree of pipelining is expected for overlapping of the several steps involved in an instruction cycle: instruction fetch, decode, effective address computation, operand fetch, and instruction execution. Similarly, a high degree of pipelining of arithmetic is also expected. In addition, the system should support asynchronous computation in the multiple-instruction–multiple-data-stream (MIMD) mode, with processors doing jobs independently. Results obtained from the computation of each process are integrated according to a certain function. These are the operating characteristics of image processing and pattern recognition, where the results of local computations are not images but are the data structures, which have to be combined to give a meaningful interpretation or description (e.g., for a scenic object).

17.3 INTERCONNECTION NETWORKS FOR SIMD COMPUTERS

Consider n processors in a system, with each processor capable of carrying out some task(s) independently on a set of input data. Many interconnection networks have been suggested for SIMD computers and some are summarized in Figure 17.2. The linear array shown in Figure 17.2a is a one-dimensional (pipeline architecture). The ring, star, tree, mesh, and systolic arrays shown in Figure 17.2b to f are of two dimensions, while the completely connected, chordal ring, three-cube, and three-cube-connected cycle networks are classified as three-dimensional in topology. Many of these interconnection networks are useful for image processing and pattern recognition purposes. However, owing to space limitations, only systolic array architecture is discussed here in more detail.

17.4 SYSTOLIC ARRAY ARCHITECTURE

Systolic array is at present a very common architectural form of computer systems. Many versions have been developed by universities and industrial organizations since the form was suggested by Kung in 1982. Systolic array, as described in Figure 17.2f, possesses the following features:

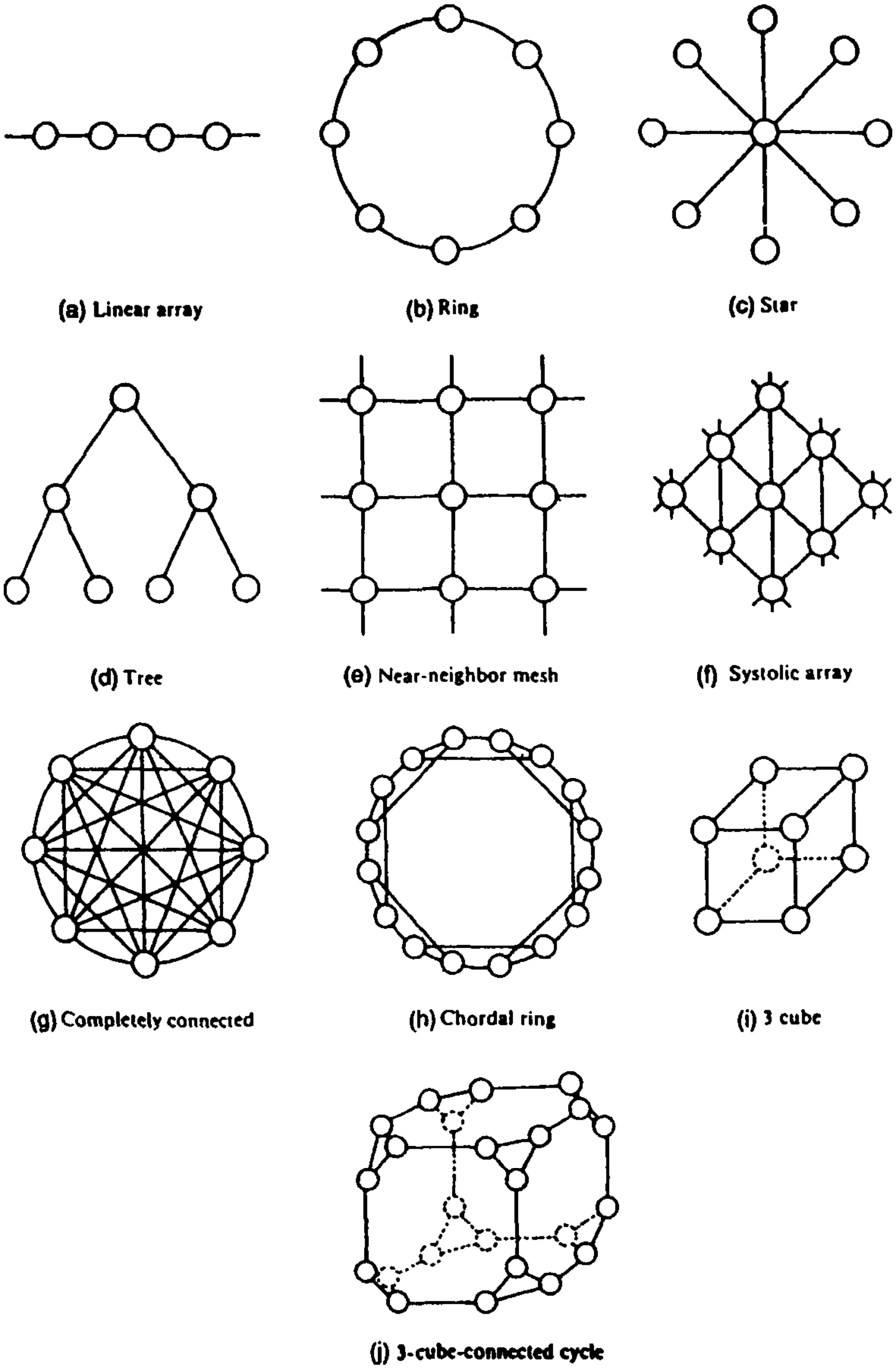


FIGURE 17.2 Static interconnection network topologies. (From Feng, 1981.)

1. It consists of a set of interconnected cells (PEs) each capable of performing simple operations such as LOAD M; ADD M; ADDB source; SUB M; SUBB source; MULB source; DIVB source, UP, DOWN, LEFT, RIGHT (for communication of data between each element and its four neighbors via communication registers); MOV destination source; and THR ##.
2. It possesses a simple and regular communication and control structure so that the PEs in the system are interconnected to form a systolic array.
3. Information flows between cells are handled in pipeline fashion.

These features explain why systolic array is cost-effective and offers high performance, as many operations are local in nature and run repetitiously through all the pixels in the image (a huge number). Figure 17.3 shows two examples of a systolic array configuration in practical use, a square and a hexagonal configuration.

Due to its simplicity, the systolic array technique has attracted a great deal of attention during the past two decades. However, the implementation of systolic arrays on a VLSI chip has many practical constraints, due mainly to its I/O barrier. For this reason, reconfigurable systolic array has become a popular topic of development in the field of computer system architecture. Figures 17.4 to 17.6 show various specialized processor array configurations that meet algorithmic needs.

There are quite a number of commercial systems capable of performing basic functions such as grabbing, specific image processing, and image display. Interested readers should contact vendors for details. Processing of color images is another challenging subject. It is hoped that more results will soon be available.

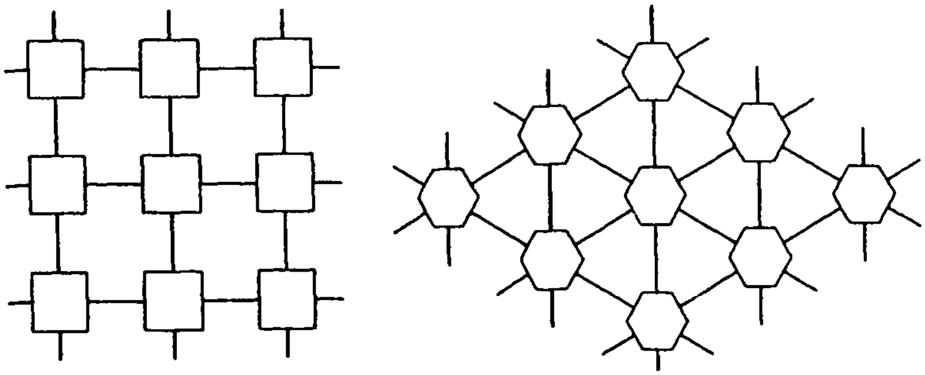
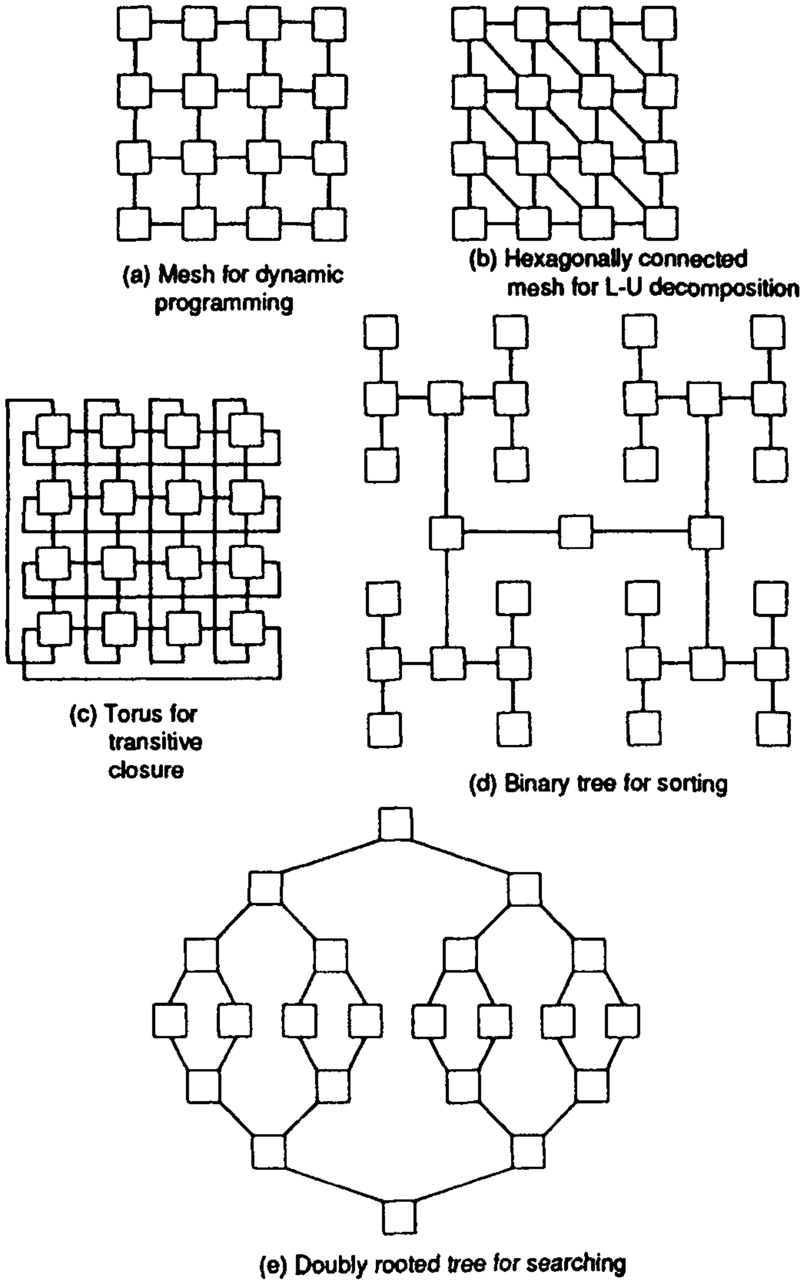


FIGURE 17.3 Two examples of a systolic array configuration.



(a) Mesh for dynamic programming

(b) Hexagonally connected mesh for L-U decomposition

(c) Torus for transitive closure

(d) Binary tree for sorting

(e) Doubly rooted tree for searching

FIGURE 17.4 Algorithmically specialized processor array configurations. (From Snyder, 1982.)

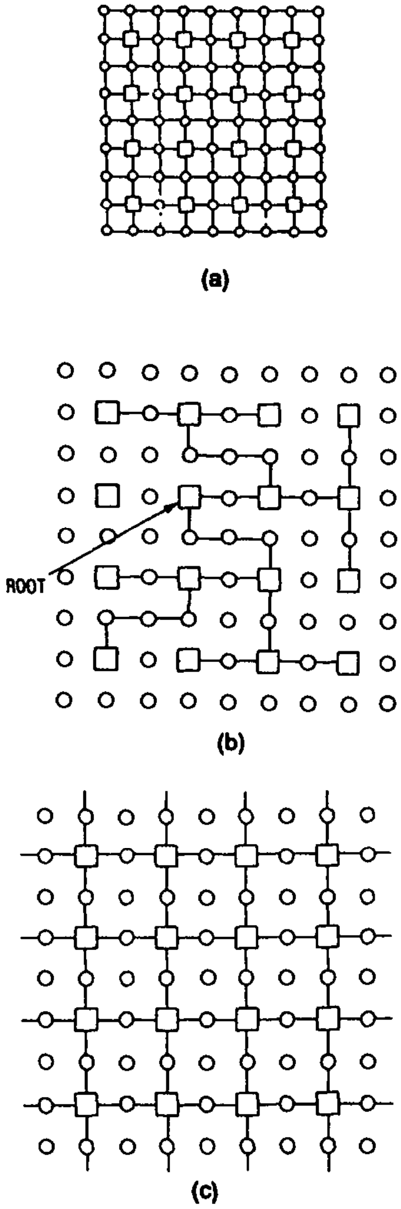


Figure 17.5 (a) Switch lattice structure and its configuration into a binary tree (b) and a mesh pattern (c). Circles represent switches; squares represent PEs. (From Snyder, 1982.)

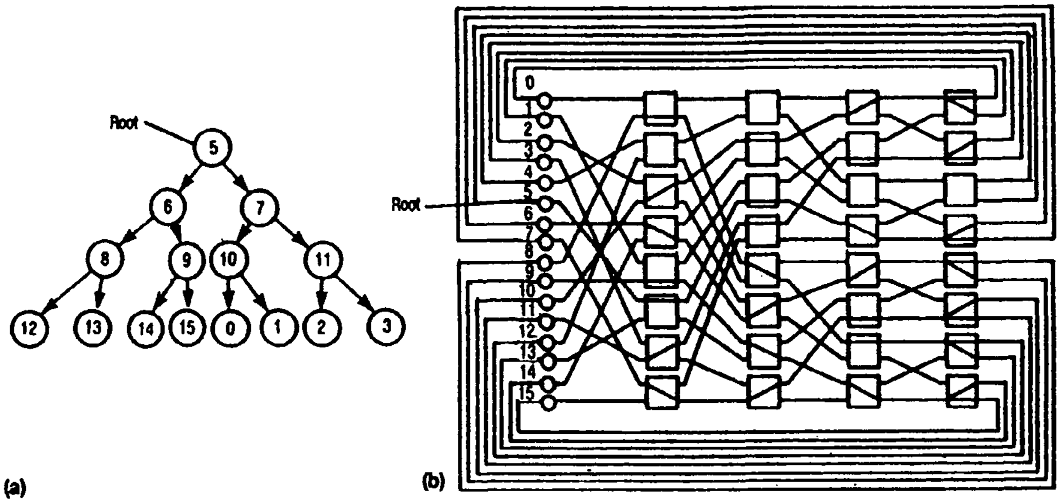


FIGURE 17.6 Realization of an algorithm for setting up a binary tree rooted at any node $R(a)$. (From Yalamanchilli and Aggarwal, 1985.)

This Page Intentionally Left Blank

Appendix A: Digitized Images

The 512×512, 256-gray-level images in Figures A.1 to A.16 can be used as large data sets for many of the pattern recognition and data preprocessing concepts developed in this book. These data sets can be used in their original form or they can be corrupted, for example, by adding noise to each pixel or by any specified gray-level transformation. By so doing, a variety of input data sets can be generated that can be used to illustrate algorithms for pattern recognition and for data preprocessing. The results obtained can be displayed on a standard line printer, dot printer, thermal printer or laser printer, as shown in the text. These digitized images were recorded on $3\frac{1}{2}$ and $5\frac{1}{4}$-in.-high-density disks.

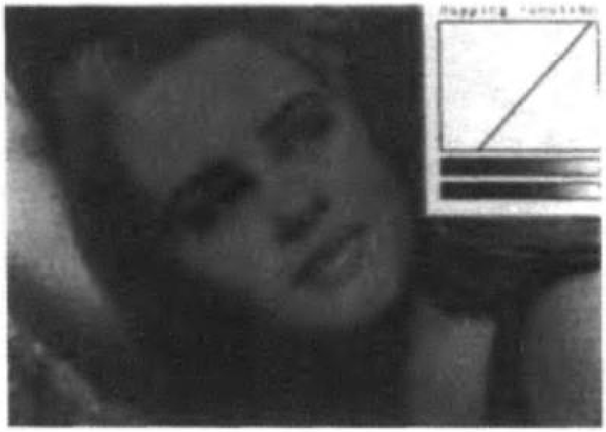


FIGURE A.1 A 512×512 256-gray-level digitized image.

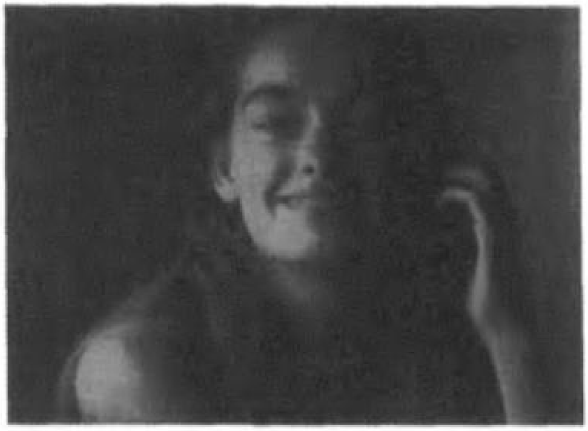


FIGURE A.2 A 512 × 512 256-gray-level digitized image.

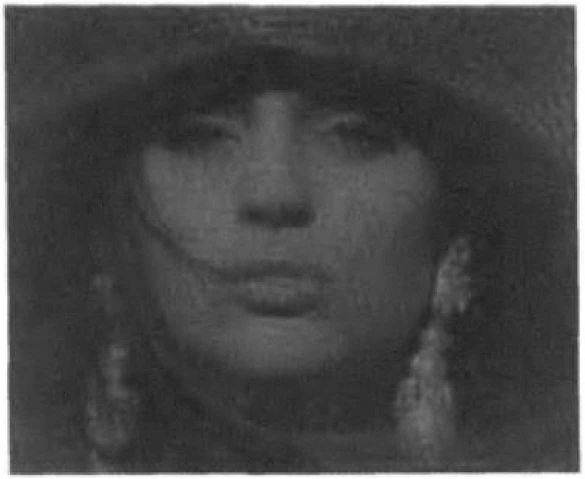


FIGURE A.3 A 512 × 512 256-gray-level digitized image.

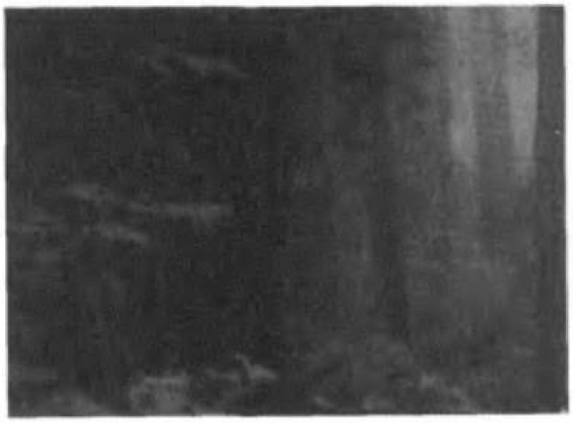


FIGURE A.4 A 512 × 512 256-gray-level digitized image.

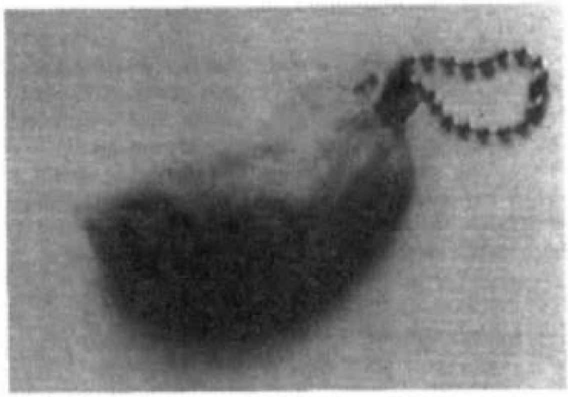


FIGURE A.5 A 512 × 512 256-gray-level digitized image.

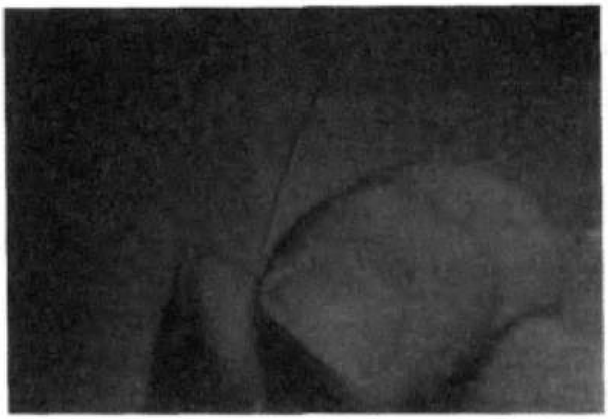


FIGURE A.6 A 512 × 512 256-gray-level digitized image.

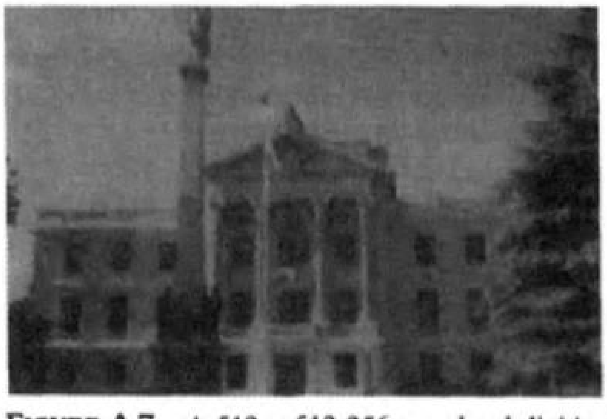


FIGURE A.7 A 512 × 512 256-gray-level digitized image.

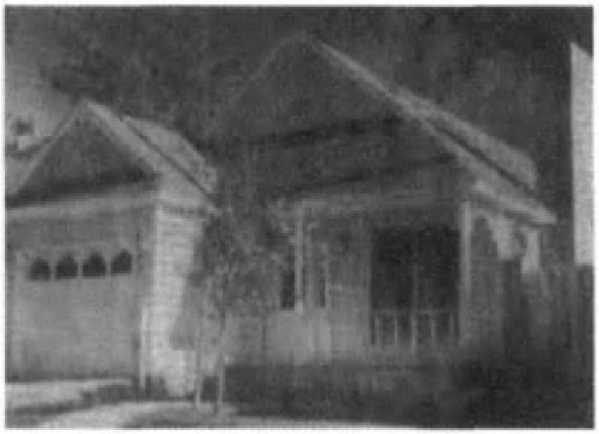


FIGURE A.8 A 512 × 512 256-gray-level digitized image.

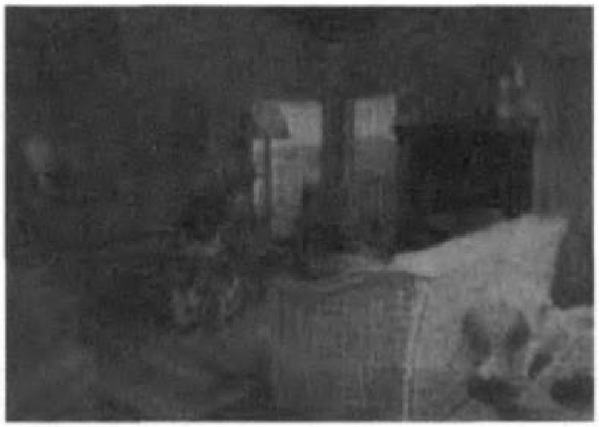


FIGURE A.9 A 512 × 512 256-gray-level digitized image.

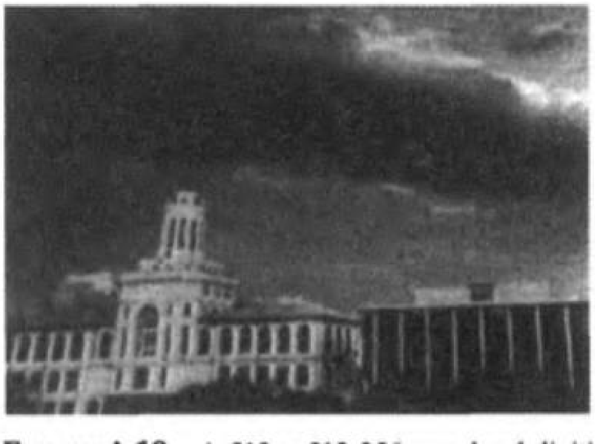


FIGURE A.10 A 512 × 512 256-gray-level digitized image.

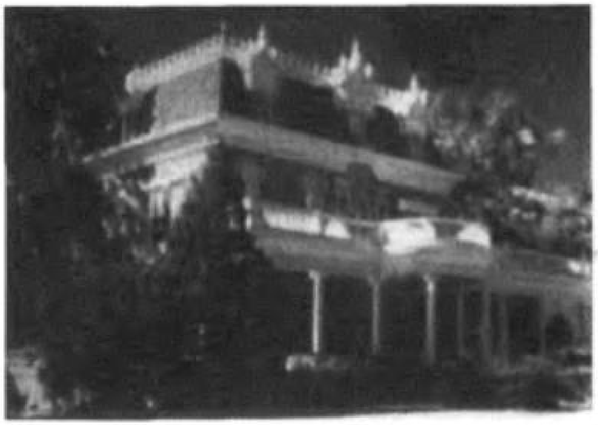


FIGURE A.11 A 512 × 512 256-gray-level digitized image.

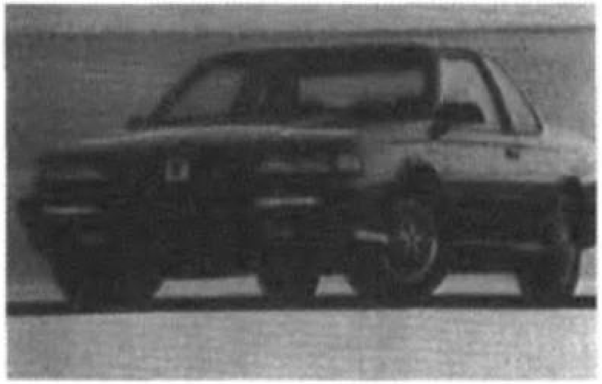


FIGURE A.12 A 512 × 512 256-gray-level digitized image.

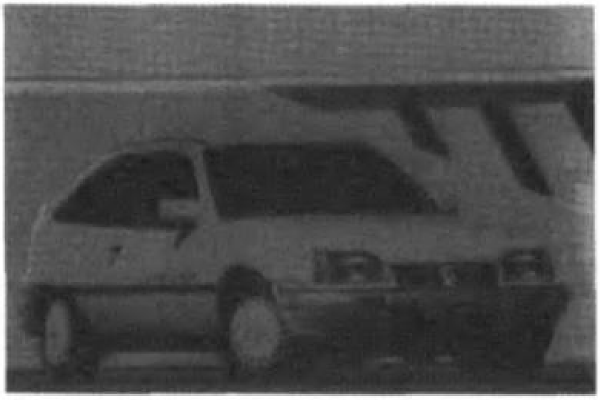


FIGURE A.13 A 512 × 512 256-gray-level digitized image.

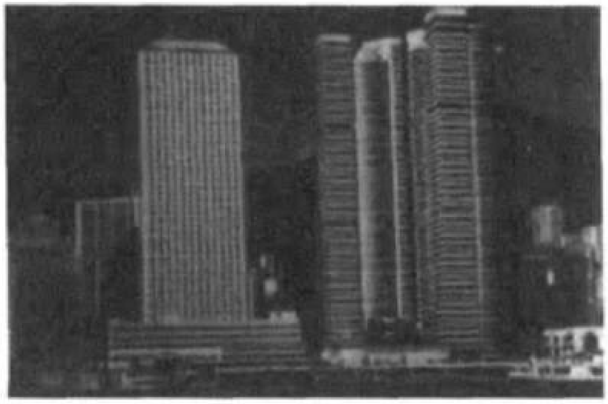


FIGURE A.14 A 512 × 512 256-gray-level digitized image.

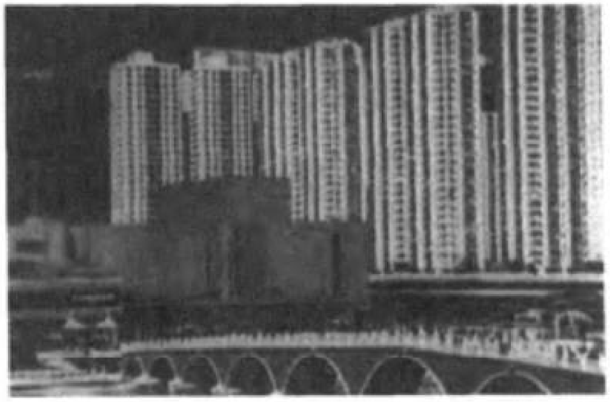


FIGURE A.15 A 512 × 512 256-gray-level digitized image.

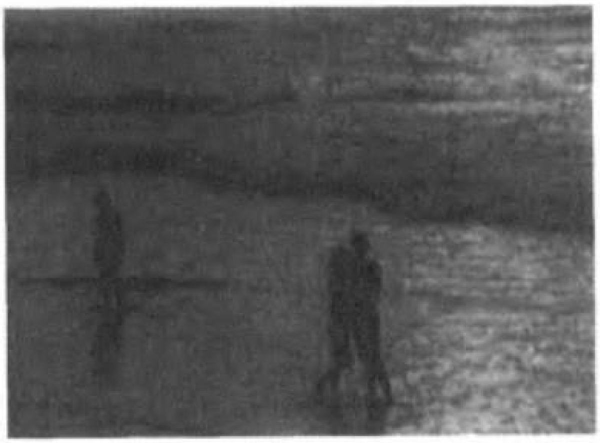


FIGURE A.16 A 512 × 512 256-gray-level digitized image.

Appendix B: Image Model and Discrete Mathematics

B.1 IMAGE MODEL

A dictionary definition of an *image* is a "representation, likeness, or imitation of an object or thing, a vivid or graphic description, something introduced to represent something else." As used in this book, an image contains descriptive information about the object it represents. For instance, a photograph displays this information in a manner that allows the human eye and brain to visualize the subject. Under such a relatively broad definition, images can be classified into several types based on their form and the method of their generation. Figure B.1 shows the set of all objects, while images form a subset of them.

Within the image subset, there are images that can be seen and perceived by eye. There are also some physical images that are nonvisible but are distributions of measurable physical properties. Photographs, drawings and paintings, optical images, and the like belong to the first category, while temperature, pressure, elevation, and population density maps are nonvisible physical images. Another subset of physical images is that of multispectral images, those that have more than one local property defined at that point. Another subset of images contains abstract mathematical images in the form of continuous functions, discrete functions, or digital images. Digital images, the numerical representation of objects, are those with which we have been primarily concerned in this book.

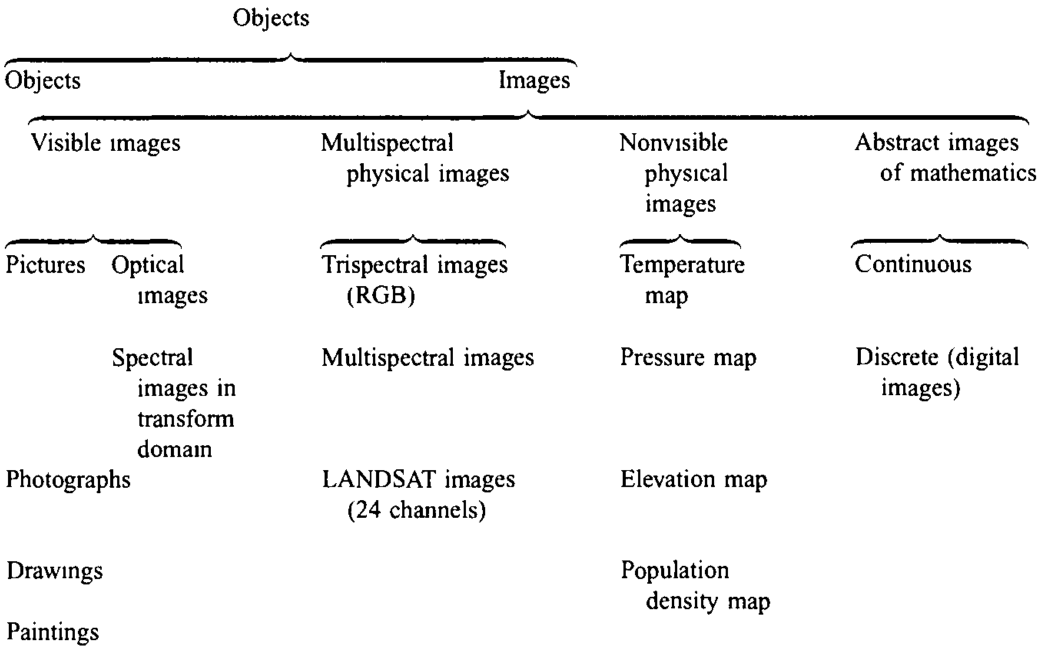


FIGURE B.1

With the images defined as above, it will be clear that digital image processing involves a sequence of operations performed on numerical representations of objects to achieve a desired result. In the case of pictures, processing modifies their form to make them more desirable for diagnostic use.

In the analysis and processing of an image, it is usually convenient and often necessary to characterize mathematically the image to be processed. An image can be represented as a spatial radiant energy distribution, which is a function of five variables shown as

$$C(x, y, z, t, \lambda) \tag{B.1}$$

where x, y, and z are spatial variables (space), t is a temporal variable (time), and λ is a spectral variable (wavelength). In general, the image function $C(x, y, z, t, \lambda)$ is somewhere between zero and A, where A is the maximum image brightness, which in image processing is set to 255 if 256 gray levels are used, or is set to 1 if normalized. This is because light intensity is a real positive quantity and is proportional to the modulus squared of the electric field. Image light function is a real, nonnegative function. With the limitations of imaging systems and recording media, and for mathematical simplicity, images are assumed to exist over a rectangular region. We than have

$$\begin{gathered} 0 \leq x \leq L_x \\ 0 \leq y \leq L_y \\ 0 \leq z \leq L_z \end{gathered} \tag{B.2}$$

Since an image is observable only over a finite period,

$$0 \leq t \leq T \tag{B.3}$$

Basically, the colors we perceive in an object are determined by the nature of the light reflected from the object. A body that favors reflectance in a limited range of visible spectrum will exhibit some shades of color. For example, red objects reflect light within a spectral range of 0.57 to 0.70 μm (10^{-6} m) while absorbing most of the energy at other wavelengths. Green objects reflect light within a spectral range of 0.48 to 0.57 μm, blue objects within 0.40 to 0.48 μm, ultraviolet objects within 0.25 to 0.40 μm, and infrared from 0.9 to 1.5 μm. That is,

$$\alpha \leq \lambda \leq \beta \tag{B.4}$$

where $\alpha = 0.25\,\mu\text{m}$ and $\beta = 1.5\,\mu\text{m}$ for the objects of our interest. A general image function $C(x, y, z, t, \lambda)$ is therefore a bounded five-dimensional function with bounded independent variables. This function is assumed to be continuous over the domain of definition.

B.2 SIMPLIFICATION OF THE CONTINUOUS IMAGE MODEL

The brightness response of a human observer to an image light function is measured by the instantaneous luminance of the light field, or

$$L(x, y, z, t) = \int_0^\infty C(x, y, z, t, \lambda) V_s(\lambda)\, d\lambda \tag{B.5}$$

where $L(x, y, z, t)$ is the brightness response of a human observer to an image function, and $V_s(\lambda)$ is the relative luminous efficiency function or spectral response of human vision. For an arbitrary red-green-blue coordinate system, the instantaneous tristimulus values are

$$\begin{aligned} R(x, y, z, t) &= \int_0^\infty C(x, y, z, t, \lambda) R_s(\lambda)\, d\lambda \\ G(x, y, z, t) &= \int_0^\infty C(x, y, z, t, \lambda) G_s(\lambda)\, d\lambda \\ B(x, y, z, t) &= \int_0^\infty C(x, y, z, t, \lambda) B_s(\lambda)\, d\lambda \end{aligned} \tag{B.6}$$

and for a multispectral coordinate system, the ith spectral image field is given as

$$F_i(x, y, z, t) = \int_0^\infty C(x, y, z, t, \lambda) S_i(\lambda)\, d\lambda \tag{B.7}$$

where $S_i(\lambda)$ represents the spectral response of the ith sensor.

In many imaging systems, the variable t does not change. For example, an image obtained from an image projection device does not change with time. This is also true when an image is sampled in frames, as in movies. The variable z is generally sampled because we usually work on plane images. Consequently, the variables t and z may be dropped in subsequent discussions. Based on the arguments above, a continuous-image model is then simplified into a two-dimensional light luminance function of spatial variables:

$$0 < f(x, y) < A \tag{B.8}$$

where $f(x, y)$ is a nonzero and finite number.

Images that we perceive in our everyday visual activity normally consist of light reflected from objects. The image function $f(x, y)$ may then be considered as being characterized by two components:

$$f(x, y) = i(x, y)r(x, y) \tag{B.9}$$

where $i(x, y)$ is the illumination component, representing the amount of source light incident on the scene being viewed and is

$$0 < i(x, y) < \infty \tag{B.10}$$

and $r(x, y)$ is the reflectance component, representing the amount of light reflected by the objects in the scene. The nature of the illumination component $i(x, y)$ is determined by the light source and varies to a very large extent. For the sun, $i(x, y)$ on the earth surface is 9000 footcandles (fc); for the moon it is 0.01 fc. The range of these two $i(x, y)$'s varies as much as 900,000-fold. The design standard for office lighting is 100 fc.

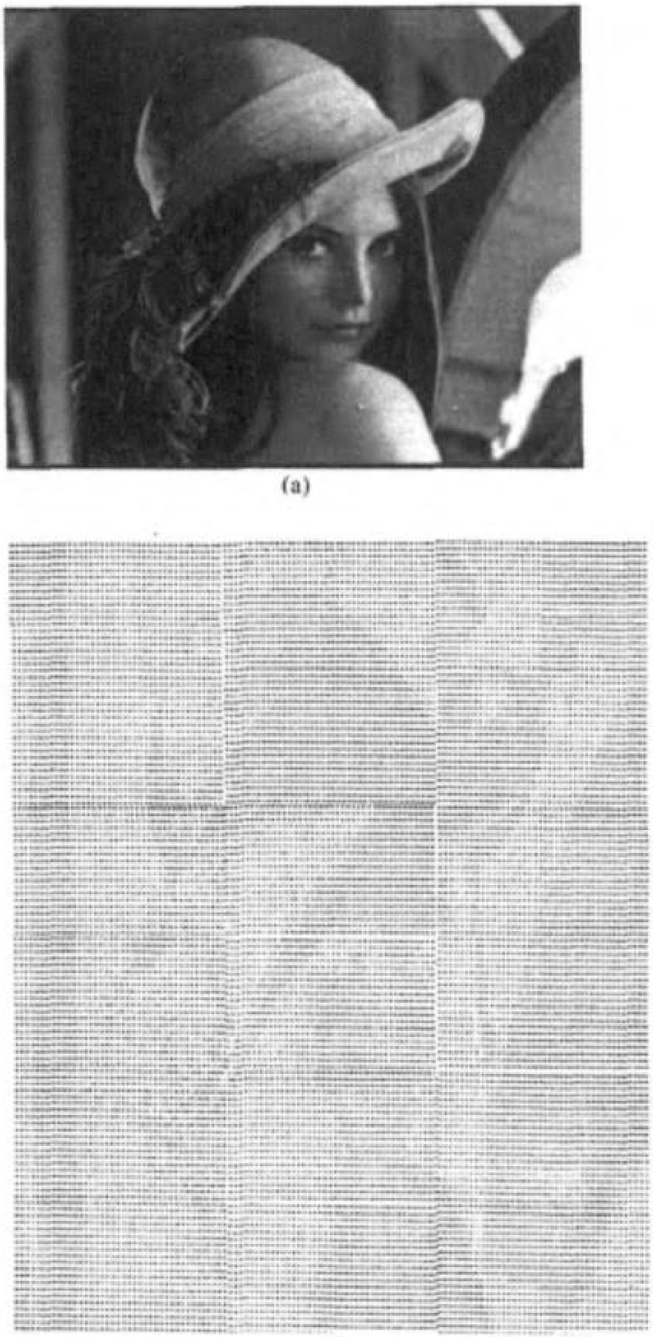
The reflectance component $r(x, y)$ represents the amount of light reflected by the objects in a scene, and is

$$0 < r(x, y) < 1 \tag{B.11}$$

where 0 indicates total absorption and 1, total reflectance. $r(x, y)$ is determined by the characteristics of objects in a scene. Some typical values of $r(x, y)$ for various objects are:

Black velvet	0.01
Stainless steel	0.65
Flat-white wall paint	0.80
Silver-plated metal	0.90
Snow	0.93

Figure B.2b shows a coded image of the image shown in Figure B.2a. Numbers on the coded image matrix are gray levels of the individual pixels, which are in the range $[L_{\min}, L_{\max}]$, [0,1], with 0 denoting black and 1 denoting white. Intermediate values are shades of gray varying from black to white.



(a)

(b)

FIGURE B.2 (a) Image; (b) a coded image for (a).

B.3 TWO-DIMENSIONAL DELTA FUNCTION

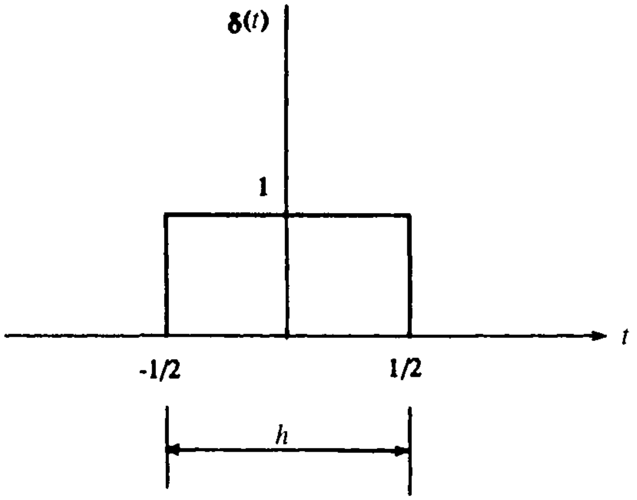
The *delta function* or *Dirac delta function*, which is a singularity operator, is a very useful technique used to characterize two-dimensional systems via impulse response or point-spread functions. This function is widely used in the analysis of systems involving sampling of continuous functions. For single-dimensional analysis, the Dirac delta function $\delta(t)$ is defined as the rectangular function shown in Figure B.3. When h approaches 0 as limit, and the area under the curve $\int_{-\varepsilon}^{\varepsilon} \delta(t)\,dt = 1$, with ε being a nonnegative number, $\delta(t)$ is a pulse at $t = 0$ and equals zero elsewhere.

In the two-dimensional case shown in Figure B.4, the two-dimensional Dirac delta function can similarly be defined as

$$\delta_n(x, y) = n^2 \operatorname{rect}(nx, ny) \tag{B.12}$$

where $n > 0$ and

$$|x| \le \frac{1}{2n} \quad \text{and} \quad |y| \le \frac{1}{2n}$$



$$\operatorname{rect}[x] = \begin{cases} 1 & |x| \le \frac{1}{2} \\ 0 & |x| > \frac{1}{2} \end{cases}$$

FIGURE B.3

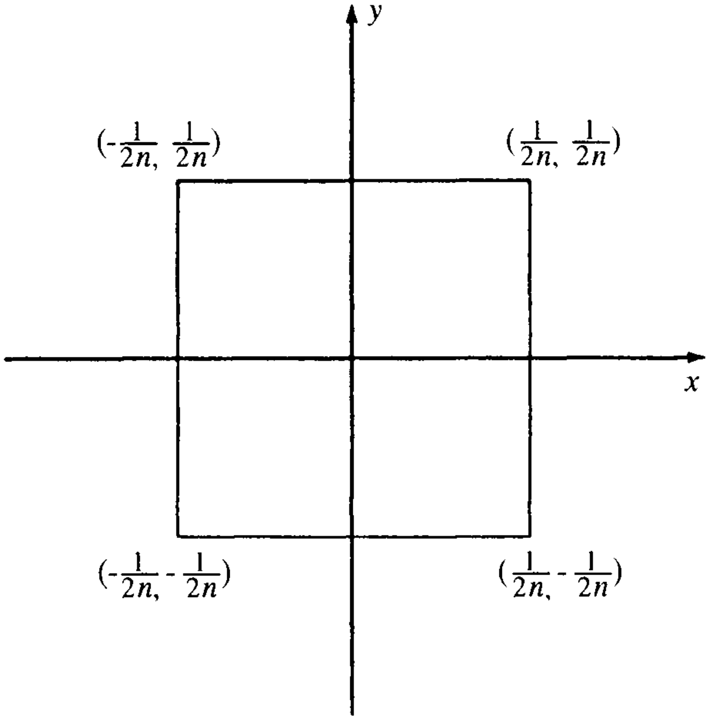

y
$(-\frac{1}{2n}, \frac{1}{2n})$
$(\frac{1}{2n}, \frac{1}{2n})$
x
$(-\frac{1}{2n}, -\frac{1}{2n})$
$(\frac{1}{2n}, -\frac{1}{2n})$

FIGURE B.4

We observe that $\delta_n(x, y)$ is nonzero only inside a $1/n \times 1/n$ square in the image plane, and

$$\int_{-\infty}^{\infty}\!\!\int \delta_n(x, y)\, dx\, dy = 1 \qquad \text{for all values of } n \tag{B.13}$$

As n approaches ∞, the two-dimensional Dirac delta function possesses the following properties:

$$\delta(x, y) = \begin{cases} \infty & \text{at the point } x = 0, y = 0 \\ 0 & \text{elsewhere} \end{cases} \tag{B.14}$$

In general, the δ function at point (ξ, η) is

$$\delta(x - \xi, y - \eta) = \begin{cases} \infty & \text{at the point } x = \xi, y = \eta \\ 0 & \text{elsewhere} \end{cases} \tag{B.15}$$

and

$$\int_{-\varepsilon}^{\varepsilon}\!\!\int \delta(x, y)\, dx\, dy = 1 \qquad \text{for } \varepsilon > 0 \tag{B.16}$$

If we multiply $F(\xi, \eta)$ by the δ function and integrate, we have

$$\iint_{-\infty}^{\infty} F(\xi, \eta)\delta(x - \xi, y - \eta)\, d\xi\, d\eta = F(x, y) \tag{B.17}$$

This is called the sifting property of the Dirac delta function.

B.4 ADDITIVE LINEAR OPERATORS

A two-dimensional system is said to be an *additive linear system* if the principle of additive superposition holds for the system:

$$\mathcal{O}\{a_1 f_1(x, y) + a_2 f_2(x, y)\} = a_1\mathcal{O}\{f_1(x, y)\} + a_2\mathcal{O}\{f_2(x, y)\} \tag{B.18}$$

where $\mathcal{O}$ is the mapping operator, and a_1 and a_2 are arbitrary scalars. With the sifting integral, an input function $f(x, y)$ can be represented as a summation of amplitude-weighted Dirac delta functions:

$$f(x, y) = \iint_{-\infty}^{\infty} f(\xi, \eta)\delta(x - \xi, y - \eta)\, d\xi\, d\eta \tag{B.19}$$

when $f(\xi, \eta)$ is the weighting factor of the impulse located at the coordinates (ξ, η) in the x–y plane (see Figure B.5), and $\delta(x - \xi, y - \eta)$ is a pulse at (ξ, η).

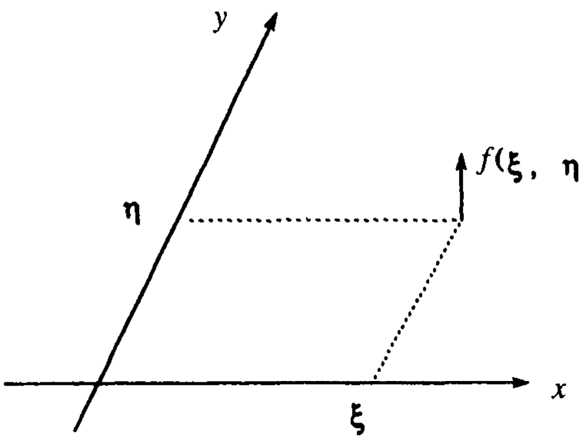


FIGURE B.5

B.5 CONVOLUTION

In the system shown in Figure B.6, the output function may be defined as

$$g(x, y) = \mathcal{O}\{f(x, y)\} \tag{B.20}$$

Substituting Eq. (B.20) for $f(x, y)$ gives

$$g(x, y) = \mathcal{O}\left\{ \iint_{-\infty}^{\infty} f(\xi, \eta)\delta(x - \xi, y - \eta)\, d\xi\, d\eta \right\} \tag{B.21}$$

Since the linear operator $\mathcal{O}$ can only be applied to the term in the integrand that is dependent on the spatial coordinate variables x and y, we have

$$g(x, y) = \iint_{-\infty}^{\infty} f(\xi, \eta)\mathcal{O}\{\delta(x - \xi, y - \eta)\}\, d\xi\, d\eta \tag{B.22}$$

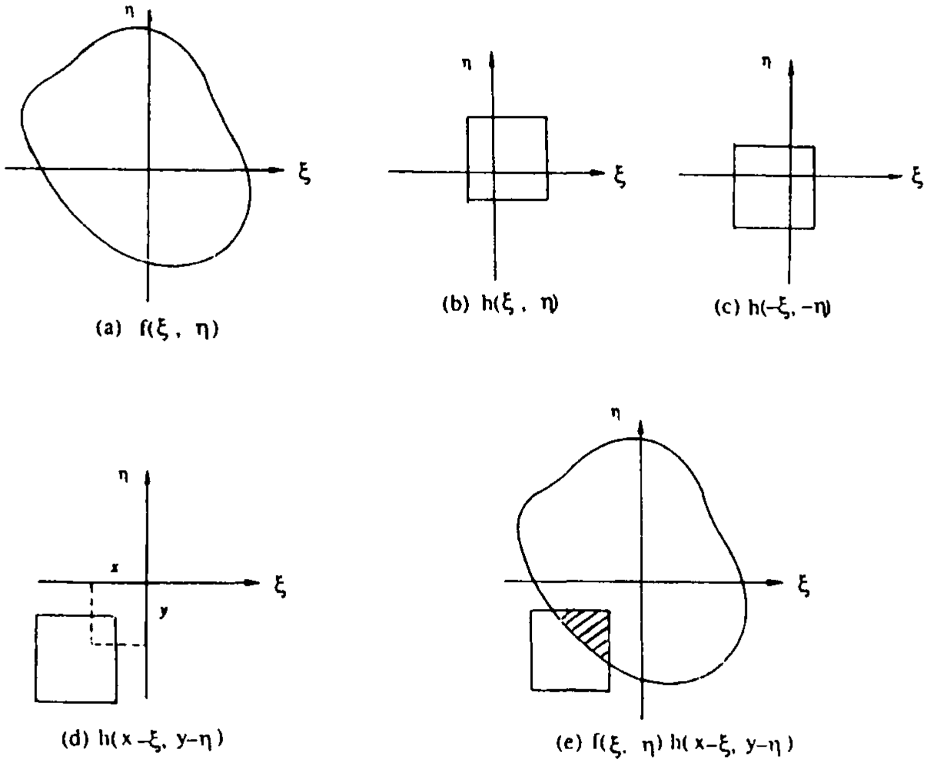


FIGURE B.6 Graphical example of a two-dimensional convolution. (a) $f(\xi, \eta)$; (b) $h(\xi, \eta)$; (c) $h(-\xi, -\eta)$; (d) $h(x - \xi, y - \eta)$; (e) $f(\xi, \eta)h(x - \xi; y - \eta)$.

or

$$g(x, y) = \iint_{-\infty}^{\infty} f(\xi, \eta)h(x - \xi, y - \eta)\} \, d\xi \, d\eta \tag{B.23}$$

where $h(x - \xi, y - \eta) = \mathcal{O}\{\delta(x - \xi, y - \eta)\}$, the impulse response of the two-dimensional system, is called the *point-spread function*. This superposition integral then reduces to the special case called the *convolution integral*, convolving the input with the impulse response. In discrete form,

$$g(x, y) = \sum_{\xi, \eta = -\infty}^{+\infty}\sum f(\xi, \eta)h(x - \xi, y - \eta) \tag{B.24}$$

Symbolically, the convolution operation $f(x, y)$, convolving with $h(x, y)$, can be expressed as

$$g(x, y) = f(x, y) * h(x, y) \tag{B.25}$$

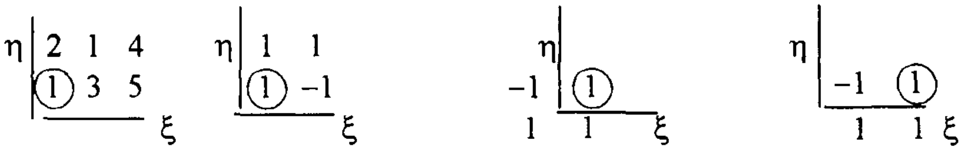
The convolution integral is symmetrical in the sense that

$$g(x, y) = \iint_{-\infty}^{\infty} f(x - \xi, y - \eta)h(\xi, \eta) \, d\xi \, d\eta \tag{B.26}$$

meaning that $h(x, y)$ convolves with $f(x, y)$.

Figure B.6 provides us with a visualization of the convolution process. In Figure B.6a and b, the input function $f(x, y)$ and impulse response are plotted in the dummy coordinate system (ξ, η). In Figure B.6c the coordinates of the impulse response are reversed. The impulse response is offset by the spatial values (x, y) shown in Figure B.6d. The shaded region in Figure B.6e shows the integrand product of the convolution integral of Eq. (B.23). The integral over this region gives the value of $g(x, y)$ at the offset coordinate (x, y). The complete function $g(x, y)$ could, in effect, be completed by sequentially scanning the reversed, offset impulse response across the input function while integrating the overlapping region.

Example. Figure B.7 shows a 2 × 2 and 3 × 2 arrays $h(\xi, \eta)$ and $f(\xi, \eta)$, where the circled elements are at the origin. The precedure involved in obtaining the convolution of these two arrays are shown step-by-step in the figure.



(a) $f(\xi, \eta)$ (3×2 array)

(b) $h(\xi, \eta)$ (2×2 array)

(c) $h(-\xi, -\eta)$

(d) $h(1-\xi, -\eta)$ $h(-\xi, -\eta)$ shifted by $x = 1$, and $y = 0$

FIGURE B.7 Example showing the step-by-step operations of the convolution of two arrays.

When the circled element of the shifted version of $h(-\xi, -\eta)$ shifts to the (0, 0), (1, 0), (1, 1), and (2, 2) elements of $f(\xi, \eta)$, the convolved results are, respectively, as follows:

$$\text{At element } (0, 0): \quad \begin{vmatrix} 0 & 1 \\ 0 & 0 \end{vmatrix} * \begin{vmatrix} -1 & 1 \\ 1 & 1 \end{vmatrix} = 1$$

$$\text{At element } (1, 0): \quad \begin{vmatrix} 1 & 3 \\ 0 & 0 \end{vmatrix} * \begin{vmatrix} -1 & 1 \\ 1 & 1 \end{vmatrix} = 2$$

$$\text{At element } (1, 1): \quad \begin{vmatrix} 2 & 1 \\ 1 & 3 \end{vmatrix} * \begin{vmatrix} -1 & 1 \\ 1 & 1 \end{vmatrix} = 3$$

and

$$\text{At element } (2, 2): \quad \begin{vmatrix} 0 & 0 \\ 1 & 4 \end{vmatrix} * \begin{vmatrix} -1 & 1 \\ 1 & 1 \end{vmatrix} = 5$$

By following this procedure, the result obtained by convolving $h(-\xi, -\eta)$ with $f(\xi, \eta)$ is shown below:

$$\begin{array}{|ccccc} 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 5 & 5 & 0 \\ 3 & 3 & 11 & 1 & 0 \\ 1 & 2 & 2 & -5 & 0 \\ \hline \end{array}$$

In general, if $h(\xi, \eta)$ is of size $(M_1 \times N_1)$ and $f(\xi, \eta)$ is of size $(M_2 \times N_2)$, the convolution of these two arrays,

$$g(\xi, \eta) = f(\xi, \eta * h(\xi, \eta)$$

will yield an array of size $(M_1 + M_2 - 1) \times (N_1 + N_2 - 1)$. In the case shown in the example, $M_1 = 2$, $N_1 = 2$, $M_2 = 3$, and $N_2 = 2$, the result of the convolution $g(\xi, \eta)$ is an array of $(2 + 3 - 1) \times (2 + 2 - 1)$, or 4×3.

B.6 DIFFERENTIAL OPERATORS*

Edge detection in images is an important task in image processing and is commonly accomplished by performing a spatial differentiation of the image field followed by a threshold operation to determine the points of steep amplitude change. The horizontal and vertical spatial derivatives of an image function $f(x, y)$ are, respectively,

$$\begin{aligned} d_x &= \frac{\partial f(x, y)}{\partial x} \\ d_y &= \frac{\partial f(x, y)}{\partial y} \end{aligned} \tag{B.27}$$

The direction-oriented spatial derivative of the image field along a vector direction r subtending an angle ϕ with the horizontal axis is given by

$$\nabla\{f(x, y)\} = \frac{\partial f(x, y)}{\partial r} = \frac{\partial f}{\partial x}\frac{dx}{dr} + \frac{\partial f}{\partial y}\frac{dy}{dr} \tag{B.28}$$

or

$$\nabla\{f(x, y)\} = d_x \cos\phi + d_y \sin\phi \tag{B.29}$$

with dx/dr and dy/dr represented by $\cos\phi$ and $\sin\phi$, respectively. The magnitude of the directional derivative (or gradient) is then

$$|\nabla\{f(x, y)\}| = \sqrt{d_x^2 + d_y^2} \tag{B.30}$$

Let us further denote the spatial second derivatives in the horizontal and vertical directions by

$$d_{xx} = \frac{\partial^2 f(x, y)}{\partial x^2} \tag{B.31}$$

$$d_{yy} = \frac{\partial^2 f(x, y)}{\partial y^2} \tag{B.32}$$

The second-order directional derivative can then be

$$\begin{aligned} \nabla^2 f(x, y) &= \frac{\partial^2 f(x, y)}{\partial r^2} = \frac{\partial}{\partial r}(\nabla f) \\ &= d_{xx} + 2d_x \times d_y + d_{yy} \end{aligned} \tag{B.33}$$

*See Pratt, W. K., *Ditigal Image Processing*, Wiley, New York, 1991, pp. 9–10.

For a high-order derivative (say, an nth-order derivative),

$$\nabla^n\{f(x,y)\} = \sum_{i=0}^{n}\binom{n}{i}\frac{\partial^i f(x,y)}{\partial x^i}\frac{\partial^{n-i}f(x,y)}{\partial y^{n-i}} \tag{B.34}$$

Note that the notation $\nabla^2 f(x,y)$ in the literature can mean either a second-order derivative or a laplacian. They differ by a cross-product term, $2d_x d_y$. The laplacian operator is not a function of spatial direction, but a scalar quantity.

B.7 PRELIMINARIES OF SOME METHODS USED FREQUENTLY IN IMAGE PREPROCESSING

Numerical Integration

For the one-dimensional image (or signal) shown in Figure B.8, where $g_i, i = 0, 1, \ldots, M-1$, are sampled at equal distances along the x direction,

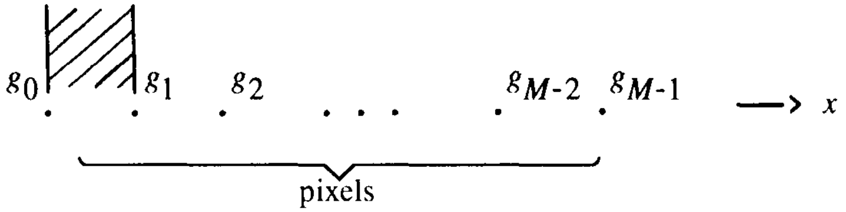


FIGURE B.8

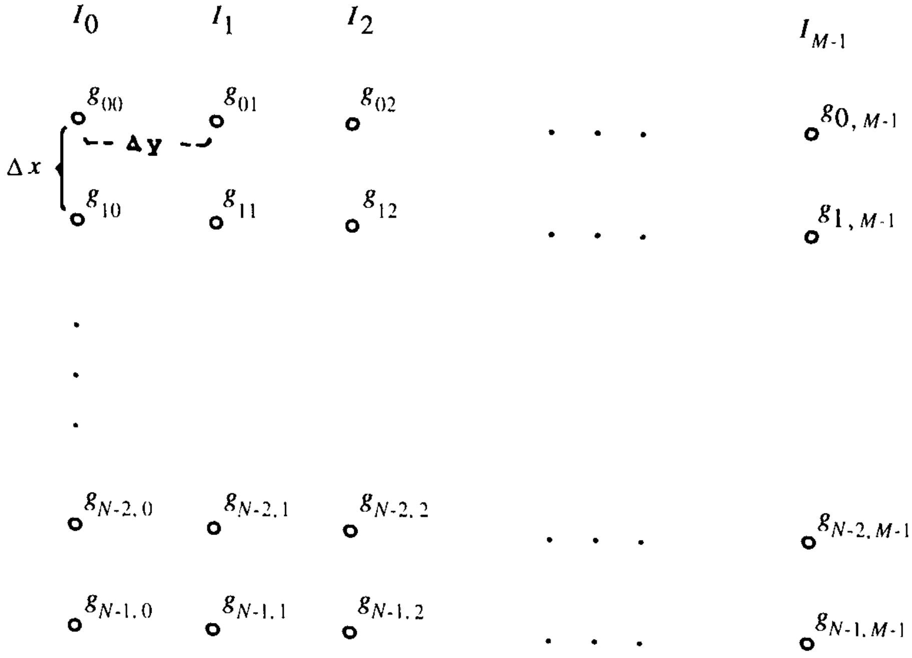


FIGURE B.9

and for the two-dimensional image (see Figure B.9), the integration $\int g(x)\,dx$ and the double integral $\int\int g(x,y)\,dx\,dy$ can be performed in any of the following ways:

1. Rectangular rule

   (a) Single integration

$$I = \int_0^{M-1} g\,dx = \left(\sum_{i=1}^{M-1} g_i\right)\Delta x \tag{B.35}$$

   or

$$I = \left(\sum_{i=0}^{M-2} g_i\right)\Delta x \tag{B.36}$$

   Depending on whether g_i or g_{i+1} is chosen for the computation. For consecutive pixels, $\Delta x = 1$.

   (b) Double integration

$$\begin{aligned} I_0 &= \Delta x \sum_{i=0}^{N-2} g_{i0} \qquad \text{along the column (single integration)} \\ I_1 &= \Delta x \sum_{i=0}^{N-2} g_{i1} \qquad \text{along the column (single integration)} \\ &\vdots \\ I_{M-1} &= \Delta x \sum_{i=0}^{N-2} g_{i,M-1} \end{aligned} \tag{B.37}$$

   or

$$I_j = \Delta x \sum_{i=0}^{N-2} g_{ij} \qquad j = 0, 1, 2, \ldots, M-1 \tag{B.38}$$

   Then

$$I_1 = \Delta y \sum_{j=0}^{M-2} I_j \tag{B.39}$$

2. Trapezoidal rule

   (a) Single integration

$$I = \int_0^{M-1} g\,dx = \frac{\Delta x}{2} \sum_{i=0}^{M-2} (g_i + g_{i+1}) \tag{B.40}$$

(b) Double integration

$$I_j = \frac{\Delta x}{2} \sum_{i=0}^{N-2} (g_{i,j} + g_{i+1,j}) \tag{B.41}$$

along the columns $j = 0, 1, 2, \ldots, M-1$; increment = 1

$$I_I = \frac{\Delta y}{2} \sum_{j=0}^{M-2} (I_j + I_{j+1}) \tag{B.42}$$

3. Simpson's $\frac{1}{3}$ rule

(a) Single integration

$$I = \int_0^{M-1} g\,dx = \frac{\Delta x}{3} \sum_{i=0}^{M-3} (g_i + 4g_{i+1} + g_{i+2}) \tag{B.43}$$

Note that the increment used in this case is 2.

(b) Double integration

$$I_j = \frac{\Delta x}{3} \sum_{i=0}^{N-3} (g_{i,j} + 4g_{i+1,j} + g_{i+2,j}) \qquad \text{increment} = 2 \tag{B.44}$$

$$I_I = \frac{\Delta y}{3} \sum_{j=0}^{M-3} (I_j + 4I_{j+1} + I_{j+2}) \qquad \text{increment} = 2 \tag{B.45}$$

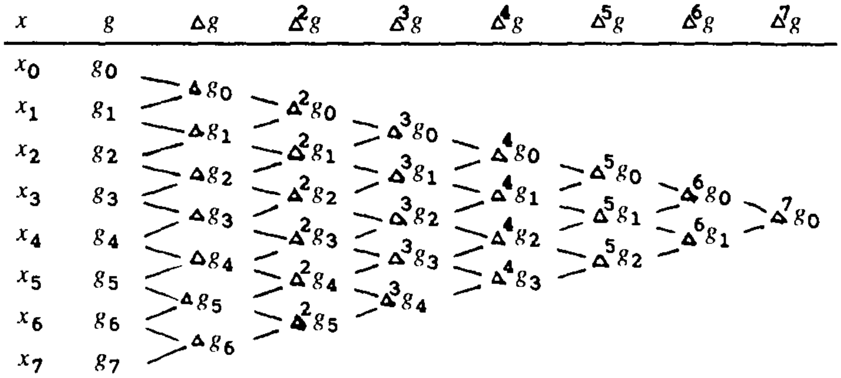
Finite Differences

The method of finite differences is another tool that is useful to our image processing. Forward differences of different orders are

$$\begin{aligned}
\Delta g_i &= g_{i+1} - g_i \\
\Delta^2 g_i &= \Delta g_{i+1} - \Delta g_i \\
&= g_{i+2} - 2g_{i+1} + g_i \\
\Delta^3 g_i &= g_{i+3} - 3g_{i+2} + 3g_{i+1} - g_i \\
&\vdots \\
\Delta^n g_i &= \sum_{j=0}^{n} (-1)^j \binom{n}{j} g_{n-j+i}
\end{aligned} \tag{B.46}$$

where

$$\binom{n}{j} = C_j^n = \frac{n!}{j!(n-j)!} \tag{B.47}$$



x g Δg $\Delta^2 g$ $\Delta^3 g$ $\Delta^4 g$ $\Delta^5 g$ $\Delta^6 g$ $\Delta^7 g$
x_0 g_0
x_1 g_1
x_2 g_2
x_3 g_3
x_4 g_4
x_5 g_5
x_6 g_6
x_7 g_7
Δg_0 Δg_1 Δg_2 Δg_3 Δg_4 Δg_5 Δg_6
$\Delta^2 g_0$ $\Delta^2 g_1$ $\Delta^2 g_2$ $\Delta^2 g_3$ $\Delta^2 g_4$ $\Delta^2 g_5$
$\Delta^3 g_0$ $\Delta^3 g_1$ $\Delta^3 g_2$ $\Delta^3 g_3$ $\Delta^3 g_4$
$\Delta^4 g_0$ $\Delta^4 g_1$ $\Delta^4 g_2$ $\Delta^4 g_3$
$\Delta^5 g_0$ $\Delta^5 g_1$ $\Delta^5 g_2$
$\Delta^6 g_0$ $\Delta^6 g_1$
$\Delta^7 g_0$

FIGURE B.10

When extended to a two-dimensional problem [i.e., for the two-variable function $g = f(x, y)$ shown in Figure B.11], the forward and backward difference quotients with respect to x at the point (i, j) (refer to Figure B.11) are

$$g_x = \frac{g_{i+1,j} - g_{i,j}}{\Delta x} \quad \text{forward difference} \tag{B.48}$$

$$g_{\bar{x}} = \frac{g_{i,j} - g_{i-1,j}}{\Delta x} \quad \text{backward difference} \tag{B.49}$$

Those with respect to y are

$$g_y = \frac{g_{i,j+1} - g_{i,j}}{\Delta y} \quad \text{forward difference} \tag{B.50}$$

$$g_{\bar{y}} = \frac{g_{i,j} - g_{i,j-1}}{\Delta x} \quad \text{backward difference} \tag{B.51}$$

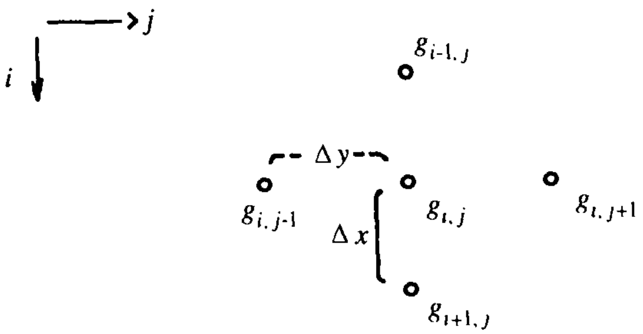


FIGURE B.11

Their corresponding finite differences of second order are

$$g_{x\bar{x}} = \frac{g_x - g_{\bar{x}}}{\Delta x} = \frac{g_{i+1,j} - 2g_{i,j} + g_{i-1,j}}{(\Delta x)^2} \tag{B.52}$$

$$g_{y\bar{y}} = \frac{g_y - g_{\bar{y}}}{\Delta y} = \frac{g_{i,j+1} - 2g_{i,j} + g_{i,j-1}}{(\Delta y)^2} \tag{B.53}$$

With the second-order finite differences shown above, the laplacian operator, $\nabla^2 g$, which is

$$\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \tag{B.54}$$

can be expressed as

$$\nabla^2 g = \frac{g_{i+1,j} - 2g_{i,j} + g_{i-1,j}}{(\Delta x)^2} + \frac{g_{i,j+1} - 2g_{i,j} + g_{i,j-1}}{(\Delta y)^2} \tag{B.55}$$

If Δx and Δy are chosen to be unity in an image file,

$$\nabla^2 g = g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1} - 4g_{i,j} \tag{B.56}$$

This is the simplified form for digital implementation of the laplacian.

So far we have discussed some of the mathematics we need for image processing. This gives you an idea as to how to manipulate the data involved in a discrete image. More mathematics will be given in the course of our future discussion where needed.

PROBLEMS

1. Work out step by step the convolution of the following two arrays.

$$\begin{array}{c|ccc} \eta & 2 & 1 & 4 \\ & \textcircled{1} & 3 & 5 \\ \hline & & & & \xi \end{array} \qquad \begin{array}{c|cc} \eta & 1 & 1 \\ & \textcircled{1} & -1 \\ \hline & & & \xi \end{array}$$

$f(\xi, \eta)$ $h(\xi, \eta)$

2. Consider the 2×2 and 4×3 arrays $h(\xi, \eta)$ and $f(\xi, \eta)$ where the circled elements are at the origin. Obtain the convolution of these two arrays.

$$\begin{array}{c|ccccc} \eta & 4 & 5 & 2 & 1 & \\ & 3 & 2 & 5 & 6 & \\ & \textcircled{1} & 2 & 4 & 5 & \\ \hline & & & & & \xi \end{array} \qquad \begin{array}{c|ccc} \eta & & & \\ & 1 & 1 & \\ & \textcircled{1} & -1 & \\ \hline & & & \xi \end{array}$$

$$f(\xi, \eta) \qquad\qquad h(\xi, \eta)$$

3. Prove that the generalized form of the forward difference of nth order is

$$\Delta^n g_i = \sum_{j=0}^{n} (-1)^j \binom{n}{j} g_{n-j+i}$$

4. Generalize the expression for a high-order (nth order) derivative:

$$\nabla^n\{f(x, y)\} = \sum_{i=0}^{n} \binom{n}{j} \frac{\partial^i f(x, y)}{\partial x^i} \times \frac{\partial^{n-1} f(x, y)}{\partial y^{n-i}}$$

Appendix C: Digital Image Fundamentals

C.1 Sampling and Quantization of an Image

The image model $f(x, y)$ is a two-dimensional light luminance function of spatial variables x and y. If this function $f(x, y)$ is digitized spatially, it is referred to as *image sampling*. When the image is digitized in amplitude, it is called *gray-level quantization*.

In image processing, images are assumed for mathematical simplicity over a rectangular region. Digitization of the spatial coordinates of $f(x, y)$ gives an $N \times N$ image array as shown in Figure C.1. Each element of this image array is referred to as an image element or picture element (pixel). N, which is the number of pixels in each row, or in each column, is frequently chosen to be 2^n, an integer power of 2. Readers of Chapter 14 will know the reason for such a choice. G, which is the number of gray levels used in representing the image function at the pixel (x, y) and $x, y = 0, 1, 2, \ldots, N - 1$, is also chosen to be an integer power of 2, namely 2^m. With such a choice of N and m, the total number of bits required to store a digitized image of size $N \times N$ image array and of 2^m gray levels for its representation is

$$b = N \times N \times m$$

If $N = 512$ and $m = 8$ (256 gray levels), then based on the foregoing argument, to transmit a picture digitally at a TV rate, we have to have a data transmssion rate of

$$512 \times 512 \times 8 \times 30 \times 24 = 1510\,\text{MB/s}$$

$$\begin{vmatrix} f(0,0) & f(0,1) & \cdots & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & \cdots & f(1,N-1) \\ f(2,0) & f(2,1) & \cdots & \cdots & f(2,N-1) \\ \vdots & & & & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & \cdots & f(N-1,N-1) \end{vmatrix}$$

FIGURE C.1 $N \times N$ image array.

if 30 frames per second and 24 multiplex channels are assumed. The frame-grabbing time will be approximately

$$\frac{512 \times 512}{25 \times 10^6} \simeq 0.0105\,\mathrm{s} + \text{overhead for other gating time}$$

or less than one-thirtieth of a second. In general, the quality of the image depends on the spatial resolution and on the number of gray levels to be used for each pixel. If the resolution is too low, the image quality deteriorates rapidly and a checkerboard effect is noticeable. If fewer bits are used for the gray-level representation in an image (even if the spatial resolution is kept fairly high), false contouring effects in the smooth background area become more and more pronounced. This effect increases sharply as the number of gray levels used for the image representaion decreases futher. As a rule, a minimum of 64 gray levels should be used. For applications where better image quality is expectd, 128 or 256 gray levels are preferred. A 512 × 512 spatial resolution is appropriate for image reprsentation, but higher resolution (e.g., 1024 × 1024) is utilized for high-precision nondestructive industrial measurements.

It should be mentioned here that the best values of N and m to be chosen for images are very dependent on the images themselves: that is, images with relatively few details, images containing a large amount of detailed information, images with an intermediate number of details, and so on. However, several empirical conclusions can be drawn: (1) the qualtiy of images tends to increase, in general, as N and m are increased; and (2) isopreference* curves in the Nm plane tend to become more vertical as the details in the image increase (i.e., there is not much change in image quality even when the number of gray levels continues to increase).

So far uniform sampling and quantization have been discussed. But in many cases, an adaptive scheme of quantization is more suitable, due to

*An *isopreference curve* is one in which the points in the Nm plane represent images of equal subjective quality.

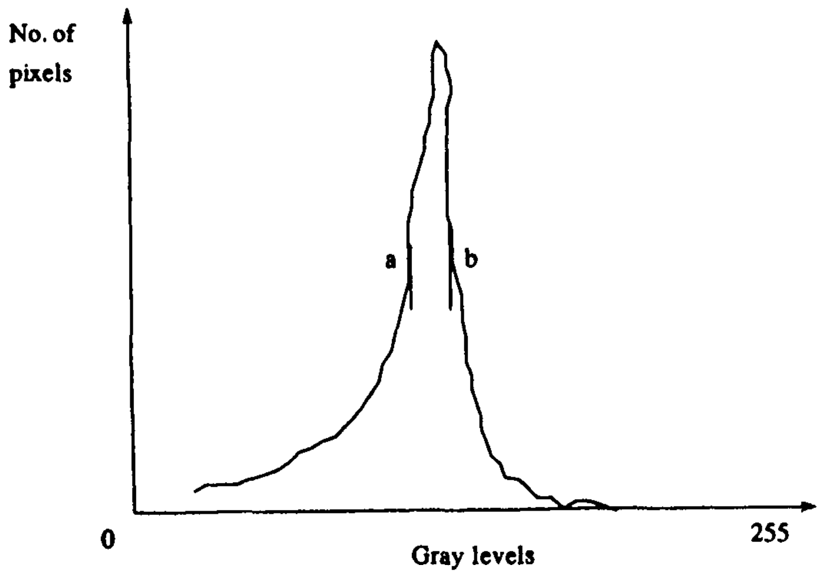


FIGURE C.2 Imaging geometry.

characteristics of the image under discussion. For example, in the neighborhood of a sharp gray-level transition, finer spatial sampling is recommended; in relatively smooth regions, coase sampling is sufficient. In case a large number of pixels fall in a certain range of gray levels, then for that range, the quantization level should be finely spaced, and outside that gray-level range, coarsely spaced quantization can be used. From the histogram shown in Figure C.2, it can be seen that a lot of pixels concentate in the range between points *a* and *b*. If we would like to have more details available from the image, a wider range of gray levels should be allocated to that part. This technique of allocating a nonlinear range of gray levels to take care of a high concentration of pixels with similar gray levels is usually referred to as *tapered quantization*.

C.2 IMAGING GEOMETRY*

C.2.1 World-Point to Image-Point Transform

As discussed in Appendix B, an image can be represented as a function of five variables as $C(x, y, z, t, \lambda)$. As mentioned previously, the image frame obtained is the spectral response of the sensor, or the brightness response of a human observer in the case of a human vision. This response usually does not change with time. Hence the 5-tuple function becomes $C(x, y, z)$.

*Much of the material in this section is from Fu, Gonzalez, and Lee (1987).

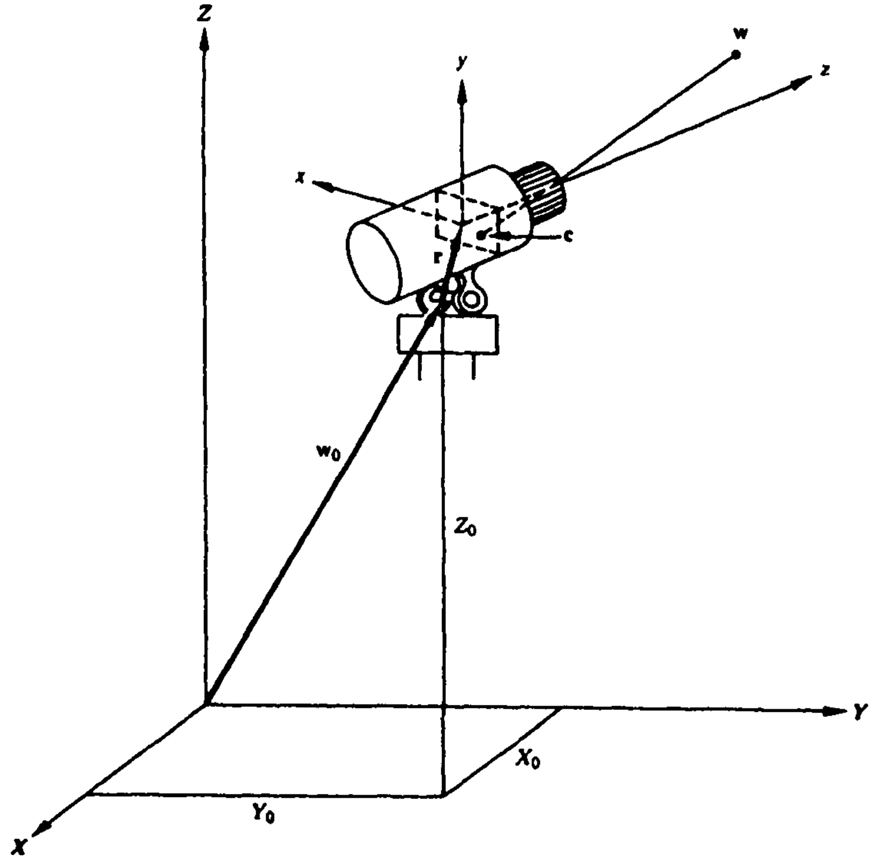


FIGURE C.3 Imaging geometry with two coordinate systems. (From Fu, Gonzalez, and Lee, 1987.)

In image processing, our work focuses mainly on the plane images. Even in the case of stereo vision, we are still working on plane images. What we work on in stereo vision is the combination of two (instead of one) plane images, which are taken from two sensors a fixed distance apart. These plane images are projections of three-dimensional points onto an image plane. Figure C.3 gives the imaging geometry with two coordinate systems, one of which is the world coordinate system (X, Y, Z) used to locate the three-dimensional points (denoted by w), and the other, which is the camera coordinate system (x, y, z) and the image point, which is denoted by c.

The vector $\mathbf{w}_0$ denotes the offset of the center of the gimbal from the origin of the (X, Y, Z) system, and the vector $\mathbf{r}$ denotes the offset of the center of the imaging plane with respect to the gimbal center. The camera is supposed to be able to pan about the X axis by an angle θ, and to tilt about the Z axis by an angle α during the optical setting.

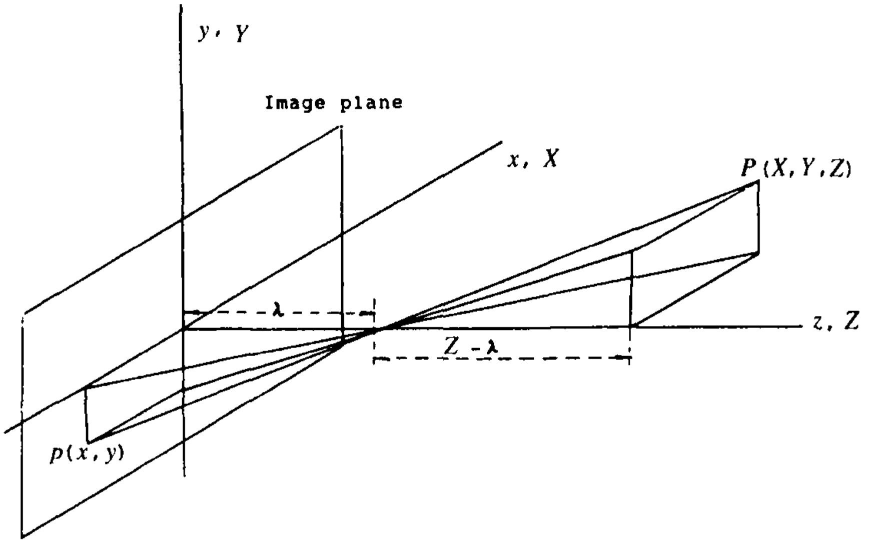


FIGURE C.4 Object-image point transformation.

Figure C.4 shows the object-image point transformation [Gonzalez and Wintz (1987)]. Point $p(x, y)$ is the projection of the point $P(X, Y, Z)$ onto the image plane; p in the figure is the point of projection and λ is the focal length of the lens. By a similar triangle relationship we have

$$\frac{x}{\lambda} = -\frac{X}{Z - \lambda} \tag{C.1}$$

and

$$\frac{y}{\lambda} = -\frac{Y}{Z - \lambda} \tag{C.2}$$

The negative signs in Eqs. (C.1) and (C.2) are used to indicate that x and X, y and Y are actually on opposite sides of the Z axis, due to the inverted object-image property. It follows from (C.1) and (C.2) that

$$x = \frac{\lambda X}{\lambda - Z} \tag{C.3}$$

and

$$y = \frac{\lambda Y}{\lambda - Z} \tag{C.4}$$

To correlate the object point with the image point, let us introduce the concept of homogeneous coordinates. Homogeneous coordinates for the object point (X, Y, Z) in three-dimensional space are represented by the 4×1 vector (kx, ky, kz, k), where k is an arbitrary nonzero constant.

Converting a point represented as a 4×1 vector from a homogeneous coordinate representation back to the world coordinate system can be accomplished by dividing the first three homogeneous coordinates by the fourth. This process can be geenralized to the conversion of a point represented as an $n \times 1$ vector from a homogeneous coordinate representation to its physical coordinates of dimension $n - 1$. For a point in the world coordinate system,

$$\mathbf{w} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{C.5}$$

Its homogeneous coordinate representation is given as

$$\mathbf{w}_h = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} \tag{C.6}$$

where k is an arbitrary, nonzero constant. This point (X, Y, Z) projects onto the camera image plane at point c, where

$$\mathbf{c} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \dfrac{\lambda X}{\lambda - Z} \\ \dfrac{\lambda Y}{\lambda - Z} \\ \dfrac{\lambda Z}{\lambda - Z} \end{bmatrix} \tag{C.7}$$

or

$$\mathbf{c} = \begin{bmatrix} KkX \\ KkY \\ KkZ \end{bmatrix} \tag{C.8}$$

where $K = \lambda / k(\lambda - Z)$. Dividing every item of (C.8) by K and using $1/K$ as the fourth item of the vector gives the homogeneous counterpart $\mathbf{c}_h$ of the image point, in a homogeneous coordinate system as

$$\mathbf{c}_h = \begin{bmatrix} kX \\ kY \\ kZ \\ k - \dfrac{kZ}{\lambda} \end{bmatrix} \tag{C.9}$$

which, in turn, can be put in the following form:

$$\mathbf{c}_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} \tag{C.10}$$

or

$$\mathbf{c}_h = \mathbf{P} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} \tag{C.11}$$

where **P** is a transformation matrix, which relates $\mathbf{c}_h$ and $\mathbf{w}_h$ and is the matrix for transformation from a world coordiante system to a camera image plane coordinate system. It is known as the perspective transformation matrix. Knowing **c** and **P**, we can find the world coordinate point from the known image point, and vice versa, with Eqs. (C.7), (C.9), (C.10), and (C.11) and $\mathbf{P}^{-1}$, which can be computed from **P**.

Note that mapping of a three-dimensional object onto the image plane is a many-to-one transformation. We cannot recover the three-dimensional world point from its image unless we know something more about the point (e.g., its z coordinate) or if we have images taken from two cameras that are a fixed distance apart. As depicted in Figure C.3, the image plane can be in a very perturbed orientation with respect to the world coordinate system. It consists of a combination of translation, panning, and tilting of the image plane about the global coordinate axes, other than just rotations about the image plane centric system, thus making the coordinate system transformation complicated. The overall approach is to bring the camera and world coordiante systems into alignment by a set of transformations. After this has been accomplished, we simply apply the perspective transformation to obtain the image plane coordinates of any given world point.

C.2.2 Translation and Scaling

Translation of a point in the (X, Y, Z) system from one location to another by a displacement of (X_0, Y_0, Z_0) is equivalent to the coordinate system displaced to a new coordinate position, which is offset from the old coordinate system by

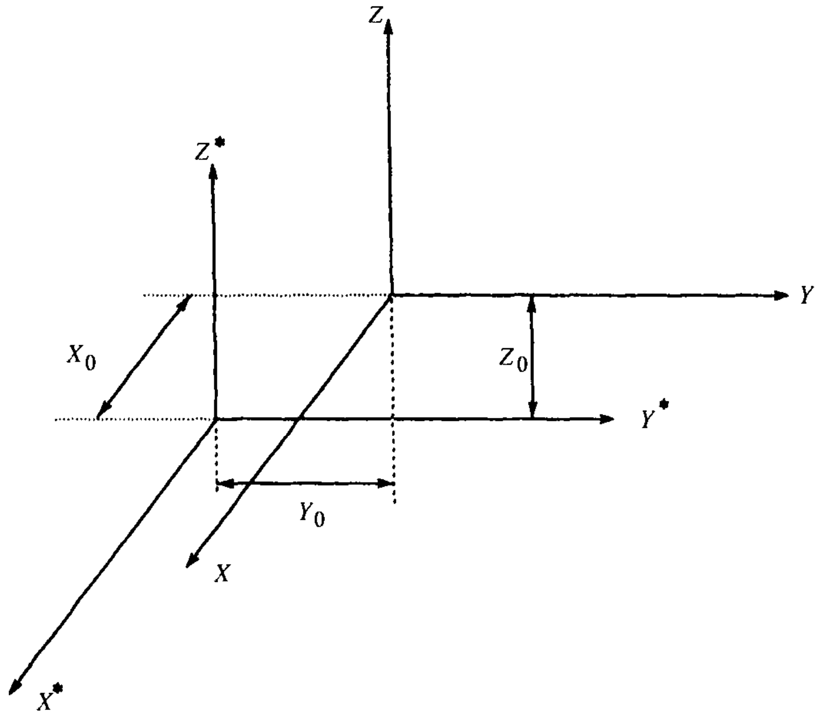


FIGURE C.5 Coordinate system XYZ and its counterpart when displaced by X_0, Y_0, and Z_0.

displacements $-X_0$, $-Y_0$, and $-Z_0$ with the point fixed (see Figure C.5). For the same point, the two system coordinates (X, Y, Z) and (X^*, Y^*, Z^*) are related by

$$\begin{aligned} X^* &= X + X_0 \\ Y^* &= Y + Y_0 \\ Z^* &= Z + Z_0 \end{aligned} \tag{C.12}$$

Expressed in matrix form, we have

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{C.13}$$

or

$$\mathbf{V}^* = \mathbf{AV} \tag{C.14}$$

where **V** is the original coordinate vector of the image point and is represented by $(X, Y, Z, 1)^T$; $\mathbf{V}^*$ is its coordinate vector after transformation and is $(X^*, Y^*, Z^*, 1)^T$. Both vectors are 4 × 1 in dimension.

$$A = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.15}$$

is the 4 × 4 transformation matrix for translation. Similarly, for *scaling*, the coordinates before and after scaling are related by

$$\begin{aligned} X^* &= s_x X \\ Y^* &= s_y Y \\ Z^* &= s_z Z \end{aligned} \tag{C.16}$$

or

$$\mathbf{V}^* = \mathbf{S}\mathbf{V} \tag{C.17}$$

where $\mathbf{V}^*$ and **V** are, respectively,

$$\mathbf{V}^* = \begin{bmatrix} X^* \\ Y^* \\ Z^* \end{bmatrix} \quad \text{and} \quad \mathbf{V} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{C.18}$$

and

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.19}$$

is the transformation matrix for scaling.

C.2.3 Rotation

The transformation for the camera mount tilt and camera mount pan are complicated because these operations are done by rotating a given point about an arbirary point in space. This involves three transformations: one translates the image plane coordinate system to the global coordinate system origin (about which the pan and tilt rotations occur); a second performs pan rotation about the X axis and/or tilt rotation about the Z axis; and the third translates the point back to its original position.

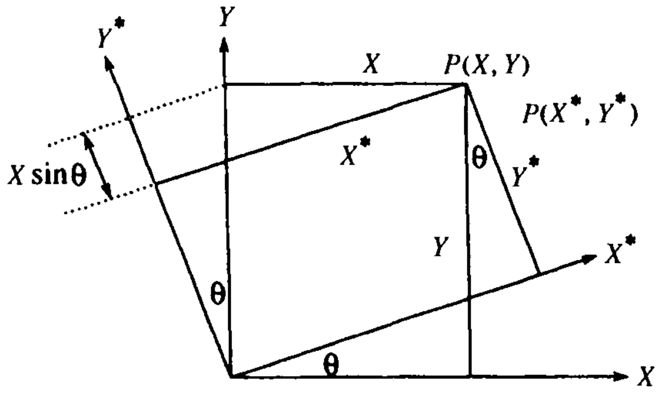


FIGURE C.6 Coordinate system XYZ and its counterpart when rotated about the Z-axis by an angle θ.

Refer to Figure C.6 for the tilt rotation of a point about the Z axis by an angle θ; we have

$$\begin{aligned} X^* &= X\cos\theta + Y\sin\theta \\ Y^* &= -X\sin\theta + Y\cos\theta \\ Z^* &= Z \end{aligned} \tag{C.20}$$

Put in matrix form, we have

$$\mathbf{V}^* = \mathbf{R}_\theta \mathbf{V} \tag{C.21}$$

where $\mathbf{R}_\theta$ is the transformation matrix for tilt rotation about the Z axis by an angle θ and is

$$\mathbf{R}_\theta \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.22}$$

Similarly, a transformation matrix can be derived for the pan rotation of a point about the X axis by an angle α as (see Figure C.7)

$$\mathbf{R}_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.23}$$

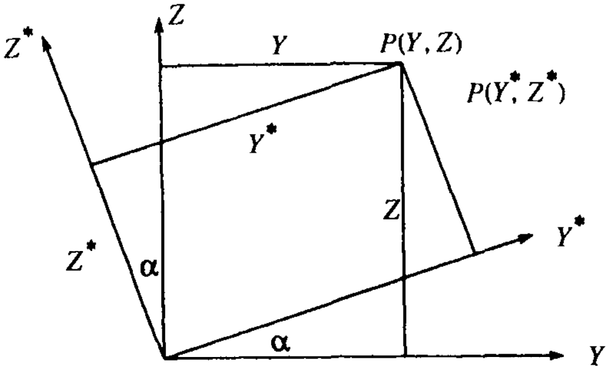


FIGURE C.7 Coordinate system XYZ and its counterpart when rotated about the X-axis by an angle α.

and that for the rotation of a point about the Y axis by an angle β as (see Figure C.8)

$$\mathbf{R}_\beta = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.24}$$

Let us get back to the problem depicted by Figure C.3 to compute the corresponding points in the world coordinate system from an image plane. Given a point in the image plane, say (x_0, y_0), if we add an arbitrary value z to these coordinates, we can form a 3×1 vector $\mathbf{c}$:

$$\mathbf{c} = \begin{bmatrix} x_0 \\ y_0 \\ z \end{bmatrix} \tag{C.25}$$

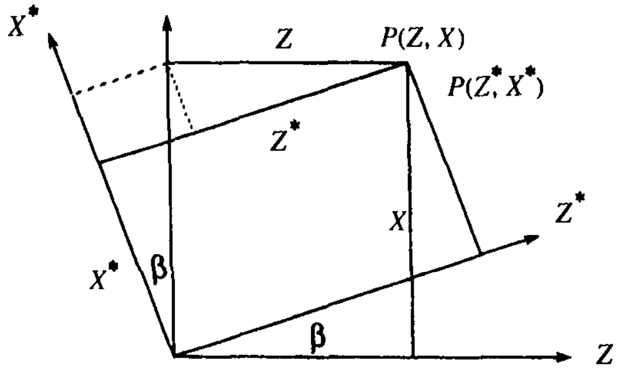


FIGURE C.8 Coordinate system XYZ and its counterpart when rotated about the Y-axis by an angle β.

The corresponding cartesian world system coordinates in homogeneous form can be computed as

$$\mathbf{W}_h = \mathbf{P}^{-1}\mathbf{c}_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{bmatrix} \begin{bmatrix} kx_0 \\ hy_0 \\ kz \\ k \end{bmatrix} \tag{C.26}$$

$$= \begin{bmatrix} kx_0 \\ ky_0 \\ kz \\ \frac{kz}{\lambda} + k \end{bmatrix} \tag{C.27}$$

where the 4 × 4 transformation matrix $\mathbf{P}^{-1}$ can be found without much difficulty and $(kx_0, ky_0, kz, k)^T$ is the homogeneous part of $\mathbf{c}$. Dividing the first, second, and third components of the vector in Eq. (C.27) by the fourth gives

$$\mathbf{w} = \begin{bmatrix} \dfrac{\lambda x_o}{\lambda + z} \\ \dfrac{\lambda y_o}{\lambda + z} \\ \dfrac{\lambda z_o}{\lambda + z} \end{bmatrix} \tag{C.28}$$

from which the X component of the point in the world coordinate system can be expressed as

$$X = \frac{\lambda x_0}{\lambda + z} \tag{C.29}$$

Similarly,

$$Y = \frac{\lambda y_0}{\lambda + z} \tag{C.30}$$

and

$$Z = \frac{\lambda z}{\lambda + z} \tag{C.31}$$

Solving for the arbitrary assigned parameter z, we have

$$z = \frac{\lambda Z}{\lambda - Z} \tag{C.32}$$

Substituting (C.32) into (C.29) and (C.30), we obtain

$$X = \frac{x_0}{\lambda}(\lambda - Z) \tag{C.33}$$

$$Y = \frac{Y_0}{\lambda}(\lambda - Z) \tag{C.34}$$

From the observations above, it can be noted that mapping of a three-dimensional object onto the image plane is a many-to-one transformation. We cannot recover the three-dimensional world point from its image unless we know something more about the three-dimensional point (e.g., its z coordinates) or if we have images taken from two cameras that are at a fixed distance apart.

Taking into consideration the displacement of the gimbal center from the origin, the pan of the x axis with respect to the Z axis, the tilt of the z axis with respect to the Y axis, and the displacement of the image plane by $\mathbf{r}[\mathbf{r} = (r_1, r_2, r_3)^T]$ with respect to the gimbal center as shown in Figure C.3, the homogeneous representation in the image plane coordinate system, $\mathbf{c}_h$, relates to $\mathbf{w}_h$, the homogeneous representation of the same point in the world coordinate system, as follows:

$$\mathbf{c}_h = \mathbf{PCRGw}_h \tag{C.35}$$

where $\mathbf{P}$ is the perspective transformation matrix that relates $\mathbf{c}_h$ and $\mathbf{w}_h$ as represented by Eq. (C.11) and (C.6). $\mathbf{C}$, $\mathbf{R}$, and $\mathbf{G}$ are the transformation matrices added to the conversion between $\mathbf{c}_h$ and $\mathbf{w}_h$ when rotation and translation of the gimbal center are taken into consideration. Among them, $\mathbf{C}$ is the translational transformation matrix for $\mathbf{r} = [-r_1, -r_2, -r_3]^T$ and is

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.36}$$

R is a product of $\mathbf{R}_\theta$, the pan of the x axis with respect to the Z axis, and $\mathbf{R}_\alpha$, the tilt of the z axis with respect to the Y axis, and therefore

$$R = R_\theta R_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta\cos\alpha & \cos\theta\cos\alpha & \sin\alpha & 0 \\ \sin\theta\sin\alpha & -\cos\theta\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.37}$$

G is the transformation matrix to translate the origin of the world coordinate system to the location of the gimbal center, and is

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.38}$$

The matrix multiplications are complicated. When worked out step by step, we have

$$\mathbf{RG} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & -X_0\cos\theta - Y_0\sin\theta \\ -\sin\theta\cos\alpha & \cos\theta\cos\alpha & \sin\alpha & X_0\sin\theta\cos\alpha - Y_0\cos\theta\cos\alpha - Z_0\sin\alpha \\ \sin\theta\sin\alpha & -\cos\theta\sin\alpha & \cos\alpha & -X_0\sin\theta\sin\alpha + Y_0\cos\theta\sin\alpha - Z_0\cos\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.39}$$

$$\mathbf{CRG} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & -X_0\cos\theta - Y_0\sin\theta - r_1 \\ -\sin\theta\cos\alpha & \cos\theta\cos\alpha & \sin\alpha & X_0\sin\theta\cos\alpha - Y_0\cos\theta\cos\alpha - Z_0\sin\alpha - r_2 \\ \sin\theta\sin\alpha & -\cos\theta\sin\alpha & \cos\alpha & -X_0\sin\theta\sin\alpha + Y_0\cos\theta\sin\alpha - Z_0\cos\alpha - r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.40}$$

$$\mathbf{PCRG} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & -X_0\cos\theta - Y_0\sin\theta - r_1 \\ -\sin\theta\cos\alpha & \cos\theta\cos\alpha & \sin\alpha & X_0\sin\theta\cos\alpha - Y_0\cos\theta\cos\alpha - Z_0\sin\alpha - r_2 \\ \sin\theta\sin\alpha & -\cos\theta\sin\alpha & \cos\alpha & -X_0\sin\theta\sin\alpha + Y_0\cos\theta\sin\alpha - Z_0\cos\alpha - r_3 \\ \dfrac{-\sin\theta\sin\alpha}{\lambda} & \dfrac{\cos\theta\sin\alpha}{\lambda} & \dfrac{-\cos\alpha}{\lambda} & \left(\dfrac{X_0\sin\theta\sin\alpha - Y_0\cos\theta\sin\alpha + Z_0\cos\alpha + r_3}{\lambda}\right) + 1 \end{bmatrix} \tag{C.41}$$

Substitute the matrix above into

$$\mathbf{c}_h = \mathbf{PCRGw}_h \tag{C.42}$$

and remember that

$$\mathbf{c} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \mathbf{c}_h = \begin{bmatrix} k'x \\ k'y \\ k'z \\ k' \end{bmatrix} \tag{C.43}$$

We then have

$$\mathbf{c}_h = \begin{bmatrix} \text{first component} \\ \text{second component} \\ \text{third component} \\ \text{fourth component} \end{bmatrix} \tag{C.44}$$

where

$$\text{First component} = k[X\cos\theta + Y\sin\theta - X_0\cos\theta - Y_0\sin\theta - r_1]$$

$$\begin{aligned}\text{Second component} = k[&-X\sin\theta\cos\alpha + Y\cos\theta\cos\alpha + Z\sin\alpha \\ &+ X_0\sin\theta\cos\alpha - Y_0\cos\theta\cos\alpha - Z_0\sin\alpha - r_2]\end{aligned}$$

$$\begin{aligned}\text{Fourth component} = \frac{k}{\lambda}[&-X\sin\theta\sin\alpha + Y\cos\theta\sin\alpha - Z\cos\alpha \\ &+ X_0\sin\theta\sin\alpha - Y_0\cos\theta\sin\alpha + Z_0\cos\alpha + r_3] + k\end{aligned}$$

The third component is not involved. Dividing the first and second components of $\mathbf{c}_h$ by the fourth gives x and y:

$$x = \lambda\frac{(X - X_0)\cos\theta + (Y - Y_0)\sin\theta - r_1}{-(X - X_0)\sin\theta\sin\alpha + (Y - Y_0)\cos\theta\sin\alpha - (Z - Z_0)\cos\alpha + r_3 + \lambda} \tag{C.45}$$

and

$$y = \lambda\frac{-(X - X_0)\sin\theta\cos\alpha + (Y - Y_0)\cos\theta\cos\alpha + (Z - Z_0)\sin\alpha - r_2}{-(X - X_0)\sin\theta\sin\alpha + (Y - Y_0)\cos\theta\sin\alpha - (Z - Z_0)\cos\alpha + r_3 + \lambda} \tag{C.46}$$

This shows that a definite relation is established between the image plane and the world coordinate system if (X_0, Y_0, Z_0), (r_1, r_2, r_3), α, β, and θ are previously fixed. Expressing the mathematical relation between $\mathbf{c}_h$ and $\mathbf{w}_h$ [Eq. (C.35)] in the following general form:

$$\begin{bmatrix} c_{h1} \\ c_{h2} \\ c_{h3} \\ c_{h4} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{C.47}$$

we can use a set of image points (say, six sets of points) of known world coordinates to do the camera calibration (i.e., to compute the camera parameters).

Appendix D: Matrix Manipulation

D.1 DEFINITION

A rectangular array

$$\mathbf{A} = [a_{ij}]_{m \times n} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \tag{D.1}$$

with $a_{ij}, i = 1, \dots, m; j = 1, \dots, n$, as the elements is called a matrix of m rows and n columns, or in brief, an $m \times n$ *matrix*. The rows

$$\alpha_i = (a_{i1}, \dots, a_{in}) \qquad i = 1, \dots, m \tag{D.2}$$

are called the *row vectors* of **A**. Similarly, the columns

$$\beta_j = (a_{1j}, \dots, a_{mj}) \qquad j = 1, \dots, n \tag{D.3}$$

are called the *column vectors* of **A**.

A transposed matrix$\mathbf{A}^T$ is obtained from matrix **A** by interchanging its rows and columns. Thus

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & & a_{m2} \\ \vdots & & & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix} \tag{D.4}$$

If $m = n$, the matrix $\mathbf{A}$ is said to be a *square matrix*. The main diagonal of this $n \times n$ matrix $\mathbf{A}$ consists of the entries $a_{11}, a_{22}, \ldots, a_{nn}$. If all other entries of $\mathbf{A}$ are zero, $\mathbf{A}$ is a *diagonal matrix*. A diagonal matrix $\mathbf{A}$ is a scalar matrix if $a_{11} = a_{22} = \cdots = a_{nn}$, and is the $n \times n$ identity matrix if $a_{11} = a_{22} = \cdots = a_{nn} = 1$.

A *lower triangular matrix* is a square matrix having all elements above the main diagonal zero. An *upper triangular matrix* can be defined similarly.

D.2 MATRIX MULTIPLICATION

Let $\mathbf{A}$ be an $m \times p$ matrix and $\mathbf{B}$ be an $p \times n$ matrix; the product

$$\mathbf{C} = \mathbf{AB} \tag{D.5}$$

is an $m \times n$ matrix with (i, j)th element represented by

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} \tag{D.6}$$

Note that the two matrices should be *conformable*. That is, the two matrices may be multipled only if the number of columns of the first equals the number of rows of the second.

Premultiplying a matrix by a conformable diagonal matrix has the effect of scaling each row of the matrix by the corresponding element in the diagonal matrix:

$$\begin{bmatrix} a_{11} & & 0 \\ & a_{22} & \\ 0 & & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} \\ a_{22}b_{21} & a_{22}b_{22} \\ a_{33}b_{31} & a_{33}b_{32} \end{bmatrix} \tag{D.7}$$

In a similar manner, postmultiplying a matrix by a conformable diagonal matrix has the effect of scaling each column by the corresponding element in the diagonal matrix. Thus

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & & 0 \\ & b_{22} & \\ 0 & & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{22} & a_{13}b_{33} \\ a_{21}b_{11} & a_{22}b_{22} & a_{23}b_{33} \\ a_{31}b_{11} & a_{32}b_{22} & a_{33}b_{33} \end{bmatrix} \tag{D.8}$$

The product of two like triangular matrices produces a third matrix of like form, such as

$$\begin{bmatrix} a_{11} & & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} b_{11} & & \\ b_{21} & b_{22} & \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} & & \\ a_{21}b_{11} + a_{22}b_{21} & a_{22}b_{22} & \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{32}b_{22} + a_{33}b_{32} & a_{33}b_{33} \end{bmatrix} \tag{D.9}$$

D.3 PARTITIONING OF MATRICES

Partitioning of matrices by rows and/or columns is helpful in the manipulation of matrices that are more complex in nature or large in size. A simple example is given here for illustration. Suppose that we have a complex simultaneous equations as follows:

$$\begin{bmatrix} 5+i & 3-i \\ 6-2i & 8+4i \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 5-5i \end{bmatrix} \tag{D.10}$$

The set of equations above may be put in the following form:

$$[\mathbf{A}_r + i\mathbf{A}_i][\mathbf{x}_r + i\mathbf{x}_i] = \mathbf{b}_r + i\mathbf{b}_i \tag{D.11}$$

or

$$\begin{aligned} \mathbf{A}_r\mathbf{x}_r - \mathbf{A}_i\mathbf{x}_i &= \mathbf{b}_r \\ \mathbf{A}_i\mathbf{x}_r + \mathbf{A}_r\mathbf{x}_i &= \mathbf{b}_i \end{aligned} \tag{D.12}$$

which may be put in a supermatrix form as

$$\begin{bmatrix} \mathbf{A}_r & -\mathbf{A}_i \\ \mathbf{A}_i & \mathbf{A}_r \end{bmatrix}\begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} \mathbf{b}_r \\ \mathbf{b}_i \end{bmatrix} \tag{D.13}$$

where

$$\mathbf{A}_r = \begin{bmatrix} 5 & 3 \\ 6 & 8 \end{bmatrix} \qquad \mathbf{A}_i = \begin{bmatrix} 1 & -1 \\ -2 & 4 \end{bmatrix}$$

$$\mathbf{b}_r = \begin{bmatrix} 6 \\ 5 \end{bmatrix} \qquad \mathbf{b}_i = \begin{bmatrix} 0 \\ -5 \end{bmatrix}$$

and $\mathbf{x}_r$ and $\mathbf{x}_i$ are column vectors constituting the real and imaginary parts of the solution vector. A set of complex simultaneous equations can then be converted into a set of real simultaneous equations.

Substituting the numerical values for $\mathbf{A}_r$, $\mathbf{A}_i$, $\mathbf{b}_r$, and $\mathbf{b}_i$ in Eq. (D.13), we obtain

$$\begin{bmatrix}\begin{vmatrix}5&3\\6&8\\1&-1\\-2&4\end{vmatrix} - \begin{vmatrix}1&-1\\-2&4\\5&3\\6&8\end{vmatrix}\end{bmatrix}\begin{bmatrix}x_r\\x_i\end{bmatrix}=\begin{bmatrix}\begin{vmatrix}6\\5\\0\\-5\end{vmatrix}\end{bmatrix} \qquad \text{(D.14)}$$

When

$$\mathbf{x}_r=\begin{bmatrix}x_1^r\\x_2^r\end{bmatrix} \quad \text{and} \quad \mathbf{x}_i=\begin{bmatrix}x_1^i\\x_2^i\end{bmatrix}$$

are used, Eq. (D.14) becomes

$$\begin{aligned}&\begin{bmatrix}5&3\\6&8\end{bmatrix}\begin{bmatrix}x_1^r\\x_2^r\end{bmatrix}-\begin{bmatrix}1&-1\\-2&4\end{bmatrix}\begin{bmatrix}x_1^i\\x_2^i\end{bmatrix}=\begin{bmatrix}6\\5\end{bmatrix}\\&\begin{bmatrix}1&-1\\-2&4\end{bmatrix}\begin{bmatrix}x_1^r\\x_2^r\end{bmatrix}+\begin{bmatrix}5&3\\6&8\end{bmatrix}\begin{bmatrix}x_1^i\\x_2^i\end{bmatrix}=\begin{bmatrix}0\\-5\end{bmatrix}\end{aligned} \qquad \text{(D.15)}$$

or

$$\begin{aligned}&\begin{bmatrix}5x_1^r+3x_2^r\\6x_1^r+8x_2^r\end{bmatrix}-\begin{bmatrix}x_1^i-x_2^i\\-2x_1^i+4x_2^i\end{bmatrix}=\begin{bmatrix}6\\5\end{bmatrix}\\&\begin{bmatrix}x_1^r-x_2^r\\-2x_1^r+4x_2^r\end{bmatrix}+\begin{bmatrix}5x_1^i+3x_2^i\\6x_1^i+8x_2^i\end{bmatrix}=\begin{bmatrix}0\\-5\end{bmatrix}\end{aligned} \qquad \text{(D.16)}$$

Solving the set of simultaneous equations above, we obtain

$$x_1^r=1.26 \quad x_2^r=-0.22$$

$$x_1^i=-0.32 \quad x_2^i=0.04$$

or

$$\mathbf{x}_1=x_1^r+ix_1^i=1.26-i0.32$$

$$\mathbf{x}_2=x_2^r+ix_2^i=-0.22+i0.04$$

D.4 COMPUTATION OF THE INVERSE MATRIX

In general, a *determinant* can be evaluated as

$$\det \mathbf{A} = \sum_{i=1}^{n} (-1)^{i+j} a_{ij} \det \mathbf{A}(i|j) \tag{D.17}$$

where a_{ij} is the (i,j)th entry of $\mathbf{A}$ and $\mathbf{A}(i|j)$ is an $(n-1) \times (n-1)$ matrix obtained by deleting the ith row and jth column of $\mathbf{A}$. The scalar $(-1)^{i+j} \det \mathbf{A}(i|j)$ is called the *cofactor* of the (i,j)th entry of $\mathbf{A}$ and is denoted simply by c_{ij}. Equation (D.17) then becomes

$$\det \mathbf{A} = \sum_{i=1}^{n} a_{ij} c_{ij} \tag{D.18}$$

In computing a specific determinant, it is frequently easier step by step to reduce the determinant to a size of lower order. This can be done by adding a multiple of one row of the determinant to another (or a multiple of one column to another).

Take (D.16) as an example. x_1' can be solved as

$$x_1' = \frac{\begin{vmatrix} 6 & 3 & -1 & 1 \\ 5 & 8 & 2 & -4 \\ 0 & -1 & 5 & 3 \\ -5 & 4 & 6 & 8 \end{vmatrix}}{|A|} \tag{D.19}$$

where

$$|A| = \begin{vmatrix} 5 & 3 & -1 & 1 \\ 6 & 8 & 2 & -4 \\ 1 & -1 & 5 & 3 \\ -2 & 4 & 6 & 8 \end{vmatrix} \tag{D.20}$$

or

$$|A| = \begin{vmatrix} 1 & -1 & 5 & 3 \\ 6 & 8 & 2 & -4 \\ 5 & 3 & -1 & 1 \\ -2 & 4 & 6 & 8 \end{vmatrix} \tag{D.21}$$

By subtracting multiples of row 1 from rows 2, 3, and 4, respectively, $|A|$ can be reduced to

$$|A| = -\begin{vmatrix} 1 & -1 & 5 & 3 \\ 0 & 14 & -28 & -22 \\ 0 & 8 & -26 & -14 \\ 0 & 2 & 16 & 14 \end{vmatrix} \tag{D.22}$$

or

$$|A| = -8\begin{vmatrix} 7 & -14 & -11 \\ 4 & -13 & -7 \\ 1 & 8 & 7 \end{vmatrix} \tag{D.23}$$

By following the same procedure, the determinant can be further reduced by

$$|A| = -400\begin{vmatrix} 0 & 7 & 6 \\ 0 & 9 & 7 \\ 1 & 8 & 7 \end{vmatrix} \tag{D.24}$$

or

$$|A| = -400\begin{vmatrix} 7 & 6 \\ 9 & 7 \end{vmatrix} = 2000$$

The *adjoint* of **A** is defined as an $n \times n$ matrix, which is the transpose of the matrix of cofactors of **A**. Thus

$$(\text{adj}\,\mathbf{A})_{ij} = c_{ji} = (-1)^{i+j}\det \mathbf{A}(j|i) \tag{D.25}$$

The *inverse matrix* for **A** is*

$$\mathbf{A}^{-1} = (\det \mathbf{A})^{-1}\text{adj}\,\mathbf{A} = \frac{1}{\det \mathbf{A}}\text{adj}\,\mathbf{A} \tag{D.26}$$

Example. Use the adjoint formula to compute the inverse of the following 3×3 matrix:

$$A = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

Solution. The determinant of the matrix **A** is

$$\det \mathbf{A} = \cos^2\theta + \sin^2\theta = 1$$

*For the denvation of this relation, see Hoffman and Kunze (1971).

The adjoint of **A** is

$$\operatorname{adj}\mathbf{A} = \begin{bmatrix} \begin{vmatrix} 1 & 0 \\ 0 & \cos\theta \end{vmatrix} & -\begin{vmatrix} 0 & -\sin\theta \\ 0 & \cos\theta \end{vmatrix} & \begin{vmatrix} 0 & -\sin\theta \\ 1 & 0 \end{vmatrix} \\ -\begin{vmatrix} 0 & 0 \\ \sin\theta & \cos\theta \end{vmatrix} & \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix} & -\begin{vmatrix} \cos\theta & -\sin\theta \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} 0 & 1 \\ \sin\theta & 0 \end{vmatrix} & -\begin{vmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{vmatrix} & \begin{vmatrix} \cos\theta & 0 \\ 0 & 1 \end{vmatrix} \end{bmatrix}$$

$$= \begin{vmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin & 0 & \cos\theta \end{vmatrix}$$

Therefore,

$$\mathbf{A}^{-1} = \frac{1}{\det\mathbf{A}}\operatorname{adj}\mathbf{A} = \begin{vmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{vmatrix}$$

This Page Intentionally Left Blank

Appendix E: Eigenvectors and Eigenvalues of an Operator

An eigenvalue and corresponding eigenvector of a matrix satisfy the property that the eigenvector multipled by the matrix yields a vector proportional to itself. In other words, an eigenvalue (or characteristic number) of an operator $\mathbf{A}$ is a number λ such that for some nonzero vector $\mathbf{x}$ the following equality holds:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \tag{E.1}$$

where $\mathbf{x}$, any nonzero vector that satisfies Eq. (E.1), is called an *eigenvector* of the operator A, and λ, the proportionality constant, is known as the *eigenvalue*. For Eq. (E.1) to be conformable, $\mathbf{A}$ must be square. Hence only square matrices have eigenvalues. The spectrum of an operator is the set of all its eigenvalues.

Let an operator $\mathbf{A}$ be represented by

$$\mathbf{A} = (a_{ik}) \tag{E.2}$$

Then

$$\mathbf{A}\mathbf{x} = \sum_{k=1}^{n} a_{1k}x_k, \ldots, \sum_{k=1}^{n} a_{nk}x_k \tag{E.3}$$

Equation (E.1) represents a system of equations as follows:

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix} = \lambda \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix} \tag{E.4}$$

or

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= \lambda x_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= \lambda x_2 \\ \vdots \qquad\qquad & \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= \lambda x_n \end{aligned} \tag{E.5}$$

After shifting terms, we obtain

$$\begin{aligned} (a_{11} - \lambda)x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= 0 \\ a_{21}x_1 + (a_{22} - \lambda)x_2 + \cdots + a_{2n}x_n &= 0 \\ \vdots \qquad\qquad & \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + (a_{nn} - \lambda)x_n &= 0 \end{aligned} \tag{E.6}$$

This system of equations will have a nonzero solution if and only if the following characteristic equation is satisfied:

$$\begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{vmatrix} = 0 \tag{E.7}$$

That is, this system of equations will have a nonzero solution if and only if λ is a root of the characteristic polynomial of the matrix.

Example. Let **A** be a (real) 3 × 3 matrix

$$\begin{bmatrix} 3 & 1 & -1 \\ 2 & 2 & -1 \\ 2 & 2 & 0 \end{bmatrix}$$

Then the characteristic polynomial for **A** is

$$\begin{vmatrix} \lambda - 3 & -1 & 1 \\ -2 & \lambda - 2 & 1 \\ -2 & -2 & \lambda \end{vmatrix} = \lambda^3 - 5\lambda^2 + 8\lambda - 4 = (\lambda - 1)(\lambda - 2)^2$$

Thus the characteristic values (eigenvalues) for **A** are 1 and 2.

By expanding Eq. (E.7) in terms of λ, the following characteristic equation results:

$$\lambda^n + C_{n-1}\lambda^{n-1} + \cdots + C_1\lambda + C_0 = 0 \tag{E.8}$$

which, in turn, can be factorized into the form

$$(\lambda - \lambda_1)(-\lambda_2)\cdots(\lambda - \lambda_n) = 0 \tag{E.9}$$

showing that a matrix of order n has n eigenvalues.

Comparing Eq. (E.8) with (E.9) and remembering that the *trace* of a matrix is defined as the sum of the diagonal elements of the matrix, we obtain

$$\begin{aligned} \mathrm{tr}(\mathbf{A}) &= a_{11} + a_{22} + \cdots + a_{nn} \\ &= -C_{n-1} = \lambda_1 + \lambda_2 + \cdots + \lambda_n \end{aligned} \tag{E.10}$$

that is, the sum of the eigenvalues of a matrix is equal to the trace of the matrix. Also, from Eqs. (E.7) to (E.9),

$$|\mathbf{A}| = (-1)^n C_0 = \lambda_1 \lambda_2 \cdots \lambda_n \tag{E.11}$$

that is, the product of the eigenvalues of a matrix equals the determinant of the matrix.

This Page Intentionally Left Blank

Appendix F: Notation

CHAPTER 1

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Pattern or pattern vector

n Number of dimensions of the pattern vector

r Number of dimensions of the feature vector

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

Pattern space

$\mathbf{z}_k^m = \begin{bmatrix} z_{k1}^m \\ z_{k2}^m \\ \vdots \\ z_{ki}^m \\ \vdots \\ z_{kn}^m \end{bmatrix}$	*m*th prototype in class *k*
$k = 1, \ldots, M$	Pattern class index
$m = 1, \ldots, N_k$	Prototype index
M	Number of pattern classes
ω_k	Pattern class *k*
N_k	Number of prototypes in the *k*th class
$d(\cdot)$	A distance function
a, b, c, d	Primitives (or terminals) used for structural analysis

CHAPTER 2

$\mathbf{z}$	Prototype vector
$\mathbf{d}$	Desired output vector
$\mathrm{f}(\mathbf{z}, \mathbf{w})$	Actual output response
$(\mathbf{z}, \mathbf{d})$	Training pair
$r_k(\mathbf{z}) = \sum_{i=1}^{N} L_{ik} p(\mathbf{d}_i \vert \mathbf{z})$	Risk function
$p(\mathbf{d}_i \vert \mathbf{z})$	Probability that **z** is the same as $\mathbf{d}_i$ from the (**z**, **d**) pairs

CHAPTER 3

$d(\mathbf{x})$	Decision or discriminant function
$d_k(\mathbf{x})$	Values of the discriminant function for patterns **x** in class *k*
$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w_{n+1} \end{bmatrix}$	Augmented weight vector

$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix}$	Augmented pattern vector
$d_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x}$	Linear decision function
$D(\mathbf{x}, \mathbf{z}_k)$	A metric, euclidean distance of an unknown pattern $\mathbf{x}$ from $\mathbf{z}_k$
$\mathbf{z}_k$	Prototype average (or class center) for class ω_k
N_k	The number of prototypes used to represent the category k
$d_k^m(\mathbf{x})$	Value of discriminant function for the m prototype in class k
$\forall$	For all
$\in$	Belongs to
$\notin$	Does not belong to
max	Maximum
$\sum$	Summation
$\exists$	There exists
$\nexists$	There does not exist
$\mathbf{w}_i^T = (w_{i1}, w_{i2}, \ldots, w_{in}),$ $i = 1, \ldots, R$	n-dimensional weight vectors for the various discriminant functions in layered machine
$\mathbf{A} = \begin{vmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & & & \vdots \\ w_{n1} & & \cdots & w_{nn} \end{vmatrix}$	Weight matrix used in quadratic discriminant function
$\mathbf{B} = \begin{vmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{vmatrix}$	Weight vector used in quadratic discriminant function
$C = w_{n+1}$	Weight constant used in quadratic discriminant function
$\phi(\mathbf{x})$	ϕ function (or generalized decision function) used for pattern classification

$f_i(\mathbf{x}), i = 1, \ldots$	Linearly independent, real and single-valued functions
$D(N, n)$	Linear dichotomies on N patterns in n-dimensional space
$P_{N,m}$	$\dfrac{\text{number of } \phi \text{ dichotomies}}{\text{total possible number of dichotomies}}$
λ	Ratio of N to $M + 1$
$\psi(\mathbf{x}, \mathbf{z}_k^M)$	A potential function of $\mathbf{x}$ and $\mathbf{z}_k^m$ defined over the pattern space

CHAPTER 4

T	Margin (or threshold)
$\mathbf{w}(k), \mathbf{w}(k+1)$	Weight vectors at the kth and $(k+1)$th correction steps
$\mathbf{Z} = [\mathbf{z}_1^1, \mathbf{z}_2^1, \ldots, \mathbf{z}_1^{N_1}, \mathbf{z}_2^{N_2}]$	Training sequence
∇	Gradient operator
$J(\mathbf{w})$	A criterion function used in weight adjustment procedure
ρ_k	A positive scale factor, used to set the training step size
$J_p(\mathbf{w})$	Perceptron criterion function
P	Set of samples misclassifed by $\mathbf{w}(k)$
$J_r(\mathbf{w})$	Relaxation criterion function
$\operatorname{sgn} d_i(\mathbf{z})$	Equal to $+1$ or -1 according to $d_i(\mathbf{z}) \geq$ or < 0
N	Total number of prototypes for all classes
N_i	Total number of prototypes for class ω_i
M	Total number of classes
$\mathbf{e}$	Error vector
$J_s(\mathbf{w})$	Sum-of-squared-error criterion function
$z^{\#}$	Pseudoinverse or generalized inverse of $\mathbf{z}$
$\delta b_i(k)$	Adjustment for $b(k)$

CHAPTER 5

$p(\omega_i)$	A priori probability of class ω_i
$p(\mathbf{x})$	Probability density function of $\mathbf{x}$, or probability that $\mathbf{x}$ occurs without regard to category to which it belongs
$p(\mathbf{x}\|\omega_i)$	Likelihood function of class ω_i
$p(\omega_i\|\mathbf{x})$	Probability that $\mathbf{x}$ comes from ω_i
L_{ij}	Loss or penalty due to deciding $\mathbf{x} \in \omega_j$ when in fact $\mathbf{x} \in \omega_i$
$r_k(\mathbf{x})$	Conditional average loss (or conditional average risk of misclassifying $\mathbf{x}$ as in ω_k)
$\delta(k-1)$	Kronecker delta function
N	Normal density function
$\mathbf{m}_k$	Mean vector for class k
$\mathbf{C}_k$	Covariance matrix for class k
$\phi_k(\mathbf{x})$	Arbitrary functions used for determining the statistical discriminant function
c_{ik}	Coefficients for use with $\phi_k(\mathbf{x})$ in determining the statistical discriminant function

CHAPTER 6

$\cup$	Union
$\cap$	Intersection
$\zeta(\mathbf{x}_i, \mathbf{x}_j)$	Dissimilarity measure between two patterns
α_k	Weighting coefficient
σ_{km}^2	Variance of the *m*th cluster in the *k*th direction
$r(\mathbf{x}_i, \mathbf{x}_j)$	Mahalanobis distance from $\mathbf{x}_i$ to $\mathbf{x}_j$
C^{-1}	Inverse of the covariance matrix
$d_t(x_i, x_i)$	Tanimoto coefficient
$D_{\alpha\beta}$	Interset distance between two separate sets $[\mathbf{x}_\alpha^i]$ and $[\mathbf{x}_\beta^j]$, $i = 1, 2, \ldots, N_\alpha$; $j = 1, 2, \ldots, N_\beta$

$D_{\alpha\alpha}$	Intraset distance in the set $[\mathbf{x}^i_\alpha]$, $i = 1, 2, \ldots, N_\alpha$
τ	Membership boundary for a specified cluster
θ	Fractional constant for setting the indeterminate region
$\mathbf{z}_i$	ith cluster
$\mathbf{z}_i(t+1)$	i cluster computed with $(t+1)$ pattern samples assigned to it
$\mathbf{C}(t+1)$	Variance computed with $(t+1)$ pattern samples
Maximin	Maximum of the minimum distances
$\Omega_k(i)$	A set of k-nearest neighbors of the sample points $i, i = 1, 2, \ldots, N$, basing on euclidean distance measure
$P_k(i)$	Potential of the pattern sample point i
$d(i, j)$	Euclidean distance between the sample points i and j
$\xi_k(i)$	A set of sample points k-adjacent to the sample point i
$\mathrm{SIM}(m, n)$	Similarity measure used for merging two most similar subclusters
$\mathrm{SIM}_1(m, n)$	Represents the difference in density between the cluster and the boundary
$\mathrm{SIM}_2(m, n)$	Represents the relative size of the boundary to that of the cluster
$Y_k^{m,n}$	Denotes the set of points which are in the cluster m while, at the same time, their respective k adjacent points are in cluster n
W_m	A set of points contained in sub-cluster m
$\mathrm{BP}_k^{m,n}$	Average of $P_k(i)$ over all i in $\mathrm{Y}_k^{m,n}$
$P_k^{sc}(m)$	Average of $P_k(i)$ over all points in cluster m
$N(\cdot)$	The number of elements of the set inside the bracket
φ	Denotes a set of unordered pairs of subclusters
$S_j(k)$	Cluster domain j at the kth iteration

N_j	The number of samples in $S_j(k)$
M	Number of clusters desired
η	Minimum number of samples desired in a cluster
σ_s	Maximum standard deviation allowed
δ	Minimum distance required between clusters
L	Maximum number of pairs of cluster center which can be lumped
I	Number of iterations allowed
$\mathbf{z}_{il}, \mathbf{z}_{jl}$	Cluster centers to be lumped
N_{il}, N_{jl}	Number of samples in clusters $\mathbf{z}_{il}$ and $\mathbf{z}_{jl}$
R_{ij}	Clustering parameter suggested by Davies and Bouldin to obtain natural partitionings of the data sets
R_i	Maximum of R_{ij}
$\bar{R}$	The average of the similarity measures of each cluster with its most similar cluster
$\lambda_c(N_c)$	A performance index used by DYNOC to determine the optimal clusters
E_i	"Sampling" or multiple centers or cores
S_i	Cluster domains
$D(E_i, S_i)$	Degree of similarity of E_i to S_i
Q	A criterion function used by the dynamic clusters method
s_{ij}	Elements (i, j) of the similarity matrix S used in the graph theoretic method
θ	Threshold distance denoting the similarity between two pattern points
S', S'', S'''	Reduced similarity matrices
$\mathsf{G} = [G_1, G_2, \ldots, G_M]$	Graphs
$\mathsf{R} = [R_1, R_2, \ldots, R_M]$	Regions of influence
GG	Gabriel graph
RNG	Relative neighborhood graph

(p_i, p_j)	Line segment
β	A factor of relative edge consistency
Λ	Diagonal matrix

CHAPTER 7

r_p^2	Mahalanobis distance produced by the p features
d_i^2	Contribution of the ith feature to the Mahalanobis distance
ξ	Positive arbitrary constant (less than 1)
w_{jj}	Feature weighting coeficients
$\overline{D}^2$	Intraset distance for pattern points after transformation
σ_j^2	Sample variance of the components along the x_j coordinate direction
ρ_1	Lagrange multiplier
$\prod_{j=1}^{n}$	Continual product
x_{ijk}	The observation (or the intensity of picture element) in the kth channel for picture element j in scan line i
$\hat{m}_l$	The sample mean vector for the lth category
$\hat{\mathbf{C}}_l$	The covariance matrix for the lth category
$\mathbf{x}$	The observation vector of dimension n
$\mathbf{y}$	The transformed vector of dimension p
$\mathbf{C}$	$p \times n$ transformation matrix
$\mathbf{A}$	The covariance of the means of the categories
$\mathbf{X}$	$n \times M$ matrix of all the category means composed of all mean vectors $\mathbf{m}_k$
M	Number of categories
$\mathbf{n}$	$M \times 1$ vector of the number of observations in the categories
$\mathbf{N}$	$M \times M$ matrix of the number of observations in the categories

n_l	The number of observations for category l
$\mathbf{W}$	Combined covariance matrix for all the categories
I	$p \times p$ identity matrix
Λ	Diagonal matrix
C_{y1}, C_{y2}	Within-class of the projection data
Y_1, Y_2	Projections of the pattern points from ω_1 and ω_2
$J(\mathbf{w})$	Fisher's criterion
S_B	Between-class covariance matrix
S_W	Total within-class covariance matrix
BEG	Basic event generation
$\mathbf{E}_{ik}$	Basic events for class ω_i by means of BEG
F_{ik}	A subset of features for each basic event $\mathbf{E}_{ik}$
$M(.,.)$	Merging function
$\mathbf{R}^n$	Pattern space
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_i}$	Training samples from class ω_i
$\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{N_j}$	Training samples from classes other than ω_i
$\mathrm{dist}(\mathbf{y}_l \mid \mathbf{E}_{ik})$	The distance between $\mathbf{y}_l$ and $\mathbf{E}_{ik}$
T_A	Threshold, a certain positive number
F_{ik}^l	The sets of effective features for l training samples not in class ω_i

CHAPTER 8

$\mathbf{z}_i = \begin{vmatrix} z_{i1} \\ z_{i2} \\ \vdots \\ z_{in} \end{vmatrix}$	Prototype vector $\mathbf{z}_i$
$\mathbf{x} = \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix}$	Unknown pattern vector, input to the multilayer perceptron

$\mathbf{y} = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{vmatrix}$	Output of the multilayer perceptron
$\mathbf{w} = \begin{vmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{vmatrix}$	Weight vector
$\mathbf{w}(k), \mathbf{w}(k+1)$	Weight vectors at kth and $(k+1)$th correction steps
$d(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$	Discriminant function
ω_k	Pattern class k
$x_i, i = 1, 2, \ldots, N$	Neurons in the input layer
$x'_j, j = 1, 2, \ldots, N_1$	Neurons in the first hidden layer
$x''_k, k = 1, 2, \ldots, N_2$	Neurons in the second hidden layer
$y_l, l = 1, 2, \ldots, M$	Neurons in the output layer
$w_{ji}, i = 1, 2, \ldots, N$	Weights connecting the neuron x'_j of the first hidden layer to neurons x_i of the input layer
$w_{kj}, j = 1, 2, \ldots, N_1$	Weights connecting the neuron x''_k of the second hidden layer to neurons x'_j of the first hidden layer
$w_{lk}, k = 1, 2, \ldots, N_2$	Weights connecting the neuron y_l of the output layer to neurons x_k of the second hidden layer
$\theta'_j, j = 1, 2, \ldots, N_1$	Bias used in the computation of the output of neuron x'_j of the first hidden layer
$\theta''_k, k = 1, 2, \ldots, N_2$	Bias used in the computation of the output of neuron x''_k of the second hidden layer
$\theta'''_l, l = 1, 2, \ldots, M$	Bias used in the computation of the output of neurons y_l of the output layer
ΔW_{ji}	Increment of the weight adjustment
$\partial E / \partial W_{ji}$	Gradient of the mapping error E with respect to W_{ji}

W_{ji}	Synapse (or weight) connecting the processing element j of this layer (say Lth layer) to the processing element i of the nearest previous layer [say $(L-1)$st layer]
H	Training set
$\{\mathbf{i}^k, \mathbf{t}^k\}$	kth pair of input/output vectors for training
$\mathbf{i}^k$	kth pattern vector presented to the system
$\mathbf{O}^k$	kth actual (or computed) output vector
$\mathbf{t}^k$	kth target vector
$\mathbf{e}^k$	Output error vector when the kth input vector $\mathbf{i}^k$ is presented
$\mathbf{J}$	Performance criterion function
O_j^k	Output of the jth neuron unit in the output layer
O_i^k	Input to the particular jth neuron in the output layer from the ith neuron of the previous hidden layer
θ_{j}	Bias
f(.)	Activation function
$\sigma_{\mathrm{j}}^{\mathrm{k}}$	Error signal, which is defined as $-\partial E^k/\partial\,\mathrm{net}_j$
$\mathrm{net}_{\mathrm{j}}$	is $\sum_{l}, W_{jl}O_l^k$, where O_l^k is the output of a neuron in a previous layer.
$\partial E^k/\partial\,\mathrm{net}_j$	Sensitivity of the pattern error on the activation of the jth unit
η	Learning rate

CHAPTER 9

$\mathbf{x}$	d-dimensional input vector
$\mathbf{t}$	Target vector
$y_k(\mathbf{x})$	kth, component of $\mathbf{y}(\mathbf{x})$, the function to map the input space to the one-dimensional target space
N	The number of the input vectors $\mathbf{x}^n, n = 1, 2, \ldots, N$

M	The number of basis functions in the hidden unit
$\phi_i(\lvert\mathbf{x} - \hat{\mathbf{c}}_i\rvert)$	A set of basis functions with the Euclidean distance of the input vector **x** from a center $\mathbf{c}_i$
$\mathbf{c}_j$	The vector used to determine the center of basis function ϕ_j and has elements c_{ji}
$\phi_i(\mathbf{x}) = \exp\left(-\frac{1}{2\sigma_i^2}\lvert\mathbf{x} - \mathbf{c}_i\rvert^2\right)$	Another form of the basis functions with the euclidean distance of the input vector **x** from a center $\mathbf{c}_i$
$\phi_i(\mathbf{x}) = \sigma^2/[\sigma^2 + \lvert\mathbf{x} - \mathbf{c}_i\rvert^2]$	Another form of the basis functions with the euclidean distance of the input vector **x** from a center $\mathbf{c}_i$
$\phi(\mathbf{x}) = (x^2 + \sigma^2)^{-\alpha}, \alpha > 0$	Another form of the basis functions
$\phi(\mathbf{x}) = x^2 \ln \mathbf{x}$	Another form of the basis functions
$\mathfrak{I}$	Cost function for use in the error minimization method
$\phi = \begin{vmatrix} \phi_1 \\ \phi_2 \end{vmatrix}$	The corresponding ϕ resulting from the mapping of the radial basis function, where $\phi_1 = \exp(-\lvert\mathbf{x} - \mathbf{c}_1\rvert^2)$, and $\phi_2 = \exp(-\lvert\mathbf{x} - \mathbf{c}_2\rvert^2)$
$\phi^1, \phi^2, \ldots, \phi^9$	Represent respectively the pattern points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_9$ in the transform space
$y(j)$	Computed output of the network
$d(j)$	Desired output of the network
$e(j)$	Error
N	Number of input/desired output training pairs
$W_1(k + 1)$	Weight (synapse) at $(k + 1)$st correction step
$\mathbf{c}_i(k + 1)$	$\mathbf{c}_i$ at $(k + 1)$st correction step
$\sigma_i(k + 1)$	Value of σ_i at $(k + 1)$st correction step
$p(\omega_i \vert \mathbf{x})$	A posteriori probabilities
$p(\mathbf{x} \vert \omega_i)$	Likelihood function of class ω_i with respect to **x**
$p(\omega_i)$	A priori probability of class ω_i
$p(\mathbf{x})$	The probability that **x** occurs without regard to the category in which it belongs

CHAPTER 10

$\mathbf{x}$	Binary bipolar input vector
$\mathbf{c}_i$	Binary bipolar encoded class prototype vector
$\mathrm{HD}(\mathbf{x}^T, \mathbf{c}_i)$	Hamming distance between the input vector $\mathbf{x}$ and the encoded class prototype vector $\mathbf{c}_i$
$\mathbf{a}$	The number of components which agree in the two binary bipolar n-tuple vectors
$\mathbf{d}$	The number of components which differ in the two binary bipolar n-tuple vectors
n	Total number of components in the binary bipolar vector
$\mathrm{net}_j = \mathbf{x}^T\mathbf{w}_j + b_j, j = 1, 2, \dots, M$	Output of neurons in the Hamming net
$\mathbf{W}$	Weight matrix of the Hamming net
$\mathbf{y} = [y_1, y_2, \dots, y_M]^T$	Input to the Maxnet
$\mathbf{O} = [O_1, O_2, \dots, O_M]^T$	Output of the Maxnet
$\mathbf{W}_M$	Weight matrix for the Maxnet
$f(.)$	Activation function
net^k	Results obtained after kth recurrent processing of the Maxnet
ϑ_j	External stimulus signal on the neuron j
γ_{jk}	The lateral feedback weights connected to neuron j
$y_j, j = 1, 2, \dots, m$	Output responses of the network
$\gamma_{jk}y_{j+k}, k = -K, -K+1, \dots, K$	Lateral interaction of neurons on neuron j in the output layer
$\mathbf{x}_k = (x_{k1}, x_{k2}, \dots, x_{kn})^T$, $k = 1, 2, \dots, N$	Pattern vectors k
$\mathbf{w}_j = (w_{j1}, w_{j2}, \dots, w_{jn})^T$, $j = 1, 2, \dots, m$	Synaptic weight vector of neuron j in the output layer
$\mathbf{w}_j(t)$	Synaptic weight vector $\mathbf{w}_j$ of neuron j at discrete time t
$\mathbf{w}_j(t+1)$	Updated weight vector $\mathbf{w}_j$ of neuron j at time $t+1$

$\beta(t)$	Learning rate control parameter
Λ_j	Spatial neighborhood
$d(j)$	Euclidean distance between the inputs and the output neuron j

CHAPTER 11

$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$	n-dimensional input vector
$\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$	Output
$u_1, u_2, \ldots, u_n$	Nodes representing the intermediate status of the output during iterations
W_{ji}	Weight from output of node i to input of node j, $i, j = 1, 2, \ldots, n$; and $i \neq j$. It specifies the contribution of the output signal y_i of the neuron i to the net potential acting on neuron j
$\text{net}_j, i = 1 \sim n \text{ and } j \neq i$	Net potential, which is $\sum_i W_{ji} y_i + x_j - \theta_j$
x_j	External input to neuron j
y_i	Output of the neuron $i, i = 1, 2, \ldots, n; j \neq i$
θ_j	Threshold applied externally to neuron j
$\text{sgn}\{\cdot\}$	Signum function
$\mathbf{S}_\mu, \mu = 1, 2, \ldots, l$	A set of l bipolar pattern vectors
$\mathbf{W}$	Weight matrix, which is $(1/n)\sum \mathbf{s}_\mu \mathbf{s}_\mu^T - (k/n)\mathbf{I}, \mu = 1, 2, \ldots, l$
$\mathbf{S}_\mu \mathbf{S}_\mu$	Outer product of the vector $\mathbf{S}_\mu$ with itself
$\mathbf{I}$	Identify matrix

CHAPTER 12

$p_r(r)$	Probability density function (or the relative occurrence frequency)
$\int_{r_1}^{r_i} p_r(r)\, dr$	Cumulative density function
r	Input image intensity or gray levels normalized within the range (0, 1)
n_i	Number of pixels at gray level r_i

s	Output image intensity
$T(r)$	Intensity transformation function
$H_r(j)$	Historgram of the input image
$H_s(k)$	Historgram of the image after gray-level transformation
$f'(x, y)$	Hypothetically noise-free input image
$n(x, y)$	The noise
$f(x, y)$	The noisy image
$\hat{f}(X, y)$	Expected processed image
$H(x, v)$	Process transfer function
$G(u, v)$	Processed image in transform domain
ϕ	Cost function
$g(x, y)$	The smoothed image
$(G-f)^2$	Squared difference between $g(x, y)$ and $f(x, y)$

CHAPTER 13

$B_{i,1}$	Linear B spline
$B_{i,2}$	Quadratic B spline
$B_{i,m}$	mth-degree B spline
ϕ_l	Angular change for use with Fourier descriptor for shape discrimination
$[a_i]_1^n$	A chain of codes for graph representation
L_i^s	A straight line segment
a_{ix}	x component of a chain link
a_{iy}	y component of the chain link
X_i^s	$\sum_{j=i-s+1}^{i} a_{jx}$
Y_i^s	$\sum_{j=i-s+1}^{i} a_{jy}$
Θ_i^s	$\tan^{-1}\left(\frac{Y_i^s}{X_i^s}\right)$
δ_i^s	$\Theta_{i+1}^s - \Theta_{i-1}^s$

γ_{ij}	A criterion used for template matching

CHAPTER 14

$f(x)$	One-dimensional function
$F(u)$	Fourier transform of $f(x)$
$f(x, y)$	Two-dimensional image model
$F(u, v)$	Fourier transform of $f(x, y)$
$\phi(u, v)$	Phase spectrum of $f(x, y)$
$E(u, v)$	Energy spectrum of $f(x, y)$
$g(x, y; u, v)$	Forward transform kernel
$h(x, y; u, v)$	Inverse transform kernel
δ	Dirac delta function
$g_c(x, u)$ or $g_{\text{col}}(x, u)$	Forward column transform kernel
$g_r(y, v)$ or $g_{\text{row}}(y, v)$	Forward row transform kernel
$F_x(x, v) = \int_{-\infty}^{\infty} f(x, y)e^{-j2\pi vy}\, dy$	Columnwise transform $f(x, y)$
$F_y(u, y) = \int_{-\infty}^{\infty} f(x, y)e^{-j2\pi ux}\, dx$	Rowwise transform $f(x, y)$
$S(u)$	A sampling impulse train
$W_N = e^{-j2\pi/N}$	A notation used for simplifying the expressions of the Fourier transform pair used as the kernel for the sequence of N terms
$W_{N/2} = e^{-j2\pi/(N/2)}$	Used as the kernel for the sequence of $N/2$ terms
$G(u) = \sum_{r=0}^{(N/2)-1} f(2r)(W_{N/2})^{ru}$	$(N/2)$-point discrete Fourier transform
$H(u) = \sum_{r=0}^{(N/2)-1} f(2r+1)(W_{N/2})^{ru}$	$(N/2)$-point discrete Fourier transform
$G_1(u) = \sum_{l=0}^{(N/4)-1} g(2l)(W_{N/4})^{lu}$	That part of $G(u)$ for even values of r
$G_2(u) = \sum_{l=0}^{(N/4)-1} g(2l+1)(W_{N/4})^{lu}$	That part of $G(u)$ for odd values of r

$H_1(u) = \sum_{l=0}^{(N/4)-1} h(2l)(W_{N/4})^{lu}$	That part of $H(u)$ for even values of r
$H_2(u) = \sum_{l=0}^{(N/4)-1} h(2l+1)(W_{N/4})^{lu}$	That part of $H(u)$ for odd values of r
$g(x, u) = \frac{1}{N}\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)}$	Forward transform kernel for Walsh transform
$b_k(z)$	kth bit in the binary representation of z with the zeroth bit as the least significant one
$g(x, u) = \frac{1}{N}(-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)}$	Forward transform kernel for Hadamard transform
H_N	Hadamard matrix of order N
$H(u, v)$	$N \times N$ symmetric Hadamard transformation matrix
$F_H(u, v)$	Hadamard transform of $f(x, y)$
$\mathbf{x}$	Original image vector
$\mathbf{y}$	Transformed vector
$(\mathbf{x} - \mathbf{m}_x)$	Centralized image vector
$\mathbf{B}$	$N^2 \times N^2$ transformation matrix
λ_i	Variance of the ith element of $\mathbf{y}$ along eigenvector $\mathbf{e}_i$

CHAPTER 15

FT	Fourier transform
STFT	Short time Fourier transform
CWT	Continuous wavelet transform
DWT	Discrete wavelet transform
IDWT	Inverse discrete wavelet transform
$\psi_{jk}(t)$	Basis functions in wavelet transform
$W_\psi f(t)$	One-dimensional continuous wavelet transform of $f(t)$
$\langle f, \psi_{j,k} \rangle$	Inner products of the function $f(t)$ with each of the basis functions in one-dimensional wavelet transform

$\mathbf{c}_{jk}$	Set of expansion coefficients obtained from discrete wavelet transform
$f(t) = \sum_j \sum_k c_{jk}\psi_{jk}(t)$	One-dimensional inverse wavelet transform
$\psi_{j,kx,ky}(x, y)$	Basis functions in two-dimensional wavelet transform
$W_\psi f(x, y)$	Two-dimensional continuous wavelet transform of $f(x, y)$
$\langle f(x, y), \psi_{j,kx,ky}\rangle$	Inner product of $f(x, y)$ with each of the basis functions $\psi_{j,kx,ky}$ in two-dimensional wavelet transform
$f(x, y) = \sum_j \sum_{kx} \sum_{ky} c_{j,kx,ky}\psi_{j,kx,ky}(x, y)$	Two-dimensional inverse wavelet transform
$\varphi(t)$	Basic scaling function
L^2	The space of all functions $f(t)$ with a well-defined integral of the squares of modulus of the function
$v_0 = \mathrm{span}_k[\varphi_k(t)]$	Space spanned by $\varphi_k(t)$ or $\varphi(t - k)$
$v_j = \mathrm{span}_k[\varphi_k(2^j t)]$	Space spanned by $\varphi_{jk}(t)$ or $2^{j/2}\varphi(2^j t - k)$
$\psi_{jk}(t)$	Wavelets
$\oplus$	Superposition
$\perp$	Orthogonality
$*$	Convolution
$\downarrow 2$	Downsampling by 2 (decimation)
$\uparrow 2$	Upsampling (interpolation)
$x(n)$ and $y(n)$	Input and output of the downsampler
A	Approximation, referring to the high-scale-factor, low-frequency components of the signal
D	Details, referring the low-scale factor, high-frequency components of the signal
$x'(n)$ and $y(n)$	Input and output of the upsampler (interpolator)
$\varphi_{kx,kx} = \varphi(2^j x - k_x)\varphi(2^j y - k_y)$	Two-dimensional scaling function, which is an orthogonal basis function at scale 2^j for the image function $f(x, y)$

$\psi^1_{kx,ky}(x, y) = \varphi(2^j x - k_x)\psi(2^j y - k_y)$	One of the three two-dimensional wavelets for the decomposition of the image function $f(x, y)$
$\psi^2_{kx,ky}(x, y) = \psi(2^j x - k_x)\varphi(2^j y - k_y)$	One of the three two-dimensional wavelets for the decomposition of the image function $f(x, y)$
$\psi^3_{kx,ky}(x, y) = \psi(2^j x - k_x)\psi(2^j y - k_y)$	One of the three two-dimensional wavelets for the decomposition of the image function $f(x, y)$
$\mathrm{c}^{\mathrm{LL}}_{-1,kx,ky}$	The first subimage giving the approximation coefficients at a coarse resolution of 2^{-1}
$\mathrm{d}^{\mathrm{HL}}_{-1,kx,ky}$	"Details" coefficients at a coarse resolution of 2^{-1}
$\mathrm{d}^{\mathrm{LH}}_{-1,kx,ky}$	"Details" coefficients at coarse resolution of 2^{-1}
$\mathrm{d}^{\mathrm{HH}}_{-1,kx,ky}$	"Details" coefficients at coarse resolution of 2^{-1}

CHAPTER 16

$\nabla^2 f(x, y)$	Laplacian operator
$f_{xx}(n_1, n_2)$	Second derivative with respect to x at the pixel (n_1, n_2)
$f_{yy}(n_1, n_2)$	Second derivative with respect to y at the pixel (n_1, n_2)
$\sigma^2_f(n_1, n_2)$	Local variance at the pixel (n_1, n_2)
$S(x)$	Transmitted intensity
μ_{eff}	Effective attenuation coefficient
m^y_r	The mean of the projected road samples
m^y_{nr}	The mean of the projected noroad samples
v^2_r	The variance in the projected road samples
v^2_{nr}	The variance in the projected noroad samples
$P(x\|\mathrm{Road})$	Conditional probability that pixel x is from the road class
$P(x\|\mathrm{Noroad})$	Conditional probability that pixel x is from noroad class

P(Road)	A priori probability of a pixel being road
P(Noroad)	A priori probability of a pixel being Noroad
β	Ratio of P(Noroad) to P(Road)

Bibliography

BOOKS

Aggarwal, J. K., Duda, R. O., and Rosenfeld, A., eds. (1977). *Computer Methods in Image Analysis*, IEEE Press, New York.

Agrawala, A. K., ed. (1977). *Machine Recognition of Patterns*, IEEE Press, New York.

Ahmed, N., and Rao, K. R. (1975). *Orthogonal Transforms for Digital Signal Processing*, Springer-Verlag, Berlin.

Aho, A. V., and Ullman, J. D. (1972). *The Theory of Parsing, Translation and Compiling*, vol. 1, Prentice-Hall, Englewood Cliffs, NJ.

Akansu, A. N., and Hadda, R. A. (1993). *Multiresolution Signal Decomposition*, Academic Press, New York.

Anderberg, M. R. (1973). *Cluster Analysis for Applications*, Academic Press, New York.

Andrews, D. F. (1973). Graphical techniques for high-dimensional data, in *Discriminant Analysis and Applications* (T. Cacoullos, ed.), Academic Press, New York, pp. 37–59.

Andrews, H. C. (1970). *Computer Techniques in Image Processing*, Academic Press, New York.

Andrews, H. C. (1972). *Introduction to Mathematical Techniques in Pattern Recognition*, Wiley, New York.

Andrews, H. C., and Hunt, B. R. (1977). *Digital Image Restoration*, Prentice-Hall, Englewood Cliffs, NJ.

Arbib, M. A. (1972). *The Metaphorical Brain: An Introduction to Cybernetics as Artificial Intelligence and Brain Theory*, Wiley, New York.

Awcock, G. W., and Thomas, R. (1995). *Applied Image Processing*, McGraw Hill, New York.

Batchelor, B. G. (1974). *Practical Approach to Pattern Classification*, Plenum Press, New York.

Batchelor, B. G., ed. (1978). *Pattern Recognition—Ideas in Practice*, Plenum Press, New York.

Bhaskaran, V., and Konstantinides, K. (1995). *Image and Video Compression Standards*, Kluwers Academic Publishers.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*, Oxford University Press.

Bongard, M. (1970). *Pattern Recognition* (J. K. Hawkins, ed.; T. Cheron, trans.), Spartan Books, Washington, DC.

Bose, N. K., and Liang, P. (1996). *Neural Network Fundamentals with Graphics, Algorithms and Applications*, McGraw Hill, New York.

Bow, S. T. (1984). *Pattern Recognition—Applications to Large Data-Set Problems*, Marcel Dekker, New York.

Bow, S. T. (1992). *Pattern Recognition and Image Preprocessing*, Marcel Dekker, New York.

Brigham, E. (1974). *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, NJ.

Burrus, C. S., Gopinath, R., and Guo, H. (1998). *Introduction to Wavelets and Wavelet Transform*, Prentice-Hall, Englewood Cliffs, NJ.

Cacoullos, T., ed. (1973). *Discriminant Analysis and Applications*, Academic Press, New York.

Cappellini, V., and Marconi, R., eds. (1986). *Advances in Image Processing and Pattern Recognition*, Elsevier B. V., North Holland.

Carlson, P. F. (1977). *Introduction to Applied Optics for Engineers*, Academic Press, New York.

Castleman, K. R. (1979). *Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Chang, S. K. (1989). *Principles of Pictorial Information System Design*, Prentice-Hall, Englewood Cliffs, NJ.

Chasen, S. H. (1978). *Geometric Principles and Procedures for Computer Graphic Applications*, Prentice-Hall, Englewood Cliffs, NJ.

Chen, C. H. (1973). *Statistical Pattern Recognition*, Spartan Books, Washington, DC.

Cornsweet, T. N. (1970). *Visual Perceptron*, Academic Press, New York.

Dainty, J. C., and Shaw, R. (1974). *Image Science*, Academic Press, New York.

Dodwell, P. C. (1970). *Visual Pattern Recognition*, Holt, Rinehart and Winston, New York.

Dodwell, P. C. (1971). *Perceptual Processing*, Appleton-Century-Crofts (Meredith), New York.

Dougherty, E. R., and Giardina, C. R. (1987). *Image Processing—Continuous to Discrete*, vol. 1., Prentice-Hall, Englewood Cliffs, NJ.

Dougherty, E. R., and Giardina, C. R. (1988). *Morphological Methods in Image and Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Duda, R. O., and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*, Wiley, New York.

Everitt, B. (1974). *Cluster Analysis*, Wiley (Halsted Press), New York.

Fausett, L. (1994). *Fundamentals of Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ.

Fu, K. S. (1968). *Sequential Methods in Pattern Recognition and Machine Learning*, Academic Press, New York.

Fu, K. S. (1971). *Pattern Recognition and Machine Learning*, Plenum Press, New York.

Fu, K. S. (1974). *Syntactic Methods in Pattern Recognition*, Academic Press, New York.

Fu, K. S., ed. (1976). *Digital Pattern Recognition*, Springer-Verlag, Berlin.

Fu, K. S., ed. (1977). *Syntactic Pattern Recognition Applications*, Springer-Verlag, Berlin.

Fu, K. S. (1982). *Syntactic Pattern Recognition and Applications*, Prentice-Hall, Englewood Cliffs, NJ.

Fu, K. S., Gonzalez, R. C., and Lee. (1987). *Robotics: Control, Sensıng, Vision and Intelligence*, McGraw-Hill Inc., New York.

Fukunaga, K. (1992). *Introduction to Statistical Pattern Recognition*, 2nd ed., Academic Press, New York.

Gilio, W. K. (1978). *Interactive Computer Graphics*, Prentice Hall, Englewood Cliffs, NJ.

Gonzalez, R. C., and Thomason, M. G. (1978). *Syntactic Pattern Recognition: An Introduction*, Addison-Wesley, Reading, MA.

Gonzalez, R. C., and Wintz, P. A. (1977). *Digital Image Processing*, Addison-Wesley, Reading, MA.

Gonzalez, R. C., and Wintz, P. A. (1987). *Digital Image Processing*, 2nd ed., Addison-Wesley, Reading, MA.

Goodman, J. W. (1968). *Introduction to Fourier Optics*, McGraw-Hill, New York.

Grasseli, A., ed. (1969). *Automatic Interpretation and Classification of Images*, Academic Press, New York.

Haykın, S. (1994). *Neural Networks: A Comprehensive Foundation*, Macmillan, New York.

Hebb, D. O. (1949). *The Organization of Behavior, a Neuropsycological Theory*, John Wiley, New York.

Helstrom, C. W. (1968). *Statistical Theory of Signal Detection*, Pergamon Press, Elmsford, NY.

Hoffman, K., and Kunze, R. (1971). *Linear Algebra*, Prentice-Hall, Englewood Cliffs, NJ.

Hopcroft, J. E., and Ullman, J. D. (1969). *Formal Languages and Their Relation to Automata*, Addison-Wesley, Reading, MA.

Huang, T. S., ed. (1975). *Picture Processing and Digital Filtering*, Springer-Verlag, Berlin.

Huang, T. S. (1981). *Two-Dimensional Digital Signal Processing II—Transforms and Median Filters*, Springer-Verlag, Berlin.

Huang, T. S., and Tretiak, O. J., eds. (1972). *Proceedings of the 1969 Symposium on Picture Bandwidth Compressing*, Gordon and Breach, New York.

Hwang, K., and Brigg, (1984). *Computer Architecture and Parallel Processing*, McGraw-Hill, New York.

Jackson, P. C. (1974). *Introduction to Artificial Intelligence*, Mason-Charter, New York.

Jain, A. K. (1989). *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Jollife, I. T. (1986). *Principal Component Analysıs*, Springer-Verlag, New York.

Julesz, B. (1971). *Foundations of Cyclopian Perception*, University of Chicago Press, Chicago.

Kamp, Y., and Hasler (1990). *Recursive Neural Networks for Associative Memory*, John Wiley, Chichester, UK.

Kanal, L. N., ed. (1968). *Pattern Recognition*, Thompson Books, Washington, DC.

Kanal, L. N., and Rosenfeld, A., eds. (1981). *Progress in Pattern Recognition*, North-Holland, Amsterdam.

Kaneff, S. (1970). *Picture Language Machines*, Academic Press, New York.

Kasturi, R., and Trivedi, M., eds. (1990). *Image Analysis Applications*, Marcel Dekker, New York.

Klinger, A., Fu, K. S., and Kunii, T. L., eds. (1977). *Data Structures, Computer Graphics, and Pattern Recognition*, Academic Press, New York.

Kmoth, D. E. (1968). *The Art of Computer Programming*, vol. 1, Addison-Wesley, Reading. MA.

Kohonen, T. (1977). *Associative Memory: a system—theoretical approach*, Berlin: Springer Verlag.

Kohonen, T. (1984). *Self Organization and Associative Memory*, Berlin: Springer Verlag.

Kohonen, T. (1995). *Self-Organizing Maps*, Springer-Verlag, Berlin.

Kovalevsky, V. A. (1980). *Image Pattern Recognition*, Springer-Verlag, Berlin.

Lipkin, B. S., and Rosenfeld, A., eds. (1970). *Picture Processing and Psychopictures*, Academic Press, New York.

Loeve, M. (1948). *Fonctions aléatoires de second ordre*, Hermann, Paris, pp. 299–352.

Lowe, D. (1995). "Radial based functions." In M. A. Arbib (Ed.) *The Handbook of Brain Theory and Neural Networks*, Cambridge, MA, MIT Press.

Meisel, W. S. (1972). *Computer-Oriented Approaches to Pattern Recognition*, Academic Press, New York.

Mendel, J. M., and Fu, K. S. (1970). *Adaptive Learning and Pattern Recognition Systems*, Academic Press, New York.

Michie, D., Spiegelhalter, D. J., and Taylor, C. C., eds (1994). *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, Ltd., London.

Nadler, M., and Smith, E. P. (1993). *Pattern Recognition Engineering*, John Wiley, New York.

Morrison, D. F. (1976). *Multivariate Statistical Methods*, McGraw-Hill, New York.

Nake, F., and Rosenfeld, A., eds. (1972). *Graphic Language*, North-Holland, Amsterdam.

Nevatia, R. (1977). *Structured Descriptions of Complex Curved Objects from Recognition and Visual Memory*, Springer-Verlag, Berlin.

Newman, W. M., and Sproull, R. F. (1973). *Principles of Interactive Graphics*, McGraw-Hill, New York.

Nilsson, N. J. (1965). *Learning Machines*, McGraw-Hill, New York.

Oppenheim, A. V., and Schaefer, R. W. (1975). *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Pandya, A., and Macy, R. (1996). *Pattern Recognition with Neural Network in C + +*, CRS Press, Inc.

Patrick, E. A. (1972). *Fundamentals of Pattern Recognition*, Prentice-Hall, Englewood Cliffs, NJ.

Pavlidis, T. (1977). *Structural Pattern Recognition*, Springer-Verlag, New York.

Pearson, D. E. (1975). *Transmission and Display of Pictorial Information*, Wiley, New York.

Pratt, W. K. (1978). *Digital Image Processing*, Wiley, New York.

Preston, K., Jr. (1972). *Coherent Optical Computers*, McGraw-Hill, New York.

Reza, F. M. (1961). *An Introduction to Information Theory*, McGraw-Hill, New York.

Rogers, D. F., and Adams, J. A. (1976). *Mathematical Elements for Computer Graphics*, McGraw-Hill, New York.

Rogers, D. F., and Adams, B. (1977). *Princıples of Interactive Computer Graphıcs*, McGraw-Hill, New York.

Rosenblatt, F. (1961). *Principles of Neurodynamics: Perceptions and the Theory of Braın Mechanisms*, Spartan Books, Washıngton, DC.

Rosenfeld, A. (1969). *Picture Processing by Computer*, Academic Press, New York.

Rosenfeld, A., and Kak, A. C. (1976). *Digital Picture Processing*, Academic Press, New York.

Schalkoff, R. J. (1989). *Digital Image Processing and Computer Vision—An Introduction to Theory and Implementations*, Wiley, New York.

Sebestyen, G. S. (1962). *Decisıon Making Processes in Pattern Recognition*, ACM Monograph Series, Macmillan, New York.

Simpson, P. K. (1990). *Artificial Neural Systems*, Pergamon Press, Inc., Elmsford, NY.

Sklansky, J. (1973). *Pattern Recognition*, Dowden, Hutchinson & Ross, Stroudsburg, PA.

Strang, G., and Nguyen, T. (1996). *Wavelets and Filter Banks*, Wellesley-Cambridge Press.

Takanori, O. (1976). *Three-Dimensıonal Imagıng Techniques*, Academic Press, New York.

Tatsuoka, M. (1971). *Multivariate Analysis*, Wiley, New York.

Theodoridis, S., and Koutroumbas, K. (1999). *Pattern Recognition*, Academic Press, New York.

Tippet, J. T, ed. (1965). *Optical and Electro-Optical Information Processing*, MIT Press, Cambridge, MA.

Tou, J. T., and Gonzalez, R. C. (1974). *Pattern Recognition Prıncıples*, Addison-Wesley, Reading, MA.

Tsypkin, Y. Z. (1971). *Adaptation and Learning in Automatic Systems*, Academic Press, New York.

Tukey, J. W. (1971). *Exploratory Data Analysıs*, Addison-Wesley, Reading, MA.

Uhr, L., ed. (1966). *Pattern Recognition*, Wiley, New York.

Ullman, J. R. (1973). *Pattern Recognition Techniques*, Crane, Russak, New York.

Vetti, M., and Kovacevic, J. (1995). *Wavelets and Subband Coding*, Prentice-Hall, Englewood Cliffs, NJ.

Watanabe, S. (1969a). *Knowing and Guessing*, Wiley, New York.

Watanabe, S., ed. (1969b). *Methodologies of Pattern Recognition*, Academic Press, New York.

Watanabe, S. (1972). *Frontiers of Pattern Recognition*, Academic Press, New York.

Weizenbaum, J. (1976). *Computer Power and Human Reason*, Freeman, San Francisco.

Winton, P. H. (1977). *Artificial Intelligence*, Addison-Wesley, Reading, MA.

Wyszecki, G. W., and Stiles, W. S. (1967). *Color Science*, Wiley, New York.

Young, T. Y., and Calvert, T. W. (1974). *Classification, Estimation and Pattern Recognition*, American Elsevier, New York.

Yu, F. T. S. (1982). *Optical Information Processing*, Wiley, New York.

Zurada, J. M. (1992). *Introduction to Artificıal Neural Systems*, West Publishing Co.

Zusne, L. (1970). *Visual Perception of Form*, Academic Press, New York.

PAPERS AND JOURNALS

Abramson, N., and Braverman, D. (1962). Learning to recognize patterns in a random environment, *IRE Trans. Inf. Theory*, vol. IT-8, no. 5, pp. S58–S63.

Aeberhard, S., Coomans, D., and Devel, O. (1994). Comparative analysis of statistical Pattern recognition methods in high dimensional setting. *Pattern Recognition*, vol. 27, no 8, pp. 1065–1077.

Aggarwal, R. C., and Burrus, C. S. (1975). Number theoretic transforms to implement fast digital convolution, *Proc. IEEE*, vol. 63, pp. 550–560.

Agmon, S. (1954). The relaxation method for linear inequalities, *Can. J. Math.*, vol. 6, no. 3, pp. 382–392.

Ahuja, N. (1982). Dot pattern processing using voronoi neighborhood, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no. 3, May, pp. 336–343.

Ahuja, N., An, B., and Schachter, B. (1985). Image representation using Voronoi Tesselation, *Comput. Graphics*, *Image Process*, vol. 29, no. 3, March, pp. 286–295.

Aizerman, M. A., Braverman, E. M., and Rozoner, L. I. (1964a). The method of potential functions in the problem of determining the characteristics of a function generator from randomly observed points, *Autom. Remote Control*, vol. 25, no. 12, pp. 1546–1556.

Aizerman, M. A., Bravennan, E. M., and Rozoner, L. I. (1964b). Theoretical foundations of the potential function method in pattern recognition, *Autom. Remote Control*, vol. 25, no. 6, pp. 821–837.

Aizerman, M. A., Braverman, E. M., and Rozoner, L. I. (1965). The Robbins-Monro process and the method of potential functions, *Autom. Remote Control*, vol. 26, no. 11, pp. 1882–1885.

Allen, G. R., Bonrud, L. O., Cosgrove, J. J., and Stone, R. M. (1973). Machine processing of remotely sensed data, *Conf. Proc.*, Purdue University, Oct. 1973, IEEE Cat. 73CHO 834-2GE, pp. 1A25–1A42.

Alvertos, D., Brzakovic, D., and Gonzalez, R. C. (1989). Camera geometries for image matching in 3-D machine vision, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 9, Sept., pp. 897–915.

Amir, I. (1990). Algorithm for finding the center of circular fiducials, *Comput. Graphics, Image Process.*, vol. 49, no. 3, March, pp. 398–406.

Anderson, T. W., and Bahadur, R. R. (1962). Classification into two multivariate normal distributions with different covariance matrices, *Ann. Math. Stat.*, vol. 33, pp. 420–431.

Andrews, D. F. (1972). Plots of high-dimensional data, *Biometrics*, vol. 28, pp. 125–136.

Andrews, H. C. (1971). Multi-dimensional rotations in feature selection, *IEEE Trans. Comput.*, vol. C-20, no. 9, p. 1045.

Andrews, H. C. (1972). Some unitary transformations in pattern recognition and image processing, in *Information Processing*, vol. 71 (C. V. Freiman, ed.), North-Holland, Amsterdam.

Andrews, H. C. (1974). Digital image restoration: a survey, *IEEE Comput.*, vol. 7, no. 5, pp. 36–45.

Andrews, H. C., and Patterson, C. L. (1976). Digital interpolation of discrete images, *IEEE Trans. Comput.*, vol. C-25, no. 2, pp. 196–202.

Ansari, N., and Delp, E. J. (1990). Partial shape recognition: a landmark-based approach, *IEEE Trans. on P.A.M.I.*, vol. 12, no. 5, pp. 470–483.

Arcelli, C., and Levialdi, S. (1971). Concavity extraction by parallel processing, *IEEE Trans. Syst. Man Cybern.*, vol. 1, pp. 349–396.

Arcelli, C., and Levialdi, S. (1972). Parallel shrinking in three dimensions, *Comput. Graphics Image Process.*, vol. 1, pp. 21–30.

Argyle, E. (1971). Techniques for edge detection, *Proc. IEEE*, vol. 59, no. 2, pp. 285–287.

Asano, A., Itoh, K., and Ichioka, Y. (1990). The nearest neighbor median filter: some deterministic properties and implementations, *Pattern Recognition*, vol. 23, no. 10, pp. 1059–1066.

Asada, H., Brady, M. (1986). The curvature primal sketch, *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 1, Jan., pp. 2–14.

Augustson, J. G., and Minker, J. (1970). An analysis of some graph-theoretical cluster techniques, *J. ACM*, vol. 17, no. 4, pp. 571–588.

Aurenhammer, F., and Edelsbrunner, H. (1984). An optimal algorithm for constructing the weighted voronoi diagram in the plane, *Pattern Recognition*, vol. 17, no. 2, pp. 251–258.

Avtonomova, V. A. (1973). A statistical model of the observation process in pattern recognition problems, *Autom. Remote Control*, vol. 34, no. 10, pt. 1, pp. 1566–1574.

Avtonomova, V. A., and Kholodilov, Y. M. (1973). Expenmental investigation of the visual image recognition characteristics, *Autom. Remote Control*, vol. 34, no. 8, pt. 2, pp. 1342–1346.

Babu, C., and Chitti, V. (1973). On the application of probabilistic distance measures for the extraction of feature from imperfectly labeled patterns, *Int. J. Comput. Inf. Sci.*, vol. 2, no. 2, pp. 103–114.

Badhwar, G. D., Lyndon, B., Austın, W. W., and Carnes, J. G. (1982). A semi-automatic technique for multitemporal classification of a given crop wıthin a Landsat scene, *Pattern Recognition*, vol. 15, no. 3, pp. 217–230.

Bakke, K. I., and McMurtry, G. J. (1969). A pattern recognition algorithm using the concept of intrinsic dimensionality, *Proc. Purdue Symp. Inf. Process.*, Apr. 1969, pp. 446–452.

Ball, G. H. (1965). Data analysis ın the social sciences: What about the details? *Proc. Fall Joint Comput. Conf.*

Ball, G. H., and Hall, D. J. (1965a). Isodata, an iterative method of multivariate analysis and pattern classification, *Proc. IFIPS Congr.*

Ball, G. H., and Hall, D. J. (1965b). Isodata, a novel method of data analysis and pattern classification, *NTIS Rep. AD699616*, Tech. Rep. of SRI project, Stanford Research Institute, Menlo Park, CA.

Bargel, B. (1980). Classification of remote sensed data by texture and shape features in different spectral channels, *Proc. 5th Int. Conf. Pattern Recognition*, Miami Beach, FL, Dec. 1–4, 1980 (IEEE, New York, 1980), pp. 2–4.

Barnard, S. T. (1980). Automated inspection using grey-scale statistics, *Proc. 5th Int. Conf. Pattern Recognition*, Miami Beach, FL, Dec. 1–4, 1980, (IEEE, New York, 1980), pp. 269–272.

Barnes, D. I., and Silverman, H. E. (1972). A class of algorithms for fast digital image registration, *IEEE Trans. Comput.*, vol. C-21, no. 2, pp. 179–186.

Bartels, R. H., Beatty, J. C., and Barsky, B. A. (1990). An introduction to splines use in computer graphics and geometric modeling, *Comput. Graphics, Image Proces.*, vol. 50, no. 1, April, pp. 125–126.

Batchelor, B. G. (1973). Instability of the decision surfaces of the nearest-neighbor a potential function classifiers, *Inf. Sci.*, vol. 5, p. 179.

Batchelor, B. G. (1978). Experimental and pragmatic approaches to pattern recognition, *Kybernetes*, vol. 7, no. 4, pp. 269–277.

Batchelor, B. G., and Hand, D. J. (1975). On the graphical analysis of PDF estimators for pattern recognition, *Kybernetes*, vol. 4, p. 239.

Batchelor, B. G., and Wilkins, B. R. (1969). Method for location of clusters of pattern to initialize a learning machine. *Electron. Lett.*, vol. 5, no. 20, pp. 481–483.

Bebb, J. E., and Stromberg, W. D. (1973). An overview of image processing, *System Development Corp. Rep. AD-757119.*

Becker, P. W., and Nielson, D. A. (1972). Pattern recognition using dynamic pictorial information, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-2, no. 3, pp. 434–437.

Beni, G., and Liu, X. (1994). A least biased fuzzy clustering method, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 9, pp. 954–960.

Benjamin, R. (1980). Generalizations of maximum entropy pattern analysis, *Proc. IEEE*, vol. 27, no. 5, pp. 341–353.

Bennett, J. R., and MacDonald, J. S. (1975). On the measurement of curvature in a quantized environment, *IEEE Trans. Comput.*, vol. C-2, no. 8, pp. 803–820.

Bergholm, F. (1987). Edge focusing, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 6, Nov., pp. 726–741.

Bergland, D. (1969). Fast Fourier transform hardware implementation: a survey, *IEEE Trans. Audio Electroacoust.*, vol. AU-17, pp. 109–119.

Bidasaria, H. B. (1986). A method for almost exact histogram matching for two digitized images, *Comput. Graphics, Image Process.*, vol. 34, no. 1, April, pp. 93–98.

Billingsley, F. C. (1970). Application on digital image processing, *Appl. Opt.*, vol. 2, no. 2, pp. 289–299.

Billingsley, F. C. (1972). Digital image processing for information extraction, in *Machine Perceptions of Pattern and Pictures*, Conf. Ser. 13, Institute of Physics, London, pp. 337–362.

Bishop, C. M. (1991). Improving the generalization properties of radial basis function neural networks, *Neural Computation*, vol. 3, no. 4, pp. 579–588.

Bley, H. (1984). Segmentation and peprocessing of electrical schematics using picture graphs, vol. 28, no. 3, Dec., pp. 271–288.

Block, H. D. (1962). The perceptron: a model for brain functioning—I, *Rev. Mod. Phys.*, vol. 34, no. 1, pp. 123–135.

Block, H. D., Nilsson, N. J., and Duda, R. O. (1964). Determination and detection of features in patterns, in *Computer and Information Sciences*, vol. I. (J. T. Tou and R. H. Wilcox, eds.), Spartan Books, Washington, DC.

Blostein, D., and Ahuja, N. (1989). Shape from texture: integrating texture-element extraction and surface estimation, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 12, Dec., pp. 1233–1251.

Blum, H. (1964). A transformation of extracting new descriptions of shape, *Symp. Models for the Perception of Speech and Visual Form*, MIT Press, Cambridge, MA.

Boberg, J., and Salakoski, T. (1993). General formulation and evaluation of agglomerative clustering methods with metric and non-metric distances, *Pattern Recognition*, vol. 26, no. 9, pp. 1395–1406.

Boerner and Stretcker (1988). Automated X-ray inspection of aluminum castings, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 1, Jan., pp. 79–91.

Bouthemy, P. (1989). A maximum likelihood framework for determining moving edges, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 5, May, pp. 499–511.

Bovik, A. C., Huang, T. S., and Munson, D. C. (1987). The effect of median filtering on edge estimation and detection, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 2, March, pp. 181–194.

Bovic, A. C., Clark, M., and Geisler, W. S. (1990). Multichannel texture analysis using localized spatial filters, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 1, pp. 55–73.

Bow, S. T. (1962). Graphical analysis of the non-linear power system oscillation, *Sci. Sin.*, vol. 11, no. 1, 1960, pp. 131–144.

Bow, S. T. (1963). Matrix analysis of pulse wave propagation on multi-conductor system, *Sci. Sin.*, vol. 12, no. 2, pp. 245–270.

Bow, S. T. (1979a). Morphological analysis of simplified Chinese ideographs and a heuristic approach for their machine recognition, *Proc. COMPSAC 79*, Chicago, 1979.

Bow, S. T. (1979b). Development of a page reader for western language manuscript, *Automation*, vol. III, no. 3, pp. 93–109.

Bow, S. T. (1979c). Figures and several approaches for line-shaped figure processing, *Automation*, vol. III, no. 2, pp. 56–80.

Bow, S. T. (1980). Structural approach applicable to primitive description and extraction for complex Chmese ideograph recognition, *Proc. 5th Int. Conf. Pattern Recognition*, Miami Beach, Fla., Dec. 1–4, 1980; IEEE, New York, 1980.

Bow, S. T. (1982). Some considerations on the machine recognition of Chinese ideographs, *Proc. 1982 Int. Conf. Chinese Language Comput. Soc.*, Washington, DC.

Bow, S. T. (1986). Autonomous control of image sensor for the optimal acquisition of ground information for dynamic analysis, *SPIE Proc.*, vol. 729, pp. 260–264.

Bow, S. T. (1988). Remote sensing activities in China: a national report, *Proc. 8th Asian Conf. Remote Sensing*, Bangkok, Thailand, Nov. 1988, pp. J-10-1–J-10-6.

Bow, S. T. (1992). Pattern recognition technique applicable to quality control and assurance, *Proc. IASTED 20 Int. Conf. Reliability, Quality Control, and Risk Assessment*, pp. 121–124.

Bow, S. T. (1994). Visibility enhancement during the foggy weather condition, *Proc. ISATA 26th Int. Symp. Automotive Technology and Automation*, pp. 191–198.

Bow, S. T., and Chen, P. (1990). Computerized detection and identificatıon of the types of defects on crystal blanks, *SPIE Proc.*, vol. 1396, MidWest Conference, pp. 646–655.

Bow, S. T., and Chen, T. (1989). Computer Detection of Indiscernible Tiny Scratches and Cracks on Crystal, *Proc. SPIE 33rd General Conf.*, San Diego, CA, 1989.

Bow, S. T., and Chen, T. (1989a). Image processing technique applicable to the detection of tiny scratches and cracks on crystal under microscope, *Proc. 20th Annual Modeling and Simulation Conference*, pp. 483–487.

Bow, S. T., and Chen, T. (1989b). Detection and identification of visuall indiscernible defects from textural background of an unpolished quartz crystal, *Proc. SPIE Intelligent Robotics Systems and Computer Vision VIII*, vol. 1153, pp. 121–128.

Bow, S. T., and Cheng, Q. (1998). Denoising of Images with Extremely High Noise Content for Real Time Applications, *IEEE ICIPS '98 Proc. Second IEEE International Conference on Intelligent Processing Systems*, pp. 353–357.

Bow, S. T., and Cheng, Q. (1999). Wavelet based noise reduction for images with extremely high noise content, *Intelligent Engineering System through Artificial Neural Networks*, vol. 9, pp. 275–280.

Bow, S. T., and Chu, L. S. (1960). Automatic power system swing curve computing device, *Sci. Rec.*, no. 2, pp. 133–138.

Bow, S. T., and El-Masri (1987). Knowledge-based graphics understanding and description for document archival and retrieval, *SPIE Proc.*, vol. 848, pp. 640–647.

Bow, S. T., and Honnenahalli, S. (1988). Piecewise representation of curves with concatenated arcs via detection of the appropriate break points, *Proc. IEEE Int. Conf. Robotics and Automation.*

Bow, S. T., and Kasturi, R. (1990a). A graphics recognition system for interpretation of line drawings, in *Image Analysis Mach. Intell. (PAMI)*, vol. 12, no. 10, pp. 978–992.

Bow, S. T., and Kasturi, R. (1990b). Graphics-recognition system for interpretation of line drawings, in *Image Analysis Applications* (R. Kasturi and M. M. Trivedi, eds.), Marcel Dekker, New York, chap. 2, pp. 37–72.

Bow, S. T., and Kim, W. Y. (1982). Preliminary investigation on the structure of Korean characters and their machine recognition, *Proc. 1982 Int. Conf. Chinese Language Comput. Soc.*, Washington, DC.

Bow, S. T., and Qiu, Y. (1999). Wavelet based morphological processing for image noise reduction, *Intelligent Engineering System through Artificial Neural Networks*, vol. 9, pp. 287–292.

Bow, S. T., and Sa, J. (1992a). An algorithm to generate a succinct description for a complicated shape, *SPIE Proc.*, vol. 1778, pp. 142–153.

Bow, S. T., and Sa, J. (1992b). Algorithms to separate text from a mixed text/graphic document and generate a succinct description for this complex graphic, *SPIE Proc. 1771*, pp. 213–224.

Bow, S. T., and Sa, J. (1992c). Automated generation of a succinct description for a complicated shape, *Proc. SPIE Int. Symp. Applications of Digital Image Processing XV*, vol. 1778, pp. 142–153.

Bow, S. T., and Sun, Y. L. (1996). Wavelet transform for color image processing, *Proc. IEEE Int. Conf. Image Processing*, Sept. 1996, pp. 541–544.

Bow, S. T., and Sun, Y. L. (1997). Wavelet transform for monochrome and color image compression, *Proc. SPIE Int. Conf. Visual Information Processing VI*, vol. 3074, pp. 90–101.

Bow, S. T., and Van Ness, J. E. (1958). Use of phase space in transient stability studies, *Trans. AIEE*, pt. II, p. 77.

Bow, S. T., and Wang, D.-H. (1989). Verification of spring structure by image processing technique, *SPIE Proc. MidCon., Electronic Show and Convention*, Chicago, 1989.

Bow, S. T., and Wang, X. F. (1989a). On the application of pattern recognition and AI technique to the cytoscreening of vaginal smears by computer, *Proc. SPIE Int. Conf. Medical Imaging III*, Newport Beach, CA.

Bow, S. T., and Wang, X. F. (1989b). On the application of pattern recognition and AI technique to the cytoscreening of vaginal smears by computer, *Proc. SPIE Int. Conf. Med. Imaging III*, Newport Beach, CA, Jan.

Bow, S. T., and Wu, J. C. (1998). An efficient image coding scheme via tree structure of wavelet coefficients, *Intelligent Engineerıng System through Artificıal Neural Networks*, vol. 8, pp. 473–478.

Bow, S. T., and Yu, G.-Y. (1981). On board data pre-editing for remote sensing, conference paper, Second Asian Conference on Remote Sensing, Oct. 29–Nov. 3, Peking, China.

Bow, S. T., and Zhang, W. (1994). An effective artchitecture for the processing of ımages in spatial domain with artificial neutral networks, *Proc. ANNIE '94 (Neural Networks).*

Bow, S. T., and Zhou, B. (1986). Automated generation of the structural description for graphics by computer, *Proc. 31st Int. Symp. Mini and Microcomputers and Applications* (MIMI '86), pp. 58–62.

Bow, S. T., and Zhou, B. (1988). Automated generation of the structural description for graphics by computer, *Int. J. Mini and Microcomputers*, vol. 10, no. 1, pp. 9–13.

Bow, S. T., Chen, T. S., and Honnenahalli, S. (1988). Automated generation of concatenated arcs for curve representation, *SPIE Proc.*, vol. 1002, pp. 68–74.

Bow, S. T., Chen, T., and Newell, D. E. (1989). Detection and identification of visually indiscernible defects from textural background of an unpolished quartz crystal, *SPIE Proc.*, vol. 1197, pp. 206–217.

Bow, S. T., Chen, T. S., Wang, D. H., and Newell, D. E. (1989). Computer-aided high precision verification of miniature spring structure, *SPIE Proceedings*, vol. 1153, pp. 106–112, 1989.

Bow, S. T., Chen, T. S., et al. (1989). Computer inspection for spring production, *SPIE 33rd Gen. Conf.*, San Diego, CA.

Bow, S. T., Chen, Pei, Newell, D. E., Miernicki, L., et al. (1990). Automated detectıon and identification of crystal defects with image processing technique, *Proc. 12th Piezoelectric Devices Conf. Exhib.*, pp. 20–39.

Bow, S. T., Chen, P., Chen, T. (1991). Identification of submits defects on crystal blanks, *ISMM Proc. Int. Sym. Comput. Appl. Design, Simulation, Analysis*, pp. 154–157.

Bow, S. T., Kasturi, R., et al. (1988). A system for recognition and description of graphics, *Proc. 9th Int. Conf. Pattern Recognition*, pp. 255–259.

Bow, S. T., Kasturi, R., et al. (1990). A system for interpretation of line drawings, *IEEE trans. Pattern Analysis Mach. Intell. (PAMI)*, vol. 12, no. 10, pp. 978–992.

Bow, S. T., Wang, C. C., Yau, H. Y., and Tsou, K. N. (1964). Logic control technique applicable to power system fault diagnosis and handling, *Sci. Sin.*, vol. 13, no. 8, pp. 119–122.

Bow, S. T., Wang, X., and Zhang, J. (1992). Using the technique of color image processıng to enhance the cytomorphological deformation to detect cancer, *Proc. SPIE Int. Symp. Applications of Digital Image Processing XV*, vol. 1771, pp. 183–194.

Bow, S. T., Yu, R. T., and Zhou, B. (1985). Optimal acquisition of ground information for dynamic analysis, *Proc. Int. Conf. Advances in Image Processing and Pattern Recognition*, pp. 35–37. Also appeared in V. Cappellini and R. Marconi, eds., *Advances in Image Processing and Pattern Recognition*, Elsevier, New York, 1986, pp. 21–26.

Bow, S. T., Zhang, J., and Wang, X. F. (1992). Enhancing the cytomorphological deformation with color image processing, *SPIE Proc.*, vol. 1778, pp. 52–63.

Bow, S. T., Zhang, J., and Wang, X. (1993). Highlighting the differences between the positive and pseudo-positive cyto-architectonics, *SPIE Proc.*, vol. 1898 (Medical Imaging), pp. 342–352.

Bow, S. T., Yu, R. T., and Zhou, B. (1986). Optimal acquisition of ground information for dynamic analysis, in *Advances and Image Processing and Pattern Recognition* (V. Cappellini and R. Marconi, eds.), Elsevier, B.V., North Holland.

Brakke, K. A., Mantock, J. M., and Fukungawa, K. (1982). Systematic feature extraction, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no. 3, pp. 291–297.

Brauner, R., and Epstein, D. (1981). Automated chip (die) inspection, *Int. J. Hybrid Microelectron*, vol. 4, no. 2, pp. 111–115.

Braveman, E. M. (1965). On the method of potential functions, *Autom. Remote Control*, vol. 26, no. 12, pp. 2130–2138.

Brooks, R. A., and Bovik, A. C. (1990). Robust techniques for edge detection in multiplicative Weibull image noise, *Pattern Recognition*, vol. 23, no. 10, pp. 1047–1057.

Brown, B. R., and Evans, S. H. (1969). Perceptual learning in pattern discrimination tasks with two and three scheme categories, *Psychonom. Sci.*, vol. 15, no. 3, pp. 101–103.

Brown, D., Hall, M., and Lal, S. (1970). Pattern transformation by neural nets, *J. Physiol.*, vol. 209, p. 7.

Brown, N., Oreb, B. F., and Hariharen, p. (1982). Pseudocolor picture display with a microcomputer, *J. Phys. E.*, vol. 15, no. 7, pp. 703–704.

Brown, R. (1963). Logical properties of adaptive networks, *Stanford Electron. Lab. Q. Res. Rev.*, no. 4.

Browning, J. D., and Tanimoto, S. L. (1982). Segmentation of pictures into regions with a tile-by-tile method, *Pattern Recognition*, vol. 15, no. 1, pp. 1–10.

Bryant, A., and Bryant, J. (1989). Recognizing shapes in planar binary images, *Pattern Recognition*, vol. 22, no. 2, pp. 155–164.

Brzakovic, D., Beck, H., and Sufi, N. (1990). An approach to defect detection in materials characterized by complex textures, *Pattern Recognition*, vol. 23, no. 1/2, pp. 99–108.

Bumbaca, F., and Smith, K. C. (1987). Design and implementation of a colour vision model for computer vision applications, *Comput. Graphics, Image Process.*, vol. 39, no. 2, Aug., pp. 226–245.

Burckhardt, C. B. (1968). Information reduction in holograms for visual displays, *J. Opt. Soc. Am.*, vol. 58, no. 2, pp. 241–246.

Burns, J. B., Hanson, A. R., and Riseman, E. M. (1986). Extracting straight lines, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 4, July, pp. 425–455.

Bush, D. A. (1981). Automatic feature extraction system test bed, *IEEE Comput. Soc. Conf. Pattern Recognition, Image Process.*, Dallas, TX, Aug. 3–5, 1981, pp. 615–617.

Butt, E. B., et al. (1968). Studies on visual textual manipulation and synthesis, *Tech. Rep. 68–64*, Computer Science Center, University of Maryland, College Park.

Buturovic, L. J. (1993). Improving k-nearest neighbor density and error estimates, *Pattern Recognition*, vol. 26, no. 4, pp. 611–616.

Caelli, T., and Nagendran, S. (1987). Fast edge-only matching techniques for robot pattern recognition, *Comput. Graphics, Image Process.*, vol. 39, no. 2, Aug., pp. 131–143.

Canny, J. (1986). A computational approach to edge detection, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, Nov., pp. 679–698.

Carman, C. S., and Merickel, M. B. (1990). Supervising ISODATA with an information theoretic stopping rule, *Pattern Recognition*, vol. 23, no. 1/2, pp. 185–197.

Carpenter, G. A., and Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine, *Comput. Graphics, Image Process.*, vol. 37, no. 1, Jan., pp. 54–115.

Carpenter, G. A., and Grossburg, S. (1988). The art of adaptive pattern recognition by a self-organizing neural network, *IEEE Computer*, vol. 21, no. 3, March, pp. 77–88.

Capson, D. W. (1984). An improved algorithm for the sequential extraction of boundaries from a raster scan, *Comput. Graphics, Image Process.*, vol. 28, no. 1, Oct., pp. 109–125.

Caulfield, H. J., Haimes, R., and Draper, J. S. (1979). Preprocessing of multispectral images for image processing, *Proc. Soc. Photo-opt. Instrum. Eng.*, vol. 186, pp. 131–134.

Cavanagh, P. (1987). Reconstructing the third dimension: interaction between color, texture, motion, binocular disparity and shape, *Comput. Graphics, Image Process.*, vol. 37, no, 2., Feb., pp. 171–195.

Celenk, M. (1990). A color clustering technique for image segmentation, *Comput. Graphics, Image Process.*, vol. 52, no. 2, Nov., pp. 145–170.

Chang. C. L., and Lee, R. C. T. (1973). A heuristic relaxation method for nonlinear mapping in cluster analysis, *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-3, no. 2, pp. 197–200.

Chang, S. K. (1971). The reconstruction of binary patterns from their projection, *Commun. ACM*, vol. 14, no. 1, pp. 21–25.

Chang, T., and Kuo, C. C. J. (1993). Texture analysis and classification with tree structured wavelet transform, *IEEE Trans. Image Proc.*, vol. 2, no. 4, pp. 429–442.

Charnes, A. (1964). On some fundamental theorems of perception theory and their geometry, in *Computer and Information Sciences*, vol. I (J. T. Tou and R. H. Wilcox, eds.), Spartan Books, Washington, DC.

Chaudhuri, B. B. (1982). Bayes' error and its sensitivity in statistical pattern recognition in noisy environment, *Int. J. Syst. Sci.*, vol. 13, no. 5, pp. 559–570.

Chaudhuri, B. B. (1985). Applications of quadtree, octree, and binary tree decomposition techniques to shape analysis and pattern recognition, *IEEE Trans. Pattern Ansl. Mach. Intell.*, vol. PAMI-7, no. 6, Nov., pp. 652–661.

Chaudhury, S., Acharyya, A.., Subramanian, S., and Parthasarathy, G. (1990). Recognition of occluded objects with heuristic search, *Pattern Recognition*, vol. 23, no. 6, pp. 617–635.

Chen, C. H., and Yen, C. (1982). Object isolation in FLIR images using Fischer's linear discriminant, *Pattern Recognition*, vol. 15, no. 3, pp. 153–159.

Chen, J. S., Huertas, A., and Medioni, G. (1987). Fast convolution with Laplacian-of-Gaussian masks, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. MAPI-9, no. 4, July, pp. 584–590.

Chen, J. S., and Medioni, G. (1989). Detection, localization, and estimation of edges, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 2, pp. 191–198.

Chen, L. H., and Tsai, W. H. (1988). Movement preserving line detection, *Pattern Recognition*, vol. 21, no. 1, pp. 45–54.

Chen, M. H., and Yan, P. F. (1989). A multiscaling approach based on morphological filtering, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, July, pp. 694–700.

Chen, P. H., and Wintz, P. A. (1976). Data Compression for satellite images, *TR-EE-76-9*, School of Electrical Engineering, Purdue University, West Lafayette, Ind.

Chien, C. H., and Aggarwal, J. K. (1984). A normalized quadtree representation, *Comput. Graphics, Image Process.*, vol. 26, no. 3, June, pp. 331–346.

Chien, C. H., and Aggarwal, J. K. (1989). Model construction and shape recognition from occluding contours, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 4, April, pp. 372–389.

Chien, Y. T. (1970). The threshold effect of a nonlinear learning algorithm for pattern recognition, *Inf. Sci.*, vol. 2, pp. 351–368.

Chien, Y. T. (1976). Interactive pattern recognition: techniques and system, *IEEE Comput.*, vol. C-9, no. 5, May 1, 1976.

Chien, Y. T., and Fu, K. S. (1967). On the generalized Karhumen-Loeve expansion, *IEEE Trans. Inf. Theory*, vol. IT-13, no. 3, pp. 518–520.

Chien, Y. T., and Ribak, R. (1971). Relationship matrix as a multidimensional data base for syntactic pattern generation and recognition, *Proc. Two-Dimensional Signal Process. Conf.*, University of Missouri, Columbia.

Chin, R. T. (1982). Automated visual inspection techniques and applications: a bibliography, *Pattern Recognition*, vol. 15, no. 4, pp. 343–358.

Chin, R. T., Wan, H. K., Stover, D. L., and Iverson, R. D. (1987). A one-pass thinning algorithm and its parallel implementation, *Comput. Graphics, Image Process.*, vol. 40, no. 1, Oct., pp. 30–40.

Chittineni, C. B. (1982). Some approaches to optimal cluster labeling with applications to remote sensing, *Pattern Recognition*, vol. 15, no. 3, pp. 201–216.

Chomsky, N. (1956). Three models for the description of language, *POIT*, vol. 2, no. 3, pp. 113–124.

Chowdhury, N., and Murthy, C. A. (1997). Minimal spanning tree based clustering technuque: relationship with Bayes' classifier, *Pattern Recognition*, vol. 30, no. 11, pp. 1919–1929.

Chuang, G. C. H., and Kuo, C. C. J. (1996). Wavelet descriptor of planar curves: theory and applications, *IEEE Trans. Image Proc.*, vol. 5, no. 1, pp. 56–71.

Clark, D. C., and Gonzalez, R. C. (1981). Optimal solution of linear inequalities with applications to pattern recognition, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, no. 6, pp. 643–655.

Clark, J. J. (1989). Authenticating edges produced by zero-crossing algorithms, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 1, pp. 43–57.

Clowes, M. B. (1969). Transformational grammars and the organization of pictures, in *Automatic Interpretation and Classification of Images* (A. Grasselli, ed.), Academic Press, New York.

Clowes, M. B. (1971). On seeing things, *Artif. Intell.*, vol. 2, pp. 79–116.

Clowes, M. B. (1972). Scene analysis and picture grammars, in *Machine Perception of Pattern and Pictures*, Conf. Ser. 13, Institute of Physics, London, pp. 243–256.

Cofer, R. H. (1972). Picture acquisition and graphical preprocessing system, *Proc. 9th Annu. IEEE Region III Conv.*, Charlottesville, Va.

Cofer, R. H., and Tou, J. T. (1971). Preprocessing for pictorial pattern recognition, *Proc. 21st NATO Tech. Symp. Artif. Intell.*, Italy.

Cofer, R. H., and Tou, J. T. (1972). Automated map reading and analysis by computer, *Proc. Fall Joint Comput. Conf.*

Cohen F. S., Fan, Z., and Attali, L. S. (1991).Automated inspection of textile fabrics using textural models, *IEEE Trans. PAMI*, vol. 13, no. 8, pp. 803–808.

Coifman, R. R., Meyer, Y., and Wickerhauser, M. V. (1992). Wavelet analysis and signal processing, in *Wavelets and Their Applications* (M. B. Ruskai et al., eds.), pp. 153–178.

Collins, D. C., and Meisel, W. S. (1971). Structural analysis of biological wave forms, *Proc. Conf. Eng. Med. Biol.*, Las Vegas, Nev., Oct. 31–Nov. 4, 1971.

Colwell, R. N. (1965). The extraction of data from aerial photograph by human and mechanical means, *Photogrammetrics*, vol. 20, pp. 211–228.

Connelly, S., and Rosenfeld, A. (1990). A pyramid algorithm for fast curve extraction, *Comput. Graphics, Image Process.*, vol. 49, no. 3, March, pp. 332–345.

Cooley, J. W., and Turkey, J. W. (1965). An algorithm for the machine computation of complex Fourier series, *Math. Comp.*, vol. 19, pp. 297–301.

Cooper, D. R., and Cooper, P. W. (1964). Nonsupervised adaptive signal detection and pattern recognition, *Inf. Control*, vol. 7, no. 3, pp. 416–444.

Cooper, P. W. (1964). Hyperplanes, hyperspheres and hyperquadrics as decision boundaries, in *Computer and Information Sciences*, vol. I (J. T. Tou, and R. H. Wilcox, eds.), Spartan Books, Washington, DC.

Cooper, P. W. (1967). Some topics in nonsupervised adaptive detection for multivariate normal distributions in *Computer and Information Sciences*, vol. II (J. T. Tou, ed.), Academic Press, New York.

Cormack, R. M. (1971). A review of classification, *J. R. Stat. Soc., Ser., A*, vol. 134, pt. E.

Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications to pattern recognition, *IEEE Trans. Electron. Comput.*, vol. EC-14, no. 3, pp. 326–334.

Cover, T. M. (1969). Learning in pattern recognition, in *Methodologies of Pattern Recognition* (S. Watanabe, ed.), Academic Press, New York.

Cover, T. M., and Hart, P. E. (1967). Nearest-neighbor pattern classification, *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27.

Crespi-Reghizzi, S. (1971). An effective model for grammar inference, *IFIP Congr.—71*, Yugoslavia.

Daniell, G. J. (1980). Maximum entropy algorithm applied to image enhancement, *Proc. IEEE*, vol. 127, no. 5, pp. 170–172.

Darwish, A. M., and Jain, A. K. (1988). A rule-based approach for visual pattern inspection, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, No. 1, pp. 56–68.

Davies, D. L., and Bouldin, D. W. (1979). A cluster separation measure, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 224–227.

Davies, L. S. (1980). Computer architectures for image processing, *Proc. Workshop Picture Data Descr. Manag.* (IEEE, New York, 1980), pp. 249–254.

Davis. W. A., and Tychon (1986). Texture boundaries in digital images, *Proc. 8th Int. Conf. Pattern Recognition*, pp. 402–404.

DeCastro, E., and Morandi, C. (1987). Registration of translated and rotated image using finite Fourier transform, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, pp. 700–703.

Dehne, J. S. (1978). Image processing technique for real-time imagery, *Pattern Recognition Signal Process.*, Paris, June 25–July 4, 1978.

Denes, P. B. (1975). A scan type graphics system for interactive computing, *Proc. Conf. Comput. Graphics, Pattern Recognition Data Struc.*, May 1975 (IEEE Cat, 75CHO 981-1C), pp. 21–24.

Derin, H., and Elliott, H. (1987). Modeling and segmentation of noisy and textured images using Gibbs random fields, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol PAMI-3, no. 1, Jan., pp. 39–55.

De Souza, P. (1982a). Some decision network designs for pattern classification, *Pattern Recognition*, vol. 15, no. 3, pp. 193–200.

De Souza, P. (1982b). Texture recognition via autoregression, *Pattern Recognition*, vol. 15, no. 6, pp. 471–476.

De Floriani, L., Falcidieno, B., and Pienovi, C. (1985). Delaunay-based representation of surface defined over arbitrarily shaped domains, *Comput. Graphics, Image Process.*, vol. 32, no. 1, Oct., pp. 127–140.

De Floriani, L., Falcidieno, B., and Pienovi, C. (1989). Structured graph representation of a hierarchical triangulation, *Comput. Graphics, Image Process.*, vol. 45, no. 2, Feb., pp. 215–226.

Diday, E. (1973). The dynamic clusters method in non-hierarchical clustering, *Int. J. Comput. Inf. Sci.*, vol. 2, no. 1, pp. 61–88.

Diday, E., and Celeux, G. (1981). Optimization in cluster analysis, *Methods Oper. Res.*, no. 43, p. 327.

Djouadi, A., and Bouktache, E (1997). Afast algorithm for the nearest neighbor classifier, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 3, pp. 277–282.

Dosrst, L., and Smeulders, A. W. M. (1987). Length estimators for digitized contours, *Comput. Graphics, Image Process.*, vol. 38, no. 3, Dec., pp. 311–333.

Duabechies, I. Ten Lectures on Wavelets, SIAM, Philadelphia, 1991.

Duan, J. R., and Wintz, P. A. (1974) Information preserving coding for multispectral data, *Proc. IEEE Conf. Mach. Process. Remotely Sensed Data.*

Duan, J. R., and Wintz, P. A. (1974). Information preserving coding for multispectral scanner data, *TR-EE-74-15*, School of Electrical Engineering, Purdue University, West Lafayette, IN.

duBuf, J. M. H., Kardan, M., and Spann, M. (1990). Texture feature performance for image segmentation, *Pattern Recognition*, vol. 23, no. 3/4, pp. 291–309.

Duda, R. O., and Fossum, H. (1966). Pattern classification by iteratively determined linear and piecewise linear discriminant functions, *IEEE Trans. Electron. Comput.*, vol. EC-15, no. 2, pp. 220–232.

Duff, M. J. B., Norgren, P., Preston, J., Jr., and Toriwaki (1978). Theoretical and practical consideration in the application of neighborhood logic to image processing, *Proc. 4th Int.. Joint Conf. Pattern Recognition*, Kyoto, Japan, Nov. 7–10, 1978, pp. 139–145.

Duff, M. J. B., Watson, D. M., and Deutsch, E. S. (1974). A parallel computer for array processing, *Inf. Process—74 (Proc. JFIP Congr.)*, pp. 94–97.

Duff, M. J. B., Watson, D. M., Fountain, T. J., and Shaw, G. K. (1973). A cellular logic array for image processing, *Pattern Recognition*, vol. 5, pp. 229–247.

Dunham, J. G. (1986). Optimum uniform piecewise linear approximation of planar curves, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 1, Jan., pp. 67–75.

Dvoretzky, A. (1956). On stochastic approximation, in *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability* (J. Neyman, ed.), University of California Press, Berkeley, pp. 39–55.

Dyer, C. R., and Rosenfeld, A. (1981). Parallel image processing by memory-augmented cellular automata, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, no. 1, pp. 29–41.

Edwards, J. A., and Fitelson, M. M. (1973). Notes on maximum entropy processing, *IEEE Trans. Inf. Theory*, vol. IT-19, no. 2, pp. 232–234.

Ehrich, R. W. (1978). A view of texture topology and texture description, *Comput. Graphics, Image Process.*, vol. 8, no. 2, pp. 174–202.

Eichmann, G., and Kasparis, T. (1988). Topologically invariant texture descriptors, *Comput. Graphics, Image Process.*, vol. 41, no. 3, March, pp. 267–281.

Eklundh, J. O. (1972). A fast computer method for matrix transposing, *IEEE Trans. Comput.*, vol. C-21, July, pp. 801–803.

Eklundh, J. O., and Rosenfled, A. (1981). Image smoothing bases based on neighbor linking, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, no. 6, pp. 679–683.

Evans, T. G. (1971). Grammatical inference techniques in pattern analysis, in *Software Engineering* (J. T. Tou, ed.), Academic Press, New York.

Fainzil'berg, L. S. (1978). Pattern recognition methods in thermographic metal composition analysis, *Cybernetics*, vol. 14, no. 6, pp. 951–955.

Farrow, A. S. J. (1974). TV scanner to computer in real time, in *Oxford Conference on Computer Scanning*, vol. 2 (P. G. Davey and B. M. Hawes, eds.), Nuclear Physics Laboratory, Oxford, pp. 407–422.

Feldman, J. (1967). First thoughts on grammatical inference, *Artif. Intell. Memo 55*, Computer Science Dept., Stanford University, Stanford, CA.

Feldman, J. (1969). Some decidability results on grammatical inference and complexity, *Artif. Intell. Memo 93*, Computer Science Dept., Stanford University, Stanford, CA.

Feldman, J. A., and Yakomovsky, Y. (1975). Decision theory and artificial intelligence: a semantics-based region analyzer, *Artif. Intell.*, vol. 5, no. 4, pp. 349–372.

Feldman, J., Gips, J., and Reder, S. (1969). Grammatical complexity and inference, *Artif. Intell. Memo 89*, Computer Science Dept., Stanford University, Stanford, CA.

Feng, H. Y. F., and Pavlidis, T. (1975). Decomposition of polygons into simpler components: feature generation for syntactic pattern recognition, *IEEE Trans. Comput.*, vol. C-24, no. 6, pp. 636–650.

Feng, T. Y. (1981). A survey of interconnection network, *IEEE Computer*, Dec., pp. 12–27.

Fenker, R. M., Jr., and Evans, S. H. (1971). A model for optimizing the effectiveness of man-machine decision making in a pattern recognition system, *Tech. Rep.*, Texas Christian University, Institute for the Study of Cognitive Systems, Fort Worth, June 1971, AD-730944.

Ferrari, L. A., Sankar, P. V., Sklansky, J. and Leeman, S. (1986). Efficient two-dimensional filters using B-spline functions, *Comput. Graphics*, Image Process., vol. 35, no. 2, Aug., pp. 152–169.

Findler, N. V. (1979). A family of similarity measures between two strings, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 1, pp. 116–118.

Fink, W. (1976). Image coloration as an interpretation aid, *Proc. SPIE/OSA Conf. Image Process.*, Pacific Grove, CA, vol. 74, pp. 209–215.

Firschein, O., Eppler, W., and Fischler, M. A. (1979). A fast defect measurement algorithm and its array processor mechanization, *Proc. 1979 IEEE Comput. Soc. Conf. Pattern Recognition, Image Process.*, pp. 109–113.

Firschein, O., Rauch, H. E., and Eppler, W. G. (1980). Track assembly and background suppression using an array processor and neighborhood coding, *Proc. Soc. Photo-opt. Instrum Eng.*, vol. 241, pp. 258–266.

Fischler, M. A. (1969). Machine perception and description of pictorial data, *Proc. Joint Int. Conf. Artif. Intell.*, Washington, DC.

Fischler, M. A., and Bolles, R. C. (1986). Perceptual organization and curve partitioning, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 1, Jan., pp. 100–105.

Fisk, C. J. (1972). Imagine a computer program for image enhancement and display of two-dimensional data, *Sandia Laboratories Res. Rep. SC-RR-72-0286*, Albuquerque, N.M., July.

Fitts, J. M. (1980). Automatic target identification of blurred images with superresolution features, *Proc. Soc. Photo-opt. Instrum. Eng.*, vol. 252, pp. 85–91.

Fitzpatrick, J. M., and Leuze, M. R. (1987). A class of one-to-one two-dimensional transformations, *Comput. Vision, Graphics, Image Process.*, vol. 39, no. 3, Sept., pp. 369–382.

Fletcher, L. A., and Kasturi, R. (1988). A robust algorithm for text separation from mixed text/graphics images, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 6, pp. 910–918.

Fogg, D. A. (1984). Contour to rectangular grid conversion using minimum curvature, *Comput. Graphics, Image Process.*, vol. 28, Oct., pp. 85–91.

Foley, D. H. (1971). The probability of error on the design set as a function of the sample size and dimensionality, *Tech. Rep., RACD-TR-71-171*, Griffiss Air Force Base, NY, Dec.

Foley, D. H. (1972). Considerations of sample and feature, size, *IEEE Trans. Inf. Theory*, vol. IT-18, no. 5, pp. 618–626.

Foley, D. H., and Sammon, J. W. (1975). An optimal set of discriminant vectors, *IEEE Trans. Comput.*, vol. C-24, no. 3, pp. 281–289.

Ford, N. L., Batchelor, B. G., and Wilkins, B. R. (1970). Learning scheme for the nearest neighbor classifier, *Inf. Sci.*, vol. 2, p. 139.

Forshaw, M. R. B. (1988). Speeding up the Marr-hildreth edge operator, *Comput. Graphics, Image Process.*, vol. 41, no. 2, Feb., pp. 172–185.

Forst, L. L. (1990). LANELOK: Lane-sensing system provides first step in automated vehicle guidance, *Search*, General Motors Research Laboratories.

Fountain, T. J., Matthews, K. N., and Duff, M. J. B. (1988). The CLIP7A Image Processor, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 3, pp. 310–319.

Freeman, H. (1977). Computer processing of line drawing images; *ACM Comput. Serv.*, vol. 6, pp. 57–97.

Freeman, H. (1978). Shape description via the use of critical points, *Pattern Recognition*, vol. 10, pp. 159–166.

Freeman, H., and Saghri (1978). Generalized chain code for planar curves, *Proc. 4th Int. Joint Conf. Pattern Recognition*, pp. 701–706.

Frei, W., and Chen, C. C. (1977). Fast boundary detection: a generalization and a new algorithm, *IEEE Trans. Comput.*, vol. TC-26, pp. 988–998.

Frieden, B. R. (1972). Restoring with maximum likelihood and maximum entropy, *J. Opt. Soc. Am.*, vol. 62, pp. 511–518.

Frieden, B. R. (1975). Image enhancement and restoration, in *Picture Processing and Digital Filtering* (T. S. Huang, ed.), Springer-Verlag, Berlin, pp. 179–246.

Frieden, B. R., and Burke, J. J. (1972). Restoring with maximum entropy—II. Superresolution of photographs of diffraction blurred images, *J. Opt. Soc. Am.*, vol. 62, pp. 1207–1210.

Frigui, H., and Krishnapuram, R. (1996). A comparison of fuzzy shell clustering methods for the detection of ellipses, *IEEE Trans. Fuzzy Systems*, vol. 4, no. 2, pp. 193–199.

Frigui, H., and Krishnapuran, R. (1997). Clustering by competitive agglomeration, *Pattern Recognition*, vol. 30, no. 7, pp. 1109–1119.

Fu, K. S. (1970). Stochastic automata as models of learning systems, in *Adaptive, Learning, and Pattern Recognition Systems* (J. M. Mendel and K. S. Fu, eds.), Academic Press, New York.

Fu, K. S. (1971). On syntactic pattern recognition and stochastic languages, *Tech. Rep. TR-EE-71-21*, School of Electrical Engineering, Purdue University, Lafayette, IN.

Fu, K. S. (1972). A survey of grammatical inference, *Tech. Rep. TR-EE-72-13*, School of Electrical Engineering, Purdue University, Lafayette, IN.

Fu, K. S. (1978). Recent advances in syntactic pattern recognition, *Proc. 4th Int. Joint Conf. Pattern Recognition*, Kyoto, Japan, Nov. 7–10, 1978.

Fu, K. S. (1979). Size normalization and pattern orientation problems in syntactic clustering, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-9, no. 1, pp. 55–58.

Fu, K. S. (1980). Recent developments in pattern recognition, *IEEE Trans. Comput.*, vol. C-29, no. 10, pp. 845–856.

Fu, K. S., and Bhargava, B. K. (1973). Tree systems for syntactic pattern recognition, *IEEE Trans. Comput.*, vol. C-22, no. 12, pp. 1087–1099.

Fu, K. S., and Swain, P. H. (1971). On syntactic pattern recognition, in *Software Engineering* (J. T. Tou, ed.), Academic Press, New York.

Fu, L. M. (1990). Analysis of the dimensionality of neural networks for pattern recognition, *Pattern Recognition*, vol. 23, no. 10, pp. 1131–1140.

Fukunaga, K. (1981). The optimal distance measure for nearest neighbor classification, *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 622–627.

Fukunaga, K., and Hummels, D. M. (1987). Bayes error estimation using Parsen and k-NN procedures, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, Sept., pp. 634–643.

Fukunaga, K., and Koontz, W. L. C. (1970). Application of the Karhunen-Loeve expansion to feature selection and ordering, *IEEE Trans. Comput.*, vol. C-19, no. 4, pp. 311–318.

Fukunaga, K., and Mantock, J. M. (1982). A non-parametric two-dimensional display for classification, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no. 4, pp. 427–436.

Fukunaga, K., and Olsen, D. R. (1971). An algorithm for finding intrinsic dimensionality of data, *IEEE Trans. Comput.*, vol. C-20, pp. 176–183.

Fukushima, K., and Miyake, S. (1982). Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position, *Pattern Recognition*, vol. 15, no. 6, pp. 455–470.

Furst, M. A., and Caines, P. E. (1986). Edge detection with image enhancement via dynamic programing, *Comput. Graphics, Image Process.*, vol. 33, no. 3, March, pp. 263–279.

Gabozski, R. (1990). An intelligent character recognition system based on neural networks, *Research Magazine*, Kodak Research Laboratory, Spring, pp. 12–13.

Gallant, S. I. (1990). Perceptron based learning algorithms, *IEEE Trans. Neural Networks*, vol. 1, no. 2, pp. 179–191.

Galloway, M. M. (1975). Texture analysis using grey level run lengths, *Comput. Graphics, Image Process.*, vol. 4, no. 2, pp. 172–179.

Garbay, C. (1986). Image structure representation and processing discussion of some segmentation method in cytology, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 2, March, pp. 140–146.

Gershon, R. (1986). Aspects of perception and computation in color vision, *Comput. Graphics, Image Process.*, vol. 32, no. 2, Nov., pp. 244–277, and vol. 33, no. 2, Feb., p. 259.

Ghalli, F. C. (1988). A sequential learning method for boundary detection, *Pattern Recognition*, vol. 21, no. 2, pp. 131–139.

Ghosal. S., and Mehrotra, R. (1997). A moment based unified approach to image feature detection, *IEEE Trans. Image Process.*, vol. 6, no. 6, pp. 781–794.

Ghosh, P. K. (1988). A mathematical model for shape description using Minkowski operators, *Comput. Graphics, Image Process.*, vol. 44, no. 3, Dec., pp. 239–269.

Gilbert, A. L., Giles, M. K., Flacks, G. M., and Rogers, R. B. (1979). A real time video tracking system, *Opt. Eng.*, vol. 18, no. 1, pp. 25–32. Also *Proc. 4th Int. Joint Conf. Pattern Recognition*, Kyoto, Japan, No. 7–10, 1978, p. 111015; *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, no. 1, pp. 47–56 (1980).

Gillenson, M. L., and Chandrasikaran, B. A. (1975). A neuristic strategy for developing images on a CRT, *Pattern Recognition*, vol. 7, no. 4, pp. 187–196.

Gini, G., and Gini, M. (1985). A software laboratory for visual inspection and recognition, *Pattern Recognition*, vol. 18, no. 1, pp. 43–52.

Goin, J. E., and Haberman, J. D. (1983). Automated breast cancer detection by thermography: performance goal and diagnostic feature identification, *Pattern Recognition*, vol. 16, no. 2, pp. 125–130.

Gold, E. M. (1967). Language identification in the limit, *Inf. Control*, vol. 10, no. 5, pp. 447–474.

Goldfarb, L. (1984). A unified approach to pattern recognition, *Pattern Recognition*, vol. 17, no. 5, pp. 575–582.

Gonzalez, R. C. (1972). Syntactic pattern recognition—introduction and survey, *Proc. Nat. Electron. Conf.*, vol. 27, no. 1, pp. 27–32.

Gonzalez, R. C. (1973). Generation of linguistic filter structures for image enhancement, *Proc. ACM Conf.*

Gonzalez, R. C., and Fittes, B. A. (1975). Grey-level transformations for interactive image enhancement, *Mech. Mach. Theory*, vol. 12, pp. 111–112.

Gonzalez, R. C., and Thomason, M. G. (1974a). Tree grammars and their application to pattern recognition, *Tech. Rep. TR-EE-CS-74-10*, Electrical Engineering Dept., University of Tennessee, Knoxville.

Gonzalez, R. C., and Thomason, M. G. (1974b). Inference of tree grammars for syntactic pattern recognition, *Tech. Rep. TR-EE/CS-74-20*, Electrical Engineering Dept., University of Tennessee, Knoxville.

Gonzalez, R. C., and Tou, J. T. (1968). Some results in minimum entropy feature extraction, *IEEE Conv. Rec.*—Region III.

Gonzalez, R. C., and Wagner, C. G. (1982). Moments and semi-invariants of the interclass Mahalanobis distance, *Proc. PRIP '82*, pp. 12–17.

Gonzalez, R. C., Fry, D. N., and Kryter, R. C. (1974). Results in the application of pattern recognition methods to nuclear reactor core component surveillance, *IEEE Trans. Nucl. Sci.*, vol. 21, no. 1, pp. 750–757.

Gonzalez, R. C., Lane, M. C., Bishop, A. O., Jr., and Wilson, W. P. (1972). Some results in automatic sleep-state classification, *Proc. 4th Southeastern Symp. Syst. Theory.*

Goshtasby, A. (1985). Description and discrimination of planar shapes using shape matrices, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, no. 6, Nov., pp. 738–743.

Gotchev, G. V. (1984). Computer linguistic analysis of line drawings, *Pattern Recognition*, vol. 17, no. 4, pp. 433–440.

Gotlieb, C. C., and Kreyszig, H. E. (1990). Texture descriptors based on co-occurrence matrices, *Comput. Vision, Graphics, Image Process.*, vol. 51, no. 1, July, pp. 70–86.

Gowda Chidnaanda, K., and Ravi, T. V. (1995). Divisive clustering of symbolic objects using the concepts of both similarity and dissimilarity, *Pattern Recognition*, vol. 28, no. 8, pp. 1277–1282.

Granlund, G. H. (1978). In search of a general picture processing operator, *Comput. Graphics, Image Process.*, vol. 8, no. 2, pp. 153–173.

Gray, S. B. (1971). Local properties of binary images in two dimensions, *IEEE Trans. Comput.*, vol. C-20, no. 5, pp. 551–561.

Greblicki, W. (1978). Pattern recognition procedures with non-parametric density estimates, *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-8, no. 11, pp. 809–812.

Grimson, W. E. L. (1989). On the recognition of curved objects, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, June, pp. 632–643.

Grossberg, S., and Mingolia, E. (1987). Neural dynamics of surface perception: boundary webs, illiminants, and shape-from-shading, *Comput. Graphics, Image Process.*, vol. 37, no. 1, Jan., pp. 116–165.

Guerra, C., and Pieroni, G. G. (1982). A graph-theoretic method for decomposing two-dimensional polygonal shapes into meaningful parts, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no. 4, pp. 405–408.

Gunther, F. J. (1982). A new principal components procedure to aid the analysis of Landsat MSS digital data, *Proc. PRIP '82*, pp. 38–43.

Gupta, L., and Srinath, M. D. (1988). Invariant planar shape recognition using dynamic alignment, *Pattern Recognition*, vol. 21, no. 2, pp. 235–240.

Gupta, L., Sayeh, M. R., and Tammana, R. (1990). A neural network approach to robust shape classification, *Pattern Recognition*, vol. 23, no. 6, pp. 563–568.

Guyon, I., Makhoul, J., Schwartz, R., and Vapnik, U. (1998). What size test set gives good error rate estimates? *IEEE Trans. Patttern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 52–64.

Guzman, A. (1968). Decomposition of a visual scene into three-dimensional bodies, *Proc. Fall Joint Comput. Conf.*

Haddon, J. F. (1988). Generalized threshold selection for edge detection, *Pattern Recognition*, vol. 21, no. 3, pp. 195–204.

Hahn, S., and Mendoza, E. E. (1984). Simple enhancement techniques in digital image processing, *Comput. Graphics, Image Process.*, vol. 26, no. 2, May, pp. 233–241.

Hall, E. L. (1974a). A comparison of computations for spatial frequency filtering, *Proc. IEEE*, vol. 60, no. 7, pp. 887–891.

Hall, E. L. (1974b). Almost uniform distribution for computer image enhancement, *IEEE Trans. Comput.*, vol. C-23, no. 2, pp. 207–208.

Hall, E. L. Kruger, R. P., Dwyer, S. L., Hall, D. L., McLaren, R. W., and Lodwick, G. S. (1971). A survey of preprocessing and feature extraction techniques for radiographic images, *IEEE Trans. Comput.*, vol. C-20, pp. 1032–1044.

Hall, E. L., Tie, J. B. K., McPherson, C. A., and Sadjadi, F. A. (1982). Curved surface measurement and recognition for robot vision, *Conf. Rec. 1982 Workshop Ind. Appl. Mach. Vision*, Research Triangle Park, N.C., May 3–5, 1982, pp. 187–199.

Hanaizumi, H., Inamura, M., Toyota, H., and Fujmura, (1980). Noise reduction of multispectral scanner image data using scan overlap, *Trans. Soc. Instrum. Control Eng. (Japan)*, vol. 16, no. 6, pp. 880–885.

Hanakata, K. (1974). A descriptive data structure for interactive pattern recognition, *Proc. 2nd Int. Joint Conf. Pattern Recognition*, Copenhagen, Aug. 13–15, 1974, pp. 421–423.

Hand, D. J., and Batchelor, B. G. (1975). Classification of incomplete pattern vectors using orthogonal function methods, *Proc. 3rd Int. Conf. Cybern. Syst.*, Bucharest.

Hannigan, J. F., and Gerhart, G. R. (1980). Direct electronic Fourier transform (DEFT) spectra for surveillance and countersurveillance, *Proc. Soc. Photo-opt. Instrum. Eng.*, vol. 241, pp. 113–121.

Haralick, R. M. (1971). Multi-image clustering, *Proc. 1970 Army Numer. Analy. Conf.*, U.S. Army Research Office, Rep. 71-1, Durham, N.C., Jan. 1971, pp. 75–90.

Haralick, R. M. (1978). Statistical and structural approaches to texture, *Proc. 4th Int. Joint Conf. Pattern Recognition*, Kyoto, Japan, Nov. 7–10, 1978, pp. 45–69.

Haralick, R. M. (1984). Digital step edges from zero crossing of second directional derivatives, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, no. 1, pp. 58–68.

Haralick, R. M., and Kelley, G. L. (1969). Pattern recognition with measurement space and spatial clustering for multiple images, *Proc. IEEE*, vol. 57, no. 4, pp. 654–665.

Haralick, R. M., and Lee, J. S. J. (1990). Context dependent edge detection and evaluation, *Pattern Recognition*, vol. 23, no. 1/2, pp. 1–20.

Haralick, R. M., and Shapiro, L. G. (1979). Decomposition of two-dimensional shapes by graph-theoretic clustering, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAM-1, no. 1, pp. 10–20.

Haralick, R. M., and Shapiro, L. G. (1985). Image Segmentation Techniques, *Comput. Graphics, Image Process.*, vol. 29, no. 1, Jan., pp. 100–132.

Haralick, R. M., Sternberg, S. R., and Zhuang, X. (1987). Image analysis using mathematical morphology, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 4, July, pp. 532–550.

Harms, H., Gunzer, U., and Aus, H. M. (1986). Combined local color and texture analysis of stained cells, *Comput. Graphics, Image Process.*, vol. 33, no. 3, March, pp. 364–476.

Hartley, R. L., Kitchen, L. J., Wang, C. Y., and Rosenfeld, A. (1982). Segmentation of FLIR images: a comparative study, *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-12, no. 4, pp. 553–566.

Hartman, E. J., et al. (1990). Layered neural networks with Gaussian hidden units as universal approximations, *Neural Computation*, vol. 2, no. 2, pp. 210–215.

Hastie T., and Titshirani, R. (1996). Discriminant adaptive nearest neighbor classification, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 6, pp. 607–616.

Hathaway, R. J., and Bezdek, J. C. (1995). Optimization of clustering criteria by reformulation, *IEEE Trans. Fuzzy Systems*, vol. 3, no. 2, pp. 241–245.

Haussmann, G., and Liedtke, C. E. (1984). A region extraction approach to blood smear segmentation, *Comput. Graphics, Image Process.*, vol. 25, no. 2, Feb., pp. 133–150.

Hawkins, J. K. (1970). Image processing principles and techniques, in *Advances in Information System Science* (J. T. Tou, ed.), vol. 3, Plenum Press, New York.

He, D. C., and Wang, L. (1991). Texture features based on texture spectrum, *Pattern Recognition*, vol. 24, no. 5, pp. 391–399.

He, D. C., Wang, L., and Guilbert, J. (1988). Texture discrimination based on an optimal utilization of texture features, *Pattern Recognition*, vol. 21, no. 2, pp. 141–146.

Heintzen, P. H., Malerczyk, H., and Scheel, K. W. (1971). On-line processing of video-images for left ventrical volume determination, *Comput. Biomed. Res.*, vol. 4, no. 5.

Heijmans, H. J. A. M. (1991). Theoretical aspects of gray-level morphology, *IEEE Trans. PAMI*, vol. 13, no. 6, pp. 568–582.

Heller, J. (1978). Image motion restoration, *Tech. Rep.* Dept. of Electrical Engineering, University of Tennessee, Knoxville, May.

Hemminger, T. L., and Pomalaza-Raez, C. A. (1990). Polygonal representation: a maximum likelihood approach, *Comput. Graphics, Image Process.*, vol. 52, no. 2, Nov., pp. 239–247.

Herbst, N. W., and Hill, P. M. (1972). An experimental laboratory for pattern recognition and signal processing, *Comm. ACM*, vol. 15, no. 4, pp. 231–244.

Herman, G. T. (1972). Two direct methods for reconstructing pictures from their projections—a comparative study, *Comput. Graphics, Image Process.*, vol. 1, no. 2, pp. 123–144.

Highleyman, W. H. (1961). Linear decision functions with applications to pattern recognition, Ph.D. dissertation, Electrical Engineering Dept., Polytechnic Institute of Brooklyn, New York. [A summary bearing the same title may be found in *Proc. IRE*, vol. 50, no. 6, pp. 1501–1514 (1962).].

Ho, Y. C., and Agrawala, A. K. (1968). On pattern classification algorithms—introduction and survey, *Proc. IEEE*, vol. 56, no. 12, pp. 2101–2114.

Ho, Y. C., and Kashyap, R. (1965). An algorithm form linear inequalities and its applications, *IEEE Trans. Electron. Comput.*, vol. EC-14, no. 5, pp. 683–688.

Hoff, W., and Ahuja, N. (1989). Surfaces from stereo: integrating feature matching, disparity estimation, and contour detection, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 2, pp. 121–136.

Holdermann, F., and Kazmierczak, H. (1972). Preprocessing of grey-scale pictures, *Comput. Graphics, Image Process.*, no. 1, pp. 66–80.

Holmstrom, L., Koistinen, P., Laaksonen, J., and Oha, E. (1997). Neural and statistical classifiers—taxonomy and two case studies, *IEEE Trans. Neural Networks*, vol. 9, no. 1, pp. 5–17.

Honnenahalli, S. and Bow, S.T. (1988). Piecewise representation of curves with concatenated arcs via detection of the appropriate break points, *Proc. IEEE Inter. Conf. Robotics and Automation.*

Hopfield, J. J., and Tank, D. W. (1980). Computing with neural circuit. A model, *Science* vol. 233, pp. 625–633.

Hopfield, J. J. (1987). Learning algorithms and probability distributions in feed-forward and feed-back networks, *Proc. National Academy of Sciences*, vol. 84, pp. 8429–8433.

Horn, Berthold, K. P., and Brooks, M. J. (1986). The variational approach to shape from shading, *Comput. Graphics*, *Image Process.*, vol. 33, no. 2, Feb., pp. 174–208.

Horning, J. J. (1969). A study of grammatical inference, *Tech. Rep. CS-139*, Computer Science Dept., Stanford University, Stanford, Calif.

Horning, J. J. (1971). A procedure for grammatical inference, *IFIP Congr.—71*, Yugoslavia.

Hotelling, H. (1933). Analysis of complex statistical variables into principal components, *J. Educ. Psychol.*, vol. 24, pp. 417–441, 498–520.

Huang, T. S. (1979). Trends in digital image processing research, in *Advances in Digital Image Processing* (Bad Neuenahr, Germany, Sept. 26–28, 1978), Plenum, New York, 1979, pp. 21–30.

Huang, T. S., Schreiber, W. F., and Tretiak, O. J. (1971). Image processing, *Proc. IEEE*, vol. 59, pp. 1586–1609.

Hueckel, M. H. (1973). A local visual operator which recognizes edges in lines, *J. ACM*, vol. 20, pp. 634–647.

Huertas, A., Cole, W., and Nevatia, R. (1990). Detecting runways in complex airport scenes, *Comput. Graphics*, *Image Process.*, vol. 51, no. 2, Aug., pp. 107–145.

Hummel, R. A., and Rosenfeld, A. (1978). Relaxation processes for scene labeling, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-8, no. 10, pp. 765–768.

Hunt, B. R. (1972). Data structure and computational organization in digital image enhancement, *Proc. IEEE*, vol. 60, no. 7, pp. 884–887.

Hunt, B. R. (1975). Digital image processing, *Proc. IEEE*, vol. 63, no. 4, pp. 693–708.

Hunt, B. R., and Sera, G. F. (1978). Power-law stimulus response models for measures of image quality in non-performance environments, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-8, no. 11, pp. 781–791.

Hush, D. R., and Horne, B. G. (1993). Progress in supervised neural networks, *IEEE Signal Processing Magazine*, vol. 10, no. 1, pp. 8–39.

Husson, T. R., and Abdalla, A. M. (1981). Real-time infrared image processing, *IEEE Comput. Soc. Conf. Pattern Recognition*, *Image Process.*, Dallas, Aug. 3–5, 1981, pp. 478–480.

Hwang, K., and Su, S. P. (1983). VLSI architecture for feature extraction and pattern classification, *Comput. Graphics*, *Image Process.*, vol. 24, no. 2, Nov., pp. 215–228.

Ichino, M. (1981). Nonparametric feature selection method based on local interclass structure, *IEEE Trans. on Syst. Man Cybern.*, vol. SMC-11, no. 4, pp. 289–296.

Illingworth, J., and Kitter, J. (1987). The adaptive Hough transform, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, Sept., pp. 690–698.

Inigo, R. M., and McVey, E. S. (1981). CCD implementation of a three dimensional videotracking algorithm, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-33, no. 2, pp. 230–240.

Inoue, K. (1979). Image processing and pattern recognition in welding engineering, *Syst. Control (Japan)*, vol. 23, no. 7, pp. 370–378 (in Japanese).

Jacks, E. L. (1964). A laboratory for the study of graphical man-machine communication, *AFIPS Conf. Proc.*, vol. 26-1, pp. 343–350.

Jackson, P. H. (1982). Image contrast enhancement by local maximum and local minimum operators, *IBM Tech. Disclosure Bull.*, vol. 24, no. 12, May 1982.

Jacob, R. J. K. (1976). The face as a data display, *Human Factors*, vol. 18, no. 2, pp. 189–200.

Jagadeeson, M. (1970). N-dimensional fast fourier transform, *Proc. 13th Midwest Symp. Circuit Theory*, III, vol. 2, pp. 1–8.

Jain, A. K. (1974). A fast Karhunen-Loeve transform for finite discrete images, *Proc. Nat. Electron. Conf.*, Chicago, Oct. 1974, pp. 323–328.

Jain, A. K., and Angel, E. (1974). Image restoration, modeling and reduction of dimensionality, *IEEE Trans. Comput.*, vol. C-23, pp. 470–476.

Jain, A. K., and Walter, W. G. (1978). On the optımal number of features in the classification of multivariate Gaussian data, *Pattern Recognition*, vol. 10, no. 5–6, pp. 365–374.

Jain, A., and Zongker, D. (1997). Feature selection: evaluation, application, and small sample performance, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 153–158.

Jang, B. K., and Chin, R. T. (1990). Analysis of thinning algorithm using mathematical morphology, *IEEE Trans. PAMI*, vol. 12, no. 6, pp. 541–551.

Jankley, W. J., and Tou, J. T. (1968). Automatic fingerprint interpretation and classificatıon via contextual analysis and topological coding, in *Pictorial Pattern Recognition* (G. C. Cheng et al., eds.), Thompson Books, Washington, DC.

Jardine, N., and Sibson, R. (1968). The construction of hıerarchical and non-hierarchic classification, *Comput. J.*, vol. 11, pp. 177–184.

Jarvis, R. (1974). An interactive minicomputer laboratory for graphics ımage processing and pattern recognition, *Computer*, vol. 7, no. 7, pp. 49–60.

Johansson, E. M., Dowla, F. U., and Goodman, D. M. (1992). Backpropagation learning for multilayer feedforward neural networks using conjugate gradient method, *Inter. J. Neural Systems*, vol. 2, no. 4, pp. 291–301.

Johnson, R. P. (1990). Contrast based edge detection, *Pattern Recognition*, vol. 23, no. 3/4, pp. 311–318.

Jophansson, E. M., Dowla, F. U., and Goodman, D. M. (1992). Backpropagation learning for multilayer feedforward neural networks using the conjugate gradient method, *Inter. J. Neural Systems*, vol. 2, no. 4, pp. 291–301.

Joseph, R. D. (1960). Contributions to perception theory, *Cornell Aeronaut. Lab Rep. VG-1196-G-7*.

Joseph, S. H. (1989). Processing of engineering line drawing for automatic input to CAD, *Pattern Recognition*, vol. 22, no. 1, pp. 1–12.

Kabuka, M., and McVey, E. S. (1982). A position sensing method using images, *Proc. 14th Southeast. Symp. Syst. Theory*, Blacksburg, VA, Apr. 15–16, 1982, pp. 191–194.

Kamgar-Parsi, B., Kamgar-Parsi, B., and Wechsler, H. (1990). Simultaneous fitting of several planes to point sets using neural networks, *Comput. Graphics, Image Process.*, vol. 52, no. 3, Dec., pp. 341–359.

Kanal, L. N., and Kumar, V. (1981). Parallel implementation of structural analysis algorithm. *IEEE Comput. Soc. Conf. Pattern Recognition, Image Process.*, Dallas, Aug. 3–5, 1981, pp. 452–458.

Kanal, L. N., and Randall, N. C. (1964). Recognition system design by statistical analysis, *Proc. 19th ACM Nat. Conf.*

Kankanhalli, M. S, Mehtre, B. M., and Wu, J. K. (1996). Cluster-based color matching for image retrieval, *Pattern Recognition*, vol. 29, no. 4, pp. 701–708.

Karayiannis, N. B., and Pai, P. I. (1996) Fuzzy algorithms for learning vector quantization, *IEEE Trans. Neural Networks*, vol. 7, no. 4, pp. 1196–1211.

Karayiannis, N. B., and Mi, G. W. (1997). Growing radial basis neural networks, merging supervised and unsupervised learning with network growth techniques, *IEEE Trans. Neural Networks*, vol. 8, no. 6, pp. 1492–1506.

Karhunen, K. (1947). Uber lineare Methoden in der Wahrscheinlichkeitsrechnung, *Ann. Acad. Sci. Fenn.*, Ser. A137 (translated by I. Selin on "*On Linear Methods in Probability Theory*," T-131, The RAND Corp., Santa Monica, CA, 1960).

Kashyap, R. L., and Chellappa, R. (1981). Stochastic models for closed boundary analysis: representation and reconstruction, *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 627–637.

Kashyap, R. L., and Eom, K. B. (1989). Texture boundary detection based on the long correlation model, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 1, pp. 58–67.

Kashyap, R. L., and Khotanzad, A. (1986). A model-based method for rotation invariant texture classification, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 4, July, pp. 472–481.

Kashyap, R. L., and Mittal, M. C. (1973). Picture reconstruction from projections, *Proc. First Int. Joint Conf. Pattern Recognition*, Washington, DC, IEEE Cat. 73CHO 821-9C, pp. 286–292.

Kasturi, R. K., Bow, S. T., et al. (1988). A system for recognition and description of graphics, *Proc. 9th Intt. Conf. Pattern Recognition*, Rome, Italy, Nov.

Kasturi, R., Bow, S. T., et al. (1990). A system for interpretation of line drawings, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, PAMI no. 10, pp. 978–992.

Katsinis, C., and Poularikas, A. D. (1987). Analysis of a sampling technique applied to biological images, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 6, Nov., pp. 832–835.

Kazmierczak, H. (1973). Problems in automatic pattern recognition, *Proc. Int. Comput. Symp.*, Davos, Switzerland, pp. 357–370.

Keeha, D. G. (1965). A note on learning for Gaussian properties, *IEEE Trans. Inf. Theory*, vol. IT-II, no. 1, pp. 126–132.

Keller, J. M., Chen, S., and Crownover, R. M. (1989). Texture description and segmentation through fractal geometry, *Comput. Graphics, Image Process.*, vol. 45, no. 2, Feb., pp. 150–166.

Kendall, M. G. (1973). The basic problems of cluster analysis, in *Discriminant Analysis and Applications* (T. Cacoullos, ed.), Academic Press, New York, pp. 179–191.

Ketcham, D. J. (1976). Real time image enhancement technique, *Proc. SPIE/OSA Conf. Image Process.*, Pacific Grove, CA, vol. 74, Feb. 1976, pp. 120–125.

Khotanzad, A., and Chen, J. Y. (1989). Unsupervised segmentation of textured images by edge detection in multidmensional features, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 4, April, pp. 414–420.

Kim, B. S., and Park, S. B. (1986). A fast k nearest neighbor finding algorithm based on the ordered partition, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, Nov., pp. 761–766.

Kirsch, K. A. (1964). Computer interpretation of English text and picture patterns, *IEEE Trans. Electron. Comput.*, vol. EC-13, no. 4, pp. 363–376.

Kitter, J., and Young, P. C. (1973). A new approach to feature selection based on Karhunen-Loeve expansion, *Pattern Recognition*, vol. 5, pp. 335–352.

Kittler, J. (1978). A method for determining k-nearest neighbors, *Kybernetes*, vol. 7, no. 4, pp. 313–315.

Kittler, J., Hatef, M., Duin, R., and Matas, J. (1998). On combining classifiers, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239.

Kobayashi, H., and Bahi, L. R. (1974). Image data compression by predictive coding [2 parts], *IBM J. Res. Dev.*, pp. 164–179.

Koontz, W. L. G., and Fukunaga, K. (1972a). A non-linear feature extraction algorithm using distance transformation, *IEEE Trans. Comput.*, vol. C-21, no. 1, pp. 56–63.

Koontz, W. L. G., and Fukunaga, K. (1972b). A non-parametric valley seeking technique for cluster analysis, *IEEE Trans. Comput.*, vol. C-21, no. 2, pp. 171–178.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, vol. 43, pp 59–69. Reprinted in Anderson & Rosenfeld (1988), pp. 511–521.

Kohonen, T. (1987). Adaptive, associative and self-organizing functions in neural computing, *App. Opt.*, vol. 26, no. 23, pp. 4910–4918.

Koutroumbas, K., and Kalouptsidis, N. (1994). Qualitative analysis of the parallel and asynchronous modes of the Hamming network, *IEEE Trans. Neural Networks*, vol. 5, no. 3, pp. 380–391.

Kovelevsky, V. A. (1970). Pattern recognition, heuristics or science? In *Advances in Information Systems Science* (J. T. Tou, ed.), vol. 3, Plenum Press, New York.

Kovalevsky, V. A. (1978). Recent advances in statistical pattern recognition, *Proc. 4th Int. Joint Conf. Pattern Recognition*, Kyoto, Japan, Nov. 7–10, 1978, pp. 1–12.

Kramer, M. A. (1991). Nonlinear principal component analysis using auto-associative neural networks, *AIC J.*, vol. 37, no. 2, pp. 233–243.

Krishnapuram, R., Frigui, H., and Nasraoui, O. (1995). Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation—Part II, *IEEE Trans. Fuzzy Systems*, vol. 3, no. 1, pp. 44–60.

Kruse, B. (1973). A parallel picture processing machine, *IEEE Trans. Comput.*, vol. C-22, no. 12, pp. 1075–1087.

Kuan, D., Phipps, G., and Hsueh, A. C. (1988). Autonomous robotic vehicle road following, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 5, Sept., pp. 648–658.

Kumar, V. K., and Krishan, V. (1989). Efficient parallel algorithms for image template matching on hypercube SIMD machine, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, June, pp. 665–669.

Kundu, A. (1990). Robust edge detection, *Pattern Recognition*, vol. 23, no. 5, pp. 423–440.

Kundu, A., and Mitra, S. K. (1987). A new algorithm for image edge extraction using a statistical classifier approach, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 4, July, pp. 569–577.

Kung, H. T. (1982). Why systolic architectures?, *IEEE Computer*, vol. 15, no. 1, pp. 37–46.

Kurzynski, M. W. (1988). On the multistage Bayes classifier, *Pattern Recognition*, vol. 21, no, 4, pp. 355–366.

Kushner, T., Wu, A. Y., and Rosenfeld, A. (1982). Image processing on mpp:1, *Pattern Recognition*, vol. 15, no. 3, pp. 121–130.

Laboratory for Agricultural Remote Sensing, Annual Report, vol. 4, Agricultural Experiment Station, *Res. Bull. 873*, Dec. 1970, Purdue University, Lafayette, IN.

Lacroix, V. (1988). A three-module strategy for edge detection, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 5, Nov., pp. 803–810.

Laine, A., and Fan, J. (1993). Texture classification by wavelet packet signatures, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 11, pp. 1186–1191.

Laine, A., and Fan, J. (1996). Frame represent ion for texture segmentation, *IEEE Trans. Image Processing*, vol. 5, no. 5, pp. 771–780.

Lamar, J. V., Stratton, R. H., and Simac, J. J. (1972). Computer techniques for pseudocolor image enhancement, *Proc. First USA–Japan Comput. Conf.*, pp. 316–319.

Landau, H. J., and Slepian, D. (1971). Some computer experiments in picture processing for bandwidth reduction, *Bell Syst. Tech. J.*, vol. 50, pp. 1525–1540.

Landau, U. M. (1987). Estimation of a circular arc center and its radius, *Comput. Graphics, Image Process.*, vol. 38, no. 3, June.

Leboucher, G., and Lowitz, G. E. (1979). What a histogram can tell the classifier, *Pattern Recognition*, vol. 10, no. 5–6, pp. 351–357.

Leclerc, Y. G., and Zucker, S. W. (1987). The local structure of image discontinuities in one dimension, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 3, May, pp. 341–355.

Ledley, R. S. (1964). High speed automatic analysis of biomedical pictures, *Science*, vol. 146, no. 3641, pp. 216–223.

Lee, D. T. (1982). Medial axis transformation of a planar shape, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no. 4, pp. 363–369.

Lee, H. C., and Fu, K. S. (1971). A stochastic syntax analysis procedure and its application to pattern classification, *Proc. Two-Dimensional Digital Signal Process, Conf.*, University of Missouri, Columbia.

Lee, H. C., and Fu, K. S. (1974). A syntactic pattern recognition system with learning capability, in *Information Systems: COINS IV* (J. T. Tou, ed.), Plenum Press, New York.

Lee, J. S. (1980). Digital image enhancement and noise filtering by use of local statistics, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, no. 2, pp. 165–168.

Lee, R. C. T. (1974). Sub-minimal spanning tree approach for large data clustering, *Proc. 2nd Int. Joint Conf. Pattern Recognition*, Copenhagen, p. 22.

Lee, R. C. T. (1981). Clustering analysis and its applications, in *Advances in Information Systems Science* (J. T. Tou, ed.), vol. 8, Plenum Press, New York, pp. 169–292.

Lee, S. Y., and Aggarwal, J. K. (1987). Parallel 2-D convolution on a mesh connected array processor, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 4, July, pp. 590–594.

Leese, J. A., Novak, C. S., and Taylor, V. R. (1970). The determination of cloud pattern motions from geosynchronous satellite image data, *Pattern Recognition*, vol. 2, pp. 279–292.

Lendaris, G. G., and Stanley, G. L. (1970). Diffraction pattern sampling for automatic pattern recognition, *Proc. IEEE*, vol. 58, pp. 198–216.

Leu, J. G., and Wee, W. G. (1985). Detecting the spatial structure of natural textures based on shape analysis, *Comput. Graphics, Image Process.*, vol. 31, no. 1, July, pp. 67–88.

Levialdi, S. (1968). CLOPAN: a closed-pattern analyzer, *Proc. IEEE*, vol. 115, pp. 879–880.

Levialdi, S. (1970). Parallel counting of bınary patterns, *Electron. Lett.*, vol. 6, pp. 798–800.

Li, C. C., Ameling, W., DeMori, R., Fu, K. S., Harlow, C. A., Hüting, M. K., Pavlidis, T., Pöppl, S. J., Van Bemmel, E. H., and Wood, E. H. (1979). *Cardio-Pulmonary Systems Group Report*, Dahlem Workshop Report on "Biomedical Pattern Recognition and Image Processing," held in Berlin in May 1979, pp. 299–330.

Li, H. F., Pao, D., and Jayakumar, R. (1989). Improvements and systolic implementation of the Hough transformation for straight line detection, *Pattern Recognition*, vol. 22, no. 6, pp. 697–706.

Li, H. W., Lavin, M. A., and Le Master, R. J. (1986). Fast Hough Transform: a hierarchical approach, *Comput. Graphics, Image Process.*, vol. 36, no. 2/3, Nov., pp. 139–161.

Li, W., and He, D. C. (1990). Texture classification using texture spectrum, *Pattern Recognition*, vol. 23, no. 8, pp. 905–910.

Licklider, J. C. R. (1969). A picture is worth a thousand words and its costs, *AFIPS Conf. Proc.*, vol. 34, pp. 617–622.

Lillestrand, R. L. K. (1972). Techniques for change detection, *IEEE Trans. Comput.*, vol. C-21, no. 7, pp. 654–659.

Lin, C. C., and Chellappa, R. (1987). Classification of partial 2-D shapes using Fourier descriptors, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, Sept., pp. 686–690.

Lin, Y. K., and Fu, K. S. (1983). Automatic classification of cervical cells using a binary tree classifier, *Pattern Recognition*, vol. 16, no. 1, pp. 69–80.

Lippman, R. P. (1987). An introduction to computing with neural nets, *IEEE ASSP Magazine*, April, pp. 4–22.

Liou, S. P., and Jain, R. C. (1987). Road following using vanishing points, *Comput. Graphics, Image Process.*, vol. 39, no, 1, July, pp. 116–130.

Lo, C. M. (1980). A survey of FLIR ımage enhancement, *Proc. 5th Int. Conf. Pattern Recognition*, Miami Beach, FL., Dec. 1–4., 1980 (IEEE, New York, 1980), pp. 920–924.

Loizou, G., and Maybank, S. J. (1987). The nearest neighbor and the bayes error rates, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 2, March, pp. 254–262.

Lowe, D. (1995). Radial basis function networks, in *The Handbook of Brain Theory and Neural Networks* (M. A. Arbib, ed.), MIT Press, Cambridge, Mass.

Lu, S. Y. (1979). A tree-to-tree distance and its application to cluster analysis, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 219–224.

Lu, S. Y., and Fu, K. S. (1978). Error-correcting tree automata for syntactic pattern recognition, *IEEE Trans. Comput.*, vol. C-27, no. 11, pp. 1040–1053.

Lu, Y., and Jain, R. C. (1989). Behavior of edges in scale space, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no, 4, April, pp. 337–356.

Lucas, B. T., and Gardner, K. L. (1980). A generalized classification technique, *Proc. 5th Int. Conf. Pattern Recognition*, pp. 647–653.

Lumelsky, V. J. (1982). A combined algorithm for weighting the variables and clustering in the clustering problem, *Pattern Recognition*, vol. 15, no. 2, pp. 53–60.

Lumia, R., Haralick, R. M., Zumiga, O., Shapiro, L., Pong, T. C., and Wang, F. P. (1983). Texture analysis of aerial photographs, *Pattern Recognition*, vol. 16, no. 1, pp. 39–46.

Lundsteen, C., Gerdes, T., and Phillip, K. (1982). A model for selection of attributes for automatic pattern recognition—stepwise data compression monitored by visual classification, *Pattern Recognition*, vol. 15, no. 3, pp. 243–251.

Lutz, R. K. (1980). An algorithm for the real-time analysis of digitized images, *Comput. J.*, vol. 23, no. 3, pp. 262–269.

Ma, J., Wu, C. K., and Lu, X. R. (1986). A fast shape descriptor, *Comput. Graphics, Image Process.*, vol. 34, no. 3, June, pp. 282–291.

Magee, M. J., and Aggarwal, J. K. (1984). Determining vanishing points from perspective images, *Comput. Graphics, Image Process.*, vol. 26, no. 2, May, pp. 256–267.

Mallat, S. (1989a). Multifrequency channel decompositions of images and wavelet models, *IEEE Trans. Acoustics, Speech, Signal Process.*, vol. 37, no. 12, pp. 2091–2110.

Mallat, S. G. (1989b). A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Trans. PAMI*, vol. 11, no. 7, pp. 674–693.

Man, Y., and Gath, I. (1994). Detection and separation of ring-shaped clusters using fuzzy clustering, *IEEE Trans. PAMI*, vol. 16, no. 8, pp. 855–861.

Mansbach, P. (1986). Calibration of a camera and light source by fitting to a physical model, *Comput. Graphics, Image Process.*, vol. 35, no. 2, Aug., pp. 200–219.

Mansouri, A. R., Malowwany, A. S., and Levine, M. D. (1987). Line detection in digital pictures: a hypothesis prediction/verification paradigm, *Comput. Graphics, Image Process.*, vol. 40, no. 1, Oct., pp. 95–114.

Mao, J., and Jain, A. K. (1997). Artificial neural networks for feature extraction and multivariate data projection, *IEEE Trans. Neural Networks*, vol. 6, no. 2, pp. 296–317.

Martinez-Perez, M. P., Jimenez, J., and Navalon, J. L. (1987). A thinning algorithm based on contours, *Comput. Graphics, Image Process.*, vol. 39, no. 2, Aug., pp. 186–201.

Matsuyama and Phillips (1984). Digital realization of the labeled Voronoi diagram and its application to closed boundary detection, *Proc. 7th Int. Conf. Pattern Recognition, Image Process.*, pp. 478–480.

Mazzola, S. (1990). A K-nearest neighbor-based method for the restoration of damaged images, *Pattern Recognition*, vol. 23, no. 1/2, pp. 179–184.

McCormick, B. H. (1963). The Illinois pattern recognition computer—ILLIAC III, *Trans. IEEE Electron. Comput.*, vol. EC-12, pp. 792–813.

McKenzie, D. S., and Protheroe, S. R. (1990). Curve description usıng the inverse Hough transform, *Pattern Recognition*, vol. 23, no. 3/4, pp. 283–290.

McMurtry, G. J. (1976). Preprocessing and feature selection in pattern recognition with application to remote sensing and multispectral scanner data, *IEEE 1976 Int. Conf. Cybern.*, Washington, DC, Nov. 1–3.

Merill, T., and Green, D. M. (1963). On the effectiveness of receptors in recognition systems, *IEEE Trans. Inf. Theory*, vol. IT-9, no. 1, pp. 11–27.

Mero, L. (1980). Edge extraction and line following using parallel processing, *Proc. Workshop Picture Data Descr. Manag.*, IEEE, New York, 1980, pp. 255–258.

Meyer, F. (1986). Automatic screening of cytological specimens, *Comput. Graphics, Image Process.*, vol. 35, no. 3, Sept., pp. 356–369.

Miller, W. F., and Linggard, R. (1982). A perceptually based spectral distance measure, *1982 Int. Zurich. Semin. Digital Commun. Mann-Mach. Interact.*, Zurich, Mar. 9–11, 1982, E4/143–146.

Miller, W. F., and Shaw, A. C. (1968). Linguistic methods in picture processing—a survey, *Proc. Fall Joint Comput. Conf.*

Minsky, M. L. (1961). Steps toward artificial intelligence, *Proc. IRE*, vol. 49, no. 1, pp. 8–30.

Minter, T. C., Lennington, R. K., and Chittineni, C. B. (1981). Probabilistic cluster labeling of imagery data, *IEEE Comput. Soc. Conf. Pattern Recognition Image Process.*, 1981.

Mizoguchi, R., and Kakusho, O. (1978). Hierarchical clustering algorithm based on k-nearest neighbors, *4th Int. Joint Conf. on Pattern Recognition*, Kyto, Japan, pp. 314–316.

Mori, R. I., and Raeside, D. E. (1981). A reappraisal of distance-weighted k-nearest neighbor classification for pattern recognition with missing data, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, no. 3, pp. 241–243.

Moss, R. H., Robinson, C. M., and Poppelbaum, W. J. (1983). On-line recognition (OLREC): a novel approach the visual pattern recognition, *Pattern Recognition*, vol. 16, no. 6, pp. 535–550.

Mott-Smith, J. C., Cook, F. H., and Knight, J. M. (1972). Computer aided evaluation of reconnaissance imaging compression schemes using an on-line interactive facility, *Phys. Sci. Res. Paper 480*, AFCRL-72-0115, Feb. 1972.

Mucciardi, A. N., and Gose, E. E. (1972a). An automatic clustering algorithm and its properties in high dimensional spaces, *Trans. IEEE Syst. Man. Cybern.*, vol. SMC-2, p. 247.

Mucciardi, A. N., and Gose, E. E. (1972b). Comparison of seven techniques for choosing subsets of pattern recognition properties, *Trans. IEEE Comput.*, vol. C-20, p. 1023.

Mulgrew, B. (1996). Applying radial basis functions, *IEEE Signal Process.*, vol. 13, no. 2, pp. 50–65.

Mullin, J. K. (1982). Interfacing criteria for recognition logic used with a context postprocessor, *Pattern Recognition*, vol. 15, no. 3, pp. 271–273.

Murray, G. G. (1972). Modified transforms in imagery analysis, *Proc. 1972 Symp. Appl. Walsh Functions*, pp. 235–239.

Murthy, M. J., and Jain, A. K. (1995). Knowledge-based clustering scheme for collection, management and retrieval of library books, *Pattern Recognition*, vol. 28, no. 7, pp. 949–963.

Musavi, M. T., Shirvaikar, M. V., Ramanathan, E., and Nekovei, A. R. (1988). A vision based method to automate map processing, *Pattern Recognition*, vol. 21, no. 4, pp. 319–326.

Nadler, M. (1984). Document segmentation and coding techniques, *Comput. Graphics, Image Process.*, vol. 28, no. 2, Nov., pp. 240–262.

Nadler, M. (1990). A note on the coefficients of compass mask coeficients, *Comput. Vision Graphics, Image Process.*, vol. 51, pp. 96–101.

Nagasamy, V., and Langrana, N. A. (1990). Engineering drawing processing and vectorization system, *Comput. Graphics, Image Process.*, vol. 49, no. 3, March, pp. 379–397.

Nagao, S. M., and Fukunaga, Y. (1974). An interactive picture processing system on a microcomputer, *Proc. 2nd Int. Conf. Pattern Recognition*, Copenhagen, Aug. 13–15, 1974, pp. 148–149.

Nagata, T., and Zha, H. B. (1988). Determining orientation, location and size of primitive surfaces by a modified Hough transformation technique, *Pattern Recognition*, vol. 21, no. 5, pp. 481–492.

Nagy, G. (1968). State of the art in pattern recognition, *Proc. IEEE*, vol. 56, no. 5, pp. 836–862.

Nahi, N. E. (1972). Role of recursive estimation in statistical image enhancement, *Proc. IEEE*, vol. 60, pp. 872–877.

Nalwa, V. S. (1987). Edge-detector resolution improvement by image interpolation, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 3, May, pp. 446–451.

Nalwa, V. S. (1988). Line-drawing interpretation: Straight lines and conic sections, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 4, July, pp. 514–529.

Nalwa, V. S., and Binford, T. O. (1986). On detecting edges, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, Nov., pp. 699–714.

Nalwa, V. S., and Pauchon, E. (1987). Edgel-aggregation and edge-description, *Comput. Graphics, Image Process.*, vol. 40, no. 1, Oct., pp. 79–94.

Narasimhan, R. (1962). A linguistic approach to pattern recognition, *Rep. 21*, Digital Computer Laboratory, University of Illinois, Urbana.

Narayanan, K. A., and Rosenfeld, A. (1961). Image smoothing by local use of global information, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, no. 12, pp. 826–831.

Navarro, A. (1976). The role of the associative processor in pattern recognition, *Proc. NATO Advanced Studies Inst.*, Bandol, France, Sept. 1975.

Nedeljkovic, V. (1993). A novel multilayer neural network training algorithm that minimizes the probability of classification error, *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 650–659.

Nene, S. A., and Nayar, S. K. (1997). A simple algorithm for nearest neighbor search in high dimensions, *IEEE Trans. Pattern Analysis Mach. Intell.*, vol. 19, no. 9, pp. 989–1003.

Nitzan, D., and Agin, G. J. (1979). Fast methods for finding object outlines, *Comput. Graphics, Image Process.*, vol. 9, no. 1, pp. 22–39.

Nix, A. D, and Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution, *Proc, IEEE Inter. Conf. Neural Networks*, vol. 1, pp. 55–60.

O'Gorman, L. (1990). k × k thinning, *Comput. Graphics, Image Process.*, vol. 51, no. 2, Aug., pp. 195–215.

O'Handley, D. A., and Green, W. B. (1972). Recent development in digital image processing at the image processing laboratory at the Jet Propulsion Laboratory, *Proc. IEEE*, vol. 60, no. 7, pp. 821–828.

Ojala, T., Pietikainen, M., and Harwood, D. (1996). comparative study of texture measures with classification based on feature dsitributions, *Pattern Recognition*, vol. 29, no. 1, pp. 51–59.

Okawa, Y. (1984). Automatic inspection of the surface defects of cast metals, *Comput. Graphics, Image Process.*, vol. 25, no. 1, Jan., pp. 89–112.

Okazaki, A., Kondo, T., Mori, K., Tsunekawa, S., and Kawamoto, E. (1988). An automatic circuit diagram reader with loop-structure-based symbol recognition, *IEEE Transon. PAMI*, vol. 10, no. 3, May, pp. 331–341.

Osteen, R. E., and Tou, J. T. (1973). A clique detection algorithm based on neighborhoods in graphs, *Int. J. Comput. Inf. Sci.*, vol. 2, no. 4, pp. 257–268.

Otsu, N. (1979). A threshold selection method from grey level histograms, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-9, no. 1, pp. 62–66.

Ozbourn, G. C., and Martinez, R. F. (1995). Empirically defined regions of influence for cluster analysis, *Pattern Recognition*, vol. 28, no. 11, pp. 1793–1806.

Pal, A., and Pal, S. K. (1990). Generalized guard zone algorithm (GGA) for learning: automatic selection of threshold, *Pattern Recognition*, vol. 23, no 3/4, pp. 325–335.

Pal, S. K. (1982). Optimum guard zone for self supervised learning, *Proc. IEEE*, vol. 129, no. 1, pp. 9–14.

Panda, D., Aggarwal, R., and Hummel, R. (1980). Smart sensors for terminal homing, *Proc. Soc. Photo-opt. Instrum. Eng.*, vol. 252, pp. 94–97.

Pao, T. W. (1969). A solution of the syntactic induction inference problem for a non-trivial subset of context free languages, *Interim Tech. Rep. 78-19*, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia.

Pao, Y. H. (1978). An associate memory technique for the recognition of patterns, *4th Int. Joint Conf. Pattern Recognition*, Kyoto, Japan, Nov. 7–10, 1978, pp. 405–407.

Pao, Y. H. (1981). A rule-based approach to electric power systems security assessment, *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, Dallas, TX, Aug. 1981, pp. 1-3.

Parikh, J. A., and Rosenfeld, A. (1978). Automatic segmentation and classification of infrared meteorological satellite data, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-8, no. 10, pp. 736–743.

Park, J., and Sandberg, I. W. (1991). Universal approximation using radial basis function networks, *Neural Computation*, vol. 3, no. 2, pp. 246–257.

Patrick, E. A., and Shen, L. Y. L. (1971). Interactive use of problem knowledge for clustering and decision making, *IEEE Trans. Comput.*, vol. C-20, no. 2, pp. 216–222.

Patterson, J. D., Wagner, T. J., and Womack, B. F. (1967). A mean square performance criterion for adaptive pattern classification, *IEEE Trans. Autom. Control*, vol. 12, no. 2, pp. 195–197.

Pavel, M. (1979). Skeletons in pattern recognition categories, *Proc. IEEE Comput. Soc. Conf. Pattern Recognition, Image Process.*, p. 406.

Pavel, M. (1983). Shape theory and pattern recognition, *Pattern Recognition*, vol. 16, no. 3, pp. 349–356.

Pavlidis, T. (1978). Comments on "A new shape factor," *Comput. Graphics, Image Process.*, vol. 8, no. 2, pp. 310–312.

Pavlidis, T. (1981). A flexible parallel thinning algorithm, *IEEE Comput. Soc. Conf. Pattern Recognition, Image Process.*, Dallas, pp. 162–167.

Pavlidis, T. (1986). A vector and feature extractor for document recognition, *Comput. Graphics, Image Process.*, vol. 35, no. 1, July, pp. 111–127.

Pavlidis, T., and Ali, F. (1979). A hierarchical syntactic shape analyzer, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 1, pp. 2–9.

Pavlidis, T., and Horowitz, S. L. (1974). Segmentation of plane curves, *IEEE Trans. Comput.*, vol. C-23, no. 8, pp. 860–870.

Perantonis, S. J., and Lisboa, P. J. G. (1992). Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers, *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 241–251.

Perez, A., and Gonzalez, R. C. (1987). An iterative thresholding algorithm for image segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 6, Nov., pp. 742–751.

Perkins, W. A. (1982). A learning system that is useful for industrial inspection tasks, *Conf. Rec. 1982 Workshop Ind. Appl. Mach. Vision*, Research Triangle Park, N.C., May 1982, pp. 160–167.

Perruchet, C. (1983). Constrained agglomerative hierarchical classification, *Pattern Recognition*, vol. 16, no. 2, pp. 213–218.

Persoon, E., and Fu, K. S. (1977). Shape discrimination using Fourier descriptions, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-7, pp. 170–179.

Persoon, E., and Fu, K. S. (1986). Shape discrimination using Fourier descriptors, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 8, pp. 388–397.

Peters, F. J. (1986). An algorithm for transformations of pictures represented by quadtrees, *Comput. Graphics, Image Process.*, vol. 36, no. 2/3, Nov./Dec., pp. 397–403.

Pfaltz, J. L., and Rosenfeld, A. (1969). Web grammars, *Proc. Joint Int. Conf. Artif. Intell.*, Washington, DC.

Pietikainen, M., Rosenfeld, A., and Walter, I. (1982). Split-and-link algorithms for image segmentation, *Pattern Recognition*, vol. 15, no. 4, pp. 287–298.

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowwitz, A., Greer, T., Romeny, B. T. H., Zimmerman, J. B., and Zuiderveld, K. (1987). Adaptive histogram equalization and its variation, *Comput. Graphics, Image Process.*, vol. 39, no. 3, Sept., pp. 355–368.

Plott, H., Jr., Irwin, J., and Pinson, L. (1975). A real-time stereoscopic small computer graphic display system, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-5, pp. 527–533.

Pollard, J. M. (1971). The fast Fourier transform in a finite field, *Math. Comput.*, vol. 25, no. 114, pp. 365–374.

Poppelbaum, W. J., Faiman, M, Casasent, D., and Sabd, D. S. (1968). On-line Fourier transform of video images, *Proc. IEEE*, vol. 56, no. 10, pp. 1744–1746.

Postaire, J. G. (1982). An unsupervised bayes classifier for normal patterns based on marginal densities analysis, *Pattern Recognition*, vol. 15, no. 2, pp. 103–112.

Postaire, J. G., and Vasseur, C. P. A. (1981). An approximate solution to normal mixture identification with application to unsupervised pattern classification, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, no. 2, pp. 163–179.

Postaire, J. G., Zhang, R. D., and Lecocq-Botte, C. (1993). Cluster analysis by binary morphology, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 2, pp. 170–180.

Pratt, W. K. (1977). Pseudoinverse image restoration computational algorithms, in *Optical Information Processing* (G. W. Stroke, Y. Nestenkhin, and E. S. Barrekette, eds.), vol. 2, Plenum Press, New York, pp. 317–328.

Pratt, W. K., and Davarian, F. (1977). Fast computational techniques for pseudo inverse and Wiener image restoration, *IEEE Trans. Comput.*, vol. C-26, no. 6, pp. 571–580.

Pratt, W. K., and Kruger, R. P. (1972). Image processing over the ARPA computer network, *Proc. Int. Telemetermg Conf.*, vol. 8, Los Angeles, Oct. 10–12, 1972, pp. 344–352.

Preston, K., Jr. (1971). Feature extraction by Goley hexagonal pattern transforms, *IEEE Trans. Comput.*, vol. C-20, pp. 1007–1014.

Preston, K. Jr. (1972). A comparison of analog and digital techniques for pattern recognition, *Proc. IEEE*, vol. 60, pp. 1216–1231.

Prewitt, J. M. S. (1970). Object enhancement and extraction, in *Picture Processing and Psychopictorics* (B. S. Lipkin and A. Rosenfield, eds.), Academic Press, New York, pp. 75–149.

Price, C., Snyder, W., and Rajala, S. (1981). Computer tracking of moving objects using a Fourier domain filter based on a model of the human visual system, *IEEE Comput. Soc. Conf. Pattern Recognition, Image Process.*, Dallas, Aug. 3–5, 1981.

Price, K. E. (1976). Change detection and analysis of multispectral images, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

Princen, J., Illingworth, J., and Kittler, J. (1990). A hierarchical approach to line extraction based on the Hough transform, *Comput. Graphics, Image Process.*, vol. 52, no. 1, Oct., pp. 57–77.

Rahman, M. M., Jacquot, R. G., Quincy, E. A., and Stewart, E. A. (1980). Pattern recognition techniques in cloud research: II. Application, *Proc. 5th Int. Conf. Pattern Recognition*, Miami Beach, FL., Dec. 1–4, 1980 (IEEE, New York, 1980), pp. 470–474.

Ranade, S., Rosenfeld, A., and Sammet, H. (1982). Shape approximation using quadtrees, *Pattern Recognition*, vol. 15, no. 1, pp. 31–40.

Rauch, H. E., and Firschein, O. (1980). Automatic track assembly for threshold infrared images, *Proc. Soc. Photo-opt. Instrum. Eng.*, vol. 253, pp. 75–85.

Ravichandran, G. (1995). Circular-Mellin features for texture segmentation, *IEEE Trans. Image Process.*, vol. 2, no. 12, pp. 1629–1641.

Ready, P. J., and Wintz, P. A. (1973). Information extraction, SNR improvement and data compression in multispectral imagery, *IEEE Trans. Commun.*, vol. COM-21, no. 10.

Reed, S. K. (1972). Pattern recognition and categorization, *Cognit. Psychol.*, vol. 3. pp. 382–407.

Reed, T. R., and Wechsler, H. (1990). Segmentation of textural images and gestalt organization using spatial/spatial-frequency representations, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-12, no. 1, Jan., pp. 1–12.

Reeves, A. P., and Rostampour, A. (1982). Computational cost of image registration with a parallel binary array processor, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no. 4, pp. 449–455.

Richard, M. D., and Lippmann, R. P. (1991). Neural network classifiers estimate bayesian a posteriori probabilities, *Neural Computation*, vol. 3, no. 4, pp. 461–483.

Richardson, W. (1995). Applymg wavelets to mammograms, *IEEE Eng. Med. Biol.*, vol. 14, pp. 551–560.

Ridfway, W. C. (1962). An adaptive logic system with generalizing properties, *Stanford Electron Lab. Tech. Rep. 1556-1*, Stanford University, Stanford, CA.

Riseman, E. A., and Arbib, M. A. (1977). Computational techniques in visual systems: Part II: Segmenting static scenes, *IEEE Comput. Soc. Repository R77-87.*

Roach, J. W., and Aggarwal, J. K. (1979). Computer tracking of objects moving in space, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 127–135.

Robbins, G. M. (1970). Image restoration for a class of linear spatially variant degradations, *Pattern Recognition*, vol. 2, no. 2, pp. 91–105.

Robbins, H., and Monro, S. (1951). A stochastic approximation method, *Ann. Math. Stat.*, vol. 22, pp. 400–407.

Robinson, G. S., and Frei, W. (1975). Final research report on computer processing of ERTS images *USC-IPI Rep. 640*, Image Processing Institute, University of Southern California, Los Angeles.

Rogers, D., and Tanimoto, T. (1960). A computer program for classifying plants, *Science*, vol. 132, pp. 1115–1118.

Rosenblatt, F. (1957). The perceptron: A perceiving and recognizing automation, Project PARA, *Cornell Aeronaut. Lab. Rep. 85-460-1.*

Rosenblatt, F. (1960). On the convergence of reinforcement procedures in simple perceptrons, *Cornell Aeronaut. Lab. Rep. VG-1196-G4.*

Rosenfeld, A. (1969). Picture processing by computer, *Comput. Surv.*, vol. 1, no. 3, pp. 147–176.

Rosenfeld, A. (1974). Compact figures in digital pictures, *IEEE Trans. Syst. Man Cybern.*, vol. 4, pp. 211–213.

Rosenfeld, A. (1978a). Clusters in digital pictures, *Inf. Control*, vol. 39, no. 1, pp. 19–34.

Rosenfeld, A. (1978b). Relaxation methods in image processing and analysis, *Proc. 4th Int. Joint Conf. Pattern Recognition*, Kyoto, Japan, Nov. 7–10, 1978, pp. 181–185.

Rosenfeld, A. (1982). Picture processing: 1981, *Comput. Graphics Image Process.*, vol. 19, no. 1, pp. 35–75, May 1982.

Rosenfeld, A. (1983). On connectivity properties of grayscale pictures, *Pattern Recognition*, vol. 16, no. 1, pp. 47–50.

Rosenfeld, A. (1986). Axial representation of shape, *Comput. Graphic, Image Process.*, vol. 33, no. 2, Feb., pp. 156–173.

Rosenfeld, A., and Pfaltz, J. L. (1968). Distance functions on digital pictures, *Pattern Recognition*, vol. 1, pp. 33–61.

Rosenfeld, A., Fried, C., and Orton, J. N. (1965). Automatic cloud ınterpretation, *Photogramm. Eng.*, vol. 31, pp. 991–1002.

Rubin, L. M., and Frei, R. L. (1979). New approach to forward looking infrared (FLIR) segmentation, *Proc. Soc. Photo-opt. Instrum. Eng.*, vol. 205, pp. 117–125.

Ruell, H. E. (1982). Pattern recognition in data entry, *1982 Int. Zurich Semin. Digital Commun. Man-Mach. Interact.*, Zurich, Mar. 9–11, 1982 (IEEE, New York, 1982), pp. E2/129–136.

Salari, E., and Siy, P. (1982). A grey scale thinning algorithm using contextual information, *First Annu. Phoenix Conf. Comput. Commun.*, Phoenix, AZ, May 9–12, 1982, pp. 36–38.

Samet, H. (1981). An algorithm for converting image into quadtree, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, no. 1, Jan., pp. 93–95.

Samet, H. (1984). Algorithms for the conversion of quadtrees to rasters, *Comput. Graphics, Image Process.*, vol. 26, no. 1, April, pp. 1–16.

Samet, H., Rosenfeld, A., Shaffer, C. A., and Webber, R. E. (1984). A geographic information system using quadtrees, *Pattern Recognition*, vol. 17, no. 6, pp. 647–656.

Sammon, J. W., Jr., Connell, D. B., and Opitz, B. K. (1971). Program for on-line pattern analysis, *Tech. Rep. TR-177* (2 vols.), Tome Air Development Center, Rome, Sept. 1971, AD-732235 and AD-732236.

Sanderson, A. C., and Segen, J. (1980). A pattern-directed approach to signal analysis, *Proc. 5th Int. Conf. Pattern Recognition*, Miami Beach, FL., Dec. 1–4, 1980 (IEEE, New York, 1980).

Sankar, P. V., and Ferrari, L. A. (1988). Simple algorithms and architectures for B-spline interpolation, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-10, no. 2, March, pp. 271–276.

Saridis, G. N. (1980). Pattern recognition and image processing theories, *Proc. 5th Int. Conf. Pattern Recognition*, Miami Beach, FL., Dec. 1–4, 1980 (IEEE, New York, 1980).

Saund, E. (1989). Dimensionality-reduction using connectionist networks, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-11, no. 3, pp. 304–314.

Sawchuk, A. A. (1972). Space-variant image motion degradation and restoration, *Proc. IEEE*, vol. 60, pp. 854–861.

Scaltock, J. (1982). A survey of the literature of cluster analysis, *Comput. J.*, vol. 25, no. 1, pp. 130–133.

Schachter, B. (1978). A non-linear mapping algorithm for large data set, *Comput. Graphics, Image Process.*, vol. 8, no. 2, pp. 271–276.

Schell, R. R., Kodres, U. R., Amir, H., and Tao, T. F. (1980). Processing of infrared images by multiple microcomputer system, *Proc. Soc. Photo-opt. Instrum. Eng.*, vol. 241, pp. 267–278.

Scher, A., Shneier, M., and Rosenfeld, A. (1982). Clustering of collinear line segments, *Pattern Recognition*, vol. 15, no. 2, pp. 85–92.

Schreiber, W. F. (1978). Image processing for quality improvement, *Proc. IEEE*, vol. 66, no. 12, pp. 1640–1651.

Sclove, S. L. (1981). Pattern recognition in image processing using interpixel correlation, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, no. 2, pp. 206–208.

Selfridge, P. G. (1986). Locating neuron boundaries in electron micrograph images usıng "Primal Sketch" primitives, *Comput. Graphics, Image Process.*, vol. 34, no. 2, May, pp. 138–165.

Selim, S. Z., and Ismail, M. A. (1984). K-means-type algorithms: a generalized convergence theorem and characterization of local optimality, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, no. 1, pp. 81–87.

Sethi, I. K. (1981). A fast algorithm for recognizing nearest neighbors, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, no. 3, pp. 245–248.

Setiono, R., and Liu, H. (1997). Neural network feature selector, *IEEE Trans. Neural Networks*, vol. 8, no. 3, pp. 654–662.

Shahraray, B., and Anderson, D. J. (1985). Uniform resampling of digitized contours, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, no. 6, Nov., 1985, pp. 674–681.

Shanmugar, K. S., and Paul, C. (1982). A fast edge thinning operator, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-12, no. 4, pp. 567–569.

Shapiro, L. (1988). Processor array, *Comput. Graphics, Image Process.*, vol. 41, no. 3, March, pp. 382–383.

Shapiro, S. D. (1978). Properties of transforms for the detection of curves in noisy pictures, *Comput. Graphics, Image Process.*, vol. 8, no. 2, pp. 219–236.

Shih, F. Y. C., and Mitchell, O. R. (1991). Decomposition of gray-scale morphological structuring elements, *Pattern Recognition*, vol. 24, no. 3, pp. 195–203.

Shirai, Y., and Tsuji, S. (1972). Extraction of the line drawing of three dimensional objects by sequential illumination from several directions, *Pattern Recognition*, vol. 4, pp. 343–351.

Shneier, M. O., Lumia, R., and Kent, E. W. (1986). Model-based strategies for high-level robot vision, *Comput. Graphics, Image Process.*, vol. 33, no. 3, March, pp. 293–306.

Short, R. D., and Fukunaga, K. (1982). Feature extraction using problem localization, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no. 3, pp. 323–326.

Shu, J. S. (1989). One-pixel wide edge detection, *Pattern Recognition*, vol. 22, no. 6, pp. 665–674.

Siew, L. H., Hodgson, R. M., and Wood, E. J. (1988). Texture measures for carpet wear assessment, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 1, pp. 92–105.

Silberberg, T., Peleg, S., and Rosenfeld, A. (1981). Multiresolution pixel linking for image smoothing and segmentation, *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 281, pp. 217–223.

Silverman, J. F., and Cooper, D. B. (1988). Bayesian clustering for unsupervised estimation of surface and texture models, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 4, July, pp. 482–496.

Simon, J. C. (1978). Some current topics ın clustering in relation with pattern recognition, *Proc. 4th Int. Joint Conf. Pattern Recognition*, Kyoto, Japan, Nov. 7–10, 1978, pp. 19–29.

Singh, A., and Shneier, M. (1990). Grey level corner detection: a generalization and a robust real time implementation, *Comput. Graphics, Image Process.*, vol. 51, no. 1, July, pp. 54–69.

Sinha, R. M. K., and Prasada, B. (1988). Visual text recognition through contextual processing, *Pattern Recognition*, vol. 21, no. 5, pp. 463–480.

Singleton, R. C. (1962). A test for linear separability as applied to self-organizing machines, in *Self-Organizing Systems* (M. C. Yovıts, G. T. Jacobi, and G. D. Goldstein, eds.), Spartan Books, Washington, DC.

Sklansky, J. (1978). On the Hough technique for curve detection, *IEEE Trans. Comput.*, vol. C-27, no. 10, pp. 923–926.

Sklansky, J., and Gonzalez (1980). Fast polygonal approximation of digitized curves, *Pattern Recognition*, vol. 12, pp. 327–331.

Sklansky, J., Cordella, L. P., and Levialdi, S. (1976). Parallel detection of concavities in cellular blobs, *IEEE Trans. Comput.*, vol. C-25, no. 2, pp. 187–196.

Slepian, D. (1967). Restoration of photographs blurred by image motion, *Bell Syst. Tech. J.*, vol. 40, pp. 2353–2362.

Smith, S. P., and Jain, A. K. (1982). Structure of multi-dimensional patterns, *Proc. PRIP '82*, pp. 2–7.

Snyder, H. L. (1973). Image quality and observer performances, in *Perception of Displayed Information* (L. M. Biberman, ed.), Plenum Press, New York, pp. 87–118.

Snyder, L. (1982). Introduction to the configurable highly parallel computer, *IEEE Computer*, Jan., pp. 47–64.

Soklic, M. E. (1982). Adaptive model for decision making, *Pattern Recognition*, vol. 15, no. 6, pp. 485.

Solanki, J. K. (1978). Linear and nonlinear filtering for image enhancement, *Comput. Electron Eng.*, vol. 5, no. 3, pp. 283–288.

Sondhi, M. M. (1972). Image restoration: the removal of spatially invariant degradations, *Proc. IEEE*, vol. 60, pp. 842–853.

Spann, M., and Wilson, R. (1985). A quad-tree approach to image segmentation which combines statistical and spatial information, *Pattern Recognition*, vol. 18, no. 3/4, pp. 257–270.

Specht, D. F. (1967). Generation of polynomial discriminant functions for pattern recognition, *IEEE Trans. Electron. Comput.*, vol. EC-16, no. 3, pp. 308–319.

Spragins, J. (1966). Learning without a teacher, *IEEE Trans. Inf. Theory*, vol. IT-12, no. 2, pp. 223–230.

Srivastava, J. N. (1973). An information function approach to dimensionality analysis and curved manifold clustering, in *Multivariate Analysis*, vol. 3 (P. R. Krishaiah, ed.), Academic Press, New York, pp. 369–382.

Starkov, M. A. (1981). Statistical model of images, *Avtometriya* (USSR), no. 6, pp. 6–12 (in Russian).

Stefanelli, R., and Rosenfeld, A. (1971). Some parallel thinning algorithms for digital pictures, *J. Assoc. Comput. Mach.*, vol. 18, pp. 255–264.

Stern, D., and Kurz, L. (1988). Edge detection in correlated noise using Latin Square masks, *Pattern Recognition*, vol. 21, no. 2, pp. 119–130.

Sternberg, S. R. (1986). Grayscale morphology, *Comput. Graphics, Image Process.*, vol. 35, no. 3, Sept., pp. 333–355.

Stiefeld, B. (1975). An interactive graphics general purpose NDE (nondestructive evaluations) laboratory tool, *Proc. IEEE*, vol. 63, no. 10 (special issue on laboratory automation), pp. 1431–1437.

Stockham, T. G., Jr. (1972). Image processing in the context of a visual model, *Proc. IEEE*, vol. 60, no. 7, pp. 828–842.

Strat, T. M., and Fischler, M. A.(1986). One-eyed stereo: A general approach to modeling 3-D scene geometry, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, Nov., pp. 730–741.

Strobach, P. (1989). Quadtree-structured linear prediction models for image sequence processing, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, July, pp. 742–748.

Su, T. H., and Chang, R. C. (1991a). Computing the constrained relative neighborhood graphs and constrained Gabriel graphs in Euclidean plane, *Pattern Recognition*, vol. 24, no. 3, pp. 221–230.

Su, T. H., and Chang. R. C. (1991b). Computing the k-relative neighborhood graphs in Euclidean plane, *Pattern Recognition*, vol. 24, no. 3, pp. 231–239.

Suetens, P., Haegemans, A., Osterlinck, A., and Gybels, J. (1983). An attempt to reconstruct to cerebral blood vessels from a lateral and a frontal angiogram, *Pattern Recognition*, vol. 16, no. 5, pp. 517–524.

Suk, M. S., and Song, O. (1984). Curvilinear feature extraction using minimum spanning trees, *Comput. Graphics, Image Process.*, vol. 26, no. 3, June, pp. 400–411.

Swain, P. H. (1970). On nonparametric and linguistic approaches to pattern recognition, Ph.D. dissertation, Purdue University, Lafayette, IN.

Sze, T. W. (1979). Lecture note on digital image processing, Shanghai Jiao Tong University, Shanghai, China.

Takatoo, M., Kitamura, T., Okuyama, Y., and Kobayashi, Y. (1989). Traffic flow measuring system using image processing, *SPIE Proc.*, vol. 1197, pp. 172–180.

Takiyama, R. (1981). A committee machine with low committees, *IEEE Comput. Soc. Conf. Pattern Recognition, Image Process.*, Dallas, Aug. 3–5, 1981.

Takiyama, R. (1982). A committee machine with a set of networks composed of two single-threshold elements as committee members, *Pattern Recognition*, vol. 15, no. 5, pp. 405–412.

Tanimoto, S., and Pavlidis, T. (1975). A hierarchical data structure for picture processing, *Comput. Graphics, Image Process.*, vol. 4, pp. 104–119.

Tanimoto, S. (1982). Advances in software engineering and their relations to pattern recognition and image processing, *Pattern Recognition*, vol. 15, no. 3, pp. 113–120.

Tamura, S. (1982). Clustering based on multiple paths, *Pattern Recognition*, vol. 15, no. 6, pp. 477–484.

Taxt, T., Flynn, P. J., and Jain, A. K. (1989). Segmentation of document images, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 12, Dec., pp. 1322–1329.

Taylor, W. E., Jr. (1981). A general purpose image processing architecture for "real-time" and near "real-time" image exploitation, *IEEE SoutheastCon 1981 Congr. Proc.*, Huntsville, AL, Apr. 5–8, 1981, pp. 646–649.

Teh, C. H. and Chin, R. T. (1989). On the detection of dominant points on digital curves, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 8, pp. 859–872.

Tenenbaum, J. M., Barrow, H. G., Bolles, R. C., Fischler, M. A. and Wolf, H. C. (1979). Map-guided interpretation of remotely sensed imagery, *Proc. 1979 IEEE Comput. Sci. Conf. Pattern Recognition, Image Process.*, pp. 610–617.

Thalmann, D., Demers, L.-P., and Thalmann, N. M. (1985). Locating, replacing and deleting patterns in graphics editing of line drawings, *Comput. Graphics, Image Process.*, vol. 29, no. 1, Jan., pp. 37–46.

Thomas, J. C. (1971). Phasor diagrams simplify Fourier transforms, *Electron. Eng.*, pp. 54–57.

Thomas, S. M., and Chan, Y. T. (1989). A simple approach for the estimation of circular arc center and its radius, *Comput. Graphics, Image Process.*, vol. 45, no. 3, March, pp. 362–370.

Thomason, M. G., Barrero, A., and Gonzalez, R. C. (1978). Relational database table representation of scenes, *IEEE SoutheastCon*, Atlanta, pp. 32–37.

Thorpe, C., Hebert, M. H., Kanade, T., and Shafer, S. A. (1988). Vision and navigation for the Carnegie-Mellon Navlab, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 3, pp. 362–373.

Tomita, F., Yachida, M., and Tsuji, S. (1973). Detection of homogeneous regions by structural analysis, *Proc. Int. Joint Conf. Arfit. Intell.*, Stanford, CA, Aug. 1973, pp. 564–571.

Tomoto, Y., and Taguchi, H. (1978). A new type image analyzer and its algorithm, *1978 Int. Congr. Photogr. Sci.*, Rochester, NY, Aug. 1978, pp. 20–26. Also SPSE, 1978, pp. 229–231.

Toney, E. (1983). Image enhancement through clustering specification, Masters Thesis, Pennsylvania State University, University Park, May 1983.

Torre, V., and Poggio, T. (1986). On edge detection, *IEEE Trans, Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 2, March, pp. 147–163.

Tou, J. T. (1968a). Feature extraction in pattern recognition, *Pattern Recognition*, vol. 1, no. 1, pp. 2–11.

Tou, J. T. (1968b). Information theoretical approach to pattern recognition, *IEEE Int. Conv. Rec.*

Tou, J. T. (1969a). Engineering principles of pattern recognition, in *Advances in Information Systems Science* (J. T. Tou, ed.), vol. 1, Plenum Press, New York.

Tou, J. T. (1969b). Feature selection for pattern recognition system, in *Methodologies of Pattern Recognition* (S. Watanabe, ed.), Academic Press, New York.

Tou, J. T. (1969c). On feature encoding in picture processing by computer, *Proc. Allerton Conf Circuits Syst. Theory*, University of Illinois, Urbana.

Tou, J. T. (1972a). Automatic analysis of blood smear micrographs, *Proc. 1972 Comput. Image Process. Recognition Symp.*, University of Missouri, Columbia.

Tou, J. T. (1972b). CPA: a cellar picture analyzer, *IEEE Comput. Soc. Workshop Pattern Recognition*, Hot Springs, VA.

Tou, J. T. (1979). DYNOC—a dynamic optimal cluster-seeking technique, *Int. J. Comput. Inf. Sci.*, vol. 8, no. 6, pp. 541–547.

Tou, J. T., and Gonzalez, R. C. (1971). A new approach to automatic recognition of handwritten characters, *Proc. Two-Dimensional Signal Process. Conf.*, University of Missouri, Columbia.

Tou, J. T., and Gonzalez, R. C. (1972a). Automatic recognition of handwritten characters via feature extraction and multilevel decision, *Int. J. Comput. Inf. Sci.*, vol. 1, no. 1, pp. 43–65.

Tou, J. T., and Gonzalez, R. C. (1972b). Recognition of handwritten characters by topological feature extraction and multilevel categorization, *IEEE Trans. Comput.*, vol. C-21, no. 7, pp. 776–785.

Tou, J. T., and Heydron, R. P. (1967). Some approaches to optimum feature extraction, in *Computer and Information Science*, vol. II (J. T. Tou, ed.), Academic Press, New York.

Triendle, E. E. (1971). An image processing and pattern recognition system for time variant images using TV cameras as a matrix computer, *Artif. Intell. AGARD Conf. Proc.*, London, 1971, Paper 23.

Triendle, E. E. (1979). Landsat image processing, *Advances in Digital Image Processing*, Plenum, New York, 1979, pp. 165–175.

Trier, O. D., Jain, A. K., and Taxt, T. (1996). Feature extraction methods for character recognition—a survey, *Pattern Recognition*, vol. 29, no. 4, pp. 641–661.

Tsao, Y. F., and Fu, K. S. (1981). Parallel thinning operations for digital binary images, *IEEE Comput. Soc. Conf. Pattern Recognition, Image Process.*, Dallas, Aug. 3–5, 1981, pp. 150–155.

Tsypkin, Ya. Z. (1965). Establishing characteristics of a function transformer from randomly observed points, *Autom. Remote Control*, vol. 26, no. 11, pp. 1878–1882.

Tuceryan, M., and Jain, A. K. (1990). Texture segmentation using Voronoi polygons, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 2, Feb., pp. 211–216.

Turk, M., Morgenthaler, D., Gremban, K., and Marra, M. (1988). VITS—a vision system for autonomous land vehicle navigation, *IEEE Trans. on PAMI*, vol. 10, no. 3, pp. 342–361.

Turney, J. L., Mudge, T. N., and Volz, R. A. (1985). Recognizing partially occluded parts, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, no. 4, July, pp. 410–421.

Twogood, R. E., and Ekstrom, M. P. (1976). An extension of Eklundh' matrix transposition algorithm and its applications in digital image processing, *IEEE Trans. Comput.*, vol. C-25, no. 9, pp. 950–952.

Uchiyama, T., and Arbib, M. A. (1994). An algorithm for competitive learning in clustering problems, *Pattern Recognition*, vol. 27, no. 10, pp. 1415–1421.

Uhr, L. (1971a). Flexible linguistic pattern recognition, *Pattern Recognition*, vol. 3, no. 4, pp. 363–383.

Uhr, L. (1971b). Layered recognition cone networks that preprocess classify and describe, *Proc. Two-Dimensional Signal Process. Conf.*, Columbia, MI, pp. 311–312.

Umesh, R. M. (1988). A technique for cluster formation, *Pattern Recognition*, vol. 21, no. 4, pp. 393–400.

Unser, M. (1986). Sum and difference histograms for texture classification, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 1, Jan., pp. 118–125.

Unser, M. (1995). Texture classification and segmentation using wavelet frames, *IEEE Trans. Image Process.*, vol. 4, no. 11, pp. 1549–1560.

Unser, M., and Eden, M. (1989). Multiresolution feature extraction and selection for texture segmentation, *IEEE Trans. Pattern Analysis Mach. Intell.*, vol. 11, no. 7, July, pp. 717–728.

Urquhart, R. (1982). Graph theoretical clustering based on limited neighborhood sets, *Pattern Recognition*, vol. 15, no. 3, pp. 173–187.

VanderBrug, G. J., and Nagel, R. N. (1979). Vision systems for manufacturing, *Proc. 1979 Joint Autom. Control Conf.*, Denver, June 17–21, 1979, pp. 760–770.

VanderBrug, G. J., and Rosenfeld, A. (1977). Two-stage template matching, *IEEE Trans. Comput.*, vol. 26, no. 4, pp. 384–394.

VanderBrug, G. J., and Rosenfeld, A. (1978). Linear feature mapping, *IEEE Trans. Syst. Man Cybern.*, SMC-8, no. 10, pp. 768–774.

Vetterli, M., and Herley, C. (1992). Wavelets and filter banks: theory and design, *IEEE Trans. Signal Process.*, vol. 40, no. 9, pp. 2207–2232.

Vickers, A. L., and Modestino, J. W. (1981). A maximum likelihood approach to texture classification, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no. 1, pp. 61–68.

Vilione, S. S. (1970). Applications of pattern recognition technology, in *Adaptive Learning and Pattern Recognition Systems: Theory and Applications* (J. M. Mendal and K. S. Fu, eds.), Academic Press, New York, pp. 115–162.

Vilnrotter, F. M., Nevatia, R., and Price, K. E. (1986). Structural analysis of natural textures, *IEEE Trans. Pattern Anal Mach. Intell.*, vol. PAMI-8, no. 1, Jan., pp. 76–89.

Vinea, A., and Vinea, V. (1971). A distance criterion for figural pattern recognition, *IEEE Trans. Comput.*, vol. C-20, June 1971, pp. 680–685.

Viscolani, B. (1982a). Optimization of computational time in pattern recognizers, *Pattern Recognition*, vol. 15, no. 5, pp. 419–424.

Viscolani, B. (1982b). Computational length in pattern recognizers, *Pattern Recognition*, vol. 15, no. 5, pp. 413–418.

Wald, L. H., Chou, F. M., and Hines, D. C. (1980). Recent progress in extraction of targets out of cluster, *Proc. Soc. Photo-opt. Instrum. Eng.*, vol. 253, pp. 40–55.

Walter, C. M. (1968). Interactive systems applied to the reduction and interpretation of sensor data, *Proc. Digital Equipment Comput, Users Fall Symp.*, Dec. 1968.

Walter, C. M. (1969). Comments on interactive systems applied to the reduction and interpretation of sensor data, *IEEE Comput. Commun. Conf. Res.*, 1969 (IEEE Spec. Publ. 69, 67-MVSEC), pp. 109–112.

Walton, D. J. (1989). A note on graphics editing of curved drawings, *Comput. Graphics, Image Process.*, vol. 45, no. 1, Jan., pp. 61–67.

Wandell, B. A. (1987). The synthesis and analysis of color images, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 1, Jan., pp. 2–13.

Wang, C. Y., and Wang, P. P. (1982). Pattern analysis and recognition based upon the theory of fuzzy subsets, *Conf. Proc. IEEE SoutheastCon '82*, Destin, FL., Apr. 4–7, 1982, pp. 353–355.

Wang, S., Rosenfeld, A., and Wu, A. Y. (1982). A medial axis transformation for grey scale pictures, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no, 4, pp. 419–421.

Warmack, R. E., and Gonzalez, R. C. (1972). Maximum error pattern recognition in supervised learning environments, *IEEE Conv. Rec.*—Region III.

Warmack, R. E., and Gonzalez, R. C. (1973). An algorithm for the optimal solution of linear inequalities and its application to pattern recognition, *IEEE Trans. Comput.*, vol. C-22, pp. 1065–1075.

Watanabe, S. (1965). Karhunen-Loeve expansion and factor analysis theoretical remarks and applications, *Proc. 4th Conf. Inf. Theory*, Prague.

Watanabe, S. (1970). Feature compression, in *Advances in Information Systems Science* (J. T. Tou, ed.), vol. 3, Plenum Press, New York.

Watanabe, W. (1971). Ungrammatical grammar in pattern recognition, *Pattern Recognition*, vol. 3, no. 4, pp. 385–408.

Watson, D. F., and Philip, G. M. (1984). Systematic triangulations, *Comput. Graphics, Image Process.*, vol. 26, no. 2, May, pp. 217–223.

Webb, A. R., and Lowe, D. (1990). The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis, *Neural Networks*, vol. 3, no. 14, pp. 367–375.

Wechsler, H. (1979). Feature extraction for texture discrimination, *Proc. 1979 IEEE Comput. Soc. Conf. Pattern Recognition, Image Process.*, Chicago, pp. 399–403.

Wei, M. and Mendel, J. M. (1994). Optimality tests for the fuzzy c-means algorithm, *Pattern Recognition*, vol. 27, no. 11, 1567–1573.

Weldon, T., Higgins, W., and Dunn, D. (1996). Efficient Gabor filter design for texture segmentation, *Pattern Recognition*, vol. 29, no. 2, pp. 2005–2025.

Werman, M., and Peleg, S. (1985), Min-Max operators in texture analysis, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, no. 6, Nov., pp. 730–733.

Wermser, D., Haussemann, G., and Liedtke, C. E. (1984). Segmentation of Blood Smears by Hierarchical Thresholding, *Comput. Graphics, Image Process.*, vol. 15, no. 2, Feb., pp. 151–168.

Weszka, J. S., and Rosenfeld, A. (1979). Histogram modification for threshold selection, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-9, no. 1, pp. 38–52.

Whitmore, P. G., Rankin, W. C., Baldwin, R. D., and Garcia, A. (1972). Studies of aircraft recognition training, *Tech. Rep.*, Human Research Organization, Alexandria, VA, AD-739923.

Whitney, A. W., and Blasdell, W. E. (1971). Signal analysis and classification by interactive computer graphics, in *AGARD, Artificial Intelligence*, General Electric Co., Syracuse, NY.

Widrow, B. (1962). Generalization and information storage in networks in Adaline neurons, in *Self-Organizing Systems* (M. C. Yovits, G. T. Jacobi, and D. Goldstein, eds.), Spartan Books, Washington, DC.

Widrow, B. (1973a). The rubber mask technique: I. Pattern measure and analysis, *Pattern Recognition*, vol. 5, no. 3, pp. 175–197.

Widrow, B. (1973b). The rubber mask technique: II. Pattern storage and recognition, *Pattern Recognition*, vol. 5, no. 3, pp. 199–211.

Widrow, B., and Lehr, M. A. (1990). 30 years of adaptive neural networks: perceptron, madeline, and backpropagation, *Proc. IEEE*, vol. 78, no. 9, pp. 1415–1442.

Will, P. M., and Koppleman, G. M. (1971). MFIPS: A multi-functional digital image processing system, *IBM Res.*, *RC3313*, Yorktown Heights, NY.

Will, P. M., and Pennington, K. S. (1972). Grid coding: a novel technique for image processing, *Proc. IEEE*, vol. 60, no. 6, pp. 669–680.

Wilson, R., and Spann, M. (1990). A new approach to clustering, *Pattern Recognition*, vol. 23, no. 12, pp. 1413–1425.

Winder, R. O. (1963). Bounds on threshold gate realizability, *IEEE Electron. Comput.*, vol. EC-12, no. 4, pp. 561–564.

Winder, R. O. (1968). Fundamentals of threshold logic, in *Applied Automata Theory* (J. T. Tou, ed.), Academic Press, New York.

Wolfe, J. H. (1970). Pattern clustering by multivariate mixture analysis, *Multivariate Behav. Res.*, vol. 5, p. 329.

Wolferts, K. (1974). Special problems in interacting image processing for traffic analysis, *Proc. 2nd Int. Joint Conf. Pattern Recognition*, Copenhagen, Aug. 13–15, 1974, pp. 1–2.

Wolfson, H. J. (1990). On curve matching, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 5, May, pp. 483–489.

Wong, M.. A., and Lane, T. (1982). A kth nearest neighbor clustering procedure, *Computer Science and Statistics: Proc. 13th Symp. Interface*, Pittsburgh, PA, Mar. 12–13, 1981, pp. 308–311.

Wong, R. Y. (1977). Image sensor transformations, *IEEE Trans. Syst. Man Cybern.*, vol. SMC-7, no. 12, pp. 836–841.

Wong, R. Y. (1982). Pattern recognition with multi-microprocessors, *Proc. IEEE 1982 Region 6 Conf.*, Anaheim, CA, Feb. 16–19, 1982, pp. 125–129.

Wong, R. Y., and Hall, E. L. (1977). Sequential hierarchical scene matching, *IEEE Trans. Comput.*, vol. C-27, no. 4, pp. 359–365.

Wong, R. Y., Lee, M. L., and Hardaker, P. R. (1980). Airborne video image enhancement, *Proc. Soc. Photo-opt. Instrum. Eng.*, vol. 241, pp. 47–50.

Wood, R. E., and Gonzalez, R. C. (1981). Real time digital enhancement, *Proc. IEEE*, vol. 69, no. 5, pp. 643–654.

Wu, C. L. (1980). Considerations on real time processing of space-borne aperture radar data, *Proc. Soc. Photo-opt.*, *Instrum. Eng.*, vol. 241, pp. 11–19.

Wu, J. C., and Bow, S. T. (1998). Tree structure of wavelet coefficients for image coding, *IEEE ICIPS '98 Proc. 2nd IEEE Int. Conf. Intelligent Processing Systems*, pp. 272–376.

Wu, S. Y., Subitzki, T., and Rosenfeld, A. (1981). Parallel computation of contour properties, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, no. 3, pp. 331–337.

Wuncsh, P., and Laine, A. (1995). Wavelet descriptor for multıresolution recognition of handwritten characters, *Pattern Recognition*, vol. 28, no. 8, pp. 1237–1249.

Xie, Q., Laslo, A., and Ward, R. K. (1993). vector quantization technique for nonparametric classifier design, *IEEE Trans, Pattern Anal. Mach. Intell.*, vol. 15, no. 12, pp. 1326–1330.

Yalamanchilli, S., and Aggarwal, J. K. (1985). Reconfiguration strategies for parallel architectures, *IEEE Computer*, December, pp. 44–61.

Yang, M. C. K., and Yang, C. C. (1989). Image enhancement for segmentation by self-induced autoregressive filtering, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, June, pp. 655–661.

Yang, M. C. K., Kim, C. K., Cheng, K. Y., Yang, C. C., and Liu, S. S. (1986). Automatic curve fitting with quadratic B-spline functions and its applications to computer-assisted animation, *Comput. Graphics*, *Image Process.*, vol. 33, no. 3, March, 346–365.

Yingwei, L., and Sundararajan, V. (1998). Performance evaluation of a sequential minimal RBF neural network learning algorithm, *IEEE Trans. Neural Networks*, vol. 9, no. 2, pp. 308–318.

Yu, B., and Yuan, B. (1993). A more efficient branch and bound algorithm for feature selection, *Pattern Recognition*, vol. 26, no. 6, pp. 883–889.

Yoda, H., Ohuchi, Y., Taniguchi, Y., and Ejiri (1988). An automatic wafer inspection system using pipelined image processing techniques, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 1, pp. 4–16.

Yokoyama, R., and Haralick, R. M. (1978). A texture pattern synthesis method by growth method, *Tech. Rep.*, Iwata University, Morioka, Japan.

Young, I. T. (1978). Further consideration of sample and feature size, *IEEE Trans. Inf. Theory*, vol. IT-24, no. 6, pp. 773–775.

Zahn, C. T. (1971). Graph theoretical methods for detecting and describing gestalt clusters, *IEEE Trans. Comput.*, vol. C-20, no. 1, pp. 68–86.

Zbigniew, W., and Wojcik, (1984). An approach to the recognition of contours and line-shaped objects, *Comput. Graphics*, *Image Process.*, vol. 25, no. 2, Feb., pp. 184–204.

Zhang, Q., Wang, Q., and Boyle, R. (1991). A clustering algorithm for data-sets with a large number of classes, *Pattern Recognition*, vol. 24, no. 4, pp. 331–340.

Zhang, T. Y., and Suen, C. Y. (1984). A fast parallel algorithm for thinning digital patterns, *Comm. ACM*, vol. 27, no. 3, pp. 236–239.

Zhou, Y. T., Venkateswar, V., and Chellappa, R. (1989). Edge detection and linear feature extraction using 2-D random field model, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 1, pp. 84–95.

Zhuang, X. H., and Haralick, R. M. (1986). Morphological structuring element decomposition, *Comput. Graphics*, *Image Process.*, vol. 35, no. 3, Sept., pp. 370–382.

This Page Intentionally Left Blank

Index

Absolute correction rule, 67, 69–71
ADALINE, 46, 201
Adaptive decision procedure, 17
Adjoint, 99, 616
Agglomerative approach, 114
Aperture, 424–431
Aperture function, 423–424
A posteriori probability, 83
Approximations, 492, 499–500
  successive approximation, 494
A priori probability, 83–85
Area of high activity, 302
Area of low activity, 302
Artificial neural networks (ANN), 197–199
Augmented feature space, 38
Augmented pattern vector, 35, 39
Augmented weight vector, 35
Autocorrelation, 416
Average cost, 83
Average risk, 83

Bachelor and Wilkins, algorithm, 119–121
Back-propagation algorithm, 211–219
  flowchart for, 220
Basic events, 188
  basic event generation algorithm, 188–190
Bayes' classifier, 86
Bayes decision boundary, 98, 106
Bayes decision surface, 106, 109
Bayes discriminant function, 85–86
Bayes' law (rule), 85–86, 91
Bit reversal, 448–449
Blurring, 238–239, 302
Butterworth filter, 469–473

Canonical analysis, 172
Capacity of a ϕ machine, 54–57
Carpenter-Grossberg classifier, 198–199
Cell, 12
  acidophilic, 12
  cosinophilic, 12
Chain cluster, 157
Chain code, 374–375
Character recognition, 24
Classification, 18

Classification space, 8
Classifier, 20, 23
Closing operation, 342–343
Cluster, 32
Cluster center, 117–118, 132–138
Cluster domain, 129–139, 145
Clustering analysis, 112
Clustering method, evaluation of the cluster methods, 145
Clustering transformation, 170
Cofactor, 615
Collinear string, 516
Committees machmes, 46
  training procedure for committee machines, 74
Compass gradient mask, 307
Concept of saliency of a boundary segment, 394
Conceptual recognition, 5
Conditional average loss, 83
Conditional average risk, 83, 85
Connected component generation, 514
Connected graph, 149
Connectionist model, 197
Connectivity, 343
Contour map, 24
Contour tracing, 344–345
Convex decision surface, 43
Convexity, 43
Convolution, 306, 336, 401, 416–419, 587–589
Convolution integral, 587–589
Convolution theorem, 419
Cooccurrence matrix, 352–354
Correlation, 416–419
Correlation method, 24, 416–419
Cost function, 31
Covariance, 33, 180
Covariance matrix, 97
  within-class covariance matrix, 184
  between-class covariance matrix, 184
Criterion function:
  perceptron criterion function, 73
  relaxation criterion function, 74
Cross-correlation, 416
Cumulative density function, 286–288
Cumulative distribution, 286–288
Cumulative histogram, 286–288
Curve fitting, 374
  B-spline, 374, 377–378
  concatenated arc approximation, 374, 385–392
  piecewise polynomial, 374
  polygonal approximation, 374–376
  polynomial, 374–376

Data:
  acquisitions, 9
  preprocessing, 10, 18
  reduction, 10
Decimation, 492
Decision boundary, 91
Decision classification, 20
Decision classifier, 19
Decision function, 19, 57
Decision processor, 17, 19
  training of decision processor, 17, 19
Decision surface, 33–36, 39–40, 51, 61
Decision theoretical approach, 35, 37
Degree of similarity, 143, 146
Delaunay method, 366, 372
Details, 492
Deterministic function, 34
Deterministic gray-level transformation, 271
Diagonal matrix, 173, 176, 467
Dichotomies, 52, 56
  probability of dichotomies, 55
Dichotomization, 54
  capacity of dichotomization, 57
Differential operators, 590–591
Digital image processing, 268
Digital implementation of DWT, 496
Dilation, 336–339
Dimensionality reduction, 172
Dirac delta function, 405, 584–586
Discrete Fourier transform, 401, 403, 422, 441–447
Discrete Karhunen-Loeve transform, 466–468
Discriminant functions, 33–39, 42, 48–49, 52, 57, 88, 96

[Discriminant functions]
  nonparametric training of discriminant function, 62
Dispersion, 140
Dissimilarity measure, 113
Distance function, 20
Distance measure, 118–119, 138–144
Distance metrics, 20
Divisive approach, 114
Document image analysis, 513–523
Downsampling, 492–493, 495
Dynamic cluster method, 142–144
Dynamic optimal cluster seeking technique (DYNOC), 141–142

Ecology, 31
Edge, 149, 158
Edge sharpening, 303–330
Edge weight plots, 152–153
Eigenvalues, 51–52, 94, 162, 175–176, 468, 621–623
Eigenvectors, 52, 467
Energy spectrum, 407
Erosion, 339–341
Error-correcting training methods, 66, 76
Euclidean distance, 113
  classifier, 38
  of unknown pattern from class center, 38
Euclidean space, 20, 62
Expected value, 92
Exponential filter, 469–471

Fast Fourier transform, 401, 437, 446–453
Fast Walsh transform, 459
Feature extraction, 8, 10, 14, 18
Feature ordering, 11, 170
Features, 10–11, 30
  local feature, 11
  global feature, 11
  derived feature, 15
Feature selection, 168, 188
Feature space, 8, 18–20, 168
Feature subset, 188–189
Feature vector, 11, 14, 17, 20, 32
Feature weighting coefficients, 170–171
Filter, 469–471
  highpass, 469–471
  lowpass, 469–471
Filter banks, 490, 492–493
Finite differences, 593–595
Fisher's criterion function, 185
Fisher's determinant, 182, 187
Fixed increment rule, 67, 71
FLOPS, 563
Forward transform, 405–496
Forward transform kernel, 405–410
Fourier coefficient, 483
Fourier descriptor, 380–384
Fourier spectrum, 407–417, 424–440, 482
Fourier transform, 404–454
  Fourier transform pair, 404–419
  windowed Fourier transform, 482
Fractional correction rule, 68–71
Frame grabbing time, 596
Frequency domain, 420
Frequency spectrum, 407–417, 424–440
Functional approximation, 101
Fuzzy decision, 199
Fuzzy membership function, 113

Gabriel graph, 157–161
Gaussian distribution, 82
Generalized decision function, 55–56
Generalized inverse, 77
Generalized likelihood ratio, 87
Generalized threshold value, 87
Gradient, 304
Gradient descent technique, 72
Gradient technique, 72
Grammatical inference, 22
Graphics description, 513–517
Graphics understanding, 517
Graph theoretical method, 146
Gray level distribution, 103
Gray level transformation, 271–295
Gray level transformation functions, 272
Gray level transformation matrix, 296
Gray-tone run length, 352
Ground truth, 103

Hadamard matrix, 460
Hadamard transform, 459–466, 481
Hadamard transformation pair, 462
Hamming distance, 237
Hamming net, 198–199, 236–240
  example, 241–246
Hamming net classifier, algorithm for, 238–246
Hard limiter, 201
Hermite polynomials, 101
Heuristic approach, 114, 117
Hidden layer, 206–207
Hierarchical clustering algorithm, 121–128
High-pass filtering, 471–473
  Butterworth, 471–473
  exponential, 471–473
  ideal, 471–473
  trapezoidal, 471–473
Histogram, 103–106
Histogram equalization, 288–290
Histogram modification, 278, 287
Histogram specification, 291, 295
Histogram thinning, 295–298
Ho-Kashyap algorithm, 77–78
Homogeneous coordinates, 600–601
Hopfield net, 198–199, 256–261
  architecture of the Hopfield net, 257
  operation of Hopfield net, 261–265
Hough transform, 346–348
Hyperplane, 34, 36, 63–64, 69

Identification of:
  industrial parts with x-ray, 542–545
  partially obscured objects, 392–399
  scratches and cracks on polished crystal blanks, 534–542
  scratches and cracks on unpolished crystal blanks, 530–535
Identity matrix, 51, 175, 460
Ideographs, 22
Image:
  acquisition, 8
  cytological, 12
  data preprocessing, 8
  display, 8
[Image]
  enhancement, 271, 473–476
  function, 271, 303, 412, 441, 453, 581–582
  information content, 440
  inverse transform, 401
  model, 402–403, 579
  plane, 600–601, 607
  processing, 402
  segmentation, 8
  spectrum, 401
  transform, 401, 403
Impulse, 421
Impulse sheet, 422
Indeterminate region, 45–46
Inertia function, 113
Interconnection networks, 566–571
  SIMD, 566
  MIMD, 566
Interpolation, 494
Intersample distance, 146
Intersection, 112
Interset distance, 32, 115
Intraset distance, 32, 115–116
Inverse fast Fourier transform, 401
Inverse Fourier transform, 411, 422
Inverse Hadamard transform, 462
Inverse transform, 407–408, 419, 441
Inverse transform kernel, 405, 441
Inverse Walsh transform, 455
ISODATA algorithm, 131–139,
  modification of ISODATA, 139–141

Karhunen-Loeve method, 24
Kernel:
  forward transform, 441–442
  Fourier transform, 405
  Hadamard transformation, 459–466
  inverse transform, 441
  separability, 409–410
  Walsh inverse transformation, 455
  Walsh transformation, 454–459
Kirsch edge detector, 312, 325
K-means algorithm, 129–131
k-nearest neighbor decision rule, 121–128

[*k*-nearest neighbor decision rule]
  definition of $\Omega_k(\mathbf{x}_i)$, 123–124
  definition of $P_k(\mathbf{x}_i)$, 123–124
  definition of $\xi_k(\mathbf{x}_i)$, 123–124
  definition of SIM_1(m, n), 125–127
  definition of SIM_2(m, n), 125–127
  definition of SIM(m, n), 125–127
  definition of $P_k^{sc}(m)$, 125
  definition of $P_k^{sc}(n)$, 125
  definition of $Y_k^{m,n}$ and $Y_k^{n,m}$, 126
  definition of $B_k^{m,n}$ and $B_k^{n,m}$, 127
Kohonen self-organizing feature maps, 198–199, 246–151
Kohonen SOFM algorithm, 251–252
Kronecker delta functions, 84

Lagrange multipliers, 170–171
Laplacian masks, 311
Laplacian operator, 306–311
Layered machines, 46–48
Likelihood function, 86, 161, 349
Limited neighborhood concept, 155
Limited neighborhood sets, 155
Linear dichotomies, 54–57
Linear discriminant functions, 34–38, 96–97
Linear separable classes, 43
Logarithmic Laplacian edge detector, 325
Logic circuit diagram reader, 523, 526–528
Loop, 149
Loss function, 83
Loss matrix, 84
Low-pass filtering, 468–471
  Butterworth, 468–471
  exponential, 468–471
  ideal, 468–471
  trapezoidal, 468–471

MADALINE, 46, 201
Mahalanobis distance, 92, 94, 96, 114, 169
Main diameter, 150, 167
Match count, 155
Matrices, 613
  matrix multiplication, 614–615
[Matrices]
  partitioning of matrices, 615–616
  inverse matrix, 617–619
Maximal spanning tree, 152
Maximum distance algorithm, 119–121, 141–142
Maximum likelihood decision, 86
Maximum likelihood rule, 87
Maxnet, 237–238
Mean vector, 94, 168
Measurement space, 168
Measure of similarity, 19, 94
Medial-axis transformation, 334, 378–380
Membership boundary, 118
Merging function, 188
Mexican hat function, 249
MFLOPS, 563
Minimal spanning tree, 149, 151, 366
Minimal spanning tree method, 149, 166
Minimization of sum of squared distance (K-mans algorithm), 129–131, 138, 164
Minimum distance classifiers, 40, 98
Minimum squared error procedure, 76
Minimum squared error solution, 76
Minkowski addition, 336
Minkowski subtraction, 339
Mixture features, 188
Mixture statistics, 161–164
Modes, 161
Morphological processing, 336–343
Multicenters, 142–144
Multilayer perceptron, 198, 201, 205–206
Multiresolution, 482
  pyramid decomposition, 497–498
  reconstruction structure, 502
Multiprototypes, 40
Multispectral data, 13
Multispectral scanner (MSS), 7, 15–16, 18
Multivariate Gaussian data, 168
Multivariate Gaussian distribution, 33, 168
Multivariate normal density, 95, 98

Natural association, 19, 32
Nearest neighbor classification, 122
Nearest-neighbor rule, 42
Network topologies, 567
  chordal ring, 567
  linear array, 567
  near-neighbor mesh, 567
  ring, 567
  star, 567
  systolic array, 567
  tree, 567
  3-cube, 567
  3-cube-connected cycle, 567
Negative loss function, 84
Neighborhood average algorithm, 300
Neighborhood processing, 272, 401
Neural network models, 5
Neuron, 197
Nonhierarchical clustering, 142
Nonlinear discriminant function, 34, 49–52
Nonparametric decision theoretic classification, 33
Nonparametric feature selection, application to mixed feature, 188
Nonparametric pattern recognition, 33
Nonparametric training of discriminant function, 62
Nonsupervised learning, 112
Normal density function, 94
Normal distributed patterns, 93
Normal distribution, 93
Numerical integration, 591–593
Numerical taxonomy, 31

Opening operation, 342–343
Optical spectrum, 13
Optimal acquisition of ground information, 545–551
Optimum discriminant function, 82, 89
Orthogonal matrix, 405
Orthonormal functions, 58, 102
Orthonormality, 405
Orthonormal transformation, 162
Orthonormal vector, 468

Panning, 601
Parametric pattern recognition approach, 33
Partition, 31
Path, 149
Pattern, 3–4, 16
Pattern class, 3
Pattern mappings, 205
Pattern recognition, 3–4, 30, 42
  pattern recognition system, 5–9
  pattern recognition technique, 4, 6–7
  supervised pattern recognition, 5, 32
  three phases in pattern recognition, 8
  unsupervised pattern recognition, 5, 32
Pattern space, 8, 16, 32, 34, 62
Pattern vector, 16
Perceptron criterion function, 73
Perceptrons, 73, 201
Perceptron training algorithm, 73
ϕ functions, 52
ϕ machines, 52–54
ϕ space, 52
Phoneme, 25
Phoneme recognition, 25
Pictophonemes, 22
Piecewise linear discriminant functions, 34, 42
Point processing, 272–273
Point spread function, 586
Potential functions, 57
Pragmatics, 112
Primitives, 21–24
Principal component analysis, 172
Principal component axis, 177
  principal component axis classifier design, 177
  procedure for finding the principal component axis, 178
Probability density functions, 57, 98, 101
Probability distributions, 280–286
Probability of error, 90–93
Prototype average (or class center), 38
Prototypes, 19, 29, 34, 40–42, 44, 58, 63, 70
Pseudoinverse method (technique), 76–77

Quadratic decision surface, 51
Quadratic discriminant function, 52, 96–97
Quadratic processor, 52
Quadtree, 363–370
Quantization, 597–599
  tapered, 599
  uniform, 598

Radial basis function networks (RBF), 225–231
  comparison of RBF with MLP, 234
  formulation of RBF by means of statistical decision theory, 232–234
  RBF training, 231–232
Region of influence, 156–158
Relative neighborhood graph, 156–158
Relaxation algorithm, 74
Relaxation criterion function, 74
Risk function, 83
Robert's cross operator, 304
Robotic vehicle road following, 552–559

Sampling, 420, 429, 431, 597–599
Sampling device, 437
  annular-ring, 437
  wedge-shaped, 437
Scaling, 601
Scaling function, 486, 489
  Haar scaling function, 489, 491
  two-dimensional scaling function, 499
  scaling function coefficients, 487
Scatterogram, 556
Semantics, 112
Separating surfaces, 34–36
Sequential learning approach, 348–351
Sgn (signum function), 260
Shared near neighbor maximal spanning tree, 152–155
Share near neighbor rule, 152
Short time Fourier transform (STFT), 482–484
Sigmoid logistic nonlinearity, 49, 205
Similarity function, 140
Similarity matrix, 146–149
Similarity measure, 113–114
Sinc function, 408
Smoothing, 274–303
Sobel operator, 311–312
Solution region, 65
Solution weight vector, 67, 80
Space:
  classification space, 8
  feature space, 8
  pattern space, 8
Spanning tree, 149
Spanning tree method, 149
  graph theoretic clustering based on limited neighborhood sets, 155–161
  maximal spanning tree for clustering, 152–155
  minimal spanning tree method, 149–152
Spatial domain, 401–402, 422, 429
Spatial processing, 271
Spatial resolution, 598
Spectral band, 13
Spectral characteristics, 11
Spectral distribution:
  of the scaling function, 503, 505
  of wavelets, 503, 505
Spectral range, 104–106
Spectral response, 15
Spectrum, 407–410, 413–417, 426–440
Speech recognition, 24
Standard deviations, 102, 131, 137, 139, 172, 199, 329
State conditional probability density function, 83
Statistical decision method, 83
Statistical decision theory, 83
Statistical differencing, 305
Statistical discriminant functions, 82
  training for statistical discriminant, 101
Steepest descent, 72
Structural pattern recognition, 21
Subpatterns, 20
Successive doubling, method of, 442–451
Sum of squared error criterion function, 77

Supervised learning (supervised recognition). 29–30, 33, 112
Symmetrical loss function, 83–84, 87, 91
Synaptic weights, 29
Syntactic pattern recognition, 20–23
Syntax. 112
Syntax rule, 20

Tanimoto coefficient, 114
Template matching, 419
Text strings, 513
Texture features, 27
Texture and textual images, 352–354
Thinning, 333–336
Threshold logic unit (TLU), 46
Traffic flow measuring, 551–552
Training decision processor, 17
Training set, 83
Transformation matrix, 173, 601
Transform domain, 271–272
Transform processing, 401–476
Translation, 412–416, 603
Tree, 149
Two-dimensional Founer transform, 406–409
Typology, 31

Union operator, 112
Unitary matrix, 405
Unsupervised learning, 29–32
Upsampling, 494–495

Variables:
  spatial, 580
[Variables]
  spectral, 580
  temporal, 580
Variance, 40, 92. 116
Vector gradient, 303
Vector space:
  of signals S, 486
  spanned by the scaling function, 486, 488
  spanned by the wavelet function, 487–488

Wallis operator, 325–326
Walsh transformation, 454–459
Walsh transform pair, 457
Wavelet, 481–501
  analysis, 483
  coefficient. 484–485, 494–495, 497
  functions, 485
Wavelet transform, 481–497
  continuous, 484–485
  discrete, 484–485, 494–499
  inverse discrete, 484, 494
  two-dimensional, 499
Weather forecasting, 23
Weight, of a tree, 149
Weight adjustments, 207
  in backward direction, 207
Weight array, 300
Weight coefficient, 114
Weight Euclidean distance, 113–114
Weight matrix, 98
Weight space, 62–63, 69
Weight vector, 62, 70
Widrow-Hoff rule, 79